UNIVERSITAT POLITÈCNICA DE VALÈNCIA
DEPARTAMENT DE SISTEMES INFORMÀTICS I COMPUTACIÓ

UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA

Bernoulli HMMs for Handwritten Text Recognition

Thesis
presented by Adrià Giménez Pastor
supervised by Dr. Alfons Juan Císcar and Dr. Jesús Andrés Ferrer

May 26, 2014

Bernoulli HMMs for Handwritten Text Recognition

Adrià Giménez Pastor

Thesis performed under the supervision of doctors
Jesús Andrés Ferrer and Alfons Juan Císcar
and presented at the Universitat Politècnica de València
in partial fulfilment of the
of the requirements for the degree
Doctor en Informàtica

València, May 26, 2014

Work supported by the EC (FEDER) and the Spanish MEC under the MIPRCV “Consolider Ingenio 2010” research programme (CSD2007-00018), the iTransDoc research project (TIN2006-15694-CO2-01), and the FPU grant AP2005-1840.

ABSTRACT

In last years Hidden Markov Models (HMMs) have received significant attention in the task off-line handwritten text recognition (HTR). As in automatic speech recognition (ASR), HMMs are used to model the probability of an observation sequence, given its corresponding text transcription. However, in contrast to what happens in ASR, in HTR there is no standard set of local features being used by most of the proposed systems. In this thesis we propose the use of raw binary pixels as features, in conjunction with models that deal more directly with the binary data. In particular, we propose the use of Bernoulli HMMs (BHMMs), that is, conventional HMMs in which Gaussian (mixture) distributions have been replaced by Bernoulli (mixture) probability functions. The objective is twofold: on the one hand, this allows us to better modeling the binary nature of text images (foreground/background) using BHMMs. On the other hand, this guarantees that no discriminative information is filtered out during feature extraction (most HTR available datasets can be easily binarized without a relevant loss of information).

In this thesis, all the HMM theory required to develop a HMM based HTR toolkit is reviewed and adapted to the case of BHMMs. Specifically, we begin by defining a simple classifier based on BHMMs with Bernoulli probability functions at the states, and we end with an embedded Bernoulli mixture HMM recognizer for continuous HTR. Regarding the binary features, we propose a simple binary feature extraction process without significant loss of information. All input images are scaled and binarized, in order to easily reinterpret them as sequences of binary feature vectors. Two extensions are proposed to this basic feature extraction method: the use of a sliding window in order to better capture the context, and a repositioning method in order to better deal with vertical distortions. Competitive results were obtained when BHMMs and proposed methods were applied to well-known HTR databases. In particular, we ranked first at the *Arabic Handwriting Recognition Competition* organized during the *12th International Conference on Frontiers in Handwriting Recognition* (ICFHR 2010), and at the *Arabic Recognition Competition: Multi-font Multi-size Digitally Represented Text* organized during the *11th International Conference on Document Analysis and Recognition* (ICDAR 2011).

In the last part of this thesis we propose a method for training BHMM classifiers using

discriminative training criteria, instead of the conventional Maximum Likelihood Estimation (MLE). Specifically, we propose a log-linear classifier for binary data based on the BHMM classifier. Parameter estimation of this model can be carried out using discriminative training criteria for log-linear models. In particular, we show the formulae for several MMI based criteria. Finally, we prove the equivalence between both classifiers, hence, discriminative training of a BHMM classifier can be carried out by obtaining its equivalent log-linear classifier. Reported results show that discriminative BHMMs clearly outperform conventional generative BHMMs.

RESUM

En els últims anys els models ocults de Markov (HMMs) han tingut un paper destacat en el camp del reconeixement de text manuscrit off-line (HTR). Al igual que ocorre en el reconeixement automàtic de la parla (ASR), els HMMs s'empren per a modelar la probabilitat d'una seqüència d'observacions donada la seua transcripció. En canvi, a l'inrevés del que ocorre en ASR, en HTR no hi ha un conjunt de característiques estàndard que estiga sent utilitzat per la majoria de sistemes existents. En aquesta tesis proposem utilitzar directament com a característiques els píxels originals binaritzats, conjuntament amb models específicament dissenyats per a tractar amb dades binàries. Concretament, proposem l'ús de Bernoulli HMMs (BHMMs), es a dir, HMMs convencionals on les distribucions de (mixtures de) gaussianes són reemplaçades per (mixtures de) funcions de probabilitat de Bernoulli. L'objectiu és doble: per una banda açò ens permet modelitzar la naturalesa binària de les imatges amb text (lletres/fons) utilitzant BHMMs. Per una altra banda, açò garanteix que no és perdrà informació discriminant en el procés d'extracció de característiques (la majoria de les base de dades d'HTR disponibles poden ser fàcilment binaritzades sense una pèrdua rellevant d'informació).

Tota la teoria d'HMMs necessària per a desenvolupar un toolkit basat en HMMs es revisada i adaptada per al cas dels BHMMs. Concretament, comencem definint un classificador senzill basat en BHMMs amb funcions de probabilitat de Bernoulli als estats, i acabem amb un reconeixedor de HTR continuu amb mixtures de Bernoulli embegudes. Respecte a les característiques, proposem un procés senzill d'extracció de característiques binàries amb poca pèrdua d'informació. Totes les imatges són escalades i binaritzades per a què es pugen reinterpretar fàcilment com a seqüències de vectors de característiques binàries. Es proposen dos extensions a aquest mètode bàsic d'extracció de característiques: l'ús d'una finestra relliscant per a capturar millor el context, i un mètode de reposicionament per a tractar millor les distorsions verticals. S'han obtingut resultats competitius quan els BHMMs i els mètodes proposats s'han aplicat a bases de dades ben conegudes en el àmbit del HTR. En particular, hem quedat primers en la competició *Arabic Handwriting Recognition Competition* organitzada en el *12th International Conference on Frontiers in Handwriting Recognition* (ICFHR 2010), i en la competició *Arabic Recognition Competition: Multi-font Multi-size Digitally*

Represented Text organitzada en el *11th International Conference on Document Analysis and Recognition* (ICDAR 2011).

En la part final d'aquesta tesis proposem un mètode per a entrenar classificadors basats en BHMMs fent ús de criteris d'entrenament discriminatius, en comptes del criteri Maximum Likelihood Estimation (MLE). Concretament, comencem proposant un classificador log-lineal per a dades binàries basat en un classificador de BHMMs. L'estimació dels paràmetres d'aquest model es pot dur a terme utilitzant qualsevol criteri d'entrenament discriminatiu per a models log-lineals. En particular, presentem les fórmules per diversos criteris basats en el criteri MMI. Per a acabar, demostrem l'equivalència entre ambdós classificadors, i per tant, demostrem que l'entrenament discriminatiu d'un classificador de BHMMs por dur-se a terme obtenint el model log-lineal equivalent. Els resultats reportats demostren clarament que els BHMMs discriminatius milloren als BHMMs convencionals.

RESUMEN

En los últimos años los modelos ocultos de Markov (HMMs) han tenido un papel destacado en el campo del reconocimiento de texto manuscrito off-line (HTR). Al igual que ocurre en el reconocimiento automático del habla (ASR), los HMMs se utilizan para modelar la probabilidad de una secuencia de observaciones dada su transcripción. En cambio, al contrario de lo que ocurre en ASR, en HTR no hay un conjunto de características estándar que esté siendo utilizado por la mayoría de sistemas existentes. En esta tesis proponemos utilizar directamente como características los píxeles originales binarizados, conjuntamente con modelos específicamente diseñados para tratar con datos binarios. Concretamente, proponemos el uso de Bernoulli HMMs (BHMMs), es decir, HMMs convencionales donde las distribuciones de (mixturas de) gaussianas son reemplazadas por (mixturas de) funciones de probabilidad de Bernoulli. El objetivo es doble: por un lado esto nos permite modelizar la naturaleza binaria de las imágenes con texto (letras/fuentes) utilizando BHMMs. Por otro lado, esto garantiza que no se perderá información discriminante en el proceso de extracción de características (la mayoría de las bases de datos de HTR disponibles pueden ser fácilmente binarizadas sin una pérdida relevante de información).

En esta tesis, toda la teoría de HMMs necesaria para desarrollar un toolkit basado en HMMs es revisada y adaptada para el caso de los BHMMs. Concretamente, empezamos definiendo un clasificador sencillo basado en BHMMs con funciones de probabilidad de Bernoulli en los estados, y acabamos con un reconocedor de HTR continuo con mixturas de Bernoulli embebidas. Respecto a las características, proponemos un proceso sencillo de extracción de características binarias con poca pérdida de información. Todas las imágenes son escaladas y binarizadas para que se puedan reinterpretar fácilmente como secuencias de vectores de características binarias. Se proponen dos extensiones a este método básico de extracción de características: el uso de una ventana deslizante para capturar mejor el contexto, y un método de reposicionamiento para tratar mejor las distorsiones verticales. Se han obtenido resultados competitivos cuando los BHMMs y los métodos propuestos se han aplicado a bases de datos conocidas en el ámbito del HTR. En particular, hemos quedado primeros en la competición *Arabic Handwriting Recognition Competition* organizada en la *12th International Conference on Frontiers in Handwriting Recognition* (ICFHR 2010), y en

la competición *Arabic Recognition Competition: Multi-font Multi-size Digitally Represented Text* organizada en la *11th International Conference on Document Analysis and Recognition* (ICDAR 2011).

En la parte final de esta tesis proponemos un método para entrenar clasificadores basados en BHMMs haciendo uso de criterios de entrenamiento discriminativos, en vez del criterio Maximum Likelihood Estimation (MLE). Concretamente, empezamos proponiendo un clasificador log-lineal para datos binarios basado en un clasificador de BHMMs. La estimación de los parámetros de este modelo se puede hacer utilizando cualquier criterio de entrenamiento discriminativo para modelos log-lineales. En particular, presentamos las fórmulas para diversos criterios basados en el criterio MMI. Para acabar, demostramos la equivalencia entre ambos clasificadores, y por lo tanto, demostramos que el entrenamiento discriminativo de un clasificador de BHMMs puede llevarse a cabo obteniendo el modelo log-lineal equivalente. Los resultados reportados demuestran claramente que los BHMMs discriminativos mejoran a los BHMMs convencionales.

CONTENTS

CHAPTER 1

INTRODUCTION

Contents

Automatic Handwritten Text Recognition (HTR) is a research field that aims at developing automatic computer systems which take as input an image containing handwritten text, and obtains as output the transcription of the text contained in that image.

One of the most popular approaches in HTR is the use of Hidden Markov Models (HMM). In fact, HMMs are used in HTR in a very similar way as they have been used in the past in Automatic Speech Recognition (ASR), where HMMs are the standard approach. As in ASR, an input image (observations) is transformed into a sequence of fixed dimension real feature vectors. This sequence is transcribed using language models and HMMs for real data, that is, HMMs in which the emission probability models real data, for instance Gaussian mixture distributions. However, while in ASR the feature extraction process is quite standard, in HTR there is not any kind of feature extraction process which could be considered standard for HMMs.

A peculiarity of HTR is that text images have a binary nature, that is, the unique relevant information provided by a pixel in order to recognize text is to discern if that pixel belongs to the foreground (text) or the background. In many HTR tasks, images can be easily binarized using a simply binarization technique, as the Otsu's method, without a relevant loss of information required for their recognition. Binary data can be better modeled using probabilistic models that deal more directly with binary data. A well known model is the Bernoulli mixture distributions. Bernoulli mixture distributions have been successfully used in several woks involving input binary data, for example Saeed and Babri [2008]. An example more related to HTR is Juan and Vidal [2004], in which a Bernoulli mixtures classifier is used to classify isolated handwritten Indian digits. Using a similar motivation than in Juan and Vidal [2004], in this thesis we propose and analyze the use of Bernoulli HMMs for HTR.

1.1 Scientific Goals

The goals addressed in this thesis are summarized in the following points:

Propose an HMM based on Bernoulli mixtures: The first goal of this thesis is the definition of an HMM based on Bernoulli mixtures (Bernoulli HMM). The basic idea is to define a conventional HMM model in which mixture Gaussian distributions are replaced by Bernoulli mixture probability functions.

Review of the HMM formulae using Bernoulli HMM: There are several HMM toolkits available for research, the most famous is the HTK [Young et al., 1995]. However, all of them only implement conventional HMMs for continuous data. For this reason we decided to implement a specific software for Bernoulli HMMs. In order to develop this software we need to review the HMM formulae related to recognition and parameter estimation: embedded HMMs, parameter estimation by means of Baum-Welch algorithm, recognition using the Viterbi algorithm, pruning techniques, etc. In fact, we think that this document could also be used, at some extent, as a review of HMMs.

Propose a binary feature extraction process for HTR: The motivation beyond the use of Bernoulli HMMs is to directly recognize binarized input images. As was shown in Juan and Vidal [2004], good results could be obtained, in isolated digit recognition, by directly using binarized input images as feature vectors, that is, all images are resized to

the same size (width $\times$ height) and binarized, treating each pixel as a binary feature. In fact, this is one of the main motivations of this thesis. However, because here we are working with text sequences of variable length, using only one binary feature vector per image does not seem to be a very good approach. A more reasonable representation will be a binary feature vector sequence. Therefore, an important goal of this thesis is to determine a proper transformation of input images into sequences of binary feature vectors, in which values are directly related to binarized pixels of the input image.

Evaluate the proposed methods on well-known HTR corpora: We plan to test Bernoulli HMMs, and the proposed feature extraction techniques, on well-known HTR corpora. Specifically, we are particularly interested on apply Bernoulli HMM to the *IAM* database of English handwritten text, and the *IFN/ENIT* database of Arabic handwritten text. Both are very well-known databases which have been used during last years to compare HTR systems.

Discriminative Bernoulli HMMs: In last years, some progress has been achieved in ASR by using discriminative training techniques to estimate HMM parameters instead of using the conventional Baum-Welch algorithm. Our last goal in this thesis is to apply some of these techniques to Bernoulli HMMs in order to improve the performance of Bernoulli HMMs on HTR.

1.2 Document Structure

Regarding the structure, the thesis is organized as follows. In Chapter 2 some concepts, which are not the focus of this thesis but necessary, are defined: definition of HTR, language models, error measures, etc. In Chapter 3 we introduce all the databases used in this thesis. In the following three chapters we develop and test BHMMs using the conventional training criterion for HMMs (MLE). In particular, basic BHMMs are introduced in Chapter 4, and in Chapter 5 we extend basic BHMMs to Embedded Bernoulli mixture HMMS for isolated handwritten word recognition. The use of BHMMs in conjunction to language models to general HTR is explained in Chapter 6. The last two chapters introduce the discriminative training for BHMMs. We begin by developing a MMI training scheme for a simple Bernoulli mixture classifier in Chapter 7. Then, we extend that training scheme to a BHMM classifier in Chapter 8. Finally, conclusions are discussed in Chapter 9.

Bibliography

A. Juan and E. Vidal. Bernoulli mixture models for binary images. In *Proc. of ICPR 2004*, volume 3, Cambridge (UK), August 2004.

M. Saeed and H. Babri. Classifiers based on bernoulli mixture models for text mining and handwriting recognition tasks. In *Neural Networks, 2008. IJCNN 2008. (IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*, pages 2169–2175, 2008. doi: 10.1109/IJCNN.2008.4634097.

S. Young et al. *The HTK Book*. Cambridge University Engineering Department, 1995.

CHAPTER 2

PRELIMINARIES

Contents

2.1 Handwritten Text Recognition

The aim of Handwritten Text Recognition (HTR) is to develop computer systems which take as input an image containing handwritten text, and obtain as output the transcription of the text contained in that image. That is, computer systems that, in some sense, are able to emulate the human ability to read.

It is worth noting that despite isolated character recognition can be considered as a solved problem, or near to be, for several kinds of scripts ([Ciresan et al., 2010, Mozaffari and Soltanizadeh, 2009, Romero et al., 2007]), the HTR problem is still far to be considered as a solved problem. Conventional Optical Character Recognition (OCR) systems are not suitable to the HTR task, because they are designed to deal with isolated characters, and the task of segmenting handwritten text before its recognition is often very difficult. Thus, while HTR systems have to deal with unsegmented sequences of characters and can be used to perform isolated character recognition, the field of application of conventional OCR systems is reduced to printed text recognition, where characters can be easily segmented, and some special cases of handwritten text recognition in which writers have been required to write isolated characters.

Most of current HTR systems raise the recognition of handwritten text as a two step process. In the first step the text lines or words, depending on the system, are segmented and extracted, while in the second step each resulting image is transcribed. The first step involves layout analysis and image processing techniques. In this first step a system must deal with some of the tasks described below (a detailed description of the techniques discussed here can be found in Pastor i Gadea [2007]):

Thresholding and background removal: Thresholding has as objective to classify pixels into background and foreground, in order to separate as much as possible the background, which contains irrelevant and noisy information, from the foreground, which contains the information related to the text.

Noise removal: It is common that images contain some random variation of brightness or color information produced during the acquisition process. This useless information is appointed as noise. Solutions for the noise removal problem involve techniques related to filters and mathematical morphology.

Skew correction: Due to the image acquisition process, which is usually carried out by hand using a camera or scanner, images usually appear skewed. This phenomenon has a significant impact in the segmentation and extraction of lines or words. One simple technique used to solve this problem is to find the rotation angle which when applied the resulting rotated image optimizes some criterion.

Block or field extraction: In addition to the text, documents also contain images and blank areas. There are documents where most of the content is either printed or handwritten text, whereas there are other documents which are mostly graphics as diagrams, maps, etc. Thus an important problem in text recognition is to detect which parts of the image are text and extract them. This problem is commonly referred to Layout Analysis, and involves techniques such as connected components, RLSA, and others.

Line and word segmentation: Once blocks of text have been detected and extracted they must be segmented into lines. A common approach used to achieve this is the use of horizontal projections. In this technique projection valleys, that is lines with few black pixels, are expected to be inter-line blank areas, while peaks are expected to be the text line bodies. In some systems and additional word segmentation step is done, using similar techniques than those applied to segment lines.

Despite that the HTR problem entails obtaining a transcription from a document image, nowadays most HTR researchers focus their efforts on the second step. In fact, almost all HTR databases provide the document images and the segmentation into lines or words.

The first approaches to HTR were based on existing conventional OCR systems. In these approaches word images were segmented into characters and then recognized. As said before this is a very difficult task, thus what was really done in these cases is to consider many segmentations, recognize them all and try to get the correct recognition with the aid of external sources, such as language models or rules [Breuel, 1994a,b]. Nevertheless, in the last years these uncoupled approaches have been replaced by approaches in which the segmentation and the recognition are carried out at the same time. These approaches usually follow a three module scheme as described in Figure 2.1: preprocess, feature extraction and recognition.

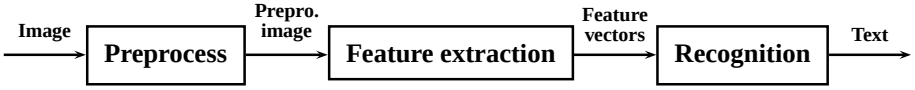


Figure 2.1: Typical scheme of a HTR system for segmented images

The aim of the preprocess module is to normalize the input image in order to facilitate the recognition. There is no standard concerning the structure and functions of the preprocess module. However, most of the preprocess modules of existing HTR systems carry out, to a greater or lesser degree, the following tasks:

Brightness and color normalization: All images are normalized in order to obtain similar color and brightness values for shapes, and similar color and brightness for background. When input images are gray level ones this is carried out usually by means of contrast normalization or using thresholding techniques.

Slant correction: Text slanting is a handicap for most part of recognizers, not only due to the fact that different writing styles can have different slanting, but due to the fact that letters appear vertically overlapped, making difficult the segmentation problem.

Size normalization: This normalization tries to achieve the same size of a letter in all input images, so vertically as horizontally. Vertical normalization usually requires from a previous detection of the text ascenders, text descenders and text body.

In the feature extraction module images from the preprocess module are converted into a sequence of feature vectors which are fed into the recognizer module. Finally, the recognition module obtains the most suitable transcription for the image. This recognition module is usually implemented as an statistical recognizer which has been automatically trained from labeled data. That is,

$$W^* = \underset{W}{\operatorname{argmax}}\ p(W \mid \mathbf{O}), \tag{2.1}$$

where W^* is the most probable transcription and $\mathbf{O}$ is the sequence of feature vectors. Depending on whether W is a sequence of words or a word, we will talk about isolated word recognition or about continuous text recognition.

The use of Hidden Markov models (HMMs) for HTR have been consolidate in last years, see for example Günter and Bunke [2004], Plamondon and Srihari [2000], Toselli et al. [2004], Xue and Govindaraju [2006]. The technology is used in similar way as it is used in Speech Recognition, where HMMs have been the standard technology for years, see Rabiner and Juang [1993]. A relevant difference is that in Speech Recognition input is a one-dimensional signal in the time axis, while in HTR the input signal is a bidimensional one in the plane. Therefore, during feature extraction input is usually sampled over the horizontal axis at some frequency, obtaining a one-dimensional signal. This transformation is the main reason to explain the importance of preprocess in HMM handwritten text recognizers. However, there are more benefits than disadvantages in the HMM technology: well-known algorithms for training and recognizing, existence of mature public available software, and easily integration with language models. When HMMs are used (2.1) is rewritten using the Bayes' theorem as follows

$$W^* = \underset{W}{\operatorname{argmax}}\ p(W)p(\mathbf{O} \mid W), \tag{2.2}$$

where $p(W)$ and $p(\mathbf{O} \mid W)$ are usually treated as two independent models which are integrated during the search process. The first term is usually referred to as the language model, while the second term is usually referred to as the acoustic model (visual or character model in HTR), which is implemented using HMMs. In continuous text recognition the language model is modeled using n-grams or other finite state models, while in isolated word recognition language model is modeled using a table of prior probabilities, one for each word (class), as in traditional statistical classifiers. HMMs could be used to model each word of the considered vocabulary, or at character level. Details about how HMMs are used to model $p(\mathbf{O} \mid W)$ will be discussed in detail along this thesis.

2.2 Preprocess Methods

As commented above in the previous section HTR systems usually preprocess input images before the feature extraction process. In the experiments carried in this thesis we do not have always used the same preprocess, moreover, in some experiments no preprocess was applied at all. In this section we present the preprocess techniques which have been used at some point in this thesis.

2.2.1 Slant Correction

The used slant correction method consists of two steps. In the first step we calculate the slant angle, and in the second step we apply a shear operation over the input image. Given a slant angle α the shear operation is defined as

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \text{shear}_\alpha \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x + y\tan(-\alpha) \\ y \end{pmatrix}. \tag{2.3}$$

In order to obtain the slant angle α, we try different slant angles and choose that angle that over the deslanted input image maximizes some objective function. In particular we select that angle that maximizes the standard deviation of the vertical projections

$$\alpha^* = \underset{\alpha \in [45:135]}{\operatorname{argmax}} \sqrt{\sum_{m=1}^{\text{cols}} \frac{(\mu - v(\text{shear}_\alpha(m)))^2}{\text{cols}}}\,, \tag{2.4}$$

where $v(\text{shear}_\alpha(m))$ is the vertical projection in column m of the input image after applying a shear operation using α as slant angle, μ is the average of the vertical projection, and $cols$ is the number of columns in the vertical projection. The vertical projection is calculated as

$$v(x) = \sum_y f(x, y)\,. \tag{2.5}$$

A more detailed explanation of this method is available in Pastor i Gadea [2007].

2.2.2 Vertical Size Normalization

Two vertical size normalization techniques have been used in this thesis. In the first technique, input images are first horizontal segmented using blank spaces. So, each segment is not related to a word but a sequence of shapes with a small separation between them. The main objective of this segmentation is to detect the text body at segment level instead of globally. Each segment is then smoothed using the *Run-Length Smoothing Algorithm* (RLSA). This smoothing is applied in order to better obtain the upper and lower outlines of each segment. Finally, approximating the upper and lower outlines with straight lines the text body for each segment is detected. Once, the text body has been detected for each segment it is mandatory to resize them to the same height. This height is calculated as the mean (plus five pixels) of all detected text body heights in the input image. A more detailed explanation of this method is available in Pastor i Gadea [2007].

The second technique is a more elaborated method based on a Neural Network classifier. Unfortunately, we do not have access to the software based on Neural Networks and we only report results in one corpus in which we had access to the preprocessed images. The details related to this technique can be found in España-Boquera et al. [2011], Gorbe-Moya et al. [2008].

2.2.3 Otsu's Method

The Otsu's method is a very well known binarization technique, which belongs to the family of global thresholding binarization techniques. In a global thresholding technique a thresholding value is obtained in some way for each input gray level image, and then all pixels greater than that threshold are taken as white pixels, and the remaining are taken as black pixels.

In the Otsu's method the threshold value for each input image is calculated by maximizing inter-class variance of gray values and minimizing intra-class variance of gray values. In more detail, for a given threshold T we split all gray level values in two classes: the gray

values greater than T and the gray values smaller than T. Then, we calculate the mean and probabilities for each class as follows

$$p_1 = \sum_{g=0}^{T} h_g \,, \qquad p_2 = \sum_{g=T+1}^{L-1} h_g = 1 - p_1 \,, \tag{2.6}$$

$$\mu_1 = \frac{1}{p_1} \sum_{g=0}^{T} g h_g \,, \qquad \mu_2 = \frac{1}{p_2} \sum_{g=T+1}^{L-1} g h_g \,, \tag{2.7}$$

where L denotes the greatest gray level value and h is the normalized histogram of the given input image. Finally, the thresholding value is selected as follows

$$T^* = \underset{T}{\operatorname{argmax}}\; p_1 p_2 (\mu_1 - \mu_2)^2 \,. \tag{2.8}$$

The Otsu's method has been used as binarization method in all BHMMs experiments of this thesis. The choice of the Otsu's method was done due to its robustness, speed and performance. Some preliminary work was done related to the choice of the binarization method, but apparently the impact on the final recognition of the binarization method used is small. It is worth noting that the databases used in the experimentation, as we will see in the next chapter, are not very difficult to binarize.

2.3 *n*-gram Models

The use of *n*-gram models is a common approach in pattern recognition for modeling symbol sequences for a given finite alphabet. Thus, given a sequence of symbols w_1^T of length T, in a *n*-gram model the probability for that sequence is calculated using the Markov chain approximation

$$p(w_1^T) = p(w_1) \prod_{t>1} p(w_t \mid w_1^{t-1}) p(\$ \mid w_1^T), \tag{2.9}$$

where $\$$ is a special symbol used to indicate the end of the sequence. In a n-gram model the probabilities of (2.9) are converted into parameters. However, the order of the number of parameters needed is upper bounded by $T_{max} \cdot W$, being T_{max} the length of the largest sequence considered and being W the size of the alphabet. This is obviously a huge and impractical amount of parameters. In the *n*-gram approach this problem is solved assuming that the probability of a word does not depends on its position and only depends on the $n-1$ previous words, hence the *n* in the name. So, taking into account this assumption (2.9) can be rewritten as

$$p(w_1^T) = p(w_1) \prod_{t>1} p(w_t \mid w_{\max\{t-n+1,1\}}^{t-1}) p(\$ \mid w_{\max\{T-n+1,1\}}^{T}), \tag{2.10}$$

and if we assume that all sequences implicitly include special symbols <s> and </s> in positions 0 and $T+1$ respectively, indicating start and end of sequence, then the expression can be simplified as

$$p(w_1^T) = \prod_{t=1}^{T+1} p(w_t \mid w_{\max\{t-n+1,0\}}^{t-1}). \tag{2.11}$$

Then, the number of parameters needed in a n-gram model is upper bounded by W^n, but trends to the Zipf's law which it is typically smaller.

Despite its simplicity, the n-gram model is nowadays the most wide-spread model used for language modeling. Besides from being used in HTR, it is used in machine translation, speech recognition and human language technologies [Goodman, 2001]. Its success lies in a very good tradeoff between the time required during recognition and the error rates obtained. In addition, the parameter estimation can be easily carried out from a training set using the Maximum Likelihood Estimation (MLE), since no hidden variable is required in a *n*-gram model. According to MLE, $p(w \mid h)$, the probability of word w given a history h, can be estimated as

$$p(w \mid h) = \frac{N(w,h)}{N(h)}, \tag{2.12}$$

where $N(w,h)$ and $N(h)$ are the concurrences in the training set of $h \cdot w$ and h respectively. However, this estimation gives zero probability to all unseen events, which is an important drawback since the number of parameters (W^n) is often larger than the different events in the training set. This problem is solved by smoothing the model, that is, modifying the original probability distribution in order to obtain a similar distribution but without zero probabilities.

There are several smoothing techniques available for *n*-gram models. In fact, much of the current researching effort on language modeling is spent developing new smoothing techniques for *n*-gram models. The most common techniques are the interpolation technique and the back-off one. In the interpolation approach probabilities are smoothed

$$\tilde{p}(w \mid h) = \lambda_{w,h} p(w \mid h) + B_h \beta(w \mid h), \tag{2.13}$$

where $\tilde{p}(w \mid h)$ denotes the smoothed probability, $\lambda_{w,h}$ is a factor used to discount mass probability from the original distribution, B_h is the total amount of discounted probability given history h, that is,

$$B_h = 1 - \sum_w \lambda_{w,h} p(w \mid h), \tag{2.14}$$

and finally $\beta(w \mid h)$ is a smoothed more simple language model, usually this smoothed model is a *(n-1)*-gram model [Chen and Goodman, 1996].

While in the interpolation approach discounted probability is distributed over all events, in the back-off approach it is only distributed over non seen events as follows

$$\tilde{p}(w \mid h) = \begin{cases} \lambda_{w,h} p(w \mid h) & N(w,h) \neq 0 \\ B_h \hat{\beta}(w \mid h) & N(w,h) = 0 \end{cases}, \tag{2.15}$$

where $\hat{\beta}(w \mid h)$ is the distribution $\beta(w \mid h)$ normalized over all unseen events, that is

$$\hat{\beta}(w \mid h) = \frac{\beta(w \mid h)}{\sum_{w:N(w,h)=0} \beta(w \mid h)}. \tag{2.16}$$

There is another approach which is a modification of the back-off approach. In this approach discounted probability is obtained only from these events which has been seen less than a

given threshold s [Katz, 1987].

$$\tilde{p}(w \mid h) = \begin{cases} p(w \mid h) & N(w,h) \geq s \\ \lambda_{w,h} p(w \mid h) & 0 < N(w,h) < s \\ B_h \hat{\beta}(w \mid h) & N(w,h) = 0 \end{cases} . \tag{2.17}$$

Regardless of the approach used to smooth *n*-gram models in all cases parameters $\lambda_{w,h}$ are needed. There are several techniques in order to calculate those factors, which are usually mentioned as discounting techniques. There are many discounts, as the *Witten-Bell* discount or the Absolute discount. However, most successful techniques are probably those based on the *Good-Turing* discount Good [1953], Nadas [1985], as the *Kneser-Ney* discount Kneser and Ney [1995]. Discount methods is a wide research field which is not the focus of this thesis, but as an example of discount techniques the expression of the *Good-Turing* discount is shown in the next equation.

$$\lambda_{w,h} = \frac{n_{r+1}(r+1)}{r n_r}, \tag{2.18}$$

being $N(w,h) = r$ and n_r the number of events which have been appeared r times in the training set. More information on discount techniques can be found in Andrés-Ferrer [2010].

2.4 EM Algorithm

A wide-spread criteria used in automatic parameter estimation, from a given training set $\mathbf{x}_1^N$, is the maximum likelihood estimation (MLE), which tries to maximize the log-likelihood of the training set, which is equivalent to maximize the following function

$$L(\mathbf{\Theta}; \mathbf{x}_1^N) = \sum_n \log p(\mathbf{x_n} \mid \mathbf{\Theta}), \tag{2.19}$$

that is, MLE consists in solving

$$\hat{\mathbf{\Theta}} = \underset{\mathbf{\Theta}}{\operatorname{argmax}} \sum_n \log p(\mathbf{x_n} \mid \mathbf{\Theta}). \tag{2.20}$$

This problem can be solved in most cases by performing first derivatives, since in most cases the MLE its a simple convex optimization problem. Some examples of this are the cases of MLE for Gaussian distributions, Bernoulli distributions or multinomial distributions. However, when the observations ($\mathbf{x}_1^N$) are modeled as incomplete data finding the solution to (2.20) is not a trivial task, and no closed form solution can be found with the previous approach anymore. Examples of probabilistic models with incomplete data are the hidden Markov models or mixture of probabilistic models, which are in fact wide-spread models in almost pattern recognitions fields. The EM algorithm was just proposed by Dempster et al. [1977] in order to apply the MLE criterion to those kind of models.

In the EM algorithm incomplete data is represented as hidden variables usually denoted as $\mathbf{z}_1^N$. Thus the original probabilistic distribution can be redefined as

$$p(\mathbf{x} \mid \mathbf{\Theta}) = \int p(\mathbf{x}, \mathbf{z} \mid \mathbf{\Theta}) d\mathbf{z} \,, \tag{2.21}$$

and then the MLE estimation function (2.20) can be rewritten as

$$\hat{\mathbf{\Theta}} = \underset{\mathbf{\Theta}}{\operatorname{argmax}} \sum_n \log \int p(\mathbf{x_n}, \mathbf{z_n} \mid \mathbf{\Theta}) d\mathbf{z_n}. \tag{2.22}$$

The previous equation is solved by the EM algorithm by means of a two step iterative approach. The first step is usually called *Expectation (E)* step , since the expected value of log-likelihood is calculated given a previous estimation of the parameters and the known data. While in the second step the parameters which maximize the expression of E step are calculated. Last step is usually denoted as *Maximization (M)* step, hence the name of the algorithm. The EM algorithm is proved to maximize the log-likelihood in each iteration [Wu, 1983]. A more schematic description of the algorithm is shown below.

Initialization: set $k = 0$ and choose initial $\mathbf{\Theta}_{(\mathbf{0})}$

Loop:

1. **E step:** for all $\mathbf{\Theta}$, compute

$$\mathcal{Q}(\mathbf{\Theta} \mid \mathbf{\Theta}^{(k)}) = \sum_n E(\log p(\mathbf{x_n}, \mathbf{z_n} \mid \mathbf{\Theta}) \mid \mathbf{x_n}, \mathbf{\Theta}^{(k)}) \tag{2.23}$$

2. **M step:** compute

$$\mathbf{\Theta}^{(k+1)} = \underset{\mathbf{\Theta}}{\operatorname{argmax}}\, \mathcal{Q}(\mathbf{\Theta} \mid \mathbf{\Theta}^{(k)}) \tag{2.24}$$

Until: $L(\mathbf{\Theta}^{(k+1)}; \mathbf{x}_1^N) - L(\mathbf{\Theta}^{(k)}; \mathbf{x}_1^N) \leq \epsilon$

2.5 Evaluation Metrics

All results results in this thesis are presented using one of these metrics: the classification error rate or the Word Error Rate (WER). The classification error rate is the conventional metric used in pattern recognition for classification problems. The metric is the percentage of errors in the test set and is calculated as,

$$\text{Error} = \frac{E}{N} \cdot 100 \ , \tag{2.25}$$

where E and N are respectively the number of errors and the number of samples in the test set. This metric is used in this thesis on all those experiments of isolated word (token) recognition, in which each word (token) is treated as a class.

The classification error rate can also be used in continuous HTR, in which a sequence of words (tokens) is obtained as a result of the recognition process. Nevertheless, it is somewhat a strict metric, since an erroneous word on the sequence implies that the sequence is considered wrong. WER tries to measure the error at word level instead of at sentence level. In order to do that, for each predicted sentence the minimum number of insertions, deletions and

substitutions needed to convert it into the reference is calculated. The WER is then calculated as

$$\text{WER} = \frac{I + D + S}{W} \cdot 100 \,, \tag{2.26}$$

where I, D and S are respectively the minimum number of insertions, deletions and substitutions needed, while W is the total number of words in the reference. I, D and S can be easily obtained computing the Levenshtein distance between the reference sentence and the recognized sentence. Note that the WER can be greater than 100% due to the insertions, for example, imagine that in the prediction the number of words is greater than in the reference and all them are wrong. Note also that the classification error is a particular case of the WER in which recognized and reference sequences have always one word. Sometimes the WER is calculated at character (symbol) level instead of at word level, in that cases we will refer to the character error rate.

Bibliography

Jesús Andrés-Ferrer. *Statistical approaches for natural language modelling and monotone statistical machine translation*. PhD thesis, Universidad Politécnica de Valencia, Valencia (Spain), Feb 2010. Advisors: A. Juan and F. Casacuberta.

T.M. Breuel. A system for the off-line recognition of handwritten text. volume 2, pages 129 –134 vol.2, oct. 1994a. doi: 10.1109/ICPR.1994.576889.

T.M. Breuel. Design and Implementation of a System for the Recognition of Handwritten Responses on US Census Forms. In *IAPR Workshop on Document Analysis Systems*, Kaiserslautern, 1994b.

Stanley F. Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, pages 310–318, Morristown, NJ, USA, 1996. Association for Computational Linguistics. doi: http://dx.doi.org/10.3115/981863.981904.

Dan Claudiu Ciresan, Ueli Meier, Luca Maria Gambardella, and Jürgen Schmidhuber. Deep Big Simple Neural Nets Excel on Handwritten Digit Recognition. *CoRR*, abs/1003.0358, 2010.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977. doi: http://dx.doi.org/10.2307/2984875. URL `http://dx.doi.org/10.2307/2984875`.

S. España-Boquera, M.J. Castro-Bleda, J. Gorbe-Moya, and F. Zamora-Martinez. Improving offline handwritten text recognition with hybrid hmm/ann models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(4):767–779, april 2011. ISSN 0162-8828. doi: 10.1109/TPAMI.2010.141.

I. J. Good. The population frequencies of species and the estimation of population parameters. *Biometrika*, 40(3-4):237–264, 1953. doi: 10.1093/biomet/40.3-4.237. URL `http://biomet.oxfordjournals.org/content/40/3-4/237.abstract`.

Joshua T. Goodman. A bit of progress in language modeling. Technical report, 2001.

Jorge Gorbe-Moya, Salvador España Boquera, Francisco Zamora-Martínez, and María José Castro Bleda. Handwritten Text Normalization by using Local Extrema Classification. In *PRIS*, pages 164–172, 2008.

Simon Günter and Horst Bunke. HMM-based handwritten word recognition: on the optimization of the number of states, training iterations and Gaussian components. *Pattern Recognition*, 37:2069–2079, 2004.

Slava M. Katz. Estimation of probabilities from sparse data for the language model component of a speech recognizer. In *IEEE Transactions on Acoustics, Speech and Signal Processing*, pages 400–401, 1987.

R. Kneser and H. Ney. Improved backing-off for M-gram language modeling. volume 1, pages 181 – 184 vol.1, may. 1995. doi: 10.1109/ICASSP.1995.479394.

Saeed Mozaffari and Hadi Soltanizadeh. ICDAR 2009 Handwritten Farsi/Arabic Character Recognition Competition. In *Proc. of the 10th Int. Conf. on Document Analysis and Recognition (ICDAR 2009)*, pages 1413–1417, Barcelona (Spain), July 2009.

A. Nadas. On Turing's formula for word probabilities. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 33(6):1414 – 1416, dec. 1985. ISSN 0096-3518.

Moisés Pastor i Gadea. *Aportaciones al reconocimiento automático de texto manuscrito*. PhD thesis, Dep. de Sistemes Informàtics i Computació, València, Spain, Oct 2007. Advisors: E. Vidal and A.H. Tosselli.

Réjean Plamondon and Sargur N. Srihari. On-Line and Off-Line Handwriting Recognition: A Comprehensive Survey. *IEEE Trans. on PAMI*, 22(1):63–84, 2000.

Lawrence Rabiner and Biing-Hwang Juang. *Fundamentals of speech recognition*. Prentice-Hall, 1993.

V. Romero, A. Giménez, and A. Juan. Explicit Modelling of Invariances in Bernoulli Mixtures for Binary Images. In *3rd Iberian Conference on Pattern Recognition and Image Analysis*, volume 4477 of *LNCS*, pages 539–546. Springer-Verlag, Girona (Spain), June 2007.

A. H. Toselli, A. Juan, D. Keysers, J. González, I. Salvador, H. Ney, E. Vidal, and F. Casacuberta. Integrated Handwriting Recognition and Interpretation using Finite-State Models. *International Journal of Pattern Recognition and Artificial Intelligence*, 18(4):519–539, 2004.

C. F. Jeff Wu. On the convergence properties of the em algorithm. *The Annals of Statistics*, 11(1):95–103, 1983. ISSN 00905364. doi: 10.2307/2240463. URL `http://dx.doi.org/10.2307/2240463`.

Hanhong Xue and Venu Govindaraju. Hidden Markov Models Combining Discrete Symbols and Continuous Attributes in Handwriting Recognition. *IEEE Trans. on PAMI*, 28:458–462, 2006.

CHAPTER 3

DATABASES

Contents

Public available databases are required in pattern recognition in order to evaluate and compare the different techniques. However, maybe due to a lack of interest or maybe due to the difficulty to obtain good databases, continuous HTR has suffered for years of a lack of public available databases. In last years, this situation seems to be changing and new databases of continuous HTR have been published, in particular for Latin and Arabic scripts. The current number of public available databases is not large, but good enough to contrast results between researchers.

In this work we have tested proposed techniques over six different databases: *CENPARMI Arabic cheque database*, *IAM Handwriting Database*, *IFN/ENIT - database*, *RIMES database*, *GERMANA database* and *RODRIGO database*. We think that six database is enough to properly test the techniques proposed in this work. Moreover, these databases gather some very interesting features: different kind of scripts (Arabic and Latin), state of the art comparable results (most of the current published results on Arabic and Latin scripts have been obtained from IFN/ENIT, IAM and RIMES), ancient and modern writing, mono and multi writer tasks, different kind of tasks (isolated word recognition and line recognition), and more. In next sections all six databases are described in detail.

3.1 CENPARMI Arabic cheque database

The CENPARMI database is a database of handwritten Arabic cheques [Al-Ohali et al., 2004]. Previous to this database no attempts were reported in Arabic cheque processing, where cheque processing means all tasks that must be performed by a bank officer in order to process an incoming cheque for a client. It is worth noting, that there are too many differences between Arabic and Latin scripts, and therefore conventional techniques developed for Latin scripts can not be directly applied to this task. Besides the fact of a different script and that text is written from right to left, Arabic is written in cursive script following strict rules. That is, within one word two consecutive letters will be connected or not depending on which letters they are. Therefore, character segmentation becomes a difficulty task since white spaces can appear inside words, and connected letters are difficulty to segment. Moreover, in some parts of Arabian world Indian digits are more popular than Arabic numerals. Due to all these reasons, this database was collected by the Center for Pattern Recognition and Machine Intelligence (CENPARMI) in order to provide a public database which can be used to develop and compare Arabic cheque processing systems.

The data were collected from about 7000 real cheques from Al Rajhi Banking and Investment Corporation. All cheques were scanned at 300 dpi, removing in all cases the personal information. Figure 3.1 shows two examples. From these 7000 cheques, 3000 cheques without stamps on top of their legal amounts were selected in order to extract the data. In a semisupervised procedure four different objects were extracted from each cheque: courtesy amount, legal amount, Indian digits and Arabic sub-words (sequences of connected letters). As a result of this procedure four different datasets were obtained, one for each kind of object: 2499 legal and courtesy amounts, 29498 sub-words and 15175 Indian digits. Each dataset is divided into training and test sets, using approximately the 75% of samples for training. Furthermore, training and testing sets are divided into touching and non-touching sets, that is, non-touching sets contain isolated objects which have been correctly segmented,

while touching sets contain images with one or more touching objects which could not be correctly segmented.

Figure 3.1: Two examples of Arabic cheques extracted from the CENPARMI Arabic cheque database

In this work we have focused in the non-touching Arabic sub-words dataset, which is used in order to perform isolated word (sub-word in this case) recognition experiments, and in the non-touching Indian digits dataset, which is used in order to perform isolated character recognition experiments. Table 3.1 gathers some statistics of the sub-words dataset, while in Table 3.2 statistics and examples of the ten most frequent sub-words are shown. Regarding the Indian digit dataset, the standard protocol is a simple partition with 7390 samples for training and 3035 for testing (excluding the extra classes *delimiter* and *comma*). In Figure 3.2 some digit examples are shown. Additionally, a comparison of the best result obtained in this thesis and published results is shown in Table 3.3. We do not report a comparison for the sub-words dataset, since we do not known other published results.

Table 3.1: Some statistics of the CENPARMI non-touching Arabic sub-words dataset

	No. of samples	No. of sub-words	No. of Singletons
Train	19813	96	12
Test	8172	101	28
All	27985	101	16

Table 3.2: Ten most frequent sub-words of the CENPARMI non-touching Arabic sub-words dataset. From right to left: sub-word identification, number of train samples, number of test samples and image example

Id	Train	Test	Example
1-08	2061	859	
1-14	1896	781	
2-09	1726	724	
2-13	1301	529	
1-10	1175	490	
1-00	1159	521	
3-22	1077	442	
3-23	1005	410	
1-11	961	396	
2-10	745	304	

3.2 IAM Handwriting Database

Developed in the "Institut für Informatik und angewandte Mathematik" (IAM) in the "Universität Bern", IAM Handwriting Database is probably the current reference database for hand-

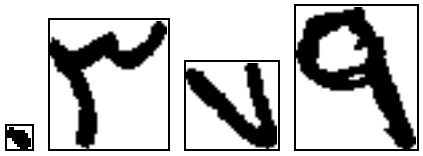


Figure 3.2: Examples of the CENPARMI non-touching Arabic Indian digit dataset. From left to right the digits corresponding to 0, 3, 7 and 9

Table 3.3: Result comparison on CENPARMI non-touching Arabic Indian digit dataset. Similar protocols but not exactly the same

	Methodology	**ERR%**
[Juan and Vidal, 2004]	Bernoulli mixtures (BM)	2.3
[Romero et al., 2007]	BM with explicit modeling of invariances	1.9
	Discriminative BM	2.0

written text recognition using Latin script [Marti and Bunke, 2002]. Unlike other databases which are made up from pre-existent handwritten documents, in this database all documents were written before the scanning process. In order to do this, a collection of 500 English texts were collected from the LOB corpus. LOB is a corpus containing British English texts extracted from an heterogeneous set of sources and matters, see Johansson et al. [1978]. Extracted texts were segmented into fragments of about five sentences, and printed onto forms which were automatically generated using Latex. Each form was handwritten by one or several persons. Finally, handwritten forms were scanned with a resolution of 300 dpi at a grey-level resolution of 8 bits. A total of 657 different writers participated, obtaining 1539 handwritten forms. The only restrictions imposed to writers were to stop writing if there was not space left on the form, and using rulers. Two examples of handwritten forms are shown in Figure 3.3.

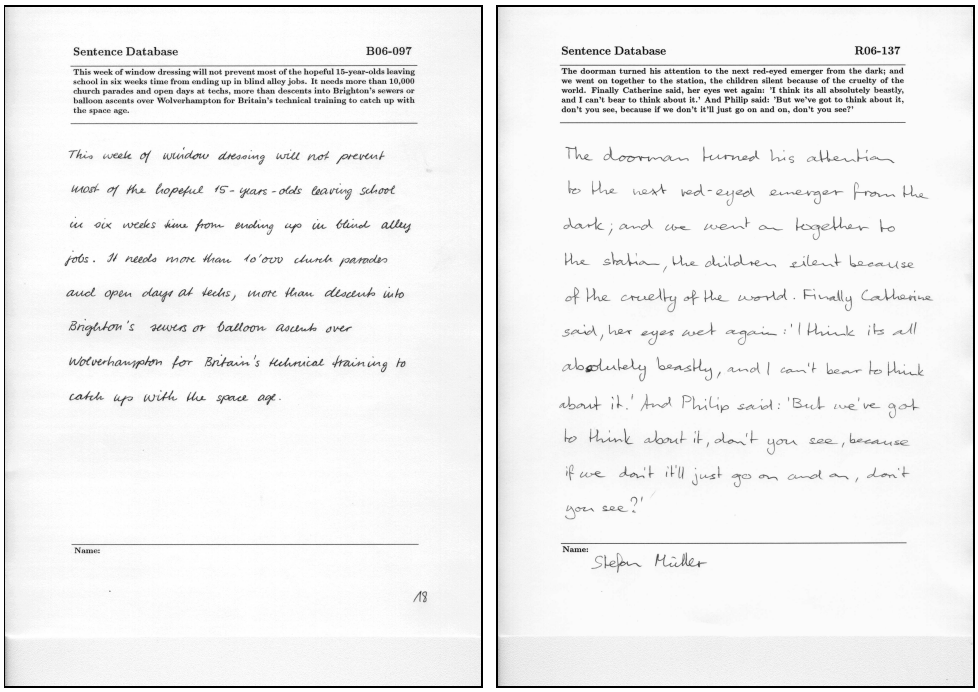


Figure 3.3: Example of two handwritten forms from the IAM Handwriting Database

Table 3.4: Published results on the IAM words dataset. Protocols differ so results are for guidance only

	Methodology	**Lexicon Size**	**ERR%**
[Günter and Bunke, 2004]	HMMs	3 997	20.5
[Bianne-Bernard et al., 2011]	Context-dependent HMMs	10 500	32.7
[Bianne-Bernard et al., 2011]	System Combination	10 500	21.9
	Bernoulli HMMs	**1 117**	21.4
	Bernoulli HMMs	**10 208**	25.8

Using automatically techniques, handwritten text was extracted from forms, segmented into lines, then segmented into words, and finally labeled with its transcription. In addition to the transcription, words have associated more meta-data as for example the grammatical tag of the word, or a flag to indicate the segmentation correctness. From this process two datasets were obtained: a word dataset for handwritten isolated word recognition and a line dataset for continuous handwritten recognition. In addition, a third sentence dataset is available which was manually obtained by means of segmenting the samples of the line dataset to avoid images containing more than one sentence. In this work we have performed experiments with both word and line datasets.

3.2.1 IAM word dataset

The IAM word dataset is the result of segmenting into words all handwritten forms. This dataset contains 115320 images, 96456 of them labeled as correctly segmented. The number of different words is 13542, the most frequent word is *the* with 5826 images, while the number of words with ten or less images is 12362 (about 91% of all dataset) and the number of singletons is 7185 (about 53% of all dataset). No experimentation protocol is defined for the dataset. Figure 3.4 shows some examples.

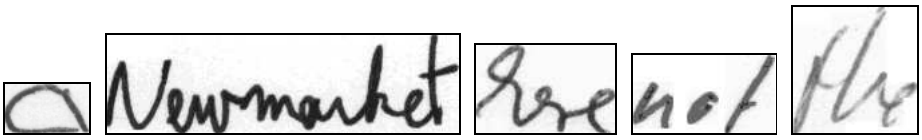


Figure 3.4: Some examples of the IAM word dataset. From left to right: *a*, *Newmarket*, *here*, *not* and *the*

In Table 3.4 we shown some published results on this dataset, and the best results obtained in this work. However, since there is no a standard protocol defined, the results are for guidance only.

3.2.2 IAM line dataset

The IAM line dataset is the result of segmenting into lines all handwritten forms. This dataset contains 13353 samples, 11344 of them labeled as correctly segmented. The total number

of words is 115174, the average words per line is 9, and the size of the lexicon is 13528. Figure 3.5 shows some examples.

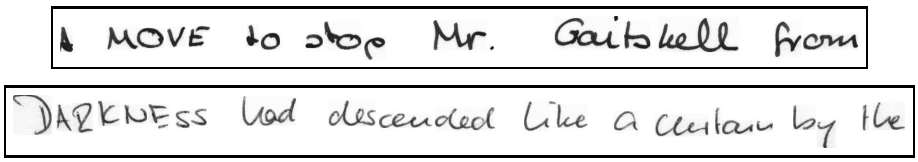


Figure 3.5: Two examples from the IAM line dataset

There is no a standard protocol for the IAM line dataset. However, in most publications the protocol described in Bertolami et al. [2007] is used to compare results. This experimentation protocol is raised as a writer independent task, and consists of training, test and validation sets. Basic statistics of these sets are presented in Table 3.5. The details about the development of the experimental protocol are unknown. This protocol has became the de-facto standard protocol for this dataset. In Table 3.6 we show a table with the best results published using this protocol and the best result obtained in this thesis.

Table 3.5: Standard de facto experimental protocol for the IAM line dataset

	Training	Test	Validation
No. Samples	6161	2781	920
No. writers	283	161	56

Table 3.6: Best published results on the standard protocol for the IAM lines dataset

	Methodology	**WER%**
[Bertolami et al., 2007]	HMM	34.2
[Graves et al., 2009]	RNN	25.9
[España-Boquera et al., 2011]	Hybrid HMM+NN	21.2
[Kozielski et al., 2013]	Tandem HMM+NN	13.3
	Bernoulli HMMs	31.1

3.3 IFN/ENIT - database

IFN/ENIT database is an Arabic handwritten text database which contains handwritten Tunisian town/villages names, see Pechwitz et al. [2002]. It is then a database for isolated word recognition. In last years this database has been used in several Arabic handwritten competitions, see Märgner and Abed [2007, 2009, 2010], Märgner et al. [2005], Märgner and El Abed [2011], becoming this database a reference in the Arabic handwritten area. In order to build the database 946 Tunisian town/villages were selected, and 411 writers were asked to fill 5 forms with 12 names from the possible names with their corresponding postcodes. Forms were made guarantying that each word appears at least 3 times in the database, and each

character shape occur at minimum more than 200 times. The only aid to writing was the printing of dark light rectangles in the backside of the form to indicate where to write the words. Figure 3.6 shows two examples of forms.

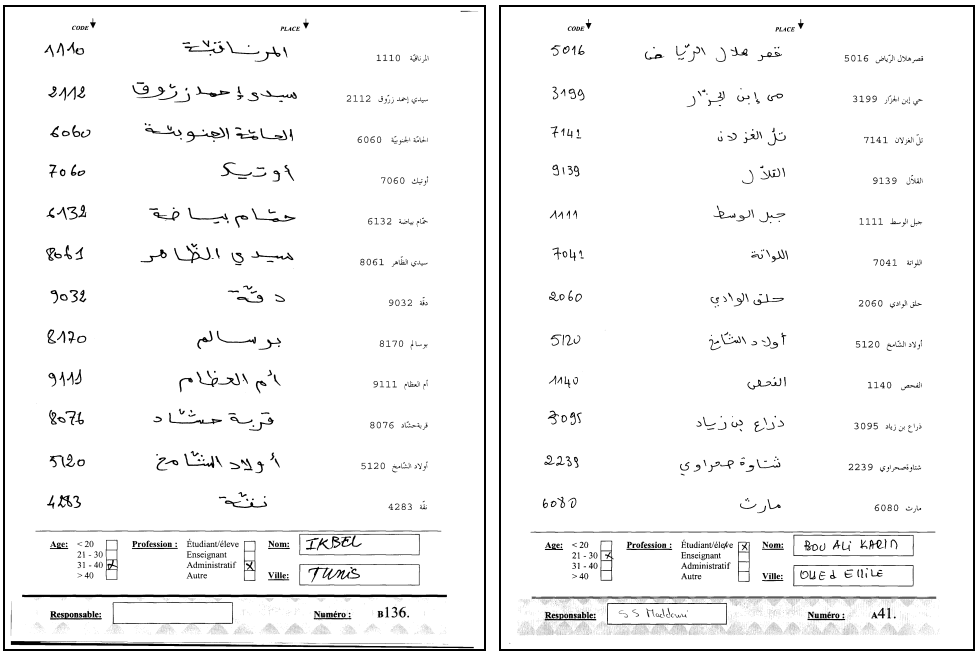


Figure 3.6: Example of two handwritten forms from the IFN/ENIT - database

Forms were scanned with 300 dpi and, binarized and automatically segmented. Using a semi-automatically process, segmented images were labeled with the postcode, the Arabic word in ISO 8859-6, and with a sequence of Arabic character shapes using 306 different shapes, since each letter can appear in four different forms depending on its position in the word (begin,middle,end or isolated form). It is worth noting that ISO 8859-6 does not encode the shape information.

The resulting database is composed by 32492 different images divided into 5 sets (a,b,c,d and e). The first four sets are the original sets of the database, while the set e was used as test set in the ICDAR 2005 competition, see Märgner et al. [2005], being lately released. Thus, it is a common practice publishing results doing a cross validation experiment with the first four sets, and a final experiment using sets a,b,c,d for training and the set e for testing. Note, that while the number of classes is 946 (the postcodes), the size of the lexicon is greater since some names appear written in different ways. Table 3.7 shows some statistics for the five sets, while Figure 3.7 shows some examples of samples.

Apart from the 5 previously described sets, there are two extra sets (s and f) which have not been still published. These two sets have been used in the last four competitions as test sets. In Table 3.8 the best results from last four competitions is shown. It is worth noting, that we participated in the ICFHR 2010 competition, using a Bernoulli HMM system, and we

Table 3.7: Some statistics of the IFN/ENIT-database sets

	No. Samples	Lexicon
a	6537	1588
b	6710	1634
c	6477	1498
d	6735	1564
e	6033	733

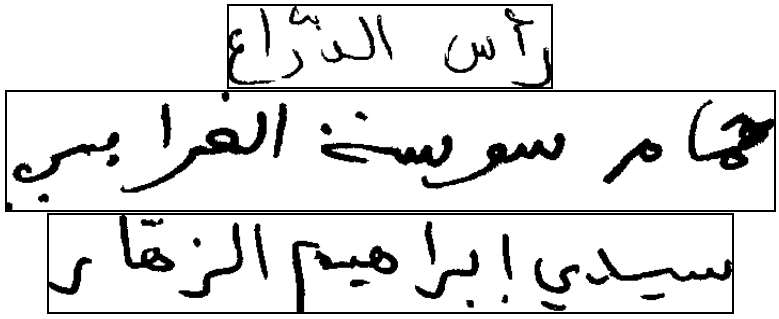


Figure 3.7: Some examples of samples of the IFN/ENIT-database

ranked first [Märgner and Abed, 2010].

Table 3.8: Best results from last four Arabic Handwriting Recognition Competitions

System	**Methodology**	**Conference**	**ACC%**	
			set *f*	set *s*
Siemens	HMMs	ICDAR 2007	87.22	73.94
MDLSTM	Neural Networks	ICDAR 2009	**93.37**	81.06
UPV PRHLT (**This thesis**)	Bernoulli HMMs	ICFHR 2010	92.20	**84.62**
RWTH-OCR	Tandem HMM+NN	ICDAR 2011	92.20	84.55

3.4 RIMES database

The RIMES (*Reconnaissance et Indexation de données Manuscrites et de fac similÉS*) database is a database of French handwritten text letters, developed in order to evaluate automatic systems of recognition and indexing of handwritten letters. This database has been used last years in several competitions to test several document analysis tasks as document layout analysis, isolated character recognition, or more recently, handwritten text lines recognition. But the task in which it has been more tested and compared is the recognition of isolated handwritten words [Grosicki and H., 2009, Grosicki et al., 2009, Grosicki and El Abed, 2011].

In order to collect the data, volunteers were asked to write handwritten letters, such those sent by individuals to companies by fax or postal mail. Each volunteer was required to write

using a given fictional identity (same sex as the real one) and up to 5 scenarios, which were chosen among 9 realistic themes: change of personal information, information request, opening and closing, modification of contract or order, complaint, payment difficulties, reminder letter, damage declaration with further circumstances and a destination. The volunteers were free to write the letters as they wanted, the only requirement was to use white paper and black ink. More than 1300 people, writing up to 5 mails, contributed to the creation of the RIMES database with more than 12723 pages corresponding to 5605 mails.

As previously mentioned, this dataset has been used in several tasks, but in this thesis we will report only results on the handwritten word recognition task, in particular on the dataset used in the *ICDAR 2009 Handwriting Recognition Competition* [Grosicki and H., 2009]. This dataset is composed by 59202 different images divided into training, validation and test sets. Results reported on this dataset are usually presented using three different protocols. In the first one, a list of 100 words containing the correct one is provided for each testing image. In the second protocol a dictionary containing all 1612 words from test is provided. In the third and final protocol, the given dictionary also contains words from the training set. In this work all results will be presented using the second protocol. Table 3.9 shows some statistics for the three sets, while Figure 3.8 shows some examples of samples. The results of the ICDAR 2009 competition, and our best obtained results, are shown in Table 3.10.

Table 3.9: Some statistics of the RIMES words dataset used in ICDAR 2009

	No. Samples	Lexicon
training	44196	4508
validation	7542	1636
test	7464	1612

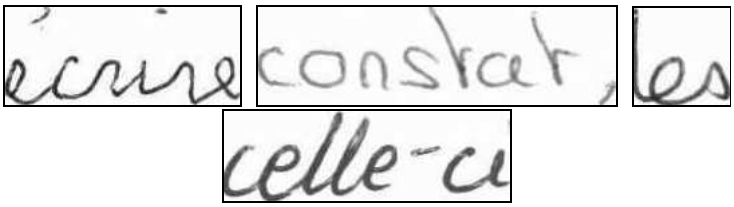


Figure 3.8: Some examples of the RIMES word dataset. From left to right: *écrire*, *constat*, *les* and *celle-ci*

3.5 GERMANA database

The GERMANA database is the result of scanning and annotating the manuscript entitled "Noticias y documentos relativos a Doña Germana de Foix, última Reina de Aragón", see Pérez et al. [2009]. This manuscript was written by Vicent Salvador in 1891 and deals about the life of Germana de Foix (1488-1538), niece of King Louis XII of France and second wife of Ferdinand the Catholic of Aragon. Most of the document is written in Spanish, nevertheless

Table 3.10: Test-set classification error on RIMES obtained with BHMMs and different systems participating at the ICDAR 2009 competition (using the WR2 protocol). NN and MRF refer, respectively, to Neural Networks and Markov Random Fields

System	**Methodology**	**WER%**
TUM	NN	6.8
UPV	Hybrid NN+HMM	13.9
BHMM (this thesis)	HMM	16.8
SIEMENS	HMM	18.7
ParisTech (1)	Hybrid NN+HMM	19.8
IRISA	HMM	20.4
LITIS	HMM	25.9
ParisTech (2)	HMM	27.6
ParisTech (3)	HMM	36.2
ITESOFT	MRF+HMM	40.6

other 5 languages appear in the document, mainly Catalan and Latin. This is due to the fact that at the end of the document several historical documents are appended.

The scanning process of the manuscript was carefully done by experts from the Valencian Library, since this library is where the original documents is preserved. Pages were scanned at 300 dpi in true colors. As expected from an historical document, scanned pages have noise effects like spots, transparency of back side, etc. The annotation process was carried out in the PRHLT group in the "Universitat Politècnica de València" by paleography experts. This transcription was done respecting as much as possible the original text layout and shapes employed. Additionally, all text blocks were marked with minimal enclosing rectangles, and each text line was marked by its baseline. This was done with he aid of the GiDoc plugin for the GNU Image Manipulation Program (GIMP), see Serrano et al. [2010]. Table 3.11 gathers some statistics of the manuscript, while Figure 3.9 shows an example of a page from the manuscript.

Table 3.11: Some statistics of the GERMANA database

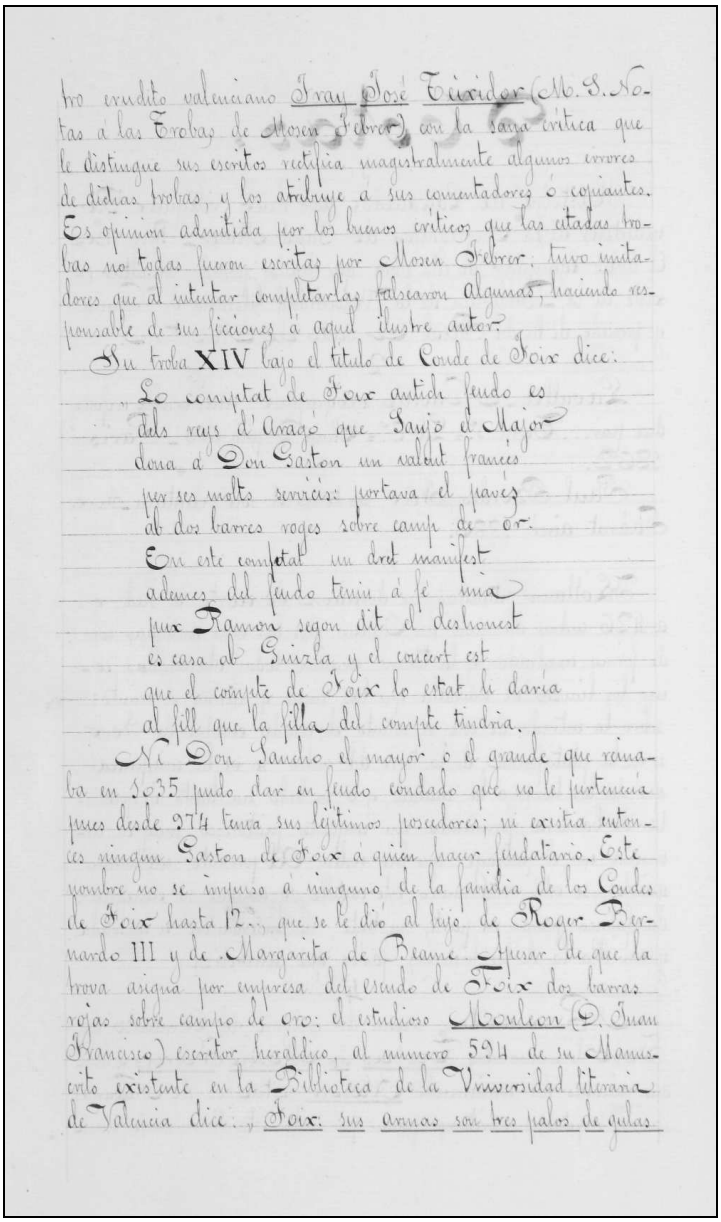
Language	Pages	Lines	Words(K)	Lexicon(K)	Singletons(%)
Spanish	595	16599	176.8	19.9	55.6
Catalan	87	2417	26.9	4.6	63.2
Latin	29	951	8.3	3.4	69.2
French	8	266	3.0	1.1	71.1
German	8	228	1.5	0.6	52.7
Italian	2	68	0.8	0.3	67.3

3.6 RODRIGO database

As the GERMANA database, the RODRIGO database is the transcription of an old manuscript, see Serrano and Juan [2010]. Nevertheless, this manuscript is older (1545) than the GER-

MANA manuscript (1891). The manuscript has 853 pages, written entirely in old Castilian (Spanish) by a single author. Due to its antique this manuscript presents some interesting singularities: embellishing writing, omission of natural spaces between words and addition of artificial spaces, using of an Humanistic script, etc.

The manuscript was carefully scanned at 300 dpi in true color by experts from the "Ministerio de Cultura". As in the GERMANA case, the annotation process, text block detection, and line marking was carried out by paleography experts in the PRHLT group of the "Universitat Politècnica de València" with the aid of the GiDoc prototype. The transcription was carried out respecting the original manuscript layout, shapes and spelling mistakes. The transcription contains 20357 lines and 223447 words. The size of the lexicon is 20610, where 12004 are singletons (the 58% of the lexicon), and the number of different symbols is 119. Figure 3.10 shows a page of the manuscript.



tro erudito valenciano Fray José Teixidor (M. S. Notas á las Trobas de Mosen Febrer) con la sana crítica que le distingue sus escritos rectifica magistralmente algunos errores de dichas trobas, y los atribuye á sus comentadores ó copiantes. Es opinion admitida por los buenos críticos que las citadas trobas no todas fueron escritas por Mosen Febrer; tuvo imitadores que al intentar completarlas falsearon algunas, haciendo responsable de sus ficciones á aquel ilustre autor.

Su troba XIV bajo el titulo de Conde de Foix dice:

„Lo comptat de Foix antich feudo es
dels reys d'Aragó que Sanjo el Major
dona á Don Gaston un valent frances
per ses molts serveis: portava el pavés
ab dos barres roges sobre camp de or.
En este comptat un dret manifest
ademes del feudo tenia á fé mia
pux Ramon segon dit el deshonest
es casa ab Guizla y el concert est
que el compte de Foix lo estat li daria
al fill que la filla del compte tindria.

Ni Don Sancho el mayor ó el grande que reinaba en 1035 pudo dar en feudo condado que no le pertenecia pues desde 974 tenia sus legitimos poseedores; ni existia entonces ningun Gaston de Foix á quien hacer feudatario. Este nombre no se impuso á ninguno de la familia de los Condes de Foix hasta 12.., que se le dió al hijo de Roger Bernardo III y de Margarita de Beame. Apesar de que la troya asigna por empresa del escudo de Foix dos barras rojas sobre campo de oro: el estudioso Monleon (D. Juan Francisco) escritor heraldico, al número 594 de su Manuscrito existente en la Biblioteca de la Universidad Literaria de Valencia dice.. „Foix: sus armas son tres palos de gulas.

Figure 3.9: Page extracted from the GERMANA database

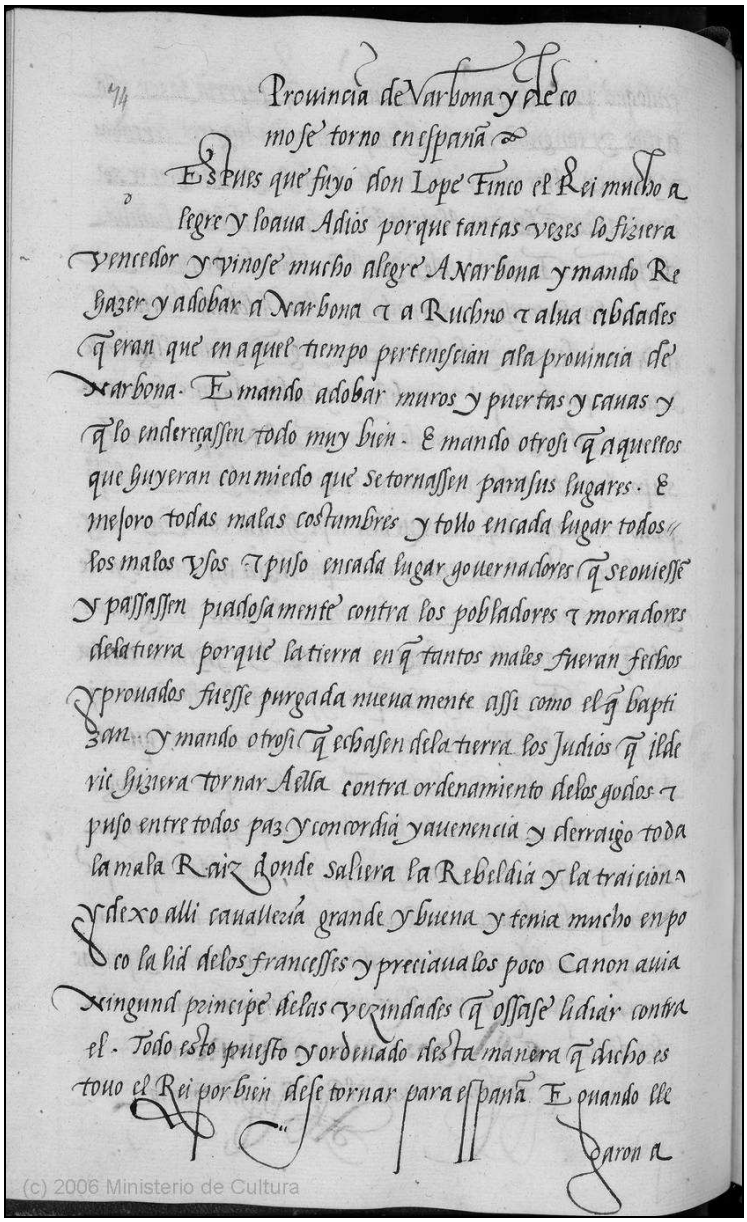


Figure 3.10: Page from the RODRIGO database

Bibliography

Yousef Al-Ohali, Mohamed Cheriet, and Ching Suen. Databases for recognition of handwritten Arabic cheques. *Pattern Recognition*, 36:111–121, 2004.

R. Bertolami, S. Uchida, M. Zimmermann, and H. Bunke. Non-uniform slant correction for handwritten text line recognition. In *Proc. 9th Int. Conf. on Document Analysis and Recognition (ICDAR 2007)*, pages 18–22, 2007.

A.-L. Bianne-Bernard, F. Menasri, R. Al-Hajj Mohamad, C. Mokbel, C. Kermorvant, and L. Likforman-Sulem. Dynamic and contextual information in hmm modeling for handwritten word recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(10): 2066 –2080, oct. 2011. ISSN 0162-8828. doi: 10.1109/TPAMI.2011.22.

S. España-Boquera, M.J. Castro-Bleda, J. Gorbe-Moya, and F. Zamora-Martinez. Improving offline handwritten text recognition with hybrid hmm/ann models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(4):767–779, april 2011. ISSN 0162-8828. doi: 10.1109/TPAMI.2010.141.

A. Graves, M. Liwicki, S. Fernandez, R. Bertolami, H. Bunke, and J. Schmidhuber. A Novel Connectionist System for Unconstrained Handwriting Recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(5):855 –868, may 2009. ISSN 0162-8828. doi: 10.1109/TPAMI.2008.137.

E. Grosicki and El Abed H. ICDAR 2009 Handwriting Recognition Competition. In *Proc. of the International Conference on Document Analysis and Recognition (ICDAR 2009)*, pages 1398–1402, 2009.

E. Grosicki, M. Carré, J. M. Brodin, and E. Geoffrois. Results of the RIMES evaluation campaign for handwritten mail processing. In *Proc. of the International Conference on Document Analysis and Recognition (ICDAR 2009)*, pages 941–945, 2009.

Emmanuèle Grosicki and Haikal El Abed. ICDAR 2011 - French Handwriting Recognition Competition. In *ICDAR '11*, pages 1459 – 1463, Beijing (China), 2011.

Simon Günter and Horst Bunke. HMM-based handwritten word recognition: on the optimization of the number of states, training iterations and Gaussian components. *Pattern Recognition*, 37:2069–2079, 2004.

S. Johansson, G.N. Leech, and H. Goodluck. *Manual of information to accompany the Lancaster-Oslo/Bergen Corpus of British English, for use with digital Computers*. Department of English, University of Oslo, Norway, 1978.

A. Juan and E. Vidal. Bernoulli mixture models for binary images. In *Proc. of ICPR 2004*, volume 3, Cambridge (UK), August 2004.

Michal Kozielski, Patrick Doetsch, and Hermann Ney. Improvements in rwth's system for off-line handwriting recognition. In *Proc of the 12th Int. Conf. on Document Analysis and Recognition (ICDAR 2013)*, pages 935–939, 2013.

V. Märgner and H. E. Abed. ICDAR 2007 - Arabic Handwriting Recognition Competition. In *Proc. of the 9th Int. Conf. on Document Analysis and Recognition (ICDAR 2007)*, volume 2, pages 1274–1278, Curitiba (Brazil), September 2007.

V. Märgner and H. E. Abed. ICDAR 2009 Arabic Handwriting Recognition Competition. In *Proc. of the 10th Int. Conf. on Document Analysis and Recognition (ICDAR 2009)*, pages 1383–1387, Barcelona (Spain), July 2009.

V. Märgner and H. E. Abed. ICFHR 2010 - Arabic Handwriting Recognition Competition. In *Proceedings of the 12th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 709–714, Kolkata (India), November 2010.

V. Märgner, M. Pechwitz, and H. E. Abed. ICDAR 2005 Arabic Handwriting Recognition Competition. In *Proc. of the 8th Int. Conf. on Document Analysis and Recognition (ICDAR 2005)*, volume 1, pages 70–74, Seoul (Korea), 2005.

Volker Märgner and Haikal El Abed. ICDAR 2011 - Arabic Handwriting Recognition Competition. In *ICDAR '11*, pages 1444 – 1448, Beijing (China), sep 2011.

U.V. Marti and H. Bunke. The IAM-database: an English sentence database for offline handwriting recognition. 5(1):39–46, 2002.

M. Pechwitz, S. Snoussi Maddouri, V. Märgner, N. Ellouze, and H. Amiri. IFN/ENIT - DATABASE OF HANDWRITTEN ARABIC WORDS. In *7th Colloque International Francophone sur l'Ecrit et le Document, CIFED*, pages 21–23, Hammamet (Tunis), October 2002.

Daniel Pérez, Lionel Tarazón, Nicolas Serrano, Francisco-Manuel Castro, Oriol Ramos-Terrades, and Alfons Juan. The GERMANA database. In *Proceedings of the 10th International Conference on Document Analysis and Recognition*, pages 301–305, Barcelona (Spain), July 2009. IEEE Computer Society.

V. Romero, A. Giménez, and A. Juan. Explicit Modelling of Invariances in Bernoulli Mixtures for Binary Images. In *3rd Iberian Conference on Pattern Recognition and Image Analysis*, volume 4477 of *LNCS*, pages 539–546. Springer-Verlag, Girona (Spain), June 2007.

N. Serrano and A. Juan. The RODRIGO database. In *Proceedings of the The seventh international conference on Language Resources and Evaluation (LREC 2010)*, Malta, May 19-21 2010.

N. Serrano, L. Tarazón, D. Pérez, O. Ramos, and A. Juan. The GIDOC prototype. In *Proceedings of the 10th International Workshop on Pattern Recognition in Informatic Systems (PRIS 2010)*, June 2010.

CHAPTER 4

BERNOULLI HIDDEN MARKOV MODELS

Contents

4.1 Introduction

Hidden Markov models (HMMs) have received significant attention in off-line handwriting recognition during the last years [Günter and Bunke, 2004, Plamondon and Srihari, 2000, Toselli et al., 2004, Xue and Govindaraju, 2006]. As in speech recognition [Jelinek, 1997, Rabiner and Juang, 1993], HMMs are used to model the probability (density) of an observation sequence, given its corresponding text transcription or simply its class label.

Observation sequences typically consist of fixed-dimension feature vectors which are computed locally, using a sliding window along the handwritten text image. However, there is no standard set of local features being used by most of the proposed systems; on the contrary, it seems that each system proposed is tuned using a significantly different set of features. For instance, in Toselli et al. [2004], the preprocessed text image is transformed into a sequence of 60-dimensional feature vectors, each comprising 20 normalized gray levels plus 40 gray-level derivatives (20 horizontal and 20 vertical). In Günter and Bunke [2004], however, only 9 local features are computed: 3 characterizing the sliding window globally, and 6 capturing additional information about the writing. Another example can be found in Xue and Govindaraju [2006], where both discrete and continuous features are combined.

In this chapter, we explore the possibility of not using elaborated local features but using raw binary pixels instead. The objective is twofold: on the one hand, this allows us to better modeling the binary nature of text images by introducing probabilistic models that deal more directly with binary data. To this purpose, we propose the use of *Bernoulli HMMs*, that is, HMMs in which the state-conditional probability (density) function is not a conventional Gaussian (mixture) density, but a multivariate Bernoulli (mixture) probability function. On the other hand, this guarantees that no discriminative information is filtered out during feature extraction, which now has to be somehow integrated into recognition.

The chapter is organized as follows. General theory about HMMs is revised in Section 4.2. The definition of Bernoulli HMM and its EM-based maximum likelihood estimation are given in Section 4.3. In Section 4.4, some empirical results are reported on two tasks of handwriting word recognition. Finally, some concluding remarks and future work are discussed in Section 4.5.

4.2 HMMs

HMMs are used to model the probability (density) of an observation sequence. In the HMM approximation it is assumed that the observation sequence has been generated by a known finite state machine which in each state generates an observation according to a certain probability distribution. However, the sequence of responsible states for the generation of the observation sequence is unknown, remaining as a hidden variable. More formally, in a way similar to Jelinek [1997], we characterize an HMM as follows:

1. M, the number of states in the model. Individual states are labeled as $\{1, 2, \ldots, M\}$ and we denote the state at time t as q_t. In addition, we define the special states I and F for *start* and *stop*.

2. The state-transition probability distribution, $A = [a_{ij}]$, where

$$a_{ij} = P(q_{t+1} = j \mid q_t = i)\,, \qquad i, j \in \{1, 2, \ldots, M\} \cup \{I, F\}\,, \tag{4.1}$$

that is, the probability to carry out a transition from state q_t to state q_{t+1}. For convenience, we set $a_{IF} = 0$.

3. The observation probability (density) function, $B = \{b_j(o)\}$, in which

$$b_j(o_t) = P(o_t \mid q_t = j)\,, \tag{4.2}$$

defines the probability (density) function in state j, where $j \in \{1, 2, \ldots, M\}$.

For convenience, the specification of an HMM can be compacted as

$$\boldsymbol{\Theta} = (A, B)\,, \tag{4.3}$$

and in advance we will refer to the parameters of a HMM as $\boldsymbol{\Theta}$. Therefore, the probability (density) of an observation sequence $O = o_1, \ldots, o_T$ is given by:

$$P(O \mid \boldsymbol{\Theta}) = \sum_{\mathbf{q}} P(O, \mathbf{q} \mid \boldsymbol{\Theta}) \cong \sum_{\mathbf{q}} \prod_{t=0}^{T} a_{q_t q_{t+1}} \prod_{t=1}^{T} b_{q_t}(o_t)\,, \tag{4.4}$$

where we have uncovered the latent variables $\mathbf{q} = (q_0, q_1, \ldots, q_{T+1})$, which represent all the possible state sequences (or paths), such that $q_1, \ldots, q_T \in \{1, \ldots, M\}$ are the regular states chosen out of a total of M states, and the first and last states have been fixed to ($q_0 = I$) and ($q_{T+1} = F$), respectively.

4.2.1 Parameter Estimation

Maximum likelihood estimation of the parameters governing an HMM can be carried out using the EM algorithm for HMMs. Given an observation sequence $O = o_1, \ldots, o_T$, the HMM probability can be written as the marginalization of a function without hidden variables,

$$P(O \mid \boldsymbol{\Theta}) = \sum_{\mathbf{q}} P(O, \mathbf{q} \mid \boldsymbol{\Theta})\,, \tag{4.5}$$

where $\mathbf{q}$ denotes a state sequence of length T (if start and stop states are considered $T + 2$), and $P(O, \mathbf{q} \mid \boldsymbol{\Theta})$ is the so called completed model. For convenience we will replace variable $\mathbf{q}$ by $(T + 2)$ different variables $q_t \in \{I, 1, \ldots, M, F\}$ satisfying $\mathbf{q} = (q_0, \ldots, q_{T+1})$ and

$$q_t = \begin{cases} I & t = 0 \\ F & t = T + 1 \\ j \in \{1, \ldots, M\} & 1 \le t \le T \end{cases}\,. \tag{4.6}$$

Another possible representation can be obtained using $(T+2)\times(M+2)$ variables $q_{tj} \in \{0, 1\}$ satisfying

$$q_{tj} = \begin{cases} 1 & q_t = j \\ 0 & \text{otherwise} \end{cases}\,. \tag{4.7}$$

Note that (4.5) may be rewritten by using any of the three possibilities, simply by substituting the sum in $\mathbf{q}$ with as many sums as the number of used variables.

If $\mathbf{q}$ is represented by means of binary variables q_{tj}, then the complete model can be written as

$$P(O, \mathbf{q} \mid \mathbf{\Theta}) = \left[\prod_{t=0}^{T} \prod_{j} \prod_{j'} a_{jj'}^{q_{tj} q_{t+1j'}} \right] \left[\prod_{t=1}^{T} \prod_{j=1}^{M} b_j(o_t)^{q_{tj}} \right] . \tag{4.8}$$

It is worth noting, that the term $\prod_j \prod_{j'} a_{jj'}^{q_{tj} q_{t+1j'}}$ is equivalent to $a_{q_t q_{t+1}}$ since for any other pair of states $a_{jj'}^{q_{tj} q_{t+1j'}} = a_{jj'}^{0} = 1$. In a similar way, $b_{q_t}(o_t) = \prod_{j=1}^{M} b_j(o_t)^{q_{tj}}$. In fact this is the approach used in (4.4). The main difference between these two approaches is that in (4.8) $a_{jj'}$ refers to a parameter, while $a_{q_t q_{t+1}}$ refers to a function that selects a parameter given the values of q_t and q_{t+1}, which is a more compact expression but a more impractical one to perform mathematical operations.

At this point we are able to raise the EM algorithm for the HMM case. In the E step function $\mathcal{Q}(\mathbf{\Theta} \mid \mathbf{\Theta}^{(r)})$ (2.23) is written as

$$\mathcal{Q}(\mathbf{\Theta} \mid \mathbf{\Theta}^{(r)}) = E\big(\sum_{n} \log P(O_n, \mathbf{q}_n \mid \mathbf{\Theta}) \mid O_1^N, \mathbf{\Theta}^{(r)} \big) , \tag{4.9}$$

which, by applying to (4.8) the properties of the logarithm and expected value operators, can be rewritten as

$$\begin{aligned} \mathcal{Q}(\mathbf{\Theta} \mid \mathbf{\Theta}^{(r)}) = \sum_{n} & \left[\sum_{t=0}^{T} \sum_{j} \sum_{j'} (q_{ntj} q_{nt+1j'})^{(r)} \log a_{jj'} \right] \\ & \left[\sum_{t=1}^{T} \sum_{j=1}^{M} q_{ntj}^{(r)} \log b_j(o_{nt}) \right] \end{aligned} , \tag{4.10}$$

where $(q_{ntj} q_{nt+1j'})^{(r)}$ and $q_{ntj}^{(r)}$ are defined as

$$\begin{aligned} (q_{ntj} q_{nt+1j'})^{(r)} &= E(q_{ntj} q_{nt+1j'} \mid O_1^N, \mathbf{\Theta}^{(r)}) \\ &= \frac{P(q_{ntj} = 1, q_{nt+1j'} = 1, O_n \mid \mathbf{\Theta}^{(r)})}{P(O_n \mid \mathbf{\Theta}^{(r)})} , \end{aligned} \tag{4.11}$$

$$q_{ntj}^{(r)} = E(q_{ntj} \mid O_1^N, \mathbf{\Theta}^{(r)}) = \frac{P(q_{ntj} = 1, O_n \mid \mathbf{\Theta}^{(r)})}{P(O_n \mid \mathbf{\Theta}^{(r)})} . \tag{4.12}$$

Where $(q_{ntj} q_{nt+1j'})^{(r)}$ is the expected value of performing a transition between states j and j' at time t, while $q_{ntj}^{(r)}$ is the expected value of being j the current state at time t.

In the M step the set of parameters $\mathbf{\Theta}$ that maximizes (4.10) is taken as the new set of parameters. However, equation (4.10) is constrained to $\sum_{j'} a_{jj'} = 1$ and then we use Lagrange parameters as follows

$$\mathcal{L}(\mathbf{\Theta} \mid \mathbf{\Theta}^{(r)}) = \mathcal{Q}(\mathbf{\Theta} \mid \mathbf{\Theta}^{(r)}) - \sum_{j} \lambda_j \left(\sum_{j'} a_{jj'} - 1 \right) , \tag{4.13}$$

which still maximizes $\mathcal{Q}$ while respecting the constraints. Note that more constraints may be required depending on the definition of b_j. In a way, the E step can be interpreted as the calculation of the hidden variables given the old parameters, and the M step can be seen as a typical MLE estimation without hidden variables. Thus, using first derivatives over (4.13) $a_{jj'}^{(r+1)}$ are calculated as

$$a_{jj'}^{(r+1)} = \frac{\sum_n \sum_{t=0}^{T} (q_{ntj} q_{nt+1j'})^{(r)}}{\sum_n \sum_{t=0}^{T} q_{ntj}^{(r)}} = \frac{\langle N(j,j') \rangle}{\langle N(j) \rangle}\,, \tag{4.14}$$

where $\langle N(j,j') \rangle$ and $\langle N(j) \rangle$ are both expected values. $\langle N(j,j') \rangle$ refers to the number of times the transition between states j and j' has been completed, while $\langle N(j) \rangle$ refers to the number of transitions completed in which the source state was j, which is equivalent to the number of feature vectors generated in the state j. In a similar way, the parameters related to the probability observation (density) function $b_j(o)$ would be computed.

Nevertheless, the calculation of $(q_{ntj} q_{nt+1j'})^{(r)}$ and $q_{ntj}^{(r)}$ in E step requires the calculation of $P(O_n \mid \mathbf{\Theta}^{(r)})$ and other similar probabilities which can only be calculated by trying all possible values of Q. This task can be efficiently computed using the Baum-Welch re-estimation recursion, which raise the task as a dynamic programming problem. In this approach two recursive probabilities are raised, the forward probability and the backward one. The forward probability for each sample n, state j and time t, $\alpha_{nt}(j) = P(o_{n1}, \ldots, o_{nt}, q_{nt} = j \mid \mathbf{\Theta})$, is calculated as

$$\alpha_{nt+1}(j) = \begin{cases} b_j(o_{n1}) a_{Ij} & 1 \leq j \leq M, t = 0 \\ b_j(o_{nt+1}) \left[\sum_{i=1}^{M} \alpha_{nt}(i) a_{ij} \right] & 1 \leq j \leq M \\ & 1 \leq t < T_n \end{cases}, \tag{4.15}$$

while the backward probability, $\beta_{nt}(j) = P(o_{nt+1}, \ldots, o_{nT_n} \mid q_{nt} = j, \mathbf{\Theta})$, is calculated as

$$\beta_{nt}(j) = \begin{cases} a_{jF} & 1 \leq j \leq M, t = T_n \\ \sum_{j'=1}^{M} a_{jj'} b_{j'}(o_{nt+1}) \beta_{nt+1}(j') & 1 \leq j \leq M \\ & 1 \leq t < T_n \end{cases}. \tag{4.16}$$

Thus, the probability (density) of an observation can be calculated using forward probabilities

$$P(O_n \mid \mathbf{\Theta}) = \sum_{j=1}^{M} \alpha_{nT_n}(j) a_{jF}\,, \tag{4.17}$$

and equations (4.11) and (4.12) can be rewritten as

$$(q_{ntj} q_{nt+1j'})^{(r)} = \frac{\alpha_{nt}^{(r)}(j) a_{jj'}^{(r)} b_{j'}^{(r)}(o_{nt+1}) \beta_{nt+1}^{(r)}(j')}{P(O_n \mid \mathbf{\Theta}^{(r)})}\,, \tag{4.18}$$

$$q_{ntj}^{(r)} = \frac{\alpha_{nt}^{(r)}(j) \beta_{nt}^{(r)}(j)}{P(O_n \mid \mathbf{\Theta}^{(r)})}\,. \tag{4.19}$$

4.2.2 Search for the Most Likely State Transition Sequence

A common question when working with HMMs is; given an observation sequence $O = o_1, \ldots, o_T$ which is the most likely state sequence, $\mathbf{q} = (q_1, \ldots, q_T)$, that generates it? Or more formally,

$$\mathbf{q}^* = \underset{\mathbf{q}}{\operatorname{argmax}} \prod_{t=0}^{T} a_{q_t q_{t+1}} \prod_{t=1}^{T} b_{q_t}(o_t) \ . \tag{4.20}$$

This problem can be solved by using the well known Viterbi algorithm [Jelinek, 1997, Viterbi, 1967]. In the Viterbi algorithm the probability (or score) of the most likely path until time t that ends in state j is defined recursively as

$$D(t,j) = \begin{cases} a_{Ij} b_j(o_1) & t = 1 \\ \max\limits_{j'} \left[D(t-1,j') a_{j'j} \right] b_j(o_t) & t > 1 \end{cases} . \tag{4.21}$$

Thus the probability of the most likely state transition can be calculated as

$$D(T+1,F) = \max_{j'} D(T,j') \ a_{j'F} \ . \tag{4.22}$$

From (4.21) it is easy obtain a matrix which for each pair (t,j) holds a back-pointer to the previous state of the sequence

$$B(t,j) = \begin{cases} I & t = 1 \\ \underset{j'}{\operatorname{argmax}} \left[D(t-1,j') a_{j'j} \right] b_j(o_t) & t > 1 \end{cases} , \tag{4.23}$$

then the most likely state transition q^* can be obtained by calculating

$$B(T+1,F) = \underset{j'}{\operatorname{argmax}} D(T,j') \ a_{j'F} \ , \tag{4.24}$$

and following recursively the back-pointer until the state I is reached.

4.3 Bernoulli Hidden Markov Model (BHMM)

The definition of Bernoulli HMM and its Baum-Welch re-estimation formulae do not differ significantly from that of the conventional HMMs, based on either discrete (multinomial) probability functions or continuous (Gaussian) densities. As seen in Section 4.2, differences are only related to the definition and the parameter estimation of $b_j(o_t)$. In this section we only describe the basic differences. Please see Rabiner and Juang [1993] and Young et al. [1995] for more details about conventional HMMs based on continuous densities.

Let $O = (\mathbf{o}_1, \ldots, \mathbf{o}_T)$ be a sequence of D-dimensional binary *observation vectors* and let M be a set of states. A Bernoulli HMM is an HMM in which the probability of observing $\mathbf{o}_t$ in state j at time t follows a multivariate Bernoulli distribution with prototype $\mathbf{p}_j$, i.e.,

$$b_j(\mathbf{o}_t) = P(\mathbf{o}_t \mid q_t = j) = \prod_{d=1}^{D} p_{jd}^{o_{td}} \left(1 - p_{jd}\right)^{1 - o_{td}} , \tag{4.25}$$

where p_{jd} is the probability for bit d to be 1 when the observation vector is generated in state j. It is worth noting that (4.25) is just the product of conditionally independent unidimensional Bernoulli variables, and that consequently it cannot capture any kind of dependencies or correlations between individual bits. The parameter vector associated with state j, $\mathbf{p}_j = (p_{j1}, \ldots, p_{jD})^t$, is referred as the *prototype* of the Bernoulli distribution in state j.

One interesting property of Bernoulli prototypes lies in their direct visual interpretation. That is, each prototype of dimension $W \times H$ can be interpreted as a gray-scale picture of width W and height H, where $1 - p_{xy}$ is the color of pixel (x, y). Following this approach Figure 4.1 shows a visual example of a simple synthetic BHMM (1×2 prototypes). In this example an observation sequence, made up from one-dimensional feature vectors (top of image), is generated by a three state BHMM. In the Bernoulli prototypes, which are drawn in the middle of image, gray color is related to probability 0.5 and white color to probability 0. In the example, the first state is the responsible for generating the first three vectors, since the second state can not generate vectors with a black pixel on top. However, observations 4 and 5 can be generated by both states 2 and 3. In the example state 2 generates the observation 4 and performs a transition to the state 3 which generates the remaining observation vectors. It is easy to check that the probability of generating the observation sequence in the example, which is calculated by multiplying the transitions probabilities by the prototype ones, is 0.13608.

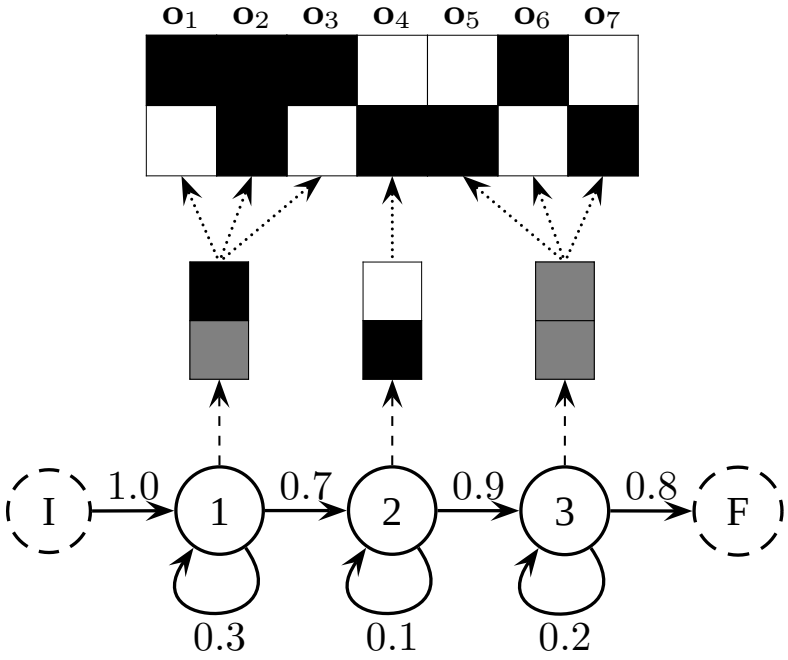


Figure 4.1: Visual example of a BHMM

Recall from Section 4.2.1 that Maximum likelihood estimation of the parameters governing an HMM can be carried out using the *EM algorithm* for HMMs; i.e. using *Baum-Welch (forward-backward)* re-estimation formulae. Assume that the likelihood is calculated with

respect to N sequences $O_1, \ldots, O_N$; with $O_n = (\mathbf{o}_{n1}, \ldots, \mathbf{o}_{nT_n})$ for all $n = 1, \ldots, N$. At the end of iteration r, the Bernoulli prototype corresponding to state j has to be updated as

$$\mathbf{p}_j^{(r+1)} = \frac{\sum_n \sum_{t=1}^{T_n} q_{ntj}^{(r)} \mathbf{o}_{nt}}{\sum_n \sum_{t=1}^{T_n} q_{ntj}^{(r)}} \,, \tag{4.26}$$

that is, each element of the Bernoulli prototype is updated as

$$p_{jd}^{(r+1)} = \frac{\sum_n \sum_{t=1}^{T_n} q_{ntj}^{(r)} o_{ntd}}{\sum_n \sum_{t=1}^{T_n} q_{ntj}^{(r)}} = \frac{\langle N(j,d) \rangle}{\langle N(j) \rangle} \,, \tag{4.27}$$

where $q_{ntj}^{(r)}$ is the expectation value of $\mathbf{o}_{nt}$ to be generated in state j, which it is calculated in E step as described in (4.12), and $\langle N(j,d) \rangle$ is the expected value of the number of binary feature vectors generated in state j for which the bit d is one.

Overtraining is a common problem of the MLE criterion, and in order to amend this problem, Bernoulli prototypes are smoothed by a linear interpolation with a uniform prototype, $\mathbf{0.5}$,

$$\tilde{\mathbf{p}} = (1 - \delta)\, \mathbf{p} + \delta\, \mathbf{0.5} \,, \tag{4.28}$$

where δ is usually optimized in a validation set. A typical value is $\delta = 10^{-6}$.

4.4 Experiments

In order to test the proposed model, experiments were carried out using two corpus based on real tasks: recognition of handwritten Arabic cheques and recognition of handwritten English text. The first corpus we used is the non-touching Arabic sub-words dataset from the CENPARMI Arabic cheque database, which is described in Section 3.1. The second corpus we used is the IAM word dataset, which, as previously mentioned in Section 3.2.1, is a handwritten English text dataset.

Experiments were carried out without applying any preprocess to input images, apart from those mandatory preprocessing steps related to the feature extraction. Therefore, all input images were scaled in height to the same size while maintaining the original aspect ratio. Different heights (D) were considered: 10 and 20. In addition, an Otsu's binarization was carried out on the *IAM* words dataset.

Since both datasets have a great variability of the number of samples per class, we carried out experiments with subsets from the original datasets. In the case of Arabic subwords dataset the ten most frequent subwords were selected, and then they were divided into training and testing sets respecting the original proportion. For the *IAM* words dataset, the classes with at least 50 samples were selected (this includes samples from the 657 writers), and for each class the 80% of samples were selected for training and the others for testing. In Table 4.1 some characteristics of the subsets are shown.

Experiments were carried out by varying number of states, $Q \in \{10, 20, 40, 80, 160, 320\}$. For each value of Q, all words were modeled using Bernoulli HMMs with the number of states fixed to Q. The Bernoulli HMMs were initialized using a left-to-right topology with

Table 4.1: Number of classes, number of training samples, number of testing samples and average aspect ratio in both testing and training samples, for the Arabic subword dataset with the ten most frequent subwords, and for the *IAM* words dataset with words that at least have 50 samples

	N. Classes	N. S. training	N. S. testing	A. A. Ratio
Arabic subwords	10	13106	5456	1.26 ± 0.66
IAM words	180	44492	11122	1.63 ± 1.00

skips as follows: for each training sample we define a state sequence $\mathbf{q}$, which is used to obtain initial parameters estimations using the MLE criterion. Specifically, for each training sample its binary feature vectors are distributed, from left to right at same distance each from other, over the states. After each Baum-Welch iteration, each Bernoulli prototype $\mathbf{p}$ was smoothed as explained in (4.28). For each class-conditional Bernoulli HMM, 10 Baum-Welch iterations were executed.

For each experiment, several repetitions were performed by means of randomly selecting testing and training sets, while respecting the original proportions for each class. With the Arabic subwords dataset about 10 repetitions for $Q = \{320\}$ and 30 repetitions for $Q = \{10, 20, 40, 80, 160\}$ were carried out. With the *IAM* words dataset about 2 repetitions for $Q = \{160, 320\}$ were carried out, for $Q = \{10, 20, 40, 80\}$ 10 and 5 repetitions were carried out for $D = 10$ and $D = 20$, respectively.

In Figure 4.2 the results for the Arabic subwords dataset are shown. The best result (10.9%) is obtained with $D = 20$ and $Q = 320$. For $D = 20$ the results could be improved by increasing the Q, however for $D = 10$ the best result is achieved with $Q = 160$. The lowest classification error (14.3%) is obtained with $D = 20$ and $Q = 160$.

In Figure 4.3 the results for the *IAM* words dataset are shown. As in Arabic subwords, the best results are obtained with $D = 20$, despite that better results are obtained with $D = 10$ for low values of Q. The best result obtained, 31.0%, is similar to the best result in Günter and Bunke [2004] using a single Gaussian density in each HMM state. It is worth noting, however, that we use an independent Bernoulli HMM for each class while, in Günter and Bunke [2004], each class-conditional continuous HMM is built from more elementary HMMs at character level.

4.5 Concluding Remarks

Bernoulli HMMs have been proposed for off-line handwriting recognition in order to directly model text image data in binary form. Empirical results have reported on two tasks of off-line handwritten word recognition: Arabic subwords from *CENPARMI* corpus, and English words from *IAM* database. In both cases each word (subword) was modeled with one HMM, and only the required preprocess to obtain binary images of same height was applied. Feature vectors of different sizes, as well as HMMs with different number of states, were tested. The results on the Arabic subwords task are promising. In the case of the *IAM* words, the results

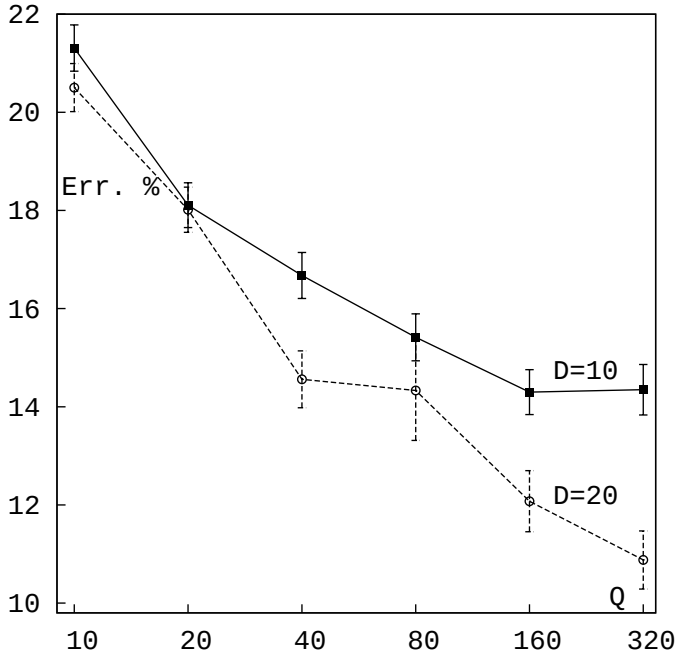


Figure 4.2: Error classification with different number of states (Q) and heights (D) for the Arabic subwords dataset, with the ten most frequent subwords and several repetitions for each point

were very similar to those obtained using HMMs with one Gaussian per state.

In next chapter we will focus on the use of Bernoulli HMMs at subword (character) level and extend them by using Bernoulli mixtures instead of single Bernoulli probability functions in each state. Moreover, a comparison with Gaussian HMMs recognizers will be presented.

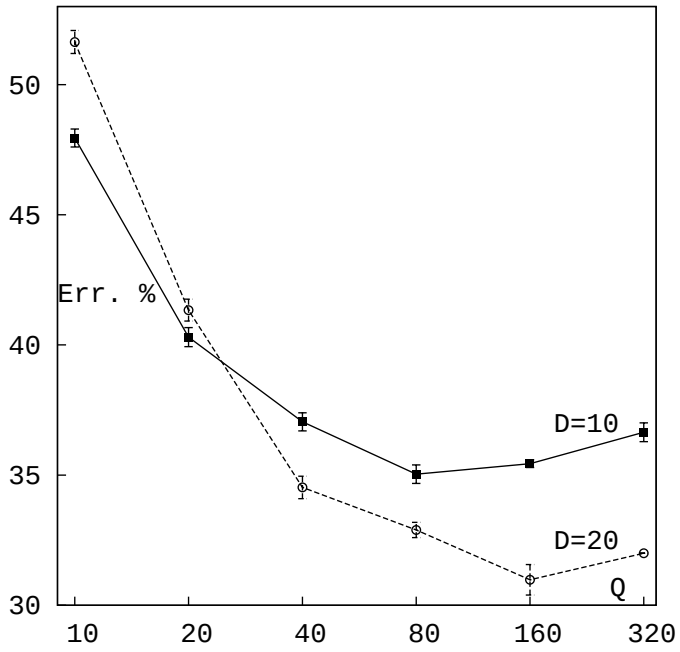


Figure 4.3: Error classification with different number of states (Q) and heights (D) for the *IAM* words dataset, with the words that have at least 50 samples and several repetitions for each point

Bibliography

A. Giménez-Pastor and A. Juan-Císcar. Bernoulli HMMs for Off-line Handwriting Recognition. In *Proc. of the 8th Int. Workshop on Pattern Recognition in Information Systems (PRIS 2008)*, pages 86–91, Barcelona (Spain), June 2008.

Simon Günter and Horst Bunke. HMM-based handwritten word recognition: on the optimization of the number of states, training iterations and Gaussian components. *Pattern Recognition*, 37:2069–2079, 2004.

Frederick Jelinek. *Statistical Methods for Speech Recognition*. MIT Press, 1997.

Réjean Plamondon and Sargur N. Srihari. On-Line and Off-Line Handwriting Recognition: A Comprehensive Survey. *IEEE Trans. on PAMI*, 22(1):63–84, 2000.

Lawrence Rabiner and Biing-Hwang Juang. *Fundamentals of speech recognition*. Prentice-Hall, 1993.

A. H. Toselli, A. Juan, D. Keysers, J. González, I. Salvador, H. Ney, E. Vidal, and F. Casacuberta. Integrated Handwriting Recognition and Interpretation using Finite-State Models. *International Journal of Pattern Recognition and Artificial Intelligence*, 18(4): 519–539, 2004.

A. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *Information Theory, IEEE Transactions*, 13(2):260 – 269, 1967.

Hanhong Xue and Venu Govindaraju. Hidden Markov Models Combining Discrete Symbols and Continuous Attributes in Handwriting Recognition. *IEEE Trans. on PAMI*, 28:458–462, 2006.

S. Young et al. *The HTK Book*. Cambridge University Engineering Department, 1995.

CHAPTER 5

EMBEDDED BERNOULLI MIXTURE HMMS

Contents

5.1 Introduction

We have proposed the use of Bernoulli HMMs for HTR in order to directly deal with binarized input images. Basic motivation is to better modeling the binary nature of text images, and to guarantee that no discriminative information is filtered out during feature extraction.

The direct method to model handwritten words with Bernoulli HMMs is to use an independent, separate Bernoulli HMM for each word. We did it in Chapter 4, where successful results were obtained in a task of word classification with a moderate number of (words) classes. However, this direct approach becomes impractical in the case of classification tasks involving a large number of classes, due to lack of training data for infrequent classes, which results in poorly estimated HMM parameters and degraded classifier performance. Following the usual approach in speech recognition [Jelinek, 1997, Rabiner and Juang, 1993], from where the HMM methodology was imported, word-conditional HMMs are instead built from shared, *embedded* HMMs at symbol (subword) level, that is, all word classes (sentences) are modeled by concatenation of subword (character) HMMs, and thus only one HMM per character has to be trained. In this way, each training word image contributes to the estimation of its constituent symbol HMMs, all symbol HMMs are reliably estimated, and infrequent words are better modeled.

HMMs at symbol level are usually simple in terms of number of states and topology; e.g., 6 states and a linear topology in which each state can only be reached from its preceding state or itself (loop). On the other hand, state-conditional probability (density) functions depend on the type of output that has to be emitted. In the common case of real-valued feature vectors, Gaussian mixtures are generally preferred since, as with finite mixture models in general, their complexity can be adjusted to the available training data by simply varying the number of mixture components. Another good reason for their use is the availability of reliable software from the speech recognition community [Young et al., 1995].

In this chapter, we presents the needed formulae to apply BHMMs in the embedded approach, and in addition, in a similar way that in the Gaussian HMMs case, we propose the substitution of multivariate Bernoulli probability functions by Bernoulli mixture ones, which have been proved to be a more flexible distribution [Juan and Vidal, 2004]. Empirical results are reported in which embedded Bernoulli HMMs and embedded Bernoulli mixture HMMs are compared with both, independent Bernoulli HMMs, and conventional embedded (Gaussian) HMMs [Günter and Bunke, 2004, Pastor i Gadea, 2007]. Furthermore, in this chapter we explore the use of a sliding window with BHMMs, which is a common technique used in other HTR systems based on Gaussian mixture HMMs [Dreuw et al., 2009]. In contrast to our basic approach, in which narrow, one-column slices of binary pixels are fed into BHMMs, now we use a sliding window of adequate width to better capture image context at each horizontal position of the word image.

The chapter is organized as follows. We first explain the use of BHMMs at subword level in Section 5.2. In Section 5.3 we review plain Bernoulli mixtures, while embedded Bernoulli mixture HMMs are presented in Section 5.4. In Section 5.5 we describe in detail how embedded Bernoulli mixture HMMs are used for handwritten word recognition, and in Section 5.6 the formulae required for maximum likelihood estimation is shown. Our basic extension to plain BHMMs, which will be referred as *windowed BHMMs*, is explained in Section 5.7. In Section 5.8, empirical results are reported. Final, concluding remarks are

discussed in Section 5.9.

5.2 Embedded BHMMs

As discussed in the introduction, BHMMs at global (line or word) level are built from shared, embedded BHMMs at character level. More precisely, let C be the number of different characters (symbols) from which global BHMMs are built, and assume that each character c is modeled with a different BHMM of parameter vector $\mathbf{\Theta}_c$. Let $\mathbf{\Theta} = \{\mathbf{\Theta}_1, \ldots, \mathbf{\Theta}_C\}$, and let $O = (\mathbf{o}_1, \ldots, \mathbf{o}_T)$ be a sequence of feature vectors generated from a sequence of symbols $S = (s_1, \ldots, s_L)$, with $L \leq T$. The probability of O can be calculated, using embedded HMMs for its symbols, as

$$P(O \mid S, \mathbf{\Theta}) = \sum_{i_1, \ldots, i_{L+1}} \prod_{l=1}^{L} P(\mathbf{o}_{i_l}, \ldots, \mathbf{o}_{i_{l+1}-1} \mid \mathbf{\Theta}_{s_l}) \,, \tag{5.1}$$

where the sum is carried out over all possible segmentations of O into L segments, that is, all sequences of indices $i_1, \ldots, i_{L+1}$ such that

$$1 = i_1 < \cdots < i_L < i_{L+1} = T + 1 \,; \tag{5.2}$$

and $P(\mathbf{o}_{i_l}, \ldots, \mathbf{o}_{i_{l+1}-1} \mid \mathbf{\Theta}_{s_l})$ refers to the probability (density) of the l-th segment, as given by (4.4) using the HMM associated with symbol s_l.

5.3 Bernoulli Mixture

Let $\mathbf{o}$ be a D-dimensional feature vector. A finite mixture is a probability (density) function of the form

$$P(\mathbf{o} \mid \mathbf{\Theta}) = \sum_{k=1}^{K} \tau_k \, P(\mathbf{o} \mid \mathbf{\Theta}_k) \,, \tag{5.3}$$

where K is the number of mixture components, τ_k is the kth component coefficient, and $p(\mathbf{o} \mid \mathbf{\Theta}_k)$ is the kth component-conditional probability (density) function. The mixture is controlled by a parameter vector $\mathbf{\Theta}$ comprising the mixture coefficients and a parameter vector for the components, $\mathbf{\Theta}_k$. It can be seen as a generative model that first selects the kth component with probability τ_k and then generates $\mathbf{o}$ in accordance with $p(\mathbf{o} \mid \mathbf{\Theta}_k)$.

A Bernoulli mixture model is a particular case of (5.3) in which each component k is modeled by a D-dimensional Bernoulli probability function, as has been defined in (4.25). Thus a Bernoulli mixture is a probability function defined as

$$p(\mathbf{o} \mid \mathbf{\Theta}) = \sum_{k=1} \tau_k \prod_d p_{kd}^{o_d} (1 - p_{kd})^{1 - o_d} \,. \tag{5.4}$$

Consider the example given in Figure 5.1. Three binary images (**a, b** and **c**) are shown as being generated from a Bernoulli prototype depicted as a gray image (black=1, white=0, Gray=0.5). The prototype has been obtained by averaging images **a** and **c,** and it is the

best approximate solution to assign a high, equal probability to these images. However, as individual pixel probabilities are not conditioned to other pixel values, there are $2^6 = 64$ different binary images (including **a, b** and **c**) into which the whole probability mass is uniformly distributed. It is then not possible, using a single Bernoulli prototype, to assign a probability of 0.5 to **a** and **c,** and null probability to any other image such as **b.** Nevertheless, this limitation is easily overcomed by using a Bernoulli mixture and allowing a different prototype to each different image shape. That is, in our example, a two-component mixture of equal coefficients, and prototypes **a** and **b,** does the job.

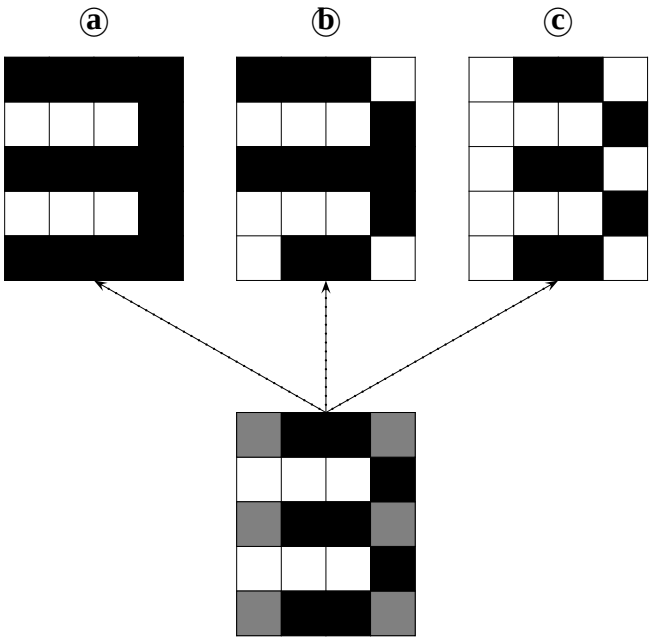


Figure 5.1: Three binary images (**a, b** and **c**) are shown as being generated from a Bernoulli prototype depicted as a gray image (black=1, white=0, gray=0.5)

5.4 Embedded Bernoulli Mixture HMMs

Embedded BHMMs can be extended to Embedded Bernoulli mixture HMMS by replacing the simple multivariate Bernoulli probability functions by Bernoulli mixture ones. That is, the observation probability function $b_j(\mathbf{o}_t)$ is now modeled as a Bernoulli mixture as described in (5.4)

$$b_j(\mathbf{o}_t) = \sum_{k=1}^{K} \tau_{jk} \prod_{d=1}^{D} p_{jkd}^{o_{td}} \left(1 - p_{jkd}\right)^{1-o_{td}} . \tag{5.5}$$

In Figure 5.2 two embedded Bernoulli mixture HMMs modeling the numbers 3 and 31 are shown. The bottom model, which is modeling the number 31, is the result of concatenating

Bernoulli mixture HMMs for digit 3, blank space, and digit 1, in that order. Note that the BHMMs for blank space and digit 1 are simpler than that for digit 3. The binary image of the number 31 shown above can only be generated from two paths, as indicated by the arrows connecting prototypes to image columns, which only differ in the state generating the second image column (either state 1 or 2 of the BHMM for the first symbol). It is straightforward to check that, according to (5.1) and (5.4), the probability of generating this image is 0.0004.

Consider now the top part of Figure 5.2, the BHMM used to model the number 3 is the same that the one used on the bottom model. Also note, that the BHMM modeling the number 3 can model two different kinds of 3, in top a number 3 ended with a vertical stroke is generated, while in bottom the number 3 is rounded on the right part, and no more possible ways to end the number 3 are possible. As mentioned in the previous section this is due to the fact that Bernoulli mixtures are used instead of simple Bernoullis, in which is not possible to generate both endings without allowing other type of endings.

It is worth noting that a simple BHMM is a particular case of Bernoulli mixture HMMs using embedded models, since it can be understood as a Bernoulli mixture HMM made up of an unique embedded BHMM with one mixture component per state. Thus, in advance we will refer to embedded Bernoulli mixture HMMs as BHMMs, and only when required, or when we want to remark some aspects of the model, we will use the terms: simple BHMM, Bernoulli mixture HMM, or embedded BHMM.

5.5 BHMM-based Handwriting Word Recognition

Given an observation sequence $O = (\mathbf{o}_1, \ldots, \mathbf{o}_T)$, its most probable transcription is obtained by application of the conventional Bayes decision rule

$$S^* = \underset{S \in W}{\operatorname{argmax}} \; P(S \mid O) \tag{5.6}$$

$$= \underset{S \in W}{\operatorname{argmax}} \; P(S, O) \tag{5.7}$$

$$= \underset{S \in W}{\operatorname{argmax}} \; P(S)\, P(O \mid S)\,, \tag{5.8}$$

where W is the set of possible transcriptions; $P(S)$ is usually approximated by an *n-gram language model* [Goodman, 2001]; and $P(O \mid S)$ is a *text image model* which, in this chapter, is modeled as a BHMM (built from shared, embedded BHMMs at character level), as defined in (5.1). An interesting case arises when the set of possible transcriptions reduces to a (small) finite set of *words (class labels).* In this case, $P(S)$ is simply the *prior* probability of the word S, while $P(O \mid S)$ is the probability of observing O when it is known that the handwritten word is S.

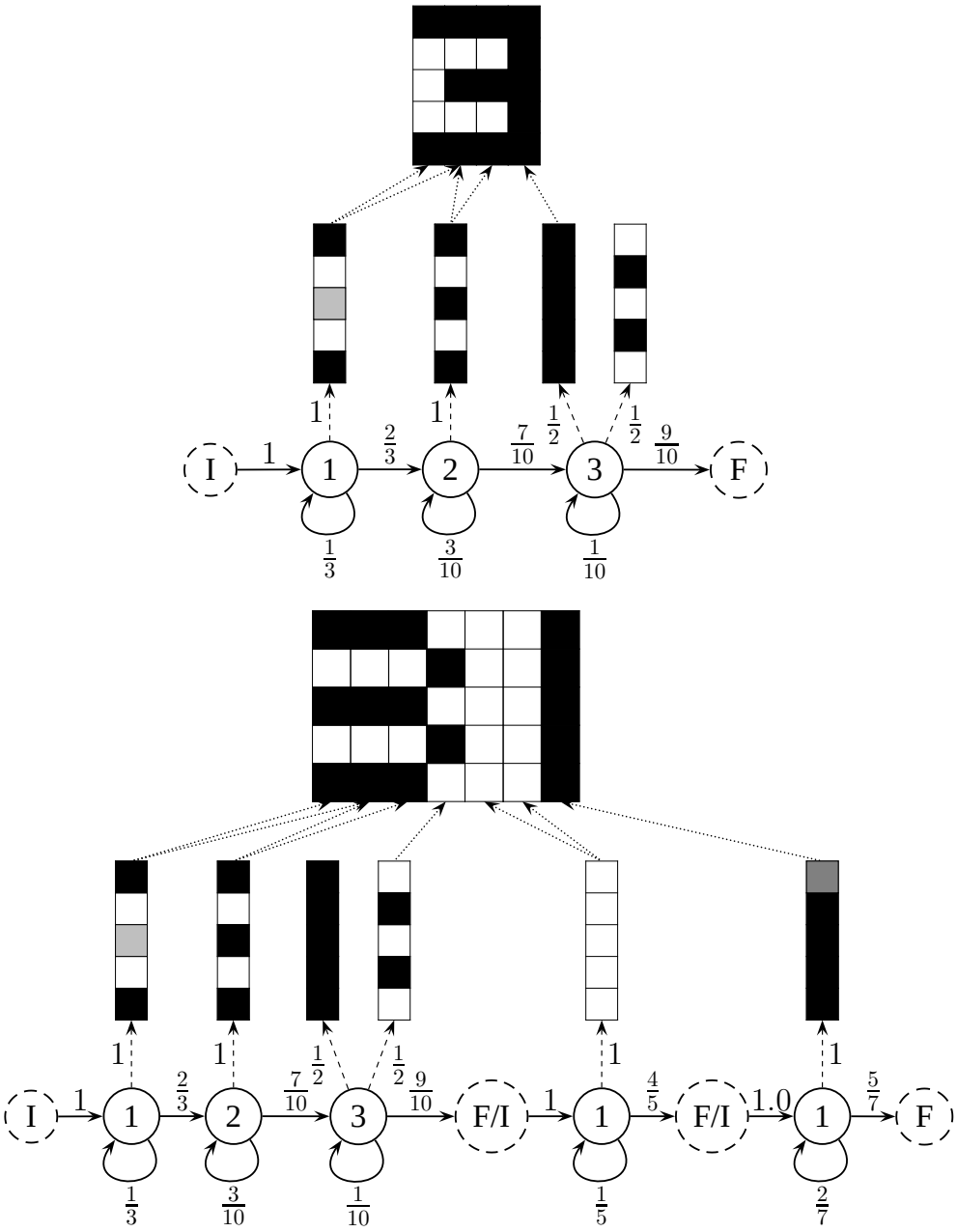


Figure 5.2: Bernoulli mixture HMM examples for the numbers 3 (top) and 31 (bottom), together with binary images generated from them. Note that the Bernoulli mixture HMM example for the number 3 is also embedded into that for the number 31. Bernoulli prototype probabilities are represented using the following color scheme: black=1, white=0,gray=0.5 and light gray=0.1

5.5.1 The Forward Algorithm

As with conventional BHMMs (4.17), we use dynamic programming in order to efficiently compute $P(O \mid S)$ as a BHMM probability of the form given in (5.1). For each time t, symbol s_l, and state j we define the forward probability for the embedded case $\alpha_{lt}(j)$ as

$$\alpha_{lt}(j) = p_{\boldsymbol{\theta}}(O_1^t, q_t = (l, j) \mid S)\,, \tag{5.9}$$

that is, the probability of generating O up to its tth element and ending at state j of the BHMM related to symbol s_l. This definition includes (5.1) as the particular case in which $t = T$, $l = L$ and $j = F$; that is,

$$p_{\boldsymbol{\theta}}(O \mid S) = \alpha_{LT(F)}\,. \tag{5.10}$$

To compute $\alpha_{LT(F)}$ we must first take into account that, for each position l in S, except for the first, the initial state of the BHMM for s_l is joined with final state of its preceding BHMM, i.e.

$$\alpha_{lt}(I) = \alpha_{l-1t}(F) \qquad \begin{matrix} 1 < l \leq L \\ 1 \leq t \leq T \end{matrix}\ . \tag{5.11}$$

Having (5.11) in mind, we can proceed at symbol level as with conventional BHMMs (4.15). In the case of final states, we have

$$\alpha_{lt}(F) = \sum_{i=1}^{M_{s_l}} \alpha_{lt}(i) a_{s_l i F} \qquad \begin{matrix} 1 \leq l \leq L \\ 1 \leq t \leq T \end{matrix}\ , \tag{5.12}$$

while, for regular states, $1 \leq j \leq M_{s_l}$, we have

$$\alpha_{lt}(j) = \left[\sum_{i \in \{I, 1, \ldots, M_{s_l}\}} \alpha_{lt-1}(i) a_{s_l i j} \right] b_{s_l j}(\mathbf{o}_t)\,, \tag{5.13}$$

with $1 \leq l \leq L$ and $1 < t \leq T$. The base case is for $t = 1$

$$\alpha_{l1}(i) = \begin{cases} a_{s_1 I i}\ b_{s_1 i}(\mathbf{o}_1) & l = 1, 1 \leq i \leq M_{s_1} \\ 0 & \text{otherwise} \end{cases}\ . \tag{5.14}$$

The forward algorithm uses a dynamic programming table for $\alpha_{lt}(\cdot)$ which is computed forward in time to avoid repeated computations.

Figure 5.3 shows an application example of the forward algorithm to the BHMM and observation of Figure 5.2 (bottom). Non-null (and a few null) entries of the dynamic programming table are represented by graph nodes aligned with states (vertically) and time (horizontally). Node borders are drawn in black or gray, depending on whether they are in valid paths (i.e. those from which the observation sequence can be generated) or not. Also, those associated with special states are drawn with dotted lines. Numbers at the top of each node refer to $\alpha_{lt}(\cdot)$ and thus, for instance, the probability of generating O up to the third image column and ending at state 2 of the BHMM for the first symbol is $\alpha_{13}(2) = \frac{10}{450}$. Computation dependencies between nodes are represented by arrows, which are labeled

above by, first, the transition probability, and then the observation probability at the target state (see (5.4)). For instance, the numbers above the arrow pointing to node $\alpha_{13}(4)$ are: $a_{s_1 23} \cdot b_{s_1 3}(\mathbf{o}_4) = \frac{7}{10} \cdot (\frac{1}{2} \cdot 0 + \frac{1}{2} \cdot 1^5) = \frac{7}{10} \cdot \frac{1}{2}$.

From Figure 5.3, we can clearly see that, as indicated in Section 5.4, there are only two paths from which the observation can be generated. They share all nodes drawn with black borders except the two nodes aligned with the second observation vector. In accordance with (5.10), the probability of the observation sequence is $\alpha_{37}(F) = 0.0004$.

5.5.2 The Backward Algorithm

The *backward algorithm* is similar to the forward algorithm but, as it name indicates, it uses a dynamic programming table which is computed backward in time [Rabiner and Juang, 1993, Young et al., 1995]. The basic definition in this case is the *backward probability*

$$\beta_{lt}(j) = p_{\boldsymbol{\theta}}(O_{t+1}^T \mid q_t = (l, j), S)\,, \tag{5.15}$$

which measures the probability of generating O_{t+1}^T given that the tth vector was generated in the state j of the BHMM related to symbol s_l. Using this definition (5.1) can be rewritten as

$$p_{\boldsymbol{\theta}}(O \mid S) = \sum_{j=1}^{M_{s_1}} a_{s_1 I j} b_{s_1 j}(\mathbf{o}_1)\beta_{11}(j)\ , \tag{5.16}$$

Taking into account again that the final state of each BHMM is joined with the final state of its successor BHMM we have that

$$\beta_{lt}(F) = \beta_{l+1t}(I) \qquad \begin{matrix} 1 \leq l < L \\ 1 \leq t < T \end{matrix}\ , \tag{5.17}$$

therefore the backward probability for the initial and the regular states, $i \in \{I, 1, \ldots, M_{s_l}\}$, can be efficiently computed as

$$\beta_{lt}(i) = a_{s_l i F}\beta_{lt}(F) + \sum_{j=1}^{M_{s_l}} a_{s_l ij} b_{s_l j}(\mathbf{o}_{t+1})\beta_{lt+1}(j) \qquad \begin{matrix} 1 \leq l \leq L \\ 1 \leq t < T \end{matrix}\ , \tag{5.18}$$

where the base case is defined for $t = T$ as

$$\beta_{lT}(i) = \begin{cases} a_{s_L i F} & l = L, 1 \leq i \leq M_{s_L} \\ 0 & \text{otherwise} \end{cases}\ . \tag{5.19}$$

5.5.3 The Viterbi Algorithm

Although the forward and backward algorithms efficiently compute the exact value of $p_{\boldsymbol{\theta}}(O \mid S)$, it is common practice to approximate it by the so-called *Viterbi* or *maximum approximation*, in which the sums in (4.4) and (5.1) are replaced by the $\max$ operator, i.e.

$$p_{\boldsymbol{\theta}}(O \mid S) \approx \max_{\substack{i_1, \ldots, i_{L+1} \\ q_1, \ldots, q_T}} \prod_{l=1}^{L} \hat{p}_{\boldsymbol{\theta}_{s_l}}(\mathbf{o}_{i_l}^{i_{l+1}-1}, q_{i_l}^{i_{l+1}-1})\,, \tag{5.20}$$

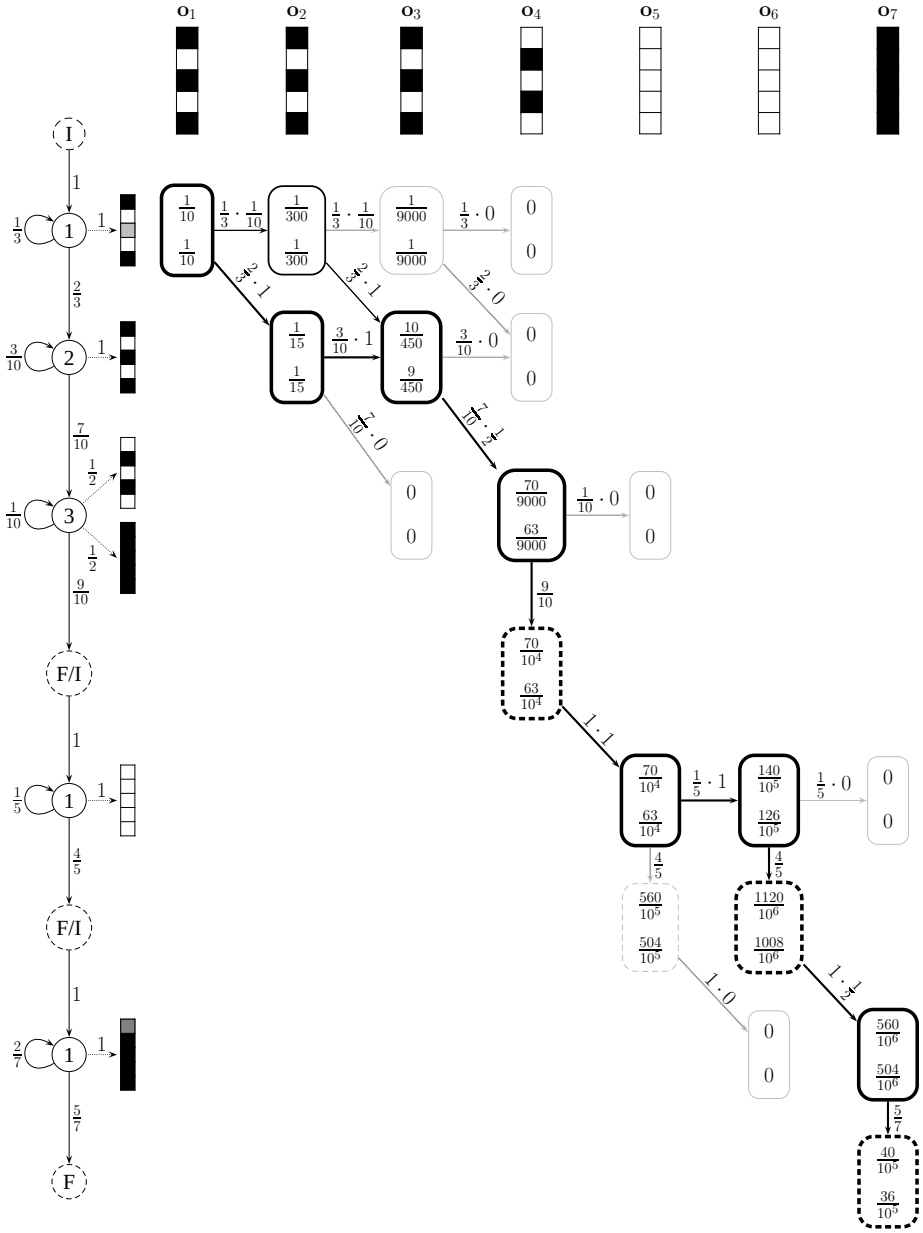


Figure 5.3: Application example of the forward and Viterbi algorithms to the the BHMM and observation of Figure 5.2 (bottom). Numbers at the top of the nodes denote forward probabilities, while those at the bottom refer to Viterbi scores

where the $\hat{p}_{\boldsymbol{\theta}_{s_l}}$ is defined as

$$\hat{p}_{\boldsymbol{\theta}_{s_l}}(\mathbf{o}_{i_l}^{i_{l+1}-1}, q_{i_l}^{i_{l+1}-1}) = a_{s_l I q_{i_l}} \cdot \prod_{t=i_l}^{i_{l+1}-2} a_{s_l q_t q_{t+1}} \cdot a_{s_l q_{i_{l+1}-1} F} \cdot \prod_{t=i_l}^{i_{l+1}-1} b_{s_l q_t}(\mathbf{o}_t)\,. \tag{5.21}$$

In contrast to the exact definition, this approximation allows us to identify a single, best state sequence or *path* associated with the given observation sequence. The well-known *Viterbi algorithm* efficiently computes this approximation using dynamic programming recurrences similar to those used by the forward algorithm. Formally, we need to compute the probability $Q_{lt}(j)$ of the most likely path up to time t that ends with the state j from the BHMM for symbol s_l. For the specials states, it can be computed as

$$Q_{lt}(I) = Q_{l-1t}(F) \qquad \begin{matrix} 1 \leq l \leq L \\ 1 \leq t \leq T \end{matrix}\ , \tag{5.22}$$

$$Q_{lt}(F) = \max_{1 \leq j \leq M_{s_l}} Q_{lt}(j)\, a_{s_l j F} \qquad \begin{matrix} 1 \leq l \leq L \\ 1 \leq t \leq T \end{matrix}\ . \tag{5.23}$$

while, for the regular states with $1 \leq l \leq L$ and $1 < t \leq T$, we have

$$Q_{lt}(j) = \left[\max_{i \in \{I, 1, \ldots, M_{s_l}\}} Q_{lt-1}(i)\, a_{s_l ij}\right] b_{s_l j}(\mathbf{o}_t)\,, \tag{5.24}$$

The base case is defined for $t = 1$

$$Q_{l1}(i) = \begin{cases} a_{s_1 I i}\; b_{s_1 i}(\mathbf{o}_1) & l = 1, 1 \leq i \leq M_{s_1} \\ 0 & \text{otherwise} \end{cases}\ . \tag{5.25}$$

Clearly, the Viterbi algorithm can be seen as a minor modification of the forward algorithm in which only the most probable path is considered in each node computation. Indeed, the application example shown in Figure 5.3 is used both, for the forward and Viterbi algorithms. Now, however, the relevant numbers are those included at the bottom of each node, which denote $Q_{lt}(j)$; i.e., at row 2 and column 3, we have $Q_{1\,3}(2) = \frac{9}{450}$. Consider the generation of the third observation vector at the second state (for the first symbol). It occurs after the generation of the second observation vector, either at the first or the second states, but we only take into account the most likely case. Specifically, the corresponding Viterbi score is computed as

$$Q(1,3,2) = \max\left\{\frac{1}{15} \cdot \frac{3}{10} \cdot 1, \frac{1}{300} \cdot \frac{2}{3} \cdot 1\right\} = \max\left\{\frac{9}{450}, \frac{1}{450}\right\} = \frac{9}{450}\,.$$

Note that forward probabilities do not differ from Viterbi scores up to $Q(1,3,2)$, since it corresponds to the first (and only) node with two incoming paths. The Viterbi approximation to the exact probability of generating the observation sequence is obtained at the final node: $Q(3,7,F) = 0.00036$. The most likely path, drawn with thick lines, is retrieved by starting at this node and moving backwards in time in accordance with computation of Viterbi scores. The final Viterbi score in this example (0.00036) is a tight lower bound of the exact probability (0.00040). In practice, it usually happens that this is the case and the Viterbi algorithm provides tight bounds on the exact probability.

5.6 Maximum Likelihood Parameter Estimation

Maximum likelihood estimation of the parameters governing an BHMM made up of embedded BHMMs does not significantly differ from the non embedded case. The well-known EM (Baum-Welch) re-estimation formulae can be directly applied as described in Section 4.2.1 and Section 4.3 since for each word we can build a virtual BHMM by concatenating the BHMMs related to its symbols, but taking into account that the parameters are defined at symbol level. Thus, let $(O_1, S_1), \ldots, (O_N, S_N)$, be a collection of N training samples in which the nth observation has length T_n, $O_n = (\mathbf{o}_{n1}, \ldots, \mathbf{o}_{nT_n})$, and was generated from a sequence of L_n symbols ($L_n \leq T_n$), $S_n = (s_{n1}, \ldots, s_{nL_n})$. At iteration r, in the E step the expected values of the hidden variables $(q_{ntcj}q_{nt+1cj'})^{(r)}$ and $q_{ntcj}^{(r)}$, which are defined in (4.18) and (4.19), are rewritten for the embedded case as

$$(q_{ntcj}q_{nt+1cj'})^{(r)} = \sum_{l:s_{nl}=c} \frac{\alpha_{nlt}^{(r)}(j)a_{cjj'}^{(r)}b_{cj'}^{(r)}(o_{nt+1})\beta_{nlt+1}^{(r)}(j')}{P(O_n \mid S_n, \boldsymbol{\theta}^{(r)})}\,, \tag{5.26}$$

$$q_{ntcj}^{(r)} = \sum_{l:s_{nl}=c} \frac{\alpha_{nlt}^{(r)}(j)\beta_{nlt}^{(r)}(j)}{P(O_n \mid S_n, \boldsymbol{\theta}^{(r)})}\,, \tag{5.27}$$

where the forward and backward probabilities are defined in (5.9) and (5.15), and subindex cj denotes the jth regular state of the BHMM modeling the symbol c. Note that the only difference between the embedded and the non embedded case is that a state from a BHMM modeling a particular symbol (c, j) can appear modeling several virtual states (l, j), hence an expectation value related to a symbol c is calculated as the sum over all virtual states for which $s_{nl} = c$ is true.

Bernoulli mixture components require the introduction of a new set of hidden variables $z_{tljk} \in \{0, 1\}$, where $z_{tljk} = 1$ denotes that the tth feature vector has been generated by the kth mixture component of the state j of BHMM related to symbol s_l. If the EM formulae is developed for embedded Bernoulli mixture HMMs, in a similar way as we do in Section 4.2.1, we realize that at E step, at iteration r, the expected value $(q_{ntcj}z_{ntcjk})^{(r)}$ may also be calculated as

$$\begin{aligned}(q_{ntcj}z_{ntcjk})^{(r)} &= \sum_{l:s_{nl}=c} \frac{\alpha_{nlt}^{(r)}(j)\tau_{cjk}^{(r)}\prod_{d=1}^{D} {p_{cjkd}^{(r)}}^{o_{ntd}}(1-p_{cjkd}^{(r)})^{1-o_{ntd}}\beta_{nlt}^{(r)}(j)}{P(O_n \mid S_n, \boldsymbol{\theta}^{(r)})b_{cj}^{(r)}(\mathbf{o}_{nt})} \\ &= \frac{\prod_{d=1}^{D} {p_{cjkd}^{(r)}}^{o_{ntd}}(1-p_{cjkd}^{(r)})^{1-o_{ntd}}}{b_{cj}^{(r)}(\mathbf{o}_{nt})} \cdot \sum_{l:s_{nl}=c} \frac{\alpha_{nlt}^{(r)}(j)\beta_{nlt}^{(r)}(j)}{P(O_n \mid S_n, \boldsymbol{\theta}^{(r)})} \\ &= \frac{\prod_{d=1}^{D} {p_{cjkd}^{(r)}}^{o_{ntd}}(1-p_{cjkd}^{(r)})^{1-o_{ntd}}}{b_{cj}^{(r)}(\mathbf{o}_{nt})} \cdot q_{ntcj}^{(r)}\,.\end{aligned} \tag{5.28}$$

In the M step, transition parameters of $\boldsymbol{\theta}_c$ are updated as described in (4.14), but using the expected values computed in (5.26) and (5.27). While the Bernoulli prototype corresponding

to the kth component of the state j in the BHMM for character c has to be updated as

$$\mathbf{p}_{cjk}^{(r+1)} = \frac{\sum_n \sum_{t=1}^{T_n} (q_{ntcj} z_{ntcjk})^{(r)} \mathbf{o}_{nt}}{\sum_n \sum_{t=1}^{T_n} (q_{ntcj} z_{ntcjk})^{(r)}} \ , \tag{5.29}$$

that is, each element of the Bernoulli protoype is updated as

$$p_{cjkd}^{(r+1)} = \frac{\sum_n \sum_{t=1}^{T_n} (q_{ntcj} z_{ntcjk})^{(r)} o_{ntd}}{\sum_n \sum_{t=1}^{T_n} (q_{ntcj} z_{ntcjk})^{(r)}} = \frac{\langle N(c,j,k,d) \rangle}{\langle N(c,j,k) \rangle} \ , \tag{5.30}$$

where $\langle N(c,j,k) \rangle$ and $\langle N(c,j,k,d) \rangle$ are both expected values. $\langle N(c,j,k) \rangle$ refers to the number of binary feature vectors generated in the kth mixture component from the jth state of the symbol c, and $\langle N(c,j,k,d) \rangle$ refers to the number of those generated binary feature vectors in which the bit d is one. Equation (5.29) is similar to (4.26) but using the new expected values (5.28). Similarly, the kth component coefficient of the state j in the BHMM for character c has to be updated as

$$\tau_{cjk}^{(r+1)} = \frac{\sum_n \sum_{t=1}^{T_n} (q_{ntcj} z_{ntcjk})^{(r)}}{\sum_n \sum_{t=1}^{T_n} q_{ntcj}^{(r)}} = \frac{\langle N(c,j,k) \rangle}{\langle N(c,j) \rangle} \ . \tag{5.31}$$

where $\langle N(c,j) \rangle$ is the expected value of the number of feature vectors generated in the jth state of symbol c.

5.7 Windowed BHMMs

Given a binary image normalized in height to H pixels, we may think of a feature vector $\mathbf{o}_t$ as its column at position t or, more generally, as a concatenation of columns in a window of W columns in width, centered at position t. This generalization has no effect neither on the definition of BHMM nor on its maximum likelihood estimation, though it might be very helpful to better capture image context at each horizontal position of the image. As an example, Figure 5.4 shows a binary image of 4 columns and 5 rows, which is transformed into a sequence of 4 15-dimensional feature vectors (first row) by application of a sliding window of width 3. For clarity, feature vectors are depicted as 3×5 subimages instead of 15-dimensional column vectors. Note that feature vectors at positions 2 and 4 would be indistinguishable if, as in our previous approach, they were extracted with no context ($W = 1$).

Although one-dimensional, "horizontal" BHMMs for image modeling can properly capture non-linear horizontal image distortions, they are somewhat limited when dealing with vertical image distortions, and this limitation might be particularly strong in the case of feature vectors extracted with significant context. To overcome this limitation, we have considered three methods of window *repositioning* after window extraction: *vertical*, *horizontal*, and *both*. The basic idea is to first compute the center of mass of the extracted window, which is then repositioned (translated) to align its center to the center of mass. This is done in accordance with the chosen method, that is, horizontally, vertically, or in both directions.

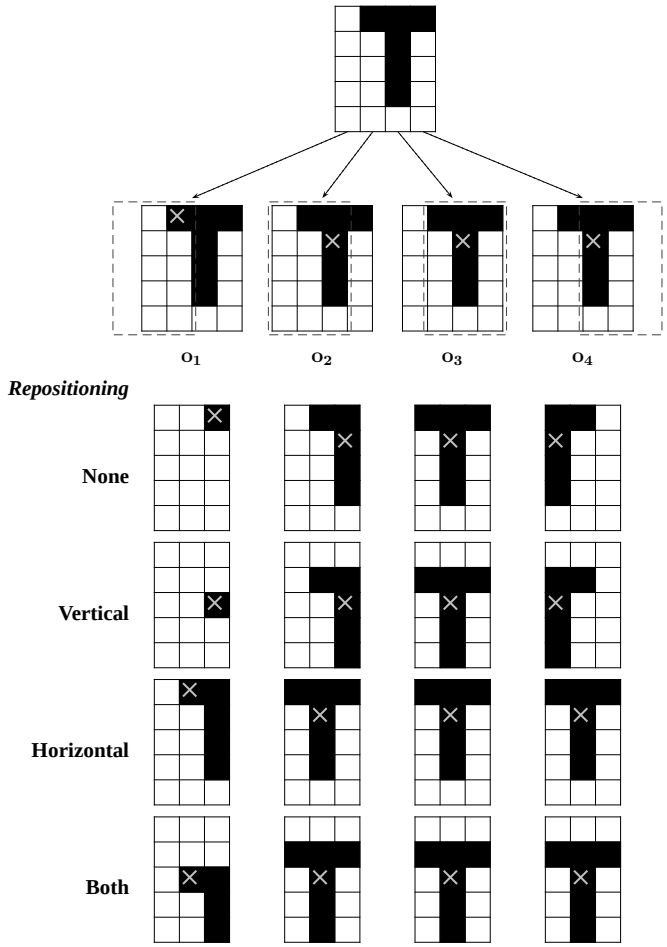


Figure 5.4: Example of transformation of a 4×5 binary image (bottom) into a sequence of 4 15-dimensional binary feature vectors $O = (\mathbf{o}_1, \mathbf{o}_2, \mathbf{o}_3, \mathbf{o}_4)$ using a window of width 3. The standard method (no repositioning) is compared with the three repositioning methods considered: vertical, horizontal, and both directions

Obviously, the feature vector actually extracted is that obtained after repositioning. An example of feature extraction is shown in Figure 5.4 in which the the standard method (no repositioning) is compared with the three methods repositioning methods considered.

To illustrate the effect of repositioning with real data, Figure 5.5 shows the sequence of feature vectors extracted from a real sample of the IFN/ENIT database, with and without (both) repositioning. As intended, (vertical or both) repositioning has the effect of normalizing vertical image distortions, especially translations.

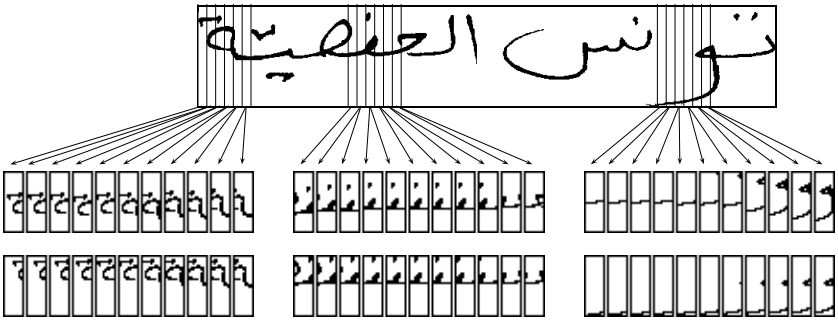


Figure 5.5: Original sample *pf069_011* from IFN/ENIT database (top) and its sequence of feature vectors produced with and without (both) repositioning (center and bottom, respectively)

5.8 Experiments

Experiments were carried out using three different datasets for isolated handwritten word recognition: the IAM word dataset (see Section 3.2.1), the IFN/ENIT database (see Section 3.3) and the RIMES word dataset (see Section 3.4). We have organized this section in six subsections. We begin the section carrying out experiments in order to assess the behavior of embedded BHMMs. Then, these experiments are extended in the second subsection by introducing Bernoulli mixture HMMs. In the third and fourth subsections, windowed BHMMs without repositioning, which extend Bernoulli mixture HMMs, and with repositioning are respectively tested on the IFN/ENIT database. Finally, according to the very good results obtained in IFN/ENIT, in the last two subsections we report new results on IAM and RIMES database using windowed BHMMs with repositioning techniques.

5.8.1 Embedded BHMMs

Experiments were carried over the IAM word dataset using embedded BHMMs. Apart from the BHMM results, we also report comparative results using embedded Gaussian HMMs. Regarding the dataset, we selected those samples which are marked as correctly segmented in the corpus, and which belong to a word with at least 10 samples. All input gray level images were then preprocessed before transforming them into sequences of feature vectors. Preprocessing consisted of three steps: gray level normalization, deslanting, and size normalization of ascenders and descenders [Pastor i Gadea, 2007].

Selected samples were randomly split into 30 80%-20% training-test partitions at the writer level to ensure writer-independent testing. This means about 59000 samples for training and 14000 for testing. The lexicon comprises 1117 different words and the alphabet is composed by 71 characters (upper and lowercase letters, punctuation signs, digits, etc.). This task is similar to that described in Günter and Bunke [2004].

For the Bernoulli system, feature extraction was carried out by rescaling the image to height 30 while respecting the original aspect ratio, and applying an Otsu's binarization to the resulting image. Therefore, the observation sequence is in fact a binary image of height

30. In the Gaussian case, the feature vector dimension is 60, where the first 20 values are gray levels, and the other 40 are horizontal and vertical gray level derivatives [Pastor i Gadea, 2007]. In this case, we used the well-known HTK software [Young et al., 1995].

Experiments were carried out by varying number of states, $Q \in \{4, 6, 8, 10, 12\}$, and comparing our Bernoulli system to a conventional system based on Gaussian HMMs. Both systems were initialized by first segmenting the training set using a "neutral" model analogous to that in Young et al. [1995], and then using the resulting segments to perform a Viterbi initialization. The model was trained with 4 EM iterations, and the recognition was performed using the Viterbi algorithm. As in conventional HMM systems [Young et al., 1995], the Viterbi algorithm was used in combination with a table of prior probabilities so as to find the most probable transcription (class) of each test image. Figure 5.6 shows the results, where each point is the average of 30 repetitions (30 random splits). Vertical bars denote $\pm$ standard deviation multiplied by two.

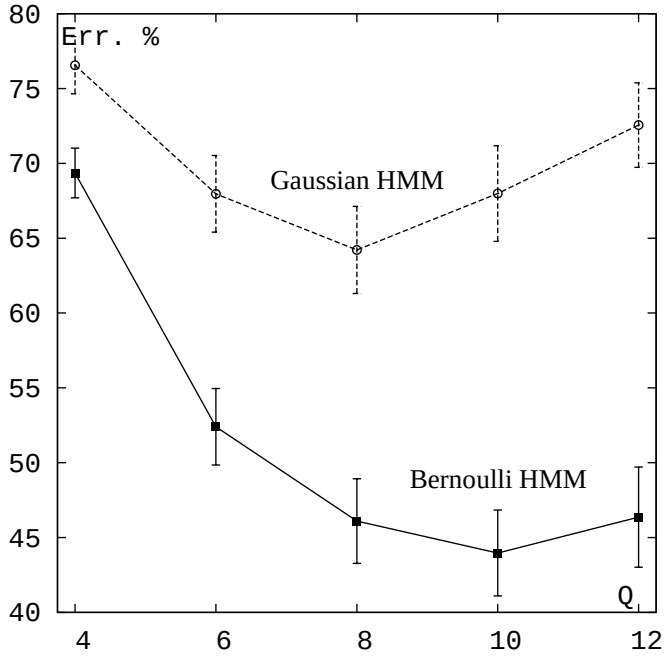


Figure 5.6: Classification error (in %) as a function of the number of states for the Bernoulli HMM system without mixtures and the conventional, Gaussian HMM system

The results obtained with the Bernoulli system are much better than those given by the Gaussian system. In particular, the best result for the Bernoulli system is a 44.0% classification error, obtained with $Q = 10$. In contrast, the best result for the Gaussian system is a 64.2% classification error, obtained with $Q = 8$.

We extended the experiment by using a different number of states for each HMM. To

decide the number of states for each character, we first Viterbi-segmented all training data using BHMMs of 3 states for punctuation signs and 6 states for other ones, and then computed the average length of the segments associated with each character. Given an average segment length $\bar{T}_c$ for character c, its number of states was set to $F \cdot \bar{T}_c$, where F is a *factor* measuring the average number of states that are required to emit a feature vector. Thus, its inverse, $\frac{1}{F}$, can be interpreted as a *state load*, that is, the average number of feature vectors that are emitted in each state. For instance, $F = 0.2$ means that only a fraction of 0.2 states is required to emit a feature vector or, alternatively, that $\frac{1}{0.2} = 5$ feature vectors are emitted on average in each state. The results obtained are very similar to those reported above; i.e. the Bernoulli system outperforms the Gaussian system with error rates similar to those in Figure 5.6. In both systems the best results are obtained with $F = 0.4$. Specifically, a classification error of 43.6% is obtained using BHMMs, while the classification error obtained using the Gaussian system is 61.9%.

We concluded these experiments by repeating those shown in Figure 5.6, but using one BHMM per word instead of one BHMM per character while (approximately) keeping the same number of parameters. Using BHMMs at word level and $Q = 1$ (1117 Bernoulli prototypes) a classification error of 89.3% was achieved, while with embedded BHMMs and $Q = 10$ (710 Bernoulli prototypes) we had a classification error of 44.0%. Moreover, using BHMMs at word level and $Q = 10$ (11170 Bernoulli prototypes) the classification error is 64%; that is, it is still not better than that obtained with embedded BHMMs.

5.8.2 Embedded Bernoulli mixture HMMs

In this subsection, the experiments from the previous subsection are extended by replacing the conventional Bernoulli models by Bernoulli mixture models. Therefore, experiments were carried out over the IAM word dataset, by varying number of states, $Q \in \{4, 6, 8, 10, 12\}$, varying the number of mixture components per state $K \in \{1, 4, 16, 64\}$, and comparing our Bernoulli recognizer with a conventional recognizer based on (embedded) Gaussian mixture HMMs. For $K = 1$, both recognizers were initialized as in the previous subsection. For $K > 1$, both recognizers were initialized by splitting the mixture components of the trained model with $K/4$ mixture components per state. Models have been trained with 4 EM iterations, and the recognition was performed using the Viterbi algorithm. Figure 5.7 shows the results for the Gaussian and Bernoulli recognizers, where, as before, each point is the average of 30 repetitions (30 random splits). Due to computational costs, only 15 repetitions were carried out for $K = 64$.

From the results in Figure 5.7, it becomes clear that Bernoulli mixture HMMs achieve similar or better results than those of the conventional Gaussian mixture HMMs. Moreover, Bernoulli HMMs have a more stable behavior, as compared with Gaussian HMMs, in terms of flatter curves. However, as the value of K increases, the difference between the two recognizers decreases. The best results for them both are obtained with $K = 64$. In particular, the best result for the Bernoulli recognizer is an error rate of 30.9% (with $Q = 10$), while the Gaussian recognizer is slightly worse: 31.3% (with $Q = 8$). It must be noted that the optimal Bernoulli recognizer is much simpler than the optimal Gaussian recognizer: $1.4M$ and $4.4M$ parameters respectively.

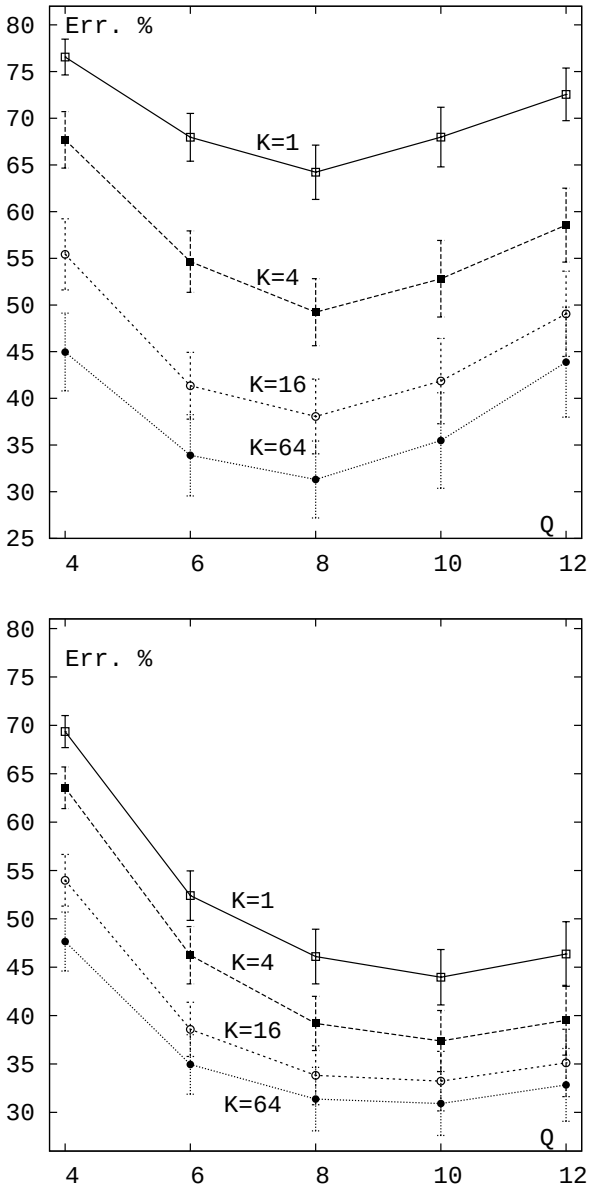


Figure 5.7: Classification error-rate (%) as a function of the number of states, for varying number of components (K). Top: Gaussian HMMs. Bottom: Bernoulli HMMs

5.8.3 Windowed BHMMs

In this subsection we test the impact of using a sliding window during feature extraction. Experiments were carried out on the well-known IFN/ENIT database of Arabic handwritten

Tunisian names (see Section 3.3). More precisely, for the experiments reported below, each image was first rescaled in height to $D = 30$ rows, while keeping the original aspect ratio, and then binarized using Otsu's binarization method. The resulting set of binary images was partitioned into five folds labeled as a, b, c, d and e, following the standard protocol.

We tried different values for the sliding window width W (1, 3, 5, 7, 9 and 11) and also different values for number of mixture components per state K (1, 2, 4, 8, 16, 32, 64). However, taking into account our previous, preliminary results in Khoury et al. [2010], we only tried BHMMs with 6 states as character models. BHMMs were initialized using the same approach than in the previous subsection. As usual, recognition of test images was performed by using the Viterbi algorithm.

Figure 5.8 shows the Word Error Rate (WER%) as a function of the number of mixture components, for varying sliding window widths. Each WER estimate (plotted point) was obtained by cross-validation with the first 4 standard folds (abcd). It is clear that the use of sliding windows improves the results to a large extent. Specifically, the best result, 7.4%, is obtained for $W = 9$ and $K = 32$, although very similar results are obtained for $W = 7$ and $W = 11$. It is worth noting that the best result achieved with no sliding windows ($W = 1$) is 17.7%, that is, 10 absolute points above of the best result achieved with sliding windows.

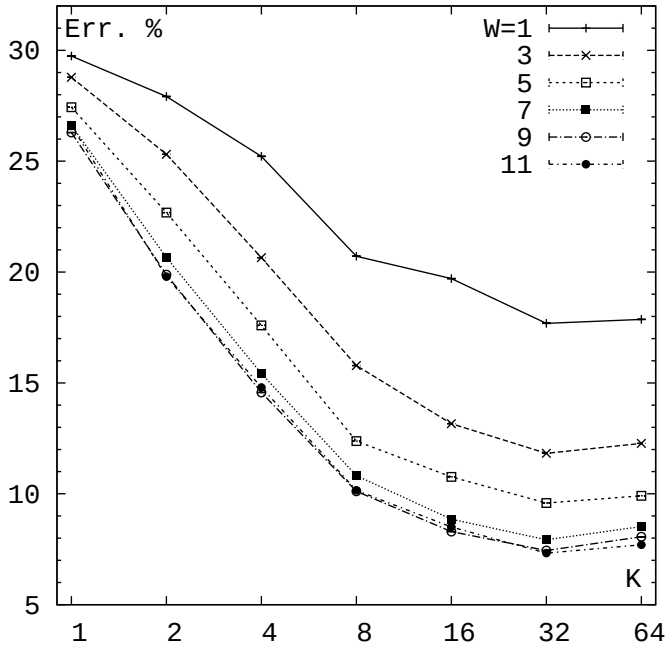


Figure 5.8: Classification error (%) on IFN/ENIT as a function of the number of mixture components (K) for varying sliding window widths (W)

To get some insight into the behavior of windowed BHMMs, the model for character ﺝ,

trained from folds abc with $W = 9$ and $K = 32$, is (partially) shown in Figure 5.9 (bottom) together with its Viterbi alignment with a real image of the character ﺟ, extracted from sample *de05_007* (top). As in Figure 5.2, Bernoulli prototypes are represented as gray images where the gray level of each pixel measures the probability of its corresponding pixel to be black (white $= 0$ and black $= 1$). From these prototypes, it can be seen that each state from right to left accounts for a different local part of ﺟ, as if the sliding window was moving smoothly from right to left. Also, note that the main stroke of the character ﺟ appears almost neatly drawn in most prototypes, whereas its upper dot appears blurred, probably due to a comparatively higher variability in window position.

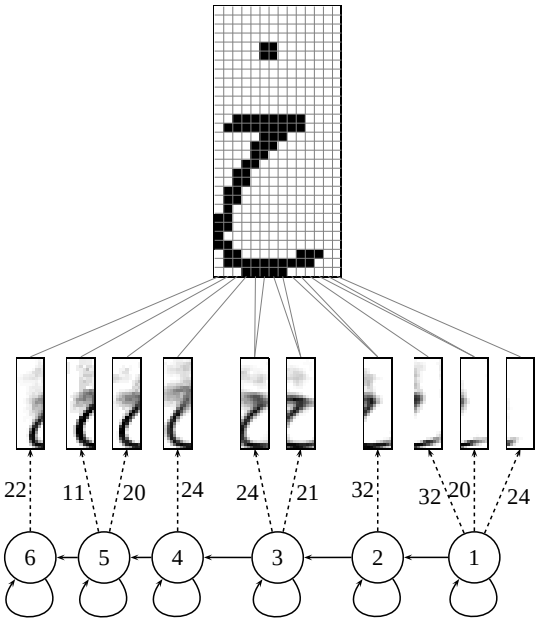


Figure 5.9: BHMM for character ﺟ, trained from folds abc with $W = 9$ and $K = 32$ (bottom), together with its Viterbi alignment with a real image of the character ﺟ, extracted from sample *de05_007* (top)

Following previous results in Khoury et al. [2010], in the experiments discussed above we only tried BHMMs with 6 states. However, in Dreuw et al. [2009] where conventional (Gaussian) HMMs are tested on IfN/ENIT, the authors claim that Arabic script might be better modeled with character HMMs of variable number of states since Arabic letters are highly variable in length (as opposed to Latin letters). In order to check this claim, experiments similar to those described above were repeated with character BHMMs of different number of states. To decide the number of states of each character, BHMM, we proceeded as in Section 5.8.1, but now segmenting the training samples was carried out using BHMMs with 4 states per character.

Figure 5.10 shows the WER as a function of the factor F, for different number of mixture components K and a window width of $W = 9$ (with which we obtained the best results in the previous experiments). The best result now, 7.3% (obtained with $F = 0.4$ and $K = 32$), is similar to the 7.4% obtained with 6 states per character. Therefore, in our case, the use of character models of different number of states does not lead to a significant improvement of the results.

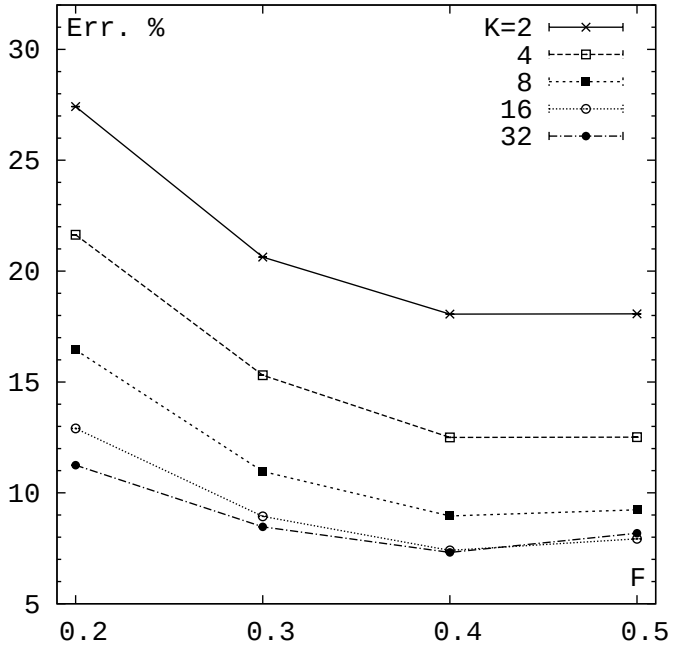


Figure 5.10: Classification error rate (%) on IFN/ENIT as a function of the factor F for varying values of the number of mixture components (K)

Although the results with variable number of states do not lead to significant improvements, it is interesting to see that there are cases in which, as expected, Arabic letters are better modeled with them. An example is shown in Figure 5.11 using the sample *dm33_037*. This sample was recognized using BHMMs with $W = 9$, $K = 32$ and both, 6 states (top) and variable number of states, with $F = 0.4$ (bottom). In both cases, the recognized word is Viterbi-aligned at character level (background color) and state level (bottom and upper ticks). Although it was incorrectly recognized as النفَاتية with BHMMs of 6 states (top), it was correctly recognized as الدخّانية with BHMMs of variable number of states (bottom). Note that there are two letters, 'ل' and 'د', that are written at the same vertical position (or column) and thus it is very difficult for our BHMMs to recognize them as two different letters. Anyhow, the incorrectly recognized word (top) is actually not very different in shape from the correct

one; e.g. the characters 'ز' and 'ر' are very similar.

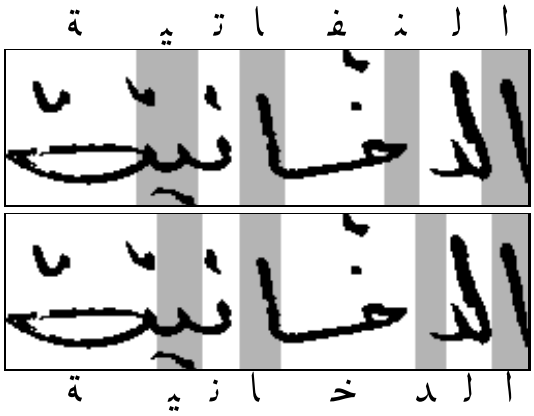


Figure 5.11: The sample *dm33_037* is incorrectly recognized as النفَاتية with BHMMs of 6 states (top), but correctly recognized as الدخّانية with BHMMs of variable number of states (bottom); the background color is used to represent Viterbi alignments at character level

5.8.4 Window Repositioning

Apart from the windowed BHMM, in Section 5.7 we also proposed three different window repositioning techniques to deal with vertical (horizontal) text distortions, which as we discussed, are difficult to model with conventional windowed BHMMs. To test these techniques on IFN/ENIT, we used the best settings found above, that is, $W = 9$, $K = 32$ and BHMMs of variable number of states with $F = 0.4$. We compared the standard technique (no repositioning) with the three repositioning techniques introduced in this work: vertical, horizontal and both directions (see Section 5.7). Results are given in Table 5.1 for each of the four partitions considered above (abc-d, abd-c, acd-b, and bcd-a) and the partition abcd-e, which is also often used by other authors.

From the figures in Table 5.1 it is clear that vertical window repositioning significantly improves the results obtained with the standard method or horizontal repositioning alone. The result obtained for the abcd-e partition with vertical (or both) repositioning, $\mathbf{6.1}\%$, represents a 50% relative error reduction with respect to the 12.3% of WER obtained without repositioning. As said in the introduction, our windowed BHMM system with vertical repositioning ranked first at the ICFHR 2010 Arabic Handwriting Recognition Competition. The best results on the test sets f and s (only known by the organization) from the last four competition editions [Märgner and El Abed, 2011] are provided in the Table 3.8 from Section 3.3.

Table 5.1: Classification error rate (%) of four repositioning techniques: none (no repositioning), vertical, horizontal and both. We used $W = 9$ and BHMMs of variable number of states ($F = 0.4$) and $K = 32$

		Error %			
Training	Test	None	Vertical	Horizontal	Both
abc	d	7.5	4.7	8.4	4.8
abd	c	6.9	3.6	7.7	3.8
acd	b	7.7	4.5	8.1	4.4
bcd	a	7.6	4.4	8.2	4.6
abcd	**e**	**12.3**	**6.1**	**12.4**	**6.1**
abcde	e	4.0	2.2	3.9	2.0

5.8.5 More Results on IAM

Due to the very good results obtained on the IFN/ENIT database using windowed BHMMs with repositioning, we have extended the experimentation over the IAM word dataset using windowed BHMMs. Furthermore, we have changed the protocol from Section 5.8.1 by a more challenging protocol. In particular, we have used IAM words dataset on the basis of the standard protocol for IAM lines, which is a writer independent protocol comprising $6\,161$ lines for training, 920 for validation and $2\,781$ for testing (see 3.2.2). Only words annotated as correctly segmented were used, which resulted in $46\,956$ words for training, $7\,358$ for validation and $19\,907$ words for testing. We used a closed vocabulary of $10\,208$ words for recognition, that is, the vocabulary of all words occurring in the training, validation and test sets. Class priors were computed as a smoothed unigram language model.

A first series of experiments was conducted on the training and validation data so as to determine appropriate preprocessing and feature extraction options. We tested different preprocessing alternatives, from no preprocessing at all to a full preprocessing method consisting of three conventional steps: gray level normalization, deslanting, and size normalization [Pastor i Gadea, 2007]. It is worth noting that, in this context, size normalization refers to a procedure for vertical size normalization of three different areas in the text line image (ascenders, text body and descenders), which of course might not be correctly located in all cases (see Section 2.2.2). On the other hand, feature extraction comprised three steps: rescaling of the preprocessed image to a given height D, binarization by Otsu's method, and final feature extraction by application of a window of a given width W and a particular repositioning technique. We tested different values of D (30 and 40) and W (9 and 11), and also each of the four repositioning techniques discussed above.

The best results in our first series of experiments were obtained with a two-step preprocessing including gray level normalization and deslanting, followed by feature extraction with $D = 40$, $W = 9$ and vertical repositioning. Using these settings, a second series of experiments was conducted on the training and validation data in which we tested different values for the number of states Q (4, 6, 8, 10 and 12) and the number of mixture components per state K (1, 4, 16 and 64). BHMMs were trained as described in Section 5.8.2. The results are shown in Figure 5.12. Note that our best result in it, 24.8%, was obtained with $K = 64$ and $Q = 8$.

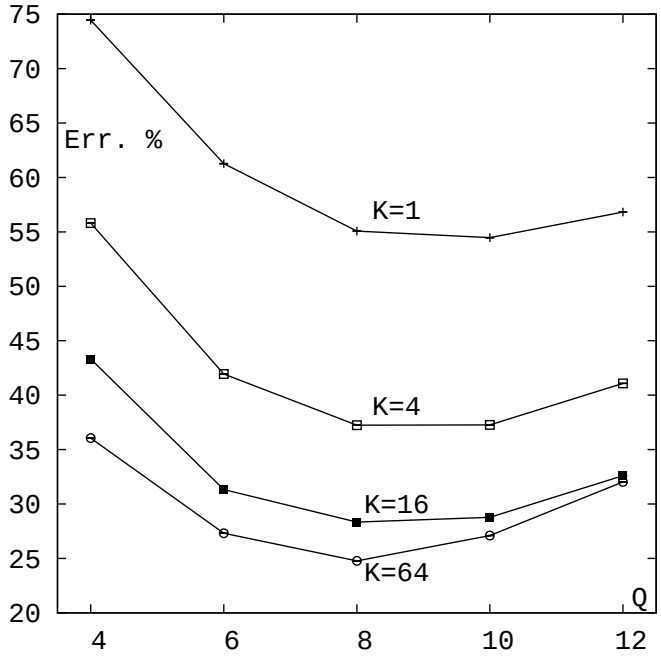


Figure 5.12: Classification error rate(%) on IAM words, using windowed BHMMs, as a function of the number of states (Q) for several number of mixture components (K)

As usual in recognition of handwritten text lines, we may fine-tune system performance by adequately weighting the importance of class priors with respect to class-conditional likelihoods. This is done by introducing a *grammar scale factor* G to scale class priors. We tested several values of G on the validation set using a system trained in accordance with the best results obtained in the previous series of experiments. A WER of 22.4% was achieved with $G = 90$.

In our final experiment on the IAM words dataset, we trained a system on the training and validation sets, using the best settings found above for preprocessing, feature extraction and recognition. It achieved a WER of 25.8% on the test set, which is quite good in comparison with other recent results on IAM words using the protocol described here [Bianne-Bernard et al., 2011]. In particular, as it can be seen in Table 5.2, BHMMs are much better than the two systems based on HMM technology alone, though the combination of these two systems with a third, hybrid system (combining HMMs and Neural Networks) achieves even better results.

Although the results are not directly comparable with the results reported in Section 5.8.2, since the experiment protocols used differ, our previous result with BHMMs, 29.6%, was not as good as the 25.8% of WER reported here.

Table 5.2: Test-set Classification error rate on IAM words obtained with BHMMs and other techniques reported in [Bianne-Bernard et al., 2011]

System	Error %
BHMM (this work)	25.8
Context-independent HMM (CI)	35.4
Context-dependent HMM (CD)	32.7
Combination (CI+CD+Hybrid)	21.9

5.8.6 Results on RIMES

For the experiments reported below, we have adopted the WR2 protocol used in the handwritten word recognition competition held at ICDAR 2009, which is described in Section 3.4. It comprises 44 196 samples for training, 7 542 for validation and 7 464 for testing. The lexicon to be used during recognition is that of the set to be recognized (1 636 words for validation and 1 612 for testing), and the alphabet consists of 81 characters. As above, class priors were computed as a smoothed unigram language model.

As with IAM words, a first series of experiments was conducted on the training and validation data to decide on adequate options and parameter values for preprocessing, feature extraction and recognition. In particular, we tried three preprocessing alternatives, two repositioning techniques and different number of states ($Q = 4$, 6, 8, 10) and mixture components ($K = 1$, 4, 16 and 64). Other parameter values used were $D = 30$ and $W = 9$. The best WER, 21.7%, was obtained with a two-step preprocessing including deslanting and size normalization, followed by feature extraction with $D = 30$, $W = 9$ and vertical repositioning; and then BHMM trained with $Q = 8$ and $K = 64$. Also as with IAM words, the performance of this system was fine-tuned by trying several values of the grammar scale factor G on the validation data. We achieved a WER of 18.7% with $G = 120$.

The best options and parameter values found on the validation set were used to train a system from the training and validation data, which was finally evaluated on the test set. We obtained a WER of 16.8%. In Table 3.10 of Section 3, this result is compared with those reported at the ICDAR 2009 competition (using the WR2 protocol) [Grosicki et al., 2009]. From these results, it becomes clear that our windowed BHMM system with vertical repositioning achieves comparatively good results.

5.9 Concluding Remarks

Embedded Bernoulli HMMs have been proposed for handwritten word recognition. They have been formally described first, and then empirically compared with conventional Gaussian HMMs on a task of handwritten word classification from the IAM database. We have first studied the impact use of simple Bernoulli HMMs at subword (character) level. We have obtained a classification error of 44.0%, which is 20 points better than the best result obtained with a conventional, Gaussian-based HMM system without mixtures. It is also worth noting that the proposed system works with less features and parameters than the conventional system (30 vs 60 and half of the parameters). On the other hand, the proposed system

has been also compared with Bernoulli HMM-based classifier at word level. As expected, the advantage of using subword models has been clearly confirmed.

We have also studied the use of Bernoulli mixture emission probability functions. In this case, the results of the Bernoulli based recognizer using binarized input images as features, are similar or better than those of the Gaussian mixture based recognizer using real features (gray levels and derivatives). Nevertheless, the feature extraction required for the Bernoulli recognizer is minimal, moreover, it is much simpler in terms of number of parameters.

Apart from our previous basic approach, in which narrow, one-column slices of binary pixels are fed into BHMMs, we have used a sliding window of adequate width to better capture image context at each horizontal position of the word image. Furthermore, windowed BHMMs have been improved by the introduction of window *repositioning* techniques. In particular, we have considered three techniques of window *repositioning* after window extraction: *vertical*, *horizontal*, and *both*. They only differ in the way in which extracted windows are shifted to align mass and window centers (only in the vertical direction, horizontally or in both directions). The experiments, reported over the well-known IFN/ENIT database of handwritten Tunisian town names, have carefully studied the effects of the window width, the number of states, and repositioning. As expected, the best results have been obtained with an adequate adjustment of the window width, number of states, number of mixture components and, what it seems even more important, (vertical) window repositioning after window extraction. An error of 6.1% has been achieved on the standard abcd-e partition. Furthermore, these repositioning techniques have been also tested on the IAM word dataset and the RIMES word dataset. In both cases reported results were competitive. It is worth noting, that the vertical repositioning technique proposed here has been used recently for other researchers in the field [Doetsch et al., 2012, Su et al., 2013].

Bibliography

A.-L. Bianne-Bernard, F. Menasri, R. Al-Hajj Mohamad, C. Mokbel, C. Kermorvant, and L. Likforman-Sulem. Dynamic and contextual information in hmm modeling for handwritten word recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(10): 2066 –2080, oct. 2011. ISSN 0162-8828. doi: 10.1109/TPAMI.2011.22.

Patrick Doetsch, Mahdi Hamdani, Adrià Giménez, Jesús Andrés-Ferrer, Alfons Juan, and Hermann Ney. Comparison of Bernoulli and Gaussian HMMs using a vertical repositioning technique for off-line handwriting recognition. In *2012 International Conference on Frontiers in Handwriting Recognition (ICFHR 2012)*, pages 3 – 7, 2012.

Philippe Dreuw, Georg Heigold, and Hermann Ney. Confidence-Based Discriminative Training for Model Adaptation in Offline Arabic Handwriting Recognition. In *Proc. of the Int. Conf. on Document Analysis and Recognition (ICDAR 200)*, pages 596–600, Barcelona (Spain), July 2009.

A. Giménez and A. Juan. Bernoulli HMMs at Subword Level for Handwritten Word Recognition. In *4th Iberian Conference on Pattern Recognition and Image Analysis*, volume 5524 of *LNCS*, pages 497–504. Springer-Verlag, Póvoa de Varzim (Portugal), June 2009a.

A. Giménez and A. Juan. Embedded Bernoulli Mixture HMMs for Handwritten Word Recognition. In *Proc. of the 10th Int. Conf. on Document Analysis and Recognition (ICDAR 2009)*, pages 896–900, Barcelona (Spain), July 2009b.

Adrià Giménez, Ihab Khoury, and Alfons Juan. Windowed Bernoulli Mixture HMMs for Arabic Handwritten Word Recognition. In *Proceedings of the 12th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 533–538, Kolkata (India), November 2010. IEEE Computer Society.

Adrià Giménez, Ihab Khoury, Jesús Andrés-Ferrer, and Alfons Juan. Handwriting word recognition using windowed Bernoulli HMMs. *Pattern Recognition Letters*, 35(0):149 – 156, 2014. ISSN 0167-8655. doi: http://dx.doi.org/10.1016/j.patrec.2012.09.002.

Joshua T. Goodman. A bit of progress in language modeling. Technical report, 2001.

E. Grosicki, M. Carré, J. M. Brodin, and E. Geoffrois. Results of the RIMES evaluation campaign for handwritten mail processing. In *Proc. of the International Conference on Document Analysis and Recognition (ICDAR 2009)*, pages 941–945, 2009.

Simon Günter and Horst Bunke. HMM-based handwritten word recognition: on the optimization of the number of states, training iterations and Gaussian components. *Pattern Recognition*, 37:2069–2079, 2004.

Frederick Jelinek. *Statistical Methods for Speech Recognition*. MIT Press, 1997.

A. Juan and E. Vidal. Bernoulli mixture models for binary images. In *Proc. of the 17th Int. Conf. on Pattern Recognition (ICPR 2004)*, volume 3, Cambridge (UK), August 2004.

Ihab Khoury, Adrià Giménez-Pastor, and Alfons Juan. Arabic Handwritten Word Recognition Using Bernoulli Mixture HMMs. In *The 3rd Palestinian International Conference on Computer and Information Technology (PICCIT)*, Hebron (Palestine), Feb 2010.

Volker Märgner and Haikal El Abed. ICDAR 2011 - Arabic Handwriting Recognition Competition. In *ICDAR '11*, pages 1444 – 1448, Beijing (China), sep 2011.

Moisés Pastor i Gadea. *Aportaciones al reconocimiento automático de texto manuscrito*. PhD thesis, Dep. de Sistemes Informàtics i Computació, València, Spain, Oct 2007. Advisors: E. Vidal and A.H. Tosselli.

Lawrence Rabiner and Biing-Hwang Juang. *Fundamentals of speech recognition*. Prentice-Hall, 1993.

Bing Su, Xiaoqing Ding, Liangrui Peng, and Changsong Liu. A Novel Baseline-independent Feature Set for Arabic Handwriting Recognition . In *Proc. 12th Int. Conf. on Document Analysis and Recognition (ICDAR 2013)*, pages 1282–1286, 2013.

S. Young et al. *The HTK Book*. Cambridge University Engineering Department, 1995.

CHAPTER 6

BERNOULLI HMMS FOR CONTINUOUS HTR

Contents

6.1 Introduction

In Chapters 4 and 5, an isolated handwritten word recognizer was proposed in which binary image pixels are directly fed into word-conditional *embedded Bernoulli mixture HMMs*, that is, embedded HMMs in which the emission probabilities are modeled with Bernoulli mixtures. As in Saon and Belaïd [1997], the basic idea is to ensure that no discriminative information is filtered out during feature extraction, which in some sense is integrated into the recognition model. In this chapter, this idea is extended to general, continuous handwritten text recognition; that is, whole sentences instead of isolated words. As in speech recognition, HMMs are used to model the probability of the input images given a transcription, and *n*-gram models are used to model the probability of a transcription [Bertolami et al., 2007, Pastor i Gadea, 2007, Rabiner and Juang, 1993]. Empirical results are reported on three database of handwritten text lines recognition: the well-known IAM database, the Germana database, and the Rodrigo database. We also report results from last two printed Arabic text recognition competitions [Slimane et al., 2011, 2013], in which we obtained state of the art results using continuous BHMMs.

The chapter is organized as follows. In Section 6.2 we explain how BHMMs are extended to continuous HTR. In Section 6.3, empirical results are reported. Concluding remarks are discussed in Section 6.4.

6.2 BHMM-based Continuous HTR

As described in (5.7), given an observation sequence $O = (\mathbf{o}_1, \ldots, \mathbf{o}_T)$, its most probable transcription is obtained as

$$w^* = \underset{w_1^N \in W}{\operatorname{argmax}}\ p(w_1^N)\, p(O \mid w_1^N)\,, \tag{6.1}$$

where W is the set of possible transcriptions. In contrast to handwritten word recognition, where W is a finite set of words, in continuous handwritten text recognition (HTR) W is a set with all possible word sequences which could be obtained from a finite set of words. Therefore, modeling directly $p(w_1^N)$ with prior probabilities, as we did in Section 5.5, is not feasible for continuous HTR, because, even though the number of possible word sequences is bounded, the number is usually too large to obtain good parameter estimations. Consequently, $p(w_1^N)$ is modeled by applying the Markov chain decomposition

$$p(w_1^N) = \prod_{i=1}^{N} p(w_i \mid w_1^{i-1})\,, \tag{6.2}$$

which is usually approached using *n*-gram models

$$p(w_1^N) = \prod_{i=1}^{N} p(w_i \mid w_{i-n+1}^{i-1})\,. \tag{6.3}$$

For more details about *n*-gram models see the Section 2.3.

As a consequence of the introduction of the n-gram models into (6.1) two main questions arise: how this affects to the parameter estimation of BHMMs, and how this affects to the recognition process. Regarding the first question, parameter estimation of BHMMs can be carried out as described in Section 5.6. Since, according to the maximum likelihood criterion the parameters of the BHMMs and *n*-gram models can be independently estimated.

The second question requires a more elaborated answer. In handwritten word recognition calculating $p(O \mid w_1^N)$ (or the approximated Viterbi probability) and then adding $p(w_1^N)$ is a possible approach. However, in continuous HTR this approach is unfeasible due to the too large number of word sequences.

An advantage of n-gram models, which we can use to efficiently carry out the recognition, is that they can be easily reinterpreted as probabilistic finite state models. Each probability $p(w \mid h)$ can be reinterpreted as the transition probability between the states h, where h is the current history, and $h' = \gamma(h \cdot w)$, where $\gamma(h \cdot w)$ is the new history resulting from appending w to h. Note, that here by history we mean the most recent $n-1$ observed words (left to right).

This interpretation of an n-gram is very important because we can replace the words in the n-gram model by its corresponding BHMMs, which in turn are built concatenating the BHMMs related to its symbols. The result of this process is a huge BHMM, whose most probable path can be obtained using the Viterbi algorithm. Depending on how BHMMs are embedded into the n-gram model we talk of two different strategies: embedding BHMMs into LM states and embedding BHMMs into LM edges.

6.2.1 Embedding BHMMs into LM States

This embedding strategy can only be applied to n-gram models with $n > 1$, and in general, to any finite state model in which states are words. A good example of this kind of finite state models are wordnets [Young et al., 1995]. In the case of n-gram models, each state is representing the history, that is, the last $n-1$ seen words. In order to apply this embedding strategy, we assume that the word related to each LM state is the last word of the history, which it is also the last observed word. This is the main reason because this strategy can not be applied to 1-grams, since in 1-grams the history is empty, and we can not associate any word to the empty history.

As its name indicates, in this embedding strategy the huge BHMM used in the search is obtained by replacing each state of the language model by the BHMM related to the word of the state. Each state of the resulting BHMM can be represented by a tuple (h, k, j), which denotes the jth state of the kth symbol of the word which has been embedded into the state h. An example of how BHMMs are embedded into language model states is shown in Figure 6.1. The example represents four transitions of a bigram model. For simplicity, we do not represent the BHMMs related to each character.

The search of the best path on the resulting BHMM can be efficiently carried out by means of the Viterbi algorithm. Specifically, we are trying to solve (6.1), which after applying the *maximum approximation* over $p(O \mid w_1^N)$, in a similar way than we did in Section 5.5.3, can

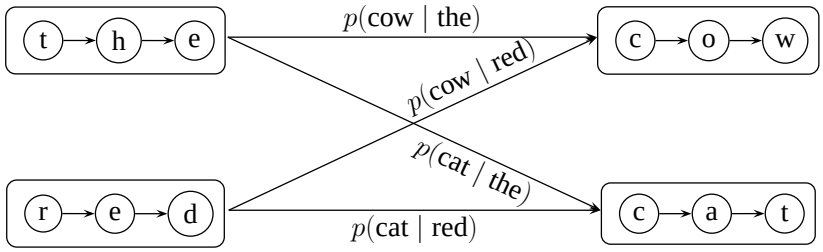


Figure 6.1: A visual example of how BHMMs are embedded into LM states.

be rewritten as

$$w^* \approx \underset{w_1^N \in W}{\operatorname{argmax}} \left(p(w_1^N) \cdot \max_{\substack{i_1, \ldots, i_{|w_1^N|+1} \\ q_1, \ldots, q_T}} p(\mathbf{o}_1^T, q_1^T, i_1^{|w_1^N|+1} \mid w_1^N) \right) \tag{6.4}$$

$$\approx \underset{w_1^N \in W}{\operatorname{argmax}} \left(\Big[\prod_{m=1}^{N} p(w_m \mid w_{m-n+1}^{m-1}) \Big] \Big[\max_{\substack{i_1, \ldots, i_{|w_1^N|+1} \\ q_1, \ldots, q_T}} \prod_{l=1}^{|w_1^N|} \hat{p}_{\theta_{s_l}}(\mathbf{o}_{i_l}^{i_{l+1}-1}, q_{i_l}^{i_{l+1}-1}) \Big] \right) \tag{6.5}$$

$$= \underset{w_1^N \in W}{\operatorname{argmax}} \max_{\substack{i_1, \ldots, i_{|w_1^N|+1} \\ q_1, \ldots, q_T}} \prod_{m=1}^{N} \Big[p(w_m \mid w_{m-n+1}^{m-1}) \prod_{l=|w_1^{m-1}|+1}^{|w_1^m|} \hat{p}_{\theta_{s_l}}(\mathbf{o}_{i_l}^{i_{l+1}-1}, q_{i_l}^{i_{l+1}-1}) \Big], \tag{6.6}$$

where here $|w_1^N|$ does not denote the length of w_1^N in words, N, but it denotes the length of w_1^N in symbols (characters). According to the Viterbi algorithm, the probability of w^* can be efficiently calculated by defining a dynamic programming table Q. Where each entry, $Q(t, h, k, j)$, is the probability of the most likely path up to time t (binary feature vector $\mathbf{o}_t$) that ends in the state (h, k, j) of the huge BHMM. Specifically, in the case of regular states Q will be defined as

$$Q(t, h, k, j) = \max_{\substack{\mathbf{w} \,:\, \mathcal{H}(\mathbf{w}) = h \\ i_1^L \,:\, i_L \leq t \\ q_1^t \,:\, q_t = j}} \underbrace{p(\mathbf{w})}_{\text{LM score}} \underbrace{\cdot p(\mathbf{o}_1^{i_{L-k+1}-1}, q_1^{i_{L-k+1}-1}, i_1^{L-k+1} \mid w_1^{N-1})}_{\text{Symbols in } w_1^N} \cdot \tag{6.7}$$

$$\underbrace{p(\mathbf{o}_{i_{L-k+1}}^{i_L-1}, q_{i_{L-k+1}}^{i_L-1}, i_{L-k+2}^{L} \mid i_{L-k+1}, w_N)}_{\text{Previous symbols in } w_N} \cdot \tag{6.8}$$

$$\underbrace{p(\mathbf{o}_{i_L}^{t}, q_{i_L}^{t} \mid i_L, w_N)}_{k\text{th symbol in } w_N}, \tag{6.9}$$

where N denotes the number of words, $\mathcal{H}$ returns the last $n-1$ words ($\mathcal{H}(\mathbf{w}) = w_{N-n+2}^N$), and L is the position in the whole symbol sequence of the kth symbol from w_N ($L = |w_1^{N-1}| + k$). Equation (6.7) can also be used for the special states I and F, by simply redefining the restriction related to the segmentation variables i. Specifically, for the state I

we will require $i_1^L : i_L = t+1$, while in the F case we will require $i_1^{L+1} : i_{L+1} = t+1$. Note that (6.7) is a recursive function. For example, the two first terms in (6.7) can be replaced by Q as follows

$$Q(t,h,k,j) = \max \; Q(i_{L-k+1}-1, h, 1, I) \cdot \tag{6.10}$$

$$p(\mathbf{o}_{i_{L-k+1}}^{i_L-1}, q_{i_{L-k+1}}^{i_L-1}, i_{L-k+2}^{L} \mid i_{L-k+1}, w_N) \cdot \tag{6.11}$$

$$p(\mathbf{o}_{i_L}^{t}, q_{i_L}^{t} \mid i_L, w_N) \; , \tag{6.12}$$

or for example, we can also replace the previous symbols section as follows

$$Q(t,h,k,j) = \max \; Q(i_L - 1, h, k, I) \cdot p(\mathbf{o}_{i_L}^{t}, q_{i_L}^{t} \mid i_L, w_N) \; . \tag{6.13}$$

Note as well that using Q we can calculate the probability of the most probable path, w^*, as

$$p(w^*) \approx \max_{h=v\cdot w} \; Q(T, h, |w|, F) \cdot p(\$ \mid h) \; , \tag{6.14}$$

where \$ is the special word used to denote end of sentence. The word sequence related to the most probable path, w^*, can be easily retrieved by building a parallel table of back-pointers, as we did in Section 5.5.3.

In any case, in order to efficiently compute Q we proceed as follows. For the special states I and F with $1 < t \leq T$ we have

$$Q(t,h,k,I) = \begin{cases} Q(t,h,k-1,F) & 1 < k \leq |w| \\ \max\limits_{\substack{h' \,:\, h \,=\, \gamma(h' \,\cdot\, w) \\ h' \,=\, v \,\cdot\, w'}} Q(t,h',|w'|,F) p(w \mid h') & k = 1 \end{cases} \; , \tag{6.15}$$

and, for any $h = v \cdot w$,

$$Q(t,h,k,F) = \max_{1 \leq j \leq M_{w_k}} Q(t,h,k,j) \cdot a_{w_k j F} \; , \tag{6.16}$$

where w_k is the kth symbol of the word w, and $|w|$ is its length in symbols. For the case of regular states, being $1 < t \leq T$ and $h = v \cdot w$, we have

$$Q(t,h,k,j) = \left[\max_{j' \in \{I,1,\ldots,M_{w_k}\}} Q(t-1,h,k,j') \cdot a_{w_k j' j} \right] b_{w_k j}(\mathbf{o}_t) \, . \tag{6.17}$$

Finally, the base case is defined for $t = 1$, $k = 1$ and any $h = w$

$$Q(1,h,1,j) = \begin{cases} p(w) \cdot a_{w_1 I j} \cdot b_{w_1 j}(\mathbf{o}_1) & 1 \leq j \leq M_{w_1} \\ 0 & \text{otherwise} \end{cases} \; . \tag{6.18}$$

6.2.2 Embedding BHMMs into LM Edges

This strategy can be applied to any n-gram model, or to any finite state model in which transitions between states are representing words. In fact, this is a more generic approach than the previous one, since a finite state model in which states are representing words can be

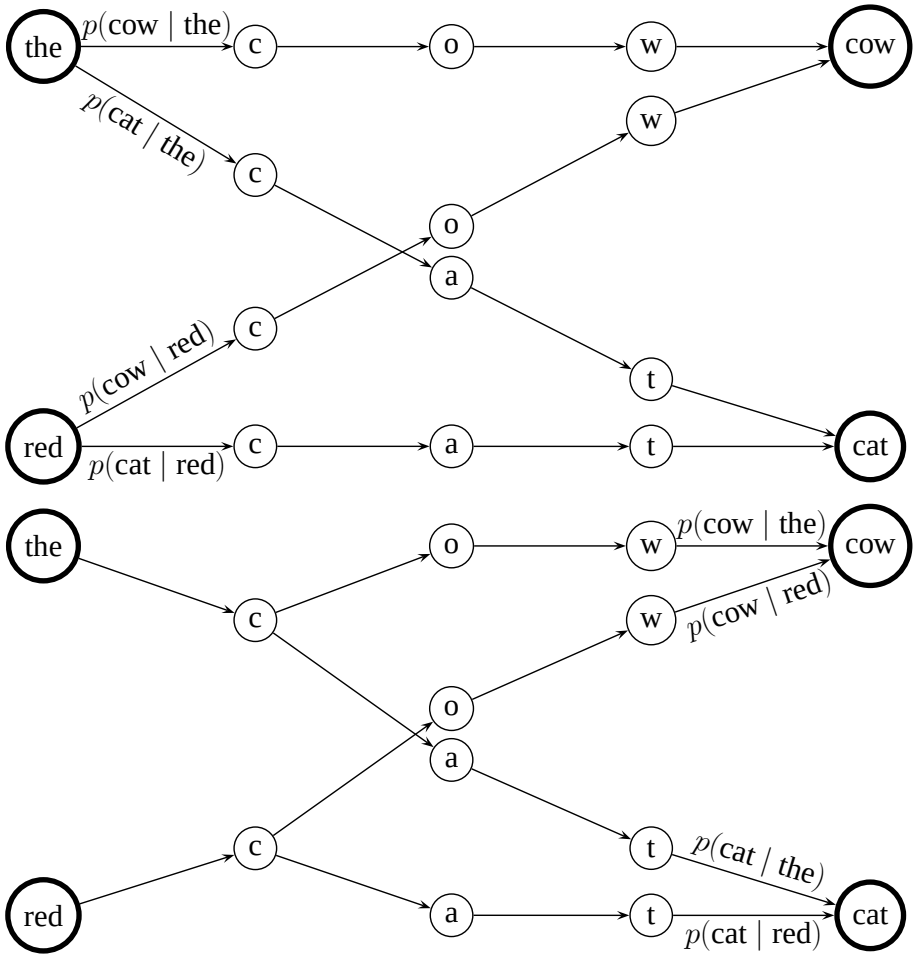


Figure 6.2: Top: A visual example of how BHMMs are embedded into edges. Bottom: The same example using a prefix tree. LM probabilities are applied at the positions where they appear.

easily interpreted as a finite state model in which edges are representing words by means of linking to the each edge the word related to its incoming state.

In this strategy, the huge BHMM used during recognition is obtained by replacing each edge by the BHMM related to its linked word. In this case, each state of the resulting BHMM is a tuple (h, w, k, j), which denotes the jth state of the kth symbol of word w which has been embedded into the edge related to $p(w \mid h)$. On top of Figure 6.2 a visual example of embedding BHMMs into LM edges is shown.

A disadvantage of this approach with respect to embedding BHMMs into states is that the search space is bigger. For example, assume an imaginary setting in which we have a

smoothed bigram model of 100 words (100 states with 100 outgoing edges), where all words have the same length (5 characters), and each character is modeled with a BHMM model of 3 states. In the case in which BHMMs are embedded into states, the number of states of the resulting BHMM is $(100 \cdot 1 \cdot 5 \cdot 3) = 1500$. However, when embedding BHMMs into edges we have that the number of states is $(100 \cdot 100 \cdot 5 \cdot 3) = 150000$. For this reason, a common approach is to represent the outgoing words of a LM state as a prefix tree. Although using this solution the number states is drastically reduced, the number of states is still to large with respect to the embedding into states approach. On bottom of Figure 6.2 we have the example on top but using a prefix representation of the outgoing edges from each state. It is worth noting, that due to the use of the prefix tree, LM probabilities must be calculated at the end of the edges.

In any case, when pruning techniques are used, which is quite common in practice, this disadvantage is irrelevant. Moreover, in some aspects this is a more practical approach. For example, using this approach is quite easy to generate wordgraphs [Ney and Ortmanns, 1999], since words are generated before reaching ingoing language model states.

As in the state embedding strategy we can apply the Viterbi algorithm. Specifically, in this case we define a dynamic programming table Q, where each entry $Q(t, h, w, k, j)$ is the probability of the most likely path up to time (binary feature vector) t that ends in the state (h, w, k, j) of the huge BHMM. In a similar way as we did in (6.7), we define $Q(t, h, w, k, j)$ in the case of regular state as

$$Q(t,h,w,k,j) = \max_{\substack{\mathbf{v}\cdot w \,:\, \mathcal{H}(\mathbf{v}) = h \\ i_1^L \,:\, i_L \le t \\ q_1^t \,:\, q_t = j}} \underbrace{p(\mathbf{v}) \cdot p(\mathbf{o}_1^{i_{L-k+1}-1}, q_1^{i_{L-k+1}-1}, i_1^{L-k+1} \mid \mathbf{v})}_{\text{Previous words}} \cdot \tag{6.19}$$

$$\underbrace{p(w \mid h)}_{\text{LM edge}} \cdot \underbrace{p(\mathbf{o}_{i_{L-k+1}}^{i_L-1}, q_{i_{L-k+1}}^{i_L-1}, i_{L-k+2}^{L} \mid i_{L-k+1}, w)}_{\text{Previous symbols in } w} \cdot \tag{6.20}$$

$$\underbrace{p(\mathbf{o}_{i_L}^{t}, q_{i_L}^{t} \mid i_L, w)}_{k\text{th symbol in } w} , \tag{6.21}$$

where v is a sequence the words, maybe empty, and L is now defined as $v| + k$. The main difference with (6.7) is that now h does not include the current word, since word recognition is carried out in the LM edge. It is worth noting, that in this case $p(w \mid h)$ can be calculated at the beginning of the edge, as we did here, or at any other position in the word, as for example at the end of the word.

Eficcient calculation of Q is performed as follows. For the special states I and F, with $1 < t \le T$, $Q(t, h, w, k, j)$ is calculated as follows

$$Q(t,h,w,k,I) = \begin{cases} Q(t,h,w,k-1,F) & 1 < k \le |w| \\ \max\limits_{(h',w'):\gamma(h'\cdot w')=h} Q(t,h',w',|w'|,F)p(w \mid h) & k = 1 \end{cases} , \tag{6.22}$$

and, for any pair (h, w),

$$Q(t,h,w,k,F) = \max_{1 \le j \le M_{w_k}} Q(t,h,w,k,j) \cdot a_{w_k j F} , \tag{6.23}$$

where w_k is the kth symbol of the word w, and $|w|$ is its length in symbols. For the case of regular states, being $1 < t \leq T$, we have

$$Q(t,h,w,k,j) = \left[\max_{j' \in \{I,1,\ldots,M_{w_k}\}} Q(t-1,h,w,k,j') \cdot a_{w_k j' j}\right] b_{w_k j}(\mathbf{o}_t)\,. \tag{6.24}$$

Finally, the base case is defined for $t = 1$, $h = \cdot$, $k = 1$ and any w as

$$Q(1,\cdot,w,1,j) = \begin{cases} p(w) \cdot a_{w_1 I j} \cdot b_{w_1 j}(\mathbf{o}_1) & 1 \leq j \leq M_{w_1} \\ 0 & \text{otherwise} \end{cases}\,, \tag{6.25}$$

where here $h = \cdot$ is referring to the empty history.

6.2.3 Pruning Techniques

Despite of the Viterbi algorithm, the recognition process in continuous HTR is still too demanding of time and memory. For example, using a bigram model for a vocabulary of $30K$ words, an average length of 5 characters per word, and 6 states per character; the number of different values which must be calculated according to (6.17) for a given instant t is $30K \times 5 \times 6 = 900K$, that is, about one million of entries at each instant. For this reason, pruning techniques are required to restrict the search space of the Viterbi algorithm. Among the pruning techniques, there are two that stand out due to their popularity.

Beam search: At each instant only those hypothesis whose probability is above a certain dynamic threshold, which depends on the probability of the best hypothesis at that instant, are expanded. This pruning is one of the more effective techniques, and must be carefully tuned. This technique requires the most probable path at current time to be known, hence, the pruning is carried out after expanding all previous pruned hypothesis. In order to accelerate the process, the threshold is usually dynamically calculated using the current best hypothesis without waiting to expand all hypothesis, that is, recalculating it every time that the best hypothesis changes.

Histogram pruning: At each instant only the N best hypothesis are expanded. Where N is a predefined number of active hypothesis. This is a very simple technique which allows a strict control over time and memory requirements.

Beam search and histogram pruning are usually used together. From both techniques, the beam search is the one with more impact over the speed recognition, since usually very few hypothesis are expanded at each time. However, the beam search does not allow the control of the maximum number of hypothesis at each instant, and in certain instants this number can be too large, and hence, the speed recognition slow. For this reason, when the beam search is used the histogram pruning is also applied, in order to provide at each instant an upper bound to the beam search.

6.2.4 Constrained Search

This section is the result of a collaborative work. The motivation was to take advantage of the BHMM recognizer, in order to develop a constrained recognizer for conventional Gaussian

mixtures. This constrained recognizer was used into an interactive handwritten recognition framework, in order to automatically improve the recognized text partially supervised by the user. That is, input image were re-recognized fixing that words that have been previously supervised by the user. We do not have any result on BHMMs and therefore we decided not to report any result related to constrained search in this thesis. Therefore, in this section we briefly describe how the dynamic programming table Q defined in the previous Section 6.2.2 can be modified in order to add some constraints. For more details about this collaboration please refer to Serrano et al. [2010b, 2013].

We will focus on constraints of type $c = (c_b, c_e, \bar{w})$, by which the word $\bar{w}$ mus be recognized from the segment $\mathbf{o}_{c_b}^{c_e}$. It is worth noting, that we are not forcing the word $\bar{w}$ to begin or end at specified segment. For simplicity, we will reformulate Q applying only one constraint.

Therefore, let Q_c a dynamic programming table, where each entry $Q_c(t, h, w, k, j)$ is the probability of the most likely path up to time t that ends in the state (h, w, k, j) of the huge HMM, constrained to the given constraint $c = (c_b, c_e, \bar{w})$. From the Q defined in Section 6.2.2 we can define Q_c as follows. Note, that here we are using $Q_c(t, h, w, k, j) = Q(t, h, w, k, j)$ to denote that Q_c is calculated as Q. First, we need to ensure that the word $\bar{w}$ is recognized in $\mathbf{o}_{c_b}^{c_e}$, so for the case of regular states we have

$$Q_c(t, h, w, k, j) = \begin{cases} 0 & c_b \leq t \leq c_e, w_k \neq \bar{w} \\ Q(t, h, w, k, j) & \text{other case} \end{cases} . \tag{6.26}$$

Second, previous equation avoids another word other than $\bar{w}$ to be recognized, but it does not prevent $\bar{w}$ is recognized more than once. Therefore, for the special state I when $k = 1$ we have

$$Q_c(t, h, w, 1, I) = \begin{cases} 0 & c_b \leq t < c_e \\ Q(t, h, w, 1, I) & \text{other case} \end{cases} . \tag{6.27}$$

Finally, for any other case

$$Q_c(t, h, w, k, j) = Q(t, h, w, k, j) . \tag{6.28}$$

6.3 Experiments

Experiments were carried out using three different databases of continuous handwritten text recognition: The IAM database, the Germana database and the Rodrigo database. The IAM database is a synthetic database of English handwritten text documents, which is one of the most popular databases used to evaluate HTR systems. While, the other two databases were obtained from two real historical documents written in Spanish. For more details about these databases see the sections 3.2.2, 3.5 and 3.6. Besides the previous experiments, we also report at the end of this section the results obtained using continuous BHMMs on the last two ICDAR competitions on recognition of printed Arabic text.

6.3.1 The IAM Database

In this section we carried out experiments using the IAM database. Results were reported using two preprocess alternatives: the preprocessed images that were used in España-Boquera

et al. [2011], and no preprocessing at all. As in previous chapter, feature extraction consisted from three steps: rescaling of the preprocessed image to a given height D, binarization by Otsu's method, and final feature extraction by application of a window of a given width W with vertical repositioning.

The language model was derived from three different English text corpora (LOB corpus, Brown corpus and Wellington corpus) in a similar way as described in Bertolami et al. [2007]. We used a bigram model, with an underlying vocabulary consisting on about $30\,000$ words. Note, that the vocabulary was not closed over the validation or test sets.

In a first series of experiments, the height D and the grammar scale factor were tuned on the validation set for both preprocessing alternatives. In particular, we tested several height values, $D \in \{15, 20, 30, 40\}$, and several grammar scale factor values, $\{10, 15, 20, 25, 30\}$. In all cases, window width was fixed $W = 9$. This value was selected in accordance to the results obtained in the previous chapter. The best results were obtained using height $D = 20$ for the preprocessing case, and $D = 40$ for the non preprocessed case. In both cases, the best grammar scale factor value was 25.

With these settings, a second series of experiments were carried out on the validation set in order to adjust an appropriate number of states per BHMM, $Q \in \{4,6,8, 10, 12\}$, and the number of mixture components per state, $K \in \{1, 4, 16, 64\}$. As usual (see Section 5.8), for $K = 1$, the recognizer was initialized by first segmenting the training set using a "neutral" model, and then using the resulting segments to perform a Viterbi initialization. For $K > 1$, it was initialized by splitting the mixture components of the trained model with $K/4$ mixture components per state. The results obtained are shown in Fig. 6.3.

Analyzing the results in Figure 6.3, it is seen that for both cases appropriate values for Q and K are 6 and 64 respectively. In particular we obtained a 29.1% of WER in the non preprocessed case, and a 25.5% of WER in the preprocessed case; that is, four points better than in the non preprocessed case. Using these values of Q and K new models were trained, for both preprocessing cases, using all training and development data, and then they were used to recognize the test set. In this set we obtained a 34.5% and 31.1% for the non preprocessed and the preprocessed cases respectively. These results are better than the 35.5% WER reported in Bertolami et al. [2007], which is obtained with a similar system based on Gaussian HMMs, geometrical features and flexible number of states.

6.3.2 The Germana Database

In this section, experiments were carried out using the Germana database, which is the transcription of a Spanish manuscript from 1891 (see 3.5 for more details). Specifically, we only used the first 179 pages, which comprise the unique part on the book in which Spanish is the only language. From page 180 till the end other 5 languages appear, sometimes as the unique language of the page, sometimes mixed with other languages. This pages were divided into training, development and test sets. In Table 6.1 some statistics of the partition used are shown.

Original scanned pages were segmented into lines, and then preprocessed, using the GiDoc prototype [Serrano et al., 2010a]. Preprocessing consisted of three steps: gray level normalization, deslanting, and size normalization of ascenders and descenders. Preprocessed lines were then transformed into sequences of binary feature vectors as usual: image rescal-

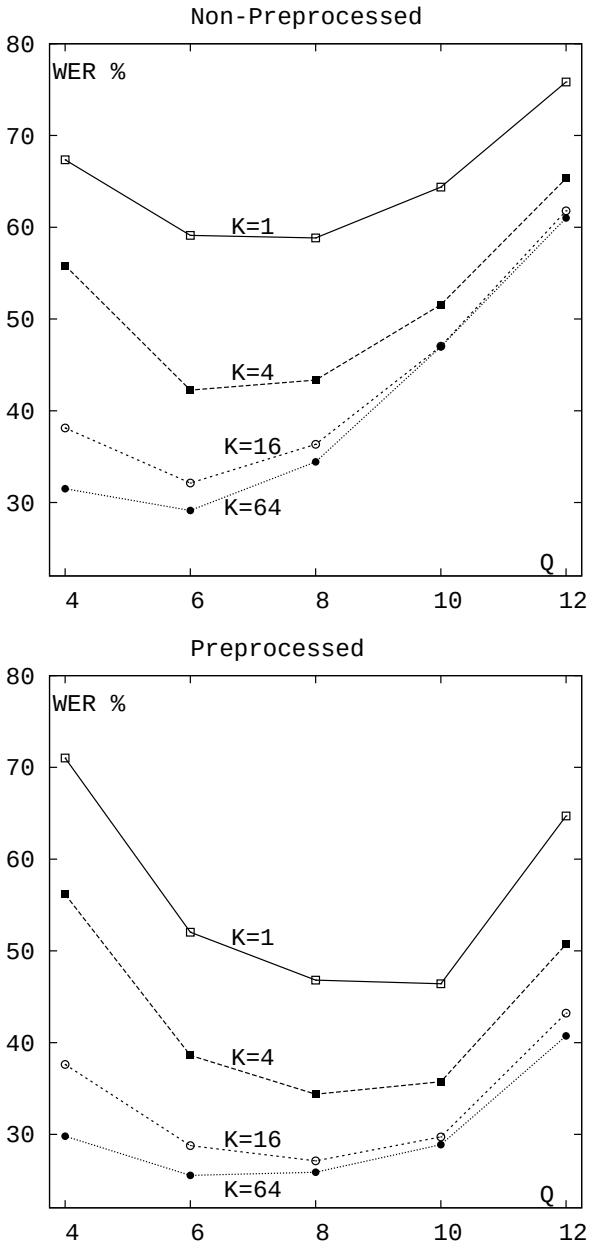


Figure 6.3: WER (%) on IAM lines dataset as a function of the number of states (Q), for varying number of components (K). Top: Non preprocessed. Bottom: Preprocessed

Table 6.1: Statistics of the partition used in the Germana experimentation

	Pages	Lines	Words	Lexicon	Singletons
Train	140	3004	30119	8496	6052
Dev.	20	367	3550	1585	1287
Test	17	381	3773	1738	1434
All	177	3752	37442	10007	7025

ing to height D, Otsu's binarization, and feature vector extraction by applying a vertical repositioned window of width W.

According to the statistics shown in Table 6.1, the percentage of singletons in the partition, about 19%, is too high. This is an important issue related to the estimation of the language model. On one hand, singletons appearing in test and development will be incorrectly recognized, since they will not appear in the language model. On the other hand, n-grams related to the training singletons will be poorly estimated, and they will be useless for test recognition since they do not appear in the test or development sets. Furthermore, the amount of running words in training is small. In order to properly estimate a language model we adopted the method proposed in del Agua et al. [2012]. In this paper, which also uses the Germana database for experimentation, a character based language model is used instead of a conventional one based on words. That is, words are split on characters and treated as words for language model estimation and recognition. After recognition, characters are joined again using the blank character as word separator.

Several experiments were carried out over the development set in order tune the meta parameters of the model. In particular, we tested several height values, $D \in \{15, 20, 30, 40\}$, several number of states, $Q \in \{4, 6, 8, 10, 12\}$, several mixture components per state, $K \in \{1, 4, 16, 64\}$, and several window widths, $W \in \{7, 9, 11, 13\}$. Regarding the recognition parameters, we also tested several grammar scale factor values, $\{10, 15, 20, 25, 30\}$, and order values for the character based n-gram model, $n \in \{9, 10, 11, 12\}$. The best result, $\mathbf{16.5}\%$, was obtained using features of height $D = 30$ and window width $W = 13$, HMMs with $Q = 6$ and $K = 64$, and carrying out the recognition with a grammar scale factor 40 and an 11-gram language model. In Figure 6.4, the WER as a function of the number of states and mixture components per state is shown for the best case.

Using the best configuration from development we carried out a final experiment over the test set. In this case the development set was included into the training set. We obtained a $\mathbf{13.8}\%$, which is close to the 12.1% obtained in del Agua et al. [2012] using discriminative Gaussian HMMs.

6.3.3 The Rodrigo Database

We finish the experimentation section carrying out experiments over the Rodrigo database. As in the Germana database, the Rodrigo database is the transcription of a Spanish manuscript, in this case from 1545 (further details in 3.6). For the experimentation, the database were divided into training, development and test sets. Some statistics are shown in Table 6.2.

As we did in Germana database, original pages were segmented into lines using the GIDOC. However, time lines were extracted without applying any kind of preprocess. Over

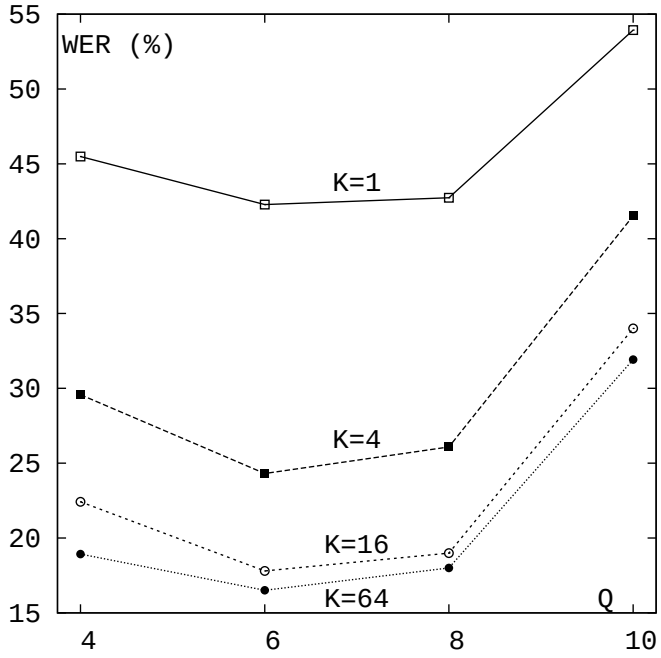


Figure 6.4: WER (%) on Germana dataset as a function of the number of states (Q), for varying number of components (K)

Table 6.2: Statistics of the partition used in the Rodrigo experimentation

	Pages	Lines	Words	Lexicon	Singletons
Train	410	10001	109143	13480	8112
Dev.	205	5010	55195	7453	4587
Test	227	5346	59105	7523	4513
All	842	20357	223443	20288	11738

the segmented lines the same feature extraction process used en previous experiments was applied: image scaling to a given height, Otsu's binarization, and vertical window repositioning. For language modeling, we used conventional word based n-gram models.

As usual, we began the experimentation carrying out several experiments over the development set. Regarding the training, we tuned the following parameters: height $D \in \{15, 20, 30, 40\}$, number of states $Q \in \{2, 4, 6, 8, 10\}$, number of mixture components $K \in \{1, 4, 16, 64\}$, and window width $W \in \{7, 9, 11\}$. While for recognition we tested these parameters: grammar scale factor values $\{10, 15, 20, 30, 40\}$, and n-gram order $n \in \{2, 3, 4\}$. The best result, 26.6%, was obtained using height $D = 30$, $Q = 4$ states, $K = 64$ mixture components per state, and a window width of $W = 9$. Recognition was carried out using a grammar scale factor of 20 with a 4-gram model. In Fig. 6.5, the WER as a function of the

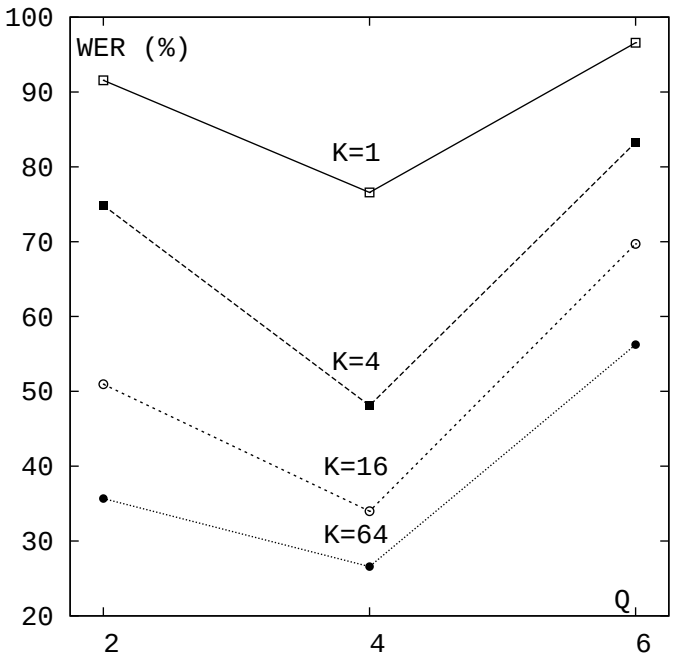


Figure 6.5: WER (%) on Rodrigo dataset as a function of the number of states, for varying number of components (K)

number of states and mixture components per state is shown for the best case.

Using the best configuration from development, we carried out a final experiment on the test set. In this experiment the development set was included into the training set. We obtained a 20.1%, and a character error rate of 6.0%. We do not have comparative results, but taking into account the low character error rate, and the problematic of the corpus (archaic language, embellishing writing, etc) we think that this is a good result.

6.3.4 Results on Printed Arabic Text

Given the good results obtained for handwritten Arabic text on IFN/NEIT in Chapter 5, we decided to participate in the *1st Arabic Recognition Competition: Multi-fon Multi-size Digitally Represented Text* on ICDAR 2011 [Slimane et al., 2011].

The database used for this competition was The Arabic Printed Text Image (APTI) database. The APTI database is a collection of images of Arabic Printed words. It was recently published by Slimane et al. [2009] for large-scale benchmarking of open-vocabulary, multi-font, multi-size and multi-style text recognition systems in Arabic. It consists of 113284 different single words, each one available in 10 different fonts, 10 different font sizes, and also 4 different styles.

APTI is divided into six balanced sets (set1 , set2 , ..., set6) to allow for flexibility in

Table 6.3: Results (character error rate) of the **UPV-PRHLT-REC1** system on the second protocol of ICDAR 2011 APTI competition.

Font / Size	**6**	**8**	**10**	**12**	**18**	**24**	**Average**
Andalus	1.1	5.2	3.9	3.3	3.3	3	3.3
Arabic Transparent	1.0	4.8	4.5	3.9	3.8	3.9	3.6
Simplified Arabic	0.8	3.8	3.3	3.1	3.0	3.6	3.8
Traditional Arabic	10.7	18.1	15.7	16.4	16.5	15	15.4
Diwani Letter	9.1	24.2	22.2	21.9	5.1	20.4	17.2

the design of experimental protocols. Each set has different words, but characters are equally distributed. The five first sets are available for the scientific community. The sixth set is kept by the authors for future evaluation of systems in blind mode. The are two big differences between the APTI task and the IFN/ENIT database. The first difference is that the APTI task is a printed text recognition task. The second big difference is that the vocabulary of set6 is unknown and it is supposed to be much larger.

For the competition 2 protocols were proposed. In the first protocol font and style were fixed to *Arabic Transparent* and *Plain* respectively, while several font sizes were tested (6, 8, 10, 12, 18 and 24). The second protocol is equal to the first protocol but adding more fonts: *Diwani letter*, *Andalus*, *Arabic Transparent*, *Simplified Arabic* and *Traditional Arabic*. In both cases set6 was used for evaluation, and the font size for each test sample was known.

The systems we send were trained from input images scaled in height to 40 pixels (while keeping the aspect ratio) after adding a certain number of white pixel rows to both top and bottom sides of each image, and then binarized with the Otsu's algorithm. A sliding window of width $W = 9$ was applied without repositioning. The number of states per character was adjusted to $Q = 5$ states for images with font size of 6, and $Q = 6$ states for other font sizes. Similarly, the number of mixture components per state was empirically adjusted to $K = 64$. On the other hand, in order to deal with out of vocabulary words in test, recognition was carried out using a language model (5-gram) at character level.

Two systems were submitted: **UPV-PRHLT-REC1** and **UPV-PRHLT-REC2**. The first system obtained better results and ranked first in the competition. This system was used for both tasks/protocols. In the first task (one font) one model for each font size was trained and used later to recognize the test corpus. For the second task, for each font size, a different model for each font was trained. The test corpus was recognized on all models, and the recognized text word with the highest probability was selected. In Table 6.3, the results of the first system on the second protocol are shown.

A second competition on the APTI database was released in ICDAR 2013: *Competition on Multi-font and Multi-size Digitally Represented Arabic Text* [Slimane et al., 2013]. Despite in this new competition 4 protocols were proposed, we only participated in the first three protocols. The first protocol (PC0) is the same that the first protocol of the first competition, that is, font and style were fixed to *Arabic Transparent* and *Plain* respectively, while several font sizes were tested (6, 8, 10, 12, 18 and 24). The second protocol (PC1) is equal to the first protocol but now the font size for test samples is unknown. Finally, in the third protocol (PC2) the *Arabic Transparent* font from the second protocol is replaced by the ligatured font

Table 6.4: A summary of the results (character error rate) obtained by the UPV-PRHLT systems in the ICDAR 2013 APTI competition.

	6	**8**	**10**	**12**	**18**	**24**	**Average**
PC0	0.52	0.07	0.04	0.04	0.04	0.04	0.12
PC1	0.59	0.06	0.03	0.01	0.02	0.02	0.12
PC2	2.84	1.53	1.29	1.23	1.24	1.19	1.55

DecoType Naskh.

The systems we send were very similar to the systems send to the previous competitions. The main difference is that this time we have applied vertical repositioning. Specifically, basic training consisted on scaling input images in height to 40 pixels (while keeping the aspect ratio), and then binarized with the Otsu's algorithm. A sliding window of width $W = 9$ using the vertical repositioning was applied. The number of states per character was adjusted to $Q = 7$ states for all font sizes. Similarly, the number of mixture components per state was empirically adjusted to $K = 128$.

Three variants of the UPV-PRHLT system were submitted: **UPV-PRHLT-REC1** (for protocol PC0), **UPV-PRHLT-RECPC2** (protocol PC1) and **UPV-PRHLT-RECPC3** (protocol PC2). For protocol PC0, where the size selection option is enabled, six different models were trained on the *Arabic Transparent* font images, one model for each font size. For all test images of a specific font size, a specific model was selected to recognize test images. For protocol PC1, only one model for all font sizes was trained on the *Arabic Transparent* font images. For protocol PC2, one model for all font sizes was trained on the *DecoType Nash* font images. A summary of the results obtained in the competition are shown in Table 6.4. These results, which are quite better than those obtained without repositioning, ranked second in the competition, and they are very close to the best system.

6.4 Concluding Remarks

Embedded Bernoulli mixture HMMs have been proposed for Continuous Handwritten Text Recognition and comparatively good results have been obtained on the well-known IAM database. In particular, a 31.1% of test-set WER has been achieved. Experiments on real ancient documents have been also carried out. Although there are not comparable results, the obtained results are quite good for a an automatic transcription from an ancient document. Finally, state of the art results have been obtained on printer Arabic text recognition using BHMMs and a language model at character level, which solves the problem of out of vocabulary words. In fact, reported results in printed Arabic were ranked first and second respectively on ICDAR 2011 and ICDAR 2013 competitions.

Bibliography

R. Bertolami, S. Uchida, M. Zimmermann, and H. Bunke. Non-uniform slant correction for handwritten text line recognition. In *Proc. 9th Int. Conf. on Document Analysis and Recognition (ICDAR 2007)*, pages 18–22, 2007.

Miguel A. del Agua, Nicolás Serrano, Jorge Civera, and Alfons Juan. Character-Based Handwritten Text Recognition of Multilingual Documents. In Doroteo Torre Toledano, Alfonso Ortega Giménez, António Teixeira, Joaquín González Rodríguez, Luis Hernández Gómez, Rubén San Segundo Hernández, and Daniel Ramos Castro, editors, *Advances in Speech and Language Technologies for Iberian Languages*, Communications in Computer and Information Science, pages 187–196. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-35291-1. doi: 10.1007/978-3-642-35292-8_20. URL `http://dx.doi.org/10.1007/978-3-642-35292-8_20`.

S. España-Boquera, M.J. Castro-Bleda, J. Gorbe-Moya, and F. Zamora-Martinez. Improving offline handwritten text recognition with hybrid hmm/ann models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(4):767–779, april 2011. ISSN 0162-8828. doi: 10.1109/TPAMI.2010.141.

Adrià Giménez-Pastor and Alfons Juan. Embedded Bernoulli Mixture HMMs for Continuous Handwritten Text Recognition. In *13th International Conference on Computer Analysis of Images and Patterns (CAIP)*, pages 197–204. Springer-Verlag, 2009.

Ihab Khoury, Adrià Giménez, Alfons Juan, and Jesús Andrés-Ferrer. Arabic Printed Word Recognition Using Windowed Bernoulli HMMs. In *17th Internarional Conference on Image, Analysis and Processings (ICIAP 2013)*, pages 330–339, Naples (Italy), Sep 2013.

H. Ney and S. Ortmanns. Dynamic programming search for continuous speech recognition. *IEEE Signal Processing Magazine*, 16(5):64–83, September 1999.

Moisés Pastor i Gadea. *Aportaciones al reconocimiento automático de texto manuscrito*. PhD thesis, Dep. de Sistemes Informàtics i Computació, València, Spain, Oct 2007. Advisors: E. Vidal and A.H. Tosselli.

Lawrence Rabiner and Biing-Hwang Juang. *Fundamentals of speech recognition*. Prentice-Hall, 1993.

G. Saon and A. Belaïd. High Performance Unconstrained Word Recognition System Combining HMMs and Markov Random Fields. *Int. J. of Pattern Recognition and Artificial Intelligence*, 11(5): 771–788, 1997.

N. Serrano, L. Tarazón, D. Pérez, O. Ramos, and A. Juan. The GIDOC prototype. In *Proceedings of the 10th International Workshop on Pattern Recognition in Informatic Systems (PRIS 2010)*, June 2010a.

Nicolás Serrano, Adrià Giménez, Alberto Sanchis, and Alfons Juan. Active Learning Strategies in Handwritten Text Recognition. In *Proceedings of the ICMI-MLMI 2010*, pages 1–9. ACM, 2010b.

Nicolás Serrano, Adrià Giménez, Jorge Civera, Alberto Sanchis, and Alfons Juan. Interactive handwriting recognition with limited user effort. *International Journal on Document Analysis and Recognition (IJDAR)*, pages 1–13, 2013. ISSN 1433-2833. doi: 10.1007/s10032-013-0204-5. URL `http://dx.doi.org/10.1007/s10032-013-0204-5`.

F. Slimane, R. Ingold, S. Kanoun, A.M. Alimi, and J. Hennebert. A New Arabic Printed Text Image Database and Evaluation Protocols. In *Document Analysis and Recognition, 2009. ICDAR '09. 10th International Conference on*, pages 946–950, 2009. doi: 10.1109/ICDAR.2009.155.

Fouad Slimane, Slim Kanoun, Haikal El Abed, Adel M. Alimi, Rolf Ingold, and Jean Hennebert. ICDAR 2011 - Arabic Recognition Competition: Multi-font Multi-size Digitally Represented Text. In *ICDAR '11*, pages 1149 – 1453, Beijing (China), 2011.

Fouad Slimane, Slim Kanoun, Haikal El Abed, Adel M. Alimi, Rolf Ingold, and Jean Hennebert. ICDAR2013 Competition on Multi-font and Multi-size Digitally Represented Arabic Text. In *ICDAR '13*, pages 1465 – 1469, Washington, DC (USA), 2013.

S. Young et al. *The HTK Book*. Cambridge University Engineering Department, 1995.

CHAPTER 7

MIXTURE MULTI-CLASS LOGISTIC REGRESSION MODELS FOR BINARY IMAGES

Contents

7.1 Introduction

As we have analyzed in previous chapters, Bernoulli-based models are competitive for handwritten text recognition (HTR). HTR usually involve text images which are only composed by black and white colors. Since Bernoulli-based models are very well suited for such cases, binaryzed input images are directly fed into a Bernoulli-based model without the need of a sophisticated feature extraction process. Bernoulli mixtures have been successfully applied to isolated character recognition [Juan and Vidal, 2004, Romero et al., 2007]. In this thesis, we have obtained very good results in continuous HTR and isolated handwritten word recognition using HMMs with Bernoulli mixture emission probabilities at the states (BHMMs).

All previously mentioned Bernoulli-based models are known to be generative models. Generative models are classifiers based on the optimal Bayes classifier [Duda and Hart, 1973]. The optimal Bayes classifier is a very well-spread classification technique. The Bayes classifier, which minimizes the classification error rate, is the one that selects the class, c, that maximizes the posterior probability $p_r(c \mid \mathbf{x})$ for a given input $\mathbf{x}$. The posterior probability is usually unknown and needs to be approximated. In the case of generative models, the posterior probability is approximated by a joint probability model $p_\theta(c, \mathbf{x})$ which is parametrized by $\boldsymbol{\theta}$.

Generative models have two great advantages among many others. On the one hand, the parameters of the generative models are easily understandable for researchers. For instance, in the Bernoulli-based models, the model parameters can be displayed as grey level images, so that we can know which pixels are more probable than others within a class. On the other hand, generative models are mostly trained with maximum likelihood estimation (MLE) criterion. One of the advantages of this criterion is that there are well-known algorithms for training generative models with hidden variables such as the EM [Dempster et al., 1977].

Despite the good properties of the MLE criterion, it has a very important drawback when used in classification problems. The MLE is aimed at explaining the probability distribution underlying the training sample, that is, maximizing the likelihood of the joint probability function $p_\theta(c, \mathbf{x})$. However, we are interested in simply classifying samples, and there is no guarantee that the MLE parameters are the most suitable for classifying.

The discriminative models and criteria are aimed at classifying the data without explaining it, and, hence, they directly approximate the posterior class probability by a model $p_\lambda(c|\mathbf{x})$ which is parametrized by $\boldsymbol{\lambda}$. However, discriminative parameters are difficult to understand provided that they do not explain the input. Discriminative parameters are usually estimated by the *maximum mutual information (*MMI*)* criterion, which directly maximizes the likelihood of the posterior probability function $p_\lambda(c \mid \mathbf{x})$. In contrast to MLE, the parameters estimated with MMI maximize the differences between classes in order to better classify samples. Unfortunately, there is no closed form solution for the MMI criterion, and few unsatisfactory algorithms are available for finding the optimal parameters. This problem is specially important for discriminative models with hidden variables

The generalized iterative scaling (GIS) algorithm [Darroch and Ratcliff, 1972] finds the optimal discriminative parameters accordingly to the MMI criterion for a special family of discriminative models, the so-called *log-linear or maximum entropy models (LLM)*. However, GIS is not suited for LLM with hidden variables. Recently, in Heigold et al. [2008] a similar algorithm, namely GGIS, has been proposed for training LLM with hidden variables. Due

to its very interesting properties, we will pay special attention to the case of mixture of log-linear models Heigold et al. [2008], which approximate the posterior probability with a set of parameters $\boldsymbol{\lambda}$ as follows

$$p_\lambda(c \mid \mathbf{x}) = \frac{1}{\mathcal{Z}(\mathbf{x})} \sum_k \exp(\boldsymbol{\lambda}^T \mathbf{f}(\mathbf{x}, c, k)) \,, \tag{7.1}$$

where $\mathbf{f}$ is a given vector feature functions, and k is a hidden variable. It is worth noting, that for the special case in which the number of mixture components is 1, (7.1) is a conventional log-linear model which can be trained using the GIS algorithm. Although the GIS and GGIS find the optimal parameters accordingly to the MMI criterion, a huge amount of iterations are required [Heigold et al., 2008], and the RPROP algorithm [Riedmiller and Braun, 1993] is usually employed instead, since it obtains similar results while providing faster convergence. The RPROP algorithm is a hill-climbing algorithm, which has convergence ratios independent of the gradient since only its sign is used by the algorithm. In this way, the training is accelerated provided that the GIS algorithm, and in particular the GGIS algorithm, have slow convergence ratios because of the gradient slope. However, the RPROP algorithm introduces new meta-parameters during the training process, and the convergence is not guaranteed.

In this chapter we will study the following issues:

1. We propose a particular case of mixture log-linear models for binary data inspired by Bernoulli mixtures, a mixture of multi-class logistic regression (MMLR).
2. We prove the equivalence between Bernoulli mixtures classifiers and MMLRs for binary data. Consequently, the equivalence of first order log-linear models for binary inputs and Bernoulli classifiers is also proved.
3. We provide a MMI training scheme for Bernoulli (mixtures) classifiers by means of their equivalence with MMLRs.
4. We provide the capability to understand discriminative parameters of the MMLR from a generative perspective by means of their equivalence with Bernoulli mixture classifiers.

The chapter is organized as follows. In Section 7.2, we start reviewing the Bernoulli mixture classifiers and its maximum likelihood estimation. In Section 7.3, the MMLR classifier similar to the Bernoulli mixture classifier is proposed. The Section 7.4 proves that both classifiers are equivalent, by providing a proof scheme that may be used for different generative models. The experiments carried out in a task of hand written Indian digits recognition are gathered in Section 7.5. Final thoughts and future work are discussed in the last Section.

7.2 Bernoulli Mixture Classifier

Given a binary input vector $\mathbf{x} \in \{0,1\}^D$ and a class c from the set of classes $\{1, \ldots, C\}$; a Bernoulli mixture classifier is defined as follows

$$c^* = \underset{c}{\operatorname{argmax}} \; p_\theta(c \mid \mathbf{x}) \,, \tag{7.2}$$

where the class posterior $p_\theta(c \mid \mathbf{x})$ is approximated as follows

$$p_\theta(c \mid \mathbf{x}) = \frac{p_\theta(c, \mathbf{x})}{\mathcal{Z}(\mathbf{x})} = \frac{p_\theta(\mathbf{x} \mid c) p_\theta(c)}{\mathcal{Z}(\mathbf{x})}\,, \tag{7.3}$$

where $\mathcal{Z}(\mathbf{x})$, which equals the input probability, $p_\theta(\mathbf{x})$; ensures the class posterior probability to sum up to 1,

$$\mathcal{Z}(\mathbf{x}) = \sum_c p_\theta(c)\, p_\theta(\mathbf{x} \mid c)\,. \tag{7.4}$$

The class prior probability, $p_\theta(c)$, is modeled as a table π_c; and $p_\theta(\mathbf{x} \mid c)$ is assumed to follow a Bernoulli mixture probability distribution as we previously defined in Section 5.3. This is the usual way in which Bernoulli mixture classifiers are defined. However, a more compact and equivalent definition, which is more convenient for our current purpose, can be obtained by directly modelling the joint probability function as follows

$$p_\theta(\mathbf{x}, c) = \sum_{k=1}^{K} p_\theta(\mathbf{x}, c, k)\,. \tag{7.5}$$

Each joint probability is decomposed in two terms

$$p_\theta(\mathbf{x}, c, k) = \pi_{ck}\, p_\theta(\mathbf{x} \mid c, k)\,, \tag{7.6}$$

where π_{ck} embrace the prior coefficient of the k-th component of class c and the prior probability of class c (π_c); and where $p_\theta(\mathbf{x} \mid c, k)$ follows a multivariate Bernoulli probability distribution. Summarizing, the Bernoulli mixture classifier approximates the posterior probability as follows

$$p_\theta(c \mid \mathbf{x}) = \frac{1}{\mathcal{Z}(\mathbf{x})} \sum_{k=1}^{K} \left(\pi_{ck} \prod_{d=1}^{D} p_{ckd}^{x_d} \cdot (1 - p_{ckd})^{1-x_d}\right), \tag{7.7}$$

with the parameters $\theta = \{\pi; \mathbf{p}\}$, where all parameters are required to be probabilities, and in particular the mixture coefficients are constrained to sum 1, *i.e.*

$$\sum_{c,k} \pi_{ck} = 1\,. \tag{7.8}$$

Finally, the Bernoulli mixture model classifies accordingly to the following rule

$$c^* = \operatorname*{argmax}_c \frac{p_\theta(c, \mathbf{x})}{\mathcal{Z}(\mathbf{x})} = \operatorname*{argmax}_c p_\theta(c, \mathbf{x})\,, \tag{7.9}$$

where $\mathcal{Z}(\mathbf{x})$ is a constant over the maximization variable c and it is consequently ignored.

Given a training set of samples $\{\mathbf{x}_n, c_n\}_{n=1}^N$, the parameters θ of a Bernoulli mixtures are usually estimated by maximizing the MLE training criterion [Duda and Hart, 1973], $F_{\text{MLE}}(\theta)$, over the training set using the EM algorithm [Juan and Vidal, 2004]. Where $F_{\text{MLE}}(\theta)$ is the the likelihood of the joint probability over the training set and is formally defined as

$$F_{\text{MME}}(\theta) = \sum_{n=1}^{N} \log(p_\theta(c_n, \mathbf{x}_n))\,. \tag{7.10}$$

7.3 Mixture of Multi-class Logistic Regression

In this section, we propose a mixture of multi-class logistic regression (MMLR) model inspired by the Bernoulli mixture classifier. Given a binary input vector $\mathbf{x} \in \{0,1\}^D$ and a class $c \in \{1,\ldots,C\}$, the proposed MMLR classifier is defined as

$$c^* = \underset{c}{\operatorname{argmax}}\ p_\lambda(c \mid \mathbf{x})\,, \tag{7.11}$$

where the posterior probability is modeled as

$$p_\lambda(c \mid \mathbf{x}) = \sum_{k=1}^{K} p_\lambda(c,k \mid \mathbf{x})\,, \tag{7.12}$$

with k denoting the selected mixture component for the current class, analogously to (7.5). The component and class posterior probability in (7.12) is modeled as a log-linear combination of binary features $f_i(\mathbf{x},c,k)$,

$$p_\lambda(c,k \mid \mathbf{x}) = \frac{\exp(\sum_i \lambda_i f_i(\mathbf{x},c,k))}{\mathcal{Z}(\mathbf{x})}\,, \tag{7.13}$$

where $\mathcal{Z}(\mathbf{x})$ is the normalization constant defined as

$$\mathcal{Z}(\mathbf{x}) = \sum_c \sum_k \exp(\sum_i \lambda_i f_i(\mathbf{x},c,k))\,. \tag{7.14}$$

Note that (7.13) is a multi-class logistic regression model.

Finally, given an index of features $i = (\tilde{c},\tilde{k},d)$ where $\tilde{c}$ ranges in the domain $\{1,\ldots,C\}$, $\tilde{k}$ in $\{1,\ldots,K\}$ and d in $\{0,1,\ldots,D\}$; the feature $f_i(\mathbf{x},c,k) = f_{\tilde{c},\tilde{k},d}(\mathbf{x},c,k)$ is defined as follows

$$f_{\tilde{c},\tilde{k},d}(\mathbf{x},c,k) = \begin{cases} \delta(c,\tilde{c})\delta(k,\tilde{k}) & d=0 \\ \delta(c,\tilde{c})\delta(k,\tilde{k})x_d & 1 \le d \le D \end{cases}\,. \tag{7.15}$$

The proposed model MMLR model is a particular case of log-linear model with hidden variables, and hence, it can be trained using the MMI criterion, which is defined as

$$F_{\text{MMI}}(\lambda) = \sum_{n=1}^{N} \log(p_\lambda(c_n \mid \mathbf{x}_n))\,, \tag{7.16}$$

and can be decomposed as a function of the numerator minus the normalization score as

$$\begin{aligned} F_{\text{MMI}}(\lambda) = &\sum_n \log\left(\sum_k \exp(\sum_i \lambda_i f_i(\mathbf{x}_n,c_n,k))\right) \\ &- \sum_n \log\left(\sum_{c'}\sum_k \exp(\sum_i \lambda_i f_i(\mathbf{x}_n,c',k))\right)\,. \end{aligned} \tag{7.17}$$

As discussed in the introduction the parameter estimation for log-linear models with hidden variables using the MMI criterion can be carried out using the GGIS or the RPROP algorithms [Heigold et al., 2008, Riedmiller and Braun, 1993]. Note, that in the case of one mixture component per class $K = 1$ the sum over k is vanished, and therefore, the proposed model is a particular case of log-linear model, which can be trained using the GIS algorithm [Darroch and Ratcliff, 1972]. Regarding parameter estimation, the main difference between conventional log-linear models and log-linear models with hidden variables, is that the MMI criterion is convex for the former and not it is for the latter.

The MMI criterion has the disadvantage that easily over-fits to the training data. A typically solution to amend this problem is to add a regularization term to the criterion

$$F_{\mathrm{C}}(\boldsymbol{\lambda}) = F_{\mathrm{MMI}}(\boldsymbol{\lambda}) - C \sum_i (\lambda_i^{(0)} - \lambda_i)^2 \,, \tag{7.18}$$

where $\boldsymbol{\lambda}^{(0)}$ is either a reliable estimation of the parameters or simply $\mathbf{0}$. The GGIS and GIS algorithms cannot optimize (7.18), so, the RPROP algorithm is usually used in that case.

7.4 Equivalence Between Classifiers

In this section we prove that the Bernoulli mixture classifier defined in Section 7.2 is *equivalent* to the MMLR model defined in Section 7.3. A generative classifier is said to be *equivalent* to a discriminative classifier if for a given set of generative parameters $\boldsymbol{\theta}$, discriminative parameters, $\boldsymbol{\lambda}$, can be found such that

$$\underset{c}{\operatorname{argmax}}\ p_\theta(\mathbf{x}, c) = \underset{c}{\operatorname{argmax}}\ p_\lambda(c \mid \mathbf{x}) \,; \tag{7.19}$$

and vice-versa. Therefore, an equivalence proof has two parts:

- How to define the discriminative parameters from the generative parameters,and
- how to define the generative parameters from the discriminative parameters.

7.4.1 From Generative to Discriminative Parameters

Unlike the converse direction, it is quite simple to prove that given a Bernoulli mixture classifier it can be re-parameterized into the model proposed in Section 7.3. Left probability in (7.19) can be rewritten as

$$\sum_{k=1}^{K} \exp(\log \pi_{ck} + \sum_{d=1}^{D} x_d \log p_{ckd} + (1 - x_d) \log(1 - p_{ckd})) \,. \tag{7.20}$$

If we group the terms that depend on x_d in the previous equation, then we obtain

$$\sum_k \exp(\log \pi_{ck} + \gamma_{ck} + \sum_d x_d \log \frac{p_{ckd}}{1 - p_{ckd}}) \,, \tag{7.21}$$

with

$$\gamma_{ck} = \sum_d \log(1 - p_{ckd}) \,. \tag{7.22}$$

Finally, if (7.21) is re-parameterized as

$$\lambda_{ck0} = \log \pi_{ck} + \gamma_{ck} \,, \tag{7.23}$$

$$\lambda_{ckd} = \log \frac{p_{ckd}}{(1 - p_{ckd})} \,, \tag{7.24}$$

then a unnormalized discriminative MMLR classifier which is equivalent to the generative Bernoulli mixture classifier is obtained. Note that the normalization constant $\mathcal{Z}(\mathbf{x})$ is irrelevant during classification.

7.4.2 From Discriminative to Generative Parameters

In this subsection we prove that the MMLR classifier defined in in Section 7.3, is *equivalent* to the Bernoulli mixture classifier defined in Section 7.2. For doing that, we prove that the log-lineal model in (7.13) is equivalent to the joint probability defined in (7.6) but for a constant that does not depend on neither the class nor the component. Note that although this transformation may seem trivial, it is not the case, since the discriminative parameters does not need to verify any constraint, while the generative parameters must verify (7.8). We start by rewriting (7.6) in the following form

$$p_\theta(\mathbf{x}, c, k) = \exp\Big(\log \pi_{ck} + \gamma_{ck} + \sum_{d=1}^{D} x_d \log \frac{p_{ckd}}{1 - p_{ckd}}\Big) \,. \tag{7.25}$$

Log-linear models such as (7.13) are the optimal distribution that maximize the *entropy* while verifying the following expectation constraints [Berger et al., 1996, Jaynes, 1957]

$$N_i = E[N_i], \qquad \forall i \,, \tag{7.26}$$

where N_i are the sample counts of the feature f_i, and $E[N_i]$ is the expected value of the corresponding feature count. Each log-linear parameter, λ_i, corresponds to a Lagrange multiplier [Boyd and Vandenberghe, 2004] that accounts for the i-th constraint in (7.26), under the the dual maximum entropy formulation point of view. By analyzing the log-linear models form the maximum entropy point of view, the normalization constraint in (7.8) can be introduced as another constraint and Lagrange multiplier, say λ_{000}. This new parameters is not related with any feature since it accounts for an additional constraint. Moreover, introducing this parameter does not modifies the posterior probabilities computed by the log-linear model in (7.13) with the features in (7.15). Note that introducing the new parameter, λ_{000}, is equivalent to multiplying the numerator and denominator by $\exp(\lambda_{000})$. Therefore, Equation (7.13) is rewritten as

$$p_\lambda(k, c \mid \mathbf{x}) = \frac{1}{\mathcal{Z}(\mathbf{x})} \exp(\lambda_{000} + \lambda_{ck0} + \sum_d x_d \lambda_{ckd}) \,, \tag{7.27}$$

for an arbitrary and unknown λ_{000}.

There are two types of parameters in (7.27) and (7.25). The first group corresponds to the parameters that multiply the input features, x_d, while the second group multiplies the feature 1. In other words, equivalence between (7.25) and (7.27) is proven if the following equivalences are verified

$$\sum_d x_d \lambda_{ckd} = \sum_d x_d \big[\log p_{ckd} - \log(1 - p_{ckd}) \big] \,, \tag{7.28}$$

$$\lambda_{000} + \lambda_{ck0} = \log \pi_{ck} + \gamma_{ck} \,, \tag{7.29}$$

where π_{ck} must verify (7.8).

Equation (7.28) is verified if

$$\lambda_{ckd} = \log p_{ckd} - \log(1 - p_{ckd}) \,, \tag{7.30}$$

from where we can work out the value of p_{ckd}

$$p_{ckd} = \frac{\exp(\lambda_{ckd})}{1 + \exp(\lambda_{ckd})} \,. \tag{7.31}$$

Although the equivalence in (7.29) is more difficult to verify, the value of π_{ck} can be worked out as

$$\pi_{ck} = \exp(\lambda_{000}) \cdot \exp\left(\lambda_{ck0} - \gamma_{ck}\right) \,, \tag{7.32}$$

where γ_{ck} is defined in (7.22) with the values of p_{ckd} defined by (7.31).

Recall that the prior parameters π_{ck} must sum up to 1 and, hence, by plugging (7.32) into (7.8), the following constraint must be verified

$$\sum_{c'k'} \exp(\lambda_{000}) \exp\left(\lambda_{c'k'0} - \gamma_{c'k'}\right) = 1 \,, \tag{7.33}$$

from where the value of $\exp(\lambda_{000})$ is worked out

$$\exp(\lambda_{000}) = \left[\sum_{c'k'} \exp\left(\lambda_{c'k'0} - \gamma_{c'k'}\right) \right]^{(-1)} \,. \tag{7.34}$$

Finally, the solution is given by plugging (7.34) into (7.32) as follows

$$\pi_{ck} = \frac{\exp\left(\lambda_{ck0} - \gamma_{ck}\right)}{\sum_{c'k'} \exp\left(\lambda_{c'k'0} - \gamma_{c'k'}\right)} \,. \tag{7.35}$$

Note that if we had not introduced the additional discriminative parameter, λ_{000}, then we could not have found a transformation from the discriminative parameters into the generative ones.

Finally, if we define the generative parameters as indicated in (7.31) and (7.35), then the generative model in (7.25) is equivalent to the discriminative model in (7.27), but for

the constant factor $\mathcal{Z}(\mathbf{x})$. Given the previous relationship, it is straightforward to prove the equivalence between the parameters for the full classifiers as follows:

$$\begin{aligned}\hat{c} &= \underset{c}{\operatorname{argmax}}\ p_\lambda(c \mid \mathbf{x}) = \underset{c}{\operatorname{argmax}} \sum_k p_\lambda(c, k \mid \mathbf{x}) \\ &= \underset{c}{\operatorname{argmax}} \sum_k \mathcal{Z}(\mathbf{x}) p_\lambda(c, k \mid \mathbf{x}) \\ &= \underset{c}{\operatorname{argmax}} \sum_k p_\theta(c, k, \mathbf{x}) = \underset{c}{\operatorname{argmax}}\ p_\theta(c, \mathbf{x})\end{aligned} \tag{7.36}$$

In summary, the MMLR adds no improvement or flexibility over the Bernoulli mixture classifier apart from the possibility of discriminatively training it. The transformations provided in this section not only allow us to initialize the MMLR classifier with the MLE parameters, but also to train a MMLR classifier and then obtain its equivalent generative parameters so that they can be analyzed.

7.5 Experiments

Experiments were carried out in order to assess the proposed Bernoulli mixture MMI training algorithm with respect to the standard EM training algorithm. For the experimentation, we focused on the non-touching part of the Indian digits database from the well-known Arabic cheque database provided by CENPARMI (see Section 3.1).

In order to obtain properly normalized images, both in size and position, two simple preprocessing steps were applied. First, each digit image was pasted onto a square background whose center was aligned with the digit mass center. This square background was a white image large enough (64×64) to accommodate most samples. Second, given a size S, each digit image was subsampled into $S \times S$ pixels, from which its corresponding binary vector was built (with a dimension of $D = S^2$ binary bits). Figure 7.1 shows some preprocessed example for each digit. Particularly interesting is the 9 of the middle example which clearly is an outlier and would affect the discriminative prototypes if they are allowed to get over-fitted.

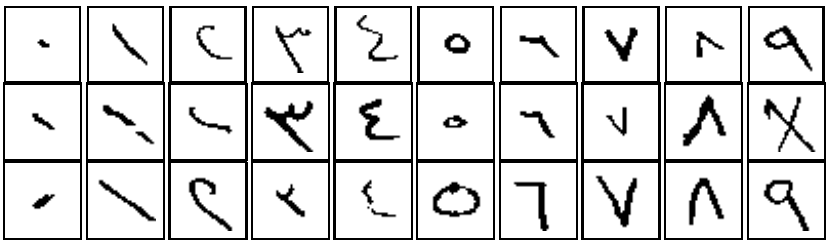


Figure 7.1: Some examples for each Indian digit with size 30×30

All experiments were carried out using the standard experimental procedure for *classification error rate (CER)* estimation in the CENPARMI Indian digits task. We divided the experimentation in two parts. In the first one experiments were carried out using one mixture

component per class, in order to better understand the impact of the MMI training. In the second part we extend the experimentation by using several mixture components per class. Reported results will show that MMI training clearly outperforms conventional MLE training.

7.5.1 Experiments with One Mixture Component per Class

We began by comparing the performance of the RPROP and GIS algorithms. Remember that the GIS algorithm can be used in the case of one mixture component per class. Results are shown in Figure 7.2. The experiment was carried out using images subsampled at 30×30 and a Bernoulli model trained with the MLE criterion as the initial model for initialize both algorithms. The convergence of the RPROP algorithm is considerably faster than the GIS algorithm, which after $20K$ iterations is still far from the RPROP performance after $1K$ iterations. This difference in the speed convergence results in better classification error rates. After 100 iterations, the RPROP algorithm achieves a 2.9% CER, while the GIS algorithm after 100 iterations scores 4.2%; in fact after $20K$ iterations the GIS algorithm achieves a 3.8% which is still far from the CER obtained with the RPROP. Note that both algorithms would converge in the limit to the same CER (except for numerical precisions).

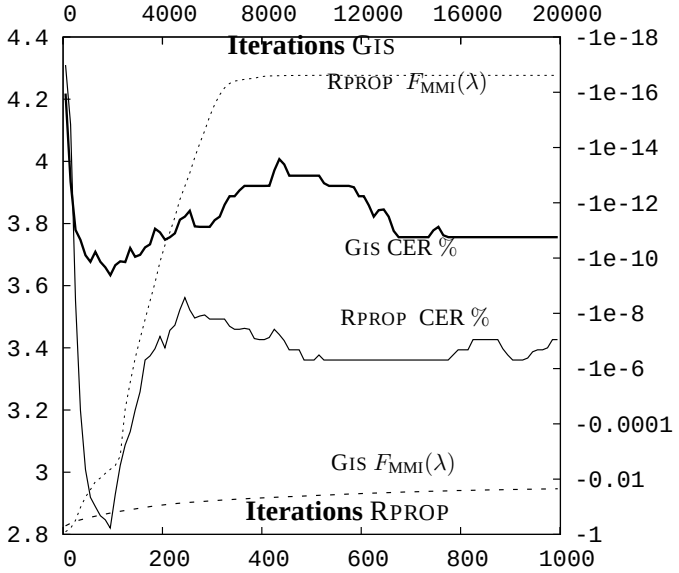


Figure 7.2: GIS and RPROP performance comparison. On left y-axis CER % on testing, on right y-axis the MMI criterion

In the previous experiment the discriminative parameters were initialized using the MLE criterion with the aid of the transformation explained in section 7.4.1. The figure 7.3 compares the performance of the RPROP algorithm using the previous initialization and an uniform initialization of the generative parameters before the conversion. Although, it is observed that the MLE initialization achieves better performance, the uniform initialization, which scored

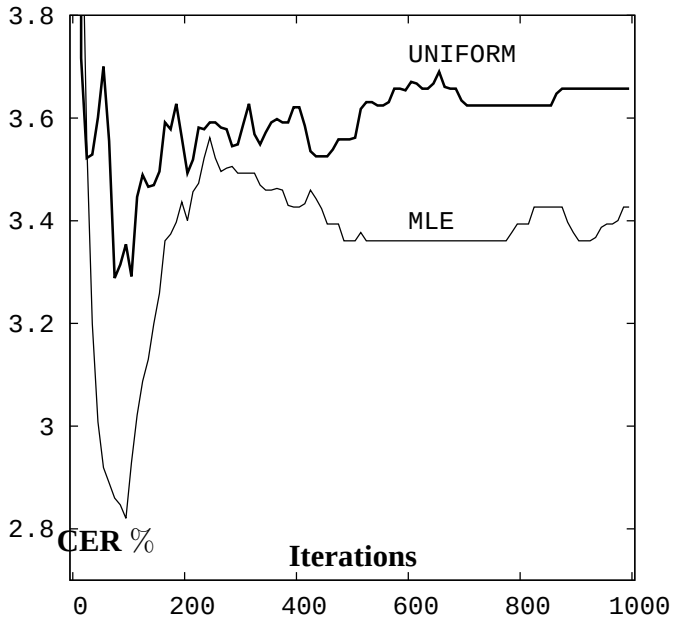


Figure 7.3: Comparison between MLE and uniform initialization using the RPROP algorithm

15.9% CER after first iteration, achieves similar error rates after 20 iterations. Note that in the limit, both initializations would converge to the same CER (except for numerical precision errors) but not for the same prototypes since the optimal set of parameters for the MMI criterion and log-linear models is not unique.

In all previous experiments images were subsampled at 30×30, since it was proved to be a optimal value for Bernoulli mixture classifiers [Juan and Vidal, 2004]. Figure 7.4 shows a comparison between the RPROP algorithm and the standard MLE criterion for several subsample sizes: 14×14, 20×20 and 30×30. The results show that the subsample size 30×30 is the best choice for both training algorithms. Moreover, the profit of the discriminative training and the RPROP stability is increased accordingly to the the subsample size. The best performance that we obtained using one mixture component per class, is achieved by RPROP with 30×30 at a score of 2.8%. In the cited work [Juan and Vidal, 2004], the best result that authors obtain by means of a latent mixture of Bernoulli classifier is 2.5%, which is very similar to the our 2.8% using only one mixture component per class and MMI training.

Although, the RPROP algorithm reduces the test error very quickly in its first 100 iterations; for more iterations, the model gets over-fitted to the training data and then the error is dramatically increased. This over-fitting problem is amended by introducing the proposed regularization term in (7.18). Figure 7.5 depicts to which extend the RPROP algorithm is able to generalize from the sample using the regularization term. Several values of C were scanned, where $C = 0$ stands for no regularization term.

As previously commented, an interesting feature of Bernoulli prototypes is their direct vi-

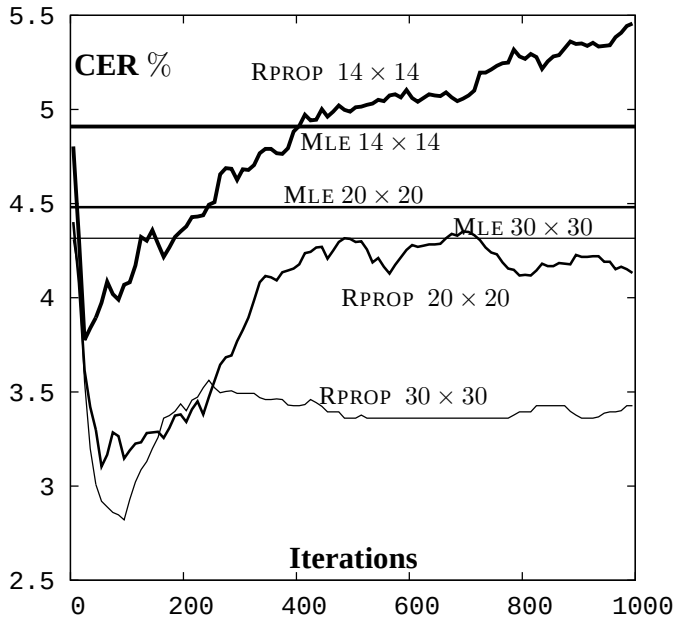


Figure 7.4: Comparison between MLE training and MMI training (RPROP) for several subsample sizes

sual interpretation. Probabilities of the Bernoulli prototypes can be interpreted as gray level pixels (1=black, 0=white). Using this approach the Bernoulli prototypes for several methods are shown in Figure 7.9. It is observed that the prototypes obtained with the MLE algorithm are, for each class, the average of all training samples. However, for discriminative methods, every individual bit is independently modified depending on its discrimination level. This fact is much more evident when the uniform initialization is used, in this case, those pixels which are not discriminative, keep their initial value (gray). Regarding the regularization term, it is observed that it tend to keep the prototypes similar to the initialization. The RPROP prototypes (without regularization term) differ more from the initialization, when compared to the GIS prototypes since the latter converges much more slowly. Note that if no regularization is used, then the prototypes tend to discriminate outliers. For instance, part of 9 in the middle example in Figure 7.1 is learned by the 9 discriminative prototype provide that this part is usually white in the other classes.

The convergence process of the prototypes with the RPROP algorithm is depicted in Figure 7.10 for the MLE initialization and in Figure 7.11 for the uniform initialization. It is clearly observed the fact that the MLE initialized prototypes seem more natural is just because of the initialization. Actually, by comparing both figures, it is seen that both initialization tend to modify the same prototype bits. Moreover, it is observed that if the RPROP algorithm is allowed to iterate more (1 000 iterations), then the prototypes strengthen features that are specific to the training corpora, for instance, features to discriminate between outliers.

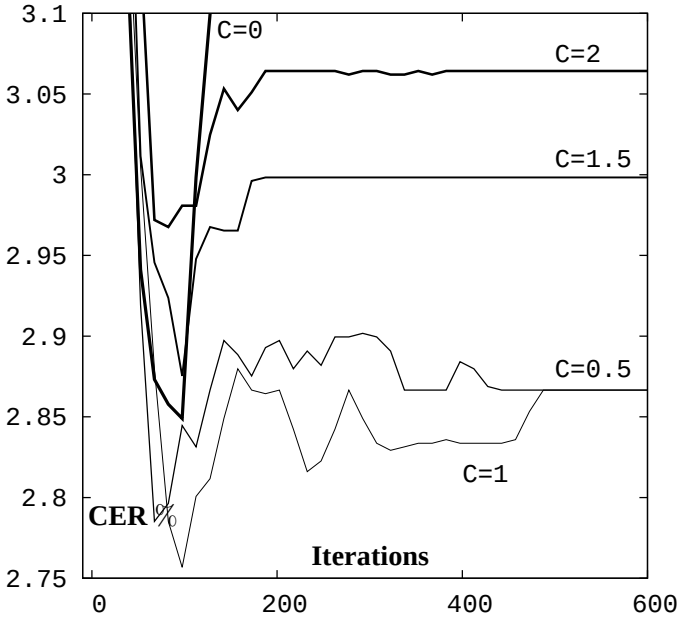


Figure 7.5: Impact of the regularization term in the RPROP algorithm

7.5.2 Experiments with Several Mixture Components per Class

In this section we extended the previous experiments by using the full proposed model, that is, by using mixtures. In the first experiment, as we did in previous experiments, we compared different initializations of the Bernoulli classifier prior to its transformation into a MMLR model. We tested two different initializations: an initialization using the EM algorithm and a hypercube initialization. The comparison was carried out using $K = 5$ mixture components, although similar results were obtained for other values of K. In the hypercube initialization all parameters were uniformly initialized and then randomly perturbed. In the EM initialization, the initial Bernoulli mixture classifier was trained using the EM algorithm, which in turn, was initialized using a conventional Bernoulli classifier trained with the MLE criterion. Afterwards, several iterations of the RPROP algorithm were performed to discriminatively train the MMLR model. Results are reported in Figure 7.6. These results are statistically significant since each point in the plot is the average of 50 repetitions. It is observed that the discriminatively trained Bernoulli mixture classifier improves the generative results, which correspond to the left-most point in the EM curve. However, the hypercube initialization outperforms the generative initialization.

In order to assess the repercussion of the number of mixtures components, K, we carried experiments varying it in the range $\{1, 2, 4, 6, 8, 10\}$ and using the hypercube initialization. The results are shown in Figure 7.7, each point is the average of 50 repetitions. A big improvement is obtained by only 2 components. The more components are added, the larger

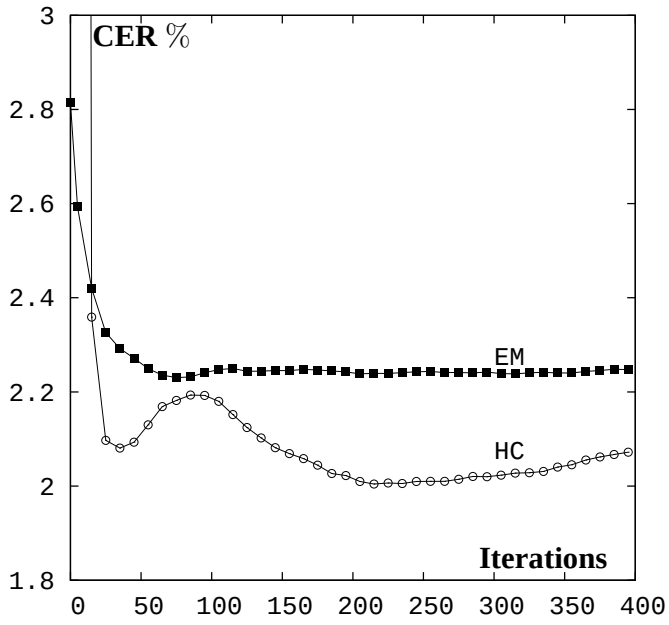


Figure 7.6: Comparison between EM and and HC initializations using the RPROP algorithm

the improvement becomes until it is saturated at 6 components. Results for $K > 6$ are not plotted since they are identical to $K = 6$. Note that the behavior differs from the generative Bernoulli mixture classifier. According to Juan and Vidal [2004], generative Bernoulli mixtures achieves the best results around $K = 15$. Moreover, generative Bernoulli mixtures with $K = 15$ have an error of 2.7% while in Figure 7.7 using $K = 6$ we obtain an error about 2.0%. The discriminative Bernoulli mixtures obtain an improvement of 25% over the generative Bernoulli mixtures by using half of the parameters. To our knowledge, the best result in this database is approximately 1.9% [Romero et al., 2007], which is similar to our result but using much more parameters.

A final experiment were performed to assess the behavior of the regularization term. Several values of C were scanned ranging from 0 (no regularization) to 0.5. Results are shown in Figure 7.8. It is observed that without regularization the error is unstable and it increases (over-fits) along with the iterations. In contrast, regularization makes the error more stable while providing the same performance. In particular for $C = 0.001$ the CER is stabilized around 2%.

As we did previously with the experiments without mixture components, Bernoulli mixture prototypes for a model with 6 mixture components per class are shown in Figure 7.12. The mixture coefficients are also represented by means of a gray level border, where black denotes 1 and white 0, as in the Bernoulli prototypes. The selected model was trained without regularization and using an hypercube initialization, in this respect, the Bernoulli prototypes are very similar to those of Figure 7.9 using uniform initialization without regularization. The

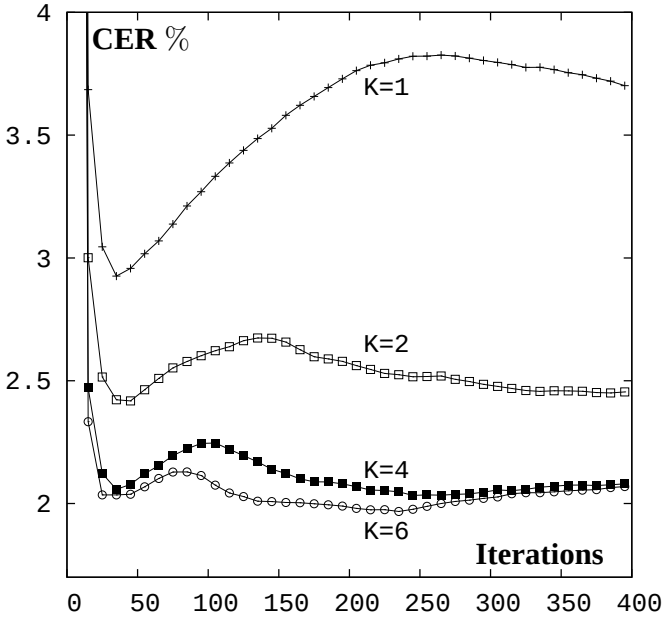


Figure 7.7: CER (%) of discriminative Bernoulli mixture classifier for several number of mixture components (K) using the hypercube initialization

interesting point of Figure 7.12 is that in all classes there is always one mixture component which coefficient is very near to 1. That is, other mixture components are apparently irrelevant. However, we performed experiments in which these mixture coefficients were removed, and worse results were obtained.

7.6 Concluding Remarks

A mixture of multi-class logistic regression (MMLR) model has been proposed for binary data. This model was inspired by Bernoulli mixture model. Afterwards, the equivalence between both classifiers has been proved. Consequently we obtained two results. On the one hand, we have provided a MMI training scheme for Bernoulli mixture classifiers. On the other hand, discriminative parameters can be interpreted by transforming them into generative parameters.

This new training scheme has been tested and compared, with the generative MLE training scheme, using different initialization methods and regularization terms, on the well-known CENPARMI Indian digits database. The proposed MMI training scheme outperforms the generative MLE criterion using half of the parameters.

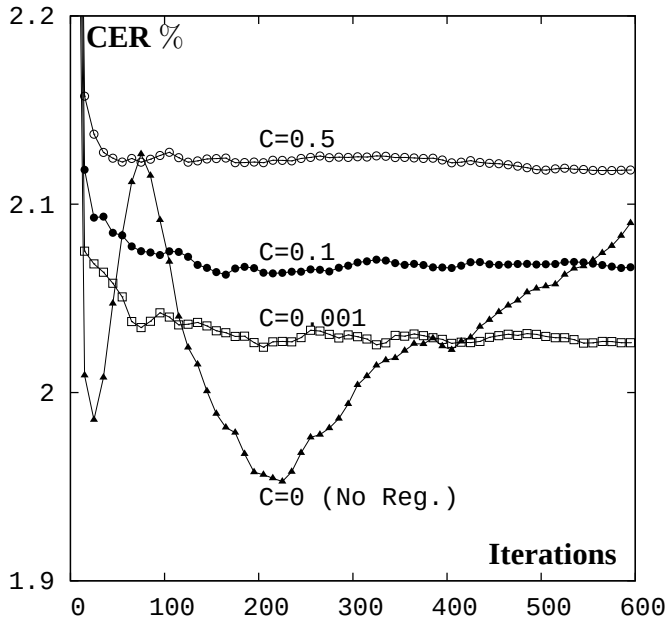


Figure 7.8: Impact of the regularization term over the CER (%)

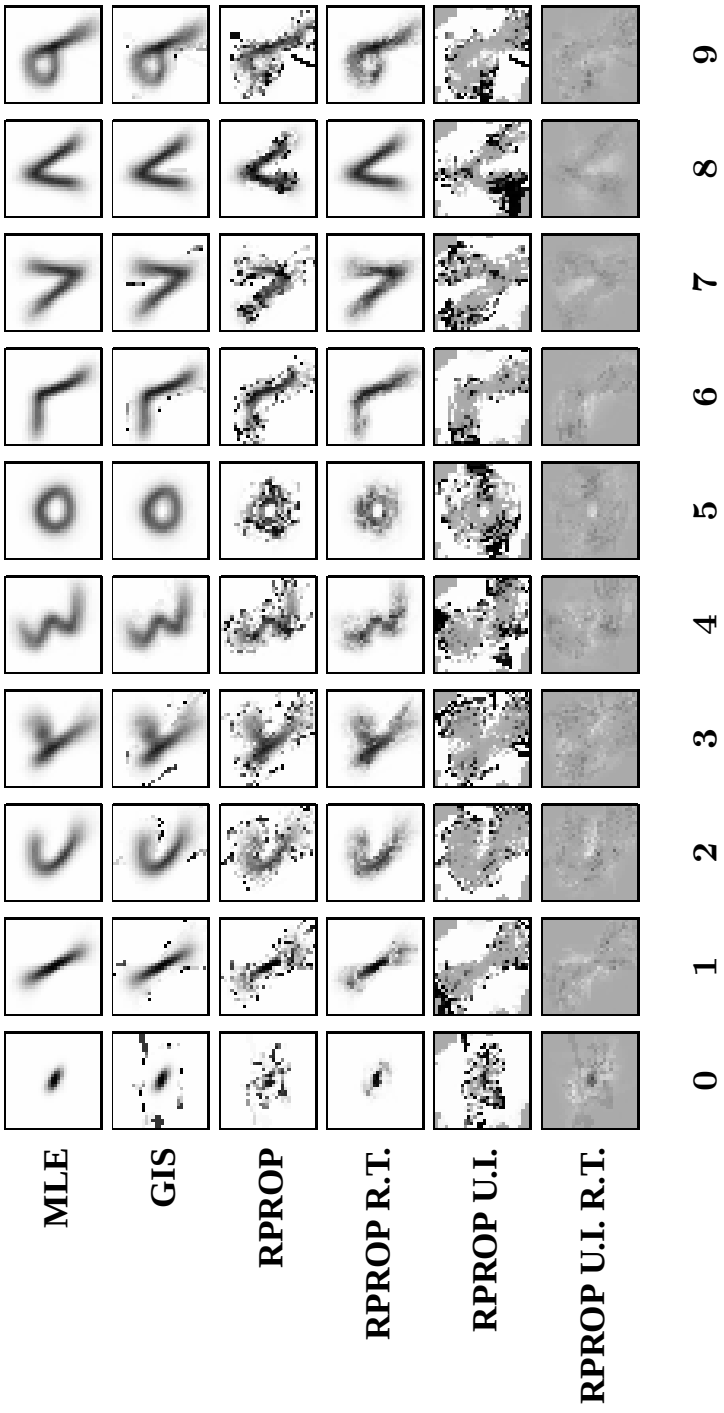


Figure 7.9: Bernoulli prototypes for several training algorithms: RPROP with and without regularization term (R.T.), and GIS. The two initializations for RPROP are also depicted: MLE and uniform (U.I.). The training algorithms performed 100 iteration for the RPROP and 2 000 for the GIS

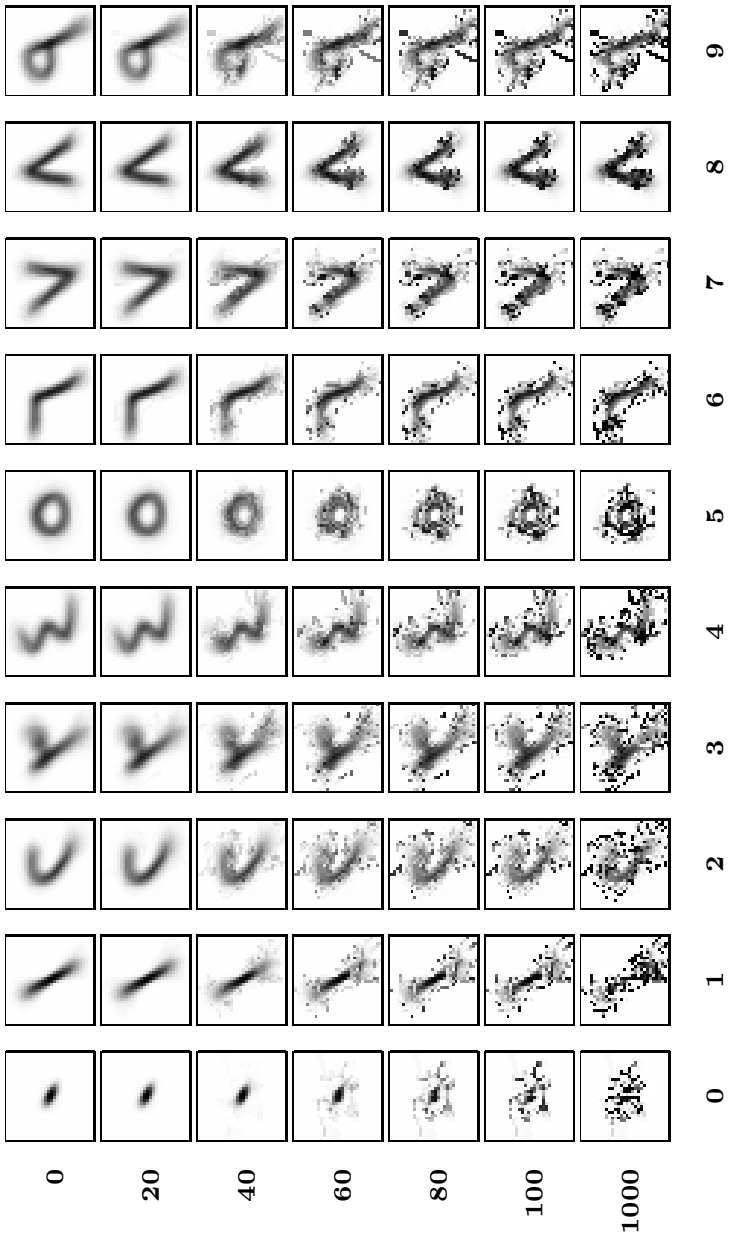


Figure 7.10: Bernoulli prototypes for some iterations of the RPROP algorithm without regularization initialized with the MLE parameters

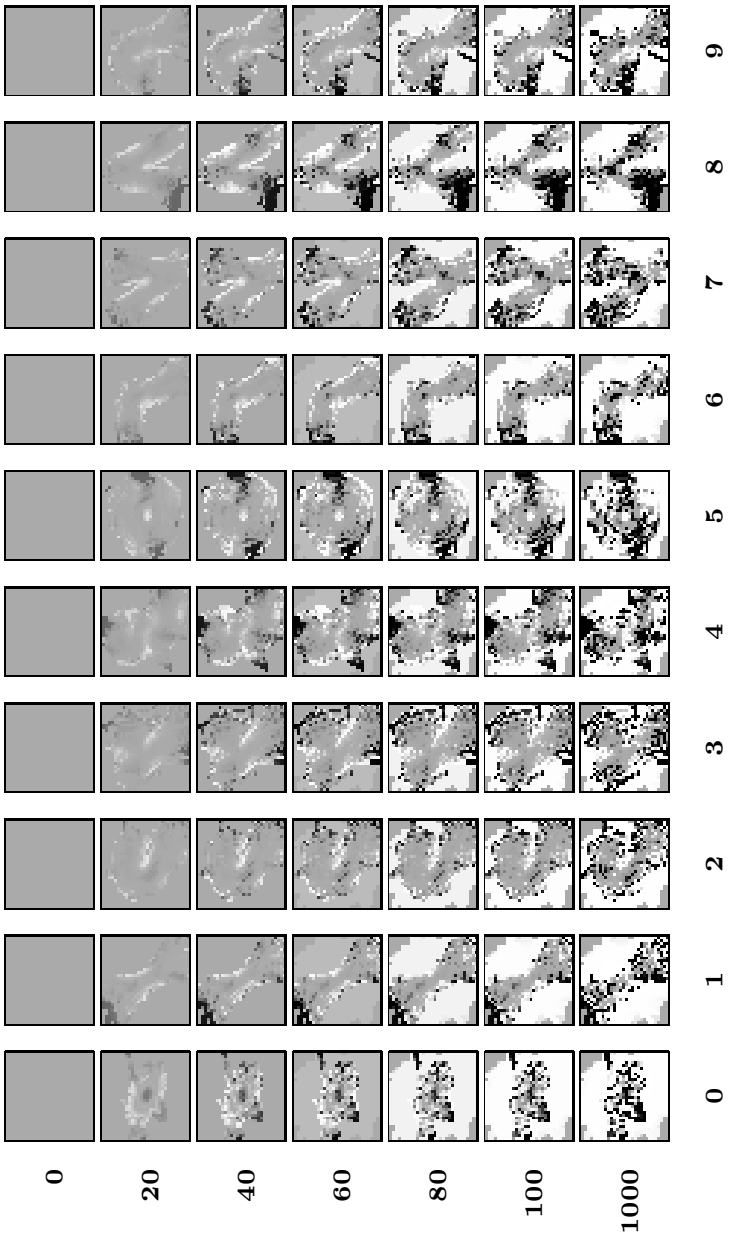


Figure 7.11: Bernoulli prototypes for some iterations of the RPROP algorithm without regularization initialized with uniform parameters

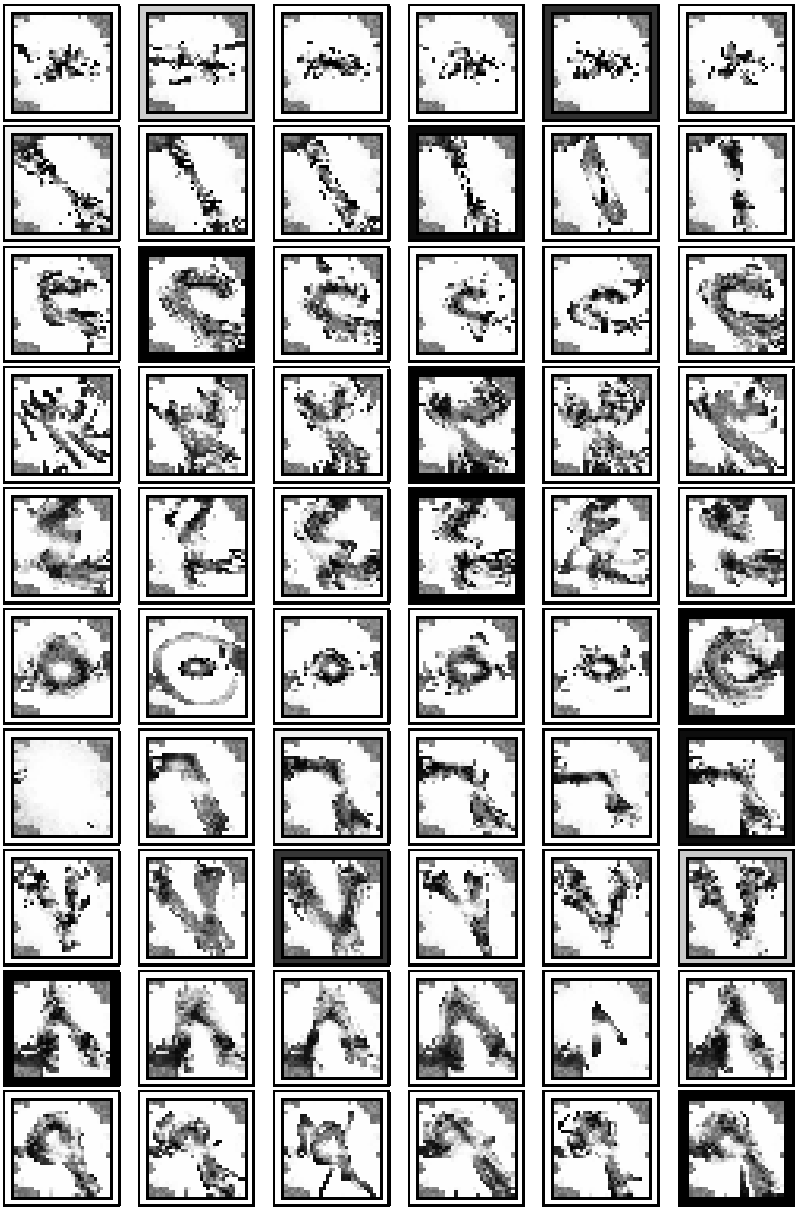


Figure 7.12: Bernoulli mixture prototypes using 6 mixture components per class and hypercube initialization. Rows and columns are related to classes and mixture components respectively

Bibliography

Adam L. Berger, Vincent J. Della Pietra, and Stephen A. Della Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22:39–71, March 1996. ISSN 0891-2017.

Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, March 2004. ISBN 0521833787.

J. N. Darroch and D. Ratcliff. Generalized Iterative Scaling for Log-Linear Models. *The Annals of Mathematical Statistics*, 43(5):1470–1480, 1972. ISSN 00034851. doi: 10.2307/2240069.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977. doi: http://dx.doi.org/10.2307/2984875. URL `http://dx.doi.org/10.2307/2984875`.

R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. J. Wiley and Sons, 1973.

Adrià Giménez, Jesús Andrés-Ferrer, Alfons Juan, and Nicolás Serrano. Discriminative Bernoulli Mixture Models for Handwritten Digit Recognition. In *11th International Conference on Document Analysis and Recognition ICDAR 2011*, pages 558 – 562. IEEE Computer Society Conference Publishing Services, September 2011. ISBN 978-0-7695-4520-2. doi: 10.1109/ICDAR.2011.118.

Georg Heigold, Thomas Deselaers, Ralf Schlüter, and Hermann Ney. A GIS-like training algorithm for log-linear models with hidden variables. In *ICASSP*, pages 4045–4048, 2008.

E. T. Jaynes. Information Theory and Statistical Mechanics. *Phys. Rev.*, 106(4):620–630, May 1957. doi: 10.1103/PhysRev.106.620.

A. Juan and E. Vidal. Bernoulli mixture models for binary images. In *Proc. of ICPR 2004*, volume 3, Cambridge (UK), August 2004.

Martin Riedmiller and Heinrich Braun. A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP Algorithm. In *IEEE International Conference on Neural Networks*, pages 586–591, 1993.

V. Romero, A. Giménez, and A. Juan. Explicit Modelling of Invariances in Bernoulli Mixtures for Binary Images. In *3rd Iberian Conference on Pattern Recognition and Image Analysis*, volume 4477 of *LNCS*, pages 539–546. Springer-Verlag, Girona (Spain), June 2007.

CHAPTER 8

DISCRIMINATIVE BHMM CLASSIFIER

Contents

8.1 Introduction

Handwritten word classifiers based on HMMs, and in particular in BHMMs, are *generative models*. As discussed in the previous chapter, generative models are usually trained using the MLE. The MLE is aimed at explaining the probability distribution that underlies in the training sample. However, we are interested in simply classifying samples, and there is no guarantee that the MLE parameters are the most suitable for classifying, even though they have been proved to be competitive. We also saw that discriminative models and criteria are a good alternative to generative models, since they are aimed at classifying the data without explaining the data distribution itself. A well-known example of dicriminative training criterion is the *maximum mutual information (MMI)* criterion.

In Chapter 7 a MMI training scheme for Bernoulli mixture classifiers was proposed and tested in a task of isolated handwritten digit recognition. The proposed approach was based on the idea of finding a similar discriminative classifier to the Bernoulli mixture classifier, and then prove the equivalence between both classifiers. The results analyzed report that discriminatively trained Bernoulli mixture classifier outperforms the generative Bernoulli mixture classifiers. In this chapter this work is extended to more complex models, the BHMMs, which are assessed in a complex isolated word recognition task.

More precisely, the contributions of this work are the follows:

1. We propose a particular case of log-linear HMM (LLHMM) classifier, which can also be interpreted as a semi-Markov conditional Markov chain (semi-CRF), for binary data inspired by the BHMM classifier.

2. We prove the equivalence between BHMMs and the proposed discriminative model for binary data.

3. We provide a discriminative training scheme for BHMM classifiers by means of their equivalence with LLHMMs, and analyze several discriminative training criteria such as MMI.

4. We provide the capability to understand discriminative parameters from a generative point of view by means of their equivalence with BHMMs.

The remainder of the chapter is organized as follows. The proposed LLHMM or semi-CRF classifier for binary data is described in Section 8.2. The Section 8.3 proves equivalence between both classifiers, and in Section 8.4 the parameter estimation for the LLHMM is analyzed. The proposed training scheme is deeply analyzed on the RIMES database in Section 8.3. We conclude the paper by summarizing and discussing the most important results and future research directions.

8.2 Log-linear HMM for Binary Data

In this section, we propose a discriminative classifier inspired by the BHMM classifier for isolated handwritten words (Chapter 5). The discriminative classifier proposed is based on a log-linear model, which is inferred from the parameters of a BHMM classifier. In what

follows, we define the log-linear model and how a log-linear HMM (LLHMM) discriminative classifier can be built using it.

8.2.1 BHMM Inspired Log-linear Model

Using the equations we introduced in Chapters 4 and 5, we get that the BHMM classifier can be expressed by plugging (4.4), (5.4), and (5.1) as follows

$$p_{\boldsymbol{\theta}}(O, S) = \sum_{\mathbf{i},\mathbf{q},\mathbf{k}} p_{\boldsymbol{\theta}}(O, S, \mathbf{i}, \mathbf{q}, \mathbf{k}) \,, \tag{8.1}$$

where by $\mathbf{i}, \mathbf{q}, \mathbf{k}$ we denote the 3 latent variables of the model, namely: the segmentation, $\mathbf{i}$, of O into L segments as defined in (5.2); the state sequence, $\mathbf{q} = (q_0, q_1, \ldots, q_{T+1})$; and the emission component at each state, $\mathbf{k}$. According to the given segmentation $\mathbf{i}$, the state sequence $\mathbf{q}$ must be valid, which implies that if t belongs to the l-th segment, then the state q_t must be a possible state of the character-level BHMM for the corresponding symbol s_l. Similarly, $\mathbf{k} = (k_1, \ldots, k_T)$ must be a valid integer sequence where k_t denotes the selected mixture component for state q_t, among all the components of the state.

The joint probability in the right-hand-side of previous equation, $p_{\boldsymbol{\theta}}(O, S, \mathbf{i}, \mathbf{q}, \mathbf{k})$, is decomposed left-to-right as follows

$$p_{\boldsymbol{\theta}}(O, S, \mathbf{i}, \mathbf{q}, \mathbf{k}) = \pi_S p_{\boldsymbol{\theta}}(O, \mathbf{i}, \mathbf{q}, \mathbf{k} \mid S) = \pi_S \, p_{\boldsymbol{\theta}}(\mathbf{i}, \mathbf{q} \mid S) \, p_{\boldsymbol{\theta}}(O, \mathbf{k} \mid \mathbf{i}, \mathbf{q}, S) \tag{8.2}$$

where $p_{\boldsymbol{\theta}}(\mathbf{i}, \mathbf{q} \mid S)$ is the transition probability of the word-level BHMM and $p_{\boldsymbol{\theta}}(O, \mathbf{k} \mid \mathbf{i}, \mathbf{q}, S)$ the emission probability. The transition probabilities are then decomposed into

$$p_{\boldsymbol{\theta}}(\mathbf{i}, \mathbf{q} \mid S) := \prod_{l=1}^{L} a_{s_l}(I, q_{i_l}) \cdot a_{s_l}(q_{i_{l+1}-1}, F) \prod_{t=i_l}^{i_{l+1}-2} a_{s_l}(q_t, q_{t+1}) \tag{8.3}$$

where the first product accounts for the input, $a_{s_l}(I, q_{i_l})$, and output, $a_{s_l}(q_{i_{l+1}-1}, F)$, transitions of the embedded model for the character s_l; and where the second product are the inner transitions within the embedded character model. In the remaining of the chapter, we will not differentiate between inner and outer transitions since this is a well-known characteristic of HMM, and by extension to our BHMM model. Furthermore, this significantly simplifies the notation. For instance, previous equation is expressed as

$$p_{\boldsymbol{\theta}}(\mathbf{i}, \mathbf{q} \mid S) := \prod_{l,t} a_{s_l}(q_t, q_{t+1}) \tag{8.4}$$

where we have also omitted the boundaries of the products, which can always be traced back to (8.3).

Similarly, the emission probability is decomposed as follows

$$p_{\boldsymbol{\theta}}(O, \mathbf{k} \mid \mathbf{i}, \mathbf{q}, S) := \prod_{l=1}^{L} \prod_{t=i_l}^{i_{l+1}-1} \tau_{s_l q_t}(k_t) \prod_{d=1}^{D} p_{s_l q_t k_t d}{}^{o_{td}} (1 - p_{s_l q_t k_t d})^{(1-o_{td})} \,. \tag{8.5}$$

where again by omitting the product boundaries is simplified to

$$p_{\boldsymbol{\theta}}(O, \mathbf{k} \mid \mathbf{i}, \mathbf{q}, S) := \prod_{l,t} \tau_{s_l q_t}(k_t) \prod_{d} p_{s_l q_t k_t d}{}^{o_{td}} (1 - p_{s_l q_t k_t d})^{(1-o_{td})} \,. \tag{8.6}$$

with $\tau_{s_l q_t}(k_t)$ and $\mathbf{p}_{s_l q_t k_t}$ being the prior and prototype of the k-th mixture component at state q_t of the character s_l.

Consequently, the model in (8.2) can be expressed as follows

$$p_{\boldsymbol{\theta}}(O, S, \mathbf{i}, \mathbf{q}, \mathbf{k}) = \exp\big(\log \pi_S + \log p_{\boldsymbol{\theta}}(\mathbf{i}, \mathbf{q} \mid S) + \log p_{\boldsymbol{\theta}}(O, \mathbf{k} \mid \mathbf{i}, \mathbf{q}, S)\big) \tag{8.7}$$

where the logarithms of the probabilities are given by

$$\log p_{\boldsymbol{\theta}}(\mathbf{i}, \mathbf{q} \mid S) = \sum_{l,t} \log a_{s_l}(q_t, q_{t+1}) \tag{8.8}$$

and

$$\log p_{\boldsymbol{\theta}}(O, \mathbf{k} \mid \mathbf{i}, \mathbf{q}, S) = \sum_{l,t} (\log \tau_{s_l q_t}(k_t) + \xi_{s_l q_t}(k_t) + \sum_{l,t,d} o_{td} \log \frac{p_{s_l q_t k_t d}}{(1 - p_{s_l q_t k_t d})} \,, \tag{8.9}$$

with $\xi_{cq}(k)$ defined as

$$\xi_{cq}(k) = \sum_{d} \log\left(1 - p_{cqkd}\right). \tag{8.10}$$

Note that the term $\xi_{cq}(k)$ is easily obtained when applying the logarithm to (8.6) by rearranging terms similarly to what we did in Chapter 7.

At this point, we reparameterize the probabilities in terms of the new parameters, $\boldsymbol{\lambda}$, as follow

$$\lambda_S = \log \pi_S \,, \tag{8.11}$$

$$\lambda_{cqq'} = \log a_c(q, q') \,, \tag{8.12}$$

$$\lambda_{cqk} = \log \tau_{cq}(k) + \xi_{cq}(k) \,, \tag{8.13}$$

$$\lambda_{cqkd} = \log \frac{p_{cqkd}}{1 - p_{cqkd}} \,, \tag{8.14}$$

for each character, c; states, q and q'; mixture component, k; and input dimension, d.

Provided the previous parameterization, the original joint probability in (8.2) is alternatively expressed as follows

$$p_{\boldsymbol{\lambda}}(O, S, \mathbf{i}, \mathbf{q}, \mathbf{k}) = \exp(\lambda_S + \sum_{l,t} \lambda_{s_l q_t q_{t+1}} + \sum_{l,t} \lambda_{s_l q_t k_t} + \sum_{l,t,d} o_{td} \lambda_{s_l q_t k_t d}) \tag{8.15}$$

In order to simplify notation, we adopt here the standard and powerful notation of log-linear models. We define an index m that ranges through all the subindexes of the previous equation, i.e., m ranges from $\{S\}$ over $\{c, q, q'\}$ and $\{c, q, k\}$ to $\{c, q, k, d\}$. We also introduce a function $g_m(O, S, \mathbf{i}, \mathbf{q}, \mathbf{k})$ that equals to the number of times the parameter λ_m is used, except for the parameters $\{\lambda_{cqkd}\}$. In this case, the function $g_m(O, S, \mathbf{i}, \mathbf{q}, \mathbf{k})$ with $m = cqkd$,

counts the number of times the d-th bit is *set* and has been generated with the k-th component in the state, q, of the character, c. The simplest case of the function is that of the prior parameters λ_S for which $g_m = 1$ (with $m = S$).

The proposed notation simplifies (8.15) into

$$p_{\boldsymbol{\lambda}}(O, S, \mathbf{i}, \mathbf{q}, \mathbf{k}) = \exp\Big(\sum_{m \in \mathcal{M}(O,S,\mathbf{i},\mathbf{q},\mathbf{k})} \lambda_m g_m(O, S, \mathbf{i}, \mathbf{q}, \mathbf{k})\Big) \tag{8.16}$$

where by $\mathcal{M}(O, S, \mathbf{i}, \mathbf{q}, \mathbf{k})$ we denote the set of values through which the index m ranges. It is important to notice that this set depends on all the variables, namely $O, S, \mathbf{i}, \mathbf{q}, \mathbf{k}$; and changes with them. However, it is simpler to define $\mathcal{M}$ as the union of all the possible indexes that our parameters require and replace the functions g_m by the so-called *feature functions* $f_m(O, S, \mathbf{i}, \mathbf{q}, \mathbf{k})$, which are equal to g_m if m is an index of a required parameter and 0 otherwise. For instance, consider again the word prior example with the new domain $\mathcal{M}$. In this case, the index m can take the value of any word, S' in the vocabulary; and then the feature function is defined as

$$f_{S'}(O, S, \mathbf{i}, \mathbf{q}, \mathbf{k}) = \delta(S, S') \tag{8.17}$$

where $\delta(a, b)$ is the Kronecker delta function, which equals 1 if both elements are equal, and 0 otherwise. The feature functions for the remaining parameters are detailed in Section 8.2.3.

Finally, equation (8.15) is expressed as

$$p_{\boldsymbol{\lambda}}(O, S, \mathbf{i}, \mathbf{q}, \mathbf{k}) = \exp\Big(\sum_{m \in \mathcal{M}} \lambda_m f_m(O, S, \mathbf{i}, \mathbf{q}, \mathbf{k})\Big) = \exp(\boldsymbol{\lambda}' \mathbf{f}(O, S, \mathbf{i}, \mathbf{q}, \mathbf{k}))\,, \tag{8.18}$$

where we can substitute the sum by its vectorial notation, where by $\boldsymbol{\lambda}'$ we denote the transpose of $\boldsymbol{\lambda}$. The model in (8.18) when plugged into (8.1) is a log-linear model with binary inputs.

8.2.2 Discriminative Classifier

Log-linear models are commonly employed to approximate posterior probabilities. From (8.18), we can approximate the posterior class probability required by the optimal Bayes' classifier as follows

$$p_{\boldsymbol{\lambda}}(S \mid O) = \sum_{\mathbf{i},\mathbf{q},\mathbf{k}} p_{\boldsymbol{\lambda}}(S, \mathbf{i}, \mathbf{q}, \mathbf{k} \mid O)\,, \tag{8.19}$$

where $p_{\boldsymbol{\lambda}}(S, \mathbf{i}, \mathbf{q}, \mathbf{k} \mid O)$ is approximated by (8.18) and the Bayes' theorem as

$$p_{\boldsymbol{\lambda}}(S, \mathbf{i}, \mathbf{q}, \mathbf{k} \mid O) = \frac{p_{\boldsymbol{\lambda}}(O, S, \mathbf{i}, \mathbf{q}, \mathbf{k})}{p_{\boldsymbol{\lambda}}(O)} = \frac{\exp(\boldsymbol{\lambda}' \mathbf{f}(O, S, \mathbf{i}, \mathbf{q}, \mathbf{k}))}{p_{\boldsymbol{\lambda}}(O)}\,. \tag{8.20}$$

It is worth noting that the denominator is a probability because of the transformation that we have performed in equations (8.11)-(8.14). However, we wish to select any arbitrary parametric vector, $\boldsymbol{\lambda}$, and in such a case, the denominator also becomes arbitrary, yielding the *LLHMM model*

$$p_{\boldsymbol{\lambda}}(S, \mathbf{i}, \mathbf{q}, \mathbf{k} \mid O) = \frac{\exp(\boldsymbol{\lambda}' \mathbf{f}(O, S, \mathbf{i}, \mathbf{q}, \mathbf{k}))}{\mathcal{Z}_{\boldsymbol{\lambda}}(O)}\,, \tag{8.21}$$

where $\mathcal{Z}_{\boldsymbol{\lambda}}(O)$ is a normalization constant defined as

$$\mathcal{Z}_{\boldsymbol{\lambda}}(O) = \sum_{S} \sum_{\mathbf{i},\mathbf{q},\mathbf{k}} \exp(\boldsymbol{\lambda}' \mathbf{f}(O, S, \mathbf{i}, \mathbf{q}, \mathbf{k}))\,, \tag{8.22}$$

that for the specific parameters in equations (8.11)-(8.14) corresponds to the marginal probability $p_{\boldsymbol{\lambda}}(O)$. The log-linear model in (8.21) is a log-linear model with hidden variables for the segmentation and for the states which have a first order dependence. This model is a variation of a semi-Markov conditional random field.

The formerly defined LLHMM is used in the optimal Bayes' rule to obtain the *LLHMM classifier*

$$S^{\star} = \underset{S}{\operatorname{argmax}}\; p_{\boldsymbol{\lambda}}(S \mid O)\,, \tag{8.23}$$

8.2.3 Feature Functions

As discussed before, in order to use the standard notation in log-linear models, we need to define the feature functions for each family of parameters.

For a given character c out of C different symbols, and for a given pair of state indexes (q, q') of that character, we define the *transition features* $f_{cqq'}(O, S, \mathbf{i}, \mathbf{q}, \mathbf{k}) = f_{cqq'}(S, \mathbf{i}, \mathbf{q})$ as follows

$$f_{cqq'}(S, \mathbf{i}, \mathbf{q}) = \sum_{l=1}^{L} \delta(s_l, c) \begin{cases} \sum_{t=i_l}^{i_{l+1}-2} \delta(q_t, q)\delta(q_{t+1}, q') & 1 \le q, q' \le M_c \\ \delta(q_{i_l}, q') & q = I, 1 \le q' \le M_c \\ \delta(q_{i_{l+1}-1}, q) & 1 \le q \le M_c, q' = F \\ 0 & \text{otherwise} \end{cases} \tag{8.24}$$

where I and F represent the *initial* and *final* states respectively, and M_c the number of states for character c. Intuitively, this feature counts the number of times the specific transition from q to q' of the character c, is used in the input $S, \mathbf{i}, \mathbf{q}$. Note that it can be 0, if, for instance, the character c is not part of word S.

For the mixture components, we define the *component features* for each character c, state q and component k as follows

$$f_{cqk}(O, S, \mathbf{i}, \mathbf{q}, \mathbf{k}) = f_{cqk}(S, \mathbf{i}, \mathbf{q}, \mathbf{k}) = \sum_{l=1}^{L} \delta(s_l, c) \sum_{t=i_l}^{i_{l+1}-1} \delta(q_t, q)\delta(k_t, k)\,. \tag{8.25}$$

Intuitively, this feature counts the number of times an emission of O is generated by the k-th component of the state q of the character c.

The final set of features are the *emission features*, which are given as follows for each character c, state q, component k and dimension d

$$f_{cqkd}(O, S, \mathbf{i}, \mathbf{q}, \mathbf{k}) = \sum_{l=1}^{L} \delta(s_l, c) \sum_{t=i_l}^{i_{l+1}-1} \delta(q_t, q)\delta(k_t, k) o_{td}\,. \tag{8.26}$$

8.3 Equivalence Between BHMMs and LLHMMs

In this section we prove that the BHMM classifier for isolated words is equivalent to the LLHMM proposed in Section 8.2. A generative classifier is said to be equivalent to a discriminative classifier if for a given set of generative parameters $\boldsymbol{\theta}$, a set of discriminative parameters $\boldsymbol{\lambda}$ can be found such that

$$\operatorname*{argmax}_{S \in W} p_{\boldsymbol{\theta}}(O, S) = \operatorname*{argmax}_{S \in W} p_{\boldsymbol{\lambda}}(S \mid O)\,; \tag{8.27}$$

and vice-versa. Note that the previous equivalence holds even when any of both probabilities is scaled by a factor that does not depends on S, and consequently, the normalization constant of the LLHMM, $\mathcal{Z}_{\boldsymbol{\lambda}}(O)$, defined in (8.22), can be removed from the right-hand side (8.27) without changing the equivalence. Note also, that in contrast to discriminative parameters, generative parameters must fulfill a set of constraints

$$\sum_{w} \pi_w = 1, \qquad \forall_{c,q} : \sum_{q'} a_{cqq'} = 1, \qquad \forall_{c,q} : \sum_{k} \tau_{cqk} = 1\,. \tag{8.28}$$

The proof of the equation is done in two steps by proving two implications: left to right, and right to left.

8.3.1 From Generative to Discriminative Parameters

Unlike the converse direction, it is relatively simple to prove that given a BHMM classifier for isolated word recognition, it can be reparameterized into a LLHMM. Recall that by definition of the LLHMM, if we set the log-linear parameters, $\boldsymbol{\lambda}$, using the generative parameters, $\boldsymbol{\theta}$, as defined in (8.11)-(8.14), then we have that

$$p_{\boldsymbol{\theta}}(O, S) = \sum_{\mathbf{i},\mathbf{q},\mathbf{k}} \exp(\boldsymbol{\lambda}'\mathbf{f}(O, S, \mathbf{i}, \mathbf{q}, \mathbf{k})) = \mathcal{Z}_{\boldsymbol{\lambda}}(O) p_{\boldsymbol{\lambda}}(S \mid O)\,. \tag{8.29}$$

Therefore, these two models when inserted into their corresponding classifiers in (8.27) produce proportional scores and, hence, select the same word or class.

8.3.2 From Discriminative to Generative Parameters

In this subsection, we prove the converse statement: that given a LLHMM classifier for isolated word recognition as defined in Section 8.2, an equivalent BHMM classifier exists. We begin expressing the right-hand model in (8.27) as follows

$$p_{\boldsymbol{\lambda}}(S \mid O) = \sum_{\mathbf{i},\mathbf{q},\mathbf{k}} \frac{h_{\boldsymbol{\lambda}}(O, S, \mathbf{i}, \mathbf{q}, \mathbf{k})}{\mathcal{Z}_{\boldsymbol{\lambda}}(O)}\,, \tag{8.30}$$

where $h_{\boldsymbol{\lambda}}(O, S, \mathbf{i}, \mathbf{q}, \mathbf{k}) = \exp(\boldsymbol{\lambda}'\mathbf{f}(O, S, \mathbf{i}, \mathbf{q}, \mathbf{k}))$.

We start instantiating the feature $h_{\boldsymbol{\lambda}}(\cdots)$ in previous equation

$$h_{\boldsymbol{\lambda}}(O, S, \mathbf{i}, \mathbf{q}, \mathbf{k}) = \exp(\lambda_S) \cdot h_{\boldsymbol{\lambda}}(\mathbf{i}, \mathbf{q}; S) \cdot h_{\boldsymbol{\lambda}}(O, \mathbf{k}; \mathbf{i}, \mathbf{q}, S) \tag{8.31}$$

with

$$h_{\boldsymbol{\lambda}}(\mathbf{i},\mathbf{q};S)=\exp\Big(\sum_{l,t}\lambda_{s_l q_t q_{t+1}}\Big)\,, \tag{8.32}$$

and

$$h_{\boldsymbol{\lambda}}(O,\mathbf{k};\mathbf{i},\mathbf{q},S)=\exp\Big(\sum_{l,t}\lambda_{s_l q_t k_t}+\sum_{l,t,d}o_{td}\lambda_{s_l q_t k_t d}\Big)\,. \tag{8.33}$$

If we compare (8.2) expanded accordingly to (8.4) and (8.5), with (8.31) expanded with (8.32); then it is observed that each of the 3 terms in the right-hand side in (8.31) can be transformed, independently, into the corresponding term in (8.2).

Firstly, we transform $h_{\boldsymbol{\lambda}}(O,\mathbf{k};\mathbf{i},\mathbf{q},S)$ into $p_{\boldsymbol{\theta}}(O,\mathbf{k}\mid\mathbf{i},\mathbf{q},S)$. Therefore, we need to transform the part of the discriminative parameters $\{\lambda_{cqk}\}$ and $\{\lambda_{cqkd}\}$ into the generative parameters $\{\tau;\mathbf{p}\}$, where τ is constrained as shown in (8.28). For doing that, (8.31) is multiplied and divided by $\exp(\sum_{l,t}\zeta_{s_l q_t})$, and then, we rearrange the multiplication into (8.33) as follows

$$\exp(\sum_{l,t}\zeta_{s_l q_t})h_{\boldsymbol{\lambda}}(O,\mathbf{k};\mathbf{i},\mathbf{q},S)=\exp\Big(\sum_{l,t}(\lambda_{s_l q_t k_t}+\zeta_{s_l q_t})+\sum_{l,t,d}o_{td}\lambda_{s_l q_t k_t d}\Big)\,, \tag{8.34}$$

whereas the division is moved into the second term in the right-hand side of (8.31), yielding

$$\exp(-\sum_{l,t}\zeta_{s_l q_t})h_{\boldsymbol{\lambda}}(\mathbf{i},\mathbf{q};S)=\exp\Big(\sum_{l,t}\bar{\lambda}_{s_l q_t q_{t+1}}\Big)\,, \tag{8.35}$$

where $\bar{\lambda}_{sqq'}=\lambda_{sqq'}-\zeta_{sq}$ will be used afterwards. The unknown parameters $\{\zeta_{sq}\}$ are introduced to force the generative parameters $\{\tau_{cq}(k)\}$ to sum 1 in the transformation.

From (8.9) and (8.34), and taking into account the constraints in (8.28), the solution must fulfill the following 3 constraints

$$\lambda_{cqkd}=\log\frac{p_{cqkd}}{1-p_{cqkd}}\,, \tag{8.36}$$

$$\lambda_{cqk}+\zeta_{cq}=\log\tau_{cq}(k)+\xi_{cq}(k)\,, \tag{8.37}$$

$$\sum_{k=1}^{K_{cq}}\tau_{cq}(k)=1\,. \tag{8.38}$$

Then, from (8.36) we work out the value of p_{cqkd}

$$p_{cqkd}=\frac{\exp(\lambda_{cqkd})}{1+\exp(\lambda_{cqkd})}\,, \tag{8.39}$$

and from (8.37) the value of $\tau_{cq}(k)$ is expressed as

$$\tau_{cq}(k)=\exp(\lambda_{cqk}-\xi_{cq}(k))\exp(\zeta_{cq})\,, \tag{8.40}$$

where $\xi_{cq}(k)$ is defined as in (8.10) using the values of $\{p_{cqkd}\}$ defined in (8.39). Although $\exp(\zeta_{cq})$ is still unknown, recall that it was introduced to tackle the normalization constraint in (8.38), and then its value is worked out by replacing (8.40) in (8.38)

$$\exp(\zeta_{cq})=\frac{1}{\sum_{k'=1}^{K_{cq}}\exp(\lambda_{cqk'}-\xi_{cq}(k'))}\,. \tag{8.41}$$

Finally, the exact value of $\tau_{cq}(k)$ is obtained by plugging (8.41) into (8.40)

$$\tau_{cq}(k) = \frac{\exp\left(\lambda_{cqk} - \xi_{cq}(k)\right)}{\sum_{k'=1}^{K_{cq}} \exp\left(\lambda_{cqk'} - \xi_{cq}(k')\right)} \,. \tag{8.42}$$

Now we focus on transforming $\exp(-\sum_{l,t} \zeta_{s_l q_t}) h_{\boldsymbol{\lambda}}(\mathbf{i}, \mathbf{q}; S)$ from (8.35) into $p_{\boldsymbol{\theta}}(\mathbf{i}, \mathbf{q} \mid S)$ as defined in (8.4). This part of the proof is similar in conception to the proof given in Heigold et al. [2008c]. Firstly we define a global transition matrix $\mathcal{Q}$ as follows

$$\mathcal{Q}_{ij} = \begin{cases} \exp\left(\bar{\lambda}_{cqq'}\right) & i = f(c,q) \text{ and } j = f(c,q') \\ 1 & i = f(c,F) \\ 0 & \text{otherwise} \end{cases} \,, \tag{8.43}$$

where $f : \mathbb{N}^2 \mapsto \mathbb{N}$ is an injective function that maps each pair composed by a character and state, into a global index or state

$$f(c,q) = \begin{cases} B_c & q = I \\ B_c + q & 1 \leq q \leq M_c \\ B_c + M_c + 1 & q = F \end{cases} \,, \tag{8.44}$$

with M_c being the number of states for the symbol c, and $B_c = 1 + \sum_{n=1}^{c-1}(2 + M_n)$ being the number of preceding states to the first state of symbol c plus 1. Since all the values of $\mathcal{Q}$ are not negative, accordingly to *Perron-Frobenius theorem* [Rao and Rao, 1998, p.473], the largest eigenvalue of $\mathcal{Q}$, ψ, is positive and unique. Furthermore, the eigenvector associated to the largest eigenvalue, $\mathbf{v}$, has only positive coefficients, and obviously because of the eigenvector definition, $\mathbf{v}$ satisfies

$$\sum_j \mathcal{Q}_{ij} v_j = \psi v_i, \quad \forall i = 1, \ldots \tag{8.45}$$

Now, the transition generative parameters are defined as

$$a_c(q, q') = \frac{\mathcal{Q}_{f(c,q)f(c,q')} v_{f(c,q')}}{\psi v_{f(c,q)}} = \frac{\exp\left(\bar{\lambda}_{cqq'}\right) v_{f(c,q')}}{\psi v_{f(c,q)}} \,, \tag{8.46}$$

where $a_c(q, q')$ verifies the normalization constraint (8.28) because of (8.45). These parameters yield a probability proportional to that of (8.35) when used in (8.4) as the generative parameters of $p_{\boldsymbol{\theta}}(\mathbf{i}, \mathbf{q} \mid S)$ (see A.1),

$$p_{\boldsymbol{\theta}}(\mathbf{i}, \mathbf{q} \mid S) = \frac{1}{\psi^{T+L}} \Big[\prod_l \frac{v_{f(s_l,F)}}{v_{f(s_l,I)}} \Big] \frac{h_{\boldsymbol{\lambda}}(\mathbf{i}, \mathbf{q}; S)}{\exp(\sum_{l,t} \zeta_{s_l q_t})} \,, \tag{8.47}$$

which is the equivalence we need but for the term $\frac{1}{\psi^{T+L}} \prod_l \frac{v_{f(s_l,F)}}{v_{f(s_l,I)}}$.

We can introduce this constant factor by multiplying and dividing (8.31) by it. The division is used in this part whereas the multiplication is added to the first term as follows

$$h_{\boldsymbol{\lambda}}(O,S,\mathbf{i},\mathbf{q},\mathbf{k}) = \\ \psi^T \cdot \exp(\bar{\lambda}_S) \cdot \frac{h_{\boldsymbol{\lambda}}(\mathbf{i},\mathbf{q};S)}{\psi^{T+L}\exp(\sum_{l,t}\zeta_{s_l q_t})}\Big[\prod_l \frac{v_{f(s_l,F)}}{v_{f(s_l,I)}}\Big] \cdot \exp(\sum_{l,t}\zeta_{s_l q_t}) h_{\boldsymbol{\lambda}}(O,\mathbf{k};\mathbf{i},\mathbf{q},S) \tag{8.48}$$

with $\bar{\lambda}_S = \lambda_S + L\log\psi - \sum_{l=1}^{L}\log\frac{v_{f(s_l,F)}}{v_{f(s_l,I)}}$.

Finally, the last part of the proof consists in the transformation of $\exp(\bar{\lambda}_S)$ into the word prior probabilities π_S. Similarly to the case of mixture coefficients, we multiply and divide the numerator of (8.31) by an unknown constant, $\exp(\zeta)$. Since the constant $\exp(\zeta)$ is independent of the word S, it can be introduced into the right-hand side of (8.27). This constant is grouped together with $\exp(\bar{\lambda})$ as follows

$$\exp(\bar{\lambda}_S + \zeta) \tag{8.49}$$

Thus, taking into account (8.49) and the constraints (8.28), we have that following equalities must hold

$$\bar{\lambda}_S + \zeta = \log\pi_S \tag{8.50}$$

$$\sum_{S\in W}\pi_S = 1 \tag{8.51}$$

and the solution is found by following a similar procedure to that of the mixture coefficients

$$\pi_S = \frac{\exp(\bar{\lambda}_S)}{\sum_{S'\in W}\exp(\bar{\lambda}_{S'})}\,. \tag{8.52}$$

In summary, we have proven that for a given set of discriminative parameters $\boldsymbol{\lambda}$, a set of generative parameters can be defined, $\boldsymbol{\theta}$, by (8.39), (8.42), (8.46), and (8.52); such that

$$\begin{aligned}
\operatorname*{argmax}_S p_{\boldsymbol{\lambda}}(S\mid O) &= \operatorname*{argmax}_S \frac{\mathcal{Z}_{\boldsymbol{\lambda}}(O)}{\exp(\zeta)}\exp(\zeta)p_{\boldsymbol{\lambda}}(S\mid O) \\
&= \operatorname*{argmax}_S \exp(\zeta)\mathcal{Z}_{\boldsymbol{\lambda}}(O)p_{\boldsymbol{\lambda}}(S\mid O) \\
&= \operatorname*{argmax}_S \sum_{\mathbf{q},\mathbf{i},\mathbf{k}}\exp(\zeta)h_{\boldsymbol{\lambda}}(O,S,\mathbf{i},\mathbf{q},\mathbf{k}) \\
&= \operatorname*{argmax}_S \sum_{\mathbf{q},\mathbf{i},\mathbf{k}}\psi^T\exp(\bar{\lambda}_S+\zeta)\cdot\frac{h_{\boldsymbol{\lambda}}(\mathbf{i},\mathbf{q};S)}{\psi^{T+L}\exp(\sum_{l,t}\zeta_{s_l q_t})}\Big[\prod_{l=1}^{L}\frac{v_{f(s_l,F)}}{v_{f(s_l,I)}}\Big] \\
&\qquad\qquad \cdot\exp(\sum_{l,t}\zeta_{s_l q_t})h_{\boldsymbol{\lambda}}(O,\mathbf{k};\mathbf{i},\mathbf{q},S) \\
\Rightarrow \operatorname*{argmax}_S &\sum_{\mathbf{q},\mathbf{i},\mathbf{k}}\psi^T\cdot\pi_S\cdot p_{\boldsymbol{\theta}}(\mathbf{i},\mathbf{q}\mid S)\cdot p_{\boldsymbol{\theta}}(O,\mathbf{k}\mid\mathbf{i},\mathbf{q},S) \\
&= \operatorname*{argmax}_S \psi^T\sum_{\mathbf{q},\mathbf{i},\mathbf{k}}p_{\boldsymbol{\theta}}(O,S,\mathbf{i},\mathbf{q},\mathbf{k}) = \operatorname*{argmax}_S p_{\boldsymbol{\theta}}(O,S)\,.
\end{aligned} \tag{8.53}$$

where by $\Rightarrow$ we highlight the step of the proof that is not symmetric.

8.4 LLHMM Parameter Estimation

The most well-known criteria for discriminative parameter estimation is the *maximum mutual information (MMI)*. Given a collection of samples $\{(O_1, S_1), \ldots, (O_N, S_N)\}$, the MMI criterion is defined as follows

$$F_{\text{MMI}}(\boldsymbol{\lambda}) = \sum_{n=1}^{N} \log\left(p_{\lambda}(S_n \mid O_n)\right). \tag{8.54}$$

The optimal discriminative parameters, $\boldsymbol{\lambda}^*$, are those that maximize F_{MMI}.

There are several algorithms for obtaining the parameters that maximize (8.54) [Heigold et al., 2008a], but commonly the *Resilient back-propagation (RPROP) algorithm* [Riedmiller and Braun, 1993] is used. The RPROP requires the computation of the gradient sign, for the parameter λ_m, which is associated with the feature $f_m(\cdot)$, the punctual derivative of F_{MMI} with respect to λ_m is given by

$$\frac{\partial F_{\text{MMI}}(\boldsymbol{\lambda})}{\partial \lambda_m} = N_m(\boldsymbol{\lambda}) - Q_m(\boldsymbol{\lambda}) \tag{8.55}$$

where $N_m(\boldsymbol{\lambda})$ and $Q_m(\boldsymbol{\lambda})$ are expected counts defined as follows

$$N_m(\boldsymbol{\lambda}) = \sum_{n=1}^{N} N_{n\,m}(\boldsymbol{\lambda}), \qquad Q_m(\boldsymbol{\lambda}) = \sum_{n=1}^{N} Q_{n\,m}(\boldsymbol{\lambda}), \tag{8.56}$$

with $N_{n\,m}(\boldsymbol{\lambda})$ and $Q_{n\,m}(\boldsymbol{\lambda})$ being the expected latent and class counts respectively. These counts are defined as follows

$$N_{n\,m}(\boldsymbol{\lambda}) = \sum_{\mathbf{i},\mathbf{q},\mathbf{k}} p_{\boldsymbol{\lambda}}(\mathbf{i},\mathbf{q},\mathbf{k} \mid O_n, S_n) f_m(O_n, S_n, \mathbf{i},\mathbf{q},\mathbf{k}), \tag{8.57}$$

and

$$Q_{n\,m}(\boldsymbol{\lambda}) = \sum_{\mathbf{i},\mathbf{q},\mathbf{k}} \sum_{S} p_{\boldsymbol{\lambda}}(S,\mathbf{i},\mathbf{q},\mathbf{k} \mid O_n) f_m(O_n, S, \mathbf{i},\mathbf{q},\mathbf{k}). \tag{8.58}$$

The probabilities $p_{\boldsymbol{\lambda}}(\mathbf{i},\mathbf{q},\mathbf{k} \mid O, S)$ and $p_{\boldsymbol{\lambda}}(S,\mathbf{i},\mathbf{q},\mathbf{k} \mid O)$ are computed as follows

$$p_{\boldsymbol{\lambda}}(\mathbf{i},\mathbf{q},\mathbf{k} \mid O, S) = \frac{\exp(\boldsymbol{\lambda}'\mathbf{f}(O,S,\mathbf{i},\mathbf{q},\mathbf{k}))}{\mathcal{Z}_{\boldsymbol{\lambda}}(O,S)}, \tag{8.59}$$

and

$$p_{\boldsymbol{\lambda}}(S,\mathbf{i},\mathbf{q},\mathbf{k} \mid O) = p_{\boldsymbol{\lambda}}(S \mid O) p_{\boldsymbol{\lambda}}(\mathbf{i},\mathbf{q},\mathbf{k} \mid O, S) = \frac{\exp(\boldsymbol{\lambda}'\mathbf{f}(O,S,\mathbf{i},\mathbf{q},\mathbf{k}))}{\mathcal{Z}_{\boldsymbol{\lambda}}(O)}. \tag{8.60}$$

Finally, $\mathcal{Z}_{\boldsymbol{\lambda}}(O)$ is the normalization constant for the model defined in (8.22) whereas $\mathcal{Z}_{\boldsymbol{\lambda}}(O,S)$ is a joint normalization constant for the output and the word, which is likewise defined as

$$\mathcal{Z}_{\boldsymbol{\lambda}}(O,S) = \sum_{\mathbf{i},\mathbf{q},\mathbf{k}} \exp(\boldsymbol{\lambda}'\mathbf{f}(O,S,\mathbf{i},\mathbf{q},\mathbf{k}))\,. \tag{8.61}$$

Note that $p_{\boldsymbol{\lambda}}(S \mid O)$ defined in (8.19) can be expressed as

$$p_{\boldsymbol{\lambda}}(S \mid O) = \frac{\mathcal{Z}_{\boldsymbol{\lambda}}(O,S)}{\mathcal{Z}_{\boldsymbol{\lambda}}(O)}\,. \tag{8.62}$$

The RPROP algorithm computes the sign of the gradient with the aid of these expected counts, and then, modifies the current parameters $\boldsymbol{\lambda}^{(k)}$ accordingly, so that a new estimate of the parameters is obtained, $\boldsymbol{\lambda}^{(k+1)}$. The algorithm starts with a rough estimate of the parameters, $\boldsymbol{\lambda}^{(0)}$, and it ends when either a maximum number of iterations have been reached, or the value of the objective function surpass a given threshold.

8.4.1 γ-MMI Criterion

A modification of the MMI criterion (8.54), the so-called γ-*MMI* criterion, leads to better performance [Povey, 2003, Schluter and Macherey, 1998]. The γ-MMI is defined by introducing a scaling factor γ into the MMI criterion as follows

$$F_{\gamma\text{-MMI}}(\boldsymbol{\lambda}) = \frac{1}{\gamma}\sum_{n=1}^{N} \log(p_{\boldsymbol{\lambda}\,\gamma}(S_n \mid O_n))\,, \tag{8.63}$$

with $p_{\boldsymbol{\lambda}\,\gamma}(S \mid O)$ defined as follows

$$p_{\boldsymbol{\lambda}\,\gamma}(S \mid O) = \frac{\left[\mathcal{Z}_{\boldsymbol{\lambda}}(O,S)\right]^{\gamma}}{\sum_R \left[\mathcal{Z}_{\boldsymbol{\lambda}}(O,R)\right]^{\gamma}}\,. \tag{8.64}$$

The basic idea is to scale the scores for each word in order to make the best words to compete one against the others even if the differences in probability are already large.

The gradient for the γ-MMI criterion in (8.63) is analogous to (8.55) but instead of using $Q_{n\,m}(\boldsymbol{\lambda})$, we now use $Q^{\gamma}_{n\,m}(\boldsymbol{\lambda})$ which is defined as follows

$$Q^{\gamma}_{n\,m}(\boldsymbol{\lambda}) = \sum_{\mathbf{i},\mathbf{q},\mathbf{k}} \sum_{S} p_{\boldsymbol{\lambda}\,\gamma}(S,\mathbf{i},\mathbf{q},\mathbf{k} \mid O_n) f_m(O_n,S,\mathbf{i},\mathbf{q},\mathbf{k})\,, \tag{8.65}$$

with the probability $p_{\boldsymbol{\lambda}\,\gamma}(S,\mathbf{i},\mathbf{q},\mathbf{k} \mid O)$ defined as

$$p_{\boldsymbol{\lambda}\,\gamma}(S,\mathbf{i},\mathbf{q},\mathbf{k} \mid O) = p_{\boldsymbol{\lambda}\,\gamma}(S \mid O) p_{\boldsymbol{\lambda}}(\mathbf{i},\mathbf{q},\mathbf{k} \mid S,O)\,, \tag{8.66}$$

where the probabilities $p_{\boldsymbol{\lambda}\,\gamma}(S \mid O)$ and $p_{\boldsymbol{\lambda}}(\mathbf{i},\mathbf{q},\mathbf{k} \mid S,O)$ are defined in (8.64) and (8.59), respectively. It is worth noting, that the calculation of $Q^{\gamma}_{n\,m}(\boldsymbol{\lambda})$ and $N^{\gamma}_{n\,m}(\boldsymbol{\lambda})$ requires a lot of computations. In appendix A.2 an example of how these values can be efficiently calculated using the Forward-Backward algorithm is shown.

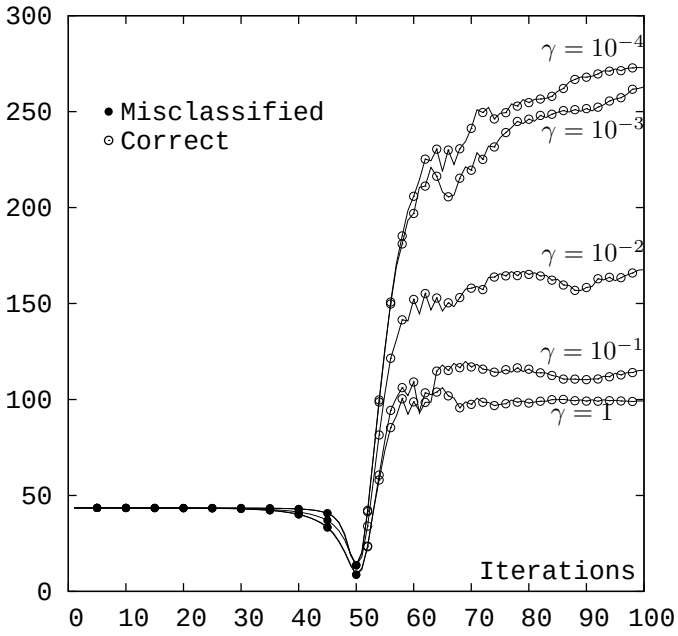


Figure 8.1: Differences (in logarithmic scale) between the most probable and the second most probable word for a given training sample (*cette*). Several values of the γ-MMI criterion are plotted. The most probable word changes at iteration 50 becoming the correct word

Fig. 8.1, summarizes the main idea behind the γ-MMI training criterion. It depicts the differences between the most probable word an the second most probable competitor for a LLHMM model (more details in Section 8.5). It is observed that these differences are of 44 points (in logarithmic scale) at the beginning, which corresponds to MLE. Additionally, the training sample is incorrectly classified at the first training iterations. Although, after 50 iterations all the γ values correctly classify the sample; smaller values of γ induce a larger difference between the correct class and its competitors.

8.4.2 The Power Approximation

It is well-known, that the MMI criterion is very sensitive to outliers [Heigold et al., 2008b, Povey, 2003]. In order to avoid this problem, we can approximate the logarithm by the power approximation and transform both MMI and γ-MMI by applying $\log u = \lim_{k\to 0} \frac{u^k-1}{k}$. For instance, in the case of the MMI, the power approximation yields

$$F_{\text{POW}_k}(\boldsymbol{\lambda}) = \sum_{n=1}^{N} \frac{p_{\boldsymbol{\lambda}}(S_n \mid O_n)^k - 1}{k}\,, \tag{8.67}$$

where k is a meta-parameter that calibrates the resemblance with the MMI criterion [Heigold, 2010]. Values near to zero produce a similar criterion to the MMI, whereas larger values yield criteria less sensible to outliers.

As in the MMI case the POW criterion can be modified by adding the γ meta-parameter providing the γ-POW criterion defined

$$F_{\gamma\text{-POW}_k}(\boldsymbol{\lambda}) = \frac{1}{\gamma} \sum_{n=1}^{N} \frac{[p_{\boldsymbol{\lambda}\,\gamma}(S \mid O)]^k - 1}{k}\,, \tag{8.68}$$

with $p_{\boldsymbol{\lambda}\,\gamma}(S \mid O)$ defined as in (8.64).

The RPROP algorithm is yet applicable to this optimization problem, by using the gradient which is computed likewise to (8.55) but with the following counts instead the standard ones

$$N_m(\boldsymbol{\lambda}) = \sum_{n=1}^{N} [p_{\boldsymbol{\lambda}\,\gamma}(S_n \mid O_n)]^k N_{n\,m}(\boldsymbol{\lambda})\,, \tag{8.69}$$

and

$$Q^{\gamma}_{m}(\boldsymbol{\lambda}) = \sum_{n=1}^{N} [p_{\boldsymbol{\lambda}\,\gamma}(S_n \mid O_n)]^k Q^{\gamma}_{n\,m}(\boldsymbol{\lambda}), \tag{8.70}$$

where $N_{n\,m}(\boldsymbol{\lambda})$ and $Q^{\gamma}_{n\,m}(\boldsymbol{\lambda})$ are defined in (8.57) and (8.65), respectively.

8.4.3 Error Rate Criterion

An interesting case is F_{POW_1} which produces the expected classification error rate as a criterion (plus a constant value). Specifically

$$F_{\text{POW}_1}(\boldsymbol{\lambda}) + N = \sum_{n=1}^{N} p_{\boldsymbol{\lambda}}(S_n \mid O_n) = \sum_{n=1}^{N} \sum_{S} p_{\boldsymbol{\lambda}}(S \mid O_n)\delta(S, S_n)\,, \tag{8.71}$$

with $\delta(S, R)$ being the Kronecker's delta function.

8.4.4 Regularization

A common undesired property of all the proposed discriminative criteria is that they easily overfit the parameters. Even criteria specially designed to avoid outlayers such as the power criterion suffer from overfitting. Since there is no clear way to smooth discriminatively trained models, a typical amendment is to add a regularization term to the criterion itself

$$F_{C*}(\boldsymbol{\lambda}) = F_{*}(\boldsymbol{\lambda}) - \frac{C}{2} \sum_{m} (\lambda^{(0)}_{m} - \lambda_m)^2\,, \tag{8.72}$$

with $F_{*}(\boldsymbol{\lambda})$ denoting the original criterion, such as F_{MMI} or $F_{\gamma\text{-POW}}$; and $\boldsymbol{\lambda}^{(0)}$ being either a reliable estimation of the parameters or simply $\mathbf{0}$.

The inclusion of the regularization term, only modifies the gradient in the following form

$$\frac{\partial F_{C*}(\boldsymbol{\lambda})}{\partial \lambda_m} = \frac{\partial F_{*}(\boldsymbol{\lambda})}{\partial \lambda_m} + C(\lambda_m^{(0)} - \lambda_m) = \big(N_m(\boldsymbol{\lambda}) - Q_m(\boldsymbol{\lambda})\big) + C(\lambda_m^{(0)} - \lambda_m)\,, \quad (8.73)$$

where the expected counts, N_m and Q_m, are calculated as in the original criterion. For instance, for γ-POW$_k$ we use equations (8.69) and (8.70) respectively.

8.5 Experiments

In this section, we perform several experiments on the RIMES database of handwritten French letters (see Section 3.4), so that the performance of several discriminative training criteria for BHMM is assessed with respect to the generative training. We achieve this goal by transforming a BHMM into a LLHMM and afterwards discriminatively training a LLHMM. Furthermore, we visually inspect several discriminative parameters by transforming them into their generative counterpart.

8.5.1 Database and Experimental Setup

Experiments were carried out over the protocol WR2 used in the handwritten word recognition competition of the ICDAR 2009, which is described in Section 3.4.A three step preprocess was applied to all input images: gray level normalization, deslanting, and vertical size normalization. Then, preprocessed images were first scaled in height to 30 pixels maintaining the aspect ratio, and then binarized with the Otsu's method. Finally, a sliding window of width 9 with vertical repositioning was applied, and as a result, sequences of 270-dimensional binary feature vectors were obtained.

In order to properly initialize the MMI training scheme the LLHMM was initialized with a BHMM classifier trained with the EM (Baum-Welch) algorithm [Rabiner and Juang, 1993], using the same training scheme than the used in the experiments of Chapter 5. The best generative BHMM, which is composed by $Q = 8$ states per character and $K = 64$ mixture components per state, obtains an error of 21.2%. Note, that in contrast to what we did in Chapter 5, we have carried out all experiments without applying any language model.

Regarding the discriminative training, the RPROP algorithm was used for optimizing the criteria. The initial discriminative parameters were obtained transforming the generative parameters of a BHMMs with $Q = 8$ states per character and $K = 26$ mixture components per state. Despite the best generative results is obtained with $K = 64$ mixture components per state, in the Chapter 7 we reported that the best classifier obtained using MMI training has half (0.4 ratio) the number of mixture component per state than its generative counterpart. Consequently, in preliminary experiments we checked that the results obtained using the conventional MMI criterion with $K = 26$ are similar or better to those obtained increasing the value of K.

Finally, all the proposed discriminative training criteria require to compute sums over all the words for calculating several values such as $\mathcal{Z}_{\boldsymbol{\lambda}}(O)$ in (8.60). Consequently discriminative training algorithms become unfeasible in a straight implementation. For this reason we have approximated the sums over all the words by a beam pruning strategy together with a histogram pruning up to 100 best hypothesis accordingly to $p(S \mid O)$.

8.5.2 Experiments

Firstly, we wanted to assess the repercussions of the regularization term in the conventional MMI criterion. For doing so, we scanned several values of the regularization parameter $C = \{0, 0.1, 1, 10, 100\}$ as introduced in (8.72), where $C = 0$ stands for not using the regularization at all. In Fig. 8.2, the classification error rate (CER) as a function of the number of RPROP iterations is plotted for different regularization values. In all cases, even with standard MMI, the CER decreases in a similar way, until iteration 60, where the best result is obtained. At this point, the behavior diverges depending on the precise value of C. If no regularization is applied ($C = 0$) the error becomes unstable and increases (over-fits) as the training iterates. However, the larger the regularization parameter is, the less overtrained the model becomes, until that for $C = 10$ the error becomes stable while providing similar performance to that at iteration 60. Note that if the regularization parameter is further increased, an slight drop in performance is observed. As expected, the regularization term reduces the overfitting problem.

Fig. 8.2 also shows that the regularized MMI criterion obtains a CER around 19.5% using only $K = 26$ components per state. If we compare it with the best generative result, which is 21.2% and is obtained using $K = 64$, we observe not only an improvement of 1.7 absolute points but also a reduction on the number of parameters of 40%, i.e. less than half the parameters are needed for such improvement.

For a deeper understanding of the MMI criterion, in Fig. 8.3 we depict the top 5 most probable words as a function of the training iterations (0,50,55,60,100) and the γ value of the γ-MMI criterion (1-MMI=MMI). We selected a common example in which the MLE misclassifies the sample and the MMI learns how to discriminate it among the other competitors. More precisely, for several training iterations the 5 most probable transcriptions are shown. In addition, for each transcription the difference (in logarithmic scale) between its score and the best score at that iteration is also shown. As expected, the correct transcription (*cette*) gains relevance with the iterations, that is, the training algorithm is modifying the model parameters in order to better classify the sample. In particular, at the beginning there is a difference of 44 points between *cette* and the best transcription (*celle*). However, at some point near to iteration 50 this situation is reverted, and from this point on the score difference keeps increasing (see Fig. 8.1). A total of 60.3% of the training samples that are misclassified by the MLE, are correctly classified at the end of the last MMI iteration. In contrast, only 1.3% of the correctly classified samples by the MLE are misclassified at the end of the training process.

In Fig. 8.4, we explored several values of γ, ranging from standard MMI ($\gamma = 1$) to 10^{-4}, and using the best regularization term obtained in the previous experiment $C = 10$. The results of this experiment are shown in Fig. 8.4. It is observed a severe over-fitting when the γ-MMI is applied, in particular for $gamma = 10^{-4}$. This over-fitting presents a strange pattern of error increasing and decreasing every 10 iterations, which casually coincides with the 100-best recalculation frequency. Consequently, in Fig. 8.6 we repeated the experiments recalculating the best words every iteration. Additionally we also plotted $\gamma = 2$. It is observed in Fig. 8.6 that the modified γ-MMI obtains a very competitive performance in terms of CER (15%) if applied properly. If we compare the best result in Fig. 8.6 with the best generative result, the former obtains an improvement boost of more than 6 absolute points

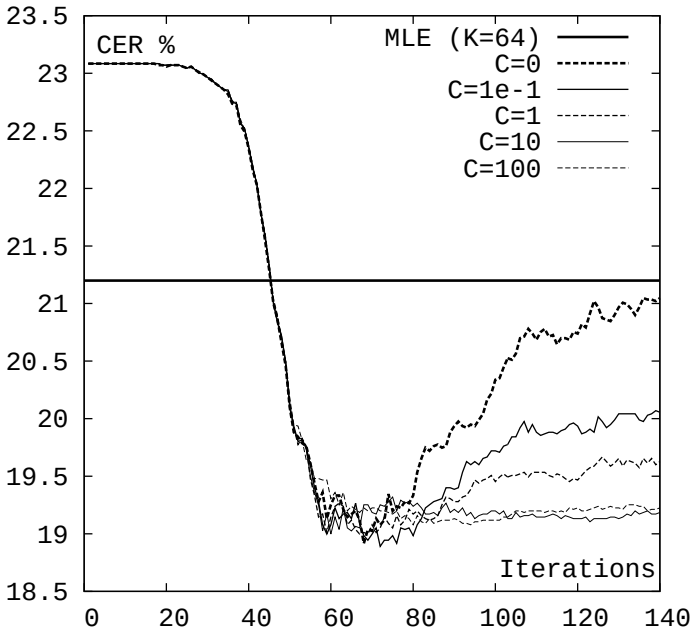


Figure 8.2: Classification error (in %) as a function of RPROP iterations for the regularized MMI criterion with several values of the regularization parameter C. Note that $C = 0$ stands for the non-regularized MMI

with respect to the latter.

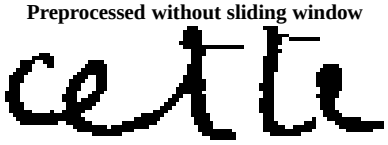
In Fig. 8.3 the behavior of the γ-MMI can be checked for a training sample. As we can see, the use of small values of γ leads to an increase of the separation between classes, which is consistent with the idea that the γ is increasing the competition between classes during the training process. For example, at iteration 100 the separation between the two best hypothesis using $\gamma = 1$ is 100 points, while using $\gamma = 10^{-3}$ the separation increases up to 260 points.

Fig. 8.5 depicts a similar experimentation to that of Fig. 8.2 but for several *test samples*. The first sample (*vous*) is a sample that is misclassified by the MLE model and it is correctly classified using γ-MMI criterion. The remaining two samples are correctly classified by the MLE criterion. However, the first one is finally misclassified by the discriminative model, while the second one remains correctly classified. It is worth noting, that these three cases represent the 10.2%, 2.2% and 74.7% of the test set, respectively.

We also carried out an experiment in order to test the γ-POW$_k$ criterion which theoretically better avoids over-fitting. For this experiment, the best configuration from previous experiments ($C = 10$ and $\gamma = 10^{-3}$) is used as starting point and then several values of the POW parameter $k = \{0, 0.1, 0.5, 1\}$ are tested. Note that γ-POW$_0$ is equivalent to γ-MMI. The results are shown in Fig. 8.7. The closer to 0 the value of k is, the more similar to the γ-MMI criterion the behavior is. Therefore, no improvement is obtained using the γ-POW$_k$

Original image

Preprocessed without sliding window



	MLE		**$\gamma-$MMI ($\gamma = 1$)**							
Iterations	**-**		**50**		**55**		**60**		**100**	
1-best	-	celle	-	celle	-	**cette**	-	**cette**	-	**cette**
2-best	44	**cette**	6	**cette**	73	celle	72	celle	100	celle
3-best	447	Cette	376	dette	342	dette	359	dette	354	dette
4-best	467	dette	406	Cette	382	Cette	388	Cette	393	Cette
5-best	499	celles	497	celles	564	celles	542	celles	485	geste

	MLE		**$\gamma-$MMI ($\gamma = 10^{-1}$)**							
Iterations	**-**		**50**		**55**		**60**		**100**	
1-best	-	celle	-	celle	-	**cette**	-	**cette**	-	**cette**
2-best	44	**cette**	7	**cette**	79	celle	94	celle	116	celle
3-best	447	Cette	375	dette	340	dette	358	dette	336	dette
4-best	467	dette	409	Cette	385	Cette	403	Cette	393	Cette
5-best	499	celles	497	celles	570	celles	556	geste	459	geste

	MLE		**$\gamma-$MMI ($\gamma = 10^{-3}$)**							
Iterations	**-**		**50**		**55**		**60**		**100**	
1-best	-	celle	-	celle	-	**cette**	-	**cette**	-	**cette**
2-best	44	**cette**	6	**cette**	128	celle	207	celle	260	celle
3-best	447	Cette	403	dette	413	dette	417	dette	440	dette
4-best	467	dette	516	celles	663	Cette	631	Cette	657	Cette
5-best	499	celles	530	Cette	667	celles	784	celles	828	telle

Figure 8.3: γ-MMI behavior on a *training* sample for several values of γ. The figures stand for the difference (in logarithmic scale) between each n-best word and the best transcription at each iteration. Bold words highlight the position of the correct word *cette*

instead of γ-MMI.

As discussed before, all experiments were carried out using $K = 26$ components per state. In order to better compare the performance of the MLE and γ-MMI criteria we carried out a final experiment, in which both criteria are tested using several components per state $K \in \{1, 4, 16, 64\}$. For the γ-MMI criteria the best parameters from previous experiments were used ($\gamma = 10^{-3}$ and $C = 10$). Results are shown in Fig. 8.8.

From the results reported in Fig. 8.8 it is clear that γ-MMI outperforms MLE in all cases. The improvement of the MMI decreases as the number of components increases. For example, the improvement using $K = 1$ is about 20 points while using $K = 64$ is about 5 points. It is worth noting, that the best result in this figure is 15.2% which is achieved using $K = 16$ components and it is very similar to the best result obtained with $K = 26$, which we chose for all the previous experimentation. If $K = 16$ components are used, this implies a reduction on the size of the model size of 640%.

Finally, a visual inspection of some Bernoulli prototypes for several training criteria is

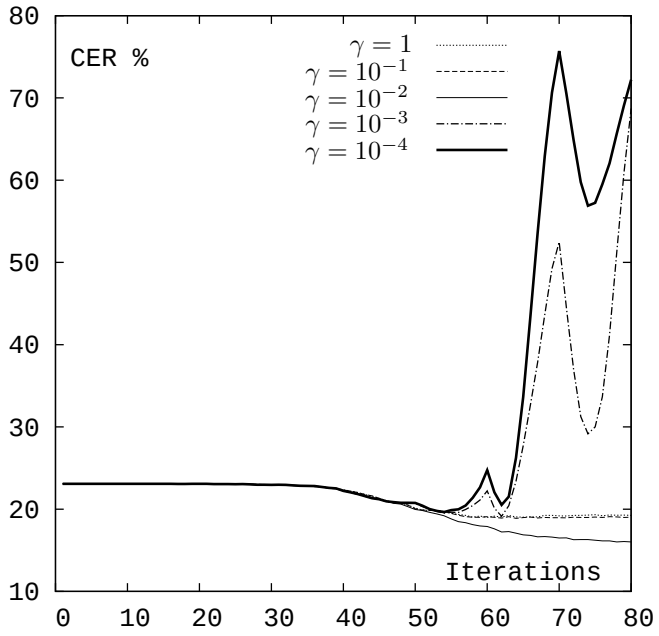


Figure 8.4: Classification error (in %) as a function of RPROP iterations for the modified γ-MMI criterion with regularization $C = 10$ and several values of γ. Note that $\gamma = 1$ corresponds to the standard MMI criterion

given in Fig. 8.9. The Bernoulli prototypes for letters e and s are shown, where the columns represent states, and rows represent mixture components. Provided that the number of mixture components in each state is large ($K = 26$) we have selected the 4 components with the highest mixture coefficients when trained using the MLE criterion. Prototypes are plotted for 3 different training criteria (from left to right): MLE training; the γ-MMI with $\gamma = 10^{-3}$ and regularization $C = 10$; and the conventional MMI training without regularization. It is worth noting that the MLE prototypes are the initial prototypes for both represented discriminative training criteria.

It is observed that the prototypes without regularization are apparently a noise version of the MLE prototypes, however we know that they have a better performance when classifying. A further observation reveals that discriminative training focus on modifying those pixels that discriminate the most while keeping the remaining pixels unmodified. These unmodified pixels are those that keep the same state (0 or 1) for many characters. When no regularization is employed, a pixel that discriminates a single training sample can be set to 1, however, those spurious pixels are eliminated by adding the regularization term.

Incorrect → Correct [10.2% of cases]

	MLE		**γ−MMI ($\gamma = 10^{-3}$)**							
Iterations	**-**		**40**		**45**		**50**		**100**	
1-best	-	virus	-	virus	-	virus	-	**vous**	-	**vous**
2-best	26	Vous	35	**vous**	15	**vous**	16	virus	89	virus
3-best	44	**vous**	40	Vous	50	Vous	93	Vous	318	Vous
4-best	82	bruits	118	bruits	165	bruits	268	bruits	350	nous
5-best	231	plus	243	plus	261	plus	323	plus	419	viens

Correct → Incorrect [2.2% of cases]

	MLE		**γ−MMI ($\gamma = 10^{-3}$)**							
Iterations	**-**		**40**		**45**		**50**		**100**	
1-best	-	**Suite**	-	**Suite**	-	**Suite**	-	suite	-	suite
2-best	97	suite	77	suite	50	suite	16	**Suite**	247	**Suite**
3-best	357	Société	376	Société	395	seule	336	seule	465	seule
4-best	405	société	413	société	402	Société	457	société	698	santé
5-best	428	Santé	431	seule	425	société	470	Société	702	suis

Correct → Correct [74.7% of cases]

	MLE		**γ−MMI ($\gamma = 10^{-3}$)**							
Iterations	**-**		**40**		**45**		**50**		**100**	
1-best	-	**que**	-	**que**	-	**que**	-	**que**	-	**que**
2-best	239	due	233	due	221	due	188	due	167	due
3-best	350	dire	371	dire	396	dire	438	dire	753	dire
4-best	539	grise	590	grise	620	date	628	date	810	date
5-best	595	avez	611	date	639	avez	659	d'une	930	quel

Figure 8.5: γ-MMI behavior on three selected *test* samples. The figures stand for the difference (in logarithmic scale) between each n-best word and the best transcription at each iteration. Bold words highlight the position of the correct word

8.6 Concluding Remarks and Future Work

In this chapter, we presented a log-linear HMM (LLHMM) to recognize isolated handwritten words that directly deals with binarized images without the need of a sophisticated feature extraction process. This model has been proved to be equivalent to Bernoulli HMMs (BHMMs), and in this way, we have provided a framework for discriminatively training BHMMs. Furthermore, this allows us to visually inspect and understand discriminative parameters by transforming them into generative ones.

Several discriminative training criteria have also been analyzed for the LLHMM model: conventional MMI, γ-MMI, and the POW approximation. We analyzed all of them discussing problems (over-fitting, computational cost) and some typical approximation to those problems (regularization term, pruning techniques). All these methods have been tested over the well-known RIMES database of handwritten French words. We have seen that the POW approximation do not report improvements. Furthermore, in all cases discriminative training clearly outperformed the conventional MLE training. In particular, very competitive results

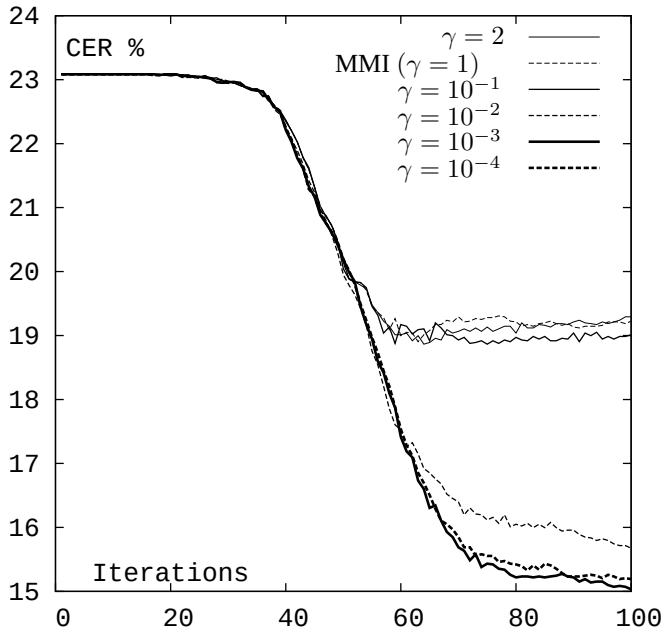


Figure 8.6: Same experimentation than Fig. 8.4 but recalculating the best words every iteration instead of every 10 iterations

were obtained using the γ-MMI training scheme which obtained nearly 15% of CER, or in other words an improvement of more than 6% of absolute points with respect to the generative counterpart. However, there are many more discriminative training criteria such as margin-based or minimum phoneme error. As future work we plan to implement and adapt these discriminative criteria to the proposed model.

The best result obtained in this work on the considered task of the RIMES database is 15%. If we compare our system with the results of the ICDAR 2009 [Grosicki et al., 2009], our system would be positioned in the third position and very close to the second system (13.9%), which is in fact a combination of hybrid HMM/MLP systems, and far from the first system (6.8), which is a system based on a hierarchy of multidimensional recurrent neural networks, and has shown to be extremely competitive in this task. Moreover, if we compare our result with the results reported on the same task in Bianne-Bernard et al. [2011], it is observed that our system outperforms the results of the three systems presented on that paper: a dynamic context-independent system based on HMMs (24.5%), a dynamic context-dependent system based on HMMs (19.6%), and a hybrid HMM/neural network system (20.5%). However, when the three systems are combined an error of 10.9% is obtained. Consequently, as future work we plan to combine the proposed discriminative BHMMs system, with the conventional generative BHMMs system and other state of the art systems, as for instance those based on recurrent neural networks [Graves and Schmidhuber, 2009], in order to measure the

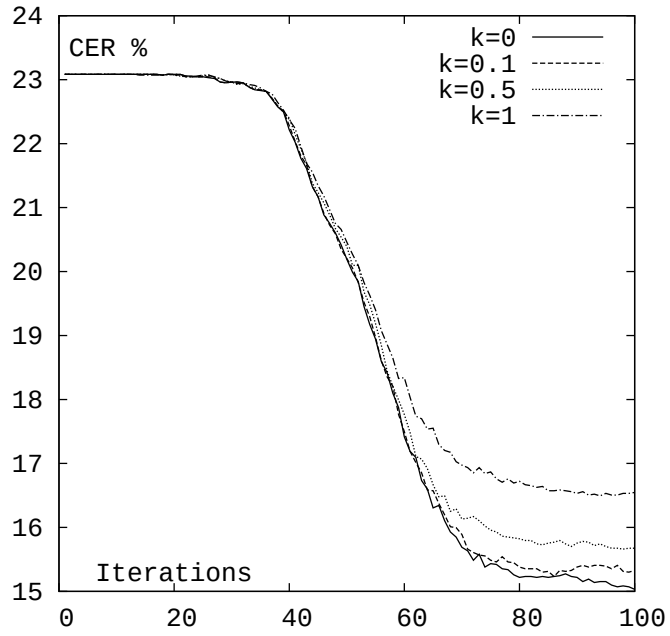


Figure 8.7: Classification error (in %) along of the number iterations for several values of the parameter k using the γ-POW$_k$ criterion and recalculating the best words on each iteration

impact of discriminative BHMMs when combined with other systems.

Finally, we intend to extend all the work developed in this thesis to continuous HTR, that is, a discriminative BHMM in which the words are replaced by word sequences, and hence, the prior probabilities are replaced by a language model.

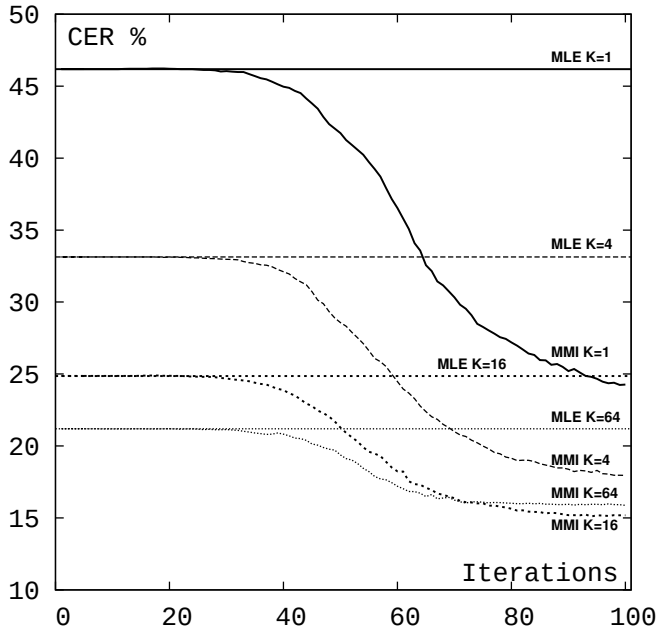


Figure 8.8: MLE and γ-MMI ($\gamma = 10^{-3}, C = 10$) criteria comparison using several components per state

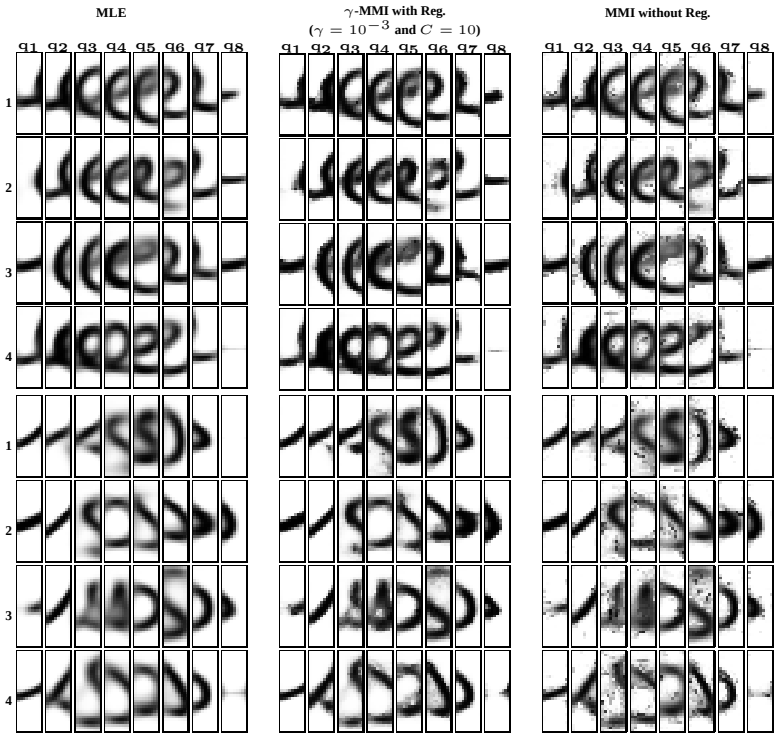


Figure 8.9: Bernoulli prototypes of letters **e** and **s** using three different training criteria (from left to right): MLE, γ-MMI with regularization and MMI without regularization

Bibliography

A.-L. Bianne-Bernard, F. Menasri, R. Al-Hajj Mohamad, C. Mokbel, C. Kermorvant, and L. Likforman-Sulem. Dynamic and contextual information in hmm modeling for handwritten word recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(10): 2066 –2080, oct. 2011. ISSN 0162-8828. doi: 10.1109/TPAMI.2011.22.

Patrick Doetsch, Mahdi Hamdani, Adrià Giménez, Jesús Andrés-Ferrer, Alfons Juan, and Hermann Ney. Comparison of Bernoulli and Gaussian HMMs using a vertical repositioning technique for off-line handwriting recognition. In *2012 International Conference on Frontiers in Handwriting Recognition (ICFHR 2012)*, pages 3 – 7, 2012.

Adrià Giménez, Jesús Andrés-Ferrer, and Alfons Juan. Discriminative Bernoulli HMMs for isolated handwritten word recognition. *Pattern Recognition Letters*, 35(0):157 – 168, 2014. ISSN 0167-8655. doi: http://dx.doi.org/10.1016/j.patrec.2013.05.016.

Alex Graves and Jürgen Schmidhuber. Offline Handwriting Recognition with Multidimensional Recurrent Neural Networks, 2009.

E. Grosicki, M. Carré, J. M. Brodin, and E. Geoffrois. Results of the RIMES evaluation campaign for handwritten mail processing. In *Proc. of the International Conference on Document Analysis and Recognition (ICDAR 2009)*, pages 941–945, 2009.

Georg Heigold. *A Log-Linear Discriminative Modeling Framework for Speech Recognition*. PhD thesis, RWTH Aachen University, Aachen, Germany, June 2010.

Georg Heigold, Thomas Deselaers, Ralf Schlüter, and Hermann Ney. A GIS-like training algorithm for log-linear models with hidden variables. In *ICASSP*, pages 4045–4048, 2008a.

Georg Heigold, Thomas Deselaers, Ralf Schlüter, and Hermann Ney. Modified MMI/MPE: A Direct Evaluation of the Margin in Speech Recognition. In *ICML' 08*, pages 384–391, Helsinki (Finland), July 2008b.

Georg Heigold, Patrick Lehnen, Ralf Schluter, and Hermann Ney. On the equivalence of Gaussian and log-linear HMMs. In *INTERSPEECH' 08*, pages 273–276, Brisbane (Australia), 2008c.

Daniel Povey. *Discriminative Training for Large Vocabulary Speech Recognition*. PhD thesis, Cambridge University Engineering Dept, Mar 2003.

Lawrence Rabiner and Biing-Hwang Juang. *Fundamentals of speech recognition*. Prentice-Hall, 1993.

C. R. Rao and M. B. Rao. *Matrix algebra and its applications to statistics and econometrics*. World Scientific, 1998. ISBN 0-387-94559-8.

Martin Riedmiller and Heinrich Braun. A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP Algorithm. In *IEEE International Conference on Neural Networks*, pages 586–591, 1993.

R. Schluter and W. Macherey. Comparison of discriminative training criteria. In *ICASSP' 98*, volume 1, pages 493–496 vol.1, may 1998. doi: 10.1109/ICASSP.1998.674475.

CHAPTER 9

CONCLUSIONS

Contents

9.1 Summary

In this thesis Bernoulli HMMs (BHMMs) have been proposed for off-line handwriting recognition in order to model text image data in binary form. The BHMM was first introduced in Chapter 4. A BHMM is defined as an HMM in which emission probability functions are modeled using multivariate Bernoulli probability functions. Empirical results were reported on two tasks of off-line handwritten word recognition: Arabic subwords from *CENPARMI* corpus, and English words from *IAM* database. In both cases each word (subword) was modeled using a BHMM, and input images were rescaled to a given height and then binarized. The results obtained, compared with conventional Gaussian HMMs, were promising.

Given the promising results from previous chapter, in Chapter 5 we delved into the use of BHMMs for handwritten word recognition. Specifically, we began the chapter proposing embedded BHMMs. They were formally described first, and then empirically compared with conventional HMMs on a task of handwritten word classification from the IAM database. In this case BHMMs clearly outperformed conventional HMMs in terms of error and number of parameters. Furthermore, embedded BHMMs were also compared with the classifier at word level from previous chapter. As expected, the advantage of using embedded models was clearly confirmed.

After studying the used of embedded BHMMs, we studied the use of Bernoulli mixture emission probability functions. As expected the use Bernoulli mixtures improved the recognition accuracy. Bernoulli mixture HMMs were also compared with a conventional HMM using Gaussian mixtures. In this case, the results of the Bernoulli based recognizer were similar or better than those of the Gaussian mixture based recognizer. Nevertheless, the feature extraction required for the BHMM recognizer was minimal, moreover, it is much simpler in terms of number of parameters.

Apart from our previous basic approach, in which narrow, one-column slices of binary pixels are fed into BHMMs, in Chapter 5 we also studied the use of a sliding window of adequate width to better capture image context at each horizontal position of the word image. Furthermore, windowed BHMMs were improved by the introduction of window *repositioning* techniques. In particular, we considered three techniques of window *repositioning* after window extraction: *vertical*, *horizontal*, and *both*. They only differed in the way in which extracted windows are shifted to align mass and window centers (only in the vertical direction, horizontally or in both directions). Experiments were reported on well-known databases for handwritten word recognition: IFN/ENIT database, the IAM word dataset and the RIMES word dataset. In all cases, reported results were competitive. Specifically, in the case of IFN/ENIT database we obtained very good results, which led us to rank first at the ICFHR 2010 Arabic Handwriting Recognition Competition.

In Chapter 6 BHMM formulae was extended in order to apply BHMMs for continuous handwritten text recognition. Experiments were carried out on the well-known IAM database. Moreover, experiments on real ancient documents were also carried out.

BHMMs are generative models which have practical advantages: easily understandable parameters, well-known training algorithms, etc. However it is known that in several applications they are outperformed by discriminative models (log-linear models, neural networks, etc.). At this point of the thesis we thought about how to apply discriminative training techniques to BHMMs. Instead of directly trying to discriminatively train a BHMM we began

in Chapter 7 by how apply a discriminative training to a simple Bernoulli mixture classifier. Specifically, a mixture of multi-class logistic regression (MMLR) model was proposed for binary data. This model was inspired by Bernoulli mixture classifier. Afterwards, the equivalence between both classifiers was proved. Consequently we obtained two results. On the one hand, we provided a MMI(discriminative) training scheme for Bernoulli mixture classifiers. On the other hand, discriminative parameters can be interpreted by transforming them into generative parameters. This training scheme was tested on the well-known CENPARMI Indian digits database. Reported results shown that the proposed discriminative training scheme clearly outperformed the conventional generative training.

Finally, in Chapter 8 we applied the same strategy from previous chapter to BHMMs. More precisely, we presented a log-linear HMM (LLHMM) to recognize isolated handwritten words that directly deals with binarized data. This model was proved to be equivalent to Bernoulli HMMs (BHMMs), and in this way, we provided a framework for discriminatively training BHMMs. Several discriminative training criteria were analyzed for the LLHMM model: conventional MMI, γ-MMI, and the POW approximation. We analyzed all of them discussing problems (over-fitting, computational cost) and some typical approximation to those problems (regularization term, pruning techniques). All these methods were tested over the well-known RIMES database of handwritten French words. We seen that the POW approximation do not report improvements. Furthermore, in all cases discriminative training clearly outperformed the conventional MLE (generative) training. In particular, very competitive results were obtained using the γ-MMI training scheme.

In summary, the main contributions of this thesis are the following:

- Bernoulli HMMs (BHMMs) have been proposed for off-line handwritten text recognition as an alternative to conventional Gaussian HMMs. The basic idea is to exploit the binary nature of handwritten text by directly modeling binarized text images.
- All the HMM theory required for training models and recognition has been reformulated in order to use BHMMs.
- For binary feature vector extraction we have proposed the use of a sliding window with repositioning. This is a very simple technique which has been proved to improve the recognition accuracy, especially when vertical repositioning is used.
- A MMI training scheme for Bernoulli mixture classifiers has been proposed. This training scheme has been proved to clearly outperform conventional MLE training. This contribution also includes the following contributions:
  - We have proposed a mixture of multi-class logistic regression (MMLR) model for binary data.
  - Equivalence between MMLR models and Bernoulli mixture classifiers has been proved.
  - We provide a way to interpret MMLR parameters as generative parameters.
- A discriminative training scheme for BHMMs has been proposed. This training scheme has been proved to clearly outperform conventional generative training. Specifically we have studied:

- We have proposed a log-linear HMM (LLHMM) to recognize isolated handwritten words that directly deals with binarized data.
- Several discriminative training criteria were analyzed for the LLHMM model: conventional MMI, γ-MMI, and the POW approximation.
- Equivalence between LLHMM for binary data classifiers and BHMMs classifiers has been proved.
- We provide a way to interpret LLHMM parameters as generative parameters.

9.2 Scientific Publications

Most of the work in this thesis has directly yielded international articles in workshops, conferences and journals. In this section, we enumerate these contributions to the scientific community, highlighting the relationship whit this thesis.

Bernoulli HMMs were first proposed (Chapter 4) in one publication in an international workshop:

- **A. Giménez-Pastor** and A. Juan-Císcar. Bernoulli HMMs for Off-line Handwriting Recognition. In Proc. of the 8th Int. Workshop on Pattern Recognition in Information Systems (PRIS 2008), pages 86 – 91, Barcelona (Spain), June 2008.

The extension of BHMMs to embedded BHMMs (Chapter 5) was first published in an international conference:

- **A. Giménez** and A. Juan. Bernoulli HMMs at Sub-word Level for Handwritten Word Recognition. In 4th Iberian Conference on Pattern Recognition and Image Analysis, volume 5524 of LNCS, pages 497 – 504. Springer-Verlag, Póvoa de Varzim (Portugal), June 2009.

while the use of Bernoulli mixture HMMs (Chapter 5) was first published in another international conference:

- **A. Giménez** and A. Juan. Embedded Bernoulli Mixture HMMs for Handwritten Word Recognition. In Proc. of the 10th Int. Conf. on Document Analysis and Recognition (ICDAR 2009), pages 896 – 900, Barcelona (Spain), July 2009.

Recently, a paper with the theory and results from Chapter 5 (both previous papers with updated results, plus windowed BHMMs with repositioning, plus more results on other databases) has been accepted in an international journal:

- **A. Giménez**, I. Khoury, J. Andrés-Ferrer, and A. Juan. Handwriting Word Recognition Using Windowed Bernoulli HMMs. Pattern Recognition Letters, 35(0): 149 – 156, 2012. ISSN 0167-8655. doi: 10.1016/j.patrec.2012.09.002.

The use of BHMMs for continuous HTR (Chapter 6) was first published in:

- **A. Giménez-Pastor** and A. Juan. Embedded Bernoulli Mixture HMMs for Continuous Handwritten Text Recognition. In 13th International Conference on Computer Analysis of Images and Patterns (CAIP), pages 197–204. Springer-Verlag, 2009.

The collaboration with Nicolás Serrano, in which the BHMM toolkit used in this thesis was adapted to conventional mixture Gaussian and modified to support a constrained search (see Chapter 6), resulted in two publications:

- N. Serrano, **A. Giménez**, A. Sanchis, and A. Juan. Active Learning Strategies in Handwritten Text Recognition. In Proceedings of the ICMI- MLMI 2010, pages 1–9. ACM, 2010b.

- N. Serrano, **A. Giménez**, J. Civera, A. Sanchis, and A. Juan. Interactive handwriting recognition with limited user effort. International Journal on Document Analysis and Recognition (IJDAR), pages 1–13, 2013. ISSN 1433-2833. doi: 10.1007/s10032-013-0204-5.

Most of the work on Arabic presented in chapters 5 and 6 (IFN/ENIT and printed Arabic) was done in collaboration with Ihab Khoury, resulting in several publications:

- I. Khoury, **A. Giménez-Pastor**, and A. Juan. Arabic Handwritten Word Recognition Using Bernoulli Mixture HMMs. In The 3rd Palestinian International Conference on Computer and Information Technology (PICCIT), Hebron (Palestine), Feb 2010.

- **A. Giménez**, I. Khoury, and A. Juan. Windowed Bernoulli Mixture HMMs for Arabic Handwritten Word Recognition. In Proceedings of the 12th International Conference on Frontiers in Handwriting Recognition (ICFHR), pages 533–538, Kolkata (India), November 2010. IEEE Computer Society.

- I. Khoury, **A. Giménez**, A. Juan, and J. Andrés-Ferrer. Arabic Printed Word Recognition Using Windowed Bernoulli HMMs. In 17th International Conference on Image, Analysis and Processings (ICIAP 2013), pages 330–339, Naples (Italy), Sep 2013.

Results and theory about discriminative Bernoulli mixture classifiers from Chapter 7 was published in the following international conferences:

- **A. Giménez**, J. Andrés-Ferrer, A. Juan, and N. Serrano. Discriminative Bernoulli Mixture Models for Handwritten Digit Recognition. In 11th International Conference on Document Analysis and Recognition ICDAR 2011, pages 558 – 562. IEEE Computer Society Conference Publishing Services, September 2011. ISBN 978-0-7695-4520-2. doi: 10.1109/ICDAR.2011.118.

The content of Chapter 8 (discriminative BHMMs) yielded a recently accepted paper on a international journal:

- **A. Giménez**, J. Andrés-Ferrer, and A. Juan. Discriminative Bernoulli HMMs for Isolated Handwritten Word Recognition. Pattern Recognition Letters, 35(0): 157 – 168, 2014. ISSN 0167-8655. doi: 10.1016/j.patrec.2013.05.016.

Related to this chapter, a paper has been recently published in collaboration with the HLTPR group from RTWH Aachen University. In this paper discriminative BHMMs were compared with Gaussian HMMs based on neural network features, using in both cases the vertical repositioning technique:

- P. Doetsch, M. Hamdani, **A. Giménez**, J. Andrés-Ferrer, A. Juan, and H. Ney. Comparison of Bernoulli and Gaussian HMMs using a vertical repositioning technique for off-line handwriting recognition. In 2012 International Conference on Frontiers in Handwriting Recognition (ICFHR 2012), pages 3 – 7, 2012.

Finally, during the making of this thesis I have collaborated in several publications, most of them not directly related with the topic of this thesis:

- **A. Giménez-Pastor**, J. Andrés-Ferrer, and A. Juan. Modelizado de la longitud para clasificación de textos. In Nicolás Pérez de la Blanca and Filiberto Plá, editors, Actas del I Workshop sobre Reconocimiento de Formas y Análisis de Imágenes (CEDI 2005), Simposio de la Asociación Española de Reconocimiento de Formas y Análisis de Imágenes (AERFAI). ISBN: 84-9732-445-5, pages 21–28. Thomson, 2005.

- J. González, A. L. Lagarda, J. R. Navarro-Cerdan, L. Eliodoro, **A. Giménez**, F. Casacuberta, J. M. Val, and F. Fabregat. Sishitra: a spanish-to-catalan hybrid machine translation system. In 5th SALTMIL Workshop on Minority Languages, pages 69–73, 2006.

- J. González-Rubio, **A. Giménez**, J. González, A. L. Lagarda, J. R. Navarro-Cerdan, L. Eliodoro, V. J. Fèlix, P. Peris, and F. Casacuberta. Una evaluación exhaustiva de sishitra, un paradigma híbrido en traducción automática. In IV Jornadas en Tecnologías del Habla (IVJTH'2006), pages 93–98, 2006.

- J. González-Rubio, J. González, A. L. Lagarda, **A. Giménez**, J. R. Navarro-Cerdan, and F. Casacuberta. Translation applications under the sishitra framework. In 3rd Language and Technology Conference, pages 453–457, 2007.

- V. Romero, **A. Giménez**, and A. Juan. Explicit Modelling of Invariances in Bernoulli Mixtures for Binary Images. In 3rd Iberian Conference on Pattern Recognition and Image Analysis, volume 4477 of LNCS, pages 539–546. Springer-Verlag, Girona (Spain), June 2007.

- J. A. Silvestre-Cerdà, M. A. del Agua, G. Garcés, G. Gascó, **A. Giménez**, A. Martínez, A. Pérez, I. Sánchez, N. Serrano, R. Spencer, J. D. Valor, J. Andrés-Ferrer, J. Civera, A. Sanchis, and A. Juan. transLectures. In Online Proc. of Advances in Speech and Language Technologies for Iberian Languages (IBERSPEECH 2012), pages 345–351, Madrid (Spain), nov 2012.

- J. A. Silvestre-Cerdà, **A. Giménez**, J. Andrés-Ferrer, J. Civera, and A. Juan. Albayzin Evaluation: The PRHLT-UPV Audio Segmentation System. In VII Jornadas en Tecnología del Habla. III Iberian SLTech Workshop (IberSPEECH 2012), pages 596–600, 2012.

- A. H. Toselli, N. Serrano, **A. Giménez**, I. Khoury, A. Juan, and E. Vidal. Language technology for handwritten text recognition. In Doroteo Torre Toledano, Alfonso Ortega Giménez, António Teixeira, Joaquín González Rodríguez, Luis Hernández Gómez, Rubén San Segundo Hernández, and Daniel Ramos Castro, editors, Advances in Speech and Language Technologies for Iberian Languages, volume 328 of Communications in Computer and Information Science, pages 178–186. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-35291-1. doi: 10.1007/978-3-642-35292-8_19.

APPENDIX $\mathcal{A}$

NOTES ON DISCRIMINATIVE BHMMS

In this appendix some points which were barely seen in Chapter 8 are explained in more detail.

A.1 Discriminative to Generative Transition Probabilities

In this section, we prove that the parameters in (8.46) yield a probability proportional to that of (8.35) when used in (8.4) as the generative parameters of $p_\theta(\mathbf{q}, \mathbf{i} \mid S)$. In order to clarify this, we plug the parameters as computed in (8.46) into (8.4) yielding

$$\prod_{l=1}^{L} \Big[\frac{\exp\left(\bar{\lambda}_{s_l I q_{i_l}}\right) v_{f(s_l, q_{i_l})}}{\psi v_{f(s_l, I)}} \cdot \prod_{t=i_l}^{i_{l+1}-2} \frac{\exp\left(\bar{\lambda}_{s_l q_t q_{t+1}}\right) v_{f(s_l, q_{t+1})}}{\psi v_{f(s_l, q_t)}} \cdot \frac{\exp\left(\bar{\lambda}_{s_l q_{i_{l+1}-1} F}\right) v_{f(s_l, F)}}{\psi v_{f(s_l, q_{i_{l+1}-1})}} \Big], \tag{A.1}$$

where by grouping elements we get

$$\frac{1}{\psi^{T+L}} \frac{h_\lambda(\mathbf{i}, \mathbf{q}; S)}{\exp(\sum_{l,t} \zeta_{s_l q_t})} \prod_{l=1}^{L} \Big[\frac{v_{f(s_l, F)}}{v_{f(s_l, I)}} \frac{v_{f(s_l, q_{i_l})}}{v_{f(s_l, q_{i_{l+1}-1})}} \prod_{t=i_l}^{i_{l+1}-2} \frac{v_{f(s_l, q_{t+1})}}{v_{f(s_l, q_t)}} \Big]. \tag{A.2}$$

Note that, in each segment l the telescope product over $\frac{v_{j'}}{v_j}$ is equal to $\frac{v_{f(s_l, F)}}{v_{f(s_l, I)}} \cdot 1$, and then equation (A.2) is reduced to

$$p_\theta(\mathbf{i}, \mathbf{q} \mid S) = \frac{1}{\psi^{T+L}} \Big[\prod_{l=1}^{L} \frac{v_{f(s_l, F)}}{v_{f(s_l, I)}} \Big] \frac{h_\lambda(\mathbf{i}, \mathbf{q}; S)}{\exp(\sum_{l,t} \zeta_{s_l q_t})}. \tag{A.3}$$

A.2 Efficient LLHMM Parameter Estimation

As seen in Section 8.4, parameter estimation for LLHMM requires the calculation of scores for all hidden variable combinations $(\mathbf{i}, \mathbf{q}, \mathbf{k})$, for example see how $\mathcal{Z}_\lambda(O)$ is calculated in (8.22). Fortunately, as to some extent the proposed LLHMM model is like a BHMM in which parameters are not restricted, the Forward-Backward algorithm can be also applied in a similar way as we did in Section 5.5. In what follows, we redefine the Forward and Backward recursions for the LLHMM case, and show an example of how they are used in parameter estimation. Specifically, we show how $Q^{\gamma}_{n\,m}(\lambda)$ (γ-MMI Criterion) is calculated for a transition parameter $\lambda_{cqq'}$.

A.2.1 The Forward Algorithm

We begin the section with some preliminary definitions

$$h_\lambda(O, \mathbf{i}, \mathbf{q}, \mathbf{k}; S) = \frac{h_\lambda(O, S, \mathbf{i}, \mathbf{q}, \mathbf{k})}{\exp(\lambda_S)} \tag{A.4}$$

$$= h_\lambda(\mathbf{i}, \mathbf{q}; S) \cdot h_\lambda(O, \mathbf{k}; \mathbf{i}, \mathbf{q}, S) \tag{A.5}$$

$$= \exp\big(\sum_{l,t} \lambda_{s_l q_t q_{t+1}} + \sum_{l,t} \lambda_{s_l q_t k_t} + \sum_{ltd} \lambda_{s_l q_t k_t d}\big)\,, \tag{A.6}$$

$$h_\lambda(O; S) = \sum_{\mathbf{i},\mathbf{q},\mathbf{k}} h_\lambda(O, \mathbf{i}, \mathbf{q}, \mathbf{k}; S)\,. \tag{A.7}$$

Note that, as we did in Section 8.3.2, $h_\lambda(\cdot)$ is used to denote scores or scaled probabilities.

In a similar way that in Section 5.5.1, the forward function is defined as

$$\alpha_{Slt}(j) = h_\lambda(O_1^t, i_l \le t < i_{l+1}, q_t = j; S)\,, \tag{A.8}$$

that is, the score of O up to its tth element for a given word S, when it is calculated using all hidden variable combinations which end at state j of symbol s_l, and divided by $\exp(\lambda_S)$. Using this function $h_\lambda(O; S)$ is calculated as

$$h_\lambda(O; S) = \alpha_{SL_S T(F)}\,, \tag{A.9}$$

where L_S and is the number of symbols in S, and T is the length of O. The forward function is a recursion which can be efficiently calculated using dynamic programming. For the special states I and F it is calculated as follows

$$\alpha_{Slt}(I) = \alpha_{Sl-1t}(F) \qquad \begin{matrix} 1 < l \le L_S \\ 1 \le t \le T \end{matrix}\,, \tag{A.10}$$

$$\alpha_{Slt}(F) = \sum_{j=1}^{M_{s_l}} \alpha_{Slt}(j) \exp(\lambda_{s_l jF}) \qquad \begin{matrix} 1 \le l \le L_S \\ 1 \le t \le T \end{matrix}\,, \tag{A.11}$$

while, for regular states, $1 \le j \le M_{s_l}$, we have

$$\alpha_{Slt}(j) = \left[\sum_{i \in \{I,1,\ldots,M_{s_l}\}} \alpha_{Slt-1}(i) \exp(\lambda_{s_l ij})\right] b_{s_l j}(\mathbf{o}_{td})\,, \tag{A.12}$$

with $1 \leq l \leq L_S$ and $1 < t \leq T$, and being $b_{cq}(\mathbf{o})$ defined as follows

$$b_{cq}(\mathbf{o}) = \exp\big(\sum_{k=1}^{K_{cq}}[\lambda_{cqk} + \sum_{d}\lambda_{cqkd}\mathbf{o}_d]\big)\,. \tag{A.13}$$

The base case is for $t = 1$

$$\alpha_{Sl1}(i) = \begin{cases} \exp(\lambda_{s_1 Ii})\; b_{s_1 i}(\mathbf{o}_1) & l = 1, 1 \leq i \leq M_{s_1} \\ 0 & \text{otherwise} \end{cases}\,. \tag{A.14}$$

It is worth noting, that different words with common prefixes can share the $\alpha(.)$ values for the prefix.

A.2.2 The Backward Algorithm

Similarly to the generative case (5.5.2) the backward function is defined as follows

$$\beta_{Slt}(j) = h_\lambda(O_{t+1}^T; i_l \leq t < i_{l+1}, q_t = j, S)\,, \tag{A.15}$$

that is, the score of processing O_{t+1}^T for a given word S, knowing that the tth vector was processed in the state j of the symbol s_l, and divided by $\exp(\lambda_S)$. As in the forward case, the backward function can be efficiently calculated using dynamic programming as follows

$$\beta_{Slt}(F) = \beta_{Sl+1t}(I) \qquad \begin{matrix} 1 \leq l < L_S \\ 1 \leq t < T \end{matrix}\,, \tag{A.16}$$

$$\beta_{Slt}(i) = \exp(\lambda_{s_n l i F})\beta_{Slt}(F) + \sum_{j=1}^{M_{s_l}} \exp(\lambda_{s_l ij}) b_{s_l j}(\mathbf{o}_{t+1})\beta_{Slt+1}(j) \qquad \begin{matrix} 1 \leq l \leq L_S \\ 1 \leq t < T \end{matrix}\,, \tag{A.17}$$

where the base case is defined for $t = T$ as

$$\beta_{SlT}(i) = \begin{cases} \exp(\lambda_{s_L iF}) & l = L_S, 1 \leq i \leq M_{s_l} \\ 0 & \text{otherwise} \end{cases}\,. \tag{A.18}$$

A.2.3 Example of Parameter Estimation Using Forward-Backward

In this section we show how the Forward-Backward algorithm is used in parameter estimation. In particular, we focus on the case of $\lambda_{cqq'}$, a parameter related to an transition between normal states, for the γ-MMI Criterion. And more specifically, we focus on the calculation of $Q^\gamma_{n\,(cjj')}(\lambda)$, where $Q^\gamma_{n\,(cjj')}(\lambda)$ denotes $Q^\gamma_{nm}(\lambda)$ with $m = (cjj')$, and $Q^\gamma_{nm}(\lambda)$ is defined in (8.65).

We begin the example rewriting $Q^{\gamma}_{n\,(cjj')}(\lambda)$. Putting (8.64), (8.66), and (8.59), into (8.65) and rearranging terms we have that $Q^{\gamma}_{n\,(cjj')}(\lambda)$ can be expressed as follows

$$Q^{\gamma}_{n\,(cjj')}(\lambda) = \sum_{\mathbf{i},\mathbf{q},\mathbf{k}} \sum_{S} p_{\lambda\,\gamma}(S, \mathbf{i}, \mathbf{q}, \mathbf{k} \mid O_n) f_{cjj'}(O_n, S, \mathbf{i}, \mathbf{q}, \mathbf{k}) \tag{A.19}$$

$$= \sum_{S} \sum_{\mathbf{i},\mathbf{q},\mathbf{k}} p_{\lambda\,\gamma}(S \mid O) p_{\lambda}(\mathbf{i}, \mathbf{q}, \mathbf{k} \mid S, O) f_{cjj'}(O_n, S, \mathbf{i}, \mathbf{q}, \mathbf{k}) \tag{A.20}$$

$$= \sum_{S} \Big[\frac{\left[\mathcal{Z}_{\lambda}(O_n, S)\right]^{\gamma-1}}{\sum_R \left[\mathcal{Z}_{\lambda}(O_n, R)\right]^{\gamma}} \sum_{\mathbf{i},\mathbf{q},\mathbf{k}} h_{\lambda}(O_n, S, \mathbf{i}, \mathbf{q}, \mathbf{k}) f_{cjj'}(O_n, S, \mathbf{i}, \mathbf{q}, \mathbf{k}) \Big] . \tag{A.21}$$

The normalization term $\mathcal{Z}_{\lambda}(O, S)$ can be easily calculated using the forward algorithm as follows

$$\mathcal{Z}_{\lambda}(O, S) = \sum_{\mathbf{i},\mathbf{q},\mathbf{k}} h_{\lambda}(O, S, \mathbf{i}, \mathbf{q}, \mathbf{k}) \tag{A.22}$$

$$= \exp(\lambda_S) \sum_{\mathbf{i},\mathbf{q},\mathbf{k}} \frac{h_{\lambda}(O, S, \mathbf{i}, \mathbf{q}, \mathbf{k})}{\exp(\lambda_S)} \tag{A.23}$$

$$= \exp(\lambda_S) h_{\lambda}(O; S) = \exp(\lambda_S) \alpha_{SL_S T(F)} . \tag{A.24}$$

The inner sum in (A.21), which we will refer as $\xi^{(n)}_{S(cjj')}$, can be rewritten using forward and backward function as follows

$$\xi^{(n)}_{S(cjj')} = \sum_{\mathbf{i},\mathbf{q},\mathbf{k}} h_{\lambda}(O_n, S, \mathbf{i}, \mathbf{q}, \mathbf{k}) f_{cjj'}(O_n, S, \mathbf{i}, \mathbf{q}, \mathbf{k}) \tag{A.25}$$

$$= \sum_{\mathbf{i},\mathbf{q},\mathbf{k}} \Big[h_{\lambda}(O_n, S, \mathbf{i}, \mathbf{q}, \mathbf{k}) \sum_{l=1}^{L} \delta(s_l, c) \sum_{t=i_l}^{i_{l+1}-2} \delta(q_t, j) \delta(q_{t+1}, j') \Big] \tag{A.26}$$

$$= \sum_{l=1}^{L} \delta(s_l, c) \sum_{\mathbf{i}} \sum_{t=i_l}^{i_{l+1}-2} \sum_{\mathbf{q},\mathbf{k}} \delta(q_t, j) \delta(q_{t+1}, j') h_{\lambda}(O_n, S, \mathbf{i}, \mathbf{q}, \mathbf{k}) \tag{A.27}$$

$$= \sum_{l:s_l=c} \sum_{t=1}^{T-1} h_{\lambda}(O_n, S, i_l \leq t < i_{l+1} - 1, q_t = j, q_{t+1} = j') \tag{A.28}$$

$$= \sum_{l:s_l=c} \sum_{t=1}^{T-1} \exp(\lambda_S) \cdot h_{\lambda}(O_{n\,1}^{\;\;t}, i_l \leq t < i_{l+1}, q_t = j; S) \cdot \tag{A.29}$$

$$\exp(\lambda_{cjj'}) \cdot b_{cj'}(\mathbf{o}_{nt+1}) \cdot \tag{A.30}$$

$$h_{\lambda}(O_{n\,t+2}^{\;\;T}; i_l \leq t+1 < i_{l+1}, q_{t+1} = j', S) \tag{A.31}$$

$$= \exp(\lambda_S) \cdot \sum_{l:s_l=c} \sum_{t=1}^{T-1} \alpha^{(n)}_{Slt}(j) \cdot \exp(\lambda_{cjj'}) \cdot b_{cj'}(\mathbf{o}_{nt+1}) \cdot \beta^{(n)}_{Slt+1}(j') . \tag{A.32}$$

Finally, putting all together we get that

$$Q^{\gamma}_{n\,(cjj')}(\lambda) = \frac{\sum_S \left[\exp(\lambda_S)\cdot \alpha^{(n)}_{SL_ST(F)}\right]^{\gamma-1}\xi^{(n)}_{S(cjj')}}{\sum_R \left[\exp(\lambda_R)\cdot \alpha^{(n)}_{RL_RT(F)}\right]^{\gamma}}\,. \tag{A.33}$$

It is worth noting, that using the same procedure than in the previous case is easy to see that $N^{\gamma}_{n\,(cjj')}(\lambda)$ can be calculated as

$$N^{\gamma}_{n\,(cjj')}(\lambda) = N_{n\,(cjj')}(\lambda) = \frac{\xi^{(n)}_{S(cjj')}}{\exp(\lambda_S)\cdot \alpha^{(n)}_{SL_ST(F)}}\,, \tag{A.34}$$

where here $S = S_n$.

LIST OF FIGURES

LIST OF TABLES