Unsupervised naive Bayes for data clustering with mixtures of truncated exponentials

José A. Gámez
Computing Systems Department
Intelligent Systems and Data Mining Group – $i^3\mathcal{A}$
University of Castilla-La Mancha
Albacete, 02071, Spain

Rafael Rumí and Antonio Salmerón
Department of Statistics and Applied Mathematics
Data Analysis Group
University of Almería
Almería, 04120, Spain

Abstract

In this paper we propose a naive Bayes model for unsupervised data clustering, where the class variable is hidden. The feature variables can be discrete or continuous, as the conditional distributions are represented as mixtures of truncated exponentials (MTEs). The number of classes is determined using the *data augmentation* algorithm. The proposed model is compared with the conditional Gaussian model for some real world and synthetic databases.

1 Introduction

Unsupervised classification, frequently known as clustering, is a descriptive task with many applications (pattern recognition, ...) and that can be also used as a preprocessing task in the so-called knowledge discovery from data bases (KDD) process (population segmentation and outlier identification).

Cluster analysis or *clustering* (Anderberg, 1973; Jain et al., 1999) is understood as a decomposition or partition of a data set into groups in such a way that the objects in one group are similar to each other but as different as possible from the objects in other groups. Thus, the main goal of cluster analysis is to detect whether or not the general population is heterogeneous, that is, whether the data fall into distinct groups.

Different types of clustering algorithms can be found in the literature depending on the type of approach they follow. Probably the three main approaches are: partition-based clustering, hierarchical clustering, and probabilistic model-based clustering. From them, the first two approaches yield a *hard* clustering in the sense that clusters are exclusive, while the third one yield a *soft* clustering, that is, an object can belong to more than one cluster following a probability distribution.

In this paper we focus on probabilistic clustering, and from this point of view the data clustering problem can be defined as the inference of a probability distribution for the given database. In this work we allow nominal and numeric variables in the data set, and the novelty of the approach lies in the use of MTE (*Mixture of Truncated Exponential*) (Moral et al., 2001) densities to model the numeric variables. This model has been shown as a clear alternative to the use of Gaussian models in different tasks (inference, classification and learning) but to our knowledge its applicability to clustering remains to be studied. This is precisely the goal of this paper, to do an initial study of applying the MTE model to the data clustering problem.

The paper is structured as follows: first, Sections 2 and 3 give, respectively, some preliminaries about probabilistic clustering and mixtures of truncated exponentials. Section 4 describes the method we have developed in order to obtain a clustering from data by using mixtures of truncated exponentials to model numerical variables. Experiments over several datasets are described in Section 5. Finally, and having in mind that this is a first approach, in Section 6 we discuss about some improvements and applications to our method, and also our conclusions are presented.

2 Probabilistic model-based clustering

The usual way of modeling data clustering in a probabilistic approach is to add a hidden random variable to the data set, i.e., a variable whose value has been missed in all the records. This hidden variable, normally referred to as the *class* variable, will reflect the cluster membership for every case in the data set.

From the previous setting, probabilistic model-based clustering is modeled as a mixture of models (see e.g. (Duda et al., 2001)), where the states of the hidden *class* variable correspond to the components of the mixture (the number of clusters), and the multinomial distribution is used to model discrete variables while the Gaussian distribution is used to model numeric variables. In this way we move to a problem of learning from unlabeled data and usually the EM algorithm (Dempster et al., 1977) is used to carry out the learning task when the graphical structure is fixed and *structural* EM (Friedman, 1998) when the graphical structure also has to be discovered (Peña et al., 2000). In this paper we focus on the simplest model with fixed structure, the so-called *Naive Bayes* structure (fig. 1) where the class is the only root variable and all the attributes are conditionally independent given the class.

Once we decide that our graphical model is fixed, the clustering problem *reduces* to take a dataset of instances and a previously specified number of clusters (k), and work out each

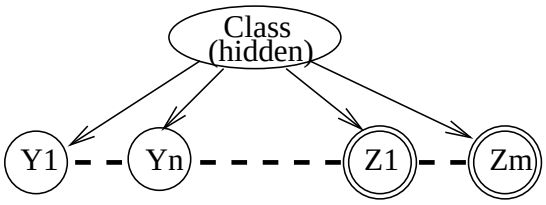

Figure 1: Graphical structure of the model. $Y_1, \ldots, Y_n$ represent discrete/nominal variables while $Z_1, \ldots, Z_m$ represent numeric variables.

cluster's distribution (multinomial or gaussian) and the population distribution between the clusters. To obtain these parameters the EM (*expectation-maximization*) algorithm is used. The EM algorithm works iteratively by carrying out the following two-steps at each iteration:

- Expectation.- Estimate the distribution of the hidden variable C, conditioned on the current setting of the parameter vector θ^k (i.e. the parameters needed to specify the non-hidden variables distributions).

- Maximization.- Taking into account the new distribution of C, use maximum-likelihood to obtain a new set of parameters θ^{k+1} from the observed data.

The algorithm starts from a given initial starting point (e.g. random parameters for C or θ) and under fairly general conditions it is guaranteed to converge to (at least) a local maximum of the log-likelihood function.

Iterative approaches have been described in the literature (Cheeseman and Stutz, 1996) in order to discover also the number of clusters (components of the mixture). The idea is to start with $k = 2$ and use EM to fit a model, then the algorithm tries with $k = 3, 4, ...$ while the log-likelihood improves by adding a new component (cluster) to the mixture. Of course some criteria has to be used in order to prevent overfitting.

To finish with this section we discuss about *how to evaluate the obtained clustering?*. Obviously, a clustering is useful if it produces some interesting insight in the problem we are analyzing. However, in practice and specially in order

to test new algorithms we have to use a different way of measuring the quality of a clustering. In this paper and because we produce a probabilistic description of the dataset, we use the log-likelihood (logL) of the dataset ($\mathbf{D}$) given the model to score a given clustering:

$$logL = \sum_{\mathbf{x} \in \mathbf{D}} \log p(\mathbf{x}|\{\theta, C\}).$$

3 Mixtures of Truncated Exponentials

The MTE model is an alternative to Gaussian models, and can be seen as a generalization to discretization models. It is defined by its corresponding potential and density as follows (Moral et al., 2001):

Definition 1. (MTE potential) Let $\mathbf{X}$ be a mixed n-dimensional random vector. Let $\mathbf{Y} = (Y_1, \ldots, Y_d)$ and $\mathbf{Z} = (Z_1, \ldots, Z_c)$ be the discrete and continuous parts of $\mathbf{X}$, respectively, with $c + d = n$. A function $f : \Omega_{\mathbf{X}} \mapsto \mathbb{R}_0^+$ is a *Mixture of Truncated Exponentials potential (MTE potential)* if one of the next conditions holds:

i. $\mathbf{Y} = \emptyset$ and f can be written as

$$f(\mathbf{x}) = f(\mathbf{z}) = a_0 + \sum_{i=1}^{m} a_i \exp\left\{\sum_{j=1}^{c} b_i^{(j)} z_j\right\} \quad (1)$$

for all $\mathbf{z} \in \Omega_{\mathbf{Z}}$, where a_i, $i = 0, \ldots, m$ and $b_i^{(j)}$, $i = 1, \ldots, m$, $j = 1, \ldots, c$ are real numbers.

ii. $\mathbf{Y} = \emptyset$ and there is a partition $D_1, \ldots, D_k$ of $\Omega_{\mathbf{Z}}$ into hypercubes such that f is defined as

$$f(\mathbf{x}) = f(\mathbf{z}) = f_i(\mathbf{z}) \quad \text{if} \quad \mathbf{z} \in D_i \ ,$$

where each f_i, $i = 1, \ldots, k$ can be written in the form of equation (1).

iii. $\mathbf{Y} \neq \emptyset$ and for each fixed value $\mathbf{y} \in \Omega_{\mathbf{Y}}$, $f_{\mathbf{y}}(\mathbf{z}) = f(\mathbf{y}, \mathbf{z})$ can be defined as in ii.

Definition 2. (MTE density) An MTE potential f is an *MTE density* if

$$\sum_{\mathbf{y} \in \Omega_{\mathbf{Y}}} \int_{\Omega_{\mathbf{Z}}} f(\mathbf{y}, \mathbf{z}) d\mathbf{z} = 1 \ .$$

A univariate MTE density $f(x)$, for a continuous variable X, can be learnt from a sample as follows (Romero et al., 2006):

1. Select the maximum number, s, of intervals to split the domain.
2. Select the maximum number, t, of exponential terms on each interval.
3. A Gaussian kernel density is fitted to the data.
4. The domain of X is split into $k \leq s$ pieces, according to changes in concavity/convexity or increase/decrease in the kernel density.
5. In each piece, an MTE potential with $m \leq t$ exponential terms is fitted to the kernel density by iterative least squares estimation.
6. Update the MTE coefficients to integrate up to one.

When learning an MTE density $f(y)$ from a sample, if Y is discrete, the procedure is:

1. For each state y_i of Y, compute the Maximium Likelihood Estimator of $p(y_i)$, which is $\dfrac{\#y_i}{\text{sample size}}$.

A *conditional MTE density* can be specified by dividing the domain of the conditioning variables and specifying an MTE density for the conditioned variable for each configuration of splits of the conditioning variables (Romero et al., 2006).

In this paper, the conditional MTE densities needed are $f(x|y)$, where Y is the class variable, which is discrete, and X is either a discrete or continuous variable, so the potential is composed by an MTE potential defined for X, for each state of the conditioning variable Y.

Example 1. The function f defined as

$$f(x|y) = \begin{cases} 5 + e^{2x} + e^{x}, & y = 0, \\ & 0 < x < 2, \\ 1 + e^{-x} & y = 0, \\ & 2 \leq x < 3, \\ \frac{1}{5} + e^{-2x} & y = 1, \\ & 0 < x < 2, \\ 4e^{3x} & y = 1, \\ & 2 \leq x < 3. \end{cases}$$

is an MTE potential for a conditional $x|y$ relation, with X continuous, $x \in (0, 3]$. Observe that this potential is not actually a conditional density. It must be normalised to integrate up to one.

In general, an MTE conditional density $f(x|y)$ with one conditioning discrete variable, with states $y_1, \ldots, y_s$, is learnt from a database as follows:

1. Divide the database in s sets, according to variable Y states.

2. For each set, learn a marginal density for X, as shown above, with k and m constant.

4 Probabilistic clustering by using MTEs

In the clustering algorithm we propose here, the conditional distribution for each variable given the class is modeled by an MTE density. In the MTE framework, the domain of the variables is split into pieces and in each resulting interval an MTE potential is fitted to the data. In this work we will use the so-called $five$-parameter MTE, which means that in each split there are at most five parameters to be estimated from data:

$$f(x) = a_0 + a_1 e^{a_2 x} + a_3 e^{a_4 x} \quad . \qquad (2)$$

The choice of the $five$-parameter MTE is motivated by its low complexity and high fitting power (Cobb et al., 2006). The maximum number of splits of the domain of each variable has been set to four, again according to the results in the cited reference.

We start the algorithm with two clusters with the same probability, i.e., the hidden class variable has two states and the marginal probability of each state is 1/2. The conditional distribution for each feature variable given each of the two possible values of the class is initially the same, and is computed as the marginal MTE density estimated from the train database according to the estimation algorithm described in section 3.

The initial model is refined using the *data augmentation* method (Tanner and Wong, 1987):

1. First of all, we divide the train database into two parts, one of them properly for training and the other one for testing the intermediate models.

2. For each record in the test and train databases, the distribution of the class variable is computed.

3. According to the obtained distribution, a value for the class variable is simulated and inserted in the corresponding cell of the database.

4. When a value for the class variable for all the records has been simulated, the conditional distributions of the feature variables are re-estimated using the method described in section 3, using as sample the train database.

5. The log-likelihood of the new model is computed from the test database. If it is higher than the log-likelihood of the initial model, the process is repeated. Otherwise, the process is stopped, returning the best model found for two clusters.

In order to avoid long cycles, we have limited the number of iterations in the procedure above to 100. After obtaining the best model for two clusters, the algorithm continues increasing the number of clusters and repeating the procedure above for three clusters, and so on. The

algorithm continues to increase the number of clusters while the log-likelihood of the model, computed from the test database, is increased.

A model for n clusters in converted into a model for $n+1$ clusters as follows:

1. Let $c_1, \ldots, c_n$ be the states of the class variable.
2. Add a new state, c_{n+1} to the class variable.
3. Update the probability distribution of the class variable by re-computing the probability of c_n and c_{n+1} (the probability of the other values remains unchanged):
   - $p(c_n) := p(c_n)/2$.
   - $p(c_{n+1}) := p(c_n)/2$.
4. For each feature variable X,
   - $f(x|c_{n+1}) := f(x|c_n)$.

At this point, we can formulate the clustering algorithm as follows. The algorithm receives as argument a database with variables $X_1, \ldots, X_n$ and returns a naive Bayes network with variables $X_1, \ldots, X_n, C$, which can be used to predict the class of any object characterised by features $X_1, \ldots, X_n$.

Algorithm CLUSTER(D)

1. Divide the train database D into two parts: one with the 80% of the records in D selected at random, for estimating the parameters (D_p) and another one with the remaining 20% (D_t) for testing the current model.
2. Let *net* be the initial model obtained from D_p as described above.
3. $bestNet := net$.
4. L_0 :=log-likelihood(net,D_t).
5. Refine *net* by *data augmentation* from database D_p.
6. L_1 :=log-likelihood(net,D_t).
7. WHILE ($L_1 > L_0$)
   (a) $bestNet := net$.
   (b) $L_0 := L_1$.
   (c) Add a cluster to *net*.
   (d) Refine *net* by *data augmentation*.
   (e) L_1 :=log-likelihood(net,D_t).
8. RETURN($bestNet$)

5 Experimental evaluation

In order to analyse the performance of the proposed algorithm in the data clustering task, we have selected a set of databases[1] (see Table 1) and the following experimental procedure has been carried out:

- For comparison we have used the EM algorithm implemented in the WEKA data mining suite (Witten and Frank, 2005). This is a classical implementation of the EM algorithm in which discrete and numerical variables are allowed as input attributes. Numerical attributes are modeled by means of Gaussian distributions. In this implementation the user can specify, beforehand, the number of clusters or the EM can decide how many clusters to create by cross validation. In the last case, the number of clusters is initially set to 1, the EM algorithm is performed by using a 10 folds cross validation and the log-likelihood (averaged the 10 folds) is computed. Then, the number of clusters in increased by one and the same process is carried out. If the log-likelihood with the new number of clusters has increased with respect to the previous one, the execution continues, otherwise the clustering obtained in the previous iteration is returned.

- We have divided all the data sets into train (80% of the instances) and test (20% of the instances). Then the training set has been used to learn the model but the test set is used to assess the quality of the final model, i.e., to compute the log-likelihood.

[1] In those datasets usually used for classification the class variable has been removed.

DataSet	#instances	#attributes	discrete?	classes	source
returns	113	5	no	–	(Shenoy and Shenoy, 1999)
pima	768	8	no	2	UCI(Newman et al., 1998)
bupa	345	6	no	2	UCI(Newman et al., 1998)
abalone	4177	9	2	–	UCI(Newman et al., 1998)
waveform	5000	21	no	3	UCI(Newman et al., 1998)
waveform-n	5000	40	no	3	UCI(Newman et al., 1998)
net15	10000	15	2	–	(Romero et al., 2006)
sheeps	3087	23	5	–	real farming data (Gámez, 2005)

Table 1: Brief description of the data sets used in the experiments. The column discrete means if the dataset contains discrete/nominal variables (if so the number of discrete variables is given). The column classes shows the number of class-labels when the dataset comes from classification task.

- Two independent runs have been carried out for each algorithm and data set:
  - The algorithm proposed in this paper is executed over the given training set. Let $\#_{mte}$clusters be the number of clusters it has decided. Then, the EM algorithm is executed receiving as inputs the same training set and number of clusters equal to $\#_{mte}$clusters.
  - The WEKA EM algorithm is executed over the given training set. Let $\#_{em}$clusters be the number of clusters it has decided. Then, our algorithm is executed receiving as inputs the same training set and number of clusters equal to $\#_{em}$clusters.

  In this way we try to compare the performance of both algorithms over the same number of clusters.

Table 2 shows the results obtained. Out of the 8 databases used in the experiments, 7 refer to real world problems, while the other one (Net15) is a randomly generated network with joint MTE distribution (Romero et al., 2006). A rough analysis of the results suggest a better performance of the EM clustering algorithm: in 5 out of the 8 problems, it obtains a model with higher log-likelihood than the MTE-based algorithm. However, it must be pointed out that the number of clusters contained in the best models generated by the MTE-based algorithm is significantly lower than the number of clusters produced by the EM. In some cases, a model with many clusters, all of them with low probability, is not as useful as a model with fewer clusters but with higher probability, even if the last one has lower likelihood.

Another fact that must be pointed out is that, in the four problems where the exact number of classes is known (pima, bupa, waveform and waveform-n) the number of clusters returned by the MTE-based algorithms is more accurate. Besides, forcing the number of clusters in the EM algorithm to be equal to the number of clusters generated by the MTE method, the differences turn in favor of the MTE-based algorithm.

After this analysis, we consider the MTE-based clustering algorithm a promising method. It must be taken into account that the EM algorithm implemented in Weka is very optimised and sophisticated, while the MTE-based method presented here is a first version in which many aspects are still to be improved. Even in these conditions, our method is competitive in some problems. Models which are away from normality (as Net15) are more appropriate for applying the MTE-based algorithm rather than EM.

6 Discussion and concluding remarks

In this paper we have introduced a novel unsupervised clustering algorithm which is based on the MTE model. The model has been tested

DataSet	$\#_{mte}$clusters	MTE	EM	$\#_{em}$clusters	MTE	EM
returns	2	212	129	–	–	–
Pima	3	-4298	-4431	7	-4299	-3628
bupa	5	-1488	-1501	3	-1491	-1489
abalone	11	6662	7832	15	6817	8090
waveform	3	-38118	-33084	11	-38153	-31892
Waveform-n	3	-65200	-59990	11	-65227	-58829
Net15	4	-3418	-6014	23	-3723	-5087
Sheeps	3	-34214	-35102	7	-34215	-34145

Table 2: Results obtained. Columns 2-4 represents the number of clusters discovered by the proposed MTE-based method, its result and the result obtained by EM when using that number of clusters. Columns 5-7 represents the number of clusters discovered by EM algorithm, its result and the result obtained by the MTE-based algorithm when using that number of clusters.

over 8 databases, 7 of them corresponding to real-world problems and some of them commonly used for benchmarks. We consider that the results obtained, in compares with the EM algorithm, are at least promising. The algorithm we have used in this paper can still be improved. For instance, during the model construction we have used a train-test approach for estimating the parameters and validating the intermediate models obtained. However, the EM algorithm uses cross validation. We plan to include 10-fold cross validation in a forthcoming version of the algorithm.

Another aspect that can be improved is the way in which the model is updated when the number of clusters is increased. We have followed a very simplistic approach: for the feature variables, we duplicate the same density function conditional on the previous last cluster, while for the class variable, we divide the probability of the former last cluster with the new one. We think that a more sophisticated approach, based on component splitting and parameter reusing similar to the techniques proposed in (Karciauskas, 2005) could improve the performance of our algorithm.

The algorithm proposed in this paper can be applied not only to clustering problems, but also to construct an algorithm for approximate probability propagation in hybrid Bayesian networks. This idea was firstly proposed in (Lowd and Domingos, 2005), and is based on the simplicity of probability propagation in naive Bayes structures. A naive Bayes model with discrete class variable, actually represents the conditional distribution of each variable to the rest of variables as a mixture of marginal distributions, being the number of components in the mixture equal to the number of states of the class variable. So, instead of constructing a general Bayesian network from a database, if the goal is probability propagation, this approach can be competitive. The idea is specially interesting for the MTE model, since probability propagation has a high complexity for this model.

Acknowledgments

Acknowledgements This work has been supported by Spanish MEC under project TIN2004-06204-C03-{01,03}.

References

M.R. Anderberg. 1973. *Cluster Analysis for Applications*. Academic Press.

P. Cheeseman and J. Stutz. 1996. Bayesian classification (AUTOCLASS): Theory and results. In U. M. Fayyad, G. Piatetsky-Shapiro, P Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 153–180. AAAI Press/MIT Press.

B.R. Cobb, P.P. Shenoy, and R. Rumí. 2006. Approximating probability density functions with mixtures of truncated exponentials. *Statistics and Computing*, In press.

A. P. Dempster, N. M. Laird, and D.B. Rubin. 1977. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society*, 1(39):1–38.

R.O. Duda, P.E. Hart, and D.J. Stork. 2001. *Pattern Recognition*. Wiley.

N. Friedman. 1998. The Bayesian structural EM algorithm. In Gregory F. Cooper and Serafín Moral, editors, *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence (UAI-98)*, pages 129–138. Morgan Kaufmann.

J.A. Gámez. 2005. Mining the ESROM: A study of breeding value prediction in manchego sheep by means of classification techniques plus attribute selection and construction. Technical Report DIAB-05-01-3, Computer Systems Department. University of Castilla-La Mancha.

A.K. Jain, M.M. Murty, and P.J. Flynn. 1999. Data clustering: a review. *ACM COmputing Surveys*, 31(3):264–323.

G. Karciauskas. 2005. *Learning with hidden variables: A parameter reusing approach for tree-structured Bayesian networks*. Ph.D. thesis, Department of Computer Science. Aalborg University.

D. Lowd and P. Domingos. 2005. Naive bayes models for probability estimation. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 529–536. ACM Press.

S. Moral, R. Rumí, and A. Salmerón. 2001. Mixtures of truncated exponentials in hybrid Bayesian networks. In *Lecture Notes in Artificial Intelligence*, volume 2143, pages 135–143.

D.J. Newman, S. Hettich, C.L. Blake, and C.J. Merz. 1998. UCI Repository of machine learning databases. www.ics.uci.edu/∼mlearn/MLRepository.html.

J.M. Peña, J.A. Lozano, and P. Larrañaga. 2000. An improved bayesian structural EM algorithm for learning bayesian networks for clustering. *Pattern Recognition Letters*, 21:779–786.

V. Romero, R. Rumí, and A. Salmerón. 2006. Learning hybrid Bayesian networks using mixtures of truncated exponentials. *Inernational Journal of Approximate Reasoning*, 42:54–68.

C. Shenoy and P.P. Shenoy. 1999. Bayesian network models of portfolio risk and return. In W. Lo Y.S. Abu-Mostafa, B. LeBaron and A.S. Weigend, editors, *Computational Finance*, pages 85–104. MIT Press, Cambridge, MA.

M.A. Tanner and W.H Wong. 1987. The calculation of posterior distributions by data augmentation (with discussion). *Journal of the American Statistical Association*, 82:528–550.

I.H. Witten and E. Frank. 2005. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann.