Universitat Politècnica de València
Departamento de Sistemas Informáticos y Computación
Doctorado en Informática

Ph.D. Thesis

Model Integration in Data Mining: From Local to Global Decisions

Candidate:
Antonio Bella Sanjuán

Supervisors:
Cèsar Ferri Ramírez,
José Hernández Orallo and
María José Ramírez Quintana.

– June 2012 –

This work has been partially supported by the EU (FEDER) and the Spanish MICINN, under grants TIN2010-21062-C02-02, TIN2007-68093-C02-02; the Generalitat Valenciana under grant GV06/301 and project PROMETEO/2008/051; the UPV under grant TAMAT; and the Spanish project "Agreement Technologies" (Consolider Ingenio CSD2007-00022).

Author's address:

Departamento de Sistemas Informáticos y Computación
Universitat Politècnica de València
Camino de Vera, s/n
46022 Valencia
España

A mis padres.
A mis hermanas.
A María.

Acknowledgments

Echo la vista atrás y me acuerdo de aquel primer día en que empecé mi carrera en la UPV. Recuerdo que pensé: "al menos durante los 5 próximos años tendré que hacer este mismo recorrido todos los días". Y así fue, pero lo que no pensaba es que después de la ingeniería vendría el doctorado, así que he seguido haciendo ese mismo camino, pero en coche en vez de en tranvía.

Esta tesis, que tiene entre sus manos, en la pantalla de su ordenador o dispositivo móvil, es la culminación de todos estos años de investigación con un grupo de personas con las que muchas veces he compartido más tiempo que con mi familia y a los que les doy las gracias. Con ellos he compartido esos momentos de frustración en los que no sale nada y en los que lo más fácil sería abandonar, pero siempre me han apoyado para seguir adelante. Y, por supuesto, también hemos compartido los buenos momentos de alegría y satisfacción, que al final son los que más se recuerdan. Todo esto para decirles que si no fuese por ellos no estarían leyendo lo que están leyendo.

En particular, gracias a Salvador Lucas que empezó toda esta aventura enviándome un correo para una beca. A María Alpuente, líder del grupo ELP, que me dio la oportunidad de pertenecer a este grupo. A mis directores de tesis: José, María José y César, por toda la ayuda que me han brindado y por su paciencia y dedicación en las revisiones de los trabajos que hemos realizado juntos. A Bea por todo el tiempo que hemos compartido juntos, por todo lo que hemos llorado y sobre todo por todo lo que hemos reído. A Vicent por todos sus buenos consejos. A Josep por su "introducción a la investigación". A Raquel por nuestras charlas. A Tama, Sonia, Rafa, Michele y Vesna porque las comidas de esa época llegaron a ser el momento más divertido del día. Y a todos los compañeros que han pasado por el grupo ELP durante estos años y que no enumero porque prefiero no olvidarme de ninguno.

Y en especial, muchas gracias a mis padres, a mis hermanas y a mi novia María que saben todo lo que ha costado llegar hasta este momento y que a partir de hoy van a estar un poco más orgullosos de su hijo/hermano/novio.

No se que me deparará el futuro, eso sólo el tiempo lo dirá, pero de lo que estoy seguro es que estaría encantado de seguir recorriendo cada día este mismo camino.

Junio 2012,

Antonio Bella.

Abstract

Machine Learning is a research area that provides algorithms and techniques that are capable of learning automatically from past experience. These techniques are essential in the area of Knowledge Discovery from Databases (KDD), whose central stage is typically referred to as Data Mining. The KDD process can be seen as the learning of a model from previous data (model generation) and the application of this model to new data (model deployment). Model deployment is very important, because people and, very especially, organisations make decisions depending on the results of the models.

Usually, each model is learned independently from the others, trying to obtain the best (local) result. However, when several models have to be used together, some of them can depend on each other (e.g., outputs of a model are inputs of other models) and constraints appear on their application. In this scenario, the best local decision for each individual problem could not give the best global result, or the result could be invalid if it does not fulfill the problem constraints.

Customer Relationship Management (CRM) is an area that has originated real application problems where data mining and (global) optimisation need to be combined. For example, prescription problems deal about distinguishing or ranking the products to be offered to each customer (or simetrically, selecting the customers to whom we should make an offer). These areas (KDD, CRM) are lacking tools for a more holistic view of the problems and a better model integration according to their interdependencies and the global and local constraints. The classical application of data mining to prescription problems has usually considered a rather monolitic and static view of the process, where we have one or more products to be offered to a pool of customers, and we need to determine a sequence of offers (product, customer) to maximise profit. We consider that it is possible to perform a better customisation by tuning or adapting several features of the product to get more earnings. Therefore, we present a taxonomy of prescription problems based on the presence or absence of special features that we call negotiable features. We propose a solution for each kind of problem, based on global optimisation (combining several models in the deployment phase) and negotiation (introducing new concepts, problems and techniques). In general, in this scenario, obtaining the best global solution analytically is unreachable and simulation techniques are useful in

order to obtain good global results.

Furthermore, when several models are combined, they must be combined using an unbiased criterion. In the case of having an estimated probability for each example, these probabilities must be realistic. In machine learning, the degree to which estimated probabilities match the actual probabilities is known as calibration. We revisit the problem of classifier calibration, proposing a new non-monotonic calibration method inspired in binning-based methods. Moreover, we study the role of calibration before and after probabilistic classifier combination. We present a series of findings that allow us to recommend several layouts for the use of calibration in classifier combination.

Finally, before deploying the model, making single decisions for each new individual, a global view of the problem may be required, in order to study the feasibility of the model or the resources that will be needed. Quantification is a machine learning task that can help to obtain this global view of the problem. We present a new approach to quantification based on scaling the average estimated probability. We also analyse the impact of having good probability estimators for the new quantification methods based on probability average, and the relation of quantification with global calibration.

Summarising, in this work, we have developed new techniques, methods and algorithms that can be applied during the model deployment phase for a better model integration. These new contributions outperform previous approaches, or cover areas that have not already been studied by the machine learning community. As a result, we now have a wider and more powerful range of tools for obtaining good global results when several local models are combined.

Keywords: classifier combination, global optimisation, negotiable features, simulation, calibration, quantification.

Resumen

El aprendizaje automático es un área de investigación que proporciona algoritmos y técnicas que son capaces de aprender automáticamente a partir de experiencias pasadas. Estas técnicas son esenciales en el área de descubrimiento de conocimiento de bases de datos (KDD), cuya fase principal es típicamente conocida como minería de datos. El proceso de KDD se puede ver como el aprendizaje de un modelo a partir de datos anteriores (generación del modelo) y la aplicación de este modelo a nuevos datos (utilización del modelo). La fase de utilización del modelo es muy importante, porque los usuarios y, muy especialmente, las organizaciones toman las decisiones dependiendo del resultado de los modelos.

Por lo general, cada modelo se aprende de forma independiente, intentando obtener el mejor resultado (local). Sin embargo, cuando varios modelos se usan conjuntamente, algunos de ellos pueden depender los unos de los otros (por ejemplo, las salidas de un modelo pueden ser las entradas de otro) y aparecen restricciones. En este escenario, la mejor decisión local para cada problema tratado individualmente podría no dar el mejor resultado global, o el resultado obtenido podría no ser válido si no cumple las restricciones del problema.

El área de administración de la relación con los clientes (CRM) ha dado origen a problemas reales donde la minería de datos y la optimización (global) deben ser usadas conjuntamente. Por ejemplo, los problemas de prescripción de productos tratan de distinguir u ordenar los productos que serán ofrecidos a cada cliente (o simétricamente, elegir los clientes a los que se les debería de ofrecer los productos). Estas áreas (KDD, CRM) carecen de herramientas para tener una visión más completa de los problemas y una mejor integración de los modelos de acuerdo a sus interdependencias y las restricciones globales y locales. La aplicación clásica de minería de datos a problemas de prescripción de productos, por lo general, ha considerado una visión monolítica o estática del proceso, donde uno o más productos son ofrecidos a un conjunto de clientes y se tiene que determinar la secuencia de ofertas (producto, cliente) que maximice el beneficio. Consideramos que es posible realizar una mejor personalización del proceso ajustando o adaptando varios atributos del producto para obtener mayores ganancias. Por lo tanto, presentamos una taxonomía de problemas de prescripción de productos basada en la presencia o ausencia de un tipo de atributos especiales que llamamos atributos negociables. Pro-

ponemos una solución para cada tipo de problema, basada en optimización global (combinando varios modelos en la fase de utilización de los modelos) y negociación (introduciendo nuevos conceptos, problemas y técnicas). En general, en este escenario, obtener la mejor solución global de forma analítica es intratable y usar técnicas de simulación es una manera de obtener buenos resultados a nivel global.

Además, cuando se combinan varios modelos, éstos tienen que combinarse usando un criterio justo. Si para cada ejemplo tenemos su probabilidad estimada, esta probabilidad tiene que ser realista. En aprendizaje automático, el grado en el que las probabilidades estimadas se corresponden con las probabilidades reales se conoce como calibración. Retomamos el problema de la calibración de clasificadores, proponiendo un método de calibración no monotónico inspirado en los métodos basados en "binning". Por otra parte, estudiamos el papel de la calibración antes y después de combinar clasificadores probabilísticos. Y presentamos una serie de conclusiones que nos permiten recomendar varias configuraciones para el uso de la calibración en la combinación de clasificadores.

Por último, antes de usar el modelo, tomando decisiones individuales para cada nuevo ejemplo, puede ser necesaria una visión global del problema, para estudiar la viabilidad del modelo o los recursos que serán necesarios. La cuantificación es una tarea de aprendizaje automático que puede ayudar a obtener esta visión global del problema. Presentamos una nueva aproximación al problema de cuantificación basada en escalar la media de la probabilidad estimada. También se analiza el impacto de tener un buen estimador de probabilidades para estos nuevos métodos de cuantificación, y la relación de la cuantificación con la calibración global.

En resumen, en este trabajo, hemos desarrollado nuevas técnicas, métodos y algoritmos que se pueden aplicar durante la fase de utilización de los modelos para una mejor integración de éstos. Las nuevas contribuciones mejoran las aproximaciones anteriores, o cubren áreas que aún no habían sido estudiadas por la comunidad de aprendizaje automático. Como resultado, ahora tenemos una gama más amplia y potente de herramientas para obtener buenos resultados globales cuando combinamos varios modelos locales.

Palabras clave: combinación de clasificadores, optimización global, atributos negociables, simulación, calibración, cuantificación.

Resum

L'aprenentatge automàtic és una àrea d'investigació que proporciona algoritmes i tècniques que són capaços d'aprendre automàticament a partir d'experiències passades. Estes tècniques són essencials en l'àrea de descobriment de coneixement de bases de dades (KDD), la fase principal de la qual és típicament coneguda com a mineria de dades. El procés de KDD es pot veure com l'aprenentatge d'un model a partir de dades anteriors (generació del model) i l'aplicació d'este model a noves dades (utilització del model). La fase d'utilització del model és molt important, perquè els usuaris i, molt especialment, les organitzacions prenen les decisions depenent del resultat dels models.

Generalment, cada model s'aprén de forma independent, intentant obtindre el millor resultat (local). No obstant això, quan diversos models s'usen conjuntament, alguns d'ells poden dependre els uns dels altres (per exemple, les eixides d'un model poden ser les entrades d'un altre) i apareixen restriccions. En aquest escenari, la millor decisió local per a cada problema tractat individualment podria no donar el millor resultat global, o el resultat obtingut podria no ser vàlid si no complix les restriccions del problema.

L'àrea d'administració de la relació amb els clients (CRM) ha donat origen a problemes reals on la mineria de dades i l'optimització (global) han de ser usades conjuntament. Per exemple, els problemes de prescripció de productes tracten de distingir o ordenar els productes que seran oferits a cada client (o simètricament, triar els clients a qui se'ls deuria d'oferir els productes). Estes àrees (KDD, CRM) no tenen ferramentes per a tindre una visió més completa dels problemes i una millor integració dels models d'acord amb les seues interdependències i les restriccions globals i locals. Generalment, l'aplicació clàssica de mineria de dades a problemes de prescripció de productes ha considerat una visió monolítica o estàtica del procés, on un o més productes són oferits a un conjunt de clients i s'ha de determinar la seqüència d'ofertes (producte, client) que maximitze el benefici. Considerem que és possible realitzar una millor personalització del procés ajustant o adaptant diversos atributs del producte per a obtindre majors guanys. Per tant, presentem una taxonomia de problemes de prescripció de productes basada en la presència o absència d'un tipus d'atributs especials que cridem atributs negociables. Proposem una solució per a cada tipus de problema, basada en optimització global (combinant diversos models en la fase d'utilització) i negociació (introduint nous concep-

tes, problemes i tècniques). En general, en aquest escenari, obtindre la millor solució global de forma analítica és intractable i usar tècniques de simulació és una manera d'obtindre bons resultats a nivell global.

A més, quan es combinen diversos models, estos han de combinar-se usant un criteri just. Si per a cada exemple tenim la seua probabilitat estimada, esta probabilitat ha de ser realista. En aprenentatge automàtic, el grau en què les probabilitats estimades es corresponen amb les probabilitats reals es coneix com a calibració. Reprenem el problema de la calibració de classificadors, proposant un mètode de calibració no monotònic inspirat en els mètodes basats en "binning". D'altra banda, estudiem el paper de la calibració abans i després de combinar classificadors probabilístics. I presentem una sèrie de conclusions que ens permeten recomanar diverses configuracions per a l'ús de la calibració en la combinació de classificadors.

Finalment, abans d'usar el model, prenent decisions individuals per a cada nou exemple, pot ser necessària una visió global del problema, per a estudiar la viabilitat del model o els recursos que seran necessaris. La quantificació és una tasca d'aprenentatge automàtic que pot ajudar a obtindre aquesta visió global del problema. Presentem una nova aproximació al problema de quantificació basada a escalar la mitjana de la probabilitat estimada. També s'analitza l'impacte de tindre un bon estimador de probabilitats per als nous mètodes de quantificació, i la relació de la quantificació amb la calibració global.

En resum, en aquest treball, hem implementat noves tècniques, mètodes i algoritmes que es poden aplicar durant la fase d'utilització dels models per a una millor integració d'aquests. Les noves contribucions milloren les aproximacions anteriors, o cobrixen àrees que encara no havien sigut estudiades per la comunitat d'aprenentatge automàtic. Com a resultat, ara tenim una varietat més àmplia i potent de ferramentes per a obtindre bons resultats globals quan combinem diversos models locals.

Paraules clau: combinació de clasificadors, optimització global, atributs negociables, simulació, calibració, quantificació.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Machine Learning and Data Mining

The amount of information recorded by individuals, companies, governments and other institutions far surpasses the human capacity of exploring it in depth. Computer-aided tools are necessary to help humans find many useful patterns that are hidden in the data. *Machine Learning* is the field of computer science that is concerned with the design of systems which automatically learn from experience [Mit97]. This is a very active research field that has many real-life applications in many different areas such as medical diagnosis, bioinformatics, fraud detection, stock market analysis, speech and handwriting recognition, game playing, software engineering, adaptive websites, search engines, etc. Moreover, machine learning algorithms are the base of other research fields, such as data mining, knowledge discovery from databases, natural language processing, artificial vision, etc.

Data mining is usually seen as an area which integrates many techniques from many different disciplines, including, logically, machine learning. Data mining usually puts more emphasis on the cost effectiveness of the whole process and, very especially, on all the stages of the process, from data preparation to model deployment.

Depending on the kind of knowledge to be obtained there are several techniques, such as decision trees, support vector machines, neural networks, linear models, etc. Depending on the problem presentation, we may distinguish between supervised (predictive) modelling and unsupervised (descriptive) modelling. In this dissertation, we will mostly focus on predictive modelling. Two of the most important supervised data mining tasks are *classification* and *regression*. Both classification and regression techniques are usually widely employed in decision making.

Classification can be seen as the clarification of a dependency, in which

each dependent attribute can take a value from several classes, known in advance. Classification techniques are used when a nominal (categorical) value is predicted (positive or negative, yes or no, A or B or C, etc.). Moreover, some classifiers can accompany the predicted value by a level of confidence or probability. They are called *Probabilistic Classifiers*.

Example 1

An organisation wants to offer two new similar products (one e-book with Wi-Fi and another e-book with bluetooth and 3G) to its customers. Using past selling data experiences of similar customers and/or products, a probabilistic classifier can be learned. This model obtains the probability, for each customer, of buying each product. The organisation can deploy this learned model in the design of a mailing campaign to select the best customers for receiving an offer for each product.

The goal of *regression* techniques is to predict the value of a numerical (continuous) variable (567 metres, 43 years, etc.) from other variables (which might be of any type).

Example 2

A hospital information system (HIS) stores past data of the activity of the hospital. From these data, a regression model can be learned to predict the number of admissions in a day. Estimating the number of admissions in advance is useful to plan the necessary resources in the hospital (operating theatres, material, physicians, nurses, etc.).

As we have seen in these examples, the next step, after learning a model (in the model generation phase), is to apply it to new instances of the problem. This crucial phase is called *model deployment*. Several things may happen at this stage. For instance, we can realise that the result of a model depends on the result of other models or does not fulfill the constraints of the problem, we may need to obtain further information from the model, etc. In general, discarding the current model and learning another model is not a solution to sort out these problems. On one hand, discarding the model will be a waste of time and money. On the other hand, learning another model would give the same result.

Some machine learning techniques have been developed to be applied during the model deployment phase such as classifier combination, calibration and quantification. Some of these techniques will be very relevant in this work and will be introduced in subsequent sections.

1.2 Motivation

Examples 1 and 2 show how data mining can help to make a single decision. Typically, however, an organisation has to make several complex decisions, which are interwoven with the rest and with a series of constraints.

In Example 1, a constraint could be that both products cannot be offered to the same customer, because they are quite similar and a customer will not buy both products together. If we have marketing costs and/or a limited stock of products, as usual, the best decision will not be to offer the most attractive product to each customer. For instance, it could be better to offer *product B*, which is less desirable, to a customer that could probably buy both products, and offer *product A* to a more difficult customer that would never buy *product B*.

In Example 2, bed occupation depends on the number of admissions as well as some constraints, such as the working schedule of nurses and physicians, that have to be fulfilled. A local decision might assign the same physician (a very good surgeon) to all the surgeries in a hospital, which is clearly unfeasible.

Therefore, in real situations, as we have seen in these examples, making the best local decision for every problem does not give the best global result. Models must be integrated with the objective of obtaining a *good global result*, which follows all the constraints.

One particular view of combining multiple decisions from a set of classifiers is known as *classifier combination* or *classifier fusion* [Kun04]. The need for classifier combination is well-known. On one hand, more and more applications require the integration of models and experts that come from different sources (human experts models, data mining or machine learning models, etc.). On the other hand, it has been shown that an appropriate combination of several models can give better results than any of the single models alone [Kun04][TJGD08], especially if the base classifiers are diverse [KW02].

Basically, each model gives an output (a decision) and then, the outputs are weighted in order to obtain a single decision. This integrated model is known as *ensemble* [Die00a][Kun04], i.e., a *committee model*. Different techniques have been developed depending whether the set of base classifiers are homogeneous (a single algorithm is used and diversity is achieve through some form of variability in the data): boosting [FS96], bagging [Bre96], randomisation [Die00b], etc.; or heterogeneous (multiple algorithms are used in order to obtain diversity): stacking [Wol92], cascading [GB00], delegating [FFHO04], etc.

For instance, following with Example 2, in Figure 1.1 we show an ensemble that combines three different models in committee: opinion of a human expert,

a linear regression model and a neural network model. Each model obtains the number of hospital admissions per day. The ensemble method combines the result of the three models obtaining the global result (number of hospital admissions per day).

However, in real applications, models are not only combined in committee, but they can also be combined or related to each other in very different ways (outputs of the models are inputs of other models, possibly generating cycles or other complex structures). We use the term *chained models* to distinguish them from *committee models*. In this context, several computational techniques (linear programming, simulation, numerical computation, operational research, etc.) can be used to approach the optimisation task. The problem is that data mining models are usually expressed in a non-algebraic way (e.g. decision trees) or even as a black-box (e.g. neural networks). Moreover, in many situations, there is no on-purpose generation of a set of classifiers and we have to combine opinions from many sources (either humans or machines), into a single decision. Consequently, many optimisation techniques are no longer valid because the mathematical properties (continuity, monotonicity, etc.) of the functions that describe the data mining models are unknown.

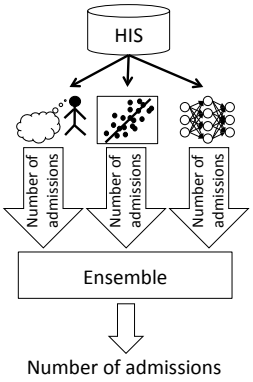


Figure 1.1: Example of models combined in committee.

In Figure 1.2, we show three chained models: opinion of a human expert, a neural network model and a decision tree model. Using the data in the HIS and her/his own knowledge, the expert says who the best surgeon is for a surgery. The output of this first model (the surgeon) is used as an input to the second model. Here, using the data in the HIS and using the surgeon selected by the first model, a neural network is applied, in order to obtain an anesthetist for the surgery. And finally, the output of the second model (the

anesthetist) is used as an input for the third model, a decision tree. Using the data from the HIS and using the anesthetist given by the second model, the decision tree assigns the operation theatre for the surgery.

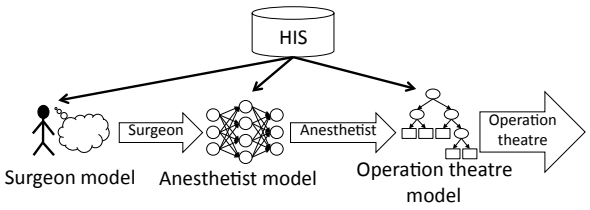


Figure 1.2: Example of chained models.

As we can see in Figure 1.2, bad decisions are now more critical than in committee (or isolated) models. A bad choice of the surgeon has strong implications on the rest of models and predictions. Even a minor maladjustment can have a big impact on the global system. Figure 1.2 also suggests a more holistic approach to the problem. For instance, we can ask questions such as: "How would the operation theatre allocation change if we had a new surgeon?". Or, in Example 1, "how many more products would we sell if we reduced the price of *product A* by a 25%?". All these questions are related to the problem of determining the output of a model if one or more input features are altered or *negotiated*. These *negotiable features* are the key variables for a *global optimisation*. One way of tackling this problem is through the notion of *problem inversion*, i.e., given the output of a data mining model, we want to get some of the inputs.

We have seen that a decision can depend on the combination of several models (either in committee or chained) but, also, a decision can depend on the total or the sum of each individual decision. In machine learning, this task is known as *quantification* and was introduced and systematised by George Forman [For05][For06][For08]. Quantification is defined as follows: "given a labelled training set, induce a *quantifier* that takes an unlabelled test set as input and returns its best estimate of the class distribution."[For06].

For instance, following with Example 1, where the organisation wants to know which customers will buy its products, it makes sense to start with a more general question for the mailing campaign design: the organisation needs to know how many products it is going to sell. This quantification is critical to assign human and economical resources, to fix stock of products or, even, to give up the campaign, if the estimated number of products to be sold is

not appropriate for the organisation. In Example 2, for instance, we may have a maximum number of surgeries per surgeon and we need to quantify whether the model is going to exceed this maximum, and gauge the model appropriately.

It is important to use an unbiased mechanism when several predictions are integrated. For example, if we have two probabilistic models and we want to obtain a list sorted by the estimated probability of both models, it is important that these probabilities are realistic. In machine learning terminology, these probabilities are said to be *calibrated*. *Calibration* is important when we are combining models in *committee*, because if the outputs of the models are not reliable, the output of the combined models would be a disaster. Nonetheless, depending on the ensemble method, the result could still be good if the ensemble method still gives more weight to the most accurate models (while ignoring the probabilities). However, when we are *chaining models*, a bad calibrated output is usually magnified, making the whole system diverge much more easily. This is more manifest when the relation between inputs and outputs is not monotonic. For instance, if we have a system handling stocks and the model predicting each product expenditure usually underestimates, we will have common stockouts.

The problem of integrating several models (in different ways) and using them for several tasks requires a much deeper analysis of how models perform in an isolated way but, more importantly, of how they work together (either as a committee or a chain) or how they can be used for aggregated decisions (e.g., quantification). Apart from the analysis of local issues, it is necessary to understand the problems in a *global optimisation* setting.

In the literature, we find works where the conjunction of data mining and (global) optimisation is studied [BGL07][PdP05][PT03]. These works address specific situations such as rank aggregation [FKM$^+$04] and cost-sensitive learning. A more general "utility-based data mining"[1] also addressed this issue, but the emphasis was placed on the economic factors of the process, rather than a global integration of data models.

Much of this work originates from real application problems which appear in the area of Customer Relationship Management (CRM) [BST00][BL99]. CRM is an application field where econometrics and mainstream data mining can merge, along with techniques from *simulation*, operational research, artificial intelligence and numerical computation.

Decisions in the context of prescription problems deal about distinguishing or ranking the products to be offered to each customer (or, symmetrically,

[1] http://storm.cis.fordham.edu/~gweiss/ubdm-kdd05.html

selecting the customers to whom we should make an offer), establishing the moment or sequence of the offers, and determining the price, warranty, financing or other associated features of products and customers. The classical application of data mining for prescription problems has usually considered a partial view of the process, where we have one or more products to be offered to a pool of customers, and we need to determine a sequence of offers (product, customer) to maximise profit. These and related problems (e.g. cross-selling or up-selling) have been addressed with techniques known as "mailing/selling campaign design" [BL99] or from the more general view of recommender systems [AT05], which are typically based on data mining models which perform good rankings and/or good probability estimations.

However, in more realistic and interactive scenarios, we need to consider that a better customisation has to be performed. It is not only the choice of products or customers which is possible, but several features of the product (or the deal) can be tuned or adapted to get more earnings.

Moreover, all these approaches can be very helpful in specific situations, but most of the scenarios we face in real data mining applications do not fit many of the assumptions or settings of these previous works. In fact, many real scenarios are so complex that the optimal decision cannot be found analytically. In this situation, we want to obtain the best global solution, but in real problems it is usually impossible to explore all the solution space. Therefore, techniques based on simulation are needed in order to obtain the best possible solution in a feasible period of time.

To sum up, we have seen that making the best local decision for every problem does not give the best global result, which can even be invalid in some cases (due to problem constraints). We have detected that there is a lack of general methods and techniques, in the model deployment phase, to address all these situations mentioned above. Therefore, we want to propose model integration and deployment methods that obtain good and feasible global results.

1.3 Research objectives

The main hypothesis of this work is that it is possible to get much better global results when dealing with complex systems if we integrate several data mining models in a way that takes global issues (constraints, costs, etc.) into account. This thesis is supported by the fact that techniques and models are generally specialised to obtain a local optimum result, instead of obtaining a global optimum result.

From the previous statement, the main objective of this thesis is to develop

new techniques and strategies in the *model deployment* phase, in order to obtain good global results when several local models are integrated.

In particular, we focus on:

- Techniques that combine local models obtaining *good global decisions* and fulfilling constraints.
- Better understanding of the relation between input and output features, the *problem inversion* approach and the use of input features for *global optimisation* and *negotiation*.
- More powerful *calibration* techniques and their relation with classifier integration.
- Methods that make a global decision from the sum of individual decisions, such as *quantification*.

Furthermore, all the developed methods and techniques have to be applicable, in the end, to real problems, especially, in the areas of CRM (prescription models, campaigns, cost quantification, ...), recommender systems, complex systems, etc.

1.4 Structure of this dissertation

The thesis is organised in two parts:

- Part I reviews the main contributions of this thesis. It is arranged in four chapters:
  - In Chapter 2, we present a taxonomy of prescription problems that considers whether the price of the product is fixed or negotiable, and the number of products and customers. We propose a solution, for each kind of problem, based on global optimisation (combining several models in the deployment phase) and negotiation (introducing new concepts, problems and techniques such as negotiable feature, problem inversion and negotiation strategies).
  - In Chapter 3, we revisit the problem of classifier calibration. We propose a new non-monotonic calibration method inspired in binning-based methods. We study the effect of calibration in probabilistic classifier combination.

  - Chapter 4 deals with the quantification problem. We present a new quantification method based on scaling probabilities, i.e., the average probability of a dataset is scaled following a normalisation formula. We analyse the impact of calibration in this new quantification method, and the relation of quantification with global calibration.
  - The last chapter of this part contains the conclusions and future work (Chapter 5).

- Part II starts with Chapter 6, which includes the list of publications associated with this thesis. Finally, Chapter 7 has the full text of the publications.

Part I

Summary of the Contributions

2

Global Optimisation and Negotiation in Prescription Problems

Examples 1 and 2 in Chapter 1 illustrate that, in real situations, making the best local decision for every problem does not give the best global result which satisfies all the constraints. In this scenario, there are many data mining problems in which one or more input features can be modified at the time the model is applied (model deployment phase), turning the problem into some kind of negotiation process. For instance, in Example 1, the price of a product changes the decision of buying it or not. In Figure 2.1, we show the actual probability of buying a product depending on its price. Only if the price of the product is less or equal than a maximum price, will the customer buy the product.

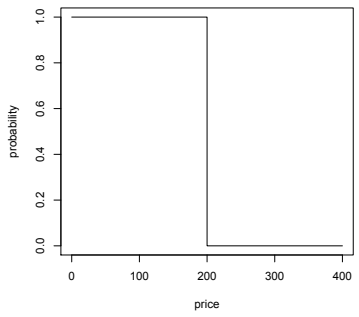


Figure 2.1: Real probability of buying a product depending on its price.

We call these attributes *negotiable features*. In Section 7.7 there is a formal definition of negotiable features with its properties. Here, we will just introduce a simpler and more intuitive definition. A *negotiable feature* is an input attribute that fulfils three conditions:

1. it can be varied at model application time,
2. the output value of the instance changes when the negotiable feature changes, while fixing the value of the rest of input attributes,
3. the relation between its value and the output value is monotonically increasing or decreasing.

For example, in a loan granting model, where loans are granted or not according to a model that has been learned from previous customer behaviours, the age of the customer is not a negotiable feature, since we cannot modify it (condition 1 is violated). The bank branch office where the contract can take place is also an input, which we can modify, but it is not a negotiable feature either since it rarely affects the output (condition 2 is violated). The number of meetings is also modifiable and it frequently affects the result, but it is usually non-monotonic, so it is not a negotiable feature either (condition 3 is violated). In contrast, the loan amount, the interest rate or the loan period are negotiable features since very large loan amounts, very low interest rates and very long loan periods make loans unfeasible for the bank.

This chapter will use a running (real) example: a CRM retailing problem dealing with houses handled by an estate agent. In what follows, the negotiable feature will be the price (denoted by π) and the problem will be a classification problem (buying or not).

In our problem, we know that customers (buyers) have a *maximum price* per property they are not meant to surpass. This maximum price is not known by the seller, but estimated with the data mining models. Conversely, the seller (real estate agent) has a *minimum price* (denoted by π_{min}) for each type of product, which typically includes the price the owner wants for the house plus the operational cost. This minimum price is not known by the buyer. Any increment over this minimum price is profitable for the seller. Conversely, selling under this value is not acceptable for the seller. Therefore, the seller will not sell the product if its price is under this minimum price. This means that the profit obtained by the product will be the difference between the selling price and the minimum price (Formula 2.1).

$$Profit(\pi) = \pi - \pi_{min} \tag{2.1}$$

Clearly, only when the maximum price is greater than the minimum price, there is a real chance of making a deal, and the objective for the seller is to maximise the *expected profit* (Formula 2.2).

$$E(Profit(\pi)) = \hat{p}(c|\pi) \cdot Profit(\pi) \tag{2.2}$$

where $\hat{p}(c|\pi)$ is the probability estimated by the model, for the class c and price π.

2.1 A taxonomy of prescription problems

Typically, data mining models are used in prescription problems to model the behaviour of one individual for one item, in a static scenario (all the problem attributes are fixed). In this thesis, we also consider the case where the items can be adapted to the individual, by adjusting the value of some *negotiable features*. We begin by devising a taxonomy of prescription problems which considers the number of products and customers involved as well as the fixed or negotiable nature of the features of each product (Table 2.1). This will help to recognise the previous work in this area and the open problems we aim to solve. We consider the different kinds of products (N), the presence or absence of negotiation and the number of customers (C). The last column shows several approaches that have been proposed for solving these problems. In each row where this case is (first) addressed in this dissertation, we indicate the section where the full paper can be consulted. We discuss each case in detail in the next sections.

Table 2.1: Different prescription problems that consider the number of different kinds of products to sell, whether the net price for the product is fixed or negotiable, and the number of customers.

Case	Kinds of products	Features	Number of customers	Approach
1	1	fixed	1	Trivial
2	1	fixed	C	Customer ranking [BL99]
3	N	fixed	1	Product ranking [BL99]
4	N	fixed	C	Joint Cutoff (Section 7.1)
5	1	negotiable	1	Negotiable Features (Section 7.7)
6	1	negotiable	C	Negotiable Features (Section 7.7)
7	N	negotiable	1	Negotiable Features (Section 7.7)
8	N	negotiable	C	Negotiable Features (Section 7.7)

2.2 Cases with fixed features

The case with one kind of product, fixed features, and one customer (case 1 in Table 2.1) is trivial. In this scenario, the seller offers the product to the customer with fixed conditions/features and the customer may buy or not the product.

The case with one kind of product, fixed features, and C customers (case 2 in Table 2.1) is the typical case, for example, of a mailing campaign design. The objective is to obtain a customer ranking to determine the set of customers to whom the mailing campaign should be directed in order to obtain the maximum profit. Data mining can help in this situation by learning a probabilistic classification model from previous customer data that includes information about similar products that have been sold to them. This model will obtain the buying probability for each customer. Sorting them by decreasing buying probability, the most desirable customers will be at the top of the ranking. Using a simple formula for marketing costs (more details can be seen in Section 7.1), we can establish a (local) threshold/cutoff in this ranking. The customers above the threshold will receive the offer for the product.

The case with N kinds of products, fixed features, and one customer (case 3 in Table 2.1) is symmetric to case 2. Instead of N customers and one product, in this case, there are N different products and only one customer. Hence, the objective is to obtain a product ranking for the customer.

The case with N kinds of products, fixed features, and C customers (case 4 in Table 2.1) is more complex than cases 2 and 3, since there is a data mining model for each product. In other words, there are N customer rankings (one for each product) and the objective is to obtain the set of pairs customer-product that gives the maximum overall profit. Note that, normally, the best local cutoff of each model (the set of customers that gives the maximum profit for one product) does not give the best global result. Moreover, there are several constraints that are frequently required in real applications (limited stock of products, the customers may be restricted to only buying one product).

Two different methods are proposed in order to obtain the global cutoff:

- *Single approach*: It is a very simple method. It is based on averaging the local cutoffs.

    1. For each product $i = 1...N$, we use the customers ranking to find the local cutoffs T_i, as in case 2 in Table 2.1.
    2. We join the rankings for all the products into one single ranking, and we sort it downwards by their expected profit.

3. The global cutoff is the average of the local cutoffs, i.e., $\frac{1}{N}\sum_{i=1}^{N} T_i$.

- *Joint simulation approach*: It is based on obtaining the global cutoff by simulation, taking all the constraints into account. It consists of:

  1. Sorting (jointly) the customers downwards by their expected profit for all the products (as in point 2 above).
  2. Calculating by simulation, using a Petri net, the accumulated profit for each threshold or cutoff (i.e., for each pair customer product). The first cutoff considers the first element of the ranking, the second cutoff the two first elements, and so on. Therefore, $N \times C$ cases (cutoffs) are simulated. In each of them all the constraints are satisfied and the accumulated profit is calculated, i.e., the sum of the profit for the elements in the ranking above the cutoff.
  3. The cutoff that gives the best accumulated profit is the *global cutoff*.

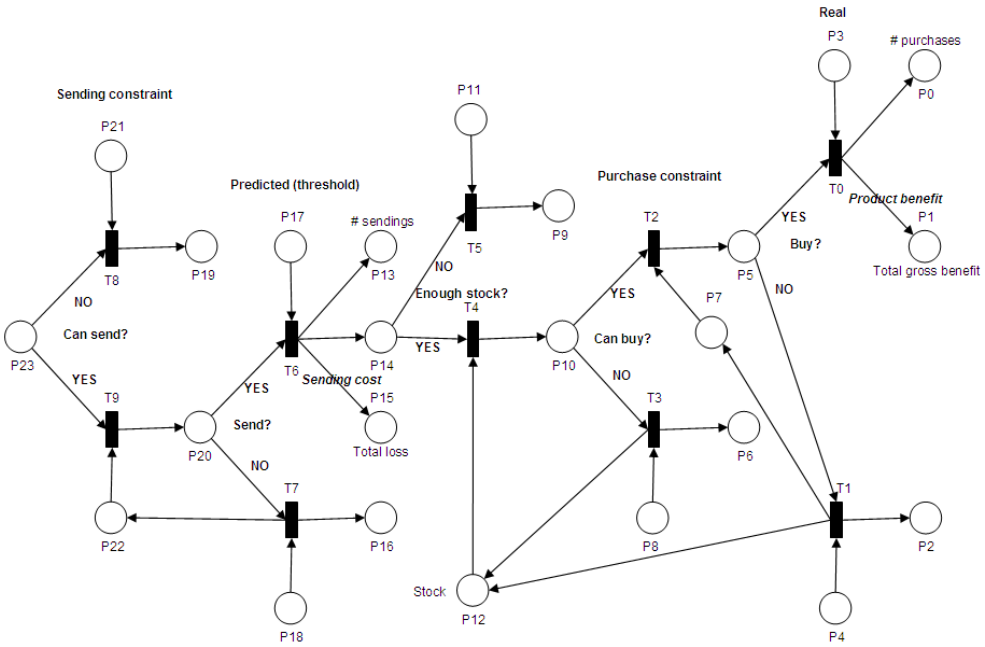


Figure 2.2: Petri net for a mailing campaign.

We adopted Petri nets as a framework to formalise the simulation because they are well-known, easy to understand and flexible. In Figure 2.2 we show

the Petri net used for simulating a mailing campaign in our paper "Joint Cutoff Probabilistic Estimation using Simulation: A Mailing Campaign Application" that can be consulted in Section 7.1. More details can be obtained in the paper, but, basically, a Petri net has places (represented by circles) and transitions (represented by black boxes). A transition is fired when all the input places have at least one token into them. When a transition is fired, it puts one token into its output places. In this way, putting the tokens into the right places in each moment, we achieve a way to calculate the accumulated profit for each cutoff, where the constraints are fulfilled

The experimental results from our paper "Joint Cutoff Probabilistic Estimation using Simulation: A Mailing Campaign Application" that can be consulted in Section 7.1 show that using simulation to set model cutoff obtains better results than classical analytical methods.

2.3 Cases with negotiable features

Before starting with the cases with negotiable features, we are going to define the techniques used to solve these cases.

2.3.1 Inverting problem presentation

Imagine a model that estimates the delivery time for an order depending on the kind of product and the units which are ordered. One possible (traditional) use of this model is to predict the delivery time given a new order. However, another use of this model is to determine the number of units (provided it is a negotiable feature) that can be delivered in a fixed period of time, e.g. one week. This is an example of an *inverse use* of a data mining model, where all inputs except one and the output are fixed, and the objective is to determine the remaining input value. A formal definition of inversion problem can be consulted in Section 7.7.

For two classes, we assume a working hypothesis which allows us to derive the probabilities for each value of the negotiable feature in an almost direct way. First, we learn a model from the inverted problem, where the datasets outputs are set as inputs and the negotiable feature is set as output. In the example, the inverted problem would be a regression problem with the kind of product and delivery time as input, and the number of units as output. Second, we take a new instance and obtain the value for the learned model. In our example, for each instance, we would obtain the number of units $\hat{a}$ of a kind of product that could be manufactured in a fixed delivery time. Third, we make the reasonable assumption of giving to the output (the negotiable

feature) the probability of 0.5 of being less or equal than $\hat{a}$. Fourth, we assume a normal distribution with mean at this value $\hat{a}$ and the relative error (of the training set) as standard deviation. And fifth, from the normal distribution we calculate its associated cumulative distribution function and, in this way, we obtain a probability for each value of the negotiable feature.

Figure 2.3 shows an example where the value obtained by the inversion problem is $305,677.9$ and the relative error is 59,209.06. On the left hand side we show a normal distribution with centre at $\hat{a} = 305,677.9$ and standard deviation $\sigma = 59,209.06$, and on the right had side we show the cumulative distribution function associated to this normal distribution. In the cumulative distribution function we can observe that it is possible to obtain the probability value (Y axis) for each price (X axis).

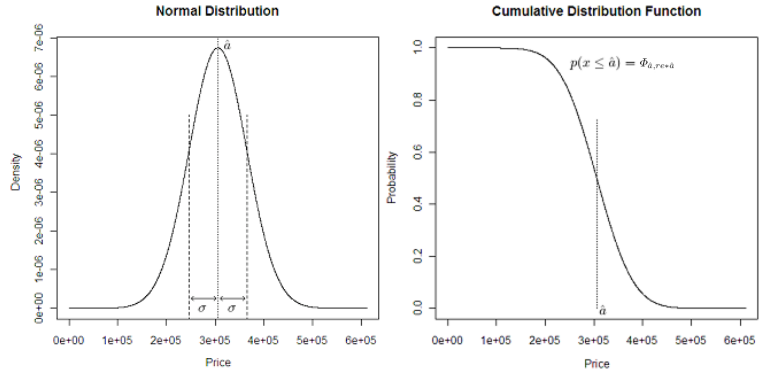


Figure 2.3: **Left:** Example of a normal distribution $\hat{a} = 305,677.9$ and $\sigma = 59,209.06$. **Right:** Associated cumulative distribution function.

2.3.2 Negotiation strategies

The novel thing in this retailing scenario is not only that we allow the seller to play or gauge the price to maximise the expected profit, but we also allow several bids or offers to be made to the same customer. This means that if an offer is rejected, the seller can offer again. The number of offers or bids that are allowed in an application is variable, but it is usually a small number, to prevent the buyer from getting tired of the bargaining.

We propose three simple negotiation strategies in this setting. For cases with one single bid, we introduce the strategy called *Maximum Expected Profit (MEP)*. For cases with more bids (multi-bid) we present two strategies: the

Best Local Expected Profit (BLEP) strategy and the *Maximum Global Optimisation (MGO)* strategy. Let us see all of them in detail below:

- Maximum Expected Profit (MEP) strategy (1 bid). This strategy is typically used in marketing when the seller can only make one offer to the customer. Given a probabilistic model each price for an instance gives a probability of buying, as shown in Figure 2.4 (top-left). This strategy chooses the price that maximises the value of the expected profit (Figure 2.4, top-right). The expected profit is the product of the probability and the price. In Figure 2.4 (top-right), the black dot is the MEP point (the maximum expected profit point). Note that, in this case, this price is between the minimum price (represented by the dashed line) and the maximum price (represented by the dotted line), which means that this offer would be accepted by the buyer.

- Best Local Expected Profit (BLEP) strategy (N bids). This strategy consists in applying the MEP strategy iteratively, when it is possible to make more that one offer to the buyer. The first offer is the MEP, and if the customer does not accept the offer, his/her curve of estimated probabilities is normalised taking into account the following: the probabilities of buying that are less than or equal to the probability of buying at this price will be set to 0; and the probabilities greater than the probability of buying at this price will be normalised between 0 and 1. The next offer will be calculated by applying the MEP strategy to the normalised probabilities. More details are given in Section 7.7.

  Figure 2.4 (centre-left) shows the three probability curves obtained in three steps of the algorithm and Figure 2.4 (centre-right) shows the corresponding expected profit curves. The solid black line on the left chart is the initial probability curve and the point labelled by 1 on the right chart is its MEP point. In this example, the offer is rejected by the customer (this offer is greater than the maximum price), so probabilities are normalised following the process explained above. This gives a new probability curve represented on the left chart as a dashed red line and its associated expected profit curve (also represented by a dashed red line on the chart on the right), with point 2 being the new MEP point for this second iteration. Again, the offer is not accepted and the normalisation process is applied (dotted green lines in both charts).

- Maximum Global Optimisation (MGO) strategy (N bids). The objective of this strategy is to obtain the N offers that maximise the expected profit. The optimisation formula can be consulted in Section 7.7. The

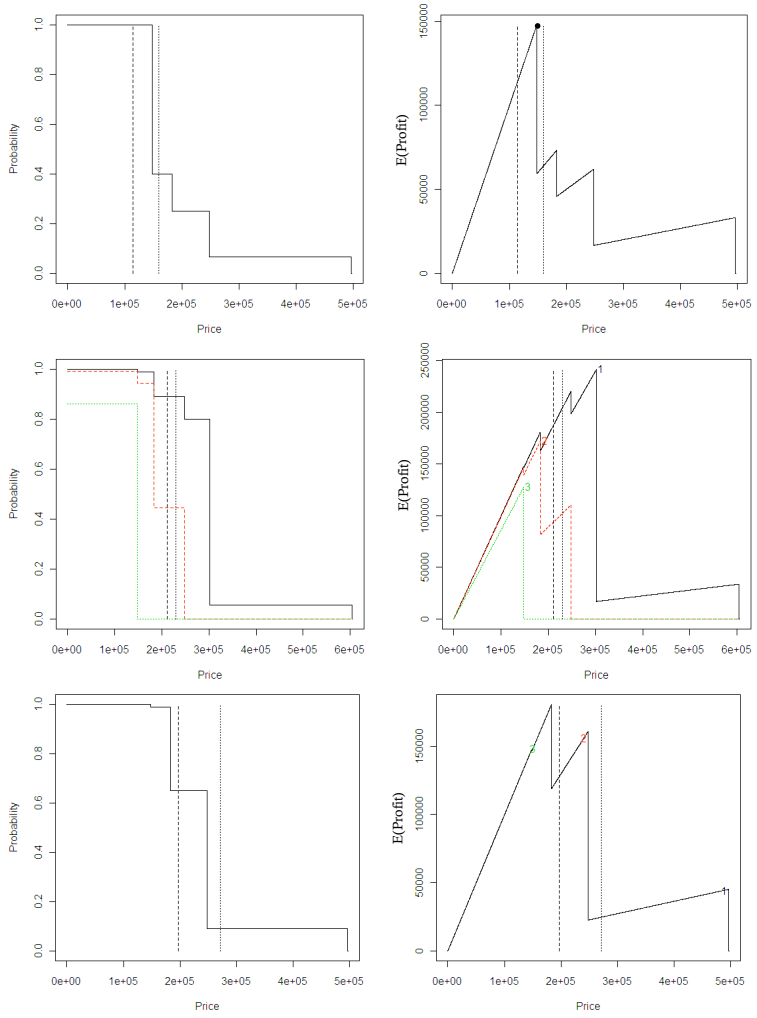


Figure 2.4: Examples of the MEP (**top**), BLEP (**centre**) and MGO (**down**) strategies. **Left**: Estimated probability. **Right**: Associated expected profit.

rationale of this formula is that we use a probabilistic accounting of what happens when we fail or not with the bid. Consequently, optimising the formula is a global approach to the problem.

Computing the N bids from the formula is not direct but can be done in several ways. One option is just using a Montecarlo approach [MU49] with a sufficient number of tuples to get the values for the prices that

maximise the expected profit. Figure 2.4 (down-right) shows the three points given by the MGO strategy for the probability curve on Figure 2.4 (down-left). As we can see, the three points are sorted in decreasing order of price.

2.3.3 Solving cases with negotiable features

After stating the negotiation strategies, we can explain how the four new problems with *negotiable features* (the last four cases in Table 2.1) are solved.

We start with the simplest negotiation scenario, where there are only one seller and one buyer who both negotiate for one product (case 5 in Table 2.1). The buyer is interested in one specific product. S/he likes the product and s/he will buy the product if its price is below a certain price that s/he is willing to pay for this product. It is clear that in this case the price meets the conditions to be a negotiable feature.

On the other hand, the goal of the seller is to sell the product at the maximum possible price. Logically, the higher the price the lower the probability of selling the item. So the goal is to maximise the expected profit (probability multiplied by profit). If probabilities are well estimated, for all the range of possible prices, this must be the optimal strategy if there is only one bid. In Figure 2.5 we show an example of the plots that are obtained for the estimated probabilities and expected profit.

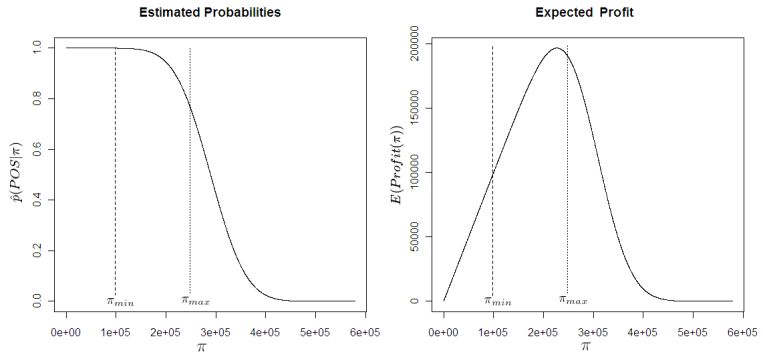


Figure 2.5: **Left:** Example of estimated probabilities. **Right:** Associated expected profit. The minimum and maximum price are also shown.

In the case with one kind of product, negotiable price, and C customers (case 6 in Table 2.1), there is a curve for each customer (Figure 2.6, Left), being all of them similar to the curve in case 5. If the seller can only make one

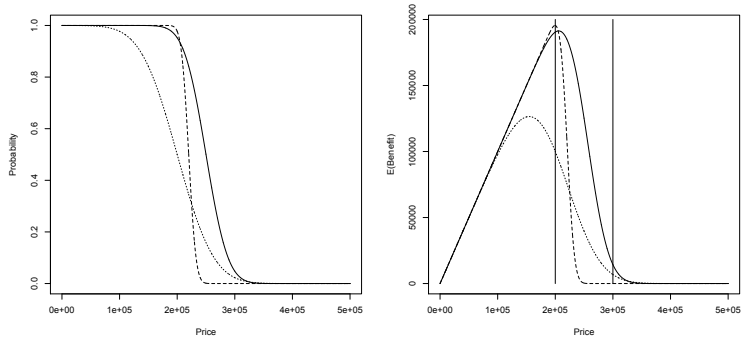


Figure 2.6: Probabilistic buying models of 3 different customers approximated by 3 normal distributions with $\mu_1 = 250,000$ and $\sigma_1 = 30,000$, $\mu_2 = 220,000$ and $\sigma_2 = 10,000$, and $\mu_3 = 200,000$ and $\sigma_3 = 50,000$. **Left**: Probability distribution function. **Right**: Associated expected profit.

offer to the customers, the seller will offer the product at the price that gives the maximum expected profit (in relation to all the expected profit curves) to the customer whose curve achieves the maximum. However, if the seller can make several offers, the seller will distribute the offers along the curves following a negotiation strategy. In this case, the seller not only changes the price of the product, but the seller can also change the customer that s/he is negotiating with, depending on the price of the product (that is, by selecting the customer in each bid who gives the greatest expected profit at this price). Therefore, these curves can be seen as a ranking of customers for each price.

The case with N kind of products, a negotiable price, and one customer (case 7 in Table 2.1) is symmetric to case 6. Instead of one curve for each customer, there is one curve for each product. In this case, the curves represent a ranking of products for that customer.

The case with N kind of products, a negotiable price, and C customers (case 8 in Table 2.1) is the most complete (and complex) of all. The objective is to offer the products to the customers at the best price in order to obtain the maximum profit. Multiple scenarios can be proposed for this situation: each customer can buy only one product; each customer can buy several products; if the customer buys something, it will be more difficult to buy another product; there is a limited stock; etc.

In cases 6, 7 and 8, we typically work with only one data mining model that has the customer's features and the product's features (one of them being the negotiable feature) as inputs. We can, of course, define C different models

in case 6, N different models in case 7, or even C, N or CxN different models for case 8. Nonetheless, this is not necessary and the higher the number of models is the more difficult is to learn and use them and is prone to overfitting.

Graphically, the most appropriate customer or product for each price is represented by the envelope curve. Therefore, in the cases 6, 7 and 8 there are several curves, but the envelope curve must be calculated giving as a result only one curve. Consequently, we can use the same negotiation strategies applied to the case 5 to the envelope curve of cases 6, 7 and 8.

2.4 Results

In this chapter, we have studied eight different prescription problems. Cases 1, 2 and 3 are classical prescription problems studied in the literature. Case 1 is trivial, and cases 2 and 3 correspond to the ranking of customers and products, respectively. In case 4 we have analysed the problem of having several rankings of customers or products. In this case, the best results have been obtained using a simulation method (Joint simulation approach) to calculate the global cutoff. The results of this study are in our paper "Joint Cutoff Probabilistic Estimation using Simulation: A Mailing Campaign Application" that can be consulted in Section 7.1. Cases 5, 6, 7 and 8 have afforded the problem of having negotiable features. We have introduced the concept of negotiable feature and developed new negotiation strategies in order to solve these problems. The negotiation strategy based on *global optimisation* (MGO) obtained the best results. This experimental study is in our paper "Using Negotiable Features for Prescription Problems" [BFHORQ11] that can be consulted in Section 7.7.

3

Similarity-Binning Averaging Calibration

In Chapter 2 we saw that when several decisions are integrated, they must be integrated using a fair criterion. For instance, in Example 1 in Chapter 1, we can calculate, for each customer, the probability of buying the *product A* and obtain the ranking of the best buyers for this product. In this case, since we only have one model and we want to perform a ranking, it is not important the magnitude of the probability, it is only important the order. That is, it does not matter if the probability of buying the product for customer 1, 2 and 3 is 0.98, 0.91 and 0.78, respectively, or 0.82, 0.80 and 0.75, because the ranking will be the same. However, if we have calculated another ranking for *product B*, and we want to combine both models in order to obtain a unified ranking for both products, the probabilities of both models must be realistic. In machine learning language, these probabilities must be *calibrated*.

Calibration is defined as the degree of approximation of the predicted probabilities to the actual probabilities. If we predict that we are 99% sure, we should expect to be right 99% of the time. A calibration technique is any postprocessing technique which aims at improving the probability estimation of a given classifier. Given a general calibration technique, we can use it to improve class probabilities of any existing machine learning method: decision trees, neural networks, kernel methods, instance-based methods, Bayesian methods, etc., but it can also be applied to hand-made models, expert systems or combined models.

In the paper "Calibration of Machine Learning Models" [BFHORQ10a], which can be consulted in Section 7.3, we present the most common calibration techniques and calibration measures. Both classification and regression are covered, and a taxonomy of calibration techniques is established. Special attention is given to probabilistic classifier calibration.

Moreover, in classifier combination, weighting is used in such a way that more reliable classifiers are given more weight than other less reliable classifiers. So, if we combine probabilistic classifiers and use weights, we can have a *double weighting* combination scheme, where estimated probabilities are used as the *second weight* for each classifier. This suggests to study the relation between calibration and combination.

This chapter is motivated by the realisation that existing calibration methods only use the estimated probability to calculate the calibrated probability (i.e., they are univariate). And, also, they are monotonic; they do not change the rank (order) of the examples according to each class estimated probability. We consider that these restrictions can limit the calibration process.

In this chapter, on one hand, we introduce a new multivariate and non-monotonic calibration method, called Similarity-Binning Averaging (SBA).

On the other hand, we also study the role of calibration before and after classifier combination. We present a series of findings that allow us to recommend several layouts for the use of calibration in classifier combination. In this study, we analyse also the effect of the SBA calibration method in classifier combination.

3.1 Calibration methods and evaluation measures

In this section we review some of the most-known calibration methods, introduce the evaluation measures we will employ to estimate the calibration of a classifier, present a taxonomy of calibration methods in terms of monotonicity and comment the multiclass extensions for calibration methods.

We use the following notation. Given a dataset T, n denotes the number of examples, and c the number of classes. The target function $f(i,j)$ represents whether example i actually belongs to class j. Also, $n_j = \sum_{i=1}^{n} f(i,j)$ denotes the number of examples of class j and $p(j) = n_j/n$ denotes the prior probability of class j. Given a classifier l, $p_l(i,j)$ represents the estimated probability of example i to belong to class j taking values in [0,1]. If there is only one classifier we omit the subindex l, and in case of binary classifiers we omit the j index (that indicates the class), i.e., we only represent the probability of one class (the positive class), because the probability of the other class (the negative class) is: $p(i,-) = 1-p(i,+)$. Therefore, we use $f(i)$ and $p(i)$, instead of $f_l(i,+)$ and $p_l(i,+)$, respectively.

3.1.1 Calibration methods

The objective of calibration methods (as a postprocessing) is to transform the original estimated probabilities. Some well-known calibration methods for binary problems are:

- The Binning Averaging method: it was proposed by [ZE02] as a method where a (validation) dataset is split into bins in order to calculate a probability for each bin. Specifically, this method consists in sorting the examples in decreasing order by their estimated probabilities and dividing the set into k bins. Thus, each test example i is placed into a bin t, $1 \leq t \leq k$, according to its probability estimation. Then the corrected probability estimate for i ($p^*(i)$) is obtained as the proportion of instances in t of the positive class.

- The Platt's method [Pla99]: Platt presented a parametric approach for fitting a sigmoid function that maps SVM predictions to calibrated probabilities. The idea is to determine the parameters A and B of the sigmoid function:

$$p^*(i) = \frac{1}{1 + e^{A \cdot p(i) + B}}$$

  that minimise the negative log-likelihood of the data, that is:

$$argmin_{A,B}\{-\sum_i f(i) log(p^*(i)) + (1 - f(i)) log(1 - p^*(i))\}$$

  This two-parameter minimisation problem can be performed by using an optimisation algorithm, such as gradient descent. Platt proposed using either cross-validation or a hold-out set for deriving an unbiased sigmoid training set for estimating A and B.

- The Isotonic Regression method [RWD88]: in this case, the calibrated predictions are obtained by applying a mapping transformation that is isotonic (monotonically increasing), known as the pair-adjacent violators algorithm (PAV) [ABE+55]. The first step in this algorithm is to order the n elements decreasingly according to estimated probability and to initialise $p^*(i) = f(i)$. The idea is that calibrated probability estimates must be a decreasing sequence, i.e., $p^*(i_1) \geq p^*(i_2) \geq \ldots \geq p^*(i_n)$. If this is not the case, for each pair of consecutive probabilities, $p^*(i)$ and

$p^*(i+1)$, such that $p^*(i) < p^*(i+1)$, the PAV algorithm replaces both of them by their probability average, that is:

$$a \leftarrow \frac{p^*(i) + p^*(i+1)}{2},\ p^*(i) \leftarrow a,\ p^*(i+1) \leftarrow a$$

This process is repeated (using the new values) until an isotonic set is reached.

3.1.2 Evaluation measures

Classifiers can be evaluated according to several performance metrics. These can be classified into three groups [FHOM09]: measures that account for a qualitative notion of error (such as accuracy or the mean F-measure/F-score), metrics based on how well the model ranks the examples (such as the Area Under the ROC Curve (AUC)) and, finally, measures based on a probabilistic understanding of error (such as mean absolute error, mean squared error (Brier score), LogLoss and some calibration measures).

- *Accuracy* is the best-known evaluation metric for classification and is defined as the percentage of correct predictions. However, accuracy is very sensitive to class imbalance. In addition, when the classifier is not crisp, accuracy depends on the choice of a threshold. Hence, a good classifier with good probability estimations can have low accuracy results if the threshold that separates the classes is not chosen properly.

- Of the family of measures that evaluate ranking quality, the most representative one is the *Area Under the ROC Curve (AUC)*, which is defined as the probability that given one positive example and one negative example at random, the classifier ranks the positive example above the negative one (the Mann-Whitney-Wilcoxon statistic [FBF$^+$03]). AUC is clearly a measure of separability since the lower the number of misranked pairs, the better separated the classes are. Although ROC analysis is difficult to extend to more than two classes ([FHoS03]), the AUC has been extended to multiclass problems effectively by approximations. In this thesis, we will use Hand & Till's extension [HT01], which is based on an aggregation of each class against each other, by using a uniform class distribution.

- Of the last family of measures, *Mean Squared Error (MSE)* or *Brier Score* [Bri50] penalises strong deviations from the true probability:

$$MSE = \frac{\sum_{j=1}^{c}\sum_{i=1}^{n}(f(i,j) - p(i,j))^2}{n \cdot c}$$

Although MSE was not a calibration measure originally, it was decomposed by Murphy [Mur72] in terms of calibration loss and refinement loss.

- A calibration measure based on overlapping binning is *CalBin* [CNM04]. This is defined as follows. For each class, we must order all cases by predicted $p(i,j)$, giving new indices i^*. Take the 100 first elements (i^* from 1 to 100) as the first bin. Calculate the percentage of cases of class j in this bin as the actual probability, $\hat{f}_j$. The error for this bin is:

$$\sum_{i^* \in 1..100} |p(i^*,j) - \hat{f}_j|$$

Take the second bin with elements (2 to 101) and compute the error in the same way. At the end, average the errors. The problem of using 100 (as [CNM04] suggests) is that it might be a much too large bin for small datasets. Instead of 100 we set a different bin length, $s = n/10$, to make it more size-independent.

3.1.3 Monotonicity and multiclass extensions

The three calibration methods described in Section 3.1.1 are monotonic; they do not change the rank (order) of the examples according to each class estimated probability. In fact, Platt's method is the only one that is strictly monotonic, i.e, if $p(i_1) > p(i_2)$, then $p^*(i_1) > p^*(i_2)$, implying that AUC and refinement loss are not affected (only calibration loss is affected). In the other two methods, ties are generally created (i.e, $p^*(i_1) = p^*(i_2)$ for some examples i_1 and i_2 where $p(i_1) > p(i_2)$). This means that refinement is typically reduced for the binning averaging and the PAV methods.

Monotonicity will play a crucial role in understanding what calibration does before classifier combination. Nevertheless, the extensions of binary monotonic calibration methods to multiclass calibration (*one-against-all* and *all-against-all* schemes) do not ensure monotonicity (more details can be consulted in the paper "On the effect of calibration in classifier combination", Section 7.8). This will motivate the analysis of the non-monotonic SBA method. Based on the concept of monotonicity, we propose a taxonomy of calibration

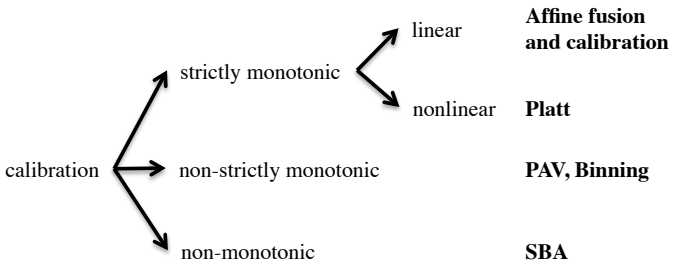


Figure 3.1: Taxonomy of calibration methods in terms of monotonicity (strictly monotonic, non-strictly monotonic, or non-monotonic methods), linearity (linear or nonlinear methods).

methods (Figure 3.1) including classical calibration methods (PAV, Binning and Platt), the SBA method and the Brümmer's affine fusion and calibration methods [Brü10].

Another important issue is whether the calibration methods are binary or multiclass. The three methods presented in Section 3.1.1 were specifically designed for two-class problems. For the multiclass case, Zadrozny and Elkan [ZE02] proposed an approach that consists in reducing the multiclass problem into a number of binary problems. A classifier is learned for each binary problem and, then, its predictions are calibrated.

Some works have compared the *one-against-all* and the *all-against-all* schemes, concluding in [RK04] that the *one-against-all* scheme performs as well as the *all-against-all* schemes. Therefore, we use the *one-against-all* approach for our experimental work because its implementation is simpler.

3.2 Calibration by multivariate Similarity-Binning Averaging

As we have shown in the Section 3.1.1, most calibration methods are based on a univariate transformation function over the original estimated class probability. In binning averaging, isotonic regression or Platt's method, this function is just obtained through very particular mapping methods, using $p(i, j)$ (the estimated probability) as the only input variable. Leaving out the rest of information of each instance (e.g., their original attributes) is a great waste of information which would be useful for the calibration process. For instance, in the case of binning-based methods, the bins are exclusively constructed by

sorting the estimated probability of the elements. Binning can be modified in such a way that bins overlap or bins move as windows, but it still only depends on one variable (the estimated probability).

The core of our approach is to change the idea of "sorting" for creating bins, into the idea of using similarity to create bins which are specific for each instance. The rationale for this idea is as follows. If bins are created by using only the estimated probability, calibrated probabilities will be computed from possibly different examples with similar probabilities. The effect of calibration is small, since we average similar probabilities. On the contrary, if we construct the bins using similar examples according to other features, probabilities can be more diverse and calibration will have more effect. Additionally, it will be sensitive to strong probability deviation given by small changes in one or more original features. This means that if noise on a variable dramatically affects the output, probabilities will be smoothed and, hence, they will be more noise-tolerant. For instance, if $(3, 2, a)$ has class $true$ and $(2, 2, a)$ has class $false$, the estimated probability for $(3, 2, a)$ should not be too close to 1.

Based on this reasoning, we propose a new calibration method that we called *Similarity-Binning Averaging (SBA)*. In this method the original attributes and the estimated probability are used to calculate the calibrated one.

The method is composed of two stages. The left side of Figure 3.2 shows *Stage 1*. In this stage, a given model M gives the estimated probabilities associated with a dataset. This dataset can be the same one used for training, or an additional validation dataset VD. The estimated probabilities $p(i, j)$ $1 \leq j \leq c$ are added (as new attributes) to each instance i of VD, creating a new dataset VDP.

The right side of Figure 3.2 shows *Stage 2* of the SBA method. To calibrate a new instance I, first, the estimated probability for each class is obtained from the classification model M, and these probabilities (one for each class) are added to the instance, thus creating a new instance (IP). Next, the k-most similar instances to this new instance are selected from the dataset VDP (for example, using the k-NN algorithm). This creates a bin. Finally, the calibrated probability of I for each class j is the average predicted class probability of this bin (i.e., the probability estimated by the k-NN algorithm for each class j of the instance I).

Let us see how the SBA algorithm works with an example. Consider that we learn a decision tree from the *Iris* dataset. In "Stage 1", we add the estimated probability for each class (Iris-setosa, Iris-versicolor and Iris-virginica) to each instance creating the VDP dataset:

In "Stage 2", we have a new instance I for which we want to obtain its

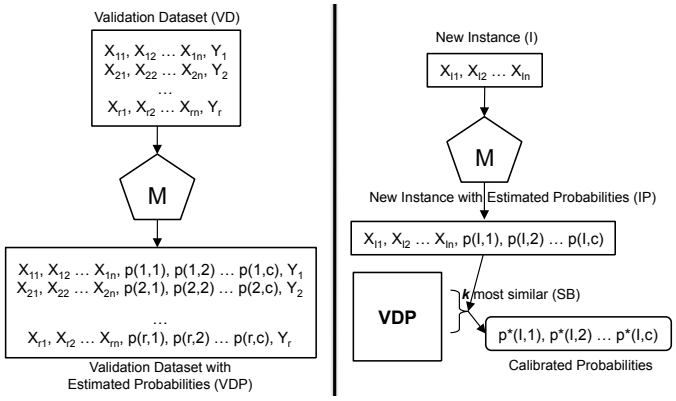


Figure 3.2: **Left:** Stage 1 of the SBA method. **Right:** Stage 2 of the SBA method.

inst#	sepall.	sepalw.	petall.	petalw.	p(setosa)	p(versicolor)	p(virginica)	class
1	5.1	3.5	1.4	0.2	0.962	0.019	0.019	setosa
2	4.9	3.0	1.4	0.2	0.962	0.019	0.019	setosa
...	...	...	...	...	...	...	...	...
150	5.9	3.0	5.1	1.8	0.02	0.041	0.939	virginica

calibrated probabilities. First, we obtain the estimated probability for each class of the instance from the decision tree. This is IP:

sepall.	sepalw.	petall.	petalw.	p(setosa)	p(versicolor)	p(virginica)
7.0	3.2	4.7	1.4	0.02	0.941	0.039

Next, a k-NN algorithm is learned from the VDP dataset, and the instance IP is classified obtaining a probability for each class, as the proportion of examples of each class from the k nearest neighbours. These are the calibrated probabilities of the instance I by the SBA method.

p(setosa)	p(versicolor)	p(virginica)
0.001	0.998	0.001

3.3 The relation between calibration and combination

One of the most common methods of classifier combination is *Bayesian Model Averaging* [HMRV99]. It consists in weighting each classifier, giving more credit to more reliable sources. However, this rationale does not necessarily entail the best combination ([Kun02][Kun04]). An alternative (and generalised) option is the weighted average combination [Kun04], using probabilities.

The estimated probability of an item i belonging to class j given by a weighted average combination of L classifiers is:

$$\tilde{p}(i,j) = \sum_{k=1}^{L} w_k p_k(i,j) \tag{3.1}$$

where $\sum_{k=1}^{L} w_k = 1$.

Formula (3.1) defines a linear combination of the classifier outputs and can be instantiated to more specific schemas depending on how w_k and p_k are chosen. In general, the use of a performance (or overall reliability) weight per classifier w_k is justified because some classifiers are more reliable than others. However, a proper calibration would give each prediction its proper weight depending on the reliability of $p_k(i,j)$ (high reliability for $p_k(i,j)$ closer to 0 and 1, and lower reliability for $p_k(i,j)$ closer to 0.5 or to the class proportion for imbalanced problems). The use of both w_k and p_k may lead to a *double weighting*, where the importance of a classifier prediction depends on the classifier quality but also on the quality (or calibration) of the specific probability estimation.

In order to better understand the relation between weights and probabilities, we firstly need to understand the meaning of the weights. There are many ways of calculating weights. A very common option is to estimate the accuracy on a validation dataset D, followed by a normalisation [Kun04], i.e., if acc_k is the accuracy of model l_k on D, then:

$$w_k = \frac{acc_k}{\sum_{m=1}^{L} acc_m}$$

If we use AUC (or MSE) as a measure, the question of whether a *double weighting* is going to be too drastic depends on how the weights are derived from these measures. For instance, a weight equal to the AUC is an option, but since AUC=0.5 means random behaviour, perhaps the GINI index (which equals $(AUC - 0.5) \times 2$) would be a better option, since it goes from 0 to 1. In

the same way, using the MSE, the expression $(1 - MSE)$ is a natural option, but a more extreme $1/MSE$ could also be considered.

Consequently, there are many open questions when mixing together probabilities and weighted combinations. Are both things redundant or even incompatible? Is calibration a good idea to get better probability estimations? If calibration is used, would weights become useless?

3.4 Results

In the paper "Similarity-Binning Averaging: A Generalisation of Binning Calibration" [BFHORQ09], which can be consulted in Section 7.2, we present the SBA calibration method in detail. We compare this method with three well-known and baseline calibration techniques: non-overlapping binning averaging, Platt's method and PAV. The experimental results show a significant increase in calibration for the two calibration measures considered (MSE and CalBin). It is true that this better calibration is partly obtained because of the increase of AUC. It is, consequently, non-monotonic. In many applications where calibration is necessary the restriction of being monotonic is not only applicable, but it is an inconvenience. In fact, when calibrating a model, the original model and class assignments can be preserved, while only modifying the probabilities. In other words, the predictions of a comprehensible model composed of a dozen rules can be annotated by the estimated probabilities while preserving the comprehensibility of the original model.

Table 3.1: Different methods to calculate weights.

Method	**Definition**
WCUnif	$w_k = \frac{1}{L}$
WCAcc	$w_k = \frac{acc_k}{\sum_{m=1}^{L} acc_m}$
WCAUC	$w_k = \frac{AUC_k}{\sum_{m=1}^{L} AUC_m}$
WCMSE	$w_k = \frac{(1-MSE_k)}{\sum_{m=1}^{L}(1-MSE_m)}$
WCGINI	$w_k = \frac{max(0,(AUC_k-0.5)\times 2)}{\sum_{m=1}^{L} max(0,(AUC_m-0.5)\times 2)}$
WCIMSE	$w_k = \frac{(1/MSE_k)}{\sum_{m=1}^{L}(1/MSE_m)}$

In the paper "On the Effect of Calibration in Classifier Combination" [BFHORQ], that can be consulted in Section 7.8, we analyse theoretically and

Table 3.2: Experimental layouts that arrange combination and calibration. *CombMet* is the combination method and *CalMet* is the calibration method.

Layout	Description and Variants
BaseModel	*BaseModel* ∈ {J48, Logistic, NB, IBk} plus a random model.
Base	The average of all the base models
CombMet	*CombMet* ∈ {WCUnif, WCAcc, WCAUC, WCGINI, WCMSE, WCIMSE}.
CalMet	*CalMet* ∈ { PAV, Platt, Binning, SBA}.
CalMet + CombMet	For different calibration and combination methods.
CombMet + CalMet	For different calibration and combination methods.
CalMet + CombMet + CalMet	For different calibration and combination methods.

experimentally the effect of four calibration methods (Binning, PAV, Platt and SBA) in the weighted average combination of classifiers, with six different methods to calculate the weights (see Table 3.1); four evaluation measures: MSE, CalBin, AUC and accuracy; and seven different layouts that can be seen in Table 3.2. From this analysis, we would like to highlight some clear messages, as follows:

- Calibration is beneficial before combination as the experimental results show, *in general*.
- The combination of classifiers does not typically give a calibrated result, as we have studied by analysing the probability distributions using truncated normal models for them. This has been confirmed by the experimental results.
- We advocate for AUC as the right measure to evaluate combination performance, precisely because the combination is generally uncalibrated.
- We recommend calibration after combination, if we are interested in good results in terms of MSE or in terms of accuracy.
- Weighted combination is compatible with probabilities even when we use calibration with the same dataset from which we derive the weights. This has been shown by the experiments. Therefore, the *double weighting* (weights and probabilities) does not show up, or at least it is counteracted by other benefits.
- The weighting methods which are best when using probabilities are GINI and IMSE, even in conjunction with calibration.

- Monotonic calibration methods are good for combination, but their influence is limited.
- SBA, the non-monotonic calibration method, is better for combination according to the experimental results.

Therefore, to summarise this chapter, we conclude that, in order to obtain good global results, it is important to calibrate the probabilities when several local models are combined. Non-monotonic methods, such as SBA, can unleash all the potential of calibration and, additionally can be directly applied to multiclass problems, for which all the other monotonic methods lose their monotonicity.

4

Quantification using Estimated Probabilities

In Chapter 1, we saw how quantification can help to make global decisions, during the model deployment phase, before making individual decisions for each problem instance. For instance, an organisation may know, by quantification methods, how many products it is going to sell, before knowing exactly which customers will buy its products. This information (quantification) is important to the organisation, because depending on this result human and economical resources can be better arranged, or the organisation might even cancel the campaign, if the estimated number of products to be sold is not profitable for the company.

Quantification is the name given to this novel machine learning task which deals with correctly estimating the number of elements of one class in a set of examples. Hence, the output of a quantifier is a real value. George Forman [For05][For06][For08] introduced and systematised this new supervised machine learning task. Since in quantification training instances are the same as in a classification problem, a natural approach is to train a classifier and to derive a quantifier from it. Forman's works have shown that just classifying the instances and counting the examples belonging to the class of interest (*classify & count*) typically yields bad quantifiers, especially when the class distribution may vary between training and test. Hence, adjusted versions of *classify & count* have been developed by using modified thresholds.

However, these works have explicitly discarded (without a deep analysis) any possible approach based on the probability estimations of the classifier. In this chapter, we present a method based on averaging the probability estimations of a classifier with a very simple scaling that does perform reasonably well, showing that probability estimators for quantification capture a richer view of the problem than methods based on a threshold. Moreover, since the

new quantification method that we propose is based on probability estimators, we are interested in studying the impact that calibrated probabilities may have for quantification.

4.1 Notation and previous work

Given a dataset T, n denotes the number of examples, and c the number of classes. We will use i to index or refer to examples, so we will express $i = 1 \dots n$ or $i \in T$ indistinctly. $f(i,j)$ represents the actual probability of example i to be of class j. We assume that $f(i,j)$ always takes values in {0,1} and is not strictly a probability but a single-label indicator function. With $n_j = \sum_{i=1}^{n} f(i,j)$, we denote the number of examples of class j. $\pi(j)$[1] denotes the prior probability of class j, i.e., $\pi(j) = n_j/n$. When referring to a particular dataset T, we will use the equivalent expression:

$$\pi_T(j) = \frac{\sum_{i \in T} f(i,j)}{|T|}$$

Given a classifier, $p(i,j)$ represents the estimated probability of example i to be of class j taking values in [0,1]. $\hat{\pi}(j)$ denotes the estimated probability of class j which is defined as:

$$\hat{\pi}_T(j) = \frac{\sum_{i \in T} p(i,j)}{|T|}$$

when referring to a dataset T. For the sake of readability, when $c = 2$, we will use the symbols $\oplus$ for the positive class and $\ominus$ for the negative class. Since the probabilities are complementary for two classes, we will focus on the positive class. $C_\theta(i,j)$ is 1 iff j is the predicted class obtained from $p(i,j)$ using a threshold θ. We can omit θ when it is embedded in the classifier or clear from the context. When $c = 2$, we will use the following measures:

$$TP = \sum_{i=1}^{n} f(i,\oplus)C(i,\oplus), \;\; TN = \sum_{i=1}^{n} f(i,\ominus)C(i,\ominus),$$

$$FP = \sum_{i=1}^{n} f(i,\ominus)C(i,\oplus), \;\; FN = \sum_{i=1}^{n} f(i,\oplus)C(i,\ominus);$$

we will also use the ratios:

[1] Note that in this chapter we have slightly changed the notation from the previous chapter, following the notation used in each paper.

$$tpr = TP/(TP + FN) \ and \ fpr = FP/(FP + TN).$$

We will use *pos* for the actual proportion of positives, i.e., $\pi(\oplus)$; and we will use *neg* for the actual proportion of negatives, i.e., $\pi(\ominus)$. Finally, the function $clip(X, a, b)$ truncates a real value X inside an interval $[a, b]$. We represent the elements of T of class $\oplus$ and $\ominus$ with $T_{\oplus}$ and $T_{\ominus}$, respectively.

Forman introduced several quantification methods that we arrange into the following three groups:

- Methods based on counting the positive predicted examples. The *classify & count* (CC) and the *adjusted count* (AC) methods belong to this group.
- Methods based on selecting a classifier threshold, but in this case, the threshold is determined from the relationship between tpr and fpr in order to provide better quantifier estimates. For instance, some methods choose one particular threshold, such as: the *X method*, which selects the threshold that satisfies $fpr = 1 - tpr$; the *Max method*, which selects the threshold that maximises the difference $tpr - fpr$; or those methods like $T50$ that select a particular rate between tpr and fpr. The *Median Sweep (MS) method*[2] is another method that tests all the thresholds in the test set, estimates the number of positives in each one, and returns a mean or median of these estimations.
- Methods based on a mixture of distributions. The *Mixture Model* (MM) [For05] is included in this group. It consists of determining the distributions from the classifier scores on the training positive ($D_{\oplus}$) and negative examples ($D_{\ominus}$) and then modelling the observed distribution D on the test set as a mixture of $D_{\oplus}$ and $D_{\ominus}$.

The best results are obtained with AC in general; however, when the training sets are small and the number of positives is small, other methods such as $T50$ or MS can get better results (at the cost of performing worse in other more balanced situations).

4.2 Quantification evaluation

Quantification outputs a real value and hence can be evaluated by the classical error measures for continuous variables, the Mean Absolute Error (MAE) and

[2]It is proposed when the tpr and fpr are estimated from cross-validation.

the Mean Squared Error (MSE). Forman only used the absolute error in his experiments, but we consider that the MSE measure is a better way to quantify differences between estimations for real values. Let us formalise these measures in order to better establish the quantification problem and goal.

Consider that we have a method that estimates the proportion of elements for each class ($\hat{\pi}_T(j)$). By calculating the absolute difference of these two values, we have the global MAE for each class:

$$GMAE_j(T) = |\pi_T(j) - \hat{\pi}_T(j)|\,,$$

and for all the classes we have:

$$GMAE(T) = \frac{1}{c} \cdot \sum_{j=1..c} GMAE_j(T)$$

Similarly, we calculate:

$$GMSE_j(T) = (\pi_T(j) - \hat{\pi}_T(j))^2 \;\; and$$

$$GMSE(T) = \frac{1}{c} \cdot \sum_{j=1..c} GMSE_j(T)$$

For binary problems, we have that:

$$GMAE_{\oplus} = GMAE_{\ominus} = GMAE \;\; and$$

$$GMSE_{\oplus} = GMSE_{\ominus} = GMSE$$

Therefore, for binary problems, we will only evaluate the error for the proportion of positives.

4.3 Quantifying by Scaled Averaged Probabilities

The idea of using an average of the probability estimations is supported by the issue that probabilities represent much richer information than just the decisions. After this rationale, the use of probabilities shapes a family of methods that we call *probability estimation & average*. The simplest method in this family is called *Probability Average* (PA). First, a probabilistic classifier is learned from the training data, such as a Probability Estimation Tree or a Naïve Bayes model. Then, the learned model is applied to the instances in the test set, obtaining a probability estimation for each one. Finally, the

average of the estimated probabilities for each class is calculated. Although this definition is multiclass, for the rest of the explanation we will concentrate on binary datasets. In this case, we only need to care about one class (the positive class), and the method is defined as follows:

$$\hat{\pi}^{PA}_{Test}(\oplus) = \frac{\sum_{i \in Test} p(i, \oplus)}{|Test|} \tag{4.1}$$

Logically, if the proportion of positive examples in the training set is different from the proportion of positive examples in the test set, the result will not be satisfactory in general. The solution comes precisely from the analysis of the extreme case when all the elements in the test set are of one class. In this case, we will get the average probability for the positive cases alone, which can only be 1 for a perfect classifier (which is not frequently the case). As in the *AC* method, the idea is to use a proper scaling.

Nevertheless, from the training set, it is possible to calculate:

- the *actual proportion of positive examples* ($\pi_{Train}(\oplus)$),
- the *positive probability average* ($\hat{\pi}_{Train}(\oplus)$),
- the *positive probability average for the positives* ($\hat{\pi}_{Train_\oplus}(\oplus)$),
- and the *positive probability average for the negatives* ($\hat{\pi}_{Train_\ominus}(\oplus)$).

From the definitions, it is easy to check the following equality:

$$\hat{\pi}_{Train_\oplus}(\oplus) \cdot \pi_{Train}(\oplus) + \hat{\pi}_{Train_\ominus}(\oplus) \cdot (1 - \pi_{Train}(\oplus)) = \hat{\pi}_{Train}(\oplus) \tag{4.2}$$

From which the following equation is derived:

$$\pi_{Train}(\oplus) = \frac{\hat{\pi}_{Train}(\oplus) - \hat{\pi}_{Train_\ominus}(\oplus)}{\hat{\pi}_{Train_\oplus}(\oplus) - \hat{\pi}_{Train_\ominus}(\oplus)}, \tag{4.3}$$

which yields a probabilistic version of Forman's adjustment (see Fig.4.1). When all instances are positive, $\hat{\pi}_{Train_\oplus}(\oplus)$ sets the maximum, and we scale this to 1. When all instances are negative, $\hat{\pi}_{Train_\ominus}(\oplus)$ sets the minimum, and we scale this to 0.

Thus, we propose a new quantification method, which we call *Scaled Probability Average* (SPA), applying Formula 4.3 in the same way as Forman to the value obtained with the PA method (Formula 4.1), i.e.,

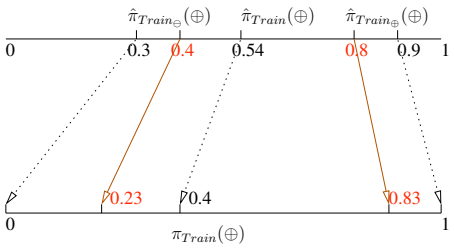


Figure 4.1: Scaling used in the SPA method. The limits in the training set are placed at 0.3 and 0.9. The estimated value for the training set is 0.54 whereas the actual proportion in the training set is 0.4. The scaling would move a case at 0.4 to 0.23 and a case at 0.8 to 0.83.

$$\hat{\pi}^{SPA}_{Test}(\oplus) = \frac{\hat{\pi}^{PA}_{Test}(\oplus) - \hat{\pi}_{Train_\ominus}(\oplus)}{\hat{\pi}_{Train_\oplus}(\oplus) - \hat{\pi}_{Train_\ominus}(\oplus)} \tag{4.4}$$

In the same way as in the SPA method, the proportion of positive examples estimated by the PA method is scaled. This scaling can also be applied to the proportion of positive examples estimated by the CC method. Therefore, we propose the *Scaled Classify & Count* (SCC) method:

$$\hat{\pi}^{SCC}_{Test}(\oplus) = \frac{\frac{\sum_{i \in Test} C(i,\oplus)}{|Test|} - \hat{\pi}_{Train_\ominus}(\oplus)}{\hat{\pi}_{Train_\oplus}(\oplus) - \hat{\pi}_{Train_\ominus}(\oplus)} \tag{4.5}$$

4.4 Quantification using calibrated probabilities

The new quantification methods that we have proposed in the previous section are based on averaging the probability estimations of classifiers. This suggests to study the impact of having good probability estimators for quantification. [For05][For06][For08] discusses that some bad classifiers can be good quantifiers, but also (and frequently) that some good classifiers can be very bad quantifiers. The key issue to understand this when we use *classify & count* and related methods is how the threshold is chosen. The key issue to understand this when we use *average probability* methods is calibration [Bri50][DF83][Pla99][ZE01][BFHORQ10a]. A random, but well-calibrated classifier can give very good results with the SPA method, when training and test class distributions match. On the contrary, a very good classifier, with high separability (e.g., a high AUC) can have a very bad calibration and give

very bad quantification results with the SPA method. But there are some other, more surprising cases. Consider the worst classifier possible, which in the binary case outputs $p(i, \oplus) = 1$ for every example i of class $\ominus$ and outputs $p(i, \oplus) = 0$ for every example i of class $\oplus$. If the dataset is balanced, quantification will be perfect. But calibration in this case is extremely poor. As a result, calibration plays a very important role for quantification using probabilities, but the phenomenon is more complex than it seems at first sight, even in the easy cases where training and test class distributions are equal.

The first intuition from the previous discussion is that we do not need to have good calibration for every single prediction to have an overall good quantification result. This suggests the distinction of what we call local (traditional) calibration and global calibration.

In Section 7.6, we formalise the notions of local and global calibration over a dataset. Global calibration is key to the success of quantification methods using probabilities. However, the relation of local to global calibration suggests that when the class distribution is the same (and especially when the dataset is imbalanced), having good local calibration implies having good global calibration (at least it sets some bounds on the error), but when class distribution changes dramatically, it is not so clear that a very good local calibration ensures good global calibration. To answer the question on whether better local calibration (alone) can make quantification improve, we experimentally analyse several classical (local) calibration methods and their effect over quantification methods.

4.5 Results

The results of these new quantification methods based on probability estimators have been published in the paper "Quantification via Probability Estimators" [BFHORQ10c] that can be consulted in Section 7.5.

The experiments show that the behaviour of the SPA method is more robust than the methods AC and T50. These methods are designed for situations where the proportion of classes is imbalanced. Even when the experimental setting is more favourable for the AC and T50 methods, the SPA method still obtains good results.

Therefore, we can conclude that the results are highly positive and show that the use of probability estimators for quantification is a good pathway to pursue, which can lead to new methods in the future.

In the paper "Local and Global Calibration. Quantification using Calibrated Probabilities" [BFHORQ10b], which can be consulted in Section 7.6,

we show a preliminary study of the effect of calibration methods for quantification. We obtain, at first sight, that (local) calibration does not improve the quantification methods. However, there are still some cases where the results for the quantification methods based on probabilities (PA and SPA) are better when the probabilities are calibrated. This is a first step that encourages the study of the effect of calibration in quantification in more depth.

5

Conclusions and Future Work

5.1 Conclusions

Machine learning and data mining techniques are frequently used by organisations to make decisions. Typically, these techniques have been focused on obtaining the best results in an idealistic environment (isolated models, without constraints, etc.). However, when these models are applied in a real environment, they need to be related to other models, with constraints that have to be fulfilled. However, some other integration techniques need to be used, in the model deployment phase, to obtain good global results. In this dissertation we have seen that it is important to have a global view of the problem, instead of only focusing on improving single results, as usually one cannot see the forest for the trees. We have presented new methods and techniques that belong to this group of deployment techniques.

First, we have introduced new techniques that combine local models in order to obtain good global decisions which meet a set of constraints. Concretely, we have developed new techniques in a general prescription problem scenario. Here, the seller has to offer several products to several customers, trying to obtain the maximum profit possible. In this scenario, one data mining model is learned for each product (or customer) and the new combination techniques are used to obtain good global results.

Second, we have studied the relation between input and output features, the problem inversion approach and the use of input features for global optimisation and negotiation. The key point of this study has been the new concept of negotiable feature. It has opened a new way of looking at a prescription problem. Concretely, we have proposed a new taxonomy of prescription problems depending on whether the problem has or not negotiable features. Using the problem inversion approach we have transformed a classification problem into a regression problem, learning a model for each product and customer

and obtaining a probability of buying the product for each value of the negotiable feature. We have developed multi-bid negotiation strategies that obtain good global results combining these individual models. From our studies we have concluded that techniques and negotiation strategies based on simulation (Joint Simulation Approach and Maximum Global Optimisation) obtain better results than other analytical techniques.

Third, we have developed more powerful calibration techniques, and we have studied their relation to classifier combination. Concretely, we have implemented a new multivariate, non-monotonic and multiclass calibration method (Similarity Binning Averaging). It is different to the classical calibration methods, which are univariate and monotonic, and has outperformed their results. Also, the experimental results have shown that this new calibration method improves the overall results, in terms of AUC, in classifier combination.

Fourth, we have presented methods that make a global decision from the sum of individual decisions. Concretely, we have developed a new quantification method based on probabilities. It has outperformed other quantification methods based on thresholds. Moreover, we have analysed the impact of having good probability estimators for the new quantification methods based on probability average estimation, and the relation of quantification with global calibration.

Overall, in this dissertation, we have developed techniques that can be applied in the model deployment phase, in order to obtain good global results when several local models are combined. However, on the one hand, there are new related issues that have appeared during this research and have not already been studied, so we will consider them as future work in the next section. On the other hand, as a more ambitious work, new techniques could be studied in order to obtain a global optimisation framework and view for all the data mining processes in an organisation.

5.2 Future work

As future work, we want to study the extension of the negotiation process to multiple negotiable features. When we have only one negotiable feature we have two dimensions (negotiable feature and probability). In the case of multiple negotiable features we have one dimension for each negotiable feature plus one (probability).

We also want to study quantification methods for regression problems. For instance, a bank may offer a new investment product. Given a regression model

learned from a subsample or a previous customer sample, we want to know the total amount of money that all the customers will invest, globally. Again, we may not require the particular amount of money for each particular customer, but the total amount. This is the case for quantification in regression, where it is not class probabilities which matters but prediction distributions. Also, another topic we want to follow, as we mentioned at the end of Chapter 3, is the effect of calibration in quantification methods based on probabilities.

In this dissertation we have studied global optimisation techniques that can be applied during the deployment phase. However, if the need for integrating several data mining models is known beforehand, the optimisation process would start in early phases of the KDD process. For example, in a hospital a model may predict the number of admissions in the emergency department and another model may predict the bed occupation. These two models are related to each other and other models are also related to them. All these models could be combined from the beginning, possibly using simulation strategies in order to integrate (connect) the models and fulfilling the problem constraints. In this way, we could have a global view of all the processes in an organisation. Changes in one model would influence automatically on the other models, always having a global optimisation of all the events in mind.

In the end, this work has emphasised that better *model integration* techniques must be based on optimising *global decisions* over *local* ones.

Bibliography

[ABE+55] M. Ayer, H.D. Brunk, G.M. Ewing, W.T. Reid, and E. Silverman. An empirical distribution function for sampling with incomplete information. *Annals of Mathematical Statistics*, 5:641–647, 1955.

[AT05] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. on Knowl. and Data Eng.*, 17(6):734–749, 2005.

[BFHORQ] A. Bella, C. Ferri, J. Hernández-Orallo, and M.J. Ramírez-Quintana. On the Effect of Calibration in Classifier Combination. (Submitted to Applied Intelligence).

[BFHORQ09] A. Bella, C. Ferri, J. Hernández-Orallo, and M.J. Ramírez-Quintana. Similarity-Binning Averaging: A Generalisation of Binning Calibration. In *Intelligent Data Engineering and Automated Learning - IDEAL 2009*, volume 5788 of *Lecture Notes in Computer Science*, pages 341–349. Springer Berlin / Heidelberg, 2009.

[BFHORQ10a] A. Bella, C. Ferri, J. Hernández-Orallo, and M.J. Ramírez-Quintana. Calibration of Machine Learning Models. In *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques*, pages 128–146. IGI Global, 2010.

[BFHORQ10b] A. Bella, C. Ferri, J. Hernández-Orallo, and M.J. Ramírez-Quintana. Local and Global Calibration. Quantification using Calibrated Probabilities. Technical report, DSIC-ELP. Universitat Politècnica de València, 2010.

[BFHORQ10c] A. Bella, C. Ferri, J. Hernández-Orallo, and M.J. Ramírez-Quintana. Quantification via Probability Estimators. *IEEE International Conference on Data Mining*, 0:737–742, 2010.

[BFHORQ11] A. Bella, C. Ferri, J. Hernández-Orallo, and M.J. Ramírez-Quintana. Using Negotiable Features for Prescription Problems. *Computing*, 91:135–168, 2011.

[BGL07] M. Better, F. Glover, and M. Laguna. Advances in analytics: integrating dynamic data mining with simulation optimization. *IBM J. Res. Dev.*, 51(3):477–487, 2007.

[BL99] M.J.A. Berry and G.S. Linoff. *Mastering Data Mining: The Art and Science of Customer Relationship Management.* Wiley, 1999.

[Bre96] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[Bri50] G.W. Brier. Verification of forecasts expressed in terms of probabilities. *Monthly Weather Review*, 78:1–3, 1950.

[Brü10] N. Brümmer. *Measuring, refining and calibrating speaker and language information extracted from speech.* PhD thesis, University of Stellenbosch, 2010.

[BST00] A. Berson, S. Smith, and K. Thearling. *Building Data Mining Applications for CRM.* McGraw Hill, 2000.

[CNM04] R. Caruana and A. Niculescu-Mizil. Data mining in metric space: an empirical analysis of supervised learning performance criteria. In *KDD*, pages 69–78, 2004.

[DF83] M.H. DeGroot and S.E. Fienberg. The comparison and evaluation of forecasters. *The statistician*, pages 12–22, 1983.

[Die00a] T.G. Dietterich. Ensemble methods in machine learning. In *First International Workshop on Multiple Classifier Systems*, pages 1–15, 2000.

[Die00b] T.G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, Boosting, and Randomization. *Machine Learning*, 40(2):139–157, 2000.

[FBF+03] P. Flach, H. Blockeel, C. Ferri, J. Hernández-Orallo, and J. Struyf. Decision support for data mining: An introduction to ROC analysis and its applications. In *Data Mining*

and Decision Support: Integration and Collaboration, pages 81–90. Kluwer Academic Publishers, Boston, 2003.

[FFHO04] C. Ferri, P.A. Flach, and J. Hernández-Orallo. Delegating classifiers. In *Proc. of the 21st International Conference on Machine learning*, page 37, New York, 2004. ACM.

[FHOM09] C. Ferri, J. Hernández-Orallo, and R. Modroiu. An experimental comparison of performance measures for classification. *Pattern Recogn. Lett.*, 30(1):27–38, 2009.

[FHoS03] C. Ferri, J. Hernández-orallo, and M.A. Salido. Volume under the roc surface for multi-class problems. Exact computation and evaluation of approximations. In *Proceedings of 14th European Conference on Machine Learning*, pages 108–120, 2003.

[FKM+04] R. Fagin, R. Kumar, M. Mahdian, D. Sivakumar, and E. Vee. Comparing and aggregating rankings with ties. In *PODS*, pages 47–58, 2004.

[For05] G. Forman. Counting positives accurately despite inaccurate classification. In *ECML*, pages 564–575, 2005.

[For06] G. Forman. Quantifying trends accurately despite classifier error and class imbalance. In *KDD*, pages 157–166, 2006.

[For08] G. Forman. Quantifying counts and costs via classification. *Data Min. Knowl. Discov.*, 17(2):164–206, 2008.

[FS96] Y. Freund and R.E. Schapire. Experiments with a new boosting algorithm. In *Proc. 13th International Conference on Machine Learning*, pages 148–146. Morgan Kaufmann, 1996.

[GB00] J. Gama and P. Brazdil. Cascade generalization. *Machine Learning*, 41(3):315–343, 2000.

[HMRV99] J.A. Hoeting, D. Madigan, A.E. Raftery, and C.T. Volinsky. Bayesian model averaging: A tutorial. *Statistical Science*, 14(4):382–417, 1999.

[HT01] D.J. Hand and R.J. Till. A simple generalisation of the area under the ROC curve for multiple class classification problems. *Machine Learning*, 45(2):171–186, 2001.

[Kun02] L.I. Kuncheva. A theoretical study on six classifier fusion strategies. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24:281–286, February 2002.

[Kun04] L.I. Kuncheva. *Combining Pattern Classifiers: Methods and Algorithms*. Wiley, 2004.

[KW02] L.I. Kuncheva and Christopher J. Whitaker. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. Submitted to Machine Learning, 2002.

[Mit97] T.M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[MU49] N. Metropolis and S. Ulam. The monte carlo method. *Journal of the American Statistical Association (American Statistical Association)*, 1949.

[Mur72] A.H. Murphy. Scalar and vector partitions of the probability score: Part ii. n-state situation. *Journal of Applied Meteorology*, 11:1182–1192, 1972.

[PdP05] A. Prinzie and D. Van den Poe. Constrained optimization of data-mining problems to improve model performance: A direct-marketing application. *Expert Systems with Applications*, 29(3):630–640, 2005.

[Pla99] J.C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in Large Margin Classifiers*, pages 61–74. MIT Press, 1999.

[PT03] B. Padmanabhan and A. Tuzhilin. On the use of optimization for data mining: Theoretical interactions and ecrm opportunities. *Management Science*, 49(10, Special Issue on E-Business and Management Science):1327–1343, 2003.

[RK04] R. Rifkin and A. Klautau. In defense of one-vs-all classification. *Journal of Machine Learning Research*, 5:101–141, December 2004.

[RWD88] T. Robertson, F.T. Wright, and R.L. Dykstra. *Order Restricted Statistical Inference*. John Wiley & Sons, 1988.

[TJGD08] S. Tulyakov, S. Jaeger, V. Govindaraju, and D. Doermann. Review of classifier combination methods. In *Machine Learning in Document Analysis and Recognition*, volume 90 of *Studies in Computational Intelligence*, pages 361–386. Springer Berlin / Heidelberg, 2008.

[Wol92] D.H. Wolpert. Stacked generalization. *Neural Networks*, 5(2):241–259, 1992.

[ZE01] B. Zadrozny and C. Elkan. Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers. In *ICML*, pages 609–616, 2001.

[ZE02] B. Zadrozny and C. Elkan. Transforming classifier scores into accurate multiclass probability estimates. In *Int. Conf. on Knowledge Discovery and Data Mining*, pages 694–699, 2002.

Part II

Publications Associated to this Thesis

6

List of Publications

1. A. Bella, C. Ferri, J. Hernández-Orallo, and M.J. Ramírez-Quintana. **Joint Cutoff Probabilistic Estimation using Simulation: A Mailing Campaign Application.** In *Intelligent Data Engineering and Automated Learning - IDEAL 2007*, volume 4881 of *Lecture Notes in Computer Science*, pages 609–619. Springer Berlin / Heidelberg, 2007.

   - Conference ranking: CORE C.

2. A. Bella, C. Ferri, J. Hernández-Orallo, and M.J. Ramírez-Quintana. **Similarity-Binning Averaging: A Generalisation of Binning Calibration.** In *Intelligent Data Engineering and Automated Learning - IDEAL 2009*, volume 5788 of *Lecture Notes in Computer Science*, pages 341–349. Springer Berlin / Heidelberg, 2009.

   - Conference ranking: CORE C.

3. A. Bella, C. Ferri, J. Hernández-Orallo, and M.J. Ramírez-Quintana. **Calibration of Machine Learning Models.** In *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques*, pages 128–146. IGI Global, 2010.

4. A. Bella, C. Ferri, J. Hernández-Orallo, and M.J. Ramírez-Quintana. **Data Mining Strategies for CRM Negotiation Prescription Problems.** In *Trends in Applied Intelligent Systems (IEA/AIE)*, volume 6096 of *Lecture Notes in Computer Science*, pages 520–530. Springer Berlin / Heidelberg, 2010.

   - Conference ranking: 46 among 701 conferences in the Computer Science Conference Ranking (Area: Artificial Intelligence / Machine Learning / Robotics / Human Computer Interaction), and CORE C.

5. A. Bella, C. Ferri, J. Hernández-Orallo, and M.J. Ramírez-Quintana. **Quantification via Probability Estimators.** *IEEE International Conference on Data Mining*, 0:737–742, 2010.

   - Conference ranking: CORE A+.

6. A. Bella, C. Ferri, J. Hernández-Orallo, and M.J. Ramírez-Quintana. **Local and Global Calibration. Quantification using Calibrated Probabilities.** Technical report, DSIC-ELP. Universitat Politècnica de València, 2010.

7. A. Bella, C. Ferri, J. Hernández-Orallo, and M.J. Ramírez-Quintana. **Using Negotiable Features for Prescription Problems.** *Computing*, 91:135–168, 2011.

   - Journal ranking: JCR 0.959 (50 of 97, Theory and Methods).

8. A. Bella, C. Ferri, J. Hernández-Orallo, and M.J. Ramírez-Quintana. **On the Effect of Calibration in Classifier Combination.** (Submitted to Applied Intelligence).

7

Publications (Full Text)

7.1 Joint Cutoff Probabilistic Estimation using Simulation: A Mailing Campaign Application

1. A. Bella, C. Ferri, J. Hernández-Orallo, and M.J. Ramírez-Quintana. Joint Cutoff Probabilistic Estimation using Simulation: A Mailing Campaign Application. In *Intelligent Data Engineering and Automated Learning - IDEAL 2007*, volume 4881 of *Lecture Notes in Computer Science*, pages 609–619. Springer Berlin / Heidelberg, 2007.

Joint Cutoff Probabilistic Estimation using Simulation: A Mailing Campaign Application*

Antonio Bella, Cèsar Ferri, José Hernández-Orallo, and María José Ramírez-Quintana

Universitat Politècnica de València, DSIC, Valencia, Spain

Abstract. Frequently, organisations have to face complex situations where decision making is difficult. In these scenarios, several related decisions must be made at a time, which are also bounded by constraints (e.g. inventory/stock limitations, costs, limited resources, time schedules, etc). In this paper, we present a new method to make a good global decision when we have such a complex environment with several local interwoven data mining models. In these situations, the best local cutoff for each model is not usually the best cutoff in global terms. We use simulation with Petri nets to obtain better cutoffs for the data mining models. We apply our approach to a frequent problem in customer relationship management (CRM), more specifically, a direct-marketing campaign design where several alternative products have to be offered to the same house list of customers and with usual inventory limitations. We experimentally compare two different methods to obtain the cutoff for the models (one based on merging the prospective customer lists and using the local cutoffs, and the other based on simulation), illustrating that methods which use simulation to adjust model cutoff obtain better results than a more classical analytical method.

1 Introduction

Data mining is becoming more and more useful and popular for decision making. Single decisions can be assisted by data mining models, which are previously learned from data. Data records previous decisions proved good or bad either by an expert or with time. This is the general picture for predictive data mining. The effort (both in research and industry) is then focussed on obtaining the best possible model given the data and the target task. In the end, if the model is accurate, the decisions based on the model will be accurate as well.

However, in real situations, organisations and individuals must make several decisions for several given problems. Frequently, these decisions/problems are interwoven with the rest, have to be made in a short period of time, and are accompanied with a series of constraints which are also just an estimation of the

* This work has been partially supported by the EU (FEDER) and the Spanish MEC under grant TIN 2007-68093-C02-02, Generalitat Valenciana under grant GV06/301, UPV under grant TAMAT and the Spanish project "Agreement Technologies" (Consolider Ingenio CSD2007-00022)

real constraints. In this typical scenario, making the best local decision for every problem does not give the best global result. This is well-known in engineering and decision making, but only recently acknowledged in data mining. Examples can be found everywhere: we cannot assign the best surgeon to each operation in a hospital, we cannot keep a fruit cargo until their optimal consumption point altogether, we cannot assign the best delivering date for each supplier, or we cannot use the best players for three matches in the same week.

In this context, some recent works have tried to find optimal global solutions where the local solutions given by local models are not good. These works address specific situations: rank aggregation [3] and cost-sensitive learning are examples of this, a more general "utility-based data mining"[1] also addresses this issue, but also some other new data mining tasks, such as quantification [5], are in this line. Data mining applied to CRM (Customer-Relationship Management) [1] is also one of the areas where several efforts have also been done.

Although all these approaches can be of great help in specific situations, most of the scenarios we face in real data mining applications do not fit many of the assumptions or settings of these previous works. In fact, many real scenarios are so complex that the "optimal" decision cannot be found analytically. Approximate, heuristic or simplified global models must be used instead. One appropriate non-analytic way to find good solutions to complex problems where many decisions have to be made is through simulation.

In this work, we connect inputs and outputs of several data mining models and simulate the global outcome under different possibilities. Through the power of repeating simulations after simulations, we can gauge a global cutoff point in order to make better decisions for the global profit. It is important to highlight that this approach does not need that local models take the constraints into account during training (i.e. models can be trained and tested as usual). Additionally, we can use data which has been gathered independently for training each model. The only (mild) condition is that model predictions must be accompanied by probabilities (see e.g. [4]) or certainty values, something that almost any family of data mining algorithms can provide. Finally, probabilities and constraints will be used at the simulation stage for estimating the cutoff.

In order to do this, we use the basic Petri Nets formalism [6], with additional data structures, as a simple (but powerful) simulation framework and we use probabilistic estimation trees (classical decision trees accompanied with probabilities [4]). We illustrate this with a very frequent problem in CRM: we apply our approach to a direct-marketing campaign design where several alternative products have to be offered to the same house list of customers. The scenario is accompanied, as usual, by inventory/stock limitations. Even though this problem seems simple at the first sight, there is no simple good analytic solution for it. In fact, we will see that a reasonable analytic approach to set different cutoffs for each product leads to suboptimal overall profits. In contrast, using a joint cutoff probabilistic estimation, which can be obtained through simulation, we get better results.

[1] (http://storm.cis.fordham.edu/~gweiss/ubdm-kdd05.html)

The paper is organised as follows. Section 2 sets the problem framework, some notation and illustrates the analytical (classical) approach. Section 3 addresses the problem with more than one product and presents two methods to solve it. Section 4 includes some experiments with the presented methods. The paper finishes in Section 5 with the conclusions.

2 Campaign Design with One Product

Traditionally, data mining has been widely applied to improve the design of mailing campaigns in Customer Relationship Management (CRM). The idea is simple: discover the most promising customers using data mining techniques, and in this way, increase the benefits of a selling campaign.

The process begins by randomly selecting a sample of customers from the company database (house list). Next, all these customers receive an advertisement of the target product. After a reasonable time, a minable view is constructed with all these customers. In this table, every row represents a different customer and the columns contain information about customers; the predictive attribute (the target class) is a Boolean value that informs whether the corresponding customer has purchased or not the target product. Using this view as a training set, a probability estimation model is learned (for instance a probability estimation tree). This model is then used to rank the rest of customers of the database according to the probability of buying the target product. The last step is to select the optimal cutoff that maximises the overall benefits of the campaign, i.e. the best cutoff of the customer list ranked by customer buying probability.

The optimal cutoff can be computed using some additional information about some associated costs: the promotion material cost (edition costs and sending cost)($Icost$), the benefit from selling one product (b) and the cost to send an advertisement to a customer ($cost$). Given all this information, the *accumulated expected benefit* for a set of customers is computed as follows. Given a list C of customers, sorted by the expected benefit (for $c_k \in C, E\_benefit(c_k) = b \times p(c_k) - cost$), we calculate the *accumulated expected benefit* as $-Icost + \sum_{k=1}^{j} b \times p(c_k) - cost$, where $p(c_k)$ is the estimated probability that customer c_k buys the product and j is the size of the sample of customers to which a pre-campaign has offered the product. The optimal cutoff is determined by the value $k, 1 \leq k \leq j$ for which the greatest accumulated expected benefit is obtained.

The concordance between the real benefits with respect to the expected benefits is very dependent on the quality of the probability estimations of the model. Therefore, it is extremely important to train models that estimate accurate probabilities (e.g. see [4]). A more reliable estimation of the cutoff can be obtained by employing different datasets of customers (or by spliting the existing dataset): a training dataset for learning the probability estimation models, and a validation dataset to compute the optimal cutoff. With this validation dataset the latter estimation of the *accumulated expected benefit* turns into a real calculation of the *accumulated benefit*, where $p(c_k)$ is changed by $f(c_k)$ in the formula, being $f(c_k)$

the response of c_k wrt. the product, such that $f(c_k) = 0$ if customer c_k does not buy the product and $f(c_k) = 1$ if c_k buys it. Then, the cutoff is determined by the greatest accumulated benefit.

Let us see an example where the benefit for the product is 200 monetary units (m.u.), the sending cost is 20 m.u. and the investment cost is 250 m.u. In Table 1 we compare the results obtained with each method. According to the *accumulated expected benefit* we will set the cutoff at 90% of the customers, which clearly differs from the maximum *accumulated benefit* (located at 70%).

Table 1. Accumulated expected benefit vs. Accumulated benefit

Customer	Buys	Probability	E(Benefit)	Acc. Exp. Benefit	Acc. Benefit
				-250	-250
3	YES	0.8098	141.96	-108.04	-70
10	YES	0.7963	139.26	31.22	110
8	YES	0.6605	112.10	143.31	290
1	YES	0.6299	105.98	249.30	470
4	NO	0.5743	94.86	344.15	450
6	NO	0.5343	86.85	431.00	430
5	YES	0.4497	69.94	500.94	**610**
7	NO	0.2675	33.50	534.44	590
9	NO	0.2262	25.24	**559.68**	570
2	NO	0.0786	-4.29	555.39	550

3 Using Simulation and Data Mining for a Campaign Design with More than One Product

The approach shown at the previous section has been successfully applied to very simple cases (i.e. one single product for each campaign), but computing optimal cutoffs by analytic methods is impossible for more complex scenarios (more than one product, constraints for the products, etc.). Therefore, in this section we develop two different approaches: one is an extension of the analytic method, and the other is a more novel and original method based on simulation.

Back on our marketing problem, the objective now is to design a mailing campaign offering N products to a customer list, but taking the following constraints into consideration: there are stock limits (as usual), each product has a different benefit, and the products are alternative, which means that each customer would only buy one of them (or none). As we have seen at Section 2, a good solution, at least apriori, could be to determine a cutoff point defining the segment of customers we have to focus on. But now, since there are several products, it is not clear how this cutoff can be defined/determined. Based on the idea of sorting the customers by their expected benefit, one possibility (what we call the *single approach*) is to combine (in some way, like adding or averaging) the optimal cutoffs which are analytically calculated for each product, in order

to obtain a unique cutoff for the global problem. An alternative method, that we call *joint simulation approach*, is to determine in a dynamic way the global cutoff. We use a validation set to simulate what will happen in a real situation if the customer receives the advertisement (of any of the N products).

Considering that all products have the same sending cost (*cost*), we define the following two alternative ways for obtaining a global cutoff using a validation set C:

1. **Single Approach**: For each product i, we downwardly sort C by the expected benefit of the customers, obtaining N ordered validation sets C_i (one for each product i). Now, for each C_i, $1 \leq i \leq N$, we determine its local cutoff point as we have explained in Section 2. Then, the global cutoff T is obtained by averaging the local cutoffs. In order to apply it, we now jointly sort the customers by their expected benefit considering all products at the same time (that is, just one ranked list obtained by merging the sets C_i). That produces as a result a single list SC where each customer appears N times. Finally, the cutoff T is applied over SC. Then, the real benefit obtained by this method will be the *accumulated benefit* for the segment of customers that will receive the advertisement for the total house list, which will be determined by this cutoff T.
2. **Joint Simulation Approach**: Here, from the beginning, we jointly sort the customers downwarded by their expected benefit of all the products, i.e. we merge the N sets C_i. However, we do not use local cutoffs to derive the global cutoff, but we calculate the cutoff by simulating $N \times |C|$ *accumulated benefits* considering all the possible cutoffs $T_j, 1 \leq j \leq N \times |C|$, where T_1 is the cutoff that only considers the first element of SC, T_2 is the cutoff that considers the two first elements of SC, and so on. Then, the best *accumulated benefit* gives the global cutoff.

To illustrate these two approaches consider a simple example consisting of 10 customers, 2 products (p_1 and p_2) and the parameters $Icost_{p_1} = 150$, $Icost_{p_2} = 250$, $b_1 = 100$, $b_2 = 200$, and $cost = 20$. Table 2 Left shows for each product the list of customers sorted by its expected benefit as well as the local cutoffs marked as horizontal lines. As we can observe, the cutoffs for products p_1 and p_2 are 90% and 70% respectively. Table 2 Right shows the global set and the global cutoff, which is marked by an horizontal line, computed by each approach. Note that the cutoff computed by the single and joint simulation methods is different. For the *single approach*, the global cutoff is 80% (the average of 90% and 70%), whereas the cutoff computed by the *joint simulation approach* is 90%.

We have adopted Petri nets [6] as the framework to formalise the simulation. Petri nets are well-known, easy to understand, and flexible. Nonetheless, it is important to highlight that the method we propose can be implemented with any other discrete simulation formalism. We used a unique Petri net to simulate the behaviour of all the customers, but we also implemented additional data structures to maintain information about customers and products (e.g. remaining stock for each product, remaining purchases for each customer). The Petri net

Table 2. Left: Customers sorted by their expected benefit for the case of two products. Right: Customers and cutoff for the Single and Joint Simulation Approaches

Product p_1			
Customer	**E(Benefit)**	f_{p_1}	**Acc. Benefit**
			-150
2	76.61	1	-70
8	75.71	1	10
9	60.37	0	-10
5	48.19	1	70
1	44.96	1	150
7	30.96	0	130
10	24.58	1	210
3	23.04	0	190
6	7.81	1	270
4	-4.36	0	250

Product p_2			
Customer	**E(Benefit)**	f_{p_2}	**Acc. Benefit**
			-250
3	141.96	1	-70
10	139.26	1	110
8	112.10	1	290
1	105.98	1	470
4	94.86	0	450
6	86.85	0	430
5	69.94	1	610
7	33.50	0	590
9	25.24	0	570
2	-4.29	0	550

	Single & Joint Approaches			
	Customer	**Product**	**Acc. Benefit**	
			-400	
	3	p_2	-220	
	10	p_2	-40	
	8	p_2	140	
	1	p_2	320	
	4	p_2	300	
	6	p_2	280	
	2	p_1	360	
	8	p_1	440	
	5	p_2	620	
	9	p_1	600	
	5	p_1	680	
	1	p_1	760	
	7	p_2	740	
	7	p_1	720	
	9	p_2	700	
Single	10	p_1	780	
	3	p_1	760	
	6	p_1	840	Joint
	2	p_2	820	
	4	p_1	800	

can work with as many products and customers as we need with no change in the Petri net structure. Other similar problems, as mailing campaigns with non-alternative products, can also be handled without changes. Figure 1 shows our Petri net which has 24 places and 10 transitions. Each customer arrives to the Petri net and, thanks to the additional data structures created, the suitable number of tokens are put in each place to allow for the suitable transitions to be enabled/disabled and fired or not. E.g. if the remaining stock of the product is not zero a place *P12* is updated with as many tokens as the current stock is, and place *P11* is put to zero. The first place enables the transition *T4* that can be fired if the rest of conditions are fulfilled (place *P14* has a token), while the second place disables the transition *T5* that cannot be fired. Only two arcs have a weight not equal to one, the arc with the *product benefit* and the arc with the *sending cost*. The first arc finishes in the place *P1* (*Total gross benefit*) and the second one finishes in the place *P15* (*Total loss*). The total (or net) benefit for each cutoff is calculated subtracting the number of tokens accumulated in the places *P1* and *P15* (that is, *Total gross benefit* -*Total loss*).

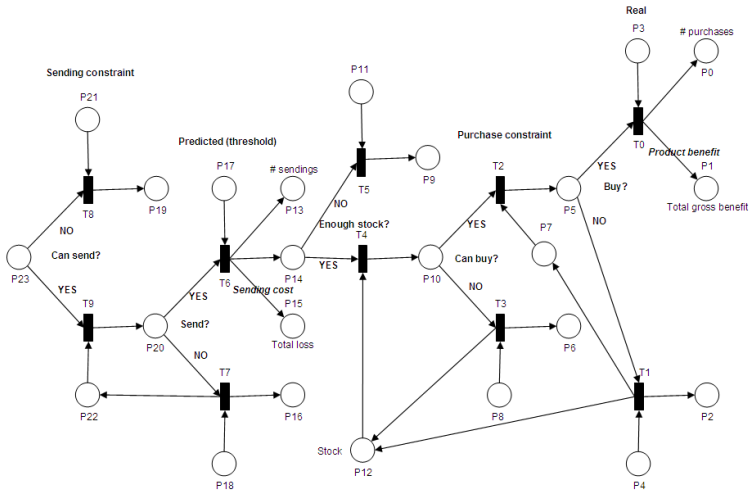


Fig. 1. Petri net for our mailing campaign

In this scenario, we consider that, at the most, only one of the N products can be bought since they are alternative products (e.g. several cars or several houses or different brands for the same product). This constraint suggests to offer to each customer only the product with the higher probability of being bought. If we impose this condition then we say that the approach is *with discarding*. In an approach with discarding, only the first appearance of each customer is taken into account. For instance, in the *single approach*, only the first occurrence of each customer in the customer segment determined by the global cutoff is preserved. Analogously, in the *joint simulation approach*, the simulation process does not consider customers that have been already processed. However, since a prospective customer who receives an offer might finally not buy the product, we consider an alternative option which allows several offers to the same customer. This approach is called *without discarding*. The combination of the two approaches and the two options for considering customer repetitions give four scenarios that will be experimentally analysed in the following section. The notation used for referring to these four different methods is: *Single WO* (Single approach without discarding), *Single WI* (Single approach with discarding), *Joint WO* (Joint simulation approach without discarding), and *Joint WI* (Joint simulation approach with discarding).

4 Experiments with N products

For the experimental evaluation, we have implemented the four methods explained at Section 3 and the Petri net in Java, and have used machine learning algorithms implemented in the data mining suite WEKA [7].

4.1 Experimental settings

For the experiments we have taken a customers file (*newcustomersN.db*) from the SPSS Clementine[2] samples, as a reference. This file has information about only 200 customers, with 8 attributes for each one, 6 of them are nominal and the rest are numeric. The nominal attributes are the *sex* of the customers (male or female), *region* where they live (inner city, rural, town, suburban), whether they are *married*, whether they have *children*, whether they have a *car* and whether they have a *mortgage*. The numeric attributes are the *age* of the customers and their annual *income*.

Since 200 customers are too few for a realistic scenario, we have implemented a random generator of customers. It creates customers keeping the attribute distributions of the example file, i.e. for numeric attributes it generates a random number following a normal distribution with the same mean and deviation as in the example file, and for nominal attributes it generates a random number keeping the original frequency for each value of the attributes in the example file.

Also, to assign a class for each customer (wether s/he buys the product or not), we implemented a model generator. This model generator is based on a random decision tree generator, using the attributes and values randomly to construct the different levels of the tree. We have two parameters which gauge the average depth of the tree and most importantly, the probability of buying each product. We will use these latter parameters in the experiments below.

So, the full process to generate a customer file for our experiments consists of generating the customer data with our random generator of customers and to assign the suitable class with a model obtained by our model generator.

Finally, these are the parameters we will consider and their possible values:

- Number of customers: 10000 (60% training, 20% validation and 20% testing)
- Number of products: 2, 3 and 4
- Probability of buying each product: 0.01, 0.05, 0.2, 0.5, 0.8, 0.95 or 0.99
- Benefits for each product: 100 monetary units (m.u.) for the product 1 and 100, 200, 500 or 1000 m.u. for the other products
- Sending cost (the same for all products): 10, 20, 50 or 90 m.u.
- Stock for each product: 0.1, 0.2, 0.5 or 1 (multiplied by number of customers)
- Investment cost for each product: benefits of the product multiplied by stock of the product and divided by 20
- Correlation (how similar the products are): 0.25, 0.5, 0.75 or 1

4.2 Experimental results

The three main experiments consist in testing 100 times the four approaches for 2, 3 and 4 products, where all the parameters are selected randomly for the cases where there are several possible values.

[2] (http://www.spss.com/clementine/)

If we look at overall results, i.e. averaging all the 100 experiments, as shown in Table 3, the results for 2, 3 and 4 products are consistent. As suggested in [2] we calculate a Friedman test and obtain that the four treatments do not have identical effects, so we calculate a post-hoc test (with a probability of 99.5%) This overall difference is clearly significant, as the significant analysis shown in Table 3, illustrates that the joint simulation approaches are better than the single ones. About the differences between with or without discarding methods, in the case of 2 products there are no significant differences. For 3 products the Single WI method wins the Single WO method, and the Joint WO method wins the Joint WI method. In the case of 4 products the approaches with discarding win the approaches without them. Moreover, in the case of 3 products, the Joint WO method clearly outperforms the other 3 methods and, in the case of 4 products is the Joint WI method which wins the rest of methods.

However, it is important to highlight that these values average many different situations and parameters, including some extreme cases where all the methods behave almost equally. This means that in the operating situations which are more frequent in real applications, the difference may be higher than the one reported by these overall results.

Moreover, in the case of 2 products, from the results of the 100 iterations we create three groups taking into account the probability of buying each product (probability of buying the product 1 is greater, equal or less than probability of buying the product 2) and 3 groups taking into account the stocks for the products (stock for the product 1 is greater, equal or less than stock for the product 2). The results obtained are shown in Figure 2. On one hand, the maximum benefit is obtained for all the methods and results are quite similar when the popularity (probability of buying) of both products is the same. On the other hand, the maximum benefit is obtained for all the methods and results are quite similar too when both products have the same stock. The results differ between the four methods especially when probabilities or stocks are different.

Table 3. Friedman test: wins (√) /loses (X)/draws(=)

	2 products				**3 products**				**4 products**			
	S.WO	**S.WI**	**J.WO**	**J.WI**	**S.WO**	**S.WI**	**J.WO**	**J.WI**	**S.WO**	**S.WI**	**J.WO**	**J.WI**
Benefits	165626	164568	171225	169485	182444	184077	186205	185694	220264	228483	231771	233724
S.WO	-	=	√	√	-	√	√	√	-	√	√	√
S.WI	=	-	=	√	X	-	√	=	X	-	=	√
J.WO	X	=	-	=	X	X	-	X	X	=	-	√
J.WI	X	X	=	-	X	=	√	-	X	X	X	-

5 Conclusion

In this paper, we have presented a new framework to address decision making problems where several data mining models have to be applied under several constraints and taking their mutual influence into account. The method is based on the conjunction of simulation with data mining models, and the adjustment of model cutoffs as a result of the simulation with a validation dataset. We have

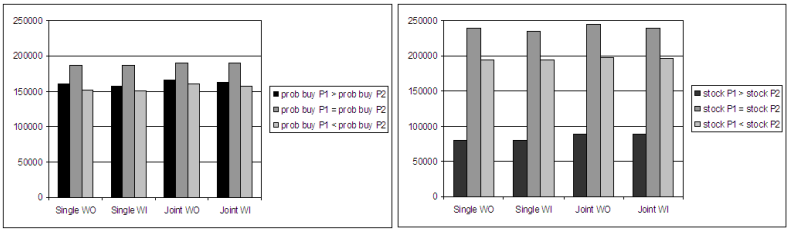


Fig. 2. Left: Variations in probability of buying. Right: Variations in stocks

applied this framework to a direct marketing problem, and we have seen that simulation-based methods are better than classical analytical ones.

This specific direct marketing problem is just an example where our framework can be used. Almost any variation of a mailing campaign design problem could be solved (without stocks, with other constraints, non-alternative products, time delays, joint replies, etc.) in some cases with no changes in the presented Petri net and, in the worst case, by just modifying the Petri net that models the constraints and the relations between models. If not only the cutoff is to be determined but also the optimal stock or other important variables, then other techniques, such as evolutionary computation might be used to avoid a combinatorial explosion of the simulation cases. In our example, though, the combinations are not so huge to allow for an exhaustive analysis of all of them.

Out from marketing, we see prospective applicability in many other domains. In particular, the ideas presented here were originated after a real problem we addressed recently in colaboration with a hospital, where resources and data mining models from different services were highly interwoven. Other domains which we are particular familiar with and we plan to use these ideas are the academic world (e.g. university), where we are using data mining models to predict the number of registered students per course each year, but until now we were not able to model the interdependencies between several courses.

References

1. M. Berry and G. Linoff. *Mastering Data Mining: The Art and Science of Customer Relationship Management*. John Wiley & Sons, Inc., 1999.
2. J. Demsar. Statistical comparisons of classifiers over multiple data sets. *JMLR*, 7:1–30, January 2006.
3. R. Fagin, R. Kumar, M. Mahdian, D. Sivakumar, and E. Vee. Comparing and aggregating rankings with ties. In *PODS '04: Proceedings of the 32nd symp. on Principles of database systems*, pages 47–58. ACM Press, 2004.
4. C. Ferri, P. Flach, and J. Hernández. Improving the AUC of Probabilistic Estimation Trees. In *Proc. of the 14th European Conf. on Machine Learning*, volume 2837 of *Lecture Notes in Computer Science*, pages 121–132, 2003.
5. G. Forman. Counting positives accurately despite inaccurate classification. In *ECML*, volume 3720 of *LNCS*, pages 564–575. Springer, 2005.
6. T. Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, 1989.
7. I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Elsevier, 2005.

7.2 Similarity-Binning Averaging: A Generalisation of Binning Calibration

2. A. Bella, C. Ferri, J. Hernández-Orallo, and M.J. Ramírez-Quintana. **Similarity-Binning Averaging: A Generalisation of Binning Calibration.** In *Intelligent Data Engineering and Automated Learning - IDEAL 2009*, volume 5788 of *Lecture Notes in Computer Science*, pages 341–349. Springer Berlin / Heidelberg, 2009.

Similarity-Binning Averaging: A Generalisation of Binning Calibration

A. Bella, C. Ferri, J. Hernández-Orallo, and M.J. Ramírez-Quintana

Universitat Politècnica de València, DSIC, València, Spain

Abstract. In this paper we revisit the problem of classifier calibration, motivated by the issue that existing calibration methods ignore the problem attributes (i.e., they are univariate). These methods only use the estimated probability as input and ignore other important information, such as the original attributes of the problem. We propose a new calibration method inspired in binning-based methods in which the calibrated probabilities are obtained from k instances from a dataset. Bins are constructed by including the k-most similar instances, considering not only estimated probabilities but also the original attributes. This method has been experimentally evaluated wrt. two calibration measures, including a comparison with other traditional calibration methods. The results show that the new method outperforms the most commonly used calibration methods.

1 Introduction

Many machine learning techniques used for classification are good in discriminating classes but are poorer in estimating probabilities. One reason for this is that when many of these techniques were developed, the relevance and need of good probability estimates was not so clear as it is today. Another reason is that learning a good classifier (a model which tells accurately between two or more classes, i.e., with high accuracy) is usually easier than learning a good class probability estimator. In fact, in the last decade, there has been an enormous interest in improving classifier methods to obtain good rankers, since a good ranker is more difficult to obtain than a good classifier, but still simpler than a good class probability estimator. There are, of course, some other approaches which have addressed the general problem directly, i.e., some classification techniques have been developed and evaluated using probabilistic evaluation measures into account, such as the Minimum Squared Error (MSE) or LogLoss, in the quest for good class probability estimators.

In this context, and instead of redesigning any existing method to directly obtain good probabilities, some calibration techniques have been developed to date. A calibration technique is any postprocessing technique which aims at improving the probability estimation of a given classifier. Given a general calibration technique, we can use it to improve class probabilities of any existing machine learning method: decision trees, neural networks, kernel methods, instance-based

methods, Bayesian methods, etc., but it can also be applied to hand-made models, expert systems or combined models.

This work is motivated by the realisation that existing calibration methods only use the estimated probability to calculate the calibrated probability (i.e., they are univariate). In fact, most calibration methods are based on sorting the instances and/or making bins, such as binning averaging [10] or isotonic regression [3], where the only information which is used to sort or create these bins is the estimated probability. The same happens with other "mapping" methods, such as Platt's method [8]. However, more information is usually available for every instance, such as the original attributes of the problem.

In this paper, we introduce a new calibration method, called Similarity-Binning Averaging (SBA), which is similar to binning methods in the sense that the calibrated probability is calculated from k elements. Instead of sorting the examples in order to compute the bins, we use similarity to compute the k-most similar instances to conform one unique bin for each example. For that purpose, our approach uses not only the estimated probability but the instance attributes too. As a consequence, the resulting learned function is non-monotonic. That means that not only calibration will be affected, but discrimination will also be affected (and hence measures such as the Area Under the ROC Curve (AUC) or even qualitative measures such as accuracy).

The paper is organised as follows, in Section 2, some of the most-known calibration evaluation measures and calibration methods are reviewed. Next, Section 3 presents our calibration method based on binning. An experimental evaluation of the different calibration methods wrt. several measures is included in Section 4. Finally, Section 5 concludes the paper and points out the future work.

2 Calibration Methods and Evaluation Measures

In this section we review some of the most-known calibration methods and introduce the evaluation measures we will employ to estimate the calibration of a classifier. We use the following notation. Given a dataset T, n denotes the number of examples, and c the number of classes. $f(i, j)$ represents the actual probability of example i to be of class j. $p(j)$ denotes the prior probability of class j, i.e., $p(j) = n_j/n$. Given a classifier, $p(i, j)$ represents the estimated probability of example i to be of class j taking values in [0,1].

2.1 Calibration Methods

As we have mentioned in the introduction, the objective of calibration methods (as a postprocessing) is to transform the original estimated probabilities[1] Some well-known calibration methods are:

– The binning averaging method [10] consists in sorting the examples in decreasing order by their estimated probabilities and dividing the set into k

[1] Scores can also be used [11].

bins (i.e., subsets of equal size). Then, for each bin $l, 1 \leq l \leq k$, the corrected probability estimate for a case i belonging to class j ($p^*(i,j)$) is the proportion of instances in l of class j.

- For two-class problems, [3] presented a pair-adjacent violators algorithm (PAV) for calculating the isotonic regression. The first step is to order decreasingly the n elements according to estimated probability and to initialise $p^*(i,j) = f(i,j)$. The idea is that calibrated probability estimates must be a monotone decreasing sequence, i.e., $p_1^* \geq p_2^* \geq \ldots \geq p_n^*$. If it is not the case, the PAV algorithm each time that a pair of consecutive probabilities, $p^*(i,j)$ and $p^*(i+1,j)$, does not satisfy the above property ($p^*(i,j) < p^*(i+1,j)$) replaces both of them by their probability average. This process is repeated (using the new values) until an isotonic set is reached.
- Platt [8] presents a parametric approach for fitting a sigmoid that maps estimated probabilities into calibrated ones[2].

2.2 Calibration Evaluation Measures

Several measures have been proposed and used for evaluating the calibration of a classifier:

- Mean Squared Error (MSE) or Brier Score penalises strong deviations from the true probability.

$$MSE = \frac{\sum_{j=1}^{c}\sum_{i=1}^{n}(f(i,j) - p(i,j))^2}{n \cdot c}$$

  Although originally MSE is not a calibration measure, it was decomposed in [7] in terms of calibration loss and refinement loss.
- A calibration measure based on overlapping binning is CalBin [2]. This is defined as follows. For each class, we must order all cases by predicted $p(i,j)$, giving new indices i^*. Take the 100 first elements (i^* from 1 to 100) as the first bin. Calculate the percentage of cases of class j in this bin as the actual probability, $\hat{f}_j$. The error for this bin is $\sum_{i^* \in 1..100} |p(i^*,j) - \hat{f}_j|$. Take the second bin with elements (2 to 101) and compute the error in the same way. At the end, average the errors. The problem of using 100 (as [2] suggests) is that it might be a much too large bin for small datasets. Instead of 100 we set a different bin length, $s = n/10$, to make it more size-independent.
- There exist other ways of measuring calibration such as Calibration Loss [5], chi-squared test through Hosmer-Lemeshow C-hat (decile bins) or H-hat (fixed thresholds), or through LogLoss.

For a more extensive survey of classification measures we refer the reader to [4].

[2] Originally, Platt proposed this method to be applied to SVM scores.

3 Calibration by Multivariate Similarity-Binning Averaging

As we have shown in the previous section, most calibration methods are based on a univariate transformation function over the original estimated class probability. In binning averaging, isotonic regression or Platt's method, this function is just obtained through very particular mapping methods, using $p(i, j)$ (the estimated probability) as the only input variable. Leaving out the rest of information of each instance (e.g., their original attributes) is a great waste of information which would be useful for the calibration process. For instance, in the case of binning-based methods, the bins are exclusively constructed by sorting the estimated probability of the elements. Binning can be modified in such a way that bins overlap or bins move as windows, but it still only depends on one variable (the estimated probability).

The core of our approach is to change the idea of "sorting" for creating bins, into the idea of using similarity to create bins which are specific for each instance. The rationale for this idea is as follows. If bins are created by using only the estimated probability, calibrated probabilities will be computed from possibly different examples with similar probabilities. The effect of calibration is small, since we average similar probabilities. On the contrary, if we construct the bins using similar examples according to other features, probabilities can be more diverse and calibration will have more effect. Additionally, it will be sensitive to strong probability deviation given by small changes in one or more original features. This means that if noise on a variable dramatically affects the output, probabilities will be smoothed and, hence, they will be more noise-tolerant. For instance, if $(3, 2, a)$ has class $true$ and $(2, 2, a)$ has class $false$, the estimated probability for $(3, 2, a)$ should not be too close to 1.

Based on this reasoning, we have implemented a new calibration method: Similarity-Binning Averaging (SBA). In this method the original attributes and the estimated probability are used to calculate the calibrated one.

We have split the method into 3 stages. The "Stage 0" is the typical learning process. A classification technique is applied to a training dataset to learn a probabilistic classification model (M). In the training dataset, X_{ij} are the attributes and Y_i is the class. This stage may not exist if the model is given beforehand (a hand-made model or an old model).

On the left of figure 1 we can observe the "Stage 1" of the SBA method. In this stage, the trained model M gives the estimated probabilities associated with a dataset. This dataset can be the same used for training, or an additional validation dataset VD, as we have shown in figure 1. The estimated probabilities $p(i, j)$ $1 \leq j \leq c$ are joined as new attributes for each instance i of VD, creating a new dataset VDP. Finally, on the right of figure 1 shows the "Stage 2" of the SBA method. To calibrate a new instance I, first, the estimated probabilities are estimated from the classification model M, and these probabilities are added to the instance creating a new instance (IP). Next, the k-most similar instances to this new instance are selected from the dataset VDP. Finally, the calibrated probability of this instance I for each class j is the predicted class probability

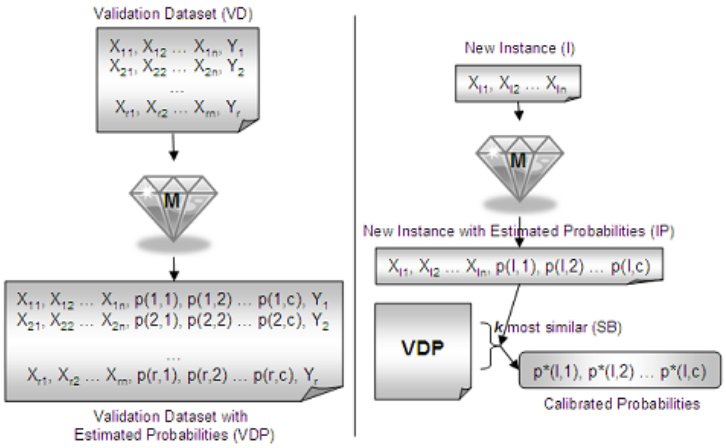


Fig. 1. Left: Stage 1 of the SBA method. Right: Stage 2 of the SBA method.

of the k-most similar instances using their attributes.

Our method is similar to "cascading" [6]. The main difference is that in our method the calibrated probability is the predicted class probability of the k-most similar instances using their attributes and class, i.e., in the first stage instead of adding the class to the instance (as cascading would do), the estimated probabilities of each class are added.

4 Experimental Results

For the experimental evaluation, we have implemented the evaluation measures and the calibration methods explained at Sections 2 and 3, and we have used machine learning algorithms implemented in the data mining suite WEKA [9].

Initially, we have selected 20 (small and medium-sized) binary datasets (table 1) from the UCI repository [1]. We evaluate the methods in two different settings: training and test set, and training, validation and test set. The reason is because the calibration methods can use or not an additional dataset to calibrate. In the training/test setting, (we add a "T" at the end of the name of the methods) randomly, each dataset is split into two different subsets: the training and the test sets (75% and 25% of the instances, respectively). In the training/validation/test setting, (we add a "V" at the end of the name of the methods) randomly, each dataset is split into three different subsets: the training, the validation and the test sets (56%, 19% and 25% of the instances, respectively). Four different methods for classification have been used (with their default parameters in WEKA): NaiveBayes, J48 (a C4.5 implementation), IBk ($k = 10$) (a k-NN implementation) and Logistic (a logistic regression implementation). A total of 400 repeti-

tions have been performed for each dataset (100 with each classifier). In each repetition, for the training/test setting, the training set is used to train a classifier and calibrate the probabilities of the model, and the test set is used to test the calibration of the model, while for the training/validation/test setting, the training set is used to train a classifier, the validation set is used to calibrate the probabilities of the model, and the test set is used to test the calibration of the model. Furthermore, in each repetition the same training, validation and test sets are used for all methods.

Table 1. Datasets used in the experiments. Size and number of nominal and numeric attributes.

#	Datasets	Size	Nom.	Num.	#	Datasets	Size	Nom.	Num.
1	Breast Cancer	286	9	0	11	House Voting	435	16	0
2	Wisconsin Breast Cancer	699	0	9	12	Ionosphere	351	0	34
3	Chess	3196	36	0	13	Labor	57	8	8
4	Horse Colic	368	15	7	14	Monks1	556	6	0
5	Credit Rating	690	9	6	15	Mushroom	8124	22	0
6	German Credit	1000	13	7	16	Sick	3772	22	7
7	Pima Diabetes	768	0	8	17	Sonar	208	0	60
8	Haberman BreastW	306	0	3	18	Spam	4601	0	57
9	Heart Statlog	270	0	13	19	Spect	80	0	44
10	Hepatitis	155	13	6	20	Tic-tac	958	8	0

The calibration methods used in the experiments are: binning averaging (with 10 bins), PAV algorithm, Platt's method, and Similarity-Binning Averaging (SBA) (with $k = 10$). All of them have been evaluated for the CalBin and MSE calibration measures. Apart from comparing the results of the calibration methods, we also compare them with two reference methods:

- Base: is the value obtained with the classification techniques without calibration.
- 10-NN[3]: is the value of using the 10 most similar instances (just with the original attributes) to directly estimate the calibrated probability. This method is just to show the importance of using the estimated probabilities as inputs to compute the similarity.

In tables 2 and 3 we show the results with respect to CalBin and MSE measures for each method (for both measures the lower the better). These values are the average of the 400 repetitions for each dataset.

As we can see in the last row of tables 2 and 3, with both calibration measures our method SBA with the training/test setting has obtained the best results and our method SBA with the training/validation/test setting has obtained good results as well.

[3] Implemented by an IBk with $k = 10$ in WEKA

Table 2. Results by dataset: Measure CalBin. Training/test setting (T) and Training/validation/test setting (V).

	ClassT	10-NNT	BinT	PAVT	PlattT	SBAT	BinV	PAVV	PlattV	SBAV
1	0.1953	0.1431	0.2280	0.2321	0.1856	0.1827	0.3092	0.2928	0.2446	0.1924
2	0.0494	0.0374	0.0647	0.0447	0.0623	0.0408	0.0791	0.0538	0.0775	0.0423
3	0.0698	0.1472	0.0501	0.0397	0.0434	0.0491	0.0548	0.0448	0.0479	0.0628
4	0.1517	0.1216	0.1533	0.1535	0.1421	0.1164	0.1996	0.1853	0.1563	0.1244
5	0.1220	0.0882	0.1060	0.1035	0.1132	0.0848	0.1408	0.1293	0.1269	0.0874
6	0.1250	0.1340	0.1263	0.1393	0.1268	0.1233	0.1933	0.1855	0.1227	0.1233
7	0.1192	0.1049	0.1220	0.1351	0.1205	0.1105	0.1889	0.1861	0.1267	0.1199
8	0.1984	0.2028	0.2316	0.2400	0.1998	0.1994	0.2877	0.2798	0.2777	0.2149
9	0.1476	0.1412	0.1690	0.1587	0.1529	0.1443	0.2247	0.1995	0.1834	0.1432
10	0.1632	0.1359	0.1643	0.1673	0.1727	0.1332	0.2082	0.1999	0.2597	0.1358
11	0.0665	0.0625	0.0777	0.0542	0.0791	0.0516	0.0945	0.0672	0.1006	0.0588
12	0.1380	0.1588	0.1179	0.0990	0.1358	0.1064	0.1701	0.1428	0.1854	0.1303
13	0.1876	0.2996	0.1984	0.1464	0.2110	0.1820	0.2914	0.1940	0.4478	0.2792
14	0.1442	0.2794	0.1618	0.1355	0.1046	0.1443	0.2067	0.1740	0.1340	0.1730
15	0.0395	0.0366	0.0418	0.0358	0.0468	0.0368	0.0434	0.0359	0.0494	0.0367
16	0.0296	0.0158	0.0270	0.0236	0.0250	0.0194	0.0297	0.0264	0.0285	0.0265
17	0.2606	0.1916	0.2343	0.2376	0.2374	0.2007	0.3207	0.2924	0.2750	0.1844
18	0.0945	0.0471	0.0636	0.0568	0.0951	0.0466	0.0733	0.0658	0.0964	0.0910
19	0.3138	0.3497	0.2995	0.2911	0.3110	0.3110	0.3615	0.3380	0.4265	0.3117
20	0.1240	0.2094	0.1260	0.1198	0.0906	0.0934	0.1736	0.1621	0.0971	0.0824
AVG.	0.1370	0.1453	0.1382	0.1307	0.1328	**0.1188**	0.1826	0.1628	0.1732	0.1310

There are some differences between the results when calibration methods are evaluated with each measure (CalBin and MSE) (tables 2 and 3). These differences come from the different nature of the measures. While CalBin is a measure that only evaluates calibration, MSE also evaluates other components.

It is important to remark that we are making general comparisons between methods in equal conditions. First of all we are comparing to classification methods without calibration (Base). Logically, calibration would not have had any sense if we had not improved the results. The second comparison is with the 10-NN method, which is related to our method, but only uses the original attributes of the problem to make the bin of the 10 elements which are more similar and to obtain the calibrated probability. The other three methods we compare to only use the estimated probability to calculate the calibrated probability. The most interesting comparison is with the binning averaging method, because our method is also based on the idea of binning.

If we observe table 3, it is important to remark how our method improves significantly the other calibration methods in terms of MSE.

Additional experiments: grouped by classification method, suitable statistical tests to confirm the differences in the results are significant and multiclass experiments can be found at: http://users.dsic.upv.es/∼abella/MulticlassExperiments.pdf.

Table 3. Results by dataset: Measure MSE. Training/test setting (T) and Training/validation/test setting (V).

	ClassT	10-NNT	BinT	PAVT	PlattT	SBAT	BinV	PAVV	PlattV	SBAV
1	0.2086	0.1912	0.2086	0.2095	0.2016	0.1998	0.2123	0.2136	0.2081	0.1982
2	0.0353	0.0262	0.0510	0.0343	0.0362	0.0306	0.0635	0.0375	0.0380	0.0316
3	0.0465	0.0648	0.0467	0.0387	0.0391	0.0227	0.0515	0.0424	0.0423	0.0332
4	0.1506	0.1351	0.1503	0.1449	0.1442	0.1336	0.1641	0.1535	0.1513	0.1371
5	0.1347	0.1121	0.1244	0.1203	0.1262	0.1160	0.1320	0.1239	0.1287	0.1176
6	0.1889	0.1795	0.1888	0.1883	0.1877	0.1814	0.1849	0.1832	0.1821	0.1800
7	0.1790	0.1749	0.1795	0.1786	0.1777	0.1821	0.1818	0.1779	0.1756	0.1835
8	0.1926	0.1992	0.1924	0.1936	0.1906	0.2005	0.1966	0.1982	0.1974	0.1957
9	0.1491	0.1435	0.1580	0.1503	0.1470	0.1469	0.1675	0.1534	0.1522	0.1390
10	0.1473	0.1294	0.1460	0.1483	0.1397	0.1305	0.1546	0.1505	0.1585	0.1271
11	0.0554	0.0568	0.0646	0.0482	0.0538	0.0456	0.0816	0.0574	0.0599	0.0524
12	0.1266	0.1297	0.1118	0.0981	0.1094	0.0996	0.1311	0.1071	0.1186	0.1141
13	0.1120	0.1233	0.1582	0.1128	0.1044	0.0907	0.2109	0.1419	0.2288	0.1196
14	0.1214	0.1047	0.1172	0.1030	0.1065	0.0517	0.1362	0.1164	0.1244	0.0819
15	0.0083	0.0006	0.0174	0.0040	0.0079	0.0001	0.0198	0.0045	0.0088	0.0003
16	0.0310	0.0311	0.0355	0.0266	0.0307	0.0241	0.0370	0.0275	0.0277	0.0370
17	0.2545	0.1760	0.2343	0.2286	0.2285	0.2080	0.2305	0.2157	0.2225	0.1847
18	0.1027	0.0814	0.0765	0.0721	0.0878	0.0690	0.0800	0.0746	0.0895	0.1042
19	0.2829	0.2459	0.2776	0.2637	0.2437	0.2692	0.2639	0.2482	0.2568	0.2276
20	0.1579	0.1141	0.1480	0.1448	0.1468	0.0817	0.1571	0.1522	0.1526	0.1108
AVG.	0.1343	0.1210	0.1343	0.1254	0.1255	**0.1142**	0.1428	0.1290	0.1362	**0.1188**

5 Conclusions

In this work we have revisited the problem of class probability calibration. We have generalised the idea of binning by constructing the bins using similarity to select the k-most similar instances. In this way, we have a different bin for each example and, of course, bins can overlap. Similarity is not computed by only using the estimated probabilities but also with the problem attributes. Our hypothesis was that calibration would be more effective the more information we are able to provide for computing this similarity. Leaving the problem attributes out (as traditional calibration methods do) is like making the problem harder than it is.

The implementation of the method is straightforward through any off-the-shelf k-NN algorithm. Consequently, our method is closely related to the cascade generalisation method [6].

The experimental results we have presented here confirm the previous hypothesis and show a significant increase in calibration for the two calibration measures considered, over three well-known and baseline calibration techniques: non-overlapping binning averaging, Platt's method and PAV. It is true that this calibration is partly obtained because of the increase of AUC and it is, consequently, non monotonic, but in many applications where calibration is necessary the restriction of being monotonic is not only applicable, but it is an

inconvenience. In fact, when calibrating a model, the original model and class assignments can be preserved, while the only thing that has to be modified is the new probabilities. In other words, the predictions of a comprehensible model composed of a dozen rules can be annotated by the estimated probabilities while preserving the comprehensibility of the original model.

As future work we are working on attribute-weighted k-NN to form the bins in order to gauge the importance of attributes for cases when there is a great number of attributes or a great number of classes. Similarly, we want to use locally-weighted k-NN, where closer examples have more weight, in order to make the method more independent from k.

Another future work is the analysis of the method for multiclass problems, because as we have seen in the definition of our method, it can be applied to multiclass problems. We have to compare with other approximations like [11] and find some other calibration methods, since binning, Platt's and PAV cannot deal directly with multiclass problems.

References

1. C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998.
2. R. Caruana and A. Niculescu-Mizil. Data mining in metric space: an empirical analysis of supervised learning performance criteria. In *Proc. of the 10th Intl. Conference on Knowledge Discovery and Data Mining*, pages 69–78, 2004.
3. M. Ayer et al. An empirical distribution function for sampling with incomplete information. *Annals of Mathematical Statistics*, 5:641–647, 1955.
4. C. Ferri, J. Hernández-Orallo, and R. Modroiu. An experimental comparison of performance measures for classification. *Pattern Recogn. Lett.*, 30(1):27–38, 2009.
5. P. Flach and E. Matsubara. A simple lexicographic ranker and probability estimator. In *18th European Conference on Machine Learning*, pages 575–582, 2007.
6. J. Gama and P. Brazdil. Cascade generalization. *Machine Learning*, 41:315–343, 2000.
7. A. H. Murphy. Scalar and vector partitions of the probability score: Part ii. n-state situation. *Journal of Applied Meteorology*, 11:1182–1192, 1972.
8. J. C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in Large Margin Classifiers*, pages 61–74. MIT Press, Boston, 1999.
9. I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Elsevier, 2005.
10. B. Zadrozny and C. Elkan. Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers. In *Proc. of the 18th Intl. Conference on Machine Learning*, pages 609–616, 2001.
11. B. Zadrozny and C. Elkan. Transforming classifier scores into accurate multiclass probability estimates. In *The 8th ACM SIGKDD Intl. Conference on Knowledge Discovery and Data Mining*, pages 694–699. ACM, 2002.

7.3 Calibration of Machine Learning Models

3. A. Bella, C. Ferri, J. Hernández-Orallo, and M.J. Ramírez-Quintana. **Calibration of Machine Learning Models.** In *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques*, pages 128–146. IGI Global, 2010.

Calibration of Machine Learning Models

ABSTRACT

Evaluation of machine learning methods is a crucial step before application, because it is essential to assess how good a model will behave for every single case. In many real applications, not only the "total" or the "average" of the error of the model is important but it is also important to know how this error is distributed or how well confidence or probability estimations are made. However, many machine learning techniques are good in overall results but have a bad distribution /assessment of the error.

In these cases, calibration techniques have been developed as postprocessing techniques which aim at improving the probability estimation or the error distribution of an existing model.

In this chapter, we present the most usual calibration techniques and calibration measures. We cover both classification and regression, and we establish a taxonomy of calibration techniques, while then paying special attention to probabilistic classifier calibration.

INTRODUCTION

One of the main aims of machine learning methods is to build a model or hypothesis from a set of data (also called evidence). After this learning process, it is usually necessary to evaluate the quality of the hypothesis as precisely as possible. For instance, if prediction errors have negative consequences in a certain application domain of a model (for instance, detection of carcinogenic cells) it is important to know exactly the accuracy the model has. Therefore, the model evaluation stage is crucial for the real application of the machine learning techniques. Generally, the quality of predictive models is evaluated using a training set and a test set (which are usually obtained by partitioning the evidence into two disjoint sets), or using some kind of cross-validation or bootstrap if more reliable estimations are desired. These evaluation methods work for any kind of estimation measure. It is important to mention that different measures can be used depending on the model. The most common measures are, for classification models, accuracy (or, inversely error), f-measure, or macro-average. In probabilistic classification, besides the percentage of correctly classified instances, other measures like logloss, mean squared error (MSE) or Brier's score, or area under the ROC curve (AUC) are used. For regression, the most common ones are to use MSE or the mean absolute error (MAE).

With a same result for a quality metric (e.g. MAE), two different models might have a different error distribution. For instance, a regression model R_1 always predicting the true value plus 1 has a MAE of 1 but it is different to a model R_2 that predicts the true value for each n examples but one, where the error for this one is n. The first seems to be more reliable or stable, or in other words, its error is more predictable. Similarly, with the same result for a quality metric (e.g. accuracy), two different models might have different error assessment. For instance, a classification model C_1 which is correct 90% of the cases with confidence 0.91 for every prediction is preferable to model C_2 which is correct 90% of the cases with confidence 0.99 for every prediction. The error self-assessment, i.e. the purported confidence is more accurate in the first case than in the second one.

In both cases above, an overall picture of these results, i.e. an empirical behaviour of how it behaves, is helpful to improve its reliability or confidence in many cases. In the first case, the regression model R_1 which always predicts the true value plus 1 is clearly uncalibrated, since predictions are usually 1 unit above the real value. Subtracting 1 unit to all prediction would calibrate R_1. Interestingly, R_2 is calibrated in the same way. For the second case, a global calibration

requires confidence estimation to be around 0.9 since the models are right 90% of the time.

So, calibration might be understood in many ways, but it is usually built around two issues: how error is distributed, and how self-assessment (confidence or probability estimation) is performed. Although both ideas can be applied for both regression and classification, the first one has been mainly discussed for regression and the second one for classification.

Both are closely related, since for a regular, predictable model such as R_1 the one which always predicts the true value plus 1, it is much easier to estimate a probability (since it is a continuous value, a probability density function). In this case, the probability estimation can be just a density function which includes all the probability to the interval between the predicted value minus 1 and the predicted value.

Estimating probabilities or confidence values is crucial in many real applications. For example, if we have accurate probabilities, decisions can be made with a good assessment of risks and costs, using utility models or other techniques from decision making. Additionally, their integration with other models (e.g. multiclassifiers) or with previous knowledge becomes more robust. In classification, probabilities can be understood as confidence degrees, especially in binary classification, thus accompanying every prediction with a reliability score (DeGroot & Fienberg, 1982). Regression models might accompany predictions by confidence intervals or by probability density functions.

In this context, and instead of redesigning any existing method to directly obtain good probabilities or a better error distribution, some calibration techniques have been developed to date. A calibration technique is any postprocessing technique which aims at improving the probability estimation or to improve error distribution of a given predictive model. Given a general calibration technique, we can use it to improve any existing machine learning method: decision trees, neural networks, kernel methods, instance-based methods, Bayesian methods, etc., but it can also be applied to hand-made models, expert systems or combined models

Depending on the task, different calibration techniques can be applied, and the definition of calibration can be stated more precisely. The most usual calibration techniques are listed below, including different names we give them in order to clarify the rest of this chapter, as well as a type code:

- TYPE CD. Calibration techniques for discrete classification ("(class) distribution calibration in classification" or simply "class calibration"): a typical decalibration arises when the model predicts examples of one or more classes in a proportion which does not fit the original proportion, i.e., the original class distribution. In the binary case (two classes) it can be expressed as a mismatch between the expected value of the proportion of classes and the actual one. For instance, if a problem has a proportion of 95% of class 'a' and 5% of class 'b', a model predicting 99% of class 'a' and 1% of class 'b' is uncalibrated, although it could have a low error (ranging from 4% to 5%) . This error distribution can be clearly seen on a confusion or contingency table. So, class calibration is defined as the degree of approximation of the true or empirical class distribution with the estimated class distribution. The standard way to calibrate a model in this way is by changing the threshold that determines when the model predicts ‘a’ or ‘b’, making this threshold stricter with class ‘a’ and milder with class ‘b’ to balance the proportion. Note that this type of calibration, in principle, might produce more error. In fact, it is usually the case when one wants to obtain a useful model for problems with very imbalanced class distribution, i.e. the minority class has very few examples. Note that we are usually interested in a match between global proportions, but this calibration can also be studied and applied locally. This is related to the problem of "repairing concavities" (Flach & Wu, 2005).

- TYPE CP. Calibration techniques for probabilistic classification ("probabilistic calibration for classification"): a probabilistic classifier is a classifier which accompanies each prediction with a probability estimation. If we predict that we are 99% sure, we should expect to be right 99% of the times. If we are only right 50% of the times, this is not calibrated because our estimation was too optimistic. Similarly, if we predict that we are only 60% sure, we should expect to be right 60% of the times. If we are right 80% of the times, this is not calibrated because our estimation was too pessimistic. In both cases, the expected value of the number or proportion of right guesses (in this case the probability or the confidence assessment) does not match the actual value. Calibration is then defined as the degree of approximation of the predicted probabilities to the actual probabilities. More precisely, a classifier is perfectly calibrated if for a sample of examples with predicted probability p, the expected proportion of positives is close to p. Note that accuracy and calibration, although dependent, are very different things. For instance, a random classifier (a coin tosser) which always assigns 0.5 probability to their predictions is perfectly calibrated. On the other side, a very good classifier can be uncalibrated if correct positive (resp. negative) predictions are accompanied by relative low (resp. high) probabilities. Also note that good calibration usually implies (except from the random coin tosser) that estimated probabilities are different for each example. For some examples, confidence will be high and for other more difficult ones, confidence will be low. This implies that measures to evaluate this type of calibration must evaluate agreement between the expected value and the real value in a local way, by using partitions or bins of the data.

- TYPE RD. Calibration techniques to fix error distribution for regression ('distribution calibration in regression'): in this case the errors are not distributed regularly along the output value. The error is concentrated in the big values or it is gone over to positive or negative values. The expected value which should be close to the actual value can be defined in several ways. For instance, the expected value of the estimated value (y_{est}) should be equal (or close) to the real value (y), i.e. $E(y_{est}) = E(y)$ or, equivalently, $E(y_{est} - y) = 0$. In the example R_1 above, $E(y_{est}) = E(y) + 1$. The mean error (its expected value) would be 1 and not 0. Another equation that shows that a model might be uncalibrated is the expected value of the quotient between the estimated value and the real value, $E(y_{est} / y)$ which should be equal or close to 1. If this quotient is greater than one, the error used to be positive for high values and negative for low values. Typically, these problems are detected and penalised by typical measures for evaluating regression models, and many technique (e.g. linear regression), generate calibrated models (at least in these two aspects mentioned above). Other kind of more sophisticated techniques or, more frequently, hand-made models, might be uncalibrated and might require a calibration.

- TYPE RP. Calibration techniques to improve probability estimation for regression ('probabilistic calibration for regression'): This is a relatively new area (Carney & Cunningham, 2006) and is applicable when continuous predictions are accompanied or substituted by a probability density function (or, more restrictively, confidence intervals). This kind of regression models are usually referred as "density forecasting" models. Instead of saying that temperature is going to be 23.2º Celsius, we give a probability density function from which we can calculate that the probability of the temperature to be between 21º and 25º is 0.9 and the probability of the temperature to be between 15º and 31º is 0.99. If our predictions are very accurate, density functions (and hence confidence intervals) should be narrower. If our predictions are bad, density functions should be broader, in order to approximate the estimated probabilities to the real probabilities. As in the type CP, a good calibration requires in general that these density functions are particular for each prediction, i.e., for some cases where the confidence is high, confidence intervals will be narrower. For difficult cases, confidence intervals will be broader. As in the type CP, measures to evaluate this type of calibration must evaluate agreement between the expected value and the real value in a local way, by using partitions or bins of the data.

Table 1 summarises these four types of calibration.

TYPE	Task	Problem	Global/Local	What is calibrated?
CD	Classification	Expected class distribution is different from real class distribution	Global or local	Predictions
CP	Classification	Expected/estimated probability of right guesses different from real proportion.	Local	Probability/confidence
RD	Regression	Expected output is different from real average output.	Global or local	Predictions
RP	Regression	Expected/estimated error confidence intervals or probability density functions are too narrow or too broad.	Local	Probability/confidence

Table 1. A taxonomy of calibration problems.

Note that types CD and RD necessarily must modify predictions in order to calibrate the results. In type CD, if we move the class threshold, some predictions change. In RD if we try, let us say, to reduce high values and increase low values, predictions also change. In contrast, for types CP and RP, calibration can be made without (necessarily) modifying predictions: only confidence assessments or probabilities need to be touched. For CP, in particular, these kinds of calibrations are known as *isotonic*. Consequently, some measures as average error will not be affected by these two types of calibrations.

Additionally, since we want to improve calibration, we need measures to evaluate this improvement. A calibration measure is any measure which is able to quantify the degree of calibration of a predictive model. For each type of calibration model, some specific measures are useful to evaluate the degree of calibration, while others are only partially sensitive or completely useless. For instance, for CP, the most common measure, accuracy (or % of errors), is completely useless. For RP, the most common measure, MSE, is completely useless. We will review some of these calibration measures in the following section.

For all the types in Table 1, type CP is the one which has devoted more attention recently. In fact, for many researchers in machine learning, the term "calibration" usually refers to this type, without the need of specifying that there are other types of calibration. Additionally, this is the type which has developed more techniques and more specific measures. Furthermore, regression techniques and measures have been traditionally developed to obtain calibrated models, so less improvement is expected from calibration techniques. For this reason, we will devote much more space to classification, and very especially to type CP.

Overall, in this chapter we give a general overview about calibration and review some of the most-known calibration evaluation measures and calibration methods which have been proposed for classification and regression. We conclude analysing some open questions and challenges which can constitute the research on calibration in the future.

CALIBRATION EVALUATION MEASURES

As mentioned in the introduction, a calibration measure is any measure which is able to quantify the degree of calibration of a classifier. As we can see in Table 2, many classical quality metrics are not useful to evaluate calibration techniques. In fact, new and specific measures have been derived or adapted to evaluate calibration, especially for types CP and RP.

TYPE	Calibration measures	Partially sensitive measures	Insensitive measures
CD	Macro-averaged accuracy, proportion of classes.	Accuracy, mean F-measure, ...	Area Under the ROC Curve (AUC), MSE^p, Logloss, ...
CP	MSE^p, LogLoss, CalBin, CalLoss		AUC, Accuracy, mean F-measure, ...
RD	Average error, Relative error	MSE^r, MAE, ...	
RP	Anderson-Darling (A2) test		Average error, relative error, MSE^r, MAE, ...

Table 2. Calibration measures (second column) for each type of calibration problem. On the third and fourth columns we show measures which are partially sensitive (but not very useful in general) or completely insensitive to each type of calibration.

The second column in the above table shows the calibration measures. This does not mean, though, that these measures *only* measure calibration. For instance, for type CP, CalBin and CalLoss only evaluate calibration, while MSE or Logloss evaluate calibration and other components at the same time. We will refer to these two types of measures as pure and impure calibration measures, respectively. Pure calibration measures have the risk that a classifier which always predicts the positive prior probability is perfectly calibrated according to these measures. Following with the type CP, some other metrics are insensitive to calibration, such as qualitative measures (accuracy, mean F-measure, etc.), provided the calibration function is also applied to the threshold, or measures of ranking (such as AUC), provided that calibration modifies the value of the probabilities but not their order. This is the reason-why calibration has emerged as an important issue, since for many traditional quality metrics, calibration issues are completely disregarded. Hence, many machine learning techniques generate uncalibrated models.

Note that we use two different terms for Mean Square Error, MSE^p and MSE^r. The reason-why is that the first one is used for classification and compares the estimated probabilities with the actual probability (0 or 1), while the second one is used for regression and compares two continuous values.

From the measures in the second column, Macro-averaged accuracy, proportion of classes, MSEp, LogLoss, CalBin, CalLoss, Average error, Relative error, Anderson-Darling (A2) test, some of them are very well-known and do not need any further definition, but a few words: macro-averaged accuracy is the average of the partial accuracies for each class, the proportion of classes is computed for the predictions on a dataset and can be compared with the real proportion. Average error and relative error are well-known in regression. Consequently, we will devote the rest of this section to explain MSE^p, LogLoss, CalBin, CalLoss for type CP and Anderson-Darling (A2) test for RP.

We will first start with the measures that are applicable to probabilistic classifiers. We use the following notation. Given a dataset T, n denotes the number of examples, and C the number of classes. $f(i, j)$ represents the actual probability of example i to be of class j. We assume that $f(i, j)$

always takes values in {0,1} and is strictly not a probability but an indicator function. With $n_j = \sum_{i=1}^{n} f(i,j)$, we denote the number of examples of class *j*. *p*(*j*) denotes the prior probability of class *j*, i.e., $p(j) = n_j / n$. Given a classifier, $p(i, j)$ represents the estimated probability of example *i* to be of class *j* taking values in [0,1].

With these definitions, we can define the measures as follows.

Mean Squared Error

Mean Squared Error (MSE) is a measure of the deviation from the true probability. We have used MSE^p to distinguish this measure with the homonym used in regression. In classification it is also know as Brier Score. It is defined as

$$MSE = \sum_{j=1}^{C} \sum_{i=1}^{n} \frac{(f(i,j) - p(i,j))^2}{n \times C}$$

Although originally MSE is not a calibration measure, it was decomposed in (Murphy, 1972) in terms of calibration loss and refinement loss. An important idea of the decomposition is that data is organised into bins, and the observed probability in each bin is compared to the predicted probability or to the global probability. As we will see, some kind of binning will be present in many calibration methods and measures. The calibration component is the only one which is affected by isotonic calibrations, since discrimination and uncertainty components are not modified if probabilities are calibrated in such a way that the order is not modified (i.e. isotonic), since bins will not be altered. For the decomposition, *T* is segmented in *k* bins.

$$MSE = \frac{\sum_{j=1}^{C} \sum_{l=1}^{k} \sum_{i=1, i \in l}^{n_l} n_l \times \left(p(i,j) - \overline{f}(i,j)\right)^2 - \sum_{l=1}^{k} n_l \times \left(\overline{f}_l(i,j) - \overline{f}(j)\right) + \overline{f}(j) \times \left(1 - \overline{f}(j)\right)}{n \times C}$$

where $\overline{f}_l(i,j) = \sum_{i=1, i \in l}^{n_l} \frac{f(i,j)}{n_l}$ and $\overline{f}(j) = \sum_{i=1}^{n} \frac{f(i,j)}{n}$. The first term measures the calibration of the classifier while the rest of the expression measures other components usually grouped under the term "refinement ".

LogLoss

Logloss is a similar measure, and it is also known as probabilistic cost or entropy. It is very related with the Kullback-Leibler distance between the real model and the inferred model (Good, 1952; Good, 1968; Dowe, Farr, Hurst, & Lentin, 1996). It is defined as follows:

$$Logloss = \sum_{j=1}^{C} \sum_{i=1}^{n} \frac{(f(i,j) \times \log p(i,j))}{n}$$

Calibration by Overlapping Bins

One typical way of measuring classifier calibration is that the test set must be split into several segments or bins, as the MSE decomposition shows (although MSE does not need to use bins to be computed). The problem of using bins is that if too few bins are defined, the real probabilities are not properly detailed to give an accurate evaluation. If too many bins are defined, the real probabilities are not properly estimated. A partial solution to this problem is to make the bins overlap.

A calibration measure based on overlapping binning is CalBin (Caruana & Niculescu-Mizil, 2004). This is defined as follows. For each class, we must order all cases by predicted $p(i, j)$, giving new indices i^*. Take the 100 first elements (i^* from 1 to 100) as the first bin. Calculate the percentage of cases of class j in this bin as the actual probability, $\hat{f}_j$. The error for this bin is $\sum_{i^* \in 1..100} \left| p(i, j) - \hat{f}_j \right|$. Take the second bin with elements (2 to 101) and compute the error in the same way. At the end, average the errors. The problem of using 100 as Caruana and Niculescu-Mizil (2004) suggest is that it might be a much too large bin for small datasets. Instead of 100 we might set a different bin length, $s = n/10$, to make it more size-independent. Formally:

$$CalBin(j) = \frac{1}{n-s} \sum_{b=1}^{n-s} \sum_{i=b}^{b+s-1} \left| p(i^*, j) - \frac{\sum_{i=b}^{b+s-1} f(i^*, j)}{s} \right|$$

Calibration Loss

In (Fawcett & Niculescu-Mizil, 2007) and, independently, in (Flach & Matsubara, 2007), the relationship between the AUC-based measures,and ROC analysis in general, with calibration was clarified. The receiver operating characteristic curve (ROC curve) is a graphical depiction of classifiers based on their performance, It is generally applied to binary classifiers. The ROC space is a two-dimensional graph in which the True positive rate (the fraction of positives correctly classified or *tp rate*) is plotted on the Y axis and the False positive rate (the fraction of negatives incorrectly classified or *fp rate*) is plotted on the X axis.. Each discrete classifier produces an (*fp rate, tp rate*) pair that corresponds to a single point in the ROC space. Probabilistic classifiers provide a value (probability or score) that represents the degree to which an instance belongs to a class. In combination with a threshold, the classifier can behave as a binary classifier assigning a class (for instance, positive) if the produced score is above the threshold and the other class (negative) otherwise. In these cases, each threshold produces one point in the ROC space, and drawing a line crossing all the points, a ROC curve is generated. The area under a ROC curve is abbreviated as AUC. "The AUC has an important statistical property: the AUC of a classifier is equivalent to the probability that the classifier will rank a randomly chosen positive instance higher than a randomly chosen negative instance. This is
equivalent to the Wilcoxon test of ranks" (Fawcett, 2006, p. 868). So, in some way, the AUC is a class separability or instance ranking measure because it evaluates how well a classifier ranks its predictions.

A perfectly calibrated classifier always gives a convex ROC curve. However, a classifier can produce very good rankings (high AUC), but probabilities might differ from the actual probabilities. A method for calibrating a classifier is to compute the convex hull or, equivalently, to use isotonic regression. Flach and Matsubara (2007) derive a decomposition of the Brier Score into calibration loss and refinement loss. Calibration loss is defined as the mean squared deviation from empirical probabilities derived from slope of ROC segments.

$$CalLoss(j) = \sum_{b=1}^{r_j} \sum_{i \in s_{j,b}} \left(p(i,j) - \sum_{i \in s_{j,b}} \frac{f(i,j)}{|s_{j,b}|} \right)$$

where rj is the number of segments in the ROC curve for class j, i.e. the number of different estimated probabilities for class j: |{p(i, j)}|. Each ROC segment is denoted by $s_{j,b}$, with $b \in 1 \ldots r_j$, and formally defined as:

$$s_{j,b} = \left\{ i \in 1 \ldots n \middle| \forall k \in 1 \ldots n : p(i,j) \geq p(k,j) \wedge i \notin s_{j,d}, \forall d < b \right\}$$

Anderson-Darling (A^2) test

From type CP, we move now to type RP, where the task is usually referred to as density forecasting, where instead of predicting a continuous value, the prediction is a probability density function. Evaluating this probability density function in terms of calibration cannot be done with typical measures such as MSEr, relative quadratic error or other classical measures in regression. (Carney & Cunningham 2006) adapt an old measure, the Anderson-Darling (A^2) test over the probability integral transform, as a measure of pure calibration. This measure is used to evaluate whether the probability density functions estimated by the regression model are accurate. For the specific definition, we refer the reader to (Carney & Cunningham, 2006).

CALIBRATION METHODS FOR TYPE CD

In the case of discrete classification, the best way to know whether a model is uncalibrated according to the number of instance per class (type CD) is to analyse the confusion matrix. The confusion matrix is a visual way of showing the recount of cases of the predicted classes and their actual values. Each column of the matrix represents the instances in a predicted class, while each row represents the instances in an actual class. One benefit of a confusion matrix is that it is easy to see if the system is confusing two classes (i.e. commonly mislabelling one as another). For example, if there are 100 test examples and a classifier, an example of a confusion matrix with three classes *a*, *b* and *c* could be as follows:

		Real		
Predicted		*a*	*B*	*c*
	a	20	2	3
	b	0	30	3
	c	0	2	40

In this matrix from 100 examples, 20 were from class *a* and all of them were well classified, 34 were from class *b*, 30 of them were well classified as *b*, 2 were misclassified as *a* and 2 were misclassified as *c*. Finally, 46 of the examples were from class *c*, 40 of them were well classified as *c*, 3 were misclassified as *a* and 3 were misclassified as *b*. If we group by class, we have a proportion of 20, 34, 46 for the real data, and a proportion of 25, 33, 42 for the predicted data. As we can see, these proportions are quite similar and, therefore, the classifier is calibrated regarding the original class distribution. On the contrary, the following matrix can be considered:

		Real	
Predicted		*a*	*B*
	a	60	2
	b	40	23

In this matrix the proportion of real data are 100, 25, while the proportion of predicted data are 62, 63. So, in this case the model is uncalibrated. One question would be if this type of disproportion is quite common. The answer is that this situation is very common, and normally the disproportion used to be in favour of the majority classes. The second question is whether there are any techniques to solve the problem after obtaining the model. And the answer is 'yes', in general. To do so, the predictions of the models must be accompanied by probabilities or reliability values (or simply, scores). In this case the threshold that splits into the classes can be changed.

We can consider the technique presented by (Lachiche & Flach, 2003). That work is specialized in Bayesian classifiers, but the technique can be applied to any classification model which accompanies its predictions by probabilities or reliabilities. A naïve Bayes classifier estimates the probabilities for each class independently. So, for example, we can have the following probabilities for each class: $p(a|x) = 0{,}2$ and $p(b|x) = 0{,}05$, and there are no more classes, so, the sum of the probabilities is not 1. In general, the naïve Bayes classifiers assigns very low probabilities, because the probability is the product of several factors that, at the end, reduce the absolute values of the probabilities. Nevertheless, this is not the problem. The problem is that the decision rule that is used to apply the model is the next one:

If $p(a|x) > p(b|x)$ then predicts a
else predicts b

The previous rule has as consequence that the result is not calibrated in most of the cases. It is possible that the previous rule produces much more examples of the class a (or vice versa) that there were in the original distribution. The solution to this problem is to estimate a threshold fitted to the original distribution.

If there are only two classes the solution is very easy, we can calculate a ratio of the two probabilities: $r = p(a|x)/p(b|x)$. This ratio comes from 0 to infinite. We can normalise it between 0 and 1 with a sigmoid, if we want. The aim is to obtain a threshold u with the test set where the distribution of the model will be similar to the original distribution. So, the rule changes to:

If $r > u$ then predicts a
else predicts b

Lachiche and Flach (2003) shows that only with an adjustment like this (in that work the threshold adjustment is based in the ROC analysis and it is extended to multiclass) the results of the models can be improved significantly. In particular, from 25 analyzed datasets, this simple optimization improved significantly the accuracy in 10 cases and was reduced only in 4 of them.

Apart from that simple approximation, there are other works where the optimum threshold fitted to the original distribution is calculated.

CALIBRATION METHODS FOR TYPE CP

Another case is the calibration of probabilistic classification models (type CP) and it requires more sophisticated techniques. In this case, the aim is that when the model predicts that the probability of the class a is 0.99, this means that the model is more confident that the class is a that when it is predicted 0.6. Determining the reliability of a prediction is fundamental in a lot of applications: diagnosis, instance selection and model combination.

Apart from the measures introduced in previous sections (MSE, logloss, CalBin and CalLoss), the fundamental tool to analyze the calibration of this type of models is the reliability diagrams (DeGroot & Fienberg, 1982). In those diagrams, the prediction space is discretised into 10 intervals (from 0 to 0.1, from 0.1 to 0.2, etc.). The examples whose probability is between 0 and 0.1 go into the first interval, the examples between 0.1 and 0.2 go into the second, etc. For each interval, the mean predicted value (in other words, the mean predicted probability) is plotted (x axis) in front of the fraction of positive real cases (y axis). If the model is calibrated the points will be near to the diagonal.

The following Figure 1 shows an example of an uncalibrated model and one calibrated model.

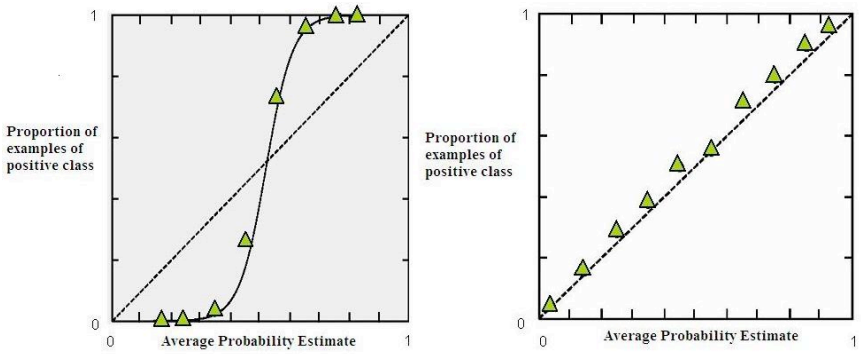


Figure 1. Reliability diagrams. Left: uncalibrated model. Right: calibrated model.

For example, in the left model there are not cases with predicted probability lower than 0.1. The next interval, where the examples have an assigned probability between 0.1 and 0.2 for the positive class (with a mean of 0.17) there are not examples of the positive class. So, these predictions have too high estimated probability. It should be nearer to 0, instead of nearer to 0.17. On the contrary, if we goes to the end of the curve, we will see that the examples with assigned probabilities between 0.7 and 0.8, all of them are from the positive class. Probably, they should have higher probability, because they are surer cases.

On the other hand, in the model on the right, we can see that the correspondence is righter: there are probabilities distributed from 0 to 1 and, moreover, they used to be the same as the percentage of examples.

There are several techniques that can calibrate a model like the left one and transform it in a model like the right one. The most common are: binning averaging, isotonic regression and Platt's method. The objective of these methods (as a postprocessing) is to transform the original estimated probabilities (scores can also be used (Zadrozny & Elkan, 2002)) $p(i, j)$ into calibrated probability estimates $p^*(i, j)$. It is important to remark that all of these general calibration methods can only be used (directly, without approaches) in binary problems, because all of them use the sorted estimated probability to calculate the calibrated probability.

When the calibration function is monotonically nondecreasing (also called isotonic). Most calibration methods presented in the literature are isotonic. This makes it reasonable to use MSE or LogLoss as measures to evaluate calibration methods, since the 'separability components' are not affected. This is clearly seen through the so-called "decompositions of the Brier score" (Sanders, 1963; Murphy, 1972) included in a previous section in this chapter.

Binning Averaging

The first calibration method is called binning averaging (Zadrozny & Elkan, 2001) and consists in sorting the examples in decreasing order by their estimated probabilities and dividing the set into k bins (i.e. subsets of equal size). Then, for each bin l, $1 \leq l \leq k$, the corrected probability estimate for a case i belonging to class j, $p^*(i,j)$, is the proportion of instances in l of class j. The number of bins must be small in order to reduce the variance of the estimates. In their paper, Zadrozny and Elhan fixed k=10 in the experimental evaluation of the method.

Example: Consider the following training set sorted by its probability of membership to positive class grouped in 5 bins.

bin	instance	score
1	e_1	0.95
	e_2	0.94
	e_3	0.91
	e_4	0.90
2	e_5	0.87
	e_6	0.85
	e_7	0.80
	e_8	0.76
3	e_9	0.70
	e_{10}	0.66
	e_{11}	0.62
	e_{12}	0.62
4	e_{13}	0.51
	e_{14}	0.49
	e_{15}	0.48
	e_{16}	0.48
5	e_{17}	0.45
	e_{18}	0.44
	e_{19}	0.44
	e_{20}	0.42

Then, if a new example is assigned a score of 0.68, then it belongs to bin 3 and its corrected probability is $\frac{0.70 + 0.66 + 0.62 + 0.62}{4} = 0.65$.

This is just how binning averaging works.

Isotonic Regression (PAV)

A slightly more sophisticated technique also for two-class problems is isotonic regression. (Ayer, Brunk, Ewing, Reid, & Silverman, 1955) presented a pair-adjacent violators algorithm (PAV) for calculating the isotonic regression. The idea is that calibrated probability estimates must be a monotone decreasing sequence, i.e., $p_1 \geq p_2 \geq ... \geq p_n$. If it is not the case, the PAV algorithm each time that a pair of consecutive probabilities, $p(i,j)$ and $p(i+1,j)$, does not satisfy the above property $p(i,j) < p(i+1,j)$ replaces both of them by their probability average, that is:

$$p^*(i,j) = p^*(i+1,j) = \frac{p(i,j) + p(i+1,j)}{2}$$

This process is repeated (using the new values) until an isotonic set is reached.
Example. The next table shows in the first column the initial scores of one dataset composed by 10 examples. The following columns represent the results obtained in the steps given by the PAV method where the last one contains the calibrated probabilities.

instance	initial score	PAV step 1	PAV step 2
e_1	0.76	0.765	0.765
e_2	0.77	0.765	0.765
e_3	0.70	0.705	0.705
e_4	0.71	0.705	0.705
e_5	0.66	0.685	0.686
e_6	0.71	0.685	0.686
e_7	0.69	0.69	0.686
e_8	0.68	0.68	0.68
e_9	0.48	0.485	0.485
e_{10}	0.49	0.485	0.485

Platt's Method

(Platt, 1999) presents a parametric approach for fitting a sigmoid that maps estimated probabilities into calibrated ones. This method was developed to transform the outputs of a support vector machine (SVM) from the original values $[-\infty, \infty]$ to probabilities, but can be extended to other types of models or probabilities variations. The idea consists on passing to the values a sigmoid function of the form:

$$p^*(i,j) = \frac{1}{1 + e^{A \times p(i,j) + B}}$$

The parameters A and B are determined such that minimise the negative log-likelihood of the data.

Platt's method is most effective when the distortion in the predicted probabilities has a sigmoid form (as in the previous example). Isotonic regression is more flexible and can be applied to any monotonic distortion. Nevertheless, isotonic regression used to present overfitting problems in some cases. Also, all the above methods can use the training set or an additional validation set for calibrating the model. The quality of the calibration might depend on this possibility and the size of the dataset. This is a recurrent issue in calibration, and it has been shown that some methods are better than others for small calibration sets (i.e. Platt's scaling is more effective than isotonic regression when the calibration set is small (Caruana & Niculescu-Mizil, 2004)).

Other Related Calibration Methods

Apart from the methods for obtaining calibrated probabilities, there exits other calibration techniques only applicable to specific learning methods. For instance, smoothing by m-estimate (Cestnik, 1990) and Laplace (Provost & Domingos, 2000) are another alternative ways of improving the probability estimates given by an unpruned decision tree. Probabilities are generated from decision trees as follows. Let *T* be a decision tree and *l* a leaf which contain *n* training

instances. If k of these instances are of one class (for instance, of positive class), then when T is applied to classify new examples it assigns a probability of $p = \frac{k}{n}$ that each example i in l belongs to the positive class. But using the frequencies derived from the count of instances of each class in a leaf might not give reliable probability estimates (for instance, if there are few instances in a leaf). So, for a two-class problem, the Laplace correction method replaces the probability estimate by $p' = \frac{k+1}{n+2}$. For a more general multiclass problem with C classes, the Laplace correction is calculated as $p' = \frac{k+1}{n+C}$. As Zadrozny and Elkan (2001) say "the Laplace correction method adjusts probability estimates to be closer to 1/2, which is not reasonable when the two classes are far from equiprobable, as is the case in many real-world applications. In general, one should consider the overall average probability of the positive class, i.e. the base rate, when smoothing probability estimates" (p. 610). Thus, smoothing by m-estimate consists in replacing the above mentioned probability estimate by $p' = \frac{k + b \cdot m}{n + m}$ where b is the base rate and m is the parameter for controlling the shift towards b. Given a base rate b, Zadrozny and Elkan (2001) suggest using m such that $b \cdot m = 10$.

Another related technique also applicable to decision trees is curtailment (Zadrozny & Elkan, 2001). The idea is to replace the score of a small leaf (i.e., a leaf with few training instances) by the estimate of its parent node, if it contains enough examples. If the parent node still have few examples, we proceed with its parent node and so on until to reach either a node sufficiently populated or the tree root.

CALIBRATION METHODS FOR RD

The regression case when the goal is to have that the expected output to be equal (or close) to the real average output (type RD), has been implicitly or explicitly considered in most regression techniques to date. This is so, because there are two numeric outputs (the predicted value and the real value), so, there is more variety of corrective functions to apply. First we are going to see which the problem is in this case. The idea can be depicted (see Figure 2) if we compare the behaviour of the test data, denoted by "real", and the model that we want to calibrate ("predicted"). In the figure, we show the behaviour of two models with a similar squared error.

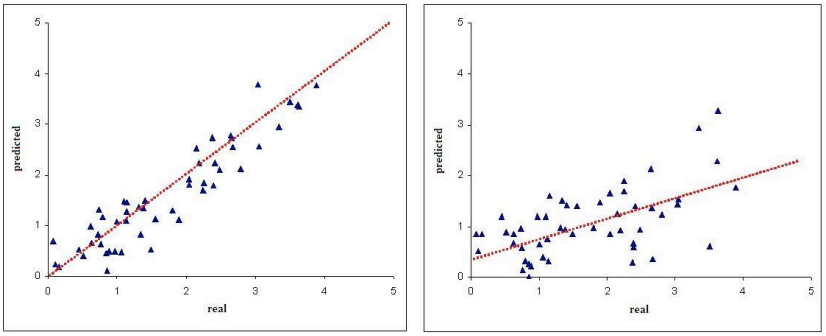


Figure 2. Calibration of regression models. Left: an uncalibrated model. Right: a calibrated model.

As we said in the introduction, the characteristic of a calibrated model for type RD is that the errors are equally distributed for the different output values, in other words, the expected value from distinct functions between the predicted value and the real value must be the right according to the function. For example, the expected value of the difference between the estimated value (y_{est}) and the real value (y) must be near to zero, so $E(y_{est} - y) = 0$. If this value is less than 0, then the real values are a little higher that the estimated ones, in average, if this value is greater than 0, the real values are a little lower than the estimated ones. Most regression models used to have this difference quite well calibrated. On contrary, the expected value of the quotient between the estimated value and the real value should be near to one, so $E(y_{est} / y) = 1$. If this quotient is greater than one, the error used to be positive for high values and negative for low values. Techniques such as linear regression use to give calibrated models, but others (nonlinear regression, local regression, neural networks, decision trees, etc.) can give uncalibrated models.

Logically, in both cases, the errors can be corrected, for example, by decreasing the high values and increasing the low values. In general, the solution comes from obtaining some type of estimation of the decalibration function between the real values and the predicted values. One of the most common approximations consists on calculate a linear regression as in the previous plots in the Figure 2 and apply it to the model, with the aim of fitting the calibration. These calibrations used to increase the mean squared error $\frac{(y - y_{est})^2}{n}$, but can achieve to reduce the relative mean squared error $\frac{(y - y_{est})^2}{(y - mean(y_{est})) \times n}$ or the error tolerance.

When the decalibration function is nonlinear (but it has a pattern), the problem of calibration becomes more complex, and some kind of nonlinear or local regression is needed to calibrate the model. In these cases, it is not properly a calibration process but a meta-learner, with several stages (stacking, cascading, etc.).

CALIBRATION METHODS FOR RP

On type RP, the prediction is a probability density function, and it is this function what we need to improve. This is a much more complex problem since improving this type of calibration can be done by mangling the prediction estimates (i.e. the MSE can be increased as the result of calibrating). Consequently, a trade-off must be found. In (Carney & Cunningham, 2006), they approach the problem by formulating it as a multi-objective problem. The two objectives of sharpness (a classical quality criterion based on negative log-likelihood (NLL) and the Anderson-Darling (A^2) test over the probability integral transform, as a measure of pure calibration.

FUTURE TRENDS

Future trends on calibration include a clearer recognition of the effects calibration has and when and how the four types of calibration are related as well as how they relate to classical quality metrics. In particular, calibration and traditional measures are sometimes conflicting, and we need to use a couple of metrics (such as A^2 and NLL in the case of type RP), or a hybrid one (such as MSE^p in the case of type CP), to select the best model.

In classification, most calibration methods we have analysed work for binary problems. As we have seen, most calibration methods are based on sorting the instances and/or making bins. But for more than two classes it is not so clear how to sort the instances or, more generally, how to make the bins. This is one of the reasons for which.the calibration problem has been less discussed. There are some works like (Zadrozny & Elkan, 2002) where the multiclass calibration problem has been studied,

but using approaches for reducing a multiclass problem to a set of binary problems and for finding an approximate solution for this problem.

Another interesting research line consists in to study in depth the relationship between ROC analysis (or its counterpart for regression, REC analysis) and calibration. For instance, the use of repairing concavities techniques in ROC analysis (Flach & Wu, 2005) to solve conflicts between the original class ranking and the new estimated probabilities.

Finally, type RP is a future trend on its own, since it is the most complex case which has been paid attention much more recently.

CONCLUSIONS

In this chapter we have addressed the problem of predictive model calibration and we have presented the most known calibration techniques for classification and regression.

We have shown that for classification, there are evaluation measures which are suitable for calibration (such as logloss, MSE, ...). And, also, there are another measures (for instance, accuracy) which are not good. Other measures are used in conjunction with calibration measures, especially the separability measures (AUC). Similarly, in regression, specific measures are needed to evaluate calibration, although must be usually accompanied by measures of sharpness.

Calibration techniques are usually based on deriving a transformation which converts the values (on types CD and RD) or the probabilities (on types CP and RP) to better estimates. Very different transformation techniques have been devised in the literature, but they usually include some kind of binning or sorting in discrete cases (classification) or some kind of integral in the continuous cases (regression).

REFERENCES

Ayer, M., Brunk, H., Ewing, G., Reid, W., & Silverman, E. (1955). An empirical distribution function for sampling with incomplete information. *Annals of Mathematical Statistics*, 5, 641–647.

Carney, M., & Cunningham, P. (2006). Making good probability estimates for regression. *17th European Conference on Machine Learning*, LNCS: Vol. 4212 (pp. 582-589).

Caruana, R., & Niculescu-Mizil, A.(2004). Data mining in metric space: an empirical analysis of supervised learning performance criteria. *Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp 69–78).

Cestnik, B. (1990). Estimating probabilities: A crucial task in machine learning. *Ninth European Conference on Artificial Intelligence* (pp. 147–149).

DeGroot, M. & Fienberg, S. (1982). The comparison and evaluation of forecasters. *Statistician*, 31(1), 12–22.

Dowe, D. L., Farr, G. E., Hurst, A. J., & Lentin, K. L. (1996). Information-theoretic football tipping. *3rd Conference on Maths and Computers in Sport* (pp 233-241).

Fawcett, T. (2006), An introduction to ROC analysis. *Pattern Recognition Letters*, 27, 861–874.

Fawcett, T. & Niculescu-Mizil, A. (2007) PAV and the ROC convex hull. *Machine Learning*, 68(1), 97–106.

Flach, P. A., & Matsubara, E. T. (2007). A simple lexicographic ranker and probability estimator. *18th European Conference on Machine Learning* (pp. 575–582).

Flach, P.A., & Wu, S. (2005). Repairing concavities in ROC curves. *International Joint Conference on Artificial Intelligence (IJCAI'05)* (pp. 702–707).

Good, I. J. (1952). Rational decisions. *Journal of the Royal Statistical Society*. Series B 14, 107-114.

Good, I. J. (1968). Corroboration, explanation, evolving probability, simplicity, and a sharpened razor. *British Journal of the Philosophy of Science*. 19, 123-143.

Lachiche, N., & Flach, P.A. (2003) Improving Accuracy and Cost of Two-class and Multi-class Probabilistic Classifiers Using ROC Curves. *International Conference on Machine Learning* (pp. 416-423).

Murphy, A. H. (1972). Scalar and vector partitions of the probability score: Part ii. n-state situation. *Journal of Applied Meteorology*, 11:1182–1192.

Platt, J. C. (1999). Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In P. J. Bartlett, B. Schölkopf, D. Schuurmans, and A. J. Smola, editors, *Advances in Large Margin Classifiers* (pp. 61–74). MIT Press, Boston.

Provost, F., & Domingos, P. (2000). *Well-trained PETs: Improving probability estimation trees* (Technical Report CDER #00-04-IS). Stern School of Business, New York University.

Sanders, F. (1963). On subjective probability forecasting. *Journal of Applied Meteorology*, 2, 191-201.

Zadrozny, B., & Elkan, C. (2001). Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers. *Eighteenth International Conference on Machine Learning* (pp. 609–616).

Zadrozny, B., & Elkan, C. (2002). Transforming classifier scores into accurate multiclass probability estimates. *Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 694–699).

KEY TERMS AND THEIR DEFINITIONS

Calibration technique: is any postprocessing technique which aims at improving the probability estimation or to improve error distribution of a given model.

Distribution calibration in classification (or simply "class calibration"): the degree of approximation of the true or empirical class distribution with the estimated class distribution.

Probabilistic calibration for classification: the degree of approximation of the predicted probabilities to the actual probabilities.

Distribution calibration in regression: the relation between the expected value of the estimated value and the mean of the real value must be unbiased at the global and local levels.

Probabilistic calibration for regression: when we have "density forecasting" models, a good calibration requires in general that these density functions are particular for each prediction, narrow when the prediction is confident and broader when it is less so.

Calibration measure: any kind of quality function which is able to assess the degree of calibration of a predictive model.

Confusion matrix: is a visual way of showing the recount of cases of the predicted classes and their actual values. Each column of the matrix represents the instances in a predicted class, while each row represents the instances in an actual class.

Reliability diagrams. In these diagrams, the prediction space is discretised into 10 intervals (from 0 to 0.1, from 0.1 to 0.2, etc.). The examples whose probability is between 0 and 0.1 go into the first interval, the examples between 0.1 and 0.2 go into the second, etc. For each interval, the mean predicted value (in other words, the mean predicted probability) is plotted (x axis) in front of the fraction of positive real cases (y axis). If the model is calibrated the points will be near to the diagonal.

BIOGRAPHY

Antonio Bella finished his degree in Computer Science at the Technical University of Valencia in 2004 and started PhD studies in Machine Learning at the Department of Information System and Computation in the same university. At the same time, in 2005 he obtained a MSc in Corporative Networks and Systems Integration and in 2007 he started a degree in Statistical Science and Technology at the University of Valencia.

Cèsar Ferri is an associate professor of computer science at the Department of Information Systems and Computation, Technical University of Valencia, Spain, where he has been working since 1999. He obtained his BSc at the Technical University of Valencia, and his MSc at the University of Pisa, Italy. His research interests include machine learning, cost-sensitive learning, relational data mining, and declarative programming. He has published several journal articles, books, book chapters and conference papers on these topics.

José Hernández-Orallo, BSc, MSc (Computer Science, Technical University of Valencia, Spain), MSc (Computer Science, ENSEA, Paris), Ph.D. (Logic, University of Valencia). Since 1996, he has been with the Department of Information Systems and Computation, Technical University of Valencia, where he is currently an Associate Professor. His research interests centre on the areas of artificial intelligence, machine learning, data mining, data warehousing and software engineering, with several books, book chapters, journal and conference articles on these topics.

María José Ramírez-Quintana received the BSc from the University of Valencia (Spain) and the Msc and PhD in Computer Science from the Technical University of Valencia (Spain). She is currently an associate professor at the Department of Information Systems and Computation, Technical University of Valencia. She has lectured several courses on software engineering, functional and logic programming, and multiparadigm programming. Her research interest include mutiparadigm programming, machine learning, data mining algorithms, model combination and evaluation, and learning from structured data, with more than 60 publications in these areas, including journal articles, books, book chapters and conference contributions.

7.4 Data Mining Strategies for CRM Negotiation Prescription Problems

4. A. Bella, C. Ferri, J. Hernández-Orallo, and M.J. Ramírez-Quintana. **Data Mining Strategies for CRM Negotiation Prescription Problems.** In *Trends in Applied Intelligent Systems (IEA/AIE)*, volume 6096 of *Lecture Notes in Computer Science*, pages 520–530. Springer Berlin / Heidelberg, 2010.

Data Mining Strategies for CRM Negotiation Prescription Problems *

A. Bella, C. Ferri, J. Hernández-Orallo, and M.J. Ramírez-Quintana

DSIC-ELP, Universidad Politécnica de Valencia, Camí de Vera s/n, 46022 Valencia, Spain

Abstract. In some data mining problems, there are some input features that can be freely modified at prediction time. Examples happen in retailing, prescription or control (prices, warranties, medicine doses, delivery times, temperatures, etc.). If a traditional model is learned, many possible values for the special attribute will have to be tried to attain the maximum profit. In this paper, we exploit the relationship between these modifiable (or negotiable) input features and the output to (1) change the problem presentation, possibly turning a classification problem into a regression problem, and (2) maximise profits and derive negotiation strategies. We illustrate our proposal with a paradigmatic Customer Relationship Management (CRM) problem: maximising the profit of a retailing operation where the price is the negotiable input feature. Different negotiation strategies have been experimentally tested to estimate optimal prices, showing that strategies based on negotiable features get higher profits.

1 Introduction

In data mining, problem features (or attributes) have been usually classified as input and output features. A problem is said to be supervised if it has output features, and it is said to be unsupervised if it does not have output features. Input features can be of many kinds: numerical, nominal, structured, etc. In fact, many data mining methods have been specialised to specific kinds of input features. In supervised learning, it is usually assumed that the goal of a model is to predict an output value given an input. Consequently, a function is learned from inputs to outputs, which is eventually applied to new cases.

However, in many application areas not all input feature values are given. This does not mean that they are unknown (i.e., null), but that they can be modified or fine-tuned at prediction time. Consider a typical data mining problem: a loan granting model where loans are granted or not according to a model which has been learnt from previous customer behaviours. It is generally assumed that given the input feature values (the customer's personal information, the loan amount, the operation's data, etc.) the model will provide an output (yes/no, a probability, a profit, etc.). But in real scenarios, the decision of whether the loan

* This work has been partially supported by the EU (FEDER) and the Spanish MEC/MICINN, under grant TIN 2007-68093-C02 and the Spanish project "Agreement Technologies" (Consolider Ingenio CSD2007-00022).

must be granted or not might change if one or more of the input feature values can be changed. Perhaps, a loan cannot be granted for 300,000 euros, but it can be granted for 250,000 euros. If the customer asks the bank's clerk "what is the maximum amount you can grant for this operation?", we have a sort of *inverse* problem. Since the model is not generally an analytical one (it is not generally a linear regression model but a neural network, SVM, decision tree or other kind of difficult-to-inverte models), the only possibility to give a precise answer to the customer is to try all the possible input combinations for the loan amount and find out the maximum value which is granted. After this inefficient and ad-hoc process, the clerk can give an answer such as: "According to our loan models, we can grant a maximum of 278,304 euros". Apart from this inefficiency, there is another more serious problem: the clerk knows that the model will give a negative answer to any amount above this maximum, for instance, 278,305 euros, which makes this loan a quite risky one.

This typical example shows that some input features are crucial in the way that they can be freely modified at prediction time. The existence of these special attributes makes it quite inefficient to develop a classification/regression model in the classical way, since whenever there is a new instance hundreds of possible values have to be tried for the special attribute in order to see which combination can attain the maximum profit, or, as the previous example, the maximum risk. Additionally, these features are frequently associated to negotiation scenarios, where more than one attempt or offer have to be made, by suitably choosing different values for this special input feature.

In this paper we analyse these special features that we call "negotiable features", and how they affect data mining problems, its presentation and its use for confidence probability estimation and decision making. As we will see, we can exploit the relation between these input features and the output to change the problem presentation. In this case, a classification problem can be turned into a regression problem over an input feature [1]. Also, we present a first general systematic approach on how to deal with these features in supervised models and how to apply these models in real negotiation scenarios where there are several attempts for the output value.

The paper is organised as follows. In Section 2 we focus on classification problems with one numerical negotiable feature, since this is the most frequent and general case, and it also includes prototypical cases, when the negotiable feature is price or time. We present a general approach to the inversion problem which transforms the classification problem into a regression one, where the negotiable feature is placed as the output. In Section 3, we describe a specific real scenario (an estate agent's) where negotiation can take place. We also develop some negotiation strategies using the previous approaches in Section 4. In Section 5, we experimentally evaluate models and negotiation strategies with a baseline approach. Section 6 includes the conclusions and future work.

2 Inverting Problem Presentation

As we have mentioned in the introduction, there are many data mining problems where one or more input attributes, we call negotiable features, can be modified

at application time. Imagine a model which estimates the delivery time for an order depending on the kind of product and the units which are ordered. One possible (traditional) use of this model is to predict the delivery time given a new order. However, another use of this model is to determine the number of units (provided it is possible to play with this value) that can be delivered in a fixed period of time, e.g. one week. This is an example of an "inverse use" of a data mining model, where all inputs except one and the output are fixed, and the objective is to determine the remaining input value.

The inversion problem can be defined as follows. Consider a supervised problem, where input attribute domains are denoted by X_i, $i \in \{1, \ldots, m\}$, and the output attribute domain is denoted by Y. We denote the target (real) function as $f : X_1 \times X_2 \times \ldots \times X_m \rightarrow Y$. Values for input and output attributes will be denoted by lowercase letters. Hence, labelled instances are then tuples of the form $\langle x_1, x_2, \ldots, x_m, y \rangle$ where $x_i \in X_i$ and $y \in Y$. The inversion problem consists in defining the function $f^I : X_1 \times \ldots \times X_{i-1} \times Y \times X_{i+1} \times \ldots \times X_m \rightarrow X_i$, where X_i is the negotiable feature. In the above example, f is the function that calculates the delivery time of an order, the negotiable feature X_i is the number of delivered units and f^I calculates this number by considering fixed the delivery time.

In the inverting problem the property that we call *sensitive* is satisfied. Fixing the values of all the other input attributes $X_j \neq X_i$ for at least n examples from the dataset D ($n \leq |D|$ being n determined by the user depending on the problem, the presence of noise, ...), there are two different values for X_i producing different output values.

We also assume a monotonic dependency between the input attribute X_i and the output. This dependency is defined under the assumption that there is a strict total order relation for the output, denoted by $\prec$, such that for every two different possible values $y_a, y_b \in Y$, we have that either $y_a \prec y_b$ or $y_b \prec y_a$. This order usually represents some kind of profit, utility or cost. For numerical outputs, $\prec$ is usually the order relation between real numbers (either $<$ or $>$, depending on whether it is a cost or profit). For nominal outputs, $\prec$ usually sets an order between the classes. For binary problems, where POS and NEG represent the positive and negative class respectively, we can just set that $NEG \prec POS$. For more than two classes, the order relation can be derived from the cost of each class. Analogously, there is also a total order relation for the input denoted as $\preceq$. Based on this order, we can establish a monotonic dependency between the input and the output features. Thus, $\forall a, b \in X_i$, if $a \preceq b$ then $f(x_1, \ldots, x_{i-1}, a, x_{i+1}, \ldots, x_m) \prec f(x_1, \ldots, x_{i-1}, b, x_{i+1}, \ldots, x_m)$ (monotonically increasing) or $\forall a, b \in X_i$, if $a \preceq b$ then $f(x_1, \ldots, x_{i-1}, a, x_{i+1}, \ldots, x_m) \succeq f(x_1, \ldots, x_{i-1}, b, x_{i+1}, \ldots, x_m)$ (monotonically decreasing).

The inversion problem is well-known [1] and seems simple at first sight, but many questions arise. First, is f^I also a function? In other words, for two different values for X_i we may have the same value for Y which will ultimately translate into two inconsistent examples for f^I (two equal inputs giving different outputs). Second, the fact that we have an example saying that a given loan amount

was granted to a customer does not mean that this is the maximum amount that could be granted to the customer. Third, deriving probabilities to answer questions such as "which loan amount places this operation at a probability of 0.95 of being a profitable customer?" seem to be unsolvable with this new presentation.

But if we take a closer look at these issues, we see that although relevant, there is still a possibility behind this problem presentation change. First, many regression techniques work well for inconsistent examples, so this is not a big practical problem. Second, it is true that cases do not represent the maximum amount, but in many cases the examples represent deals and they are frequently not very far away from the maximum. Or, in the worst case, we can understand the new task as "inferring" the typical value for X_i such that the loan is granted to the customer. And third, we *can* also obtain probabilities in a regression problem. We extend this idea further below.

If we invert the problem, how can we address the original problem again? With the original model and for only two classes, it can be done by calculating $p(POS|\langle x_1, \ldots, x_{i-1}, a, x_{i+1}, \ldots, x_m\rangle)$, for any possible value $a \in X_i$. From the inverted (regression) problem, we get: $\hat{a} = f^I(x_1, \ldots, x_{i-1}, POS, x_{i+1}, \ldots, x_m)$. If we think of $\hat{a}$ as the predicted maximum or minimum for a which makes a change on the class, a reasonable assumption is to give 0.5 probability for this, that is $p(POS|\langle x_1, \ldots, x_{i-1}, \hat{a}x_{i+1}, \ldots, x_m\rangle) = 0.5$.

The next step is to assume that the output for f^I follows a distribution with centre at $\hat{a}$. For instance, we can assume a normal distribution with mean at $\hat{a}$ and use the standard error (mean absolute error, mae, on the training set) as standard deviation σ. In other words, we use $N(\hat{a}, mae^2)$. Figure 1 shows an example of a normal distribution with centre at $\hat{a} = 305,677.9$ and standard deviation $\sigma = 59,209.06$ and its associated cumulative distribution function.

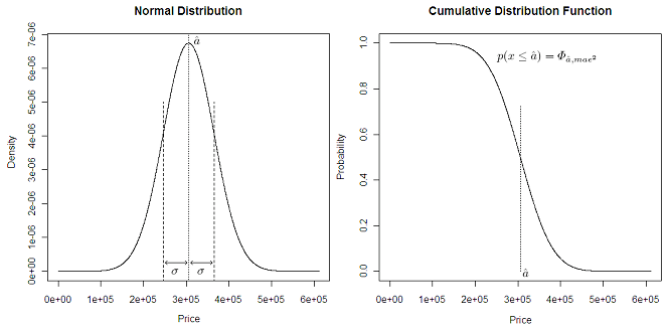


Fig. 1. Left: Example of a normal distribution $\hat{a} = 305,677.9$ and $\sigma = 59,209.06$. Right: Associated cumulative distribution function.

From here, we can derive the probability for any possible value a as the cumulative distribution function derived from the above normal, i.e., $\Phi_{\hat{a},mae^2}$.

Consequently, for solving the original problem, (1) we solve the inversion problem directly and (2) we use the predicted value of the negotiable feature as

mean of a normal distribution with the standard error as standard deviation. We call this model *negotiable feature model*.

3 Negotiation using Negotiable Feature Models: A Real Scenario

We are going to illustrate the approach we have presented in the previous section in a real example. In particular, we have studied the problem of retailing, where the (negotiable) input feature is the price (denoted by π) and the problem is a classification problem (buying or not).

We present an example using real data from an estate agent's, which sells flats and houses they have in their portfolio. We have several conventional attributes describing the property (squared metres, location, number of rooms, etc.), and a special attribute which is our negotiable feature, price. We will use the term "product" for properties to make it clear that the case is directly extensible to virtually any retailing problem where price is negotiable.

We start with the simplest negotiation scenario, where there are only one seller and one buyer who both negotiate for one product. One buyer is interested in one specific product. S/he likes the product and s/he will buy the product if its price is under a certain price that s/he is willing to pay for this product.In fact, if we reduce price to 0, the probability of having class POS approaches 1 and if we increase price to a very large amount, the probability of having class NEG approaches 1. Moreover, the relation between price and the class order $NEG \prec POS$ is monotonically decreasing.

Additionally, in our problem, the seller has a "minimum price" (denoted by π_{min}), which corresponds to the price that the owner has set when the product was included in the portfolio plus a quantity that the seller sets as fixed and variable costs. Any increment over this minimum price is profitable for the seller. Conversely, selling under this value is not acceptable for the seller. Therefore, the seller will not sell the product if its price is under this minimum price that s/he knows. Finally, the profit obtained by the product will be the difference between the selling price minus the minimum price: $Profit(\pi) = \pi - \pi_{min}$.

Obviously, the goal of the seller is to sell the product at the maximum possible price (denoted by π_{max}) which is the value such that the following equalities hold:

$f(x_1, \ldots, x_{i-1}, \pi_{max}, x_{i+1}, \ldots, x_m) = POS$

$f(x_1, \ldots, x_{i-1}, \pi_{max} + \epsilon, x_{i+1}, \ldots, x_m) = NEG, \forall \epsilon > 0$

In other words, the use for the model is: "Which is the maximum price that I can sell this product to this customer?". Logically, the higher the price the lower the probability, so the goal is more precisely to maximise the *expected profit*, which is defined as follows:

$$E\_Profit(\pi) = \hat{p}(POS|\langle x_1, \ldots, x_{i-1}, \pi, x_{i+1}, \ldots, x_m\rangle) \cdot Profit(\pi) \quad (1)$$

where $\hat{p}$ is the estimated probability given by the negotiable feature model.

To ease notation we will denote $\hat{p}(POS|\langle x_1, \ldots, x_{i-1}, \pi, x_{i+1}, \ldots, x_m\rangle)$ as $\hat{p}(POS|\pi)$, consequently, we can express (1) as:

$$E\_Profit(\pi) = \hat{p}(POS|\pi) \cdot Profit(\pi) \quad (2)$$

with the additional constraint, as mentioned, that $\pi \geq \pi_{min}$.

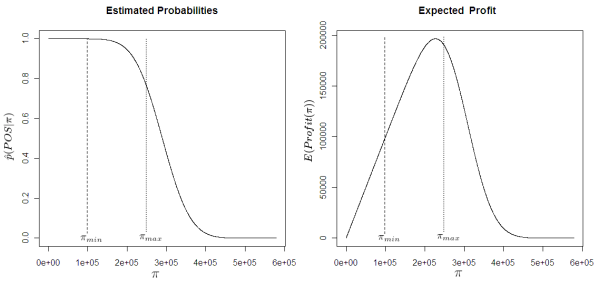


Fig. 2. Left: Example of estimated probabilities. Right: Associated expected profit. The minimum and maximum price are also shown.

So, if we have a model which estimates probabilities for the positive class, we can use formula (2) to choose the price that has to be offered to the customer. If probabilities are well estimated, for all the range of possible prices, this must be the optimal strategy. In Figure 2 an example of the plots that are obtained for the estimated probabilities and expected profit is shown.

But this is the case where we have one bid (one offer). In many negotiation scenarios, we have the possibility of making several bids, as in bargaining. In this situation, it is not so direct how to use the model in order to set a sequence of bids to get the maximum overall expected profit. For instance, if we are allowed three bids, the overall expected profit of a sequence of bids is defined as:

$E\_Profit(\langle \pi_1, \pi_2, \pi_3 \rangle) = \hat{p}(POS|\pi_1) \cdot Profit(\pi_1) + (1 - \hat{p}(POS|\pi_1)) \cdot \hat{p}(POS|\pi_2) \cdot Profit(\pi_2) + (1 - \hat{p}(POS|\pi_1)) \cdot (1 - \hat{p}(POS|\pi_2)) \cdot \hat{p}(POS|\pi_3) \cdot Profit(\pi_3)$,
where $\pi_1 > \pi_2 > \pi_3 \geq \pi_{min}$.

4 Negotiation Strategies

In the scenario described in Section 3 the seller is the agent who uses the negotiable feature models to guide the negotiation, while the buyer can only make the decision of buying or not the product.

When we have one possible offer, it is not sensible to set the price at the maximum price that our model predicts it can be sold, because in many cases, because of the prediction error, it will be overestimated and we will not sell the product. On the contrary, selling at the minimum price ensures that we sell as many products as possible, but we get minimum profit as well. In Section 3 we saw that an appropriate way of doing this is by using the expected profit.

Obviously, if the maximum price for the buyer is lower than the minimum price for the seller, the product is not sold. We will exclude these cases, since any strategy is not going to work well for them and it is not going to make any difference to include them or not in terms of comparison.

When we have more than one possible offer, we start with a first offer and if the price is less or equal than a price which is accepted by the buyer, s/he will

buy the product. Otherwise, the seller can still make another offer and follow the negotiation.

It is clear that there exists an optimum solution to this problem when the seller can make "infinite" offers to the buyer, but it is inefficient and unfeasible. This idea consists in beginning the negotiation with a very high offer and make offers of one euro less each time, until the (patient and not very intelligent) buyer purchases the product or until the price of the product is the minimum price. In this case the product would be sold by its maximum price, because the buyer purchases the product when the price offered was equal to the maximum price considered by the buyer.

In what follows we propose several strategies. One is the "baseline" method which is typically used in real estate agent's. For cases with one single bid, we introduce the strategy called "Maximum Expected Profit" (MEP), which is just the application of the expected profit as presented in the previous section. For cases with more bids (multi-bid) we present two strategies: "Best Local Expected Profit" (BLEP) strategy and "Maximum Global Optimisation" (MGO) strategy. Let us see all of them in detail below:

- Baseline method (1 bid or N bids). One of the simplest methods to price a product is to increase a percentage to its minimum price (or base cost). Instead of setting a fix percentage arbitrarily, we obtain the percentage (called α) such that it obtains the best result for the training set. For example if we obtain that the best α is 0.4, it is expected that the best profit will be obtained increasing in 40% the minimum price of the properties. If we have only 1 bid, we will increase the minimum price of the flat by α. But, if we have N bids, we will have one half of the bids with a value of α less than the calculated α and the other half of the bids with a value of α greater than the calculated α. In particular, the value of α will increase or decrease by $\alpha/(N+1)$ in each bid. For example, for 3 bids and the previous sample the three values of α for three bids would be 50%, 40% and 30%. Therefore, the first offer would be an increase of 50% over the minimum price of the product, the second an increase of 40% and the third an increase of 30%.
- Maximum Expected Profit (MEP) strategy (1 bid). This strategy is typically used in marketing when the seller can only make one offer to the customer. Each price for an instance gives a probability of buying. This strategy chooses the price that maximises the value of the expected profit. $\pi_{MEP} = argmax_{\pi}(E\_Profit(\pi))$.
- Best Local Expected Profit (BLEP) strategy (N bids). This strategy consists in applying the MEP strategy iteratively, when it is possible to make more that one offer to the buyer. The first offer is the MEP, and if the customer does not accept the offer, his/her curve of estimated probabilities is normalised taking into account the following: the probabilities of buying that are less than or equal to the probability of buying at this price will be set to 0; and the probabilities greater than the probability of buying at this price will be normalised between 0 and 1. The next offer will be calculated by applying the MEP strategy to the normalised probabilities. In the case

of the probability of buying which is associated to the price is the maximum probability, it will not be set to 0, because the expected profit would always be 0. Instead of this, the next offer is directly the half of the price. The pseudo-code is in Algorithm 1.

– Maximum Global Optimisation (MGO) strategy (N bids). The objective of this strategy is to obtain the N offers that maximise the expected profit by generalising the formula that we have presented in Section 3:
$\pi_{MGO} = argmax_{\langle \pi_1,\ldots,\pi_N \rangle}(E\_Profit(\langle \pi_1,\ldots,\pi_N \rangle) = argmax_{\langle \pi_1,\ldots,\pi_N \rangle}(\hat{p}(POS|\pi_1) \cdot Profit(\pi_1) + (1-\hat{p}(POS|\pi_1)) \cdot \hat{p}(POS|\pi_2) \cdot Profit(\pi_2) + \ldots + (1-\hat{p}(POS|\pi_1)) \cdot \ldots \cdot (1-\hat{p}(POS|\pi_{N-1})) \cdot \hat{p}(POS|\pi_N) \cdot Profit(\pi_N))$.
Getting the N bids from the previous formula is not direct but can be done in several ways. One option is just using a Montecarlo approach with a sufficient number of tuples to get the values for the prices that maximise the expected profit.

Algorithm 1: BLEP strategy

Require: N, epf (estimated probability function or curve)
Ensure: π_{BLEB}
  $\forall x, epf(x) \leftarrow \hat{p}(POS|x)$
  $\pi_1 \leftarrow \pi_{MEB}$
  $\pi \leftarrow \pi_1$
  for $\pi_i, i \in 2..N$ **do**
    if $epf(\pi) \neq \max_{x \in 0..\infty}(epf(x))$ **then**
      $\forall x, epf(x) \leftarrow 0$
      if $epf(x) \leqslant epf(\pi)$ **then**
        $epf \leftarrow normalise(epf, epf(\pi), \max_{x \in 0..\infty} epf(x))$
        $\{normalise(f(x), mix, max)$: returns normalised function of $f(x)$ from values min and max to [0..1]$\}$
      end if
      $\pi_i \leftarrow \pi_{MEB}$
      $\pi \leftarrow \pi_i$
    else
      $\pi_i \leftarrow \pi \div 2$
      $\pi \leftarrow \pi_i$
    end if
  end for
  $\pi_{BLEB} \leftarrow \langle \pi_1, \ldots, \pi_N \rangle$

5 Experiments

5.1 Experimental Settings

Experiments have been performed by using real data collected from an estate agent's. We have information of 2,800 properties (flats and houses) that were sold in the last months, for which we have the following attributes ("district", "number of rooms", "square metres" and the "owner's price"). The "owner's price" is the price which the owner wants to obtain for the property.

In the experiments we have assumed that the "owner's price" is some kind of "market price" and we have considered that it is the "maximum price". Although it is not always true because in some cases the buyer could have paid more than this for the property.

We have randomly split the dataset into a training set and a test set. 10% of the data are for training and the rest to test. This tries to simulate a realistic

situation when there are not too many data for training. Therefore, the results refer to 2,520 properties, and learning is made from 280 flats. We applied the solutions proposed in Section 2 to the data. In particular we have used a $J48$ decision tree[1] (with Laplace correction and without pruning) implemented in the data mining suite WEKA [3]. Since the predicted probability curve given by a classifier (such as the J48 classifier) typically shows discontinuities and strong steps when varying a negotiable feature, we have smoothed it with a low-pass filter with Bartlett overlapping window [2]. The parameter of the window has been set to the "minimum price" divided by 400. The "inversion problem" solution has been implemented with the $LinearRegression$ and $M5P$ regression techniques, also from WEKA.

These three learning techniques have been used to guide the three negotiation strategies explained in Section 4 (for the MGO strategy we used a Montecarlo approach using 1,000 random triplets) and they are compared to the two baseline methods also mentioned in Section 4. In the experiments the number of bids is either one or set to three, i.e., $N = 3$. Summing up, we have nine negotiation methods based on learning techniques and also two baseline methods (without learning process) using the best possible α (80% for one bid and the triplet $\langle 100\%, 80\%, 60\% \rangle$ for three bids).

5.2 Experimental Results

In Table 1 we can observe the results obtained for each method, in terms of number of sold properties, total sold price (in euros) and total profit (in euros).

Table 1. Results obtained by the negotiation strategies, baseline methods and reference methods (minimum and maximum profit). Sold price and profit measured in euros.

Method	Sold flats	Sold price	Profit
All flats sold at π_{min}	2,520	356,959,593	0
All flats sold at π_{max}	2,520	712,580,216	355,620,623
1 bid			
Baseline (80%)	1,411	200,662,464	89,183,317
MEP ($J48$)	1,360	302,676,700	129,628,471
MEP ($LinearRegression$)	1,777	354,973,300	159,580,109
MEP ($M5P$)	1,783	358,504,700	161,736,313
3 bids			
Baseline (100%, 80%, 60%)	1,588	264,698,467	124,483,288
BLEP ($J48$)	1,940	382,921,400	173,381,116
BLEP ($LinearRegression$)	2,056	400,953,200	174,832,025
BLEP ($M5P$)	2,063	404,009,700	176,874,221
MGO ($J48$)	1,733	390,529,770	176,020,611
MGO ($LinearRegression$)	1,918	475,461,200	232,600,223
MGO ($M5P$)	1,906	476,171,900	234,259,509

As we see in Table 1 all the negotiation methods outperform the baseline methods. For one bid, MEP is clearly better than the baseline method. For three bids, both BLEP and MGO are much better than the baseline method. Overall, MGO makes a global optimisation and hence get better results.

[1] In this problem, we only have data of sold properties (positive class), therefore, we have generated examples of the negative class with a price higher than the "owners's price".

On the other hand, the regression techniques outperform the $J48$ decision tree, so, for this problem the solution of inverting problem presentation outperforms the improved classifier solution. This is especially dramatic for MGO, which is the method that depends most on a good probability estimation. This means that the inversion problem using a normal distribution to get the estimated probabilities turns out to be a very useful approach.

6 Conclusions

This paper introduces a number of new contributions in the area of data mining and machine learning which can be useful for many application areas: retailing, control, prescription, and others where negotiation or fine-tuning can take place. Nonetheless, the implications can affect other decision-making problems where data mining models are used.

The first major contribution is the analysis of the relation between negotiable features and problem presentation, and more specifically the inversion problem which happens naturally when we have to modify or play with the negotiable attribute. We have seen that using the monotonic dependency we can change of problem presentation to do the inversion problem (such as changing a classification problem into a regression), where the original problem is now indirectly solved by estimating probabilities using a normal distribution.

The second major contribution is its application to real negotiation problems. We have developed several negotiation strategies and we have seen how they behave for one or more bids in a specific problem of property selling. We have shown that our approach highly improves the results of the classical baseline method (not using data mining) which is typical in this area. In the end, we show that the change of problem presentation (from classification into regression problem, using the negotiable feature, price, as output) gets the best results for the case with one bid but also for the case with three bids.

Since this work introduces new concepts and new ways of looking at some existing problems, many new questions and ideas appear. For instance, we would like to analyse how to tackle the problem when there are more than one negotiable feature at a time, especially for the inversion problem approach (since we would require several models, one for each negotiable feature).

Finally, more complex negotiation strategies and situations can be explored in the future: the customer can counter-offer, several customers, etc.

References

1. L. Devroye, L. Györfi, and G. Lugosi. *A Probabilistic Theory of Pattern Recognition (Stochastic Modelling and Applied Probability)*. Springer, 1997.
2. E.W. Weisstein. *CRC concise encyclopedia of mathematics*. CRC Press, 2003.
3. I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Elsevier, 2005.

7.5 Quantification via Probability Estimators

5. A. Bella, C. Ferri, J. Hernández-Orallo, and M.J. Ramírez-Quintana. **Quantification via Probability Estimators.** *IEEE International Conference on Data Mining*, 0:737–742, 2010.

Quantification via Probability Estimators *

A. Bella, C. Ferri, J. Hernández-Orallo, and M.J. Ramírez-Quintana

DSIC-ELP, Universitat Politècnica de València, Camí de Vera s/n, 46022 València, Spain

Abstract. Quantification is the name given to a novel machine learning task which deals with correctly estimating the number of elements of one class in a set of examples. The output of a quantifier is a real value; since training instances are the same as a classification problem, a natural approach is to train a classifier and to derive a quantifier from it. Some previous works have shown that just classifying the instances and counting the examples belonging to the class of interest (*classify & count*) typically yields bad quantifiers, especially when the class distribution may vary between training and test. Hence, adjusted versions of *classify & count* have been developed by using modified thresholds. However, previous works have explicitly discarded (without a deep analysis) any possible approach based on the probability estimations of the classifier. In this paper, we present a method based on averaging the probability estimations of a classifier with a very simple scaling that does perform reasonably well, showing that probability estimators for quantification capture a richer view of the problem than methods based on a threshold.

1 Introduction

George Forman [4][5][6] has introduced and systematised a new supervised machine learning task called 'quantification'. Quantification is defined as follows: "given a labeled training set, induce a *quantifier* that takes an unlabeled test set as input and returns its best estimate of the class distribution."[5]. For instance, consider a bank that has a credit risk assessment model (possibly a machine learning classifier), and it is assigned a new portfolio of customers (e.g., 100,000 new customers who originated from an agreement with a retailing company). A very important (and classical) question is to determine which customers the credits will be granted to. However, before resolving all of these specific decision making problems, the bank will typically require an assessment of how many credits it will grant, i.e., the bank will need to quantify how many of the customers in the portfolio will have their credit approved. The accuracy of this quantification is critical to assigning human and economical resources, long before any specific decision is made. Also, the result of this estimation may affect

* This work has been partially supported by the EU (FEDER) and the Spanish MEC/MICINN, under grant TIN 2007-68093-C02, the Spanish project "Agreement Technologies" (Consolider Ingenio CSD2007-00022) and the GVA project PROMETEO/2008/051.

the thresholds that may be established for each particular operation since the assessment of the global risk can affect the way in which each local risk is managed (e.g. the policies will probably change if the plan was to work with 25,000 positive results but we expect 50,000). The quantification problem can be found in almost any area in data mining, such as failure detection, medical diagnosis, customer-relationship management, and retailing.

The task is closely related to classification since examples have the same presentation (several input features and a nominal output feature), but it is different in that we are not interested in the specific predictions for each instance of an application dataset, but on the overall count of elements of a particular class. Consequently, quantification is applied to a *batch* of examples, and not to a single example alone. Since the output of the quantification problem is a real value, it has a relation to regression, but the input of the regressor would be a single example rather than a set of examples.

Quantification is a very frequent problem in real applications, so it is somewhat surprising that nobody in the data mining community, until George Forman, recognised, baptised, and addressed this task on its own. In [4][5][6], Forman develops several methods and defines new experimental settings to evaluate the task, especially focussing on the cases where the training class distribution is different to the test class distribution.

One of the first conclusions from these works is that the naïve solution called *classify & count (CC)* does not work well. Consequently, several other methods are introduced, by properly adjusting the threshold and also scaling the result. Other methods are not based on *CC* but are still based on thresholds. However, there is another way to address the problem, which is to consider probability estimators instead of crisp classifiers. If a classifier estimates a class probability for each example, the *CC* method becomes the *probability estimation & average (P&A)* method. Surprisingly, this approach is considered a "non-solution" in Section 3.4 in [4] and an "ill-conceived method" in Section 2.4 in [6]. The reason for this is clearly shown with an example: "For a well-calibrated classifier, the output value $y = 70\%$ indicates that, of the training cases that score similarly, approximately 70% were positive. Supposing a large test set contains the same proportion of positives, as among those cases that score $y = 70\% \pm \epsilon$, roughly 70% would be positive. However, if we repeat the test with most of the negative cases removed at random, then the proportion of positives among the remaining test cases scoring in this bin ($y = 70\% \pm \epsilon$) would be much greater. Thus, the output of 70% would greatly underestimate the new $P(+|y = 70\%)$; and likewise for every bin. Again, the end effect is that this method would underestimate at high prevalence and overestimate at low prevalence, just like the *CC* method. As a result, the use of probability estimators has not been explored since it has been considered a "non-solution".

We agree with the rationale, but this does not necessarily imply that we should not give adjusted versions of *P&A* a chance. If adjusted versions of *CC* work, such as Forman's *AC* and *T*50, we think we could explore similar (or different) adjusting methods for *P&A*. In this paper, we present a simple scal-

ing method over *P&A*, called "Scaled Probability Average", which shows very good performance. A quantifier based on probability estimation is not based on a threshold, so the appraisal of this threshold is not so critical. In fact, our method depends on the quality of all the probabilities since it considers all the information that the classifier gives us about the dataset.

Summing up, the main contributions of this paper are that we introduce probability estimation to the scene of quantification, and we derive a simple scaling method that shows good performance.

The paper is organised as follows. Section 2 introduces basic notation and terminology for the quantification problem and sets the quality measures for quantification. Section 3 derives the methods based on *P&A*, presents the plain version *Probability Average* (which is actually a bad solution in general) and then introduces the *Scaled Probability Average* method. These two methods are evaluated in Section 4 with a range of test class imbalances and then compared to other quantification methods. Finally, Section 5 wraps up the paper with the conclusions and future work.

2 Notation and Previous Work

Given a dataset T, n denotes the number of examples, and c the number of classes. We will use i to index or refer to examples, so we will express $i = 1 \dots n$ or $i \in T$ indistinctly. $f(i,j)$ represents the actual probability of example i to be of class j. We assume that $f(i,j)$ always takes values in $\{0,1\}$ and is not strictly a probability but a single-label indicator function. With $n_j = \sum_{i=1}^{n} f(i,j)$, we denote the number of examples of class j. $\pi(j)$ denotes the prior probability of class j, i.e., $\pi(j) = n_j/n$. When referring to a particular dataset T, we will use the equivalent expression $\pi_T(j) = \frac{\sum_{i \in T} f(i,j)}{|T|}$. Given a classifier, $p(i,j)$ represents the estimated probability of example i to be of class j taking values in $[0,1]$. $\hat{\pi}(j)$ denotes the estimated probability of class j which is defined as $\hat{\pi}_T(j) = \frac{\sum_{i \in T} p(i,j)}{|T|}$ when referring to a dataset T. For the sake of readability, when $c = 2$, we will use the symbols $\oplus$ for the positive class and $\ominus$ for the negative class. Since the probabilities are complementary for two classes, we will focus on the positive class. $C_\theta(i,j)$ is 1 iff j is the predicted class obtained from $p(i,j)$ using a threshold θ. We can omit θ when it is embedded in the classifier or clear from the context. When $c = 2$, we will use the following measures $TP = \sum_{i=1}^{n} f(i,\oplus)C(i,\oplus)$, $TN = \sum_{i=1}^{n} f(i,\ominus)C(i,\ominus)$, $FP = \sum_{i=1}^{n} f(i,\ominus)C(i,\oplus)$, $FN = \sum_{i=1}^{n} f(i,\oplus)C(i,\ominus)$; we will also use the ratios, $tpr = TP/(TP + FN)$ and $fpr = FP/(FP + TN)$. We will use *pos* for the actual proportion of positives, i.e., $\pi(\oplus)$; and we will use *neg* for the actual proportion of negatives, i.e., $\pi(\ominus)$. Finally, the function $clip(X, a, b)$ truncates a real value X inside an interval $[a, b]$. We represent the elements of T of class $\oplus$ and $\ominus$ with $T_\oplus$ and $T_\ominus$, respectively.

2.1 Quantification

Forman [4][5][6] was the first to identify and name the quantification problem: "quantification is accurately estimating the number of positives in the test dataset as opposed to classifying individual cases accurately." Therefore, the main objective in a quantification task is to estimate the class distribution in the target population from the distribution observed in the training dataset. This problem is especially important in many application areas, such as medicine, risk assessment, diagnosis, etc., where the training dataset does not represent a random sample of the target population (because population changes over time or the classes are highly imbalanced, with the positive class as a minority).

As Forman pointed out, quantification is a machine learning task that is quite different from classification. In quantification, we are interested in the test set as a whole in order to determine its class distributions and not in the individual predictions for each example. In fact, although accurate classifications generally give accurate class counting, an inaccurate classifier can also be a good quantifier if false positives and false negatives cancel each other. In [6], Forman introduced several quantification methods that we arrange into the following three groups:

- Methods based on counting the positive predicted examples. The *classify & count* (CC) and the *adjusted count* (AC) methods belong to this group.
- Methods based on selecting a classifier threshold, but in this case, the threshold is determined from the relationship between tpr and fpr in order to provide better quantifier estimates. For instance, some methods choose one particular threshold, such as: the *X method*, which selects the threshold that satisfies $fpr = 1 - tpr$; the *Max method*, which selects the threshold that maximises the difference $tpr - fpr$; or those methods like $T50$ that select a particular rate between tpr and fpr. The *Median Sweep (MS) method*[1] is another method that tests all the thresholds in the test set, estimates the number of positives in each one, and returns a mean or median of these estimations.
- Methods based on a mixture of distributions. The *Mixture Model* (MM)[4] is included in this group. It consists of determining the distributions from the classifier scores on the training positive ($D_{\oplus}$) and negative examples ($D_{\ominus}$) and then modelling the observed distribution D on the test set as a mixture of $D_{\oplus}$ and $D_{\ominus}$.

The best results are obtained with AC in general; however, when the training sets are small and the number of positives is small, other methods such as $T50$ or MS can get better results (at the cost of performing worse in other more balanced situations). Of the above-mentioned methods proposed by Forman, the simplest one is the CC method. It consists of learning a classifier from the training dataset and counting the examples of the test set that the classifier predicts positive ($\sum_{i \in Test} C(i, \oplus)$). This method gives poor results since it underestimates/overestimates the proportion of positives, unless the classifier

[1] It is proposed when the tpr and fpr are estimated from cross-validation.

is perfect ($tpr = 1$ and $fpr = 0$). The AC method is an improvement of the CC method, which estimates the true proportion of positives $\widehat{pos}$ by applying the equation $\widehat{pos} = \frac{\widehat{pos}' - fpr}{tpr - fpr}$, where $\widehat{pos}'$ is the proportion of predicted positives $\frac{\sum_{i \in Test} C(i, \oplus)}{|Test|}$. Forman proposed estimating tpr and fpr by cross-validation on the training set. Since this scaling can give negative results or results above 1, the last step is to clip $\widehat{pos}$ to the range [0..1]. Finally, $T50$ is another method that selects the threshold where $tpr = 50\%$ and the rest works the same as AC. This method supposedly behaves better when the denominator in the formula of $\widehat{pos}$ is unstable with AC. We have implemented these three methods in order to fairly compare the performance of our proposals with them. The reason for this choice is that these methods are the ones that are most related to ours and their performance is good, or very good, with respect to other quantification methods such as X, Max, or MS. In other words, they are representative of the state-of-the-art in quantification. Motivated by the fact that in quantification only one result per dataset is produced, Forman proposed an experimental methodology to evaluate quantification that is different from the one that is normally used for classification. It consists in varying the class distribution between the training set and the test set. For the training set, Forman randomly selected the number of positive and negative examples from a range. For the test set, he varied the percentage of positive examples to also cover scenarios with imbalanced classes.

As we have mentioned in the introduction, a natural approach to solve the quantification problem that Forman disregarded is to use a probability estimator instead of a classifier. But, we will show that there is a different way to estimate the positive proportion by using probability estimations sharing the spirit of the CC method.

Moreover, our proposal is supposed to be more robust to variations in the probability estimation of few examples than other methods based on thresholds because we take into account all the information from the dataset. For example, consider a test set with probabilities and actual classes as follows: (0.90,+), (0.55,+), (0.53,−), (0.51,−) and (0.21,−). If we set the threshold at 0.6, the proportion of positives is 20%; however, if the threshold is 0.5, then the proportion of positives is 80%. Note that this is a good classifier, with perfect ordering, i.e., AUC=1. Therefore, methods that use thresholds are less robust in the sense that a change in the estimation of few examples could cause a small shift in the threshold but a large variation in the positive count estimation. However, regarding the probabilities of all the examples, the proportion of positives is 54%. In order to have a large change here, many probability estimations would have to change significantly.

2.2 Quantification Evaluation

We use two global evaluation measures from the classical error measures for continuous variables, the Mean Absolute Error (MAE) and the Mean Squared Error (MSE) for quantification. Forman only used the absolute error in his experiments, but we consider that the MSE measure is a better way to quantify

differences between estimations for real values. Let us formalise these measures in order to better establish the quantification problem and goal.

Consider that we have a method that estimates the proportion of elements for each class ($\hat{\pi}_T(j)$). By calculating the absolute difference of these two values, we have the global MAE for each class, $GMAE_j(T) = |\pi_T(j) - \hat{\pi}_T(j)|$, and for all the classes we have $GMAE(T) = \frac{1}{c} \cdot \sum_{j=1..c} GMAE_j(T)$. Similarly, we calculate $GMSE_j(T) \left(\pi_T(j) - \hat{\pi}_T(j)\right)^2$ and $GMSE(T) = \frac{1}{c} \cdot \sum_{j=1..c} GMSE_j(T)$. For binary problems, we have that $GMAE_{\oplus} = GMAE_{\ominus} = GMAE$ and also that $GMSE_{\oplus} = GMSE_{\ominus} = GMSE$. Therefore, for binary problems, we will only evaluate the error for the proportion of positives.

3 Quantifying by Scaled Averaged Probabilities

The idea of using an average of the probability estimations is supported by the issue that probabilities represent much richer information than just the decisions, which are simply derived information from the probability estimation using a threshold. After this rationale, the use of probabilities shapes a family of methods that we call *probability estimation & average*. The simplest method in this family is called *Probability Average* (PA). First, a probabilistic classifier is learned from the training data, such as a Probability Estimation Tree or a Naïve Bayes model. Then, the learned model is applied to the instances in the test set, obtaining a probability estimation for each one. Finally, the average of the estimated probabilities for each class is calculated. Although this definition is multiclass, for the rest of the paper we will concentrate on binary datasets. In this case, we only need to care about one class (the positive class), and the method is defined as follows: $\hat{\pi}^{PA}_{Test}(\oplus) = \frac{\sum_{i \in Test} p(i,\oplus)}{|Test|}$.

Logically, if the proportion of positive examples in the training set is different from the proportion of positive examples in the test set, the result will not be satisfactory in general. The solution comes precisely from the analysis of the extreme case when all the elements in the test set are of one class. In this case, we will get the average probability for the positive cases alone, which can only be 1 for a perfect classifier (which is not frequently the case). As in the AC method, the idea is to use a proper scaling.

Nevertheless, from the training set, it is possible to calculate the *actual proportion of positive examples* ($\pi_{Train}(\oplus)$), the *positive probability average* ($\hat{\pi}_{Train}(\oplus)$), the *positive probability average for the positives* ($\hat{\pi}_{Train_{\oplus}}(\oplus)$), and the *positive probability average for the negatives* ($\hat{\pi}_{Train_{\ominus}}(\oplus)$).

From the definitions, it is easy to check the following: $\hat{\pi}_{Train_{\oplus}}(\oplus) \cdot \pi_{Train}(\oplus) + \hat{\pi}_{Train_{\ominus}}(\oplus) \cdot (1 - \pi_{Train}(\oplus)) = \hat{\pi}_{Train}(\oplus)$.

From this equation, we derive $\pi_{Train}(\oplus) = \frac{\hat{\pi}_{Train}(\oplus) - \hat{\pi}_{Train_{\ominus}}(\oplus)}{\hat{\pi}_{Train_{\oplus}}(\oplus) - \hat{\pi}_{Train_{\ominus}}(\oplus)}$, which yields a probabilistic version of Forman's adjustment (see Fig.1). When all are positives, $\hat{\pi}_{Train_{\oplus}}(\oplus)$ sets the maximum, and we scale this to 1. When all are negatives, $\hat{\pi}_{Train_{\ominus}}(\oplus)$ sets the minimum, and we scale this to 0.

Thus, we propose a new quantification method, which we call *Scaled Probability Average* (SPA), applying this last formula (*scaling*) in the same way as Forman to the value obtained with the PA method ($\hat{\pi}^{PA}_{Test}(\oplus)$), i.e., $\hat{\pi}^{SPA}_{Test}(\oplus) = \frac{\hat{\pi}^{PA}_{Test}(\oplus) - \hat{\pi}_{Train_\ominus}(\oplus)}{\hat{\pi}_{Train_\oplus}(\oplus) - \hat{\pi}_{Train_\ominus}(\oplus)}$.

$$\hat{\pi}^{SCC}_{Test}(\oplus) = \frac{\frac{\sum_{i \in Test} C(i,\oplus)}{|Test|} - \hat{\pi}_{Train_\ominus}(\oplus)}{\hat{\pi}_{Train_\oplus}(\oplus) - \hat{\pi}_{Train_\ominus}(\oplus)}$$

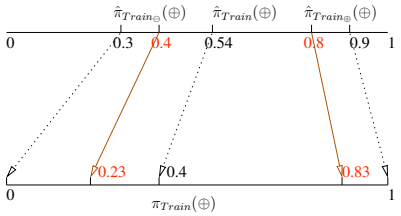


Fig. 1. Scaling used in the SPA method. The limits in the training set are placed at 0.3 and 0.9. The estimated value for the training set is 0.54 whereas the actual proportion in the training set is 0.4. The scaling would move a case at 0.4 to 0.23 and a case at 0.8 to 0.83.

In the same way as in the SPA method, the proportion of positive examples estimated by the PA method is scaled. This scaling can also be applied to the proportion of positive examples estimated by the CC method. Therefore, we propose the *Scaled Classify & Count* (SCC) method.

4 Experiments

In this section, we present an experimental evaluation of several quantification methods: those based on classification (i.e., on a threshold), namely CC, AC, T50 (explained in Section 2); our two methods based on probability average, namely PA and SPA; and a hybrid method (SCC), which is explained in Section 3. We have not used an internal cross-validation process with the training set to better estimate the thresholds, tpr and fpr, as Forman does. Instead, we have used an additional validation set to estimate the thresholds, tpr and fpr. In other words, the methods are the same, but instead of using a 50 fold cross-validation process with the training set, we have used two different sets (training and validation) and repeated the process 100 times.

The experimental setting is based on the common case where we train a classifier or probability estimator on a training dataset and we want to quantify the number of examples of one class for a different test dataset. We are especially interested in cases where class distributions vary between training and test. In order to get this variation (and being able to consider cases where the majority class examples are increased or reduced), for each problem, we divided the whole dataset into 37.5% for training, 37.5% for validation, and 25% for test. With this original test set, we constructed six different variations test sets: the whole

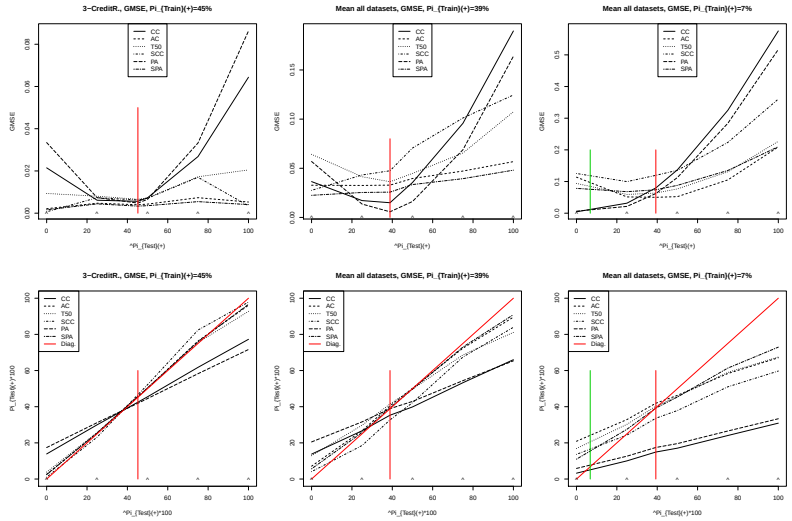


Fig. 2. Upper-Left: GMSE results with variable test positive ratios for the dataset CreditR. Upper-Centre: GMSE results with variable test positive ratios for the average of all the datasets. Upper-Right: GMSE results with variable test positive ratios for the average of all the datasets, with the proportion of positive examples in the training and validation sets reduced by 90%. Lower-(Left, Centre and Right): The value of each quantification method for each test setting. The diagonal shows the perfect value.

Table 1. GMSE measure for each dataset and the average result of all the datasets for each quantification method, and MSE, AUC, CalBin and Accuracy measures for each dataset.

#	Datasets	Size	$\pi_{Tr}(\oplus)$	CC	AC	T50	SCC	PA	SPA	MSE	AUC	CalBin	Acc.
1	W.B. Cancer	699	0.34	0.0022	0.0008	0.0667	0.0008	0.0032	0.0006	0.04	0.82	0.04	0.95
2	Chess	3196	0.48	0.0033	0.0002	0.0155	0.0024	0.0116	0.0001	0.06	0.82	0.09	0.93
3	Credit R.	690	0.45	0.0219	0.0046	0.0115	0.0070	0.0286	0.0037	0.15	0.76	0.16	0.82
4	German C.	1000	0.31	0.0921	0.0265	0.0277	0.0598	0.0878	**0.0176**	0.24	0.65	0.26	0.65
5	Pima D.	768	0.35	0.0572	0.0160	0.0247	0.0332	0.0642	0.0099	0.21	0.69	0.23	0.70
6	House V.	435	0.38	0.0044	0.0018	0.0522	0.0018	0.0048	0.0014	0.07	0.81	0.07	0.92
7	Monks1	556	0.49	0.0449	0.0072	0.0186	0.1025	0.0487	0.0057	0.16	0.72	0.23	0.79
8	Mushroom	8124	0.48	0.0268	0.0004	0.0380	0.0022	0.0211	0.0003	0.12	0.80	0.13	0.84
9	Spam	4601	0.39	0.0118	0.0003	0.0284	0.0006	0.0156	0.0003	0.10	0.80	0.10	0.88
10	Tic-tac	958	0.34	0.0626	0.0107	0.0127	0.0525	0.0671	**0.0062**	0.19	0.71	0.23	0.72
11	Breast C.	286	0.29	0.1280	0.1118	0.1171	0.1638	0.1057	**0.0829**	0.28	0.60	0.31	0.62
12	Haberman B.	306	0.27	0.2045	0.1478	0.1265	0.2555	0.1316	**0.1049**	0.28	0.59	0.31	0.60
13	Heart D.	303	0.45	0.0236	0.0108	0.0235	0.0116	0.0291	0.0083	0.15	0.75	0.17	0.80
14	Heart S.	270	0.43	0.0251	0.0141	0.0352	0.0152	0.0320	0.0111	0.15	0.75	0.18	0.80
15	Ionosphere	351	0.35	0.0429	0.0075	0.0446	0.0068	0.0487	0.0053	0.17	0.75	0.19	0.81
16	Monks2	601	0.34	0.2386	0.1826	0.2146	0.3087	**0.1168**	0.1732	0.27	0.55	0.31	0.52
17	Monks3	554	0.48	0.0010	0.0004	0.0151	0.0069	0.0151	0.0006	0.05	0.82	0.13	0.97
18	Hepatitis	155	0.19	0.1536	0.1241	0.1329	0.1454	0.1226	**0.0885**	0.27	0.66	0.30	0.66
19	Sonar	208	0.46	0.0525	0.0460	0.0622	0.0521	0.0572	0.0400	0.26	0.67	0.27	0.69
20	Spect	80	0.49	0.1009	0.0921	0.1308	0.1530	0.0722	0.0879	0.24	0.68	0.27	0.70
	AVG.		0.39	0.0649	0.0403	0.0599	0.0691	0.0542	0.0324	0.17	0.72	0.2	0.77

test set with the original proportion of classes, and five test sets changing the proportion of classes (100% of examples of positive class and 0% of negative class, 75% and 25%, 50% and 50%, 25% and 75%, and 0% and 100%). The proportions were obtained by random undersampling using a uniform distribution.

In order to have a broad range of classifiers and probability estimators, we used four different methods from WEKA [7]: NaïveBayes, J48 (a C4.5 implementation), IBk ($k = 10$) (a k-NN implementation), and Logistic (a logistic regression implementation). We used the WEKA default parameters in the methods (except the parameter k in the IBk method that is set to 10). A total of 100 repetitions were performed for each dataset (25 for each classifier). We selected 20 binary datasets (Table 1) from the UCI repository [1]. We adopted the same criterion as Forman, and the positive class is the minority class.

In Table 1, we show the results with respect to the GMSE measure[2] for each method. These values are the average of the 100 repetitions for each dataset. We also include MSE, AUC, CalBin and Accuracy measures (see, e.g. [3] for a definition of these measures). For each dataset, as suggested in [2], we calculated a Friedman test and showed that the six methods do not have identical effects, so we calculated the Nemenyi post-hoc test to compare all the methods with each other (with a probability of 99.5%). The values in bold show that this method outperforms the others and that the difference is statistically significant. Also, if several values are underlined, this indicates that these methods outperform the

[2] GMAE results portray a similar picture.

rest with a statistically significant difference, but the difference between the two is not significant.

The results are categorical as far as the use of probability estimators. Not only do they work, but they outperform all the other methods. Of course, as we expected, a proper scaling is the key to success. In the same way that AC dramatically improves CC, SPA also improves PA. However, by looking at the table in more detail we can get more insight. For instance, while CC and PA are relatively close, SPA is able to improve PA more than AC is able to improve CC (in relative terms). The reason may be found in the way that SPA is not based on thresholds (or measures such as tpr or fpr which depend on them), but on averages. Thus, the results show that SPA is more robust. Regarding the other methods, SCC is a hybrid method that uses classification to estimate the base proportion and uses the probability average to scale it. The result shows no clear improvement over CC or PA. This supports the thesis that, for SPA, it is the conjunction of PA with the scaling what makes the results good. The behaviour of $T50$ is expected, since it highly depends on the threshold and this is a method oriented to cases where the original training set is highly imbalanced.

In order to analyse the effect of imbalance (in the training set), we repeated the experiments but reduced the number of examples of the positive class by 90% in the training and validation sets. In this situation, the SPA method still obtains good results, but, in most cases, the differences between the SPA method and the AC and $T50$ methods are not statistically significant. We want to remark that AC and $T50$ methods are designed for situations where the proportion of classes are imbalanced. Even though, this second experimental setting is more favourable for the AC and $T50$ methods, SPA method still obtains good results. These results are summarised in Fig.2 (right).

Finally, in order to analyse the effect with respect to test imbalance, for each dataset, we analysed the results for the six different test set imbalances. The conclusion was that the behaviour is similar in all the datasets. The best results (as expected) are obtained with class proportions that are close to the original class proportion or close to 0.5, and much worse as we get closer to 0 or 1 proportions. In Fig.2 (left), we show a typical case of a dataset where the errors are much lower when the test proportion is close to the original training proportion. In Fig.2 (centre), we show the average results for the 20 datasets without reducing the proportion of examples of the positive class in the training set. In Fig.2 (right), we show the results of reducing the proportion of examples of the positive class by 90% in the training and validation sets. Fig.2 (centre) shows that when the proportion of examples of the positive class in the test set is more or less between 20% and 60%, the AC method is the one that obtains the best results. In the rest of proportions, the SPA is the best method because its behaviour is always quite similar. With this in mind, it is logical to think in terms of a new "mixed" method, which uses the method that obtain the best results in each interval. We have not implemented this method, but we propose its implementation and experimental evaluation as an interesting topic for future work.

5 Conclusions and Future Work

Quantification is an old and frequent problem that has only very recently received proper attention as a separate machine learning task. In our opinion, one of the most natural ways to address the problem is to average the probability estimations for each class because we use all the information provided by the probability estimator and because we do not rely so much on a good threshold choice. However, in the same way that CC does not work, this easy approach does not work. Since CC can be adjusted, we have seen that a simple probability average method can also be adjusted. We have derived a generalisation of Forman's scaling for probabilities, and we have derived a new method from it. The results are highly positive and show that the use of probability estimators for quantification is good to pursue, which can lead to new methods in the future.

As further future work, there are obviously several areas (cost quantification [6]) and experimental settings (very imbalanced training datasets) that are borrowed from Forman's seminal works for which a more exhaustive evaluation of our methods and the derivation of specialised ones should be addressed. One of these extensions is quantification for more than two classes. Our PA method is originally multiclass, but the scaling used in SPA is not. We think that proper methods (different to indirect 1vs1 or 1vsall) could be derived using probability averaging.

Finally, quantification for regression is also a machine learning task that should be investigated (as we stated in the introduction with the credit assessment problem, where an estimation of the total amount for a customer portfolio and not the number of granted credits would be the goal). In fact, quantification for regression might be closer to quantification through probability averaging than through classification.

References

1. C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998.
2. J. Demsar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, January 2006.
3. C. Ferri, J. Hernández-Orallo, and R. Modroiu. An experimental comparison of performance measures for classification. *Pattern Recogn. Lett.*, 30(1):27–38, 2009.
4. G. Forman. Counting positives accurately despite inaccurate classification. In *ECML*, pages 564–575, 2005.
5. G. Forman. Quantifying trends accurately despite classifier error and class imbalance. In *KDD*, pages 157–166, 2006.
6. G. Forman. Quantifying counts and costs via classification. *Data Min. Knowl. Discov.*, 17(2):164–206, 2008.
7. I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Elsevier, 2005.

7.6 Local and Global Calibration. Quantification using Calibrated Probabilities

6. A. Bella, C. Ferri, J. Hernández-Orallo, and M.J. Ramírez-Quintana. Local and Global Calibration. Quantification using Calibrated Probabilities. Technical report, DSIC-ELP. Universitat Politècnica de València, 2010.

Local and Global Calibration. Quantification using Calibrated Probabilities

A. Bella, C. Ferri, J. Hernández-Orallo, and M.J. Ramírez-Quintana

DSIC-ELP, Universitat Politècnica de València, Camí de Vera s/n, 46022 València, Spain

Abstract. In this document we present an study of calibration methods for quantification that complements the paper "Quantification via Probability Estimators" [3].

1 Quantification and Calibration

Given the results of the Section IV in [3], a question that arises is the impact of having good probability estimators for quantification, since the method proposed in Section III in [3] is based on a probability average. [7][8][9] discusses that some bad classifiers can be good quantifiers, but also (and frequently) that some good classifiers can be very bad quantifiers. The key issue to understand this when we use *classify & count* and related methods is how the threshold is chosen. The key issue to understand this when we use *average probability* methods is calibration [4][5][11][14][2]. A random, but well-calibrated classifier can give very good results with the SPA method, when training and test class distributions match. On the contrary, a very good classifier, with high separability (e.g. a high AUC) can have very bad calibration and give very bad quantification results with the SPA method. But there are some other, more surprising cases. Consider the worst classifier possible, which in the binary case outputs $p(i, \oplus) = 1$ for every example i of class $\ominus$ and outputs $p(i, \oplus) = 0$ for every example i of class $\oplus$. If the dataset is balanced, quantification will be perfect. But calibration in this case is extremely poor. As a result, calibration plays a very important role for quantification using probabilities, but the phenomenon is more complex than it seems at first sight, even in the easy cases where training and test class distributions are equal.

1.1 Local and Global Calibration

The first intuition from the previous discussion is that we do not need to have good calibration for every single prediction to have an overall good quantification result. This suggests the distinction of what we call local (traditional) calibration and global calibration. Let us see local calibration first.

A probability estimator is said to be perfectly calibrated [5] if as the number of predictions goes to infinity, the predicted probability $p(i, j)$ goes to the empirical probability. In other words, the expectation of example i to be of class j

is indeed $p(i,j)$. Note that this does not mean that a classifier derived from the estimator is always correct. In fact, an estimator always outputting $p(i,\oplus) = 0.5$ for every i for a balanced binary problem (with classes $\oplus$ and $\ominus$) is completely useless but perfectly calibrated.

Since the previous definition is based on an expectation or an empirical probability, this logically depends on a distribution. This means that a classifier can be perfectly calibrated with respect to a distribution but not with respect to a different distribution. For instance, the previous perfectly calibrated classifier always outputting $p(i,\oplus) = 0.5$ becomes poorly calibrated if the problem distribution changes to a situation where classes are now highly imbalanced.

Instead of distributions, we will talk about datasets, and we will define perfect calibration in a more specific way using the notion of bin:

Definition 1. *Given a dataset D, a set $B \subset D$ is an ϵ-bin if $\forall j \in \mathcal{C}$ and $\forall i,k \in B, |p(i,j) - p(k,j)| \leq \epsilon$, where $\mathcal{C}$ are the classes of the dataset.*

A 0-bin is any set such that all the elements in the bin have the same estimated probability. An ϵ-bin B is maximal if there is no other ϵ-bin B' such that $B \subset B'$. The value ϵ is called the calibration width and it is naturally assumed to be less than or equal to 1. From here we define perfect calibration as follows:

Definition 2. *A probability estimator is ϵ-calibrated for a dataset D with a tolerance τ iff for every maximal ϵ-bin B we have that:*
$\forall j \left| \frac{\sum_{i \in B} p(i,j)}{|B|} - \frac{\sum_{i \in B} f(i,j)}{|B|} \right| \leq \tau$

We need to explain the meaning of both τ and ϵ. The meaning of τ is a degree of accordance or tolerance between the expected frequency in a bin and the actual probability. This is, then, a calibration measure. When $\tau = 0$ we talk about *perfect* calibration. The meaning of τ is hence similar to existing measures of degrees of calibration, such as 'calbin' and the calibration component in the Brier's score [12][10], which are also based on the notion of binning (as well as some calibration methods (see e.g. [14][6])).

The meaning of ϵ is different and regulates the granularity/strictness of the bins. In fact, we can talk about local (traditional) calibration when $\epsilon = 0$ and we can talk about global calibration when $\epsilon = 1$. Or, more precisely, we can talk about different degrees of local/global calibration, with 0-calibration being the extreme point for local calibration and 1-calibration being the other extreme point for global calibration.

Let us see some examples. Consider for instance a binary probability estimator with values:

Example 1.
$p(1,\oplus) = 1$ with $f(1,\oplus) = 1$ $\quad$ $p(5,\oplus) = 0.75$ with $f(5,\oplus) = 1$
$p(2,\oplus) = 0.75$ with $f(2,\oplus) = 1$ $\quad$ $p(6,\oplus) = \frac{1}{3}$ with $f(6,\oplus) = 1$
$p(3,\oplus) = 0.75$ with $f(3,\oplus) = 0$ $\quad$ $p(7,\oplus) = \frac{1}{3}$ with $f(7,\oplus) = 0$
$p(4,\oplus) = 0.75$ with $f(4,\oplus) = 1$ $\quad$ $p(8,\oplus) = \frac{1}{3}$ with $f(8,\oplus) = 0$

The previous estimator is perfectly 0-calibrated, since the maximal 0-bins are $B_1 = \{1\}$, $B_2 = \{2,3,4,5\}$, $B_3 = \{6,7,8\}$. It can be seen that the average frequency for class $\oplus$ in B_1 is 1, the average frequency for class $\oplus$ in B_2 is 0.75 and the average frequency for class $\oplus$ in B_3 is $\frac{1}{3}$.

And now consider the following example

Example 2.
$p(1,\oplus) = 5/6$ with $f(1,\oplus) = 1$ $p(6,\oplus) = 5/6$ with $f(6,\oplus) = 1$
$p(2,\oplus) = 5/6$ with $f(2,\oplus) = 1$ $p(7,\oplus) = 0.7$ with $f(7,\oplus) = 1$
$p(3,\oplus) = 5/6$ with $f(3,\oplus) = 0$ $p(8,\oplus) = 0.7$ with $f(8,\oplus) = 0$
$p(4,\oplus) = 5/6$ with $f(4,\oplus) = 1$ $p(9,\oplus) = 0.6$ with $f(9,\oplus) = 1$
$p(5,\oplus) = 5/6$ with $f(5,\oplus) = 1$

This example is 0-calibrated (i.e. locally calibrated) with any tolerance $\tau \geq 0.4$) (with bins $B_1 = \{1,2,3,4,5,6\}$, $B_2 = \{7,8\}$ and $B_3 = \{9\}$). It is 0.1-calibrated with any tolerance $\tau \geq 0$ (with bins $B_1'' = \{1,2,3,4,5,6\}$ and $B_2'' = \{7,8,9\}$). It is 0.2-calibrated with any tolerance $\tau \geq 0.05$ (with bins $B_1'' = \{1,2,3,4,5,6,7,8\}$ and $B_2'' = \{7,8,9\}$). It is 1-calibrated (i.e. globally calibrated) with any tolerance $\tau \geq 0$ (with bin $B_1''' = \{1,2,3,4,5,6,7,8,9\}$).

Figure 1 shows the evolution of the minimum tolerance (τ) for different degrees of locality/globability (ϵ). As we can see the curve is typically decreasing, but not necessarily monotonic. The graph shows the behaviour of a model for classification (leftmost part of the curve, $\epsilon = 0$) and for quantification (rightmost part of the curve).

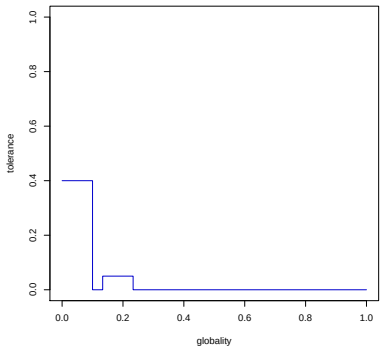


Fig. 1. A local/global calibration plot for example 2. The graph shows the minimum tolerance (*tau*) for each degree of globability (ϵ).

Although we can see from the previous example that the curve is not monotonically decreasing, we are interested in discovering any relation between local and global calibration. The following two propositions show this:

Proposition 1. *If a probability estimator is* 0*-calibrated with tolerance* τ *for dataset* D *then it is* γ*-calibrated with tolerance* τ *for every* $0 \leq \gamma \leq 1$.

Proof. Let us call $\mathcal{B}_0$ the set of maximal 0-bins and $\mathcal{B}_\gamma$ the set of maximal γ-bins.

If B is a bin in $\mathcal{B}_\gamma$ then it is clear that it can be decomposed into a new disjunct set $\mathcal{B}$ of bins $B_1, B_2, ..., B_n$ in $\mathcal{B}_0$. Logically $\bigcup_{B_k \in \mathcal{B}} = B$. For every B in $\mathcal{B}_\gamma$ we need to prove that: $\forall j \left| \frac{\sum_{i \in B} p(i,j)}{|B|} - \frac{\sum_{i \in B} f(i,j)}{|B|} \right| \leq \tau$.

The left term can be decomposed into:
$\left| \frac{\sum_{B_k \in \mathcal{B}} |B_k| \frac{\sum_{i \in B_k} p(i,j)}{|B_k|}}{\sum_{B_k \in \mathcal{B}} |B_k|} - \frac{\sum_{B_k \in \mathcal{B}} |B_k| \frac{\sum_{i \in B_k} f(i,j)}{|B_k|}}{\sum_{B_k \in \mathcal{B}} |B_k|} \right|$
and reduced into: $\left| \frac{\sum_{B_k \in \mathcal{B}} |B_k| \left(\frac{\sum_{i \in B_k} p(i,j)}{|B_k|} - \frac{\sum_{i \in B_k} f(i,j)}{|B_k|} \right)}{\sum_{B_k \in \mathcal{B}} |B_k|} \right|$

And, since that for any sum of real numbers we have that $\sum_i x_i \leq \sum_i |x_i|$, we have that this is lower than or equal to:

$$\left| \frac{\sum_{B_k \in \mathcal{B}} |B_k| \left| \frac{\sum_{i \in B_k} p(i,j)}{|B_k|} - \frac{\sum_{i \in B_k} f(i,j)}{|B_k|} \right|}{\sum_{B_k \in \mathcal{B}} |B_k|} \right| \tag{1}$$

But we know from the definition of ϵ-calibration that:
$\left| \frac{\sum_{i \in B_k} p(i,j)}{|B_k|} - \frac{\sum_{i \in B_k} f(i,j)}{|B_k|} \right| \leq \tau$ and we also have that $\bigcup_{B_k \in \mathcal{B}} = B$.

So we have that 1 is lower than or equal to:
$\left| \frac{\sum_{B_k \in \mathcal{B}} |B_k| \tau}{\sum_{B_k \in \mathcal{B}} |B_k|} \right| = \tau$

Proposition 2. *If a probability estimator is* ϵ*-calibrated with tolerance* τ *for dataset* D *with* $0 \leq \epsilon \leq 1$ *then it is* 1*-calibrated with tolerance* τ.

Proof. From the definition of ϵ-calibration we have that for every maximal ϵ-bin B_k: $\forall j \left| \frac{\sum_{i \in B_k} p(i,j)}{|B_k|} - \frac{\sum_{i \in B_k} f(i,j)}{|B_k|} \right| \leq \tau$. We can construct a set of bins $\mathcal{B}_\epsilon$ with all the maximal ϵ-bins. Consequently, the definition of 1-calibration can be decomposed as follows:
$\forall j \left| \frac{\sum_{B_k \in \mathcal{B}_\epsilon} |B_k| \frac{\sum_{i \in B_k} p(i,j)}{|B_k|}}{\sum_{B_k \in \mathcal{B}_\epsilon} |B_k|} - \frac{\sum_{B_k \in \mathcal{B}_\epsilon} |B_k| \frac{\sum_{i \in B_k} f(i,j)}{|B_k|}}{\sum_{B_k \in \mathcal{B}_\epsilon} |B_k|} \right| \leq \tau$
Operating as in the proof for proposition 1 we get the desired result.

In Example 1, which is perfectly 0-calibrated, it is easy to see that it is also perfectly 0.5-calibrated, since the 0.5-bins in this case are $B'_1 = \{1, 2, 3, 4, 5\}$, $B'_2 = \{2, 3, 4, 5, 6, 7, 8\}$, with average predictions and average frequencies for class $\oplus$ matching with values 0.8 and $\frac{4}{7}$ respectively. But consider again example

2. This example is perfectly 0.1-calibrated (with bins $B_1 = \{1, 2, 3, 4, 5, 6\}$ and $B_2 = \{7, 8, 9\}$), but is not perfectly 0.2-calibrated (with bins $B'_1 = \{1, 2, 3, 4, 5, 6, 7, 8\}$ and $B'_2 = \{7, 8, 9\}$). Consequently, the previous propositions cannot be extended to the general case stating that if a probability estimator is ϵ-calibrated for dataset D then it is γ-calibrated for every $\gamma \geq \epsilon$, as we also saw in Figure 1. We only have that this is true when $\epsilon = 0$ or $\gamma = 1$.

Additionally, any reverse implication (from global to local calibration) is not true in general as the following example shows, even with a perfect global calibration we can have a very bad local calibration:

Example 3.
$p(1, \oplus) = 1$ with $f(1, \oplus) = 0$, $\quad p(2, \oplus) = 0$ with $f(2, \oplus) = 1$

So, local calibration ($\epsilon = 0$) gives us an upper bound of the tolerance for any other calibration degree ($\epsilon > 0$), and global calibration ($\epsilon = 1$) is a lower bound of the tolerance for any other calibration degree ($\epsilon < 1$). This is relevant, since it shows that good local calibration should be beneficial for global calibration, and hence, for quantification. But it also suggests that global calibration is an easier problem than local calibration, so perhaps focussing on local calibration is not the way to go.

The previous results help us understand the role of calibration and the quantification results for a dataset. But, what happens when class distribution changes? It is clear that Example 1 will behave better than Example 3. In order to better understand why, we can adapt the previous definition to each class subset of the dataset. In particular, let us call $D_\oplus$ the subset of D which includes all the examples in class $\oplus$, while $D_\ominus$ is the subset of D which includes all the examples in class $\ominus$. We can see that Example 1 will behave better than Example 3, since for the first example in $D_\oplus$ we construct the bins $B_1 = \{1\}$, $B_2 = \{2, 4, 5\}$, $B_3 = \{6\}$ with average probabilities 1, 0.75 and 1/3 respectively. For $D_\ominus$ we construct the bins $B_4 = \{3\}$, $B_5 = \{7, 8\}$ with average probabilities 0.75 and 1/3 respectively. For Example 3, though, we have one single bin for $D_\oplus$, $B_1 = \{1\}$ with average probability 0 and a single bin for $D_\ominus$, $B_2 = \{2\}$ with average probability 1. For any reasonable loss function 1 seems better calibrated than Example 3. But note that when we use datasets with elements of only one class, then we have:

Proposition 3. *If all the elements of a dataset D are from the same class, for every ϵ and γ such that $0 \leq \epsilon \leq 1$ and $0 \leq \gamma \leq 1$ then a classifier is perfectly ϵ-calibrated for dataset D iff it is perfectly γ-calibrated.*

Proof. If all the elements are from the same class c, we have that $f(i, j) = 1$ iff $j = c$, and 0 otherwise. This means that if a classifier is perfectly ϵ-calibrated for this dataset, all the elements in all the maximal ϵ-bins must have $p(i, j) = 1$ iff $j = c$, and 0 otherwise. Consequently, the classifier is perfectly γ-calibrated for any γ.

This means that when we check with a dataset with examples of only one class, we get that global and local calibration are the same, but only if this calibration is perfect, which is not generally the case. This can also be seen in the way that the higher the class imbalance the closer global and local calibration are, especially when calibration is high. And this means that for extreme imbalances good local calibration must be important. However, it is also clear that a scaling is necessary since:

Proposition 4. *Given a dataset D with elements of several classes, if we remove the elements of all classes but j into a new dataset D', then any perfectly 0-calibrated classifier for D which does not estimate $p(i,j) = 1$ for every instance for which $f(i,j) = 1$, is not a perfectly 1-calibrated classifier for D'.*

Proof. Trivial, since by definition, the classifier does not estimate $p(i,j) = 1$ for every instance for which $f(i,j) = 1$, which is the case for every instance in D'. Consequently, the classifier cannot be perfectly 1-calibrated.

A different thing is when we remove just a part of the elements of all classes but j. In this case, a similar result can only be obtained if the subsample is made with a uniform distribution. And this, as we mentioned in the previous section is an assumption made by previous works and also by our scaling, but it is not necessarily the case in every real application. Again, when we have a distribution drift of a binary problem, when one class becomes rarer than it was, then the dubious cases of the minority class have higher probability of not being subsampled than the clear cases of the minority class. This means that in many applications a class frequency change is made with a distribution which is not uniform and perhaps we should use a softer scaling or no scaling at all.

Summing up, in this section we have formalised the notions of local and global calibration over a dataset. Global calibration is key to the success of quantification methods using probabilities. However, the relation of local to global calibration suggests that when when class distribution is the same (and especially when the dataset is imbalanced), having good local calibration implies having good global calibration (at least it sets some bounds on the error), but when class distribution changes dramatically, it is not so clear that a very good local calibration ensures good global calibration. To answer the question on whether better local calibration (alone) can make quantification improve, in the following subsection we experimentally analyse several classical (local) calibration methods and their effect over our method.

1.2 Analysis of Calibration Methods for Quantification

The objective of calibration methods (as a postprocessing) is to improve the original estimated probabilities of a given classifier. These methods can also be applied to transform scores into accurate probability estimates [11] [13]. In this section we apply some well-known calibration methods in order to study their effect on quantification. Concretely we employ three calibration techniques: Binning [14], Platt [11] and PAV [1]. For a survey of calibration we refer to [2].

The obtained results are included in Table 1. We use the methodology explained in Section IV in [3]. In this table we show the results for the 20 datasets, and the four last rows show the average of all the datasets for each calibration method and for each quantification method. For each quantification technique and dataset (each square in the table) we conducted the same statistical test used in the experimental section (so, no cross comparisons are made). The results in Table I in [3] and 1 (Before) are similar. They are not exactly the same because the seeds in the experiments to split the sets are not the same and, although we consider that the number of repetitions is enough (100 times for each dataset), with more repetitions it is expected that the difference would be lower. Another point is that there are few results in bold, so, at first sight, it seems like calibration does not improve quantification methods. However, if we observe the average of the calibration methods, the results in terms of GMSE of the quantification methods based on probabilities (PA and SPA) are better for the PAV and *Binning* calibration methods than for the case without calibration (Before), and this does not happen for other quantification methods. Therefore, not only the SPA obtains good results but also it can be improved using calibration or other techniques. These results are very preliminary and it is clear that the effect of calibration in quantification must be studied in detail, but they are a first encouraging step in a new open area. For example, designing global calibration methods that improve the quantification results more than the traditional local calibration methods.

References

1. M. Ayer, H.D. Brunk, G.M. Ewing, W.T. Reid, and E. Silverman. An empirical distribution function for sampling with incomplete information. *Annals of Mathematical Statistics*, 5:641–647, 1955.
2. A. Bella, C. Ferri, J. Hernandez-Orallo, and M.J. Ramirez-Quintana. Calibration of machine learning models. In *Handbook of Research on Machine Learning Applications*. IGI Global, 2009.
3. A. Bella, C. Ferri, J. Hernández-Orallo, and M.J. Ramírez-Quintana. Quantification via Probability Estimators. *IEEE International Conference on Data Mining*, 0:737–742, 2010.
4. G. W. Brier. Verification of forecasts expressed in terms of probabilities. *Monthly Weather Review*, 78:1–3, 1950.
5. M.H. DeGroot and S.E. Fienberg. The comparison and evaluation of forecasters. *The statistician*, pages 12–22, 1983.
6. C. Ferri, J. Hernández-Orallo, and R. Modroiu. An experimental comparison of performance measures for classification. *Pattern Recogn. Lett.*, 30(1):27–38, 2009.
7. G. Forman. Counting positives accurately despite inaccurate classification. In *European Conf. on Machine Learning*, pages 564–575, 2005.
8. G. Forman. Quantifying trends accurately despite classifier error and class imbalance. In *KDD*, pages 157–166, 2006.
9. G. Forman. Quantifying counts and costs via classification. *Data Min. Knowl. Discov.*, 17(2):164–206, 2008.
10. A.H. Murphy. A new vector partition of the probability score. *Journal of Applied Meteorology*, 12(4):595–600, 1973.

11. J. C. Platt. In *Advances in Large Margin Classifiers*, pages 61–74. 1999.
12. F. Sanders. On subjective probability forecasting. *Journal of Applied Meteorology*, 2(2):191–201, 1963.
13. B. Zadrozny and C. Elkan. Transforming classifier scores into accurate multiclass probability estimates. In *Int. Conf. on Knowledge Discovery and Data Mining*.
14. B. Zadrozny and C. Elkan. Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers. In *ICML*, pages 609–616, 2001.

Table 1. Calibration results.

	CC	AC	T50	SCC	PA	SPA
1 Before	0.0022	0.0009	0.0660	0.0011	0.0032	0.0008
1 PAV	0.0019	0.0009	0.0986	0.0012	0.0026	0.0008
1 Platt	0.0021	0.0010	0.0660	0.0027	0.0081	0.0009
1 Binn.	0.0019	0.0010	0.0861	0.0011	0.0030	0.0009
2 Before	0.0031	0.0002	0.0105	0.0024	0.0114	0.0002
2 PAV	0.0028	0.0002	0.0530	0.0008	**0.0050**	0.0001
2 Platt	0.0027	0.0002	0.0115	0.0037	0.0116	0.0001
2 Binn.	0.0027	0.0002	0.0460	0.0008	0.0053	0.0001
3 Before	0.0221	0.0043	0.0114	**0.0059**	0.0288	0.0036
3 PAV	0.0151	0.0030	0.0137	0.0082	0.0254	0.0028
3 Platt	0.0208	0.0042	0.0114	0.0208	0.0365	0.0036
3 Binn.	0.0158	0.0050	0.0325	0.0086	0.0268	0.0028
4 Before	**0.0889**	0.0216	0.0219	**0.0514**	**0.0859**	0.0163
4 PAV	0.1369	0.0397	0.0282	0.1727	0.0973	0.0158
4 Platt	0.1461	0.0359	0.0219	0.1934	0.1015	0.0170
4 Binn.	0.1432	0.0503	0.0753	0.1926	0.0983	0.0163
5 Before	0.0594	0.0123	0.0232	**0.0287**	0.0653	0.0090
5 PAV	0.0632	0.0145	0.0277	0.0604	0.0676	0.0091
5 Platt	0.0658	0.0145	0.0232	0.0655	0.0728	0.0092
5 Binn.	0.0673	0.0227	0.0562	0.0705	0.0700	0.0102
6 Before	0.0043	0.0017	0.0517	**0.0019**	0.0048	0.0013
6 PAV	0.0045	0.0018	0.0867	0.0031	0.0056	0.0013
6 Platt	0.0039	0.0017	0.0517	0.0072	0.0143	0.0014
6 Binn.	0.0044	0.0019	0.0810	0.0032	0.0061	0.0013
7 Before	0.0444	0.0058	0.0199	0.1015	0.0483	0.0045
7 PAV	0.0421	0.0049	0.0795	0.0833	0.0315	0.0038
7 Platt	**0.0401**	0.0062	0.0199	0.0810	0.0435	0.0045
7 Binn.	0.0448	0.0092	0.0670	0.0767	0.0314	0.0041
8 Before	0.0270	0.0003	0.0348	0.0025	0.0210	0.0002
8 PAV	0.0070	0.0002	0.0649	0.0027	**0.0091**	**0.0001**
8 Platt	0.0137	0.0002	0.0359	0.0195	0.0230	0.0002
8 Binn.	0.0097	0.0003	0.0664	0.0048	0.0112	0.0002
9 Before	0.0121	0.0002	0.0275	**0.0006**	0.0160	0.0002
9 PAV	0.0058	0.0002	**0.0013**	0.0012	**0.0107**	0.0002
9 Platt	0.0121	0.0002	0.0275	0.0213	0.0239	0.0002
9 Binn.	0.0070	0.0003	0.0124	0.0015	0.0114	0.0002
10 Before	0.0627	0.0116	0.0128	0.0511	0.0659	0.0072
10 PAV	0.0691	0.0116	0.0142	0.0807	**0.0567**	0.0073
10 Platt	**0.0574**	0.0122	0.0128	0.0633	0.0635	0.0074
10 Binn.	0.0711	0.0201	0.0506	0.0849	0.0619	0.0081
11 Before	0.1308	0.1145	0.1158	**0.1666**	0.1076	0.0830
11 PAV	0.1479	0.1177	0.1214	0.2042	0.1096	0.0923
11 Platt	0.2270	0.1058	0.1157	0.2881	0.1204	0.0838
11 Binn.	0.1561	0.1450	0.1580	0.2013	0.1195	0.0920
12 Before	0.2266	0.1285	0.1085	0.2924	0.1367	0.0889
12 PAV	**0.2028**	0.1183	0.0966	0.2491	0.1333	0.0856
12 Platt	0.2654	0.1410	0.1082	0.3189	0.1367	0.0871
12 Binn.	0.2118	0.1365	0.1519	0.2634	0.1349	0.0876
13 Before	0.0240	0.0112	0.0223	**0.0122**	0.0299	0.0084
13 PAV	0.0254	0.0106	0.0361	0.0199	0.0331	0.0084
13 Platt	0.0238	0.0111	0.0222	0.0208	0.0437	0.0086
13 Binn.	0.0264	0.0166	0.0456	0.0225	0.0349	0.0087
14 Before	0.0250	0.0125	0.0212	**0.0134**	0.0316	0.0087
14 PAV	0.0287	0.0115	0.0347	0.0281	0.0347	0.0091
14 Platt	0.0242	0.0114	0.0212	0.0214	0.0449	0.0088
14 Binn.	0.0274	0.0151	0.0480	0.0226	0.0364	0.0100
15 Before	0.0465	0.0098	0.0396	**0.0119**	0.0517	0.0071
15 PAV	0.0272	0.0073	0.0715	0.0173	0.0258	0.0067
15 Platt	0.0276	0.0074	0.0396	0.0259	0.0410	0.0069
15 Binn.	0.0281	0.0086	0.0663	0.0188	0.0262	0.0066
16 Before	0.2307	0.1779	0.1793	0.2906	0.1175	0.1615
16 PAV	0.2492	0.2183	0.2127	0.3019	0.1152	0.1174
16 Platt	0.3010	0.1580	0.1796	0.3340	0.1211	0.1677
16 Binn.	**0.1932**	0.1904	0.1949	0.2696	0.1094	0.0915
17 Before	0.0009	0.0005	0.0147	0.0071	0.0148	0.0006
17 PAV	0.0010	0.0005	0.0819	0.0007	**0.0016**	0.0004
17 Platt	0.0008	0.0005	0.0147	0.0050	0.0078	0.0004
17 Binn.	0.0009	0.0005	0.0848	0.0006	0.0018	0.0004
18 Before	0.1500	0.0982	0.1027	0.1318	0.1256	0.0759
18 PAV	0.1289	0.0975	0.1045	0.1222	**0.1082**	0.0688
18 Platt	0.1700	0.0899	0.0983	0.1768	0.1214	0.0759
18 Binn.	0.1394	0.1118	0.1427	0.1383	0.1160	0.0847
19 Before	0.0597	0.0510	0.0626	**0.0514**	0.0635	0.0423
19 PAV	0.0625	0.0491	0.0686	0.0863	0.0648	0.0360
19 Platt	0.0683	0.0535	0.0626	0.1315	0.0821	0.0424
19 Binn.	0.0625	0.0632	0.1026	0.0902	0.0687	0.0443
20 Before	0.1009	0.0921	0.1308	0.1530	0.0722	0.0879
20 PAV	0.0765	0.0897	0.1407	0.1327	**0.0656**	0.0793
20 Platt	0.1027	0.0854	0.1309	0.1590	0.0838	0.0869
20 Binn.	0.0830	0.0999	0.1479	0.1318	0.0722	0.0909
AVG. Before	0.0661	0.0378	0.0539	0.0689	0.0551	0.0304
AVG. PAV	0.0649	0.0399	0.0718	0.0788	0.0502	0.0273
AVG. Platt	0.0788	0.037	0.0537	0.098	0.0601	0.0307
AVG. Binn.	0.0648	0.0449	0.0858	0.0802	0.0523	0.028

7.7 Using Negotiable Features for Prescription Problems

7. A. Bella, C. Ferri, J. Hernández-Orallo, and M.J. Ramírez-Quintana. **Using Negotiable Features for Prescription Problems.** *Computing*, 91:135–168, 2011.

Using Negotiable Features for Prescription Problems *

A. Bella, C. Ferri, J. Hernández-Orallo, and M.J. Ramírez-Quintana

Universitat Politècnica de València, DSIC, València, Spain

Abstract. Data mining is usually concerned on the construction of accurate models from data, which are usually applied to well-defined problems that can be clearly isolated and formulated independently from other problems. Although much computational effort is devoted for their training and statistical evaluation, model deployment can also represent a scientific problem, when several data mining models have to be used together, constraints appear on their application, or they have to be included in decision processes based on different rules, equations and constraints. In this paper we address the problem of combining several data mining models for objects and individuals in a common scenario, where not only we can affect decisions as the result of a change in one or more data mining models, but we have to solve several optimisation problems, such as choosing one or more inputs to get the best overall result, or readjusting probabilities after a failure. We illustrate the point in the area of Customer Relationship Management (CRM), where we deal with the general problem of prescription between products and customers. We introduce the concept of negotiable feature, which leads to an extended taxonomy of CRM problems of greater complexity, since each new negotiable feature implies a new degree of freedom. In this context, we introduce several new problems and techniques, such as data mining model inversion (by ranging on the inputs or by changing classification problems into regression problems by function inversion), expected profit estimation and curves, global optimisation through a Montecarlo method, and several negotiation strategies in order to solve this maximisation problem.

Keywords: data mining, profit maximisation, function inversion problem, global optimisation, negotiation, CRM, ranking, probability estimation, negotiable features, Montecarlo method.

1 Introduction

Complex decisions are characterised by the search of a trade-off among a set of constraints, models, rules and equations, where several computational techniques

* This work has been partially supported by the EU (FEDER) and the Spanish MEC/MICINN, under grant TIN 2007-68093-C02, the Spanish project "Agreement Technologies" (Consolider Ingenio CSD2007-00022) and the GVA project PROMETEO/2008/051.

(linear programming, simulation, numerical computation, operational research, etc.) can be used to solve the optimisation problem. More and more frequently, models are not derived by experts (from the business or scientific domain involved) but automatically inferred by data mining techniques. In this context, many optimisation techniques are no longer valid, since models are usually expressed in a non-mathematical way (e.g. decision tree) or even as a black-box (e.g. neural networks). Consequently, many techniques are no longer valid because the mathematical properties (continuity, monotonicity, ...) of the functions which describe the data mining models are unknown.

As a result of all this, the combination of data mining and (global) optimisation has recently received an increasing attention [6][22][20]. Much of this work originates from real application problems which appear in the area of Customer Relationship Management (CRM) [4][3]. CRM is an application field where econometrics and mainstream data mining can merge, along with techniques from simulation, operational research, artificial intelligence and numerical computation.

Decisions in the context of prescription problems deal about distinguishing or ranking the products to be offered to each customer (or, symmetrically, selecting the customers to whom we should make an offer), establishing the moment or sequence of the offers, and determining the price, warranty, financing or other associated features of products and customers. The classical application of data mining for prescription problems has usually considered a rather monolitic and static view of the process, where we have one or more products to be offered to a pool of customers, and we need to determine a sequence of offers (product, customer) to maximise profit. These and related problems (e.g. cross-selling or up-selling) have been addressed with techniques known as "mailing/selling campaign design" [3] or from the more general view of recommender systems [1], which are typically based on data mining models which perform good rankings and/or good probability estimations.

However, in more realistic and interactive scenarios, we need to consider that a better customisation has to be performed. It is not only the choice of products or customers which is possible, but several features of the product (or the deal) can be tuned or adapted to get more earnings. We call these features 'negotiable features', and are common in everyday applications: price, delivery time, payment method, bonuses, warranties, etc. This suggests a negotiation process over these features that we want to be automated and optimal on the side of the seller.

The consequence of this more general and interactive view is that the already complex prescription problem becomes much more difficult since we have much more variables (degrees of freedom), which have to be taken into account in the optimisation problem. And the key issue is that the relation between all these variables (price, delivery time, payment method, bonuses, warranties, etc.) and the dependent variable (e.g. buying or not) is the data mining model, which can be incarnated in the form of decision trees, neural networks, support vector machines, linear, non-linear or logistic regression, nearest neighbours, etc. Of

course we could devise a specific method to handle this when the model is, e.g., a logistic regressor, but, in general, we would like to be able to use the whole range of techniques which are available in the data mining suite at hand. That means that we have to treat the models as black boxes, not assuming any particular property. In this way, the strategies we present here are general and do not need to be adapted to each type of learning algorithm.

The notion of negotiable feature in data mining models as it is presented here is new in the literature. The term appears scarcely in the literature but with other meanings [33][8]. In these works, the focus has been set on agent negotiation strategies using a traditional data mining presentation, where the specific relation between the feature and the output is not obtained. We propose to consider negotiable features as input variables for a "general function inversion problem", i.e., given a function, calculate the input combinations which produce a given output. For instance, if we have a model which gives us the probability of buying a product by a customer, we would like to use the model to obtain the possible pairs (or ranges of price and delivery time) such that the result is a probability of 0.75 of the customer buying the product.

As a consequence of being able to adjust and customise some features of the offer, it is expected that many different offers for the same product and customer could be made, so, naturally, a negotiation process appears, where seller and buyer can make bids and counteroffers. The rejection of an offer entails a recomputation of the probabilities, something which must be done on the data mining model.

The previous new family of open problems, which are common in CRM, are also usual in other areas, where prescription has to be made. For instance, medical prescription is another area where drugs and treatments have to be customised. In this case, the "class" is not whether a patient will or not buy a drug, but whether the treatment will work or not. The agents of negotiation here are not the doctor and the patient, but the doctor and the illness. Data mining models are about treatment effectivenes, and the profit is changed into a more complex evaluation of treatment costs, counterindications and recovery periods. Similarly, other areas such as education (teacher-student), social networks (recommendations), human resources (hiring), etc., can be understood and solved under the paradigm we present in this work.

The major contributions of this work are then:

- A taxonomy of prescription problems embracing the classical static problems without negotiable features and the new interactive problems which include four new open problems depending on the number of kinds of items and customers.
- The notion of negotiable feature, its formal definition and properties, which generalises prescription problems as negotiation problems.
- The study of the function inversion problem for data mining models which, in some cases, can turn a classification problem into a regression problem (or viceversa).

– The notion of profit curves derived from probabilistic models, their normalisation after a bid and the use of envelope curves to compare and combine several curves.
– Several new strategies for negotiation using data mining models, some of them shown to be better than the baseline methods, especially the one based on a global view of the negotiation.

A real scenario is used in the paper. In this scenario, a real estate agent has a pool of housings to sell and a pool of customers. We have performed a thorough series of experiments which show that the way in which the model is deployed and applied is crucial to get good results. In fact, only with the set of techniques developed in this paper, the results using data mining models surpass the results of models based on an increment over the average (baseline methods).

The paper is organised as follows. In Section 2, we quickly survey the previous work on prescription problems and highlight a new family of prescription problems which have not been addressed to date, those for which negotiable features appear. The classical prescription problems and the new ones conform a new taxonomy which is presented in this section. Section 3 introduces the notion of negotiable feature in data mining models, its formalisation and properties. In Section 4, we analyse the use of these properties to enhance data mining models and we study several options for the inversion problem that takes place when we want to determine the value for a negotiable feature given a probability or a threshold for the output. Section 5 introduces three new different negotiation strategies (maximum expected profit, best local expected profit and maximum global optimisation) and the new scenarios derived from the taxonomy. In Section 6, we analyse our methods with real data on our example domain (real estate agent's) and compare the total profit for each method and negotiation strategy on several of the new scenarios introduced in the taxonomy. In Section 7, we analyse the connections between our work and some previous or related approaches. Section 8 closes the paper with some concluding remarks and the future work.

2 A Taxonomy of Prescription Problems

Prescription problems are a very common type of predictive problems where an item is prescribed to an individual. The item can be a commercial product, a drug, a webpage, blog or reading reference, an action, etc., depending on the application. The individual can be a user, a customer, a patient, a student, a robot, etc. Data mining models are used to model the behaviour of each individual in order to assess the appropriateness of each item for each user. Typically, prescription problems are classified according to the number of kinds of items and individuals, since the problem complexity depends on the number of combinations. However, the prescription scenario using data mining models is static, in the sense that once the prescription is made, the item is ruled out and any subsequent prescription is made on other items (this does not mean, of course,

that the interaction of previous transactions is not used for subsequent prescriptions or recommendations). In this work we do consider the case that the same item (or the very prescription) can be adapted to the individual. In order to do that, the items must present some adjustable features that we call "negotiable features" (a more precise definition will be given in the following section). If these features exist it is not only the item and the individual that we have to match but also the best values for the negotiable features.

In this section we devise a taxonomy of prescription problems depending on several variables involved in the process. This will help to recognise the previous work in this area and the open problems we aim to solve. Since most of the related work comes from the area of Customer Relationship Management (CRM), from now on, instead of using the general terms 'item' and 'individual', we will use the more specific terms 'product' and 'customer', which are typical in CRM. As usual, we consider the number of customers (C) and the different kinds of products (N). But we also study the presence or absence of negotiation in the transaction. As we have stated in the introduction, if at least one feature is negotiable, then we can introduce some kind of negotiation into the process; however, if all the features are non-negotiable (fixed), then we are dealing with a traditional prescription problem. In all these problems, there is an optimality criterion (or utility function) which shapes the goal. In CRM, the goal is usually the net profit (although other goals such as customer loyalty are possible). In general, other possible criteria might depend on other resources (time, people involved, security, etc.).

Table 1. Different prescription problems that consider the number of different kinds of products to sell, whether the net price for the product is fixed or negotiable, and the number of customers.

Case	Kinds of products	Features	Number of customers	Approach
1	1	fixed	1	Trivial
2	1	fixed	C	Customer ranking [3]
3	N	fixed	1	Product ranking [3]
4	N	fixed	C	Joint Cut-off [2]
5	1	negotiable	1	-
6	1	negotiable	C	-
7	N	negotiable	1	-
8	N	negotiable	C	-

Table 1 shows eight different prescription problems that are defined by considering the number of products and customers involved as well as the fixed or negotiable nature of the features of each product. The last column shows several approaches that have already been proposed in the literature for solving some of these problems. The rows with a "-" in this column indicate cases that are (first) addressed in this paper. We discuss each of them in more detail below.

2.1 Case with one kind of product, fixed features, and one customer

Case 1 in Table 1 is trivial. In this scenario, the seller offers the product to the customer with fixed conditions/features and the customer may or not buy the

product. The seller cannot do anything more because s/he does not have more products to sell. S/he cannot negotiate the price, warranty, delivery time, etc., of the product with the customer, and s/he does not have any more customers for the product.

2.2 Case with one kind of product, fixed features, and C customers

Case 2 in Table 1 is the typical case of a mailing campaign design. The objective is to obtain a customer ranking to determine the set of customers to whom the mailing campaign should be directed in order to obtain the maximum profit. Data mining can help in this situation by learning a probabilistic classification model[1] from previous customer data that includes information about similar products that have been sold to them. This model will obtain the buying probability for each customer. Sorting them by decreasing buying probability, the most desirable customers will be at the top of the ranking. Using a simple formula for marketing costs, we can establish a threshold/cut-off in this ranking. The customers above the threshold will be offered the product. This is usually plotted using the so-called lift charts (see e.g. [3]).

2.3 Case with N kind of products, fixed features, and one customer

Case 3 in Table 1 is symmetric to case 2. Instead of N customers and one product, in this case, there are N different products and only one customer. Hence, the objective is to obtain a product ranking for the customer. Similarly, data-mining can help to learn a probabilistic estimation model from previous product data that have been sold to similar customers. This model will predict the buying probability for each product, so by putting them in order of decreasing buying probability, the most desirable products for the customer will be at the top of the ranking. This case overlaps to a great extent with the typical applications of recommender systems [1], so many techniques can be applied here, although predictive models which show good probability estimation and ranking (typically measured with the AUC, MSE or the logloss, see e.g. [11]) are custom here.

2.4 Case with N kinds of products, fixed features, and C customers

Case 4 in Table 1 is studied in [2]. This situation is more complex than the cases 2 and 3, since there is a data-mining model for each product. In other words, there are N customer rankings (one for each product) and the objective is to obtain the set of pairs customer-product that gives the maximum overall profit. Note that, normally, the best local cut-off of each model (the set of customers that gives the maximum profit for one product) does not give the best global result. Moreover, several constraints are frequently required in real applications (limited stock of products, the customers may be restricted to only buy one

[1] A probabilistic classification model is a model which outputs the probability for the class, e.g. [10].

product). Two different methods are proposed in [2] to obtain the global cut-off: one is based on merging the prospective customer lists and using the local cut-offs, and the other is based on simulation. The study in [2] shows that using simulation to set model cut-off obtains better results than classical analytical methods.

2.5 Cases with negotiable features

In this paper, we deal with cases 5, 6, 7 and 8, which, to our knowledge, have not been addressed in the literature. These cases are similar to cases 1, 2, 3 and 4 but, in these cases, at least one feature is negotiable. This represents a complete new scenario that introduces more degrees of freedom in the search space for an optimal solution. Additionally, it usually entails a negotiation process, which usually means an episode of offers and counter-offers from the negotiation parts that makes the whole process more flexible, and logically, more difficult to analyse. Before presenting our approaches to these scenarios, next we formalise the idea of having features that can be modified for a prescription, known as "negotiable features", which make a negotiation process possible.

3 Negotiable Features

As we have already mentioned in previous sections, there are many data mining problems in which one or more input features can be modified at the time the model is applied, turning the problem into some kind of negotiation process. In this section, we formally define the concept of *negotiable feature*, and we discuss which properties are derived from their special nature and how these features can be used.

3.1 Negotiable Feature Definition

Consider a supervised problem, where $f : X_1 \times X_2 \times ... \times X_m \rightarrow Y$ denotes the target (real) function; X_i, $i \in 1..m$ denote input feature (or attribute) domains, and Y denotes the output attribute domain (or class). Values for input and output attributes will be denoted by lowercase letters. Hence, labelled instances are then tuples of the form $\langle x_1, x_2, ..., x_m, y \rangle$ where $x_i \in X_i$ and $y \in Y$.

We assume that there is a *(non-)strict total order* for the output, i.e., there is a relation $\succ$ such that for every two different possible values $y_a, y_b \in Y$, we have that either $y_a \succ y_b$ or $y_b \succ y_a$. If the order is non-strict[2], we denote it as $\succeq$. This order usually represents some kind of benefit, utility or cost. For numerical outputs, $\succ$ is usually the order relation between real numbers (either $<$ or $>$, depending on whether it is a cost or benefit). For nominal outputs, $\succ$ usually sets an order between the classes. For binary problems, where POS and NEG represent the positive and negative class respectively, we can just set that

[2] As usual, a strict total order relation can be defined from a non-strict total order.

$POS \succ NEG$. For more than two classes, the order relation can be derived from the cost of each class. For instance, if we have three buying results (buy, does not buy, chums), we can order them by their costs. Note that this does not mean that classes are necessarily ordinal (such as e.g. $\{low, medium, high\}$). Analogously, we also assume there is a *non-strict total order relation* for each input attribute X_i denoted as $\succeq_i$. For readability, in what follows we will omit the subscript when it is clear from the context.

With these definitions, we can now formalise the idea of negotiable feature.

Definition 1. (Negotiable Feature for an instance)
An attribute X_i is said to be a negotiable feature (or attribute) for an instance $\langle x_1, x_2, ..., x_m, y\rangle$ iff

1. *(adjustability) The values for X_i can be varied at model application time.*
2. *(sensitivity) Fixing the values of all the other input attributes $X_j \neq X_i$ of the instance, there are two different values for X_i producing different output values.*
3. *(monotonicity) The relation between the input feature X_i and the output Y is either*
   - *monotonically increasing: for any two values $a, b \in X_i$, if $a \succeq b$ then $f(x_1, x_2, ..., a, ..., x_m) \succeq f(x_1, x_2, ..., b, ..., x_m)$*
     or
   - *monotonically decreasing: for any two values $a, b \in X_i$, if $a \succeq b$ then $f(x_1, x_2, ..., a, ..., x_m) \preceq f(x_1, x_2, ..., b, ..., x_m)$.*

Both conditions 2 and 3 define a *monotonic dependency* between the negotiable attribute and the output.

Based on the previous definition, we introduce the concept of negotiable feature for a problem.

Definition 2. (Negotiable Feature in a problem)
Let D be a set of instances defining a supervised problem f. We will say that a feature is a negotiable feature in f, if conditions 1 and 3 hold for all the instances in D, and 2 holds for a significant/relevant proportion $0 < \tau \leq 1$ of instances in D.

This value τ is determined by the user depending on the problem, the presence of noise, etc. Note that the relaxation of condition 2 (sensitivity) is due to the fact that, especially in classification problems, for some instances, the output cannot be changed by only changing the value of the negotiable attribute, i.e., the attribute cannot be negotiable for some instances (but it is for the remaining). For example, if someone cannot pay more than 500 euros per year for a life insurance, he will not sign a 1000-euro life insurance contract even if he is offered an extra accident coverage. In some case, we can have dynamic negotiable features, i.e., features that were not negotiable initially, but a special condition or charge in other attributes can make these features negotiable. For example, in the airline and railway industry, a drop in the price might make the warranty-feature negotiable.

Next, we give an example of negotiable and non-negotiable features.

Example 1. In a loan granting model, where loans are granted or not according to a model which has been learnt from previous customer behaviours, the age of the customer is not a negotiable feature, since we cannot modify it (condition 1 is violated). The bank branch office where the contract can take place is also an input, which we can modify, but it is not a negotiable feature either since it rarely affects the output (property 2 is violated). The number of meetings is also modifiable and it frequently affects the result, but it is usually non-monotonic, so it is not a negotiable feature either (property 3 is violated). In contrast, the loan amount, the interest rate or the loan period are negotiable features since very large loan amounts, very low interest rates and very long loan periods make loans unfeasible for the bank.

The definition of an order relation for inputs and outputs is usually straightforward and it allows different tasks to be seen in a uniform way. For instance, we can have the following possibilities depending on the nominal or numerical character of the input and output.

- Nominal negotiable feature and nominal output. In this classification task, there is a relation between some particular values of the negotiable attribute and some values of the class. For example, a travel agency survey will be negative whenever the traveller finds overbooking at the airport. Negotiation can take place if we ensure that no overbooking will take place.
- Numerical negotiable feature and nominal output. In this classification task, there is a relation between the range of values of the negotiable attribute and some values of the class. The loan amount, the interest rate and the loan period are examples of negotiable features for the loan problem.
- Nominal negotiable feature and numerical output. In this regression task, there is a relation between some particular values of the negotiable attribute and the range of the output. For instance, the time for a delivery monotonically depends on whether we use regular mail or fast courier. This feature can be a negotiable feature.
- Numerical negotiable feature and numerical output. In this regression task, there is a relation between the range of values of the negotiable attribute and the range of the output. For example, the time for a cargo train trip monotonically depends on the number of station stops or the cargo load.

Although we only show the four possible cases for supervised problems, the first two cases are extensible to clustering, since if we have a model, we can understand groups as classes, and use it as a classification model.

3.2 Negotiable Feature Properties

Given the previous definitions, we can analyse what we can do with negotiable features and how we can exploit their special character.

The first interesting property is that, since they are monotonic and sensitive (in general), i.e., they are monotonically dependent, the inversion problem is

possible. This means that questions such as: "tell me the maximum amount for the loan" or "tell me the maximum number of items such that the order can be delivered in one week" or "tell me the price such that there is a 0.75 probability of selling" or "tell me the dose such that the blood pressure goes down to 12" are not only sensible and useful, but solvable too. In the next section we will explore some possible solutions.

The second interesting property is also directly derived from monotonicity, and is related to the idea that negotiable features have a negotiation direction, such that once reached a minimum or maximum output, it is useless to go on negotiating on that feature. For instance, if we can grant a loan for 200,000 euros we can also grant a loan for 150,000 euros. It is especially advantageous for classification problems, but it also happens for many regression problems.

In order to clarify this implication, consider that we have a minimum or maximum (or both) for the output feature, using the derived order. The maximum is just defined as $max(Y) = y_i$ such that for all j we have that $y_i \succeq y_j$. Similarly we can define the minimum. For numerical outputs (i.e., regression), we usually have one of them, e.g. minimum price, minimum delivery time, etc., but we can also have both (e.g. minimum and maximum load). For nominal outputs, since the number of classes is finite, we always have a minimum and a maximum. In binary classification problems, the maximum is class POS and the minimum is class NEG. With this definition, we can show the following results:

Proposition 1. *Consider that there exists a maximum value for the output feature Y, denoted by y_{max}, and there is a negotiable feature X_i which is monotonically increasing (respectively monotonically decreasing) wrt. Y.*

If $f(x_1, \ldots, x_{i-1}, a, x_{i+1}, \ldots, x_m) = y_{max}$ then for every b, such that $b \succeq a$ (respectively $a \succeq b$) we also have that $f(x_1, \ldots, x_{i-1}, b, x_{i+1}, \ldots, x_m) = y_{max}$.

Proof. If the relation is monotonically increasing, we have that for any two values a, b for X_i, if $b \succeq a$ then

$$f(x_1, \ldots, x_{i-1}, b, x_{i+1}, \ldots, x_m) \succeq f(x_1, \ldots, x_{i-1}, a, x_{i+1}, \ldots, x_m)$$

Since $f(x_1, \ldots, x_{i-1}, a, x_{i+1}, \ldots, x_m) = y_{max}$ we have that

$$f(x_1, \ldots, x_{i-1}, b, x_{i+1}, \ldots, x_m) \succeq y_{max}$$

but since y_{max} is the maximum, then $f(x_1, \ldots, x_{i-1}, b, x_{i+1}, \ldots, x_m) = y_{max}$. If the relation is monotonically decreasing the proof is similar.

Proposition 2. *Consider that there exists a minimum value for the output feature Y, denoted by y_{min}, and there is a negotiable feature X_i which is monotonically increasing (respectively monotonically decreasing) wrt. Y.*

If $f(x_1, \ldots, x_{i-1}, a, x_{i+1}, \ldots, x_m) = y_{min}$ then for every b, such that $a \succeq b$ (respectively $b \succeq a$) we also have that $f(x_1, \ldots, x_{i-1}, b, x_{i+1}, \ldots, x_m) = y_{min}$.

Proof. The proof is similar to the previous proposition.

Let us see an example.

Example 2. Following with the loan problem, the loan amount (denoted by δ) is a negotiable attribute which is monotonically decreasing wrt. the class (POS means granting the loan and NEG means not granting it). In this case, $\succeq$ for the negotiable attribute is the standard order relation for real numbers $\geq$. For the class we have that $POS \succ NEG$. Consequently, from the previous propositions, we can derive the following rules for this case:

- If $f(x_1, \ldots, x_{i-1}, \delta_a, x_{i+1}, \ldots, x_m) = POS$ then for every δ_b, such that $\delta_a \geq \delta_b$ we also have that $f(x_1, \ldots, x_{i-1}, \delta_b, x_{i+1}, \ldots, x_m) = POS$, which means that, for a given customer, if we can grant him a loan for a quantity δ_a we can also grant him a loan for a lower quantity.
- If $f(x_1, \ldots, x_{i-1}, \delta_a, x_{i+1}, \ldots, x_m) = NEG$ then for every δ_b, such that $\delta_b \geq \delta_a$ we also have that $f(x_1, \ldots, x_{i-1}, \delta_b, x_{i+1}, \ldots, x_m) = NEG$, which means that, for a given customer, if we cannot grant him a loan for a quantity δ_a we cannot grant him a loan for a higher quantity either.

These rules derived from the previous two propositions will be important in the following approaches to address negotiable feature models and also to derive negotiation strategies, as we will see below in Section 4.

4 Approaches to Negotiable Feature Models

In this section we will explore three possible data mining approaches to obtain and use models in presence of negotiable features. We will focus on classification models where the negotiable feature is numerical and continuous, since this case is very frequent and its inversion problem is the most difficult one (we must go from a small set of output categories to an infinite set of input values). In the case of discrete numerical values (e.g. a discrete number of items) the model prediction (or, more precisely, the chosen value by the negotiation strategy) must be rounded up or down to the closest integer value.

Additionally, we will consider the classification model as a probabilistic model, i.e., a model which outputs probability estimations for the classes.

We will present three general approaches that use negotiable features. At first, traditional approach will be to treat the problem as a classical classification problem. A second approach is similar but using the rules derived from the negotiable feature properties in order to generate more examples and make learning easier. Finally, a third approach transforms the classification problem into a regression one, inverting the original problem.

4.1 Traditional Approach by Using the Monotonicity Property

The first approach just learns a model as usual, by using the available labelled examples to train it, and then uses this model for predicting the output value of future cases.

However, this approach has two drawbacks: (1) the inverse problem (that is, to obtain the value of the negotiable attribute for which a certain output

value is obtained) must be solved by iteratively checking different values for the negotiable feature and just selecting the one which produces the desired output, and (2) in many situations we only have examples of one class available.

The first problem can be minimised since we know that the relation between the negotiable feature and the output is monotonic. This means that instead of making an exhaustive search, we only need a binary or gradient search.

The second problem, however, is more dangerous, since in many cases it precludes from learning a useful model. Consider again the loan granting problem presented in Examples 1 and 2. It is usually the case that the bank only records past granted loans. If we use this information as it is, this means that we only have examples from the positive class, so learning is impossible (at least as a supervised problem). Usually, this is minimised by slightly changing the meaning of the class. Instead of using whether a loan has been granted or not, we change it into whether a past granted loan has been or not benefitial for the bank. But even with this typical transformation, we have a quite biased apriori distribution (a type of selection bias known as sample bias [15]), since many possible good and bad customers who did not get the loan have been ruled out from the dataset.

So, in this case, asking questions such as "what is the maximum loan amount we can grant to this customer?" or "which loan amount places this operation at a probability of 0.95 of being a profitable customer?" are more difficult to answer and more dangerous. Precisely, these questions fly around the limit between the information we have in the records and the information we do not have. Typically, in order to answer the previous questions we can draw a curve with the probability of being a profitable customer against the possible values of the negotiable feature. But part of the curve has no supporting evidence. Consequently, models constructed in this way are expected to behave bad for these questions.

4.2 Traditional Approach by Deriving More Examples

According to the second problem formulated in the previous section, and the rules that we can obtain by applying Propositions 1 and 2, we propose a new way of generating more examples which can be used to learn the classification model in a traditional approach. If we recall Proposition 1 we have that: If $f(x_1, \ldots, x_{i-1}, a, x_{i+1}, \ldots, x_m) = y_{max}$ then for every b, such that $b \succeq a$ (respectively $a \succeq b$) we also have that $f(x_1, \ldots, x_{i-1}, b, x_{i+1}, \ldots, x_m) = y_{max}$. That means that if we have an example in the training set as: $f(x_1, \ldots, x_{i-1}, a, x_{i+1}, \ldots, x_m) = y_{max}$ we can generate as many *new examples* just changing a for as many b as we like, just that $b \succeq a$ (respectively $a \succeq b$). And the same for the other property.

This means that for binary problems, we can always use one of the two properties and convert one example into hundreds or thousands of examples. In this way, we introduce some knowledge about the relation between the negotiable feature and the output in the training dataset and, as a consequence, into the learnt model. Additionally, this generation can be done in such a way that we can

compensate the apriori class distribution bias. Consequently, probability estimation can now take advantage of much more examples, much more information and, hopefully, a less biased dataset.

4.3 Inverting Problem Presentation

A quite different approach is to think about the problem as an inversion problem from scratch. Imagine a model which estimates the delivery time for an order depending on the kind of product and the units which are ordered. One possible (traditional) use of this model is to predict the delivery time given a new order. However, another use of this model is to determine the number of units (provided it is possible to play with this value) that can be delivered in a fixed period of time, e.g. one week. This is an example of an "inverse use" of a data mining model, where all inputs except one and the output are fixed, and the objective is to determine the remaining input value.

Definition 3. *(***Inversion problem***) Given a supervised problem* $f : X_1 \times X_2 \times ... \times X_i \times ... \times X_m \rightarrow Y$*, where* X_i *is the negotiable feature, the inversion problem consists in defining the function* $f^I : X_1 \times \ldots \times X_{i-1} \times Y \times X_{i+1} \times \ldots \times X_m \rightarrow X_i$.

In the above example, f is the function that calculates the delivery time of an order, the negotiable feature X_i is the number of delivered units and f^I calculates this number by considering the delivery time fixed. As mentioned in Section 3.2, the conditions of monotonicity and sensitivity that are satisfied by the negotiable attributes, will enable us to solve this problem, as we explain below.

The inversion problem is well-known [9] and seems simple at first sight, but many questions arise. First, is f^I also a function? In other words, for two different values for X_i we may have the same value for Y which will ultimately translate into two inconsistent examples for f^I (two equal inputs giving different outputs). Second, the fact that we have an example saying that a given loan amount was granted to a customer does not mean that this is the maximum amount that could be granted to the customer. Third, deriving probabilities to answer questions such as "which loan amount places this operation at a probability of 0.95 of being a profitable customer?" seem to be unsolvable with this new presentation.

Although, it may seem hard to overcome these problems, looking at these issues more carefully, we see that there is still a possible solution which is to consider the inverting problem as a regression one. This is so because, first, many regression techniques work well for inconsistent examples, so this question is not actually a big practical problem. Second, it is true that cases do not represent the maximum amount, but in many cases the examples represent deals and they are frequently not very far away from the maximum. Or, in the worst case, we can understand the new task as "inferring" the typical value for X_i such that the loan is granted to the customer. And third, we can also obtain probabilities in a regression problem.

Then, if we invert the problem, how can we address the original problem again? With the original model and for only two classes, it can be done by calculating $p(POS|\langle x_1, \ldots, x_{i-1}, a, x_{i+1}, \ldots, x_m\rangle)$, for any possible value $a \in X_i$. From the inverted (regression) problem, we get a prediction:

$$\hat{a} = f^I(x_1, \ldots, x_{i-1}, POS, x_{i+1}, \ldots, x_m)$$

If we think of $\hat{a}$ as the predicted maximum or minimum for a which makes a change on the class, a reasonable assumption is to give 0.5 probability for this point, namely $p(POS|\langle x_1, \ldots, x_{i-1}, \hat{a}, x_{i+1}, \ldots, x_m\rangle) = 0.5$.

The next step is to assume that the output for f^I follows a distribution with centre at $\hat{a}$. For instance, we can assume a normal distribution with mean at $\hat{a}$ and use the relative error (ρ) (on the training set) multiplied by $\hat{a}$, as standard deviation σ. In other words, we use $N(\hat{a}, \rho * \hat{a})$. Our assumption that the values of the negotiable attribute can be modelled by a normal distribution is a working hypothesis which allows us to derive the probabilities in an almost direct way. There are, of course, other alternative ways of estimating the distribution parameters by using a relative squared error as variance or we could use techniques such as bootstrapping. Note that we estimate a different standard deviation for each example (since this is relative to the predicted value $\hat{a}$). It is difficult to get a reliable and specific estimation for each example in that, assuming the use of any particular kind of data mining techniques, since there are many methods which do not output a reliability or expected error for each instance.

From here, we can derive the probability for any possible value a as the cumulative distribution function derived from the above normal, i.e., $\Phi_{\hat{a},\rho*\hat{a}}$.

Figure 1 shows an example of a normal distribution with centre at $\hat{a} = 305,677.9$ and standard deviation $\sigma = 59,209.06$ and its associated cumulative distribution function.

Consequently, for solving the original problem, (1) we solve the inversion problem directly and (2) we use the predicted value of the negotiable feature as mean of a normal distribution with the relative error as a relative standard deviation. We call this model *negotiable feature model*.

5 Prescription Problems with Negotiation

In this section, we propose a solution to solve the problems 5 to 8 that are described in Table 1. Our solution is based on the ideas we have presented about models that use negotiable features in the previous sections. The objective is to integrate the induced models into a negotiation process such that these models guide the negotiation. To do this, we first introduce the negotiation strategies we will use and, then, we describe the different scenarios that cover all cases. We are going to illustrate our proposals by using a general CRM problem of retailing (prescription applied to a plot selling scenario), where the (negotiable) input feature is the price (denoted by π) and the problem is a classification problem (buying or not).

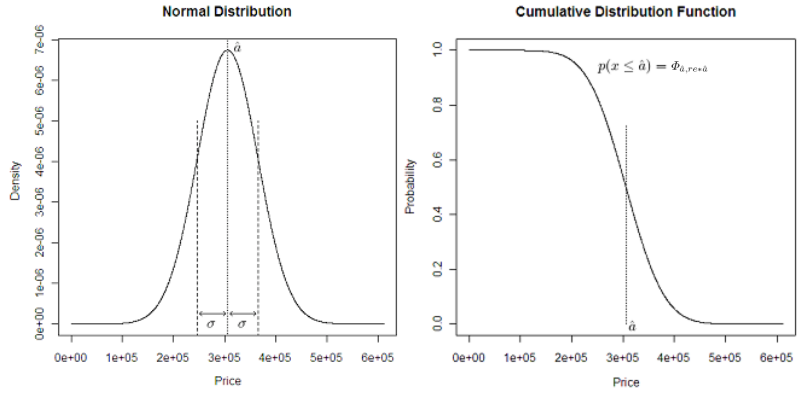


Fig. 1. Left: Example of a normal distribution $\hat{a} = 305,677.9$ and $\sigma = 59,209.06$. Right: Associated cumulative distribution function.

In our problem, we know that customers have a "maximum price" per flat they are not meant to surpass. This maximum price is not known by the seller, but estimated with the data mining models. Conversely, the seller (real estate agent) has a "minimum price" (denoted by π_{min}) for each type of product, which typically includes the price the owner wants for the house plus the operational cost. This minimum price is not known by the buyer. Any increment over this minimum price is profitable for the seller. Conversely, selling under this value is not acceptable for the seller. Therefore, the seller will not sell the product if its price is under this minimum price that s/he knows. This means that the profit obtained by the product will be the difference between the selling price and the minimum price: $Profit(\pi) = \pi - \pi_{min}$. Finally, in case that the maximum price is greater than the minimum price, there is a real chance of making a deal, and the objective for the seller is to maximise the expected profit, which is defined as follows:

$$E\_Profit(\pi) = \hat{p}(POS|\langle x_1, \ldots, x_{i-1}, \pi, x_{i+1}, \ldots, x_m\rangle) \cdot Profit(\pi) \qquad (1)$$

where $\hat{p}$ is the estimated probability by the model.

5.1 Negotiation Strategies

The novel thing in this scenario is not only that we allow the seller to play or gauge the price to maximise the expected profit, but we also allow several bids or offers made to the same customer. This means that if an offer is rejected, the seller can offer again. The number of offers or bids which are allowed in an application is variable, but it is usually a small number, to prevail the buyer from getting tired of the bargaining.

We propose three simple negotiation strategies in this setting. For cases with one single bid, we introduce the strategy called "Maximum Expected Profit" (MEP). For cases with more bids (multi-bid) we present two strategies: "Best Local Expected Profit" (BLEP) strategy and "Maximum Global Optimisation" (MGO) strategy. Let us see all of them in detail below:

- Maximum Expected Profit (MEP) strategy (1 bid). This strategy is typically used in marketing when the seller can only make one offer to the customer. Each price for an instance gives a probability of buying. This strategy chooses the price that maximises the value of the expected profit. $\pi_{MEP} = argmax_{\pi}(E\_Profit(\pi))$. In Figure 2 right, the black dot is the MEP point (the maximum expected profit point). Note that, in this case, this price is between the minimum price (represented by the dashed line) and the maximum price (represented by the dotted line), which means that this offer would be accepted by the buyer.
- Best Local Expected Profit (BLEP) strategy (N bids). This strategy consists in applying the MEP strategy iteratively, when it is possible to make more that one offer to the buyer. The first offer is the MEP, and if the customer does not accept the offer, his/her curve of estimated probabilities is normalised taking into account the following: the probabilities of buying that are less than or equal to the probability of buying at this price will be set to 0; and the probabilities greater than the probability of buying at this price will be normalised between 0 and 1. The next offer will be calculated by applying the MEP strategy to the normalised probabilities. When the probability of buying which is associated to the price is the maximum probability (this is an infrequent situation) we cannot use the model any more, and the price will not be set to 0, because the expected profit would always be 0. Instead of this, the next offer is directly the half of the bid price. The pseudo-code is in Algorithm 1. Figure 3 left shows the three probability curves obtained in three steps of the algorithm and Figure 3 right shows the corresponding expected profit curves. The solid black line on the left chart is the initial probability curve and the point labeled by 1 on the right chart is its MEP point. In this example, the offer is rejected by the customer (this offer is greater than the maximum price), so probabilities are normalised following the process explained above. This gives a new probability curve represented on the left chart as a dashed red line and its associated expected profit curve (also represented by dashed red line on the chart on the right), with point 2 being the new MEP point for this second iteration. Again, the offer is not accepted and the normalisation process is applied (dotted green lines in both charts). In order to illustrate the case where the normalisation plummets the probabilities too, Figure 4 shows the BLEP strategy when the probability associated to the MEP point in each iteration is the maximum probability.

– Maximum Global Optimisation (MGO) strategy (N bids). The objective of this strategy is to obtain the N offers that maximise the expected profit:

$$\begin{aligned}\pi_{MGO} &= argmax_{\langle\pi_1,\ldots,\pi_N\rangle}(E\_Profit(\langle\pi_1,\ldots,\pi_N\rangle)\\ &= argmax_{\langle\pi_1,\ldots,\pi_N\rangle}(\hat{p}(POS|\pi_1)\cdot Profit(\pi_1)\\ &\quad +(1-\hat{p}(POS|\pi_1))\cdot\hat{p}(POS|\pi_2)\cdot Profit(\pi_2)+\ldots\\ &\quad +(1-\hat{p}(POS|\pi_1))\cdot\ldots\cdot(1-\hat{p}(POS|\pi_{N-1}))\cdot\\ &\quad \hat{p}(POS|\pi_N)\cdot Profit(\pi_N))\end{aligned}$$

The rational of the previous formula is that we use a probabilistic accounting of what happens when we fail or not with the bid. Consequently, optimising the previous formula is a global approach to the problem.
Computing the N bids from the previous formula is not direct but can be done in several ways. One option is just using a Montecarlo approach [19] with a sufficient number of tuples to get the values for the prices that maximise the expected profit. Figure 5 right shows the three points given by the MGO strategy for the probability curve on Figure 5 left. As we can see, the three points are sorted in decreasing order of price.

For the three previous strategies, it is clear that they will only work if the data mining models perform relatively accurate predictions in terms of probability estimation[3].

Next, we will investigate the application of the previous three methods to the last four cases in Table 1.

Algorithm 1: BLEP strategy
Require: N, epf (estimated probability function or curve)
Ensure: π_{BLEP}
  $\forall x, epf(x) \leftarrow \hat{p}(POS|x)$
  $\pi_1 \leftarrow \pi_{MEP}$
  $\pi \leftarrow \pi_1$
  for $\pi_i, i \in 2..N$ **do**
    if $epf(\pi) \neq \max_{x\in 0..\infty}(epf(x))$ **then**
      $\forall x, epf(x) \leftarrow 0$
      if $epf(x) \leqslant epf(\pi)$ **then**
        $epf \leftarrow normalise(epf, epf(\pi), \max_{x\in 0..\infty} epf(x))$
        {$normalise(f(x), mix, max)$: returns normalised function of $f(x)$ from values min and max to [0..1]}
      end if
      $\pi_i \leftarrow \pi_{MEP}$
      $\pi \leftarrow \pi_i$
    else
      $\pi_i \leftarrow \pi \div 2$
      $\pi \leftarrow \pi_i$
    end if
  end for
  $\pi_{BLEP} \leftarrow \langle\pi_1,\ldots,\pi_N\rangle$

[3] In the last two strategies, we do not consider whether the offer is below the seller's minimum price. Strictly, this is not part of the strategy but it is rather more related to ascertain which of cases 5, 6, 7 or 8 we are dealing with, and also with the issue of whether we have other customers and we prefer to stop offering the product that to get closer to the minimum price.

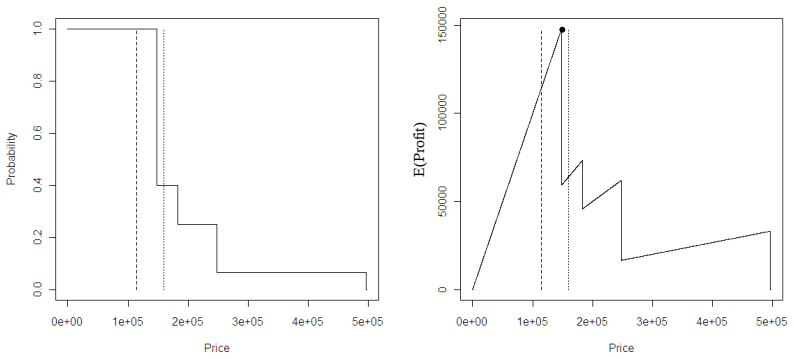


Fig. 2. Example of the MEP strategy. **Left**: Estimated probability. **Right**: Associated expected profit.

5.2 Scenario with one Product and one Customer

We start with the simplest negotiation scenario, where there are only one seller and one buyer who both negotiate for one product. The buyer is interested in one specific product. S/he likes the product and s/he will buy the product if its price is under a certain price that s/he is willing to pay for this product. It is clear that in this case the price holds the conditions to be a negotiable feature. It is sensitive, since if we reduce price to 0, the probability of having class POS approaches 1 and if we increase price to a very large amount, the probability of having class NEG approaches 1. And, finally, it is monotonic, since the relation between price and the class order $NEG \prec POS$ is monotonically decreasing. Since product and customer are fixed, we only have one degree of freedom: the price.

Obviously, the goal of the seller is to sell the product at the maximum possible price (denoted by π_{max}) which is defined as the value such that both the following equalities hold:

$f(x_1, \ldots, x_{i-1}, \pi_{max}, x_{i+1}, \ldots, x_m) = POS$

$f(x_1, \ldots, x_{i-1}, \pi_{max} + \epsilon, x_{i+1}, \ldots, x_m) = NEG, \forall \epsilon > 0.$

In other words, the use for the model is: "Which is the maximum price at which I can sell this product to this customer?" Logically, the higher the price the lower the probability. So the goal, as we said at the beginning of Section 5, is to maximise the expected profit calculated by formula (1) where $\hat{p}$ is the estimated probability given by the negotiable feature model.

To ease notation we will denote $\hat{p}(POS|\langle x_1, \ldots, x_{i-1}, \pi, x_{i+1}, \ldots, x_m\rangle)$ as $\hat{p}(POS|\pi)$. Consequently, we can express formula (1) as:

$E(Profit(\pi)) = \hat{p}(POS|\pi) \cdot Profit(\pi)$,

with the additional constraint, as mentioned, that $\pi \geq \pi_{min}$.

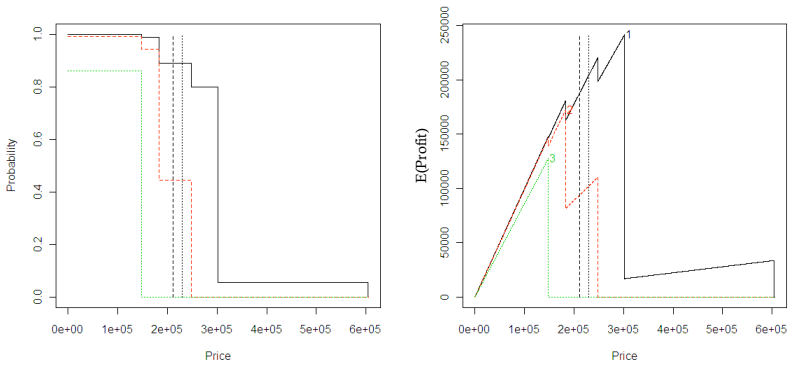


Fig. 3. Example of the BLEP strategy. **Left**: Estimated probability. **Right**: Associated expected profit.

So, if we have a model which can estimate probabilities for the positive class, we can use the previous formula for the expected profit to choose the price that has to be offered to the customer. If probabilities are well estimated, for all the range of possible prices, this must be the optimal strategy if there is only one bid. In Figure 6 we show an example of the plots that are obtained for the estimated probabilities and expected profit.

5.3 Scenario with several Products and/or several Customers

In this section we are going to study the cases 6, 7 and 8 in Table 1. The cases 6 and 7 correspond to the cases with more than one product or more than one customer, while in the case 8 there are several products and several customers. As we will see, they can be understood as an extension of case 5 combined with the rankings of customers and products that are used in cases 2 and 3 in Table 1.

In case 6 in Table 1 (one kind of product, negotiable price, and C customers), there is a curve for each customer (Figure 7, Left), which are similar to the curve in case 5. If the seller can only make one offer to the customers, the seller will offer the product at the price that gives the maximum expected profit (in relation to all the expected profit curves) to the customer whose curve achieves the maximum. However, if the seller can make several offers, the seller will distribute the offers along the curves following a negotiation strategy. In this case, the seller not only changes the price of the product, but the seller can also change the customer that s/he is negotiating with, depending on the price of the product (that is, by selecting the customer in each bid who gives the greatest expected profit at this price). Therefore, these curves can be seen as a ranking of customers for each price.

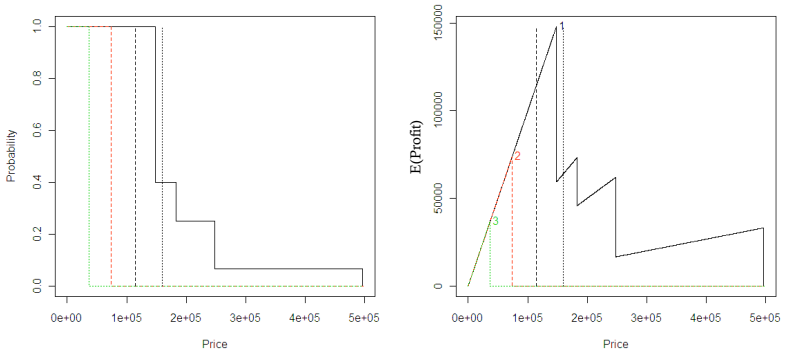


Fig. 4. Example of the BLEP strategy. **Left**: Estimated probability. **Right**: Associated expected profit.

Case 7 in Table 1 (N kind of products, a negotiable price, and one customer) is symmetric to case 6. Instead of one curve for each customer, there is one curve for each product. In this case, the curves represent a ranking of products for that customer. The learned data-mining models will help the seller to make the best decision about which product the seller offers to the customer and at what price. Figure 7 is also an example of this case since the curves would represent three different products to be offered to one customer.

Case 8 in Table 1 (N kind of products, a negotiable price, and C customers) is the most complete of all.

The objective is to offer the products to the customers at the best price in order to obtain the maximum profit. Multiple scenarios can be proposed for this situation: each customer can buy only one product; each customer can buy several products; if the customer buys something, it will be more difficult to buy another product; there is limited stock; etc. But if we understood it as the other two, the way in which it is solved does not differ to cases 6 and 7.

In cases 6, 7 and 8, we typically work with only one data mining model which has the customer's features and the product's features (one of them being the negotiable feature) as inputs. We can, of course, define C different models in case 6, N different models in case 7, or even C, N or CxN different models for case 8. Nonetheless, this is not necessary and the higher the number of models is the more difficult is to learn and use them and is prone to overfitting.

As a result, to solve cases 6, 7 and 8, we propose extending the classical concept of ranking customers or products to expected profit curves in order to obtain a ranking of customers or products for each price (similar to cases 2 and 3). For example, Figure 7 shows that, for a price of 300,000 euros the most appropriate customer is the one represented by the solid line, the second one is the customer represented by the dotted line, and the least appropriate one

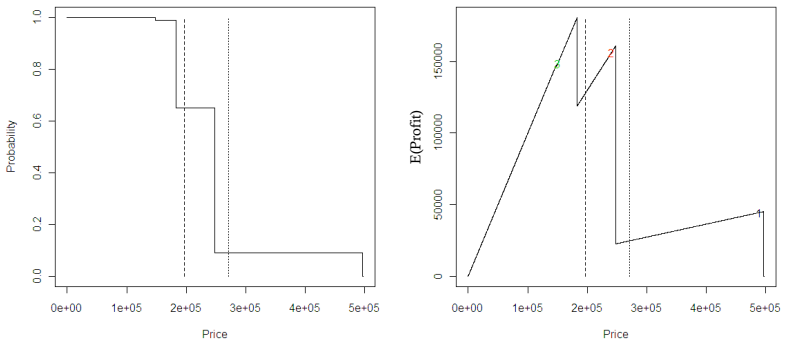


Fig. 5. Example of the MGO strategy. **Left**: Estimated probability. **Right**: Associated expected profit.

is represented by the dashed line. The situation changes for a price of 200,000 euros; at that point the most appropriate customer is the one represented by the dashed line, the second one is the customer represented by the solid line, and the least one is the one represented by the dotted line. Therefore, an important property of these probabilistic buying models is that there is a change in the ranking at the point where two curves cross.

Graphically, the most appropriate customer or product (the top of the ranking) for each price is represented by the envelope curve. Therefore, in the cases 6, 7 and 8 there are several curves, but the envelope curve must be calculated having, as a result, only one curve. Consequently, we can apply the same negotiation strategies applied to the case 5 to the envelope curve of cases 6, 7 and 8.

Example 3. We are going to explain the negotiation strategy that the seller will follow by means of an example of the case 6 (one kind of product, a negotiable price, and C customers), because the process will be the same for the cases 7 and 8, but with more curves. Therefore, we start with the simplest situation with two customers and one product.

In Figure 8, there are two curves representing the buying probabilities of two different customers. The buying probability of the first customer follows a normal distribution with $\mu_1 = 400$ and $\sigma_1 = 100$, and it is represented by a dashed line. The buying probability of the second customer follows a normal distribution with $\mu_2 = 300$ and $\sigma_2 = 200$, and it is represented by a dotted line. These are the probabilities; however, the actual values (unknown by the seller) is that the maximum buying price for customer 1 is 100 euros and 150 euros for customer 2.

We assume a simple negotiation process for this example. The negotiation strategy that we use is the Best Local Expected Profit (BLEP) strategy explained

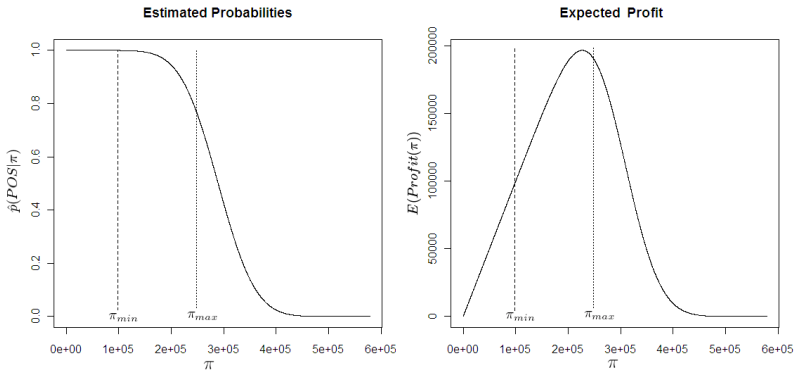


Fig. 6. Left: Example of estimated probabilities. Right: Associated expected profit. The minimum and maximum price are also shown.

in section 5.1. The trace of the negotiation process is described in Table 2 (Left) and shown graphically in Figures 9 and 10. In each iteration, the maximum of the functions is calculated (the envelope curve). The envelope curve is represented by a solid line in Figures 9 and 10.

Table 2. Left: Trace of the negotiation process. **Right:**Trace of the negotiation process with the ordering pre-process.

Offer	Price	Customer	Accepted
1	309	1	No
2	214	1	No
3	276	2	No
4	149	1	No
5	101	1	No
6	150	2	Yes

Offer	Price	Customer	Accepted
1	309	1	No
2	276	2	No
3	214	1	No
4	150	2	Yes

Note that as Table 2 (Left) shows, the third offer is greater than the second one. This is because there is more than one customer in the negotiation process and the offer is made at the price that maximises the expected profit at each iteration. Therefore, it is easy to propose an improvement for this negotiation strategy with a limited number of offers, which is similar to BLEP with n bids. This improvement is a pre-process that consists in calculating the n points and ordering them by the price before starting the negotiation. Following the example shown in Table 2 (Left), if there are only 4 bids no one will buy the product. However, with this improvement (the pre-process) customer 2 will buy the product at a price of 150 euros as shown in Table 2 (Right).

This negotiation scenario suggests that other negotiation strategies can be proposed for application to problems of this type in order to obtain the maximum

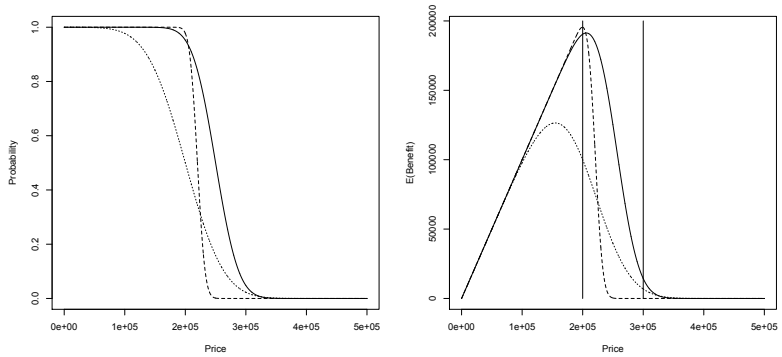


Fig. 7. Probabilistic buying models of 3 different customers approximated by 3 normal distributions with $\mu_1 = 250,000$ and $\sigma_1 = 30,000$, $\mu_2 = 220,000$ and $\sigma_2 = 10,000$, and $\mu_3 = 200,000$ and $\sigma_3 = 50,000$. **Left**: Probability distribution function. **Right**: Associated expected profit.

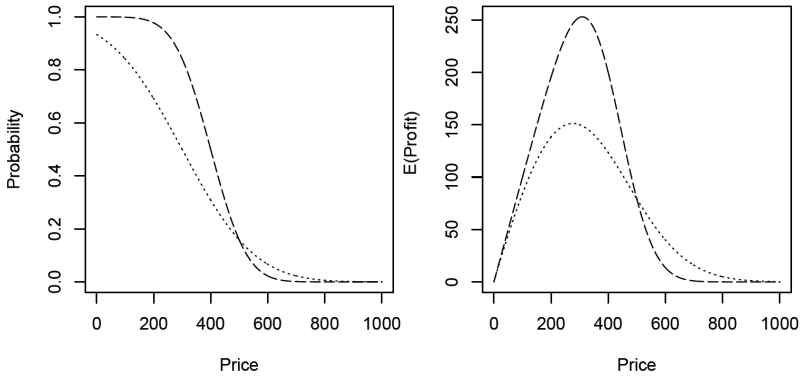


Fig. 8. Probabilistic buying models of 2 different customers approximated by 2 normal distributions with $\mu_1 = 400$ and $\sigma_1 = 100$ (dashed line), and $\mu_2 = 300$ and $\sigma_2 = 200$ (dotted line). **Left**: Probability distribution function. **Right**: Associated expected profit.

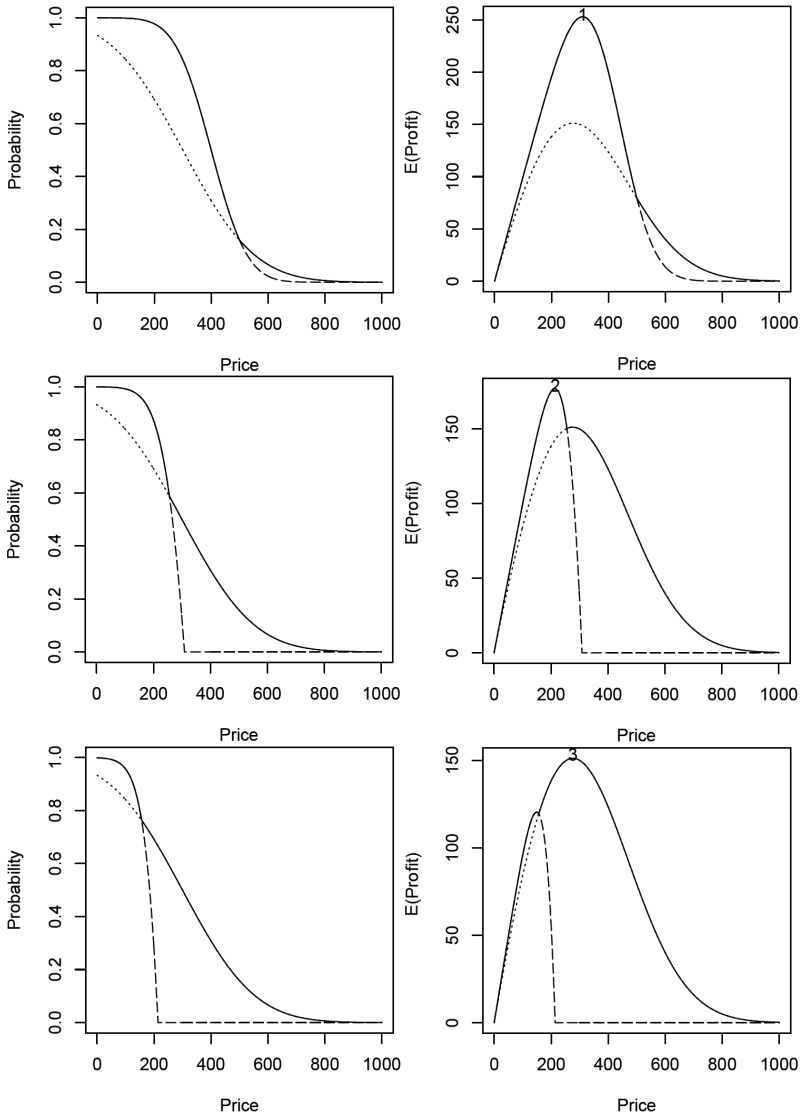


Fig. 9. Points 1, 2 and 3 in the negotiation process. **Left**: Probability distribution function. **Right**: Associated expected profit.

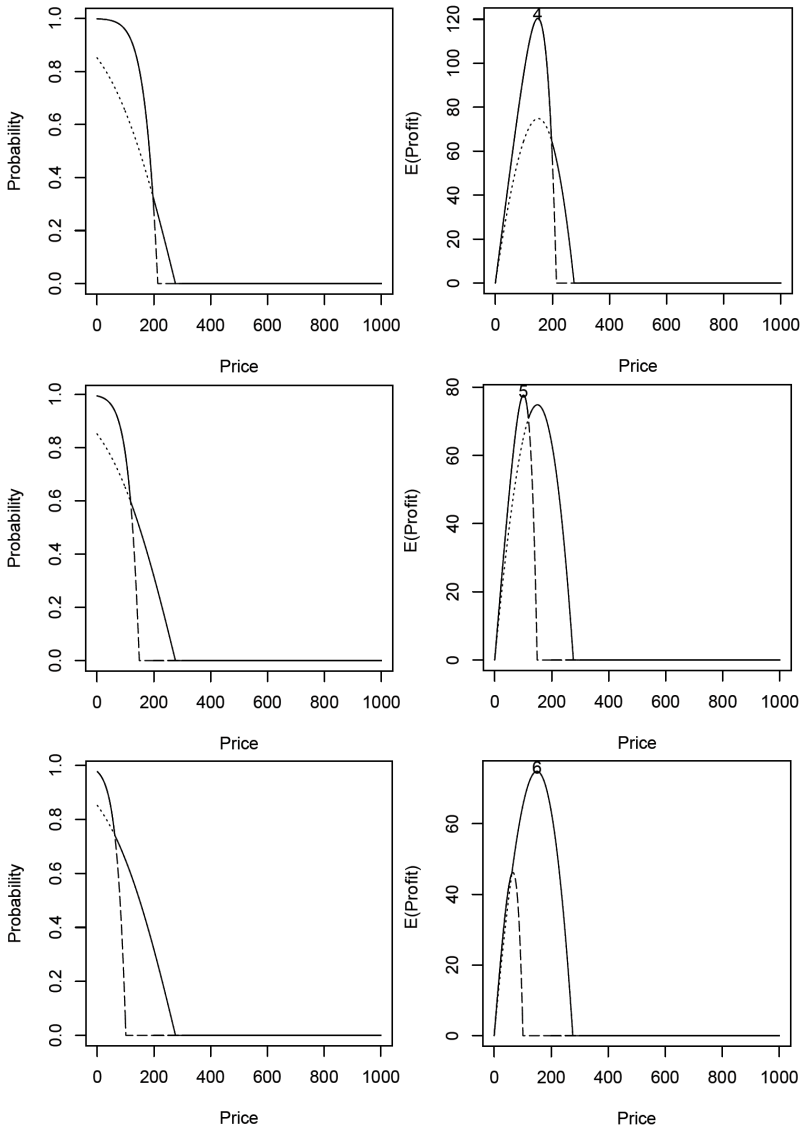


Fig. 10. Points 4, 5 and 6 in the negotiation process. **Left**: Probability distribution function. **Right**: Associated expected profit.

profit. One problem with the BLEP strategy is that it is very conservative. It might be interesting to implement more aggressive strategies that make offers at higher prices (graphically, more to the right). A negotiation strategy that attempts to do this is the Maximum Global Optimisation (MGO) strategy (with n bids). The objective of this strategy is to obtain the n offers that maximise the expected profit by generalising an optimisation formula that was presented in section 5.1.

In case 6 (One kind of product, a negotiable price, and C customers), we have presented an example with two customers and one product, but it would be the same for more than two customers. In the end, there would be one curve for each customer or product, and the same negotiation strategies could be applied.

Case 7 (N kind of products, a negotiable price, and one customer) is the same as case 6, but the curves represent the buying model of each product for each customer, and a ranking of products will be obtained for each price.

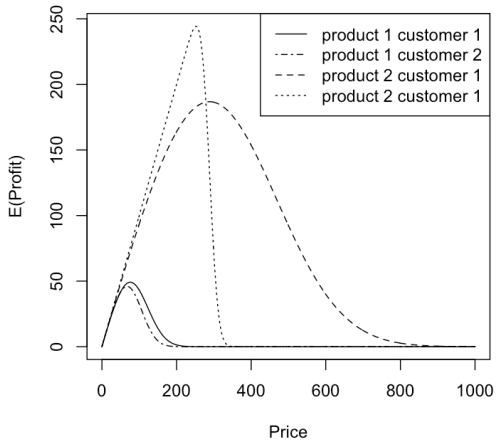


Fig. 11. Example of probabilistic models of two customers and two products.

Case 8 (N kind of products, a negotiable price, and C customers) can be studied using the same concept of expected profit curves, but there will be $N \times C$ curves. For each of the N kind of products, there will be C curves that belong to the buying model of each customer. Figure 11 presents a simple example with two products and two customers. Several settings could be possible: each customer can only buy one product, there is limited stock, etc. Therefore, the curves will change or disappear in real time, depending on the setting of the problem. In

this work we are always offering the best product to the best customer, but there is no problem in offering more than one product to the same customer or to multiple customers[4]. If we only have one product, the first customer in answering will obtain the product. For example in the case of offering the two topmost ranked products to a customer, we would work as follows. First, we would obtain the most desirable product, in the same way as case 7 in table 1 (N kinds of products, a negotiable price, and one customer). Second, the curve of this product would be deleted. And third, the most desirable of the remaining products would be obtained, again in the same way as case 7.

6 Experiments

Experiments have been performed by using real data collected from an estate agent[5]. We have information of 2,800 properties (flats and houses) that were sold during 2009 in the city of Valencia (Spain), for which we have the following attributes ("district", "number of rooms", "square metres", "owner's price" and "selling price"). "Owner's price" is the price which the owner wants to obtain for the property. "Selling price" is the final price at which the purchase took place.

We assume that the "selling price" is some kind of "market price", which is usually closer to the "maximum price". Although this is not always true because in some cases the buyer could have paid more than this for the property, it is generally a good approximation as we discussed in Section 1. In any case, it is almost impossible to obtain the real "maximum price", because it is very difficult that a customer says the maximum price that s/he could pay for a product.

We have randomly split the dataset, using a uniform distribution without replacement, into a training set and a test set. 10% of the data are for training and the rest to test. This tries to simulate a realistic situation when there are not too many data for training. Therefore, the results refer to 2,520 properties, and learning is made from 280 flats. We applied the solutions proposed in Section 4 to the data, namely the idea of populating the dataset to learn a better classification model and the idea of using regression models (inverted problem). In particular we have used the "improved classifier" solution (presented in section 4.2. In particular we learnt a $J48$ (WEKA implementation of $C4.5$ algorithm [26]) decision tree classifier (with Laplace correction and without pruning), implemented in the data mining suite WEKA [31]. It has been learned using example generation (10 positive examples and 10 negative examples for each original example using the negotiable feature rules explained in Section 4.2, so condition 2 holds for all the instances and $\tau = 1$). Since the predicted probability curve given by a classifier (such as the J48 classifier) typically shows discontinuities and strong steps when varying a negotiable feature, we have smoothed the curve with a low-pass filter with Bartlett overlapping window [30]. The parameter of the window has been set to the "minimum price" divided by 4. This value were

[4] Note that we do not need a multinomial model for this. We only need to determine when two offers are close in the ranking.

[5] Data can be found at: "http://tinyurl.com/2usbdfj" in WEKA format.

set after making some experiments (divided by 2, 3, 4 and 5) with some flats and observing that this value smoothed the curves properly. In Figure 12 we can observe the difference between the original curve and the smoothed curve. The "inverting problem presentation" solution (presented in Section 4.3) has been implemented with the *LinearRegression* and $M5P$ [25] regression techniques (both with their default parameters), also from WEKA[6].

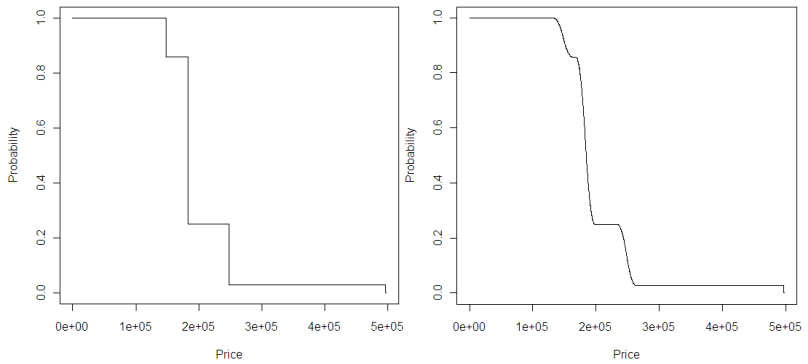


Fig. 12. Left: Example of estimated probabilities. Right: Estimated probabilities which have been smoothed by a low-pass filter with Bartlett overlapping window.

These three learning techniques have been used to guide the three negotiation strategies explained in section 5.1. For the MGO strategy we have used a Montecarlo approach [19]: 3 prices are selected using a uniform random distribution from the range of prices and the value for these 3 points is calculated using the formula in section 5.1, this operation is repeated 1,000 times and the triplet that obtain the maximum value for the formula is chosen.

In Table 3 we can observe the results for case 5 in Table 1 (one kind of product, a negotiable price, and one customer), obtained for each method, in terms of number of sold properties, deal price (in euros) and profit (in euros). In Table 4 we show the results for case 7 in Table 1 (N kinds of products, a negotiable price, and one customer). Concretely, we have set the number of

[6] The source code of the algorithm which computes the probability can be obtained at: "http://tinyurl.com/3xxpxyp". The version of the Java Development Kit used is "jdk1.6.0_14" that can be downloaded at: "http://www.sun.com/java/" and the learning algorithms come from the Weka API (version 3.6.1) "http://www.cs.waikato.ac.nz/ml/weka/". The negotiation strategies have been implemented using version 2.9.2 of R (R-project statistical package) "http://www.r-project.org/" and the script with these algorithms is accessible at: "http://tinyurl.com/33e5up6".

different products to five, i.e., $N = 5$. For these experiments with 1 customer and 5 flats, we chose each group of 5 flats randomly using a uniform distribution and without replacement. This means that we have grouped the 2,520 flats in groups of 5 flats, having 504 groups of 5 flats. Each group of 5 flats is offered to the customer and s/he can only buy one of the flats. As we have explained in section 5.3, the envelope curve of the 5 curves is calculated and the negotiation strategies are applied to it.

In Table 3 and Table 4 we compare the MEP, BLEP and MGO strategies with these baseline methods:

- Baseline method (1 bid or N bids) (1 customer and 1 product). One of the simplest methods to price a product is to add a margin (a percentage) to its minimum price (or base cost). Instead of setting a fix percentage arbitrarily, we obtain the percentage (called α) such that it obtains the best result for the training set. For example, in our experiments, the best α is 0.8, so it is expected that the best profit will be obtained by increasing the minimum price of the properties in an 80%. If we have only 1 bid, we will increase the minimum price of the flat by α. But, if we have N bids, we will have one half of the bids with a value of α less than the calculated α and the other half of the bids with a value of α greater than the calculated α. In particular, the value of α will increase or decrease by $\alpha/(N+1)$ in each bid. For example, in our experiments for 3 bids, the three values of α for three bids would be 100%, 80% and 60%. Therefore, the first offer would be an increase of 100% over the minimum price of the product, the second an increase of 80% and the third an increase of 60%.
- Baseline method (the most expensive/the cheapest) (1 customer and M products). When the seller has several products to offer to the customer, it is not as clear as the previous case how to create a baseline method, because there are several products with different prices. We have proposed two baseline methods: one associated with the most expensive product of the M products, and the other associated with the cheapest product of the M products. In other words, the baseline methods in these cases are the same as the baseline method (1 bid or N bids) (1 customer and 1 product), but in these cases the increased price is the price of the most expensive product or the price of the cheapest product.

In Table 3 we also show the results of two reference methods. The methods "All flats sold at π_{min}" and "All flats sold at π_{max}" show the hypothetic cases when all the flats are sold at the minimum price or at the maximum price. In Table 4 the methods "Selling the most expensive" and "Selling the cheapest" show the hypothetic cases when, in all the groups of 5 flats, the most expensive flat of the 5 have been sold to its maximum price, or the cheapest flat of the 5 have been sold to its maximum price.

In order to analyse the results, let us first focus on Table 3, which shows results for one customer and one flat. For one possible bid, we have that all the methods based on data mining get better results. The solution using an extended dataset but preserving the original task (J48 classifier) is slightly better

1 customer and 1 flat			
Method	**Sold flats**	**Deal price**	**Profit**
Reference			
All flats sold at π_{min}	2,520	534,769,904	0
All flats sold at π_{max}	2,520	712,580,216	177,810,312
1 bid			
Baseline (80%)	290	74,664,508	33,184,226
MEP ($J48$)	1,094	244,102,200	42,034,896
MEP ($LinearRegression$)	1,681	341,147,000	38,372,983
MEP ($M5P$)	1,707	342,805,800	38,580,279
3 bids			
Baseline (100%, 80%, 60%)	635	165,443,695	67,226,421
BLEP ($J48$)	1,211	260,298,700	43,474,928
BLEP ($LinearRegression$)	1,746	352,112,700	39,574,602
BLEP ($M5P$)	1,769	353,106,100	39,698,991
MGO ($J48$)	1,700	422,209,800	84,028,502
MGO ($LinearRegression$)	1,918	477,247,900	95,323,551
MGO ($M5P$)	1,939	487,771,600	98,376,548

Table 3. Results obtained for one product and one customer by the negotiation strategies, baseline methods and reference methods (minimum and maximum profit). Deal price and profit measured in euros.

than the problem inversion methods (Linear Regression and M5P regressor). Taking a look at the number of flats sold, it suggests that MEP with regression somehow underestimates the ideal price in this situation. For three bids, the picture changes. The baseline method is now better than BLEP. This is expected since BLEP just chooses the local optimum each time and disregards the overall picture. On the contrary, the MGO clearly surpasses the other methods, which shows that a global formulation is necessary to solve the case for several bids. If we take a look at the method, regression (and M5P in particular) is the best method for this case. As a wrapping-up of the results for one customer and one flat we can say that for one bid, MEP with a J48 classifier gives the best results, while the MGO with the M5P regressor is the best combination.

Now let us focus on Table 4, which shows results for one customer and 5 flats. For one possible bid, we have that all the methods based on data mining get better results than the "baseline (80%) (the cheapest)" but not for the "baseline (80%) (the most expensive)". Only the solution using an extended dataset but preserving the original task (J48 classifier) is better than both baselines. Again, taking a look at the number of flats sold, it suggests that MEP with regression somehow underestimates the ideal price in this situation. For three bids, the picture changes again. The "baseline method the cheapest" is the worst method while "the baseline method the most expensive" is now better than BLEP using regression, and comparable to BLEP using classification. This is again expected since BLEP just chooses the local optimum each time and disregards the overall picture. On the contrary, MGO clearly surpasses the other methods (with the only exception that MGO with J48 is worse than BLEP with J48). This also shows that a global formulation is necessary to solve the case for several bids. If we take a look at the method, regression (and M5P in particular) is the best method for this case. As a wrapping-up of the results for one customer and 5

1 customer and 5 flats			
Method	**Sold flats**	**Deal price**	**Profit**
Reference			
Selling the most expensive	504	233,943,288	58,092,977
Selling the cheapest	504	85,385,709	22,641,094
1 bid			
Baseline (80%) (the most expensive)	48	20,822,533	9,254,459
Baseline (80%) (the cheapest)	74	11,763,632	5,228,281
MEP ($J48$)	180	74,025,300	13,043,111
MEP ($LinearRegression$)	242	79,143,500	6,506,317
MEP ($M5P$)	226	72,390,900	5,613,714
3 bids			
Baseline (100%, 80%, 60%) (the most expensive)	123	50,497,241	18,936,465
Baseline (100%, 80%, 60%) (the cheapest)	144	22,025,593	8,259,598
BLEP ($J48$)	354	116,478,000	20,468,778
BLEP ($LinearRegression$)	434	123,971,100	11,146,363
BLEP ($M5P$)	431	121,216,400	10,491,057
MGO ($J48$)	288	115,919,700	20,213,901
MGO ($LinearRegression$)	339	142,570,700	24,541,886
MGO ($M5P$)	344	147,656,200	25,109,410

Table 4. Results obtained for one customer and 5 products by the negotiation strategies, baseline methods and reference methods. Deal price and profit measured in euros.

flats we can say that for one bid, MEP with a J48 classifier gives the best results, while MGO with the M5P regressor is the best combination.

Consequently, the results for one customer and one flat are in agreement with one customer and five flats (case 5 in Table 1). Since the problem with C customers and one flat (case 6 in Table 1) is symmetrical wrt. one customer and N flats (case 7 in Table 1), similar results are expected for the case of C customers and N products (case 5 in Table 1), since its solution is similar to the other two cases.

In conclusion, the MEP or BLEP negotiation strategies can obtain good results, but the MGO method is more robust, because it is a global optimisation method.

7 Related Work

This work incorporates ideas from data mining [14] (machine learning and statistics), marketing research [7], negotiation [16], agent theory [27] and, decision theory [21], in order to address a taxonomy of prescription problems with basic negotiation issues in the area of marketing and customer-relationship management. In this section we describe the possible connections between our work and some previous or related approaches. One goal of this section is to clearly see where we differ from other approaches. The difference may be found because we address different (but related) problems or because we use different techniques (and hence a comparison should be made). In both cases, however, we can take some ideas for extending our approach to more complex or general problems or to address the same problem with better techniques. Let us see this.

The first field we come up is decision theory. In prescription problems, we are always concerned about what to sell and to whom. If we have a utility function, and we have a probabilistic model for a set of decisions (choosing the product or choosing the customer), we can derive an expected utility for each set of decisions, and try to maximise this. If there is no interaction at all, we have a list of expected utilities, from which we can construct a rank and, design, for instance, a mailing campaign. If there is a finite sequence of interactions, we can analyse that with a probabilistic decision tree or with transition graphs. If this interaction may get larger or infinite, then we may require a Markov Decision Process (MDP) [24], assuming that the Markov property holds. In this case, many techniques from dynamic programming [5] and reinforcement learning [28] can be used. Although our MGO method may resemble some of the methods in reinforcement learning, we have used a Montecarlo approach, because we have an infinite multidimensional set of inputs, and we want the probabilistic model to be treated as a black box (it is a data mining model which can be linear, non-linear, non-continuous). We could also study the application of linear programming techniques for this problem.

A prescription problem with negotiable attributes not only depends on what to sell and to whom. It also deals with features, such as prices. And these features are typically continuous, so we cannot address this with discrete approaches such as MDP, which are discrete-time and discrete-action decision processes, unless we discretise these attributes in a reduced set of bins and we augment the MDP framework to consider an order relation between the actions (as the order relation we have defined for our inputs, and outputs). Although discrete-time is not a problem here, we need to look to continuous-action decision processes, as in control theory [17]. However, we fail to identify a good representation of the problem here in such a way that we do not recognise the conditions which are typically required in control problems. Additionally, we have few feedback interactions from the customers (and very few if we only consider one customer at a time), we do not have many continuous outputs from the system (just a purchase or not, not even a degree of satisfaction), so the idea of gradually adjusting and gauging which is typical in process control does not hold here either.

The closest approaches are from the broader areas of game theory [12], negotiation policies [16] and multi-agent optimisation [32]. However, much of the results and techniques have been specialised to very specific cases (see, e.g. [29]), while there are only a few general principles from game theory, such as the Nash equilibrium if the agent competes (as in our case). However, we are not concerned about the global optimum, but the optimum for one agent (typically the seller), and there are many games held at the same time sharing some issues (products and customers).

Back to the field of CRM and marketing, it seems that cross-selling, up-selling and recommender systems may be useful here, as we have already mentioned in the introduction. There are some approaches that employ data mining techniques to configure cross-selling campaigns. For instance [18] applies multinomial logit

on a CRM cross-sell application. Another example is [23]. This work presents a structural multivariate probit model to investigate how customer demand for multiple products evolves over time. However, it is not straightforward to apply this kind of data mining techniques in our framework. The issue is the family of problems we are addressing; we are not thinking about a long-term customer relationship. The examples we are using are typically related to selling houses, cars, or other products which are not repeatedly purchased by the same customer. In fact, it is more likely (custom and acceptable) to negotiate and use different prices on these products for each customer than to negotiate or use different prices for a bar of chocolate. At least at this moment of the taxonomy and the kinds of problems we consider, there is no follow-up about the customer, no real concern about customer's churn, etc. The main goal is to maximise profits for a buyer that will not been seen again after a purchase (unless a complaint or a refund).

There is an interesting relation between the notion of negotiable feature and Choice Based Conjoint (CBC) Analysis ([13]). The main objective of Conjoint Analysis is to measure an individual's or a population's preference on a set of parameters and levels. CBC Analysis is a special family of Conjoint Analysis techniques that have to choose between a set of options for several parameters. In the context of market research, subjects are customers and parameters are product's features. The output of the process is an ordinal scale which ranks all the options or, more frequently, a scale in which every option is given a value. In other words, Conjoint Analysis allows for a proper ranking of the features. Conjoint analysis presents the problem that one option cannot compensate the bad value of other options (e.g. we will not buy a flat at any price if it does not have a lift). This is related to the parameter τ in our definition of negotiable feature. An option (or feature) can be made irrelevant given the values of other features. Adaptive Choice Based Conjoint (ACBC) Analysis is an extension of Choice Based Analysis which allows for non-compensatory decision rules as in the previous example. It is frequent to see a first ACBC analysis to define which features are relevant, and second, to apply a CBC to rank them. This would be appropriate in cases where we do not exactly know the attributes which are negotiable. However, we have to clarify that CBC is based on examples which are preference expressions (e.g. 'I prefer flat A with feature values a_1, a_2, ..., a_n, over flat B with feature values b_1, b_2, ..., b_n'). In our case, our examples are like 'Customer bought (or didn't) flat A with feature values a_1, a_2, ..., a_n'. Even in cases where the data does not come from choices (general Conjoint Analysis) there is typically a pre-designed research survey, with preference questionnaires that may take several minutes. This scenario is far from the assumptions we are making here about a historical datasets with actual transactions, instead of a survey to gather information. In our case, we would prefer to adapt a classical feature selection method instead. Nonetheless, in cases where the survey can be performed, CBC analysis can be good tool to determine the negotiable attributes, especially in cases where we want to optimise the offer for more than one negotiable feature at a time, because the ranking of feature relevance as

well as their range of acceptable values can help in the combinatorial problem of finding the best set of values for the first and subsequent offers.

8 Conclusions

In this paper, we have investigated a new family of prescription problems using data mining models, where one or more features are negotiable. These problems have motivated the extension of the taxonomy of prescription problems, and the development of a series of techniques to solve the optimisation problem of maximising the result that can be obtained with the models. A key notion has been the handling of the inversion problem which appears when transforming an input (negotiable) feature into an output feature, which can turn a classification problem into a regression problem and viceversa. The probabilistic estimation for the new output feature has been solved for both cases (probabilistic classification and regression), so producing probability estimation curves and profit curves. Using these curves we have been able to devise several negotiation strategies, which have been proven to behave better as long as a more global view is taken, which usually implies more complex maximisation problems which, due to characteristics of the data mining model, have been addressed with a Montecarlo approach.

The scenario we have used as a guiding thread for the paper shows a realistic problem in the CRM field, where data mining can help a seller to make a good decision about which product should be offered to which customer and at what price, in order to obtain as much overall profit as possible. However, the techniques presented here are applicable to many other prescription problems, inside the CRM field or outside (e.g. medicine, education, law, ...), or many other situations where some kind of negotiation takes place using data mining models.

This work has therefore been focused on the model deployment stage, which is becoming a scientifical problem itself, with much more entity and shape than some years ago, when data integration and processing, data modelling and evaluation were the data mining stages where the main computational effort and techniques were developed. Data deployment in a context of global maximisation requires the hybridisation of techniques from several fields, such as linear programming, simulation, numerical computation, etc. This also implies that data mining models have to be constructed and evaluated taking care of their application context and their relation with several other models, rules, constraints and goals.

As future work, many ideas follow from this work. For instance, we plan to develop new evaluation metrics which consider the quality of a predictive data mining model not only as the accuracy of its estimations given the inputs, but also its quality when the model has to be inverted. For instance, in our experiments we have found that regression trees are better than linear regression, and, in some cases, better than the direct classification decision trees approach. This suggests the development of evaluation metrics which can be used to select the best models before application. Another issue for future analysis is the use of

more efficient techniques to compute the curves and the envelopes when we have a high number of items and customers, since the combinations are quadratic.

In this work, we only have one negotiable feature at a time, but we are studying the extension to multiple negotiable features. When we have only one negotiable feature we have two dimensions (the negotiable feature and the probability). In the case of multiple negotiable features we have one dimension for each negotiable feature plus one (the probability). For example, if we have two negotiable features, we have three dimensions, and instead of having curves, in this case, we have surfaces. MEP and MGO strategies can be applied to multiple negotiable features without any problem, but the BLEP strategy needs changes, because each negotiable feature is monotonic, but all the negotiable features could not been monotonic at the same time, and this is a problem for the normalisation phase of the BLEP algorithm. Finally, other areas of possible research are the enrichment of the scenario with counteroffers from the customer, perishable and/or limited stocks, as well as the application to other areas outside CRM, such as medical research.

References

1. G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. on Knowl. and Data Eng.*, 17(6):734–749, 2005.
2. A. Bella, C. Ferri, J. Hernández-Orallo, and M.J. Ramírez-Quintana. Joint cut-off probabilistic estimation using simulation: A mailing campaign application. In *IDEAL*, volume 4881 of *LNCS*, pages 609–619. Springer, 2007.
3. M.J.A. Berry and G.S. Linoff. *Mastering Data Mining: The Art and Science of Customer Relationship Management.* Wiley, 1999.
4. A. Berson, S. Smith, and K. Thearling. *Building Data Mining Applications for CRM.* McGraw Hill, 2000.
5. D.P. Bertsekas. Dynamic programming and optimal control, 3rd Edition. 2005.
6. M. Better, F. Glover, and M. Laguna. Advances in analytics: integrating dynamic data mining with simulation optimization. *IBM J. Res. Dev.*, 51(3):477–487, 2007.
7. N. Bradley. *Marketing Research. Tools and Techniques.* Oxford University Press, 2007.
8. J. Carbo and A. Ledezma. A machine learning based evaluation of a negotiation between agents involving fuzzy counter-offers. In *AWIC*, pages 268–277, 2003.
9. L. Devroye, L. Györfi, and G. Lugosi. *A Probabilistic Theory of Pattern Recognition (Stochastic Modelling and Applied Probability)*. Springer, 1997.
10. C. Ferri, P.A. Flach, and J. Hernández-Orallo. Improving the AUC of probabilistic estimation trees. In *14th European Conference on Machine Learning, Proceedings*, pages 121–132. Springer, 2003.
11. C. Ferri, J. Hernández-Orallo, and R. Modroiu. An experimental comparison of performance measures for classification. *Pattern Recogn. Lett.*, 30(1):27–38, 2009.
12. D. Fudenberg and J. Tirole. *Game theory.* MIT Press, 1991.
13. A. Gustafsson, A. Herrmann, and F. Huber. Conjoint analysis as an instrument of market research practice. *Conjoint measurement*, pages 3–30, 2000.
14. J. Han and M. Kamber. *Data Mining: Concepts and Techniques.* Morgan Kaufmann Publishers, 2006.

15. J. J. Heckman. *Sample Selection Bias as a Specification Error*. Econometrica, 1979.
16. N.R. Jennings, P. Faratin, A.R. Lomuscio, S. Parsons, M.J. Wooldridge, and C. Sierra. Automated negotiation: Prospects, methods and challenges. *Group Decision and Negotiation*, 10(2):199–215, 2001.
17. C. Kilian. *Modern Control Technology*. Thompson Delmar Learning, 2005.
18. S. Li, B. Sun, and R.T. Wilcox. Cross-selling sequentially ordered products: An application to consumer banking services. *Journal of Marketing Research*, 2005.
19. N. Metropolis and S. Ulam. The monte carlo method. *Journal of the American Statistical Association (American Statistical Association)*, 1949.
20. B. Padmanabhan and A. Tuzhilin. On the use of optimization for data mining: Theoretical interactions and ecrm opportunities. *Management Science*, 49(10, Special Issue on E-Business and Management Science):1327–1343, 2003.
21. M. Peterson. *An Introduction to Decision Theory*. Cambridge University Press, 2009.
22. A. Prinzie and D. Van den Poe. Constrained optimization of data-mining problems to improve model performance: A direct-marketing application. *Expert Systems with Applications*, 29(3):630–640, 2005.
23. A. Prinzie and D. Van den Poel. Exploiting randomness for feature selection in multinomial logit: A crm cross-sell application. In *Advances in Data Mining*, volume 4065 of *Lecture Notes in Computer Science*, pages 310–323. Springer Berlin / Heidelberg, 2006.
24. M. L. Puterman. *Markov Decision Processes*. Wiley, 1994.
25. J. R. Quinlan. Learning with continuous classes. In *5th Australian Joint Conference on Artificial Intelligence*, pages 343–348. World Scientific, 1992.
26. J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1993.
27. S.J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, 2003.
28. R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. The MIT press, 1998.
29. I. Vetsikas and N. Jennings. Bidding strategies for realistic multi-unit sealed-bid auctions. *Autonomous Agents and Multi-Agent Systems*, 21:265–291, 2010. 10.1007/s10458-009-9109-6.
30. E.W. Weisstein. *CRC concise encyclopedia of mathematics*. CRC Press, 2003.
31. I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Elsevier, 2005.
32. M. Wooldridge. *An Introduction to MultiAgent Systems*. John Wiley & Sons Ltd, 2002.
33. S. Zhang, S. Ye, F. Makedon, and J. Ford. A hybrid negotiation strategy mechanism in an automated negotiation system. In *ACM Conference on Electronic Commerce*, pages 256–257. ACM, 2004.

7.8 On the Effect of Calibration in Classifier Combination

8. A. Bella, C. Ferri, J. Hernández-Orallo, and M.J. Ramírez-Quintana. On the Effect of Calibration in Classifier Combination. (Submitted to Applied Intelligence).

On the Effect of Calibration in Classifier Combination

A. Bella, C. Ferri, J. Hernández-Orallo, and M.J. Ramírez-Quintana

Universitat Politècnica de València, DSIC, València, Spain

Abstract. A general approach to classifier combination considers each model as a probabilistic classifier which outputs a class membership posterior probability. In this general scenario, it is not only the quality and diversity of the models which are relevant, but the level of calibration of their estimated probabilities as well. In this paper, we study the role of calibration before and after classifier combination, focusing on evaluation measures such as MSE and AUC, which better account for good probability estimation. We present a series of findings that allow us to recommend several layouts for the use of calibration in classifier combination. We also analyse a new non-monotonic calibration method that works better for classifier combination than other monotonic calibration methods.

1 Introduction

The problem of combining multiple decisions from a set of classifiers is known as classifier combination or classifier fusion[26]. The need for classifier combination in many real applications is well-known. On the one hand, more and more applications require the integration of models and experts that come from different sources (human expert models, data mining or machine learning models, etc.). On the other hand, it has been shown that an *appropriate* combination of several models can give better results than any of the single models alone [26][34], especially if the base classifiers are diverse [28].

The term 'ensembles' [13][26] is used especially when the set of base classifiers are created on purpose and are homogeneous. Some ensemble techniques show an excellent performance, such as boosting [19], bagging [8], and randomisation [14]. If the set of classifiers is created on purpose but not homogeneous, the term 'ensemble' is not so frequently used. Some of these techniques are stacking [36], cascading [20] and delegating [17]. However, in many situations, there is no on-purpose generation of a set of classifiers and we have to combine opinions from many sources (either human or machines) into a single decision. In these cases, we have no control over the set of classifiers that we have to work with and heterogeneity is very high. In this paper, we cover these three types (or degrees) of classifier combination. Since the latter case is more challenging and general, no assumption will be made about the way in which the set of classifiers have been obtained. Consequently, we will work with classifier combination from an

uncontrolled and heterogeneous set of models, where some (or all) of them may be machine learning models, or models that have been constructed by human experts. The only (mild) assumption is that we expect all these classifiers to be able to output a posterior probability, a score, or a reliability value for their prediction.

Hence, given a set of (heterogeneous) classifiers, it is very important to assess and use their individual quality for a proper classifier combination. Typically, an overall weighting is used in such a way that more reliable classifiers are given more weight than other less reliable classifiers. The way in which 'reliability' is measured and the way in which it is used in the combination make up different weighted combination schemes (see, e.g., [26]). In the same way, diversity has typically been analysed in terms of qualitative measures, over a given dataset, such as disagreement, Q-statistic, Kappa statistic, etc. [28][27].

When classifiers are considered as rankers or probability estimators there are many more options to work with. If we understand probabilities as an indicator of reliability, we can have a "double-weighting" combination scheme, where overall classifier accuracy is used as a "first weight" while estimated class probabilities are used as a "second weight". For instance, given two classifiers a_l and a_m with similar accuracies (or any other performance measure) where $a_l < a_m$, we can still have that, for a particular item i, classifier l may be very sure (with extreme probabilities) while classifier m may be more uncertain (with medium probabilities). Of course, for other items, this might be the other way round. If probabilities are well estimated, this "second weight" will give different levels of credit to each classifier depending on the example at hand, and might be more precise than the overall weight given to each classifier.

Therefore, the key point in this combination scheme is the quality of probability estimations. When the set of classifiers is heterogeneous and originates from different sources, we cannot assume that these estimations are evenly accurate and reliable. In fact some classifiers may output more extreme probabilities (closer to 0 or to 1) than others, meaning that they will have more (second) weight in the combination. This is the typical issue of integrating opinions from several experts, when some experts express less or more confidence than they really have. We say these experts are *uncalibrated*. It is well-known that a bad classifier can have a negative influence in the combination. But it is not so frequently realised that just a single bad classifier with extremely confident probabilities can be disastrous for the combination.

It is then of the outmost importance to analyse the effect of calibration in classifier combination by studying several combination layouts and several calibration methods, and their impact on the quality of the combined classifier, according to several evaluation metrics. This is the goal of the paper.

The paper is structured as follows. In Section 2, we give further motivation for this study and we also point out to related (but partial) analyses on this issue in the literature. Section 3 summarises the most common evaluation measures and calibration methods. Section 4 includes a conceptual study on calibration and classifier combination, using several examples and identifying several im-

portant factors. Section 5 presents the experimental evaluation of the previous analysis. In Section 6, we show that monotonic calibration techniques are non-monotonic when applied to more than two classes. This justifies the application and analysis of a new non-monotonic calibration technique known as Multivariate Similarity-Binning Averaging. The whole picture is studied in Section 7 with several combination layouts. Finally, Section 8 gives an overall view of the messages that can be conveyed from this paper, leading to the conclusions and future work.

2 Context and Objectives

Classifier combination has been extensively analysed in the last few decades, establishing very important results about the number of classifiers, their diversity, the combination method, etc. In this paper, we focus on one factor that has not been properly addressed to date: the role of probability calibration in classifier combination.

This role has many aspects: different probability calibration measures, different calibration methods, different layouts where calibration has to be arranged, etc. In addition, we must consider the relation with other fundamental issues in classifier combination, such as classification quality and diversity (which can be evaluated by different families of evaluation measures). A few works have tangentially addressed some of these issues.

There are, for instance, some approaches which use combination and calibration, but generally with a very specific layout. For example, in [31], the *Expectation-Maximization algorithm* was used to modify the weights of a Bayesian Model Averaging method and to obtain a calibrated ensemble, but the effect of calibration methods before the combination was not studied.

Caruana et al [10] also used calibration and combination together. The experimental setting in [10] analysed many other factors altogether but only, included one calibration method (Platt), it was restricted to binary datasets, and the double weighting effect was not evaluated (a uniform weighting was used for Bayesian averaging).

In Bennett's Ph.D. thesis [5], a methodology is introduced to build a meta-classifier for classifying text documents by combining estimated probabilities of base classifiers and using reliability indicators. These reliability indicators are variables with additional information, not mere probabilities, and are application specific.

Brümmer's Ph.D. thesis [9] focuses on speaker and language recognisers. Instead of calibrating the probabilities estimated by a classifier, he calibrates the *log-likelihood-ratio* scores. However, the calibration and combination methods studied are always affine (i.e., linear).

The combination of classifiers and their calibration has also been indirectly addressed when trying to adapt binary calibration methods to multiclass calibration methods, since multiclass calibration can be tackled by a combination of binary calibration methods. For instance, in Gebel's Ph.D. thesis [22], there

is a study of several univariate calibration methods and their extensions to multiclass problems, but only individually, i.e., without combining them. Moreover, Gebel introduced the "Dirichlet calibration" as a multivariate calibrator that is applicable to multiclass problems directly, but its poor overall results make it only recommendable for datasets that have a balanced or slightly imbalanced class distribution.

In the end, this paper analyses the overall issue, bringing calibration to the stage of classifier combination as another key dimension of study, jointly with the well-known properties of classification accuracy and diversity. A general analysis of classifier function and calibration cannot be found in the literature.

Therefore, the objective of this paper is to undertake this analysis. Along the way, this work introduces different contributions that can be summarised as follows:

- A conceptual analysis on how calibration affects the combination of classifiers. This analysis is performed in terms of how classifier probability distributions relate to the combination results depending on the separation of the class distribution, the calibration and diversity of the base classifiers.
- An extensive experimental comparison on the effect of calibration on the combination of classifiers, using many different layouts (calibration before, after, and before and after combination), many different weighting schemes, several calibration methods, and several performance metrics.
- The analysis of a new calibration technique: Multivariate Similarity-Binning Averaging (SBA), recently introduced in [4], which is designed to be non-monotonic while still preserving independence in such a way that its results for classifier combination excel over those of other calibration methods. This will be shown in a complete battery of experiments.
- A summary of findings and results from the previous items, and some recommendations on how to use calibration in classifier combination.

The overall contribution of this work is to provide a better understanding of the role and possibilities of calibration for classifier combination, as well as the way all this should be arranged in order to obtain appropriate results.

3 Classifier Calibration and Evaluation

Given a dataset T, n denotes the number of examples, and c the number of classes. The target function $f(i,j) \to \{0,1\}$ represents whether example i actually belongs to class j. Also, $n_j = \sum_{i=1}^{n} f(i,j)$ denotes the number of examples of class j and $p(j) = n_j/n$ denotes the prior probability of class j. Given a classifier l, $p_l(i,j)$ represents the estimated probability of example i to belong to class j taking values in [0,1].

Calibration is defined as the degree of approximation of the predicted probabilities to the actual probabilities. If we predict that we are 99% sure, we should expect to be right 99% of the time. More precisely, a classifier is perfectly calibrated if, for a sample or bin of examples with predicted probability p for the

positive class, the expected proportion of positives is close to p. Formally, for any $B_r \subseteq T$ such that $p_l(i,j) = r$ for all $i \in B_r$ then $\frac{\sum_{i \in B_r} f(i,j)}{|B_r|} = r$. Note that this definition only says when a classifier is perfectly calibrated but does not give a range of values between perfect and worst calibration. We will see in section 3.1 that calibration measures usually relax the condition for bin formation in order to give a gradual measure.

Given a calibration method which modifies the probabilities, we denote the (supposedly better calibrated) probability of example i to belong to class j by $p_l^*(i,j)$. Note that accuracy and calibration, although dependent, are very different things. For instance, a binary classifier that always assigns a 0.5 probability to its predictions is perfectly calibrated for a balanced dataset, but its expected accuracy is a poor 0.5. However, a very good binary classifier can be uncalibrated if correct positive (respectively negative) predictions are accompanied by relatively low (respectively high) probabilities, e.g., a classifier which is almost always correct but its probabilities range between 0.45 and 0.55.

When the number of classes is 2 we use the special symbols $\oplus$ and $\ominus$ to represent the positive class ($j = 1$) and the negative one ($j = 2$), respectively. Also, in the binary case, we will only refer to the positive class, and we will denote the target function, the score, the estimated probability, and the calibrated probability of an example i as $f_l(i,\oplus)$, $s_l(i,\oplus)$, $p_l(i,\oplus)$ and $p_l^*(i,\oplus)$ or simply $f_l(i)$, $s_l(i)$, $p_l(i)$ and $p_l^*(i)$. For the sake of readability, we will omit the subindex l when we refer to a single classifier.

A set of L classifiers will be denoted by $l_1, l_2, \ldots, l_L$. In order to simplify the notation, sometimes we will use the indices $k \in 1..L$ to denote the classifier l_k. Finally, $\tilde{p}(i,j)$ (respectively, $\tilde{s}(i,j)$) denotes the estimated probability (respectively the score) of example i to belong to class j given by a combination method over the L classifiers.

3.1 Evaluation Measures

Classifiers can be evaluated according to several performance metrics. These can be classified into three groups [15]: measures that account for a qualitative notion of error (such as accuracy or the mean F-measure/F-score), metrics based on how well the model ranks the examples (such as the Area Under the ROC Curve (AUC)) and, finally, measures based on a probabilistic understanding of error (such as mean absolute error, mean squared error (Brier score), LogLoss and some calibration measures).

Accuracy is the best-known evaluation metric for classification and is defined as the percentage of correct predictions. However, accuracy is very sensitive to class imbalance. In addition, when the classifier is not crisp, accuracy depends on the choice of a threshold. Hence, a good classifier with good probability estimations can have low accuracy results if the threshold that separates the classes is not chosen properly.

Of the family of measures that evaluate ranking quality, the most representative one is the *Area Under the ROC Curve (AUC)*, the probability that given one positive example and one negative example at random, the classifier ranks the

positive example above the negative one (the Mann-Whitney-Wilcoxon statistic [18]). AUC is clearly a measure of separability since the lower the number of misranked pairs, the better separated the classes are. Although ROC analysis is difficult to extend to more than two classes ([16]), the AUC has been extended to multiclass problems effectively by approximations. In this paper, we will use Hand & Till's extension [23], which is based on an aggregation of each class against each other, by using a uniform class distribution.

Of the last family of measures, *Mean Squared Error (MSE)* or *Brier Score* penalises strong deviations from the true probability:

$$MSE = \frac{\sum_{j=1}^{c}\sum_{i=1}^{n}(f(i,j)-p(i,j))^2}{n\cdot c} \tag{1}$$

Although MSE was not a calibration measure originally, it was decomposed by Murphy [29] in terms of calibration loss and refinement loss. For that, the dataset T is segmented into k bins (i.e., subsets of equal size), with B_t being the elements of bin t.

$$MSE = \frac{\sum_{j=1}^{c}\sum_{t=1}^{k}\sum_{i\in B_t}|B_t|\cdot(p(i,j)-\bar{f}_t(i,j))^2}{n\cdot c} \tag{2}$$

$$MSE = \frac{\sum_{j=1}^{c}\sum_{t=1}^{k}\sum_{i\in B_t}|B_t|\cdot(p(i,j)-\bar{f}_t(i,j))^2}{n\cdot c} \tag{3}$$

$$-\frac{\sum_{j=1}^{c}\sum_{t=1}^{k}|B_t|\cdot(\bar{f}_t(i,j)-\bar{f}(j))+\bar{f}(j)\cdot(1-\bar{f}(j))}{n\cdot c}$$

where $\bar{f}_t(i,j)=\sum_{i\in B_t}\frac{f(i,j)}{|B_t|}$ and $\bar{f}(j)=\sum_{i=1}^{n}\frac{f(i,j)}{n}$. The first term measures the calibration (denoted by $MSEcal$) of the classifier while the rest of the expression measures other components that are grouped under the term "refinement" (denoted by $MSEref$). The problem of measuring calibration in that way is that the test set must be split into several segments or bins. If too few bins are defined, the real probabilities are not properly detailed to give an accurate evaluation. If too many bins are defined, the real probabilities are not properly estimated. A partial solution is to make the bins overlap.

A calibration measure based on overlapping binning is *CalBin* [11]. For each class, all cases must be ordered by predicted probability $p(i,j)$, giving new indices i^*. The 100 first elements (i^* from 1 to 100) are taken as the first bin. Next, the percentage of cases of class j in this bin is calculated as the actual probability, $\hat{f}_j$. The error for this bin is $\sum_{i^*\in 1..100}|p(i^*,j)-\hat{f}_j|$. The second bin with elements

(2 to 101) is used to compute the error in the same way. At the end, the errors are averaged. Formally:

$$CalBin(j) = \frac{1}{n-s}\sum_{b=1}^{n-s}\sum_{i^*=b}^{b+s-1}\left| p(i^*,j) - \frac{\sum_{i^*=b}^{b+s-1} f(i^*,j)}{s}\right| \quad (4)$$

Instead of 100 for the size of the bin (as [11] suggests) we set a different bin length, $s = n/10$, to make it more size-independent.

3.2 Calibration Methods

In this paper, we will work with the most commonly used calibration methods: binning averaging, Platt's method and PAV calibration. There are other methods based on assignment values [21], Bayesian approaches using asymmetric distributions [6][5], and other more elaborate approaches, such as Dirichlet calibration [22], but their performance, in general is worse than that of the three methods above. For more details, we refer the reader to [3] where a survey of calibration methods can be found.

Binning averaging was proposed by [37] as a method where a (validation) dataset is split into bins in order to calculate a probability for each bin. Specifically, this method consists in sorting the examples in decreasing order by their estimated probabilities (or scores) and dividing the set into k bins. Thus, each test example i is placed into a bin t, $1 \leq t \leq k$, according to its probability estimation. Then the corrected probability estimate for i (p_i^*) is obtained as the proportion of instances in t of the positive class.

Platt [30] presented a parametric approach for fitting a sigmoid function that maps SVM predictions to calibrated probabilities. The idea is to determine the parameters A and B of the sigmoid function $p_i^* = \frac{1}{1+e^{A \cdot p_i + B}}$ that minimise the negative log-likelihood of the data, that is: $argmin_{A,B}\{-\sum_i f_i log(p_i^*) + (1 - f_i) log(1 - p_i^*)\}$. This two-parameter minimisation problem can be performed by using an optimisation algorithm, such as *gradient descent*. Platt proposed using either cross-validation or a hold-out set for deriving an unbiased sigmoid training set for estimating A and B.

In the *Isotonic Regression* [33] method, the calibrated predictions are obtained by applying a mapping transformation that is isotonic (monotonically increasing), known as the pair-adjacent violators algorithm (PAV) [2]. The first step in this algorithm is to order the n elements decreasingly according to estimated probability and to initialise $p^*(i) = f(i)$. The idea is that calibrated probability estimates must be a decreasing sequence, i.e., $p^*(1) \geq p^*(2) \geq \ldots \geq p^*(n)$. If this is not the case, for each pair of consecutive probabilities, $p^*(i)$ and $p^*(i+1)$, such that $p^*(i) < p^*(i+1)$, the PAV algorithm replaces both of them by their probability average, that is, $a \leftarrow \frac{p^*(i)+p^*(i+1)}{2}$, $p^*(i) \leftarrow a$, $p^*(i+1) \leftarrow a$. This process is repeated (using the new values) until an isotonic set is reached. Table 1 shows an example of this calibration method applied to two classifiers.

3.3 Monotonicity and Multiclass Extensions

The three calibration methods described above are monotonic; they do not change the rank (order) of the examples according to each class estimated probability. In fact, Platt's method is the only one that is strictly monotonic, i.e, if $p(i_1) > p(i_2)$, then $p^*(i_1) > p^*(i_2)$, implying that AUC and refinement are not affected. In the other two methods, ties are generally created (i.e, $p^*(i_1) = p^*(i_2)$ for some examples i_1 and i_2 where $p(i_1) > p(i_2)$) as shown in Table 1. This means that refinement is typically reduced for the binning averaging and the PAV methods.

Monotonicity will play a crucial role in understanding what calibration does before classifier combination. However, as we will see in Section 6, the extension of calibration methods to multiclass does not preserve monotonicity. In addition, apart from overfitting, there is no reason to impose monotonicity over a caliberion method, which, in the most general case, is a transformation over the scores that leads to good probability estimation. This will motivate the analysis of a recently introduced non-monotonic calibration method called SBA.

Based on the concept of monotonicity, we propose a taxonomy of calibration methods (Figure 1) including classical calibration methods (PAV, Binning and Platt), the SBA method and Brümmer's "affine fusion and calibration" methods [9]. We are interested in calibration methods that lead to better local weights. Consequently, we will not use Brümber's method in this paper.

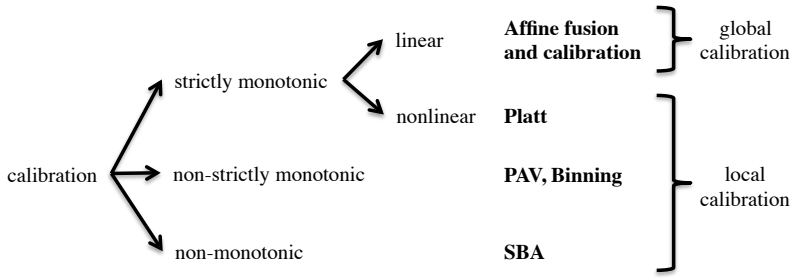


Fig. 1. Taxonomy of calibration methods in terms of monotonicity (strictly monotonic, non-strictly monotonic, or non-monotonic methods), linearity (linear or nonlinear methods). We have also indicated the methods that can be used for global and local calibration.

Another important issue is whether the calibration methods are binary or multiclass. The three methods presented in Section 3.2 were specifically designed for two-class problems. For the multiclass case, Zadrozny and Elkan [37] proposed an approach that consists in reducing the multiclass problem into a number of

binary problems. A classifier is learnt for each binary problem and, then, its predictions are calibrated.

Some works have compared the *one-against-all* and the *all-against-all* schemes, concluding in [32] that the *one-against-all* scheme performs as well as the *all-against-all* schemes. Therefore, in this paper, we will use the *one-against-all* approach for our experimental work because its implementation is simpler.

4 The Relation between Calibration and Combination

In this section, we analyse the relation between model calibration and the performance of the classifier combination. For simplicity, we restrict our conceptual analysis to binary cases. Similar relations are expected to be found if we consider the probability distribution of each class over another (so having $c \times (c-1)$ distributions). In any case, the experimental analysis in Section 5 is performed on multiclass datasets as well.

4.1 Weighted Average Combination

One of the most common methods of classifier combination is *Bayesian Model Averaging* [24]. It consists in weighting each classifier, giving more credit to more reliable sources. However, this rationale does not necessarily entail the best combination ([25][26]). An alternative (and generalised) option is the weighted average combination [26], using probabilities:

Definition 1. Weighted Average Combination. *The estimated probability of an item i belonging to class j given by a weighted average combination of L classifiers is*

$$\tilde{p}(i,j) = \sum_{k=1}^{L} w_k p_k(i,j) \tag{5}$$

We assume $\sum_{k=1}^{L} w_k = 1$. Formula (5) defines a fusion scheme that is a linear combination of the classifier outputs and can be instantiated to more specific schemas depending on how w_k and p_k are chosen. In general, the use of a performance (or overall reliability) weight per classifier w_k is justified because some classifiers are more reliable than others. However, a proper calibration would give each prediction its proper weight depending on the reliability of $p_k(i,j)$ (high reliability for $p_k(i,j)$ closer to 0 and 1, and lower reliability for $p_k(i,j)$ closer to 0.5 or to the class proportion for imbalanced problems). This use of w_k and p_k at the same time is what we refer to as "double weighting".

Example 1. Two probabilistic classifiers l_1 and l_2 are evaluated over a dataset with 10 examples as shown in Table 1, and combined using weights $w_1 = 0.75$ and $w_2 = 0.25$. The top three rows show their individual predictions and their combination. The mid three rows show the results with two new classifiers l_1^* and

l_2^*, which have been obtained from l_1 and l_2 by using a strictly monotonic calibration method over another dataset (their calibration is better but not perfect). The bottom three rows show the results with two new classifiers l_1^{pav} and l_2^{pav}, which have been obtained from l_1 and l_2 by using PAV (a non-strictly monotonic calibration method) over the same dataset (so their calibration is perfect). All the accuracies are calculated with a threshold of 0.5, and when the probability is exactly 0.5 the example is considered half a correct classification.

Example 1 shows the result of using the weighted average combination of two classifiers with weights $w_1 = 0.75$ and $w_2 = 0.25$. In the combination, the probabilities are also used, leading to different results depending on the degree of calibration of these two classifiers. In this example, we see that weights can counter-effect probabilities (and vice versa), as we see in example e_6 (which is finally a hit for $\tilde{p}$) and examples e_2 and e_8 (which are finally an error for $\tilde{p}$). So, using both weights and good probabilities entails a "double-weighting", which in some cases might be beneficial but in other cases might not. Looking at the extreme cases, with very bad probabilities, the weight w_k should be used alone (as in weighted majority voting) and, with perfect probabilities, the weights should not be used.

Table 1. Variations corresponding to Example 1. Threshold is set on 0.5 to calculate the accuracy (last column).

	Examples										Acc.
	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9	e_{10}	
True class	-	+	+	+	-	-	+	-	+	-	
p_1	0.40	0.60	0.80	0.40	0.60	0.52	0.60	0.45	0.52	0.40	7 / 10
p_2	0.90	0.10	0.20	0.20	0.20	0.00	0.20	1.00	0.20	0.10	3 / 10
$\tilde{p}$	0.525	0.475	0.650	0.350	0.500	0.390	0.538	0.588	0.440	0.325	3.5 / 10
p_1^*	0.20	0.80	0.90	0.20	0.80	0.60	0.80	0.30	0.60	0.20	7 / 10
p_2^*	0.60	0.40	0.45	0.45	0.45	0.10	0.45	0.90	0.45	0.40	3 / 10
$\tilde{p}^*$	0.300	0.700	0.788	0.263	0.713	0.475	0.713	0.450	0.563	0.250	8 / 10
p_1^{pav}	0.250	0.667	1.000	0.250	0.667	0.500	0.667	0.250	0.500	0.250	7 / 10
p_2^{pav}	0.571	0.500	0.571	0.571	0.571	0.000	0.571	0.571	0.571	0.500	6 / 10
$\tilde{p}^{pav}$	0.330	0.625	0.893	0.330	0.643	0.375	0.643	0.330	0.330	0.518	7 / 10

In order to better understand the relation between weights and probabilities, we firstly need to understand the meaning of the weights. There are many ways of calculating weights. A very common option is to estimate the accuracy on a validation dataset D, followed by a normalisation [26], i.e., if acc_k is the accuracy of model l_k on D, then $w_k = \frac{acc_k}{\sum_{m=1}^{L} acc_m}$. If we use AUC (or MSE) as a measure, the question of whether a double weighting is going to be too drastic depends on how the weights are derived from these measures. For instance, a weight equal

to the AUC is an option, but since AUC=0.5 means random behaviour, perhaps the GINI index (which equals $(AUC - 0.5) \times 2$) would be a better option. In the same way, using the MSE, $(1 - MSE)$ is a natural option, but a more extreme $1/MSE$ could also be considered. Table 2 shows the definition for the five weights we are going to use.

Table 2. Different methods to calculate weights.

Method	Definition
WCUnif	$w_k = \frac{1}{L}$
WCAcc	$w_k = \frac{acc_k}{\sum_{m=1}^{L} acc_m}$
WCAUC	$w_k = \frac{AUC_k}{\sum_{m=1}^{L} AUC_m}$
WCMSE	$w_k = \frac{(1-MSE_k)}{\sum_{m=1}^{L}(1-MSE_m)}$
WCGINI	$w_k = \frac{max(0,(AUC_k-0.5)\times 2)}{\sum_{m=1}^{L} max(0,(AUC_m-0.5)\times 2)}$
WCIMSE	$w_k = \frac{(1/MSE_k)}{\sum_{m=1}^{L}(1/MSE_m)}$

Another problem of weights is that they may overfit. Consequently, in some experimental analyses [25][26], there are cases where the use of a *uniform weighting* (WCUnif) gives better results. The question that arises is whether we can also have an overfitting problem when probabilities are calibrated.

Consequently, there are many open questions when mixing together probabilities and weighted combinations. Are both things redundant or even incompatible? Is calibration a good idea to get better probability estimations? If calibration is used, would weights become useless?

4.2 Probability Densities and Classifier Combination

After the toy example presented in Example 1, Figure 2 shows the probability densities (for the positive class, $p(i, \oplus)$) for three different classifiers (J48, Random Forest and Naïve Bayes, which are built with Weka [35]) for two problems (the *credit* and the *chess* datasets from the UCI repository [7]). Typically, the positive cases cluster around a high probability and the negative cases cluster around a low probability. When the two clusters are more distant and better delineated (with a thinner shape) there is better separability (and, hence, higher AUC). Also, in these charts, calibration can be easily traced because each bin in the histogram should have a proportion of positive examples equal to its value when the classifier is well calibrated.

A possible way of modelling the above probability distributions would be to work with scores (assuming they are between $-\infty$ and $+\infty$) instead of probabilities. In this case, we would observe that the positive and the negative examples are normally distributed, with these two normals being more or less separated. An alternative option is to work with probabilities as if they were scores clipped

between 0 and 1 and to model them via a truncated normal distribution[1]. Note that, although there are cases where probabilities do not strictly follow a (truncated) normal distribution, the aggregation of several non-normal distributions typically converges to a normal distribution. Therefore, at least for the combined model, this representation is not a strong working hypothesis helping us to conceptually analyse the effect of combining several classifiers.

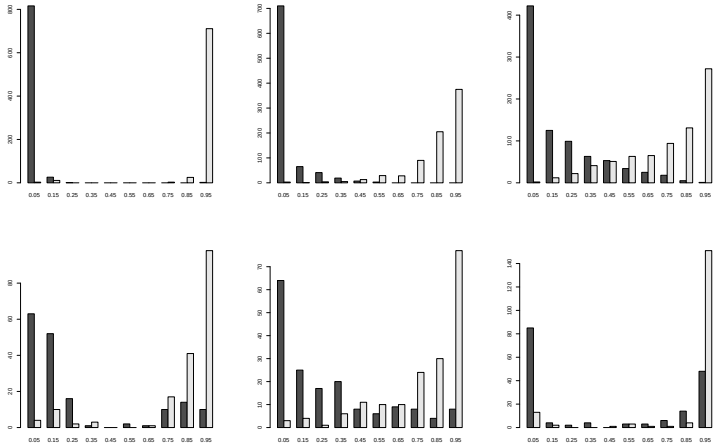


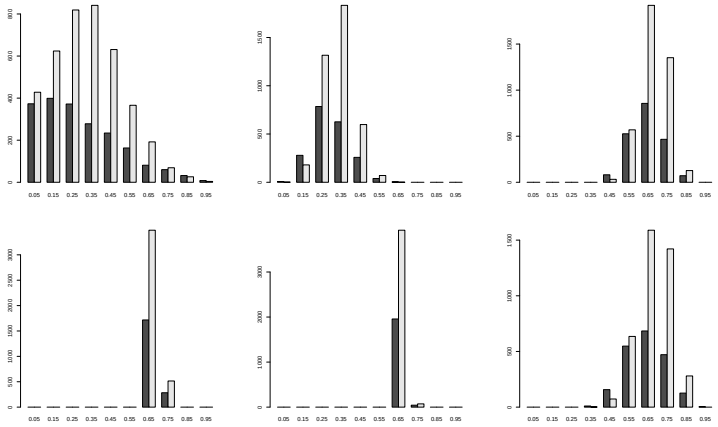
Fig. 2. Probability densities for positive (clear grey) and negative (dark grey) classes, using J48 (left), Random Forest (centre), and Naïve Bayes (right) in Weka. Top row: probability distribution for the *chess* dataset from the UCI repository. Bottom row: probability distribution for *credit* dataset.

It is easy to show that combining independent classifiers that follow normal distributions leads to a combined classifier whose positive (respectively negative) mean is the (weighted) average of the positive (respectively negative) mean of the base classifiers, but the deviation is usually lower. This means that, by using a weighted average combination, the distributions are narrowed, which implies that the combination usually improves in terms of separability (provided the original classifiers were better than random, i.e., the positive mean was greater than the negative mean).

[1] This distribution has been used to model probabilities for binary cases in the *probit model* or in *truncated regression* [1].

4.3 Calibration and Classifier Combination

A naïve view of the effect of calibration in combination would conclude that the better calibrated a classifier is, the better the reliability of its probability estimations is and, hence, the better the combination will be. However, the relationship between classifier calibration and combination is a bit more complex. When we say better, we need to be more precise about the evaluation metric that we are using.



	Top Left	Top Centre	Top Right	Bottom Left	Bottom Centre	Bottom Right
MSE	0.3680	0.3386	0.2154	0.2209	0.2199	0.2142
MSE cal.	0.1519	0.1276	0.0028	0.0008	0.0001	0.0037
MSE ref.	0.2162	0.2126	0.2134	0.2222	0.2222	0.2107
AUC	0.5615	0.6038	0.6038	0.5615	0.6093	0.6093
Acc.(0.5)	0.3855	0.3378	0.6748	0.6667	0.6667	0.6820
CalBin	0.3726	0.3674	0.0452	0.0465	0.0648	0.0395

Fig. 3. Probability densities for positive (clear grey) and negative (dark grey) examples. Top plots: (Left) a single classifier with parameters: $n^{\oplus} = 4000$, $\mu^{\oplus} = 0.3$, $\sigma^{\oplus} = 0.2$, $n^{\ominus} = 2000$, $\mu^{\ominus} = 0.15$ and $\sigma^{\ominus} = 0.3$, (Centre) the result of the combination of 5 *independent* classifiers such as the one depicted on the left, using uniform weights, (Right) Platt's calibration over the combination. Bottom plots: (Left) the result of Platt's calibration over the single classifier on the left of top plot, (Centre) the combination of 5 *independent* classifiers such as the one depicted on the left, using uniform weights, (Right) the post-calibration (using Platt's calibration) over the combination. The tables show the results of different metrics for all the cases.

Figure 3 presents an example illustrating the effect of calibration and combination over several measures. The top centre plot shows the combination of five independent classifiers with bad calibration (as the one shown on the left) using equal weight. AUC is improved, but other measures (such as accuracy) are not necessarily improved. The reason-why is that the base classifiers are not calibrated. The density histogram on the right shows a postprocessing over the combination using Platt's calibration, which is able to 'unveil' the quality of the combined classifier also for accuracy. The bottom plot of Figure 3 presents a similar process but now we calibrate the original base classifiers before their combination (the density histogram, on the left side is a calibrated version, using Platt's method, of the classifier displayed on the left side of the top plot). Since AUC is not very high, the probabilities are highly condensed in a small area of the graph. Even though Platt's calibration is not a linear scaling and we also have some truncation here, we see that there are important differences in accuracy and calibration between the two charts, but the AUC of the combination is almost the same for both cases. In addition, the two plots on the right are surprisingly very similar (although this is not generally the case).

Monotonic transformations preserve the ranking and hence have a limited effect on the results (in terms of AUC). In addition, again we see that the result of the combination after calibrating the base classifiers does not necessarily produce a better calibration of the result. This reinforces the idea of AUC as the appropriate measure of quality, since other measures can be improved by this post-calibration.

Example 2 shows the effect of calibration before and/or after combination, as a specific example of a much more complete and comprehensive battery of experiments that we will do in the following section. Now we focus on several classifiers with different calibration degrees.

Example 2. Consider a problem with 500 positive examples and 1000 negative examples and a diverse set of five classifiers with the following distribution parameters:

#	$\mu^{\oplus}$	$(\sigma_k^{\oplus})^2$	$\mu^{\ominus}$	$(\sigma_k^{\ominus})^2$
1	0.4	0.04	0.3	0.07
2	0.5	0.3	0.3	0.2
3	0.2	0.1	0.3	0.2
4	0.8	0.04	0.2	0.3
5	0.8	0.3	0.7	0.3

Note that we have diversity of base classifiers: not only are they independent, but they also have quite different distributions. There are also good classifiers and bad classifiers (the third classifier is even worse than random). The best single classifier has an AUC of 0.87. We now consider the following calibration and combination layouts: the combination of the base classifiers (comb), the

combination of the calibrated base classifiers (combcal), the calibration of the combination result (combpostcal) and the calibration of the combination of the calibrated base classifiers (combcalpostcal). The results are presented in Table 3.

Table 3. Results for several calibration and combination methods in Example 2.

Method	MSE	MSEcal	MSEref	AUC	Acc	CalBin
comb	0.1933	0.0265	0.1703	0.8010	0.7553	0.1551
combcal	0.1502	0.0744	0.0778	0.9575	0.8026	0.2485
combpostcal	0.1658	0.0030	0.1633	0.8010	0.7646	0.0485
combcalpostcal	0.0749	0.0025	0.07338	0.957534	0.896	0.0326

This example shows the relevance of calibrating before, since only with a proper pre-calibration can we align good and bad classifiers in an optimal way. However, although calibration and combination entails an average of the means and a reduced variance (which generally implies better AUC), *this does not mean that combining perfectly calibrated classifiers generates a perfectly calibrated combination*. Figure 3 also shows this.

Finally, with respect to the accuracy measure, we see that when classifiers are well-calibrated before the combination, the resulting means will be placed on either side of the centre value (the prior class proportion). This suggests that calibration has to be considered as a necessary option after combination to increase calibration and accuracy (as Example 2 shows). Nevertheless, this centre value does not need to be 0.5, so accuracy would highly depend on the decision threshold used.

In summary, *the AUC measure is chosen as a reference for the quality of combination*, since calibration measures for the combination will generally not be good and accuracy will greatly depend on a good threshold choice.

We now have a better understanding of how calibration affects the combination and we have identified the key factors involved: performance measures, use of weights, measure used to derive the weights, calibration monotonicity and moment of calibration (before and/or after combination). These and other issues are addressed through an experimental analysis below.

5 Experimental Analysis

This section provides a comprehensive experimental analysis about the effect of calibration and combination, focussing on the factors identified in the previous section.

5.1 Experimental Settings

For the experimental evaluation, we implemented the evaluation measures and calibration methods (the PAV algorithm, Platt's method, and binning averaging with 10 bins) presented in Sections 3.1 and 3.2. We also defined all the weighted combination schemes that use the weights shown in Table 2.

We selected 30 (small and medium-sized) datasets (Table 4) from the UCI repository [7]. Each dataset was split randomly into four different subsets: one for training, two for validation and one for test (25% of the instances for each set). To simulate a diverse set of classifiers that come from different sources, we used four different methods for classification implemented in the data mining suite WEKA [35] (with their default parameters) to construct several models for each problem: J48 (a C4.5 implementation), Logistic (a logistic regression implementation), IBk ($k = 10$) (a k-NN implementation) and NaïveBayes. An additional random and uncalibrated classifier (Random) was added (when necessary) to the experiments in order to compare what happens when there is a bad classifier. It has also very bad calibration, since probabilities are either 0 or 1. A uniform random distribution was used to decide which probability was set to 1. A total of 100 repetitions were performed for each dataset and classifier. The training set was used to learn each classifier. One validation set was used, in case, to calibrate the probabilities of the original classification models. The other validation set was used, in case, to calibrate the probabilities of the combined model. The training and the first validation sets were also used to tune the weights of the combination methods. The test set was used to validate the models. For each repetition, the same training, validation and test sets were used for all methods.

We evaluated the results of the different layouts for the MSE, AUC, CalBin and accuracy measures: without calibration and combination; with calibration only; with combination only; with precalibration and combination; with combination and postcalibration; and with precalibration, combination and postcalibration, as shown in Table 5. In order to ease the reproducibility of results, all the source code, scripts and datasets are available at a web page[2].

5.2 Experimental Results

We will first study the effect of several combination methods when there is a random classifier (i.e., a bad classifier) along with other more accurate classifiers. We are interested in a first assessment of the weighting methods in Table 2. Tables 6 and 7 show the results[3] of applying the combination methods to the four original classification models ($J48$, Log, $IB10$, and NB). The difference between Tables 6 and 7 is that Table 7 shows the results when a random classifier is added ($Random$). As a reference, we also include the average of all the base classifiers ($BaseAvg$).

When a random classifier is included, the mean of the measures are expected to be worse than with only the four original classifiers. In this situation, some

[2] http://users.dsic.upv.es/∼abella/SBA.zip
[3] These results are averages over datasets.

Table 4. Datasets used in the experiments. Size, number of classes, and number of nominal and numeric attributes.

#	Datasets	Size	c	Nom.	Num.
1	Breast Cancer	286	2	9	0
2	Wisconsin Breast Cancer	699	2	0	9
3	Chess	3196	2	36	0
4	Credit Rating	690	2	9	6
5	German Credit	1000	2	13	7
6	Pima Diabetes	768	2	0	8
7	Haberman Breast	306	2	0	3
8	Heart Disease	303	2	7	6
9	Heart Statlog	270	2	0	13
10	House Voting	435	2	16	0
11	Ionosphere	351	2	0	34
12	Monks1	556	2	6	0
13	Monks2	601	2	6	0
14	Monks3	554	2	6	0
15	Mushroom	8124	2	22	0
16	Mammographic Masses	961	2	4	1
17	Sonar	208	2	0	60
18	Spam	4601	2	0	57
19	Spect	80	2	0	44
20	Tic-tac	958	2	8	0
21	Autos5c	202	5	10	15
22	Cmc	1473	3	7	2
23	Iris	158	3	0	4
24	Segmentation	2310	7	0	19
25	Tae	151	3	2	3
26	Waveform	5000	3	0	21
27	Wine	178	3	0	13
28	Vowel	990	11	3	11
29	Splice	3190	3	60	0
30	Vehicle	846	4	0	18

Table 5. Experimental layouts that arrange combination and calibration.

Layout	Description and Variants
BaseModel	*BaseModel* $\in$ {J48, Logistic, NB, IBk} plus a random model.
Base	The average of all the base models
CombMet	*CombMet* $\in$ {WCUnif, WCAcc, WCAUC, WCGINI, WCMSE, WCIMSE}.
CalMet	*CalMet* $\in$ { PAV, Platt, Binning Averaging}.
CalMet+*CombMet*	For different calibration and combination methods.
CombMet+*CalMet*	For different calibration and combination methods.
CalMet+*CombMet*+*CalMet*	For different calibration and combination methods.

combination methods are more robust than others. Specifically, the $WCGINI$ and $WCIMSE$ methods obtained the best results. In order to see whether the difference between more than two methods is statistically significant, we calculated the Friedman test. If there were differences between these methods, we calculated the Nemenyi post-hoc test to compare all of the methods with each other (with a probability of 99.5%) as suggested in [12]. The results depicted in

Table 6. Results for the base classifiers and CombMet.

	MSE	AUC	CalBin	Acc.
J48	0.1397	0.8202	0.1062	0.7683
Log	0.1526	0.8316	0.1224	0.7752
IB10	0.1375	0.8453	0.1093	0.7590
NB	0.1487	0.8469	0.1278	0.7679
Base	0.1446	0.8360	0.1164	0.7676
Comb	0.1150	0.8718	0.1045	0.8052
WcombAcc	0.1143	0.8724	0.1037	0.8069
WcombAUC	0.1145	0.8726	0.1044	0.8065
WcombGINI	0.1141	**0.8736**	0.1043	0.8077*
WcombMSE	0.1145	0.8722	0.1039	0.8063
WcombIMSE	**0.1109**	0.8735	**0.0960**	**0.8104**

Table 7. Results for the base classifiers and CombMet, with 1 random classifier.

	MSE	AUC	CalBin	Acc.
J48	0.1397	0.8202	0.1062	0.7683
Log	0.1526	0.8316	0.1224	0.7752
IB10	0.1375	0.8453	0.1093	0.7590
NB	0.1487	0.8469	0.1278	0.7679
Random	0.4676	0.5009	0.4398	0.4317
Base	0.2092	0.7690	0.1811	0.7004
Comb	0.1298	0.8520	0.1337	0.7932
WcombAcc	0.1196	0.8630	0.1175	0.8035
WcombAUC	0.1203	0.8627	0.1195	0.8026
WcombGINI	0.1141	0.8729	0.1048	0.8077
WcombMSE	0.1208	0.8623	0.1202	0.8019
WcombIMSE	**0.1117**	0.8697	**0.0986**	**0.8096**

bold in Tables 6 and 7 indicate that $WCGINI$ and $WCIMSE$ are the best and that the difference with the rest of the methods is statistically significant. The results that are underlined indicate that these results are the best and that the difference with the other methods is statistically significant, even though the difference between the underlined methods is not statistically significant.

We also studied whether the difference between the results with and without a random classifier are statistically significant. In order to do so, we calculated the Wilcoxon Signed-Ranks test with a probability of 99.5% as suggested in [12]. The result shows that the difference between the pairs of methods without and with a random classifier is statistically significant, except in the case marked with the symbol $*$. The greatest differences are shown when no weighting is used ($WCUnif$). Therefore, the conclusion that comes from Tables 6 and 7 is that weights are needed when classifiers of different quality are combined, which is consistent with previous knowledge in the field [26]. However, these results show

that some weighting schemes such as $WCGINI$ and $WCIMSE$ are very robust to very bad classifiers.

The next step was to evaluate the effect of calibration and combination together. First, we evaluated whether weighting was necessary when models were well calibrated. Second, we evaluated whether calibration was good for combination. We also wanted to know whether it was better to calibrate the base models first and combine them afterwards, or to combine the models first and to calibrate the combination afterwards.

Table 8 shows the results for each pair of calibration and combination methods[4] for the J48, Log, IB10 and NB classification models, and the random classifier[5]. We also include the average of each calibration method over the base classifiers (PAV, $Platt$, and $Binn$) and the combination by $WCGINI$ and $WCIMSE$ without calibration. We applied, again, the Friedman test to the results in Table 8 using the same notation (bold and underlined).

Table 8. Results for CalMet, CalMet+CombMet and CombMet+CalMet, with 1 random classifier.

	MSE	AUC	CalBin	Acc.
WCGINI	0.1141	0.8729	0.1048	0.8077
WCIMSE	0.1117	0.8697	0.0986	0.8096
PAV	0.1568	0.7642	0.0966	0.7086
Platt	0.1499	0.7665	0.0982	0.7223
Binn.	0.1568	0.7610	0.0980	0.7066
PAV+WCGINI	0.1075	0.8770	0.0916	0.8171
Platt+WCGINI	0.1173	0.8779	0.1293	0.8129
Binn.+WCGINI	0.1082	0.8777	0.0945	0.8171
PAV+WCIMSE	**0.1061**	0.8753	0.0923	0.8193
Platt+WCIMSE	0.1178	0.8768	0.1342	0.8134
Binn.+WCIMSE	0.1075	0.8761	0.0980	0.8194
WCGINI+PAV	0.1141	0.8627	0.0708	0.8088
WCGINI+Platt	0.1117	0.8720	0.1029	0.8117
WCGINI+Binn.	0.1177	0.8528	0.0753	0.8033
WCIMSE+PAV	0.1137	0.8600	**0.0700**	0.8098
WCIMSE+Platt	0.1109	0.8683	0.1006	0.8129
WCIMSE+Binn.	0.1177	0.8490	0.0755	0.8030

[4] We tried the six weighting methods shown in Table 2, but the best results were obtained with $WCGINI$ and $WCIMSE$, so, in what follows, we only show these results.

[5] Apart from the magnitude in the results (values are generally better without a random classifier, as expected), the relative differences are similar, so the conclusions that can be drawn from one case (without random classifier) are similar to those that can be drawn from the other case (with a random classifier). From hereon, we will only show the results including the random classifier.

We can see that MSE is not much better (or even worse when using Platt before the combination) than an uncalibrated combination.The results for AUC are slightly better. In CalBin and accuracy, the difference is a little bit higher, except when Platt's calibration is used, showing that combination produces an uncalibrated classifier. When calibration is applied after combination, the results, except for CalBin, are worse in general. Therefore, if we are only interested in improving the calibration of the combined models, the best option is to calibrate their probabilities after the combination. But if we want to improve the MSE, AUC and accuracy measures (the performance of the model), it is better to first calibrate the base classifiers and then combine them.

Finally, we are going to study the effect of *calibration* + *combination* + *calibration* (Table 9). The idea is to check whether we can improve both the calibration and the performance of the combined model. The results show that calibrating the combined model improves the calibration, but does not improve MSE, AUC and accuracy. To improve CalBin, the best layout seems to be any calibration method + WCIMSE + PAV, but only the difference between the layout PAV + WCIMSE + PAV is statistically significant compared to the rest of the results (for CalBin measure) shown in Table 9 and the WCIMSE + PAV layout (in Table 8).

Table 9. MSE, AUC, CalBin and accuracy measures for CalMet+CombMet+CalMet, with 1 random classifier.

	MSE	AUC	CalBin	Acc.
PAV+WCGINI+PAV	0.1105	0.8680	0.0688	0.8145
PAV+WCGINI+Platt	0.1080	0.8764	0.0986	0.8176
PAV+WCGINI+Binn.	0.1147	0.8571	0.0748	0.8074
Platt+WCGINI+PAV	0.1117	0.8676	0.0699	0.8129
Platt+WCGINI+Platt	0.1091	0.8772	0.1005	0.8158
Platt+WCGINI+Binn.	0.1161	0.8553	0.0745	0.8057
Binn.+WCGINI+PAV	0.1108	0.8682	0.0685	0.8136
Binn.+WCGINI+Platt	0.1082	0.8773	0.0985	0.8171
Binn.+WCGINI+Binn.	0.1155	0.8559	0.0752	0.8058
PAV+WCIMSE+PAV	0.1093	0.8666	**0.0671**	0.8155
PAV+WCIMSE+Platt	0.1066	0.8747	0.0974	0.8194
PAV+WCIMSE+Binn.	0.1138	0.8538	0.0742	0.8091
Platt+WCIMSE+PAV	0.1098	0.8672	0.0682	0.8159
Platt+WCIMSE+Platt	0.1073	0.8757	0.0993	0.8188
Platt+WCIMSE+Binn.	0.1147	0.8536	0.0747	0.8076
Binn.+WCIMSE+PAV	0.1095	0.8671	0.0678	0.8156
Binn.+WCIMSE+Platt	0.1069	0.8751	0.0983	0.8188
Binn.+WCIMSE+Binn.	0.1144	0.8522	0.0749	0.8083

From the previous battery of experiments we can highlight some major findings:

– The combined model is not calibrated, as it is also shown in Section 4.
– Calibration before combination makes a limited improvement for AUC and accuracy, and no improvement (or even gets worse results) for MSE and CalBin.
– Calibration after combination gives a better picture for calibration measures, but, as expected, AUC is not increased. This is because the calibration methods are monotonic, and there is almost no increase in accuracy or MSE.
– Calibration + Combination + Calibration gives the best results in terms of calibration, but it is clearly an elaborate layout, which requires two validation datasets (one for each of the calibration processes).

From these results, it seems that calibration is only slightly effective for classifier combination, and weighting can do almost as well. Nonetheless, this statement might be more precise if we say that *monotonic* calibration (as given by PAV, Platt and Binning Averaging) does not bring an important push in performance for classifier combination. As a result, in the following section, we will focus on the development of a non-monotonic calibration method which tries to integrate more information from the dataset.

6 Nonmonotonic Calibration

Most calibration methods are based on a univariate transformation function over the original estimated class probability, as we saw in Section 3.2. This function is always monotonic (strictly monotonic for Platt's method). One possible reason-why is that if we were allowed to modify the probabilities in a non-monotonic way, we would be prone to overfitting. In the end, calibration must be an adjustment of the estimated probability values, but not a complete change in the model properties. For calibration methods, one way of doing this is to observe the ordering of the examples (given by their estimated probability), which can be achieved by using a monotone transformation.

However, is the previous rationale true for multiclass calibration? As we discussed in Section 3.3, PAV, binning averaging and Platt's methods are binary since they are only applied to one probability, i.e., they are univariate. Consequently, we have to use a one-vs-all or all-vs-all schema to turn binary calibration methods into multiclass calibration methods. Nevertheless, *the extensions of binary monotonic calibration methods to multiclass calibration do not ensure monotonicity*, as the following example shows.

Example 3. Consider a classifier for a three-class problem with classes $\{\mathbf{a}, \mathbf{b}, \mathbf{c}\}$ which outputs the following estimations for two examples 1 and 2:
$p(1, \mathbf{a}) = 0.2, p(1, \mathbf{b}) = 0.6, p(1, \mathbf{c}) = 0.2$;
$p(2, \mathbf{a}) = 0.1, p(2, \mathbf{b}) = 0.3, p(2, \mathbf{c}) = 0.6$
After a monotonic calibration for each class, we may have the following probabilities:
$p^*(1, \mathbf{a}) = 0.7$, $p^*(1, \mathbf{b}) = 0.9$, $p^*(1, \mathbf{c}) = 0.4$;
$p^*(2, \mathbf{a}) = 0.6$, $p^*(2, \mathbf{b}) = 0.4$, $p^*(2, \mathbf{c}) = 0.5$

The rankings are maintained for the three classes, that is, $\forall\, class \in \{\mathbf{a}, \mathbf{b}, \mathbf{c}\}$: $p^*(i, class) > p^*(j, class)$ iff $p(i, class) > p(j, class)$. But when we normalise, we have:

$p^*(1, \mathbf{a}) = 0.35$, $p^*(1, \mathbf{b}) = 0.45$, $p^*(1, \mathbf{c}) = 0.2$;
$p^*(2, \mathbf{a}) = 0.4$, $p^*(2, \mathbf{b}) = 0.27$, $p^*(2, \mathbf{c}) = 0.33$

which breaks the monotonicity for class $\mathbf{a}$ since now $p^*(2, \mathbf{a}) > p^*(1, \mathbf{a})$ and, thus, example 2 is ranked above example 1 for class $\mathbf{a}$.

The previous example shows that a one-vs-all approach using a monotonic calibration method does not ensure a monotonic transformation. Similar results can be obtained for the all-vs-all schema and other multiclass extensions from binary transformations simply because of the normalisation.

Therefore, does it make sense to stick to monotonic methods when, in the general multiclass case, they become non-monotonic in the end?

Following this argument, we propose the application of a calibration method which was meant to be non-monotonic from scratch [4]. The core of this approach is to change the idea of "sorting" the examples by its probability into the idea of using similarity between examples to create bins that are specific for each instance. This idea arises from the fact that if bins are created by only using the estimated probability, calibrated probabilities will be computed from possibly different examples with similar probabilities. Hence, the effect of calibration will be small since we average similar probabilities. However, if we construct the bins using similar examples according to their features, probabilities can be more diverse and calibration will have more effect.

Based on this reasoning, we have adapted a new calibration method known as Similarity-Binning Averaging (SBA) [4] for the combination setting (for which it was never analysed before). In this method the original attributes and the estimated probability are used to calculate the calibrated one.

The method is composed of two stages. The left side of Figure 4 shows "Stage 1" of the SBA method. In this stage, a given model M gives the estimated probabilities associated with a dataset. This dataset can be the same one used for training, or an additional validation dataset VD. The estimated probabilities $p(i, j)$ $1 \leq j \leq c$ are added (as new attributes) to each instance i of VD, creating a new dataset VDP.

The right side of Figure 4 shows "Stage 2" of the SBA method. To calibrate a new instance I, first, the estimated probability for each class is obtained from the classification model M, and these probabilities (one for each class) are added to the instance, thus creating a new instance (IP). Next, the k-most similar instances to this new instance are selected from the dataset VDP (for example, using the k-NN algorithm). This creates a bin. Finally, the calibrated probability of I for each class j is the average predicted class probability of this bin (i.e., the probability estimated by the k-NN algorithm for each class j of the instance I).

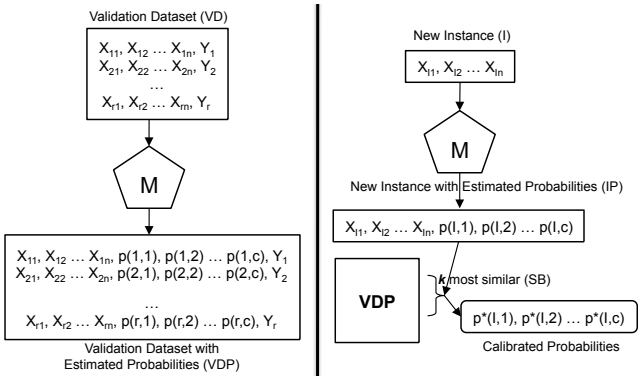


Fig. 4. Left: Stage 1 of the SBA method. Right: Stage 2 of the SBA method.

7 Experimental Results of the SBA Calibration Method

In this section, we experimentally evaluate the results of the SBA calibration method. We have seen that the calibration methods evaluated in Section 5 produce poor improvements (and do not improve the four studied measures equally or at the same time). So, in this section we want to evaluate whether the SBA method can change the picture in the context of classifier combination.

Tables 10 and 11 use the same datasets and methodology, showing the same results as Tables 6 and 7, but here we include the four calibration methods.

Table 10. MSE, AUC, CalBin and accuracy measures for base classifiers without and with calibration.

	MSE	AUC	CalBin	Acc.
J48	0.1397	0.8202	0.1062	0.7683
Log	0.1526	0.8316	0.1224	0.7752
IB10	0.1375	0.8453	0.1093	0.7590
NB	0.1487	0.8469	0.1278	0.7679
BaseAvg	0.1446	0.8360	0.1164	0.7676
PAV	0.1310	0.8301	**0.0778**	0.7779
Platt	0.1309	0.8333	0.1055	0.7749
Binn.	0.1316	0.8260	0.0807	0.7753
SBA	**0.1205**	**0.8726**	0.1022	**0.7965**

In terms of MSE, AUC and accuracy, the SBA method obtains the best results for the 20 binary datasets and the 10 non-binary datasets. In terms of

Table 11. MSE, AUC, CalBin and accuracy measures for base classifiers without and with calibration, with one random classifier.

	MSE	AUC	CalBin	Acc.
J48	0.1397	0.8202	0.1062	0.7683
Log	0.1526	0.8316	0.1224	0.7752
IB10	0.1375	0.8453	0.1093	0.7590
NB	0.1487	0.8469	0.1278	0.7679
Random	0.4676	0.5009	0.4398	0.4317
BaseAvg	0.2092	0.7690	0.1811	0.7004
PAV	0.1568	0.7642	**0.0966**	0.7086
Platt	0.1499	0.7665	0.0982	0.7223
Binn.	0.1568	0.7610	0.0979	0.7066
SBA	**0.1264**	**0.8648**	0.1080	**0.7841**

CalBin, the calibration method that obtains the best results is the PAV method. The results in bold mean that the differences are statistically significant.

The problem of non-monotonicity is that the more transformations we do, the more overfitting may occur, and, more importantly, the correlation between the classifiers may increase (loss of diversity). We calculated Pearson's correlation coefficient for the base classifiers before and after calibrating them with the three traditional calibration techniques and SBA. While the traditional calibration techniques showed no significant increase in correlation, this was effectively higher for SBA. A higher correlation is, in theory, worse (less diversity), *unless* there is a general increase in classifier quality (when classifiers get better then they necessarily must correlate more). This latter situation seems to be more consistent here, as the results for AUC are much better. Nonetheless, a more thorough analysis on the relation between non-monotonic calibration and classifier diversity should be done. In what follows, we will focus on whether the overall results are better, since many factors counter-balance here.

Next, we study the effect of combination, calibration+combination, combination+calibration and calibration+combination+ calibration in Table 12 with the SBA calibration method. We compare these results with the results in Tables 6, 7, 8 and 9, using the Friedman test.

This table shows that SBA gives the best results in terms of AUC. The improvement is now much higher than it was for the other methods. The options of using WCGINI or WCIMSE are not significant, but they are still better than other weighting options (not shown in the table). The layouts with the best results in terms of AUC are SBA + WCIMSE and SBA + WCIMSE + Platt. The difference between these results and the rest of the results in Table 12 and Table 8 are statistically significant according to the Friedman test. Otherwise, the difference between the result of the layout SBA + WCIMSE + Platt and PAV+WCIMSE (the best result in Table 8), in terms of MSE, is not statistically significant. The difference between the result of the layout SBA + WCIMSE + PAV and PAV + WCIMSE + PAV (the best result in Table 9), in terms of

Table 12. CombMet, CalMet+CombMet, CombMet+CalMet and CalMet+CombMet+CalMet results, with 1 random classifier. $*$: difference non significant respect PAV+WCIMSE+PAV. $\star$: difference non significant respect PAV+WCIMSE. $\bullet$: difference non significant respect PAV+WCIMSE and Binn.+WCIMSE.

	MSE	AUC	CalBin	Acc.
SBA+WCGINI	0.1145	0.8841	0.1124	0.8079
SBA+WCIMSE	0.1120	0.8846	0.1056	0.8104
WCGINI+SBA	0.1147	0.8781	0.0996	0.8081
WCIMSE+SBA	0.1141	0.8762	0.0975	0.8078
PAV+WCGINI+SBA	0.1123	0.8789	0.0969	0.8103
Platt+WCGINI+SBA	0.1131	0.8786	0.0978	0.8101
Binn.+WCGINI+SBA	0.1128	0.8777	0.0971	0.8086
PAV+WCIMSE+SBA	0.1100	0.8782	0.0933	0.8128
Platt+WCIMSE+SBA	0.1111	0.8781	0.0949	0.8119
Binn.+WCIMSE+SBA	0.1112	0.8765	0.0947	0.8108
SBA+WCGINI+PAV	0.1093	0.8732	0.0680	0.8161
SBA+WCGINI+Platt	0.1076	0.8842	0.1031	0.8185
SBA+WCGINI+Binn.	0.1135	0.8618	0.0734	0.8089
SBA+WCGINI+SBA	0.1149	0.8794	0.1021	0.8068
SBA+WCIMSE+PAV	0.1085	0.8732	0.0675$*$	0.8172
SBA+WCIMSE+Platt	0.1066$\star$	0.8846	0.1023	0.8198$\bullet$
SBA+WCIMSE+Binn.	0.1128	0.8620	0.0728	0.8106
SBA+WCIMSE+SBA	0.1140	0.8792	0.1002	0.8082

CalBin, is not statistically significant. And finally, the difference between the result of the layout SBA + WCIMSE + Platt, PAV + WCIMSE and Binn + WCIMSE (the best results in Table 8), in terms of accuracy, is not statistically significant.

In addition, we see again that the use of weights is compatible with this calibration method. Consequently, calibration and weighting for a wide range (monotonic or not) of calibration methods (even using the same validation dataset) is not only compatible but also beneficial.

8 Discussion and Conclusions

In general terms, we now have a better understanding of classifier combination using probabilities. The separability and location of the probability distributions is the key issue in understanding how classifier combination works. Measures such as AUC, MSE and CalBin are very useful in distinguishing these separability and location parameters.

Apart from all these findings, it is also worth considering whether the new weighting methods, layouts, and calibration methods are able to improve the state-of-the-art in classifier combination using probabilities. There is definitely

a relevant increase in the quality of the combined models over the techniques with traditional weighting.

In order to give a clearer picture of the overall improvement, Table 13 summarises this evolution of results.

Table 13. Summary of Results (using 4 models + random classifier). Each result is accompanied with some letters. For each column (MSE, AUC, CalBin, and accuracy measures) we have done a significant statistical test between the rows with the same letter. If the difference between the methods with the same letter is significant, we have put the letter of the best result in bold; and if two or more methods are better than the rest, but the difference between them is not significant, we have underlined the letters of these methods.

	MSE	AUC	CalBin	Acc.
Base	0.2092 a	0.7690 a	0.1811 a	0.7004 a
WCUnif	0.1298 **a**, b, d, e	0.8520 **a**, b, d, e	0.1337 **a**, b, d, e	0.7932 **a**, b, d, e
WCAcc	0.1196 **b**, c, d, e	0.8630 **b**, c, d, e	0.1175 **b**, c, d, e	0.8035 **b**, c, d, e
WCGINI	0.1141 **c**, **d**, **e**, f, g	0.8729 **c**, $\underline{d}$, e, f, g	0.1048 **c**, **d**, **e**, f, g	0.8077 **c**, $\underline{d}$, e, f, g
Platt+WCUnif	0.1294 d, f	0.8745 $\underline{d}$, f	0.1589 d, f	0.8063 $\underline{d}$, f
Platt+WCGINI	0.1173 e, f	0.8779 **e**, $\underline{f}$	0.1293 e, f	0.8129 **e**, f
Platt+WCGINI+Platt	0.1091 **f**, g	0.8772 $\underline{f}$, g	0.1005 **f**, g	0.8158 **f**, g
SBA+WCGINI	0.1145 g	0.8841 $\underline{g}$	0.1124 g	0.8079 g
SBA+WCGINI+SBA	0.1149 g	0.8794 g	0.1021 g	0.8068 g
SBA+WCGINI+PAV	0.1093 $\underline{g}$	0.8732 g	0.0680 **g**	0.8161 g
SBA+WCGINI+Platt	0.1076 $\underline{g}$	0.8842 $\underline{g}$	0.1031 g	0.8185 **g**

Firstly, the WCUnif layout shows an unweighted combination of the base classifiers (including one random classifier). There is a clear improvement in all the parameters over the average of the base classifiers (Base). This is significantly better if we use a weighted combination using a classical combination accuracy (WCAcc). Up to this point, this is a state-of-the-art solution. If we modify the weighting function to GINI (WCGINI), we get a significant improvement over WCAcc.

Secondly, the use of a traditional (monotonic) calibration method (Platt's) is able to improve the results (both for the unweighted case and for the weighted case using WCGINI). Nonetheless, as discussed in previous sections: using calibration before combination typically yields better (but uncalibrated) combinations, and the improvement is not applicable to MSE or CalBin. However, this can be easily sorted out by also using a postcalibration (layout: Platt + WCGINI + Platt).

Thirdly, the SBA calibration method is able to get further improvement, especially in terms of AUC. The layout SBA+WCGINI excels in AUC. Again, if we are interested in a calibrated combination or in good accuracy, we can use the layout SBA+WCGINI+PAV, which gives the best results in terms of MSE and CalBin (AUC is worse for this layout because PAV is not strictly monotonic and makes ties that may reduce the AUC). For accuracy, SBA+WCGINI+Platt seems a better option, while keeping AUC at its best.

As final recommendations, we think that classifiers that are seen as probabilistic estimators (and virtually any classifier can be converted into a probability estimator) give a more complete view of their behaviour, allowing for a more de-

tailed combination, using their own reliabilities. The notions of diversity and quality become more complex than for crisp (non-probabilistic), but this extra complexity can pay off with an increase in the performance of the combined model. Performance should be evaluated with several data metrics, but separability (measured in terms of AUC) is a good reference, since it is insensitive to miscalibration. Pursuing a combined model with good AUC makes sense since we know that we can calibrate a classifier with good AUC and get good accuracy results from these calibrated probabilities, using the by default thresholds (e.g. 0.5 for binary datasets).

From all these results and analyses, we would like to highlight some clear messages, as follows:

- Calibration is beneficial before combination as the experimental results show, *in general*. Monotonic calibration methods have a more limited influence than non-monotonic ones.
- The combination of classifiers does not typically give a calibrated result, as we have shown by analysing the probability distributions using truncated normal models for them. This has been confirmed by the experimental results.
- We advocate for AUC as the right measure to evaluate combination performance, precisely because the combination is generally uncalibrated.
- We recommend calibration after combination, if we are interested in good results in terms of MSE or in terms of accuracy.
- Weighted combination is compatible with probabilities even when we use calibration with the same dataset from which we derive the weights. This has been shown by the experiments. Therefore, the "double-weighting" is not really a problem, or at least it is counteracted by other benefits.
- The weighting methods which are best when using probabilities are GINI and IMSE, even in conjunction with calibration.
- SBA, the non-monotonic calibration method, is better for combination according to the experimental results.

This better understanding of classifier combination using probabilities is not only useful for the general case, but for specific applications and problems. We now have tools to analyse how classifiers change with calibration and combination. The distribution plots we used in section 4 can be used to analyse how calibration and combination works with a specific set of classifiers. The use of several metrics (such as AUC, MSE, accuracy, and CalBin) are a requirement to understand what is really going on when classifiers are transformed and combined.

Finally, we have also raised many new questions. More elaborate tools could be used to analyse probability distributions theoretically, especially to address the general multiclass case. The use of other diversity measures, such as Spearman's rank correlation would also be insightful. Empirical results can also be extended with more layouts, different settings, datasets sizes and features, model types, etc. In the end, calibration is a complex phenomenon by itself, which becomes even more convoluted when coupled with the already multifarious area of model combination.

References

1. Takeshi Amemiya. Regression Analysis when the Dependent Variable Is Truncated Normal. *Econometrica*, 41(6):997–1016, 1973.
2. Miriam Ayer, H.D. Brunk, G.M. Ewing, W.T. Reid, and Edward Silverman. An empirical distribution function for sampling with incomplete information. *Annals of Mathematical Statistics*, 5:641–647, 1955.
3. A. Bella, C. Ferri, J. Hernandez-Orallo, and M.J. Ramirez-Quintana. Calibration of machine learning models. In *Handbook of Research on Machine Learning Applications*, pages 128–146. IGI Global, 2009.
4. A. Bella, C. Ferri, J. Hernández-Orallo, and M.J. Ramírez-Quintana. Similarity-Binning Averaging: A Generalisation of Binning Calibration. In *Intelligent Data Engineering and Automated Learning - IDEAL 2009*, volume 5788 of *Lecture Notes in Computer Science*, pages 341–349. Springer Berlin / Heidelberg, 2009.
5. Paul N. Bennett. *Building Reliable Metaclassifiers for Text Learning*. PhD thesis, Carnegie Mellon University, 2006.
6. Paul N. Bennett, Susan T. Dumais, and Eric Horvitz. The Combination of Text Classifiers Using Reliability Indicators. *Information Retrieval*, 8(1):67–98, 2005.
7. C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998.
8. Leo Breiman. Bagging predictors. *Machine Learning*, 24:123–140, August 1996.
9. Niko Brümmer. *Measuring, refining and calibrating speaker and language information extracted from speech*. PhD thesis, University of Stellenbosch, 2010.
10. Rich Caruana, Art Munson, and Alexandru N. Mizil. Getting the Most Out of Ensemble Selection. In *ICDM '06: Proceedings of the Sixth International Conference on Data Mining*, pages 828–833, Washington, DC, USA, 2006. IEEE Computer Society.
11. Rich Caruana and Alexandru Niculescu-Mizil. Data mining in metric space: an empirical analysis of supervised learning performance criteria. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '04, pages 69–78, New York, NY, USA, 2004. ACM.
12. Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, December 2006.
13. Thomas G. Dietterich. Ensemble methods in machine learning. In *Proceedings of the First International Workshop on Multiple Classifier Systems*, MCS '00, pages 1–15, London, UK, 2000. Springer-Verlag.
14. Thomas G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, 40:139–157, August 2000.
15. C. Ferri, J. Hernández-Orallo, and R. Modroiu. An experimental comparison of performance measures for classification. *Pattern Recognition Letters*, 30:27–38, January 2009.
16. C. Ferri, J. Hernández-orallo, and M.A. Salido. Volume under the roc surface for multi-class problems. Exact computation and evaluation of approximations. In *Proceedings of 14th European Conference on Machine Learning*, pages 108–120, 2003.
17. César Ferri, Peter Flach, and José Hernández-Orallo. Delegating classifiers. In *Proceedings of the twenty-first international conference on Machine learning*, ICML '04, pages 37–45, New York, NY, USA, 2004. ACM.
18. P. Flach, H. Blockeel, C. Ferri, J. Hernández-Orallo, and J. Struyf. Decision support for data mining: An introduction to ROC analysis and its applications. In

Data Mining and Decision Support: Integration and Collaboration, pages 81–90. Kluwer Academic Publishers, Boston, 2003.
19. Yoav Freund and Robert E. Schapire. Experiments with a New Boosting Algorithm. In *International Conference on Machine Learning*, pages 148–156, 1996.
20. Joao Gama and Pavel Brazdil. Cascade generalization. *Machine Learning*, 41:315–343, December 2000.
21. Ursula Garczarek. *Classification Rules in Standardized Partition Spaces*. PhD thesis, Universitat Dortmund, 2002.
22. Martin Gebel. *Multivariate calibration of classifier scores into the probability space*. PhD thesis, University of Dortmund, 2009.
23. David J. Hand and Robert J. Till. A simple generalisation of the area under the roc curve for multiple class classification problems. *Machine Learning*, 45:171–186, October 2001.
24. Jennifer A. Hoeting, David Madigan, Adrian E. Raftery, and Chris T. Volinsky. Bayesian model averaging: A tutorial. *Statistical Science*, 14(4):382–417, 1999.
25. Ludmila I. Kuncheva. A theoretical study on six classifier fusion strategies. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24:281–286, February 2002.
26. Ludmila I. Kuncheva. *Combining Pattern Classifiers: Methods and Algorithms*. Wiley-Interscience, 2004.
27. Ludmila I. Kuncheva. Diversity in multiple classifier systems. *Information Fusion*, 6(1):3 – 4, 2005. Diversity in Multiple Classifier Systems.
28. Ludmila I. Kuncheva and Christopher J. Whitaker. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine Learning*, 51:181–207, May 2003.
29. A. H. Murphy. Scalar and vector partitions of the probability score: Part II. n-state situation. *Journal of Applied Meteorology*, 11:1182–1192, 1972.
30. John C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in Large Margin Classifiers*, pages 61–74. MIT Press, Boston, 1999.
31. Adrian E. Raftery, Tilmann Gneiting, Fadoua Balabdaoui, and Michael Polakowski. Using bayesian model averaging to calibrate forecast ensembles. monthly weather review 133, 2005.
32. Ryan Rifkin and Aldebaro Klautau. In defense of one-vs-all classification. *Journal of Machine Learning Research*, 5:101–141, December 2004.
33. Tim Robertson, F. T. Wright, and R. L. Dykstra. *Order Restricted Statistical Inference*. John Wiley & Sons, 1988.
34. Sergey Tulyakov, Stefan Jaeger, Venu Govindaraju, and David Doermann. Review of classifier combination methods. In Hiromichi Fujisawa Simone Marinai, editor, *Studies in Computational Intelligence: Machine Learning in Document Analysis and Recognition*, pages 361–386. Springer, 2008.
35. Ian H. Witten and Eibe Frank. Data mining: practical machine learning tools and techniques with java implementations. *SIGMOD Record*, 31:76–77, March 2002.
36. David H. Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 1992.
37. Bianca Zadrozny and Charles Elkan. Transforming classifier scores into accurate multiclass probability estimates. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '02, pages 694–699, New York, NY, USA, 2002. ACM.