Reframing in context: A systematic approach for model reuse in machine learning

José Hernández-Orallo
DSIC, Universitat Politècnica de València, Spain, jorallo@dsic.upv.es

Adolfo Martínez-Usó
DSIC, Universitat Politècnica de València, Spain, admarus@dsic.upv.es

Ricardo B.C. Prudêncio
Centro de Informática, Universidade Federal de Pernambuco, Recife (PE), Brazil, rbcp@cin.ufpe.br

Meelis Kull
Department of Computer Science, University of Bristol, UK, meelis.kull@bristol.ac.uk

Peter Flach
Department of Computer Science, University of Bristol, UK, peter.flach@bristol.ac.uk

Chowdhury Farhan Ahmed
ICube, Université de Strasbourg, France, cfahmed@unistra.fr

Nicolas Lachiche
ICube, Université de Strasbourg, France, nicolas.lachiche@unistra.fr

Abstract:

We describe a systematic approach called *reframing*, defined as the process of preparing a machine learning model (e.g., a classifier) to perform well over a range of operating contexts. One way to achieve this is by constructing a *versatile* model, which is not fitted to a particular context, and thus enables model reuse. We formally characterise reframing in terms of a taxonomy of context changes that may be encountered and distinguish it from model retraining and revision. We then identify three main kinds of reframing: input reframing, output reframing and structural reframing. We proceed by reviewing areas and problems where some notion of reframing has already been developed and shown useful, if under different names: re-optimising, adapting, tuning, thresholding, etc. This exploration of the landscape of reframing allows us to identify opportunities where reframing might be possible and useful. Finally, we describe related approaches in terms of the problems they address or the kind of solutions they obtain. The paper closes with a re-interpretation of the model development and deployment process with the use of reframing.

Keywords: machine learning, reframing, model reuse, operating context, cost-sensitive evaluation

AI Communications
ISSN 0921-7126, IOS Press.

1. Introduction

Reuse of learnt knowledge is of critical importance in the majority of knowledge-intensive application areas, particularly because the operating context can be expected to vary from training to deployment. In machine learning this has been most commonly studied in relation to variations in class and cost skew in classification. While one *crisp* classifier outputting class labels may be sufficient and highly specialised for one particular operating context (e.g., the positive class being ten times more likely than the negative class), it may not perform well for significantly different operating contexts (e.g., balanced classes). Instead of training several specialised models for each particular operating context, it is more cost-effective to learn one general, versatile model, such as a scoring classifier outputting scores or probabilities, which can be adapted to several contexts through an appropriate procedure, such as the choice of a decision threshold.

In this paper we develop the hypothesis that this successful but narrow approach can be generalised to many other problems and areas in machine learning, where models are required to be more general and adaptable to changes in the data distribution, data representation, associated costs, noise, reliability, background knowledge, etc. This naturally leads to a perspective in which models are not continuously retrained and re-assessed every time a change happens, but rather kept, enriched and validated in a long-term 'model life-cycle'. We define this generalised approach, which we call *reframing*, as the process of preparing and devising the model deployment procedure to perform well over a range of operating contexts beyond the specific context in which the model was trained. Figure 1 provides an illustration of this process, in which the notion of a *versatile* model, able to generalise over a range of contexts, is key.

Many other recent machine learning approaches have addressed the need to cope with context changes. Areas such as domain adaptation, transfer learning, transportability, meta-learning, cost-sensitive learning, incremental and online learning, among others, have proposed new techniques and methods. However, only some of these approaches really perform model reuse, i.e., the same model being applied *systematically* for changing contexts. Generally, in these areas the context change is analysed when it happens, rather than being anticipated. Reframing, in contrast, formalises the expected context changes before any learning takes place, parametrises the space of contexts, analyses its distribution and creates models that can systematically deal with that distribution of context changes. This can only be achieved by a versatile model, which is reframed using the particular context information for each deployment situation, and not retrained or revised whenever the operating contexts change. Rather than being an umbrella term for the above-mentioned related areas, *reframing is a distinctive way of addressing context changes by anticipating them from the outset*.

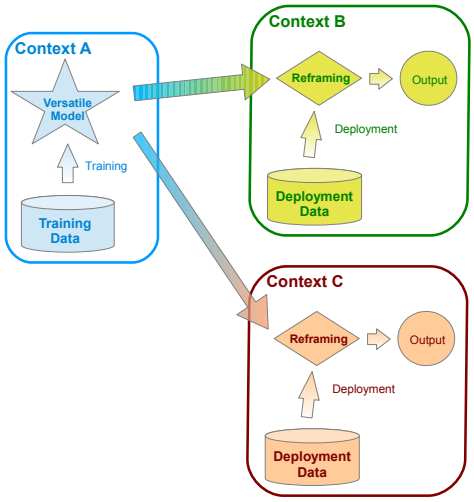


Fig. 1. A general, versatile model is learnt in context A so that it can be reframed to operate in many other contexts (e.g., B and C) without retraining it repeatedly.

The rest of the paper is organised as follows. In Section 2 we define and discuss the central notions of context and context change as they manifest themselves in machine learning. Section 3 discusses the three main alternatives for adapting to context: retraining, revising and reframing. The latter is further elaborated in Section 4, where we distinguish the three main kinds of reframing. Section 5 considers the important question of context-aware performance evaluation and visualisation. Section 6 reviews existing approaches that are related to reframing and the general goal of adapting to multiple contexts. Section 7 concludes.

2. Contexts and context changes

In this section we provide a definition of context, a taxonomy of context changes and a discussion of issues relating to context characterisation.

Definition 1. *A parametrised context* θ *is a tuple of one or more parameter values, discrete or numerical, that represent or summarise the kind of variable infor-*

Context change	Examples of parametrised context
Distribution shift (covariate, prior probability, concept)	Input or output variable distribution
Costs and evaluation function	Cost proportion, cost matrix, loss function
Data quality (uncertain, missing, or noisy information)	Noise or uncertainty degree, missing attribute set
Representation change, constraints, background knowledge	Granularity level, complex aggregates, attribute set
Task change	Binarised regression cutoff, bins

Table 1
Taxonomy of context change types and examples of their parametrisation.

mation, extrinsic to the data, that affects the data distribution, data representation, data quality, the utility function, or the task itself.

For instance, $\theta_1 = \langle \text{Bristol}, 15 \rangle$ and $\theta_2 = \langle \text{Paris}, 25 \rangle$ are contexts denoting location and temperature, which can affect data distribution in different applications (e.g., sales prediction). As another example, in binary classification, a cost matrix is a context that can be parametrised by a single value $\theta = \langle c \rangle$, where $c = c_{FN}/(c_{FN} + c_{FP})$ is known as the cost proportion, expressed in terms of the False Negative (FN) cost and the False Positive (FP) cost. For a given problem, the set of all possible contexts is denoted by $\mathbb{C}$.

2.1. Taxonomy of context changes

Next we present a taxonomy of context changes (summarised and exemplified in Table 1) that are commonly observed in machine learning applications. Although not intended to be exhaustive or mutually exclusive, this taxonomy can help to bring together previous work developed in different but related areas.

Distribution shift: The most obvious type of context change is given by a change in the data distribution. One common way of looking at a change in the data is known as *data shift* [53,49]. Data shift is usually classified into covariate shift, prior probability shift and concept drift, but more thorough classifications have been developed (see, e.g., [49]).

Costs: This context category includes, for instance, changes in misclassification costs or changes in tolerance levels and asymmetric costs for regression models [36,8,32]. Often data shift and cost context changes are closely related. For instance, in ROC analysis, class distribution and cost proportions can be combined into the notion of skew [24].

Data quality: The quality of the data can also change from context to context, due to a variety of domain-dependent reasons (e.g., faults, random fluctuations, low reliability of attributes, ...). For instance, some applications may suffer changes in the noise level (both in input and output) in such a way that models can be adapted to produce more reliable outputs [28].

Representation change: This category is observed when the attribute representation or their meaning changes from context to context. For instance, some attributes may be merged, or the granularity of the data may change [46] (e.g., a model was built for forecasting sales at city-level but will now need to be adapted to country-level).

Task change: A more radical context change is when the task itself changes. For instance, in a classification task new classes can appear in the deployment data, without having been observed in the training data [57]. As another example, a regression task may become a classification task due to new objectives. In this case, one can build a classifier for the new context by applying a cutoff to the original regressor's outputs [34].

2.2. Issues in context characterisation and detection

Often the deployment context is not explicitly given and needs to be (partly) inferred. In general, we need an estimate of the context $\hat{\theta}$ using a function $\Gamma : \mathbb{D} \times \mathbb{M} \times \mathbb{C} \to \mathbb{C}$, which takes as input: (1) some data (it can be the deployment data itself or some additional data); and optionally (2) the original model and (3) the original context when this model was built.

For instance, in binary classification we may be given a cost matrix, from where we get a single parameter c, representing the cost proportion. But we may additionally need to know the class proportion during application time. Since we can infer this proportion from a few labelled examples observed in the deployment context, Γ is a very simple procedure in this case, which requires neither the trained model m nor the original context c.

Alternatively, suppose we have learnt a model m with a training dataset that has a ratio c of positives against negatives. In deployment, we may have a small labelled dataset where we infer that the ratio is c'. In order to compensate for the distribution shift we need

to reframe the model using the *context change* calculated as $(1-c)c'/((1-c)c'+c(1-c'))$ [24]. Hence, both the deployment data and the original context are used to produce a representation for the new context *relative to the training context*.

As an example where the model itself can be helpful to infer the context, consider the input data shift, also known as covariate shift. One way to confirm a hypothetical shift is if it improves a model's performance on some deployment data [3].

It is important to highlight that *reframing*, as we will see in the following section, does not require a threshold above which a context change triggers some kind of action (e.g., a revision of the model). On the contrary, reframing works with the parametrised context information, independently of whether there is a slight or a dramatic change with respect to the most recent model deployment. In other words, reframing is not triggered when there is a significant context change, but rather applied systematically for any context (changed or not).

Finally, the description of a context can be accompanied by a distribution across $\mathbb{C}$. For instance, instead of saying that the deployment context is $\theta = \langle 3 \rangle$, we may say that we know that it is distributed as $\theta \sim \mathcal{N}(3,1)$. As a practical example [36], the assumption that cost proportions in binary classification tasks are uniformly distributed on the interval $[0,1]$ leads to a useful reinterpretation of common evaluation measures. We return to this issue in Section 5.

3. Context-aware approaches for machine learning

In this section we discuss different approaches to deal with the variety of contexts described above. We discuss the choice between retraining, revision and reframing, and the notion of a versatile model. The section includes examples of these different approaches and ends with hybrid cases crossing the boundaries of one single approach.

We assume that in each context there is a task (possibly different for each context) that consists in providing certain estimates given some input data. We denote the type of input as $\mathbb{X}$ and the type of estimates as $\mathbb{Y}$. While this definition applies to both supervised and unsupervised learning, we will focus on supervised learning examples in the following.

A context-blind approach would be to learn a model $m : \mathbb{X} \to \mathbb{Y}$ from all available training data and use this model unaltered in any deployment context. In cases where context matters, this approach will result in decreased performance whenever it fails to capture variations in context. It is nevertheless surprisingly common in machine learning: for example, classifiers are often used without taking the deployment class distribution into account. This approach can be considered as a baseline.

Perhaps the simplest kind of context-aware setting arises when the context is encoded as a feature value. This allows a decision tree, for example, to split on the context feature and to construct context-specific submodels below that split. For this to work we need sufficient training data covering a wide range of training contexts, which may be prohibitive in some situations. Whether model reuse takes place at all depends on where the context feature is used: the lower this is in the tree, the more of the model is shared across contexts. Conversely, if the context is used at the root we obtain a set of unrelated context-specific models.

The *context-as-a-feature* approach may be blind to the global influence of the context on other features or the data distribution. Also, machine learning techniques may be unable to process that information effectively. For instance, it is unusual to treat cost as an extra input feature. Instead, we will analyse below three approaches where context is considered as key information that affects the whole problem.

3.1. Retraining

There may be no need to create all context-specific models at once, as in the case of decision trees above using the context as a feature. Rather, we could create them in an on-demand fashion. That is, each time we need to adapt to a new context we collect sufficient training data for that context and build a new context-specific model. We refer to this approach as *retraining*, and it provides a second baseline to compare against. There are two different subtypes:

- **Retraining on the training data** assumes availability of original training data during deployment. It postpones learning until all information about the deployment context has arrived. This approach is standard in transfer learning [63,51] and enables the use of training data in any way beneficial for the particular deployment context.
- **Retraining on the deployment data** assumes availability of sufficient deployment data to train a new model for the particular deployment context.

In both cases, there can be knowledge reuse, such as the use of the optimal parameters or some parts of the original models from some previous training situations.

3.2. Revision

Retraining a model again and again whenever something changes is often inefficient, especially if the context change is small and the retrained model is similar to the original one. A common alternative to retraining is *model revision*, where parts of the model are patched or extended according to a new context [55,54].

Model revision is particularly appropriate when there is a concept drift [29], but it can deal with any of the other types of context changes discussed in the previous section. It is particularly natural as a result of incremental learning [39] or lifelong learning [61], but can also be used in other cases of domain adaptation when there is a mapping between two different domains [48].

A key issue in model revision is the detection of novelty or inconsistency of the new data with respect to the existing model, as in the area of theory revision [55]. In this case, the revision of a theory is triggered when the semantics of the model are affected, as it does not fully accommodate the new evidence. This can be extended to context changes, provided we can determine when the context has changed significantly to deserve a revision process.

Whether revision is a viable option depends on the model class: rules and linear models are easier to revise than neural networks and support vector machines.

3.3. Reframing and versatile models

Reframing is a context-aware approach that reuses a model m built in the training context by subjecting it to a reframing procedure that takes into account the particular deployment context. As elaborated in the next section, we distinguish three different kinds of reframing (which can be combined):

- **Output reframing:** the model m is applied unaltered on the input data of the deployment context, but the outputs of the model are transformed to fit the deployment context better;
- **Input reframing:** the input data of the deployment context are pre-processed before applying the model m, the outputs of which are used without modifications;
- **Structural reframing:** the structure of the model m is adapted in some way, e. g. only some part of the model is applied or the model is transformed or instantiated in some systematic way taking the context into account.

Which type of reframing should be used depends on what aspects of the model are reusable in other contexts. If these aspects are known in advance, then it is possible to design a training procedure which results in a *versatile model*: a model capturing the reusable knowledge. Thus, where a conventional, non-versatile model captures only such information as is necessary to deal with test instances from the same context, a versatile model captures additional information that, in combination with reframing, allows it to deal with test instances from a larger range of contexts.

Note that a versatile model might have a different data signature for input and output than the original task signatures $\mathbb{X}$ and $\mathbb{Y}$. Input and/or output reframing is then required to apply the versatile model in any particular deployment context. Furthermore, a versatile model can be constructed directly from the training data or can be *enriched* from a non-versatile model in a series of transformations. One straightforward illustration of this idea is a calibrated probability estimation tree which can be obtained from a crisp decision tree by means of a calibration process. Being a versatile model, the probability estimation tree has a different signature than the crisp tree, mapping instances into probabilities rather than classes. Output reframing is thus required by means of a decision rule which applies a threshold to the estimated probabilities.

In a way, most decision rules and feature construction procedures can be seen as reframing processes, provided that the context is used in the transformations. In supervised tasks, generative models estimating $p(X,Y)$ or $p(X|Y)$ are more versatile than discriminative models $p(Y|X)$. Here, output reframing takes the form of conditioning on X. On occasions, $p(X)$ and/or $p(Y)$ can be considered the context of the problem.

Versatile models can be composed of submodels, which may be combined in different ways depending on the context.

3.4. Choosing the best approach: Examples

Given the alternatives described above, which one is best? There is no general answer, as this may depend on the problem and the kind of context. Also, we can use more than one approach or a hybrid. Nevertheless, we can give some guidelines.

Retraining on the training data is very general and popular, because it is easy and applicable to any model class, but there are many cases where it is not applicable. For instance, the training data may have been

lost or may not exist (e.g., models have been created or modified by human experts) or may be prohibitively large (if deployment must work in restricted hardware), or the computational constraints do not allow retraining for each deployment context separately. Retraining on the deployment data can work well if there is an abundance of deployment data, but often the deployment data are limited, unsupervised or simply do not exist. For instance, in *open set recognition* [57] not all classes are present in the training data and the model has to be versatile enough to cover for classes that will appear in the deployment dataset.

Revision is usually a complex process that is highly dependent on the technique that one is using. Of course, there are cases between retraining and revision where it is difficult to draw a line, as when all parameters of a Bayesian model are acquired (and not only adjusted) on new data. Model revision changes the semantics of the model, and this does not correspond well with some types of contexts. For instance, if we have a cost change or a task representation change, it is not really the semantics that has to be modified, but the way the model is applied, which is the approach taken by reframing.

Reframing, if it is possible, appears to be the most efficient approach, because versatile models are learnt once and reused systematically, as we will see in the following sections. However, there are cases where the context changes cannot be anticipated, parametrised or inferred. It may also be hard to design a versatile model to deal with particular context changes.

Mixtures of retraining, revision and reframing are also possible. For instance, in a rule-based model, a set of conflicting rules might be resolved a priori [44] but also when the context is available. In inductive logic programming [50], a theory that adapts to changing contexts by modifying the background knowledge is not really a *model* revision, as the background knowledge is what changes. If the modification is performed systematically by the selection or weighting of some of the predicates in the background knowledge that are used for each context, this can also be considered a case of reframing.

An ensemble of models can appropriately adapt their parameters and structures according to the new incoming context. A number of strategies to updating an ensemble have been explored [56]. For instance, weights could be trained using the context from deployment data using *stacking* [67]. In this case, this would be seen as a mixture of reframing (the ensemble is reused) and retraining (the top layer that makes the final decision is retrained).

4. Kinds of reframing

In this section we detail the three kinds of reframing introduced in the previous section. We will use a signature-based process-oriented notation to describe reframing processes as functions, as different types of reframing are characterised by what arguments they take and how some processes are nested.

Definition 2. *Reframing is a function with the following signature:*

$$R : \mathbb{X} \times \mathbb{D} \times \mathbb{C} \times \mathbb{M} \to \mathbb{Y} \tag{1}$$

Given deployment data $X \in \mathbb{X}$ under context $\theta \in \mathbb{C}$ possibly using some additional data $D_a \in \mathbb{D}$, $R(X, D_a, \theta, m)$ uses model $m \in \mathbb{M}$ to output estimated data $Y \in \mathbb{Y}$ as the result of the reframing process.

The kinds of reframing we propose are summarised in Figure 2, and fully described below.

4.1. Output reframing

One of the main motivations of the reframing approach is to enable model reuse as much as possible in different contexts. In some cases, the model can be reused completely, without modifying it at all. There are two advantages of not modifying the model: first, the validation of the model can be preserved and, second, the model can be treated as a black box, without reference to whether it is a decision tree, a neural network or a support vector machine. In such cases we can just work on its inputs or its outputs.

In supervised tasks, one way of using the operating context is by modifying the output of the model as a post-process or *decision rule*. In this case, R can be expressed as follows:

$$\mathsf{R}(X, D_a, \theta, m) = \mathsf{R}_O(m(X), D_a, \theta) \tag{2}$$

This is achieved through a reframing function $\mathsf{R}_O : \mathbb{Z} \times \mathbb{D} \times \mathbb{C} \to \mathbb{Y}$ that takes the outputs of the model and performs some post-processing on them. We use $\mathbb{Z}$ instead of $\mathbb{Y}$ because the model can be a versatile model that outputs richer information, such as scores, ranks, distributions, etc. The process of output reframing is represented in Figure 2(a). In binary and multi-label classification, this approach is known as *thresholding* when the model outputs scores or probabilities. For instance, a context-driven decision rule known as *score-driven threshold choice method* [36] simply checks whether the probability estimation is lower or higher than the cost proportion, as a threshold. In this

case D_a is not needed. A related way of doing this is by calibrating the outputs taking the new context into account. In this case, some D_a may be necessary to do this calibration or to estimate, e.g., the class distribution.

Similarly, in regression, if the operating context is the parameter of the asymmetric loss [32], we can just obtain the prediction as usual (i.e., $m(X)$) and then R_O will just add or subtract a constant such that the expected loss for a given operating context is minimised. In this case, the process is referred to as *shifting*. A versatile model can be developed for cost-sensitive regression too [33]. With more traditional regression models, reframing is also possible. For instance, *tuning* [5] is consistent with the most general view of the function $\mathsf{R}(X, D_a, \theta, m)$, where D_a is exactly the training dataset D_t. In other words, the training data are *preserved* and used during the reframing process. In this case, when the deployment context is available, an optimal shift is derived (for the training set) and applied to the deployment dataset (added to or subtracted from all predictions). A constant shift can be extended to any polynomial transformation of the output variable [69], provided the loss functions is convex.

Other kinds of operating contexts have been investigated in supervised problems. For instance, reject rules [11] make it possible for classification or regression models to abstain on some examples, as the prediction error cost might be higher than not issuing a prediction. One option is to take the reject option as a class and (re)train accordingly whenever the costs of abstention or wrong predictions change. Alternatively, one can generate models that can be later reframed for different operating contexts in terms of reject rules. This has been investigated as extensions of ROC analysis, abstaining, cautious or reliable classifiers [64,52,66]. Again the idea is to exploit a soft model (or another kind of more versatile model) to determine when to issue predictions and when to abstain.

4.2. Input reframing

Another way of using the operating context is by modifying the inputs of the model, leading to the following signature:

$$\mathsf{R}(X, D_a, \theta, m) = m(\mathsf{R}_I(X, D_a, \theta)) \tag{3}$$

where the signature of the input reframing is now $\mathsf{R}_I : \mathbb{X} \times \mathbb{D} \times \mathbb{C} \to \mathbb{X}$. Now this is done through a transformation of the input space, but the model is not changed. This is why this approach can also be called a feature-transformation or transformation-based approach. The process of input reframing is shown in Figure 2(b). We assume that the reframed input X' has the same signature as X so that the original model can be applied to the reframed inputs.

To illustrate, input attribute values can be shifted from source to deployment [3]. A model is trained in source and deployed over several different contexts by transforming the input attributes to the appropriate values using only few labelled deployment data. Consider a simple scenario of building a classifier using training data taken from City 1 where most of the people buy an ice-cream when the temperature is higher than $18°C$. On the other hand, the same event happens in City 2 when the temperature is higher than $25°C$. Now if these data are rescaled by subtracting a numeric value of 7, we can easily use the trained classifier for City 1 without any modification and be able to predict whether a person in City 2 will buy an ice-cream or not. In this case, the context change can be simply defined as a shift of one input feature, and reframing is solved by adding or subtracting the shift before applying the classifier.

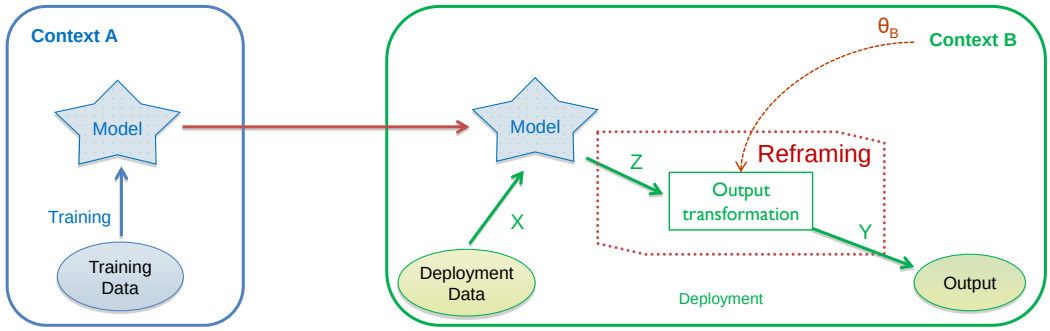
As another example, the input data can be transformed into a normalised or a discretised version [17], using the data distribution. Note that in this case we apply the transformation to the input features both during training and during testing. The model is learnt from quantiles, and patterns and rules are expressed in terms of these distribution quantiles, such as buying an ice-cream when 'the temperature is higher than 90% of the days'. We could have encapsulated the training feature distribution and do this in one step only during deployment time, with a mapping between the values for each attribute. In this case, the model would be learnt from the original attributes but it would still require the training data (or some distribution summary).

4.3. Structural reframing

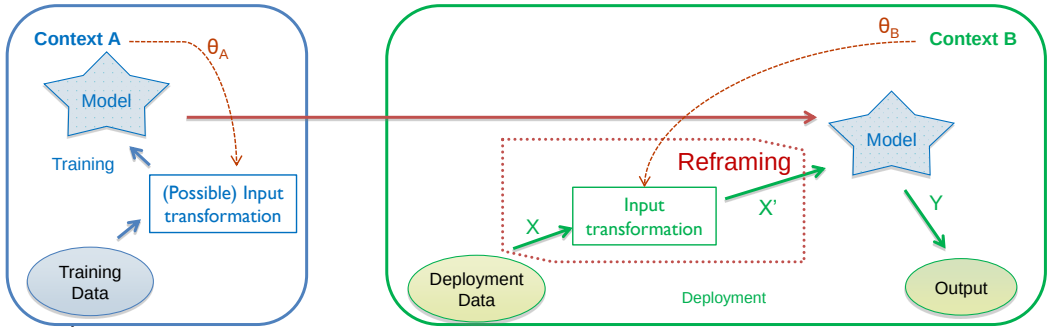
The most elaborate form of reframing occurs when the context is used to modify part of the model, such as changing some labelling rules or other systematic changes. The model is no longer treated as a black box. This is represented as follows.

$$\mathsf{R}(X, D_a, \theta, m) = m'(X) \tag{4}$$

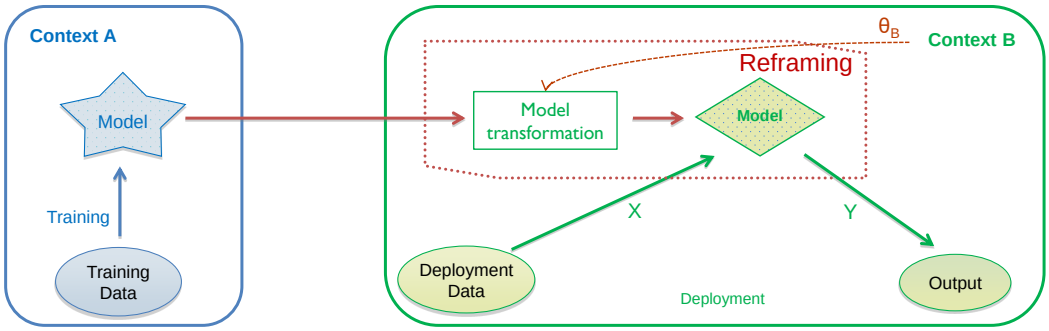
with $m' = \mathsf{R}_S(X, D_a, \theta, m)$, where the signature of structural reframing is $\mathsf{R}_S : \mathbb{X} \times \mathbb{D} \times \mathbb{C} \times \mathbb{M} \to \mathbb{M}$. The outcome is another model that results from making some structural changes to the original model, such as



(a) Output reframing, e.g. thresholding, tuning, shifting



(b) Input reframing, e.g. feature transformation



(c) Structural reframing, e.g. model adaptation and relabelling

Fig. 2. Different kinds of reframing

relabelling. The process of structural reframing is represented in Figure 2(c) where the original model undergoes structural changes to better fit the deployment data. For instance, an ensemble method can change its weights or a classification tree can be post-pruned in many different ways depending on the deployment context, such as a change in a cost matrix.

It is worth highlighting the difference between structural reframing and model revision. In structural reframing the same model is used again and again, although adapted or instantiated for each context. The notion of versatile model is key here, as the model does not need to be patched or extended incrementally as a consequence of its deployment to different contexts.

For instance, consider the cyber fraud detection problem with online bank transaction defined in [58]: "new clients (banks) operate in different contexts [...], where the type of fraud committed might differ from the generic frauds, due to variations in transaction protocols, geo-demographics and other factors". This problem motivated two new techniques for classification, *structure expansion reduction* (SER) and *structure transfer* (STRUT), which "refine" one or more decision trees for each context. The generic model is kept and refined systematically for each new context. SER may expand (grow more splits, i.e., specialise)

Fig. 3. Dimension hierarchies, Time (left) and Product (right).

the branches of the trees or reduce (prune, i.e., generalise). Hence the expansion part of SER is a revision approach. On the other hand, STRUT can be considered a structural reframing technique (although the context is not fully characterised). With STRUT, the thresholds (the actual value for the inequality for each numeric split) of the generic tree are removed and replaced by new thresholds (the structure and attributes used at each node are preserved). To avoid overfitting, for those cases where the set of labelled examples for deployment is very small, the generic tree preserves the distributions at the inner splits and the new distributions for the new context are ensured to be "similar" to the generic ones, using several divergence metrics.

The three kinds of reframing identified here can be combined in many different ways. For instance, in multidimensional aggregation by means of a data cube [46] both the input variables and the output values are aggregated depending on the operating context (the data cube). Multidimensional approaches are based on hierarchies, and examples and predictions can be aggregated at different levels of the attribute hierarchies, such as the ones shown in Figure 3. For instance, the predictions for tomatoes and weeks will be different from the predictions for vegetables and Fridays. However, machine learning models are not designed to take hierarchical attributes. In principle, a model that has been obtained for one context cannot be *directly* applied to a different context. This leads us to two major alternatives. Either we learn one model for each context (level of aggregation), which means *retraining* the model, or we learn one, more versatile, model at the highest resolution (most fine-grained) level and then aggregate their predictions (that is, *reframing*), as described in Figure 4.

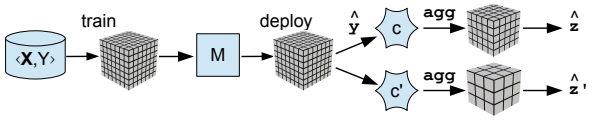


Fig. 4. Reframing schema for two multidimensional contexts c and c'. Training data is used just once at the highest resolution level to create a single model M that is applied to different operating contexts c or c' by aggregating the outputs appropriately.

5. Context plots and performance metrics

When the context is fixed, conventional context-insensitive performance metrics (see, e.g., [22]) can be used to evaluate how a model performs for that context. However, when we use the same model for several contexts we need context-aware performance metrics. In this section we discuss how to obtain such metrics and how they can be represented and related to the so-called *context plots*.

5.1. Evaluating a model on a range of contexts

Let us consider how a model (either reframed or not) can be evaluated for a given context. This is represented by a loss function $Q(R,m,D,\theta)$ which returns the loss or cost of applying a procedure R with model m to dataset D with deployment context information θ. Also, in some reframing procedures, an additional dataset may also be needed (which is considered the same for all methods, and hence does not appear as a parameter here). Here we assume that the context information that is given (θ) is correct.

Definition 3. *A model is said to dominate another in a region $\rho \subset \mathbb{C}$ if Q is equal or lower for all $\theta \in \rho$.*

We can examine the whole range of operating contexts and see in which regions one model is better than others. However, as dominance may only occur for partial regions, we may wish to have an overall metric accounting for the whole distribution of contexts.

Definition 4. *The expected loss of a model m using reframing procedure R over data D and a distribution of contexts w over the space of contexts $\mathbb{C}$ is given by:*

$$L(R,m,D,\mathbb{C},w) \triangleq \int_{\theta\in\mathbb{C}} Q(R,m,D,\theta)w(\theta)d\theta \quad (5)$$

If θ contains discrete parameters only, this can be substituted by a weighted sum instead of the integral. Note that we can only talk about a versatile model if it can be successfully reframed (even if not optimally) for a range of contexts. This generalises the approach of [35,37], which reinterprets several known performance metrics previously considered context-insensitive, such as AUC and Brier score, as aggregates over contexts.

Expected loss as defined above requires the estimation of w, the distribution of contexts. Default choices are possible but can be controversial. For instance, in classification there is a long-standing debate about whether the distribution of skews or the cost proportion must be considered uniform or can be modelled with other distributions (for instance, other beta distributions) [31,26].

5.2. Context plots

The visualisation of how the loss Q changes for a range of operating contexts is a context plot:

Definition 5. *A context plot shows Q as a function of θ for a given dataset D.*

The dimensionality of the plot depends on the number of parameters in θ, ranging from simple one-dimensional curves to complex hypersurfaces. We can plot different models m or different reframing techniques R on the same loss-context space. If we assume $w(\theta)$ uniform, then L is the area (or hypervolume) under the context curve (or hypersurface), a performance metric that aggregates over contexts.

When the context plot has more than two dimensions it can be convenient to represent the plot with a mapping or projection onto a different space $\mathbb{C}'$, with fewer parameters. The simpler representation comes at the cost of some information loss.

Definition 6. *A context reduction is a mapping from $\mathbb{C}$ to a different parameter space $\mathbb{C}'$ (usually with fewer dimensions).*

Context plots and context distributions can be extended with context reductions instead of contexts. Context reductions can also be used as transformations to yield a more representative spread on the x-axis, such as applying a logarithmic scale or applying the distribution w over the x-axis [34].

5.3. Context plots for binary classification

In binary classification, one choice for the context can be the cost proportion $\theta = \langle c \rangle$, where c is a single parameter derived from a cost matrix. A closely related type of context is the class proportion. In fact, if we have varying cost matrices and class proportions, they can be integrated (or *reduced*) into a single parameter, known as skew, as in ROC analysis.

Let us focus for the moment on the simple case where the context is just $\theta = \langle c \rangle$ and is given. On the one hand, if we have a learning technique that can use c during training (e.g., a cost-sensitive classifier), we could repeatedly retrain m for every θ. On the other hand, we can learn a *versatile* model once and for all, a scoring classifier or a probabilistic classifier, and use the so-called *threshold choice methods* as the usual reframing approach. There are several possible threshold choice methods. All of them, in our terminology, are *output reframing* methods and hence can be expressed as in Eq. (2). The most traditional options for R are the "score-fixed" method, the "score-driven" method and the "rate-driven" method. But other methods exist [36].

We can plot $Q(R,m,D,\theta)$ in terms of θ, for different models m and threshold choice methods R. If we use the optimal threshold choice method ($R_{optimal}$), these are the well known cost curves [15]. Other curves result if we use other threshold choice (reframing) methods, such as the Brier curves, the rate-driven curves, etc. [35,37]. Further curves arise from using a (cost-sensitive) retraining approach or a technique that ignores the context altogether.

Figure 5 illustrates 4 instances with scores (0.85, 0.65, 0.45, 0.25) and actual classes $(+,-,+,-)$. On the left we can observe the cost lines for each possible split together with the lower envelope, which indicates that the optimal split would be after the first positive for $c \in [0,0.5]$ and before the last negative for $c \in [0.5,1]$. On the right we show four different cost curves obtained by four threshold selection methods. From these curves we can see, for example, that score-driven and rate-driven curves have different regions of dominance but neither completely dominates the other. These context plots can be extended to cost-sensitive multilabel classification, for instance by averaging the costs for each label to obtain the context parameter [45].

5.4. Context plots for regression

Costs are also common in regression. One simple case is the asymmetric loss, where over-estimates and under-estimates have different costs. This can be modelled by a parameter $\alpha \in [0,1]$ such that $\alpha > 1/2$ means that under-estimates are more costly and $\alpha < 1/2$ that over-estimates are more costly [32]:

Definition 7. *The asymmetric absolute error ℓ^A_α is a loss function defined as follows:*

$$\ell^A_\alpha(\hat{y},y) \triangleq \begin{cases} 2\alpha(y-\hat{y}) & \text{if } \hat{y} < y \\ 2(1-\alpha)(\hat{y}-y) & \text{otherwise} \end{cases}$$

By adding or subtracting a constant value to all predictions we can get better results when α changes, by minimising expected loss. This is another case of output reframing, where the reframing process is expressed again as in Eq. (2).

The loss function here is given by

$$Q = \sum_{\langle x,y \rangle \in D} \ell^A_\alpha(R_O(m(x),\theta),y) \tag{6}$$

which gives the total ℓ^A_α for dataset D for a given model m and a context $\theta = \langle \alpha \rangle$. By plotting α from 0 to 1 on

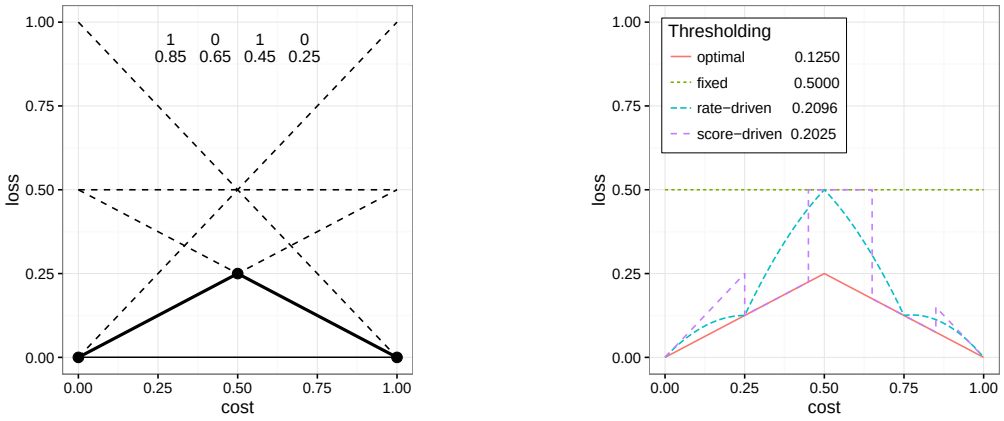


Fig. 5. Context plots for binary cost-sensitive classification, where output reframing occurs by means of different threshold choice methods. Left: Cost lines of a simple example. Right: Cost curves obtained by four different threshold choice methods.

the x-axis we obtain a context plot, which can be used to analyse the behaviour of different models. The area under this context plot (i.e., the expected loss assuming uniform α) is linearly related to the error variance of the regression model [32].

Note that the above cost-sensitive context plots for binary classification and regression can be derived analytically. For instance, in the context plots shown in Figure 5, given a classifier, we can calculate all curves in the figure analytically, as they all correspond to an *output reframing*, and the context-dependent prediction (e.g., a class being predicted after setting a threshold) can be used in Q to compare with the true labels using the context. In this way, given a versatile model (e.g., a probabilistic classifier), we can calculate the area under each curve analytically, without the need of going context by context [36]. Similarly, for regression using cost asymmetry as shown in Figure 6, the curves can be plotted analytically and their areas can be derived from the model itself.

However, in many other cases the context plots cannot be derived analytically. For instance, the context can be defined as a trade-off between misclassification cost (MC) and attribute test cost (TC) with a parameter $\alpha \in [0,1]$, such that the joint cost (JC) is defined as the weighted average $Q = \alpha MC + (1-\alpha)TC$ [42]. Several classifiers can be represented in terms of TC and MC, as in the JC plots of Figure 7, which are derived empirically, point by point. More examples of empirical context plots are given in the next section.

An interesting future direction is to consider 'process costs', that is, to use a performance metric that takes into account the cost of retraining/reframing, having too many models (in ensemble approaches),

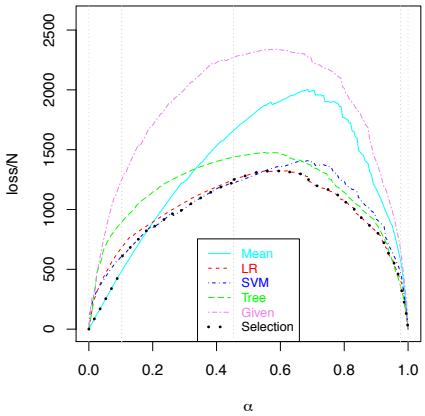


Fig. 6. Context plot for cost-sensitive regression with α a context parameter trading off between over- and under-estimation penalties, and output reframing through shifting.

parametrising the context, etc. Process cost analysis is a problem of increasing importance and the integration of these budgets into machine learning is an active research area [68].

5.5. *Further examples of context plots*

Examples of applications where we should probably consider different contexts during deployment are relatively common. Below we show more cases of context parametrisations, which have been (or could have been) represented as a context plot.

Some of the contexts seen in previous sections may be composed of several parameters (e.g., the coefficient values in some input reframing approaches [3], so θ has twice as many parameters as attributes). In the

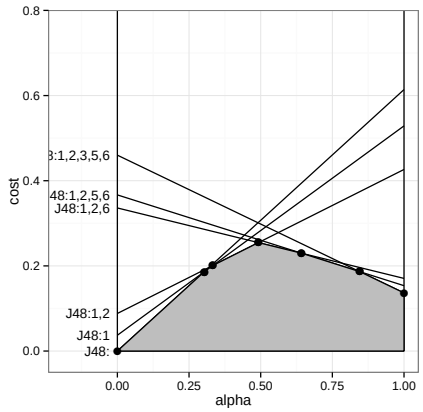


Fig. 7. Context plot for cost-sensitive classification with attribute costs and misclassification costs, with α a context parameter trading off between the two.

multidimensional datamart example [46] the operating context is an OLAP cube, and θ is not only composed of many parameters but they are also discrete. These cases lead to more complex context plots, but some representations or simplifications may still be insightful.

Regression Error Characteristic (REC) curves [8] plot the error tolerance on the x-axis versus the percentage of points predicted within the tolerance on the y-axis. Error tolerance may be context-dependent, and in this sense REC curves are another example of context plots.

Like the joint cost case described in the previous section and shown in Figure 7, most of the context plots in this section are empirical rather than analytic and need to be constructed through experiments. In the input shift example [3] the context affects the input attributes, leading to an output change through the model. As the relation between the inputs and outputs is model-dependent, it is not possible, in general, to derive the curves analytically. This means that we have to sample over operating contexts in order to plot these curves and estimate its area to obtain the expected loss (Definition 4). A similar situation happens where the context is some kind of noise that affects all input attributes, such as the measurement error in several sensors given by temperature [21]. Finally, in the multidimensional contexts example, both a context change in inputs/outputs and an output reframing is required. An analytical derivation of these curves (and their areas) is not possible in general.

6. Reframing and related approaches in the literature

The notion of context or domain adaptation has been studied before in machine learning and more generally in computer science. There are several areas that overlap with the notion of reframing under context changes, although for most of them the focus on anticipation and versatile models are absent.

Data shift [53,49,41] is a kind of context change, but most existing approaches do not exploit systematic shifts. Consequently, they often invoke retraining, but, as a kind of context change, it could be addressed with many of the approaches seen in this section.

Domain adaptation [38] is a more general term than data shift, also including "changes of representation" [7]. The terms 'domain' and 'context' are often used interchangeably, but domains are usually more open-ended and less prone to parametrisation than the notion of context seen here.

Transfer learning [63,51] is "the improvement of learning in a new task through the transfer of knowledge from a related task that has already been learned" [63]. Knowledge is transferred from source to target task in the form of "instance-transfer", "feature-representation-transfer", "parameter-transfer" and "relational-knowledge-transfer" but the model is retrained [51]. There are different kinds of transfer learning: inductive, transductive and unsupervised. Even in the transductive case, which is the case most similar to reframing, transfer learning approaches focus on the target task and reuse knowledge rather than models, as in reframing. Table 2 summarises the differences between reframing and transfer learning. In model transfer, such as in the so-called biased regularisation approach, a model learned in a domain is reused, as in reframing. However, a new model is yet again retrained using the previous parameters of the generic model plus a small set of examples of the new domain [40,43,62]. Alternatively, a new specific model and the generic one are aggregated. Also, a combination of approaches can be used [58]. Nevertheless, differently from reframing, no a priori characterisation of context is performed.

Transportability [6] is related to transfer learning and "aims to identify conditions under which causal information learnt from experiments can be reused in a different environment where only passive observations can be collected".

Meta-learning [30] is a broad area that indicates which techniques can be better for a given situation according to previous situations. Techniques, parameters

Reframing	Transfer Learning
The same general, versatile model is reused.	A new model is generated, reusing old parts (e.g., features).
The model and the reframing procedure support different contexts.	If context changes, the model is changed (but knowledge is reused).
An operating context must be known at deployment time.	No explicit target operating context.
Changes are anticipated and evaluation takes this into account	The transfer is usually triggered on purpose for the target context*.
Oriented towards context representation.	Oriented towards the target task.
Automated, given the deployment operating context.	Usually requires manual intervention*.
Success depends on model versatility and the reframing procedures.	Success depends on a good mapping between source and target.

Table 2
Comparing reframing and (transductive) transfer learning. Asterisks refer to exceptions for multi-task learning.

and other kinds of (meta-)knowledge are reused, but models are usually not kept but retrained.

Multi-task learning [10,60] learns several tasks with the same representation. It is closely related to inductive transfer, but we usually find that many tasks are simultaneously learnt. In other words, there is a generic model that is built from several tasks, usually by transforming or augmenting the feature space into a more general representation (e.g., using subspaces described by eigenvectors [20]), or by coding general and target-specific versions of the features [12]. However, despite its apparent similarity with input reframing, and the extraction of invariants across domains or the use of common parameters across domains [14], these do not really represent a context.

Cost-sensitive learning [18,65] is usually solved by retraining using cost-sensitive methods, but sometimes by output reframing. Reject rules [11,64,52,66] and other kinds of decision rules based on utility or loss functions are usually associated with cost-sensitive learning as well.

Learning from noisy data [4,27]: usually the emphasis is put on how difficult the problem is when noise is present during training or test, but noise is not considered a parametrisable context where we could say a priori that some models are better or worse for some noise contexts, or on procedures to make them better when we expect more or less noise during deployment.

Context-aware computing [1], as encountered in pattern recognition [13] or recommender systems [2], uses contextual information to improve a task. The approach is more like the consideration of the context as an extra input in the process and not in terms of reusing or adapting an existing model to a new context.

Mimetic models [9]: the adaptation of existing models to other contexts (e.g., cost contexts) has been carried out with the use of the mimetic technique, which retrains another model using some artificially generated data that takes the new context into account. However, this approach is different to reframing, as a model is retrained for each operating context.

Theory revision [55] establishes a set of tools to modify a model or theory because it has become inconsistent or insufficient with new evidence. Model revision can be used for domain adaptation and transfer learning problems [48]. As the model is reused as much as possible, theory revision is related to structural reframing. However, the notion of a parametrised context to reframe the model instead of revising it is not present.

Lifelong learning [61] and incremental learning [39]: adaptation is usually achieved through theory revision or the extension of the model. The emphasis is then put on "versatile learning systems" [16] that evolve and adapt their model or knowledge base instead of versatile models, which remain unaltered across previously-anticipated context changes.

ROC analysis and cost plots [47,25,19,23,15,26,35,36,37]: the notion of context in this paper generalises the notion of operating condition in ROC analysis. Similarly, context plots are a generalisation of cost plots, and we also borrowed the ideas of dominance and expected loss as the area under a context curve.

Only cost-sensitive learning and ROC analysis (when there is no retraining) can be seen as areas where reframing has been commonly used in the past, and generally restricted to binary classification. However, many other tasks and applications, especially with the use of input and structural reframing, present themselves as new opportunities.

In the context of these related areas, we are now in a position to highlight the distinctive characteristics of reframing:

- Contexts are identified and parametrised. Models are learnt *in anticipation* of context changes and optimised to behave well in a range of contexts.
- We do not consider a 1-to-1 transfer from a source problem to a target problem, but a systematic ap-

plication to multiple contexts. Once the reframing procedure is set, reframing is automated for any deployment data given the context.
- Even if models are intended to be general and versatile, they are usually learnt in one context and task. It is through the parametrisation of context – a kind of 'inductive context bias' – that they are expected to be adapted to other contexts.
- Models are compared with the notion of dominance for ranges of contexts. Expected performance can be plotted for a range of contexts. Aggregated metrics can be derived.
- Models are reused. The model is versatile enough to be adapted to several contexts, which avoids the cost of retraining or revision.

There are scenarios where contexts cannot be identified or parametrised, or where the model is not versatile enough. Also, it may happen that we evaluate a versatile model together with a particular reframing mechanism for a distribution of contexts, but have no clear dominance regions so that we have to keep several models or select the one that is best for the expected distribution of contexts. It may turn out that this distribution is different to the one that we finally observe in a deployment setting.

The notion of context parametrisation and its application to reframing seems to be better suited for non-interactive machine learning (supervised, semi-supervised or unsupervised) when there is a true model, but needs to be rethought for other areas of machine learning, such as instance-based learning, and particularly in domain adaptation and transfer learning in reinforcement learning [59].

While it is important to highlight these limitations, this paper has shown that there are many more cases where reframing is possible and useful than just those traditional cases concerning cost-sensitive learning and prior distribution shift already analysed in the case of binary classification using ROC analysis, cost curves and parametrised decision rules.

7. Concluding remarks

This paper has provided a unified view of a family of solutions for a set of context-change problems under the term reframing. We have generalised the types of context changes in Table 1, clarified the three types of context-aware adaptations in Section 3 and defined a taxonomy of the types of reframing.

In addition to providing common terminology and notation, we hope this paper also enables a better understanding of the commonalities and differences in problems and solutions, as well as identifying 'niches' where some of the techniques discussed could be applied or adapted.

Taking a higher-level view, we think that the notion of reframing may have a deeper and more long-term impact in how the process from data to knowledge, its validation and its application to real problems can be conceived. We finish by highlighting the key points underlying this view.

Models should be as general and flexible as possible. The data and conditions used to learn the model may change for each particular application. Versatile models integrate more information than originally needed, such as probabilities, distributions, covariance information, unpruned or alternative rules, etc.

Validation should consider a range of operating contexts, either by analytically integrating them into the validation process, simulating them with data modification or by the use of data from different situations. Furthermore, validation has to take into account that repeatedly learning for each particular application has the risk of overfitting to the operating context.

Related to the previous point, *performance metrics that account for a range of situations* instead of more short-sighted metrics that only account for one operating context should be the base for a more comprehensive model evaluation.

Learning and assessing models has a cost, so we should always *consider the overall costs involved in retraining, revising and reframing*. The possibilities and cost of properly identifying the operating context (and when this information will be available) must be considered during the whole process.

Model deployment is a very important stage. Machine learning and data mining applications are not finished when a good model is obtained. The process from data to models is just one part of the game, as the ultimate goal is high-quality decision making. Taking full advantage of a learnt model may require complex decision processes that consider all available information at deployment time.

Acknowledgments

We thank the anonymous reviewers for their comments, which have helped to improve this paper significantly. This work was supported by the REFRAME project, granted by the European Co-

ordinated Research on Long-term Challenges in Information and Communication Sciences Technologies ERA-Net (CHIST-ERA), funded by their respective national funding agencies in the UK, France and Spain (MINECO, PCIN-2013-037). It has also been partially supported by the EU (FEDER) and Spanish MINECO grant TIN2015-69175-C4-1-R and by Generalitat Valenciana PROMETEOII/2015/013.

References

[1] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, and P. Steggles. Towards a better understanding of context and context-awareness. In *Handheld and ubiquitous computing*, pages 304–307. Springer, 1999.

[2] G. Adomavicius and A. Tuzhilin. Context-aware recommender systems. In *Recommender systems handbook*, pages 217–253. Springer, 2011.

[3] C. F. Ahmed, N. Lachiche, C. Charnay, and A. Braud. Reframing continuous input attributes. In *IEEE Intl. Conf. on Tools with Artificial Intelligence (ICTAI-2014)*, pages 31–38, 2014.

[4] D. Angluin and P. Laird. Learning from noisy examples. *Machine Learning*, 2(4):343–370, 1988.

[5] G. Bansal, A. Sinha, and H. Zhao. Tuning data mining methods for cost-sensitive regression: A study in loan charge-off forecasting. *J. Management Inf. Systems*, 25:315–336, 2008.

[6] E. Bareinboim and J. Pearl. Transportability of causal effects: Completeness results. In *AAAI*, 2012.

[7] S. Ben-David, J. Blitzer, K. Crammer, F. Pereira, et al. Analysis of representations for domain adaptation. *Advances in neural information processing systems*, 19:137, 2007.

[8] J. Bi and K. P. Bennett. Regression error characteristic curves. In *ICML*, 2003.

[9] R. Blanco-Vega, C. Ferri, J. Hernández-Orallo, and M. J. Ramírez-Quintana. Estimating the class probability threshold without training data. In *ICML'06 workshop on ROC Analysis in Machine Learning*, page 9, 2006.

[10] R. Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, 1997.

[11] C Chow. On optimum recognition error and reject tradeoff. *IEEE T. Information Theory*, 16(1):41–46, 1970.

[12] Hal Daumé III. Frustratingly easy domain adaptation. In *ACL*, page 256, 2007.

[13] M. Davis, M. Smith, J. Canny, N. Good, S. King, and R. Janakiraman. Towards context-aware face recognition. In *ACM international conference on Multimedia*, pages 483–486, 2005.

[14] M. Dredze, A. Kulesza, and K. Crammer. Multi-domain learning by confidence-weighted parameter combination. *Machine Learning*, 79(1-2):123–149, 2010.

[15] C. Drummond and R.C. Holte. Cost Curves: An Improved Method for Visualizing Classifier Performance. *Machine Learning*, 65:95–130, 2006.

[16] E. Eaton and P. L. Ruvolo. Ella: An efficient lifelong learning algorithm. In *ICML*, pages 507–515, 2013.

[17] S. El Jelali, A. Braud, and N. Lachiche. Propositionalisation of continuous attributes beyond simple aggregation. In *Inductive Logic Programming*, pages 32–44. Springer, 2013.

[18] C. Elkan. The foundations of Cost-Sensitive learning. In *IJCAI*, pages 973–978, 2001.

[19] T. Fawcett. An introduction to ROC analysis. *Pattern recognition letters*, 27(8):861–874, 2006.

[20] B. Fernando, A. Habrard, M. Sebban, and T. Tuytelaars. Unsupervised visual domain adaptation using subspace alignment. In *ICCV*, pages 2960–2967. IEEE, 2013.

[21] C. Ferri, J. Hernández-Orallo, A. Martínez-Usó, and M.J. Ramírez-Quintana. Identifying dominant models when the noise context is known. In *ECML'14 workshop on Learning over Multiple Contexts*, 2014.

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

[23] P. Flach. ROC analysis. In *Encyclopedia of Machine Learning*, pages 869–875. Springer, 2010.

[24] P. Flach. Classification in context: Adapting to changes in class and cost distribution. In *ECML'14 workshop on Learning over Multiple Contexts*, 2014.

[25] P. Flach, H. Blockeel, C. Ferri, J. Hernández-Orallo, and J. Struyf. Decision support for data mining. In *Data Mining and Decision Support*, pages 81–90. Springer, 2003.

[26] P. Flach, J. Hernández-Orallo, and C. Ferri. A coherent interpretation of AUC as a measure of aggregated classification performance. In *ICML*, 2011.

[27] B. Frénay and M. Verleysen. Classification in the presence of label noise: a survey. *IEEE T. Neural Networks and Learning Systems*, 25(5), 2013.

[28] B. Frenay and M. Verleysen. Classification in the presence of label noise: A survey. *IEEE T. Neural Networks and Learning Systems*, 25(5):845–869, May 2014.

[29] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia. A survey on concept drift adaptation. *ACM Comput. Surv.*, 46(4):44:1–44:37, 2014.

[30] C. Giraud-Carrier, R. Vilalta, and P. Brazdil. Introduction to the special issue on meta-learning. *Machine learning*, 54(3):187–193, 2004.

[31] D.J. Hand. Measuring classifier performance: a coherent alternative to the area under the ROC curve. *Machine learning*, 77(1):103–123, 2009.

[32] J. Hernández-Orallo. ROC curves for regression. *Pattern Recognition*, 46(12):3395–3411, 2013.

[33] J. Hernández-Orallo. Probabilistic reframing for cost-sensitive regression. *ACM T. Knowledge Discovery from Data*, 8(4):17, 2014.

[34] J. Hernández-Orallo, C. Ferri, N. Lachiche, A. Martínez-Usó, and M.J. Ramírez-Quintana. Binarised regression tasks: Methods and evaluation metrics. *Data Mining and Knowledge Discovery*, 30(4):848–890, 2016.

[35] J. Hernández-Orallo, P. Flach, and C. Ferri. Brier curves: a new cost-based visualisation of classifier performance. In *ICML*, 2011.

[36] J. Hernández-Orallo, P. Flach, and C. Ferri. A unified view of performance metrics: Translating threshold choice into expected classification loss. *J. Machine Learning Researh*, 13:2813–2869, 2012.

[37] J. Hernández-Orallo, P. Flach, and C. Ferri. ROC curves in cost space. *Machine Learning*, 93(1):71–91, 2013.

[38] J. Jiang. A literature survey on domain adaptation of statistical classifiers, 2008. http://sifaka.cs.uiuc.edu/jiang4/domain_adaptation/survey/da_survey.pdf.

[39] W. Khreich, E. Granger, A. Miri, and R. Sabourin. A survey of techniques for incremental learning of HMM parameters. *Information Sciences*, 197:105–130, 2012.

[40] W. Kienzle and K. Chellapilla. Personalized handwriting recognition via biased regularization. In *ICML*, pages 457–464, 2006.

[41] M. Kull and P. Flach. Patterns of dataset shift. In *ECML'14 workshop on Learning over Multiple Contexts*, 2014.

[42] M. Kull and J. Hernández-Orallo. Missing values on purpose: Model selection and reframing with attribute and prediction costs. *submitted*, 2015.

[43] I. Kuzborskij and F. Orabona. Stability and hypothesis transfer learning. In *ICML*, pages 942–950, 2013.

[44] T. Lindgren. Methods for rule conflict resolution. In *ECML*, pages 262–273. Springer, 2004.

[45] H.-Y. Lo, J.-C. Wang, H.-M. Wang, and S.-D. Lin. Cost-sensitive multi-label learning for audio tag annotation and retrieval. *IEEE T. Multimedia*, 13(3):518–529, 2011.

[46] A. Martínez-Usó and J. Hernández-Orallo. Multidimensional prediction models when the resolution context changes. In *ECML*, pages 509–524. Springer, 2015.

[47] C. E. Metz. Basic principles of ROC analysis. *Seminars in nuclear medicine*, 8,4:283–298, 1978.

[48] L. Mihalkova, T. Huynh, and R. J. Mooney. Mapping and revising Markov logic networks for transfer learning. In *AAAI*, pages 608–614, 2007.

[49] J. G. Moreno-Torres, T. Raeder, R. Alaiz-Rodríguez, N. V. Chawla, and F. Herrera. A unifying view on dataset shift in classification. *Pattern Recognition*, 45(1):521–530, 2012.

[50] S. Muggleton and L. De Raedt. Inductive logic programming: Theory and methods. *J. Logic Programming*, 19:629–679, 1994.

[51] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE T. Knowledge and Data Engineering*, 22(10):1345–1359, 2010.

[52] T. Pietraszek. On the use of ROC analysis for the optimization of abstaining classifiers. *Machine Learning*, 68(2):137–169, 2007.

[53] J. Quiñonero-Candela, M. Sugiyama, A. Schwaighofer, and N. D. Lawrence. *Dataset shift in machine learning*. The MIT Press, 2009.

[54] L. De Raedt. *Interactive Theory Revision: An Inductive Logic Programming Approach*. Academic Press, 1992.

[55] B. L. Richards and R. J. Mooney. First-order theory revision. In *ICML*, pages 447–451, 1991.

[56] D. Sannen, E. Lughofer, and H. Van Brussel. Towards incremental classifier fusion. *Intell. Data Anal.*, 14(1):3–30, 2010.

[57] W. J. Scheirer, A. de Rezende-Rocha, A. Sapkota, and T. E. Boult. Toward open set recognition. *IEEE T. Pattern Analysis and Machine Intelligence*, 35(7):1757–1772, 2013.

[58] N. Segev, M. Harel, S. Mannor, K. Crammer, and R. El-Yaniv. Learn on source, refine on target: A model transfer learning framework with random forests. *arXiv preprint arXiv:1511.01258*, 2015.

[59] M. E. Taylor and P. Stone. Transfer learning for reinforcement learning domains: A survey. *J. Machine Learning Research*, 10:1633–1685, 2009.

[60] S. Thrun. Is learning the n-th thing any easier than learning the first? *Advances in neural information processing systems*, pages 640–646, 1996.

[61] S. Thrun and L. Pratt. *Learning to learn*. Springer Science & Business Media, 2012.

[62] T. Tommasi, F. Orabona, and B. Caputo. Learning categories from few examples with multi model knowledge transfer. *IEEE T. Pattern Analysis and Machine Intelligence*, 36(5):928–941, 2014.

[63] L. Torrey and J. Shavlik. Transfer learning. *Handbook of Research on Machine Learning Applications*, 3:17–35, 2009.

[64] F. Tortorella. A ROC-based reject rule for dichotomizers. *Pattern Recognition Letters*, 26(2):167–180, 2005.

[65] P. Turney. Types of cost in inductive concept learning. *Canada National Research Council Publications Archive*, 2000.

[66] S. Vanderlooy, I. Sprinkhuizen-Kuyper, and E. Smirnov. An analysis of reliable classifiers through ROC isometrics. In *ICML'06 workshop on ROC Analysis in Machine Learning*, pages 55–62, 2006.

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

[68] Z. Xu, M. J. Kusner, K. Q. Weinberger, M. Chen, and O. Chapelle. Classifier cascades and trees for minimizing feature evaluation cost. *J. Machine Learning Research*, 15:2113–2144, 2014.

[69] H. Zhao, A. P. Sinha, and G. Bansal. An extended tuning method for cost-sensitive regression and forecasting. *Decision Support Systems*, 2011.