EM initialisation for Bernoulli mixture learning*

Alfons Juan, José García-Hernández, and Enrique Vidal

DSIC, Universitat Politècnica de València, 46071 València, Spain
[ajuan,jogarcia,evidal]@dsic.upv.es

Abstract. Mixture modelling is a hot area in pattern recognition. This paper focuses on the use of Bernoulli mixtures for binary data and, in particular, for binary images. More specifically, six EM initialisation techniques are described and empirically compared on a classification task of handwritten Indian digits. Somehow surprisingly, we have found that a relatively good initialisation for Bernoulli prototypes is to use slightly perturbed versions of the hypercube centre.

Key words: Mixture Models, EM Algorithm, Multivariate Bernoulli Distribution, Initialisation Techniques, Binary Data, Indian Digits

1 Introduction

Mixture modelling is a popular approach for density estimation in both supervised and unsupervised pattern classification [7]. On the one hand, mixtures are flexible enough for finding an appropriate tradeoff between model complexity and the amount of training data available. Usually, model complexity is controlled by varying the number of mixture components while keeping the same (often simple) parametric form for all components. On the other hand, maximum likelihood estimation of mixture parameters can be reliably accomplished by the well-known *Expectation-Maximisation (EM)* algorithm.

Although most research in mixture modelling has focused on mixtures for continuous data, there are many pattern recognition tasks for which binary or discrete mixture models are better suited. This paper focuses on the use of *(multivariate) Bernoulli mixtures* for binary data and, in particular, for *binary images.* EM-based maximum likelihood estimation of Bernoulli mixtures is known even before the general statement of the EM algorithm in 1977 [3]. In fact, the basic formulae appear in a proposed problem of the classic 1973 book by Duda and Hart [4, pp. 256 and 257], who attribute to Wolfe their derivation in 1970 [4, p. 249]. In spite of being known for more than three decades, Bernoulli mixtures as such have seldom been assessed in practice. In [6], for instance, a more complex yet closely-related model is successfully tested on a conventional OCR task, but no comparative results are provided for the simpler, pure Bernoulli mixture

* Work supported by the Spanish "Ministerio de Ciencia y Tecnología" under grant DPI2001-0880-CO2-02.

model. It seems that this pure model has been only applied to non-conventional tasks such as unsupervised modelling of *electropalatographic data* [2].

During the past few years, we have found that Bernoulli mixtures are really effective in certain supervised text classification tasks [5, 9]. Moreover, since these tasks can be considered somewhat non-conventional, we have recently tried out Bernoulli mixtures on a more conventional pattern recognition task involving binary images [8]. As in the case of text classification, the results obtained are encouraging.

In view of their potential, we think that Bernoulli mixtures deserve more attention. In particular, as with any kind of mixtures, it is important to take care of *EM initialisation* in order to fine-tune Bernoulli mixture learning. This paper compares six initialisation techniques, which are described in section 4, after a review of the model and the basic theory on its EM-based maximum likelihood estimation (sections 2 and 3). Then, experimental results are reported on a classification task of handwritten Indian digits.

2 Bernoulli mixtures

A (finite) mixture model consists of a number of *mixture components, I*. It generates a D-dimensional *sample* $\boldsymbol{x} = (x_1, \ldots, x_D)^t$ by first selecting the ith component with *prior probability* $p(i)$, and then generating $\boldsymbol{x}$ in accordance with the ith *component-conditional probability (density) function* $p(\boldsymbol{x} \,|\, i)$. The priors must satisfy the constraints:

$$\sum_{i=1}^{I} p(i) = 1 \quad \text{and} \qquad p(i) \geq 0 \quad (i = 1, \ldots, I). \tag{1}$$

The *posterior probability* of $\boldsymbol{x}$ being actually generated by the ith component can be calculated via the *Bayes' rule* as

$$p(i \,|\, \boldsymbol{x}) = \frac{p(i)\, p(\boldsymbol{x} \,|\, i)}{p(\boldsymbol{x})} \tag{2}$$

where

$$p(\boldsymbol{x}) = \sum_{i=1}^{I} p(i)\, p(\boldsymbol{x} \,|\, i) \tag{3}$$

is the *(unconditional) mixture probability (density) function*.

A Bernoulli mixture model is a particular case of (3) in which each component i has a D-dimensional Bernoulli probability function governed by its own vector of parameters or *prototype* $\boldsymbol{p}_i = (p_{i1}, \ldots, p_{iD})^t \in [0, 1]^D$,

$$p(\boldsymbol{x} \,|\, i) = \prod_{d=1}^{D} p_{id}^{x_d} \, (1 - p_{id})^{1 - x_d} \tag{4}$$

Consider an arbitrary component $p(\boldsymbol{x} \mid i)$. It identifies a certain *subclass* of binary vectors "resembling" its parameter vector or *prototype* $\boldsymbol{p}_i$. In fact, each p_{id} is the probability of bit x_d to be one, whereas $1 - p_{id}$ is the opposite.

Equation (4) is just the product of independent, unidimensional Bernoulli probability functions. Therefore, a single multivariate Bernoulli component can not capture any kind of dependencies or correlations between individual bits. As with other types of mixtures, this is implicitly done by mixing several components in the right proportions.

Also as with other types of mixtures, Bernoulli mixtures can be used as class-conditional models in supervised classification tasks. Let C denote the number of supervised classes. Assume that, for each supervised class c, we know its prior $p(c)$ and its class-conditional probability function $p(\boldsymbol{x} \mid c)$, which is a mixture of I_c Bernoulli components,

$$p(\boldsymbol{x} \mid c) = \sum_{i=1}^{I_c} p(i \mid c)\, p(\boldsymbol{x} \mid c, i) \tag{5}$$

Then, the optimal Bayes decision rule is to assign each pattern vector $\boldsymbol{x}$ to a class $c^*(\boldsymbol{x})$ giving maximum a posteriori probability:

$$\begin{aligned} c^*(\boldsymbol{x}) &= \arg\max_c p(c \mid \boldsymbol{x}) && (6)\\ &= \arg\max_c p(c)\, p(\boldsymbol{x} \mid c) && (7)\\ &= \arg\max_c \log p(c) + \log p(\boldsymbol{x} \mid c) && (8)\\ &= \arg\max_c \log p(c) + \log \sum_{i=1}^{I_c} p(i \mid c) p(\boldsymbol{x} \mid c, i) && (9) \end{aligned}$$

3 Maximum likelihood estimation

Let $X = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ be a set of samples available to learn a Bernoulli mixture model. This is a statistical parameter estimation problem since the mixture is a probability function of known functional form, and all that is unknown is a parameter vector including the priors and component prototypes:

$$\boldsymbol{\Theta} = (p(1), \ldots, p(I), \boldsymbol{p}_1, \ldots, \boldsymbol{p}_I)^t. \tag{10}$$

Here we are excluding the number of components from the estimation problem, as it is a crucial parameter for controlling model complexity and receives special attention in section 5.

Following the maximum likelihood principle, the best parameter values maximise the log-likelihood function of $\boldsymbol{\Theta}$,

$$\mathcal{L}(\boldsymbol{\Theta} \mid X) = \sum_{n=1}^{N} \log \left(\sum_{i=1}^{I} p(i)\, p(\boldsymbol{x}_n \mid i) \right). \tag{11}$$

In order to find these optimal values, it is useful to think of each sample $\boldsymbol{x}_n$ as an *incomplete* component-labelled sample, which can be completed by an indicator vector $\boldsymbol{z}_n = (z_{n1}, \ldots, z_{nI})^t$ with 1 in the position corresponding to the component generating $\boldsymbol{x}_n$ and zeros elsewhere. In doing so, a complete version of the log-likelihood function (11) can be stated as

$$\mathcal{L}_C(\boldsymbol{\Theta}|X, Z) = \sum_{n=1}^{N} \sum_{i=1}^{I} z_{ni} \left(\log p(i) + \log p(\boldsymbol{x}_n|i) \right), \tag{12}$$

where $Z = \{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_N\}$ is the so-called *missing* data.

The form of the log-likelihood function given in (12) is generally preferred because it makes available the well-known *EM* optimisation algorithm (for finite mixtures) [3]. This algorithm proceeds iteratively in two steps. The E(xpectation) step computes the expected value of the missing data given the incomplete data and the current parameters. The M(aximisation) step finds the parameter values which maximise (12), on the basis of the missing data estimated in the E step. In our case, the E step replaces each z_{ni} by the posterior probability of $\boldsymbol{x}_n$ being actually generated by the ith component,

$$z_{ni} = \frac{p(i)\, p(\boldsymbol{x}_n \,|\, i)}{\sum_{i'=1}^{I} p(i')\, p(\boldsymbol{x}_n \,|\, i')} \qquad \begin{pmatrix} n = 1, \ldots, N \\ i = 1, \ldots, I \end{pmatrix}, \tag{13}$$

while the M step finds the maximum likelihood estimates for the priors,

$$p(i) = \frac{1}{N} \sum_{n=1}^{N} z_{ni} \qquad (i = 1, \ldots, I), \tag{14}$$

and the component prototypes,

$$\boldsymbol{p}_i = \frac{1}{\sum_{n=1}^{N} z_{ni}} \sum_{n=1}^{N} z_{ni} \boldsymbol{x}_n \qquad (i = 1, \ldots, I). \tag{15}$$

To start the EM algorithm, initial values for the parameters are required. To do this, it is recommended to avoid "pathological" points in the parameter space such as those touching parameter boundaries and those in which the same prototype is used for all components [2]. Provided that a non-pathological starting point is used, each iteration is guaranteed not to decrease the log-likelihood function and the algorithm is guaranteed to converge to a proper stationary point (local maximum). Also, for the sake of robustness, it is important to introduce some sort of model smoothing.

4 Initialisation techniques

As said above, the only condition for proper EM initialisation is to avoid "pathological" points in the parameter space. Unfortunately, this does not say too much

about the actual technique we should use. So, to clarify ideas, let us first distinguish between mixture proportions and Bernoulli prototypes. Clearly, a natural choice for the initialisation of mixture proportions is to be as impartial as possible, that is, to set them all to the same value:

$$p(i) = \frac{1}{I} \qquad (i = 1, \ldots, I) \tag{16}$$

The tricky problem is to devise an adequate initialisation technique for Bernoulli prototypes. In this case, the simplest yet natural option is to randomly draw each prototype from the open unit hypercube:

$$\boldsymbol{p}_i^{\text{rand}} = rand\big\{\boldsymbol{x} \in [\epsilon, 1-\epsilon]^D\big\} \qquad (i = 1, \ldots, I) \tag{17}$$

where ϵ ($0 < \epsilon \leq 0.5$) is a positive constant intended to exclude extreme probability values. This *random* initialisation was used in [2]. Generally speaking, each possible non-pathological prototype has the same chance of being chosen.

A possible drawback of initialisation (17) comes from the fact that almost all potential prototypes have nothing in common with the training data. Therefore, it is almost sure that it will pick a poor starting point in terms of likelihood. A possible remedy to this drawback is to restrict the set of potential prototypes to the training data,

$$\boldsymbol{p}_i^{\text{proto}} = \boldsymbol{x}_{rand\{1,\ldots,N\}} \qquad (i = 1, \ldots, I) \tag{18}$$

but these prototypes are completely pathological (made up of zeros and ones). A straightforward solution to this pathological nature is to linearly combine (17) and (18):

$$\boldsymbol{p}_i^{\text{rproto}} = \alpha\, \boldsymbol{p}_i^{\text{rand}} + (1-\alpha)\, \boldsymbol{p}_i^{\text{proto}} \qquad (i = 1, \ldots, I) \tag{19}$$

where α ($0 \leq \alpha < 1$) measures the "global randomness" of $\boldsymbol{p}_i^{\text{rproto}}$, as opposed to $1-\alpha$, which measures its "closeness" to the training data. We call this technique *random prototypes*. We used it in [5] ($\alpha \approx 0$) and [8] ($\alpha = 0.75$).

Although (19) will usually provide acceptable initialisations in terms of likelihood, it may be improved in some cases, especially when initialising a mixture of many prototypes. In this case, it is hardly likely that the prototypes chosen will uniformly cover all regions in which training bit vectors appear. On the contrary, it is more likely that some of these regions will become "overpopulated" by prototypes, while other regions will not be covered enough. This eventual failure can be easily prevented by considering prototypes as "facilities" to be located in a "maximally dispersed" way. More specifically, the following algorithm does the job:

$$\boldsymbol{p}_i^{\text{maxmin}} = \begin{cases} \boldsymbol{x}_{rand\{1,\ldots,N\}} & \text{if } i = 1 \\ \arg\max\limits_{\boldsymbol{x} \in X} \min\limits_{i'=1,\ldots,i-1} d(\boldsymbol{x}, \boldsymbol{p}_{i'}^{\text{maxmin}}) & \text{if } i > 1 \end{cases} \tag{20}$$

where $d(\cdot, \cdot)$ is an appropriate distance function for bit vectors (e.g. the Hamming distance). The basic idea behind (20) is simple: the ith prototype chosen is the training bit vector which is farthest away from its closest prototype among those

$(i-1$ prototypes) previously chosen. As with (18), some sort of randomisation or smoothing is also required to avoid exact zeros and ones. We used this algorithm in [9], where it was called *maxmin* initialisation.

The three initialisation techniques discussed so far (random, random prototypes and maxmin) have been used in previous works but they have not been yet compared on the same basis. Since this work is a good opportunity to consider any reasonable initialisation heuristic, we have also considered three additional techniques that can be interpreted as minor variations on the same idea: to use the same vector for all mixture prototypes. The rationale behind this idea is that we have to be as neutral as possible during EM initialisation and then rely on the EM itself for the purpose of specialising each prototype in a different data subclass. Of course, each prototype has to be (randomly) perturbed since the use of the very same vector in all components leads to a well-known pathological starting point [2]. The three minor variations we are talking about are:

1. *Hypercube centre:* all prototypes are slightly perturbed versions of $\mathbf{0.5}$.
2. *Data mean:* all prototypes are slightly perturbed versions of the data mean.
3. *Class mean:* all prototypes of the mixture for class c are perturbed versions of the class c data mean.

5 Experiments

The experiments reported in this section correspond to an OCR task consisting in the recognition of handwritten Indian digits. They were designed to compare, on the same basis, the six initialisation techniques described in the previous section. Also, they can be considered as a continuation of the experiments reported in [8].

The dataset used here comprises the 10425 digit samples included in the non-touching part of the *Indian digits database* recently provided by CENPARMI [1]. Original digit samples are given as binary images of different sizes (minimal bounding boxes). To obtain properly normalised images, both in size and position, two simple preprocessing steps were applied. First, each digit image was pasted onto a square background whose centre was aligned with the digit centre of mass. This square background was a white image large enough (64×64) to accommodate most samples though, in some cases, larger background images were required. Second, given a size S, each digit image was subsampled into $S \times S$ pixels, from which its corresponding binary vector of dimension $D = S^2$ was built. Figure 1 shows one preprocessed example of each Indian digit $(S = 30)$.

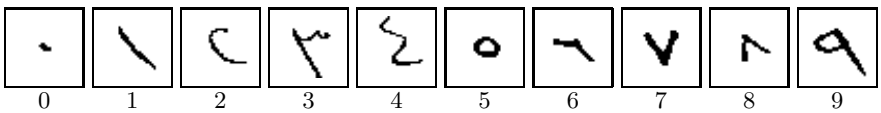

Fig. 1. 30×30 examples of each Indian digit.

The standard experimental procedure for classification error rate estimation in the CENPARMI Indian digits task is a simple partition with 7390 samples for training and 3035 for testing (excluding the extra classes *delimiter* and *comma*). Using this procedure, we obtained the results shown in Figure 2. This Figure includes six graphs arranged in a matrix of two columns and three rows: the graphs in the left column correspond to the random, random prototypes ((19) with $\alpha = 0.75$) and maxmin initialisation techniques; while those in the right column refer to the three minor variations of the "same-vector" idea proposed in the preceding section. For each initialisation technique and each $I \in \{1, 2, 5, 10, 15, 20, 25\}$, the standard experimental procedure was run 50 times, each one entailing an I-component Bernoulli mixture classifier trained from a different random seed. Each graph includes four curves computed from these runs: the (normalised) average log-likelihood of the classifier parameters for both the training and test sets, and the average classification error rate, also for both sets (error bars show standard error). Taking into account the results reported in [8] for this task, here we have only considered a resolution of $S = 20$ pixels. Also, in order to allow direct comparison, only classifiers with class-conditional mixtures of identical number of components, $I_c = I$, have been considered.

From the results shown in Figure 2, it can be said that all techniques give similar results, except random initialisation, which does not seem to be as good as the others. An immediate consequence of this result is that maxmin initialisation becomes less attractive since, in comparison with its alternatives, it is more computationally demanding and difficult to implement. Let us compare random prototypes with hypercube centre, which somehow surprisingly appears to be a good choice among the three variations of the "same-vector" idea. For ease of comparison, the error rate curves (for test data) of these techniques are plotted together in Figure 3. Although standard error bars overlap, we would say that hypercube centre is a bit superior to random prototypes.

6 Conclusions

The results presented in this paper can be considered as a continuation of previous work on the use of Bernoulli mixtures for binary data and, in particular, for binary images. Six EM initialisation techniques have been described and compared on an OCR task consisting in the recognition of handwritten Indian digits. Three of these techniques have been already used in our previous works, though here we have tried to provide a better description of them and, more importantly, a common basis for empirical comparison. The other three techniques, which are proposed here, can be interpreted as minor variations on a very simple idea (to use the same vector for all prototypes). From the empirical results obtained in the Indian digits recognition task, we can conclude that "random prototypes" (linear combination of random parameters and randomly chosen training bit vectors) and "hypercube centre" (all prototypes are slightly perturbed versions of $\mathbf{0.5}$) are relatively good initialisation techniques.

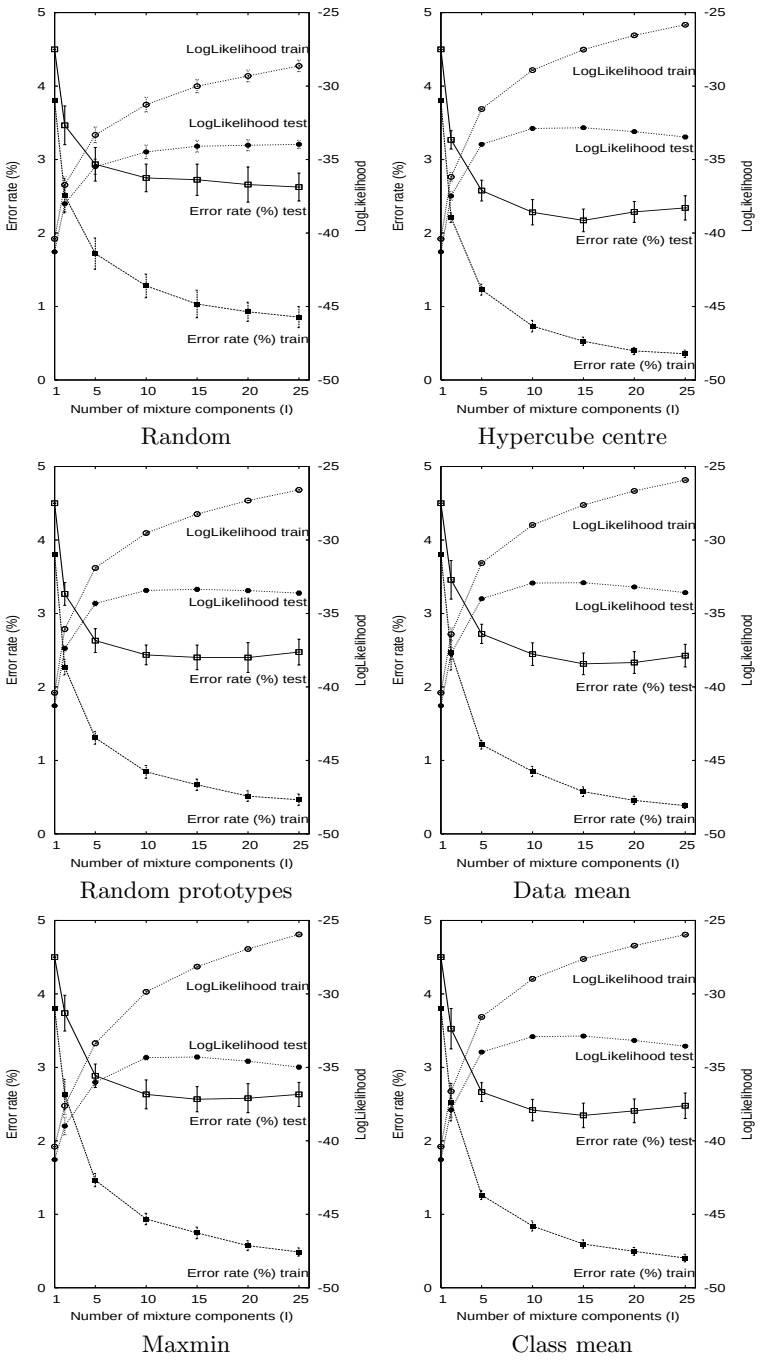

Fig. 2. Comparison of six initialisation techniques: log-likelihood and error rate (for training and test data) of the I-component Bernoulli mixture classifier ($I = 1, \ldots, 25$).

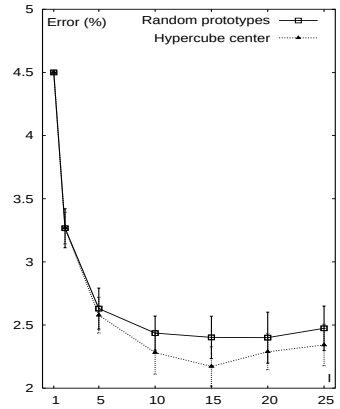

Fig. 3. Comparison of "random prototypes" and "hypercube centre" initialisations: error rate (for test data) of the I-component Bernoulli mixture classifier ($I = 1, \ldots, 25$).

References

1. Y. Al-Ohali, M. Cheriet, and C. Suen. Databases for recognition of handwritten Arabic cheques. *Pattern Recognition*, 36:111–121, 2003.
2. M. A. Carreira-Perpiñán and S. Renals. Practical identifiability of finite mixtures of multivariate Bernoulli distributions. *Neural Computation*, 12(1):141–152, 2000.
3. A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm (with discussion). *Journal of the Royal Statistical Society B*, 39:1–38, 1977.
4. R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. Wiley, 1973.
5. J. González, A. Juan, P. Dupont, E. Vidal, and F. Casacuberta. A Bernoulli mixture model for word categorisation. In *Proc. of the IX Spanish Symposium on Pattern Recognition and Image Analysis*, volume I, pages 165–170, Benicàssim (Spain), May 2001.
6. J. Grim, P. Pudil, and P. Somol. Multivariate Structural Bernoulli Mixtures for Recognition of Handwritten Numerals. In *Proc. of the ICPR 2000*, volume 2, pages 585–589, Barcelona (Spain), September 2000.
7. A. K. Jain, R. P. W. Duin, and J. Mao. Statistical Pattern Recognition: A Review. *IEEE Trans. on PAMI*, 22(1):4–37, 2000.
8. A. Juan and E. Vidal. Bernoulli mixture models for binary images. In *Proc. of the ICPR 2004*. Submitted.
9. A. Juan and E. Vidal. On the use of Bernoulli mixture models for text classification. *Pattern Recognition*, 35(12):2705–2710, December 2002.