An Adaptive Condensing Algorithm Based on Mixtures of Gaussians

M. Lozano J.M. Sotoca J.S. Sánchez
F. Pla
Dept. Lenguajes y Sistemas Informáticos, Universitat Jaume I
Campus Riu Sec, 12071 Castellón, Spain
[lozano,sotoca,sanchez,pla]@uji.es

Abstract. In this paper, a new adaptive approach to training set size reduction, estimating *probability density functions* is presented. This scheme consists of defining a very small number of prototypes that represent all the original instances, using mixtures of gaussians. Although the ultimate aim of the algorithm proposed here is to obtain a strongly reduced training set, the performance is empirically evaluated over eleven real datasets by comparing not only the reduction rate but also the classification accuracy.

1 Introduction

Currently, in many domains the size of the datasets is so extremely large that real-time systems cannot afford the time and storage requirements to process them. Under these conditions, classifying, understanding or compressing the available information can become a very problematic task. This problem is specially dramatic in the case of using some distance-based learning algorithm, such as the Nearest Neighbour (NN) rule [5]. The basic NN scheme must search through all the available training instances (large memory requirements) to classify a new input sample (slow during classification). On the other hand, since the NN rule stores every prototype in the training set (TS), noisy instances are stored as well, which can considerably degrade the classification accuracy.

The many existing proposals can be categorised into two main groups. First, those schemes that merely select a subset of the original prototypes [1, 6, 10, 19, 20] and second, those that modify them [2, 3, 4, 13] (adaptive schemes). One problem related with using the original instances is that there may not be any vector located at the precise point that would make the most accurate learning algorithm. Correspondingly, prototypes can be artificially generated to exist exactly where they are needed.

This paper focuses on the problem of appropriately reducing the TS size by artificially generating a subset of prototypes. The primary aim of the proposal presented in this paper is to obtain a considerable size reduction rate, but without an important decrease in classification accuracy. This approach is based on the estimation of the *probability density functions* (*pdf*) using a mixture of gaussians. In order to obtain this, we use an optimisation method based on the well-know *expectation-maximisation* (EM) algorithm [7, 15, 16].

In the establishment of a pattern recognition system, situations where the class conditional distribution of the system is unknown, are not strange. So, having only one TS whose points are labelled in several classes is usual. Apart from the rather trivial cases where the information is governed by a simple distribution, it is necessary to obtain an estimator that collects the different class-conditional *pdf*. Mixture models are able to represent arbitrarily complex *pdf*. This fact makes them an excellent choice for representing complex class conditional *pdf* in Bayesian supervised learning scenarios [11, 12].

The structure of the rest of this paper is as follows. Section 2 provides a description of employment of mixtures of gaussians in multi-modal class distributions. Section 3 introduces a new adaptive condensing scheme, here called *MixtGauss*, which is based on moving mixtures of gaussians to optimal locations in the feature space. The databases used and the experiments carried out are described in Section 4. Results are shown and discussed in Section 5. Finally, the main conclusions along with further extensions are depicted in Section 6.

2 Mixtures of Gaussians in Multi-Modal Class Distributions

In general, we have a TS with N_t instances, where each instance $\mathbf{x}$ is a point $\mathbf{x} = \{x_1, \ldots, x_d\} \in \Re^d$, in a d-dimensional feature space. Then, for a given sample, the aim is to assign it to the correct class where $C = \{c_1, \ldots, c_J\}$ is a finite set of the different J classes in the TS. The different points of the TS follow a spatial class distribution according to their true class-conditional *pdf* $P(\mathbf{x}|c_j)$ and the respective *a priori* probability $P(c_j)$, $c_j \in C$. So, we can say that a vector $\mathbf{x}$ can be then optimally classified using the Bayes rule or maximum *a posteriori* probability decision rule based on the knowledge of the components $P(c_j)P(\mathbf{x}|c_j)$ for each class c_j.

In practice, these class-conditional *pdf* do not have any underlying structure assumed and any prior knowledge about the shapes of these *pdf* are not required to solve the problem. So, it is only necessary to obtain a density estimation of the distribution.

A simplification of the Bayes rule, which consists of assuming that the features may be statistically independent, will be used in the algorithm proposed in this paper. This supposes that for the multivariate case, independence can be defined through the product of the probabilities in each feature as $p(\mathbf{x}) = \prod_{k=1}^{d} p(x_k)$. This rule is called Naive-Bayes [18] and can be defined as follows:

$$C_{Bayes} = \max_{j=1 \ldots c_J} P(c_j)P^*(\mathbf{x}|c_j) = \max_{j=1 \ldots c_J} P(c_j) \prod_{k=1}^{d} f^*(x_k|c_j) \tag{1}$$

where $f^*(x_k|c_j)$ is an estimation of class-conditional *pdf* of c_j in the feature k.

A natural way to deal with a density estimator is to consider a mixture density of modes. One approach to solve $P^*(\mathbf{x}|c_j)$, is retaining the capacity to reflect the local structure of the distribution by means of blocks of mixtures $P_m(\mathbf{x}|c_j)$, where each block is the product of the probabilities in each feature as [18]:

$$P^*(\mathbf{x}|c_j) = \sum_{m=1}^{M} \alpha_{m|j} P_m(\mathbf{x}|c_j) = \sum_{m=1}^{M} \alpha_{m|j} \prod_{k=1}^{d} \Theta_{m|kj}(x_k) \tag{2}$$

where M is the number of modes, $\alpha_{m|j}$ is an *a priori* probability of the mode m^{th} in the class c_j, and $\Theta_{m|kj}$ is its class-conditional *pdf* projected on the feature k.

Then, we can consider the characterisation of the distribution like a parametric unsupervised learning problem [8] using a mixture of modes of multidimensional normal distribution. The EM algorithm will be here used to estimate these unknown modes.

3 An Adaptive Condensing Scheme

The final aim of the condensing process proposed here is to obtain a point distribution representing a class, for every class present in the TS. These point distributions should represent the decision boundaries. The general shape of the decision boundaries is maintained, while some Gaussians are used for representing each class. The more Gaussians are included in the point distribution, the more accurate the representation of the decision boundaries is. In addition, a consistency criterion will be applied to the points representing a class, for instance, in Hart's algorithm, the consistency criterion consisted of the correct classification of all prototypes in the TS by NN classification using the condensed set.

In the method here presented, a condensed set CS is said to be *consistent* with respect to a TS if the estimation error is small, when the class for each pattern in the TS is estimated by the NN rule for the condensed set. Given a set of prototypes representing the class distribution, we can use the classification rate of the TS in order to measure how much consistent it is.

As the *consistent* condition is maintained, the smaller the number of prototypes needed for representing a class in the condensed set, the better the result is. On the other hand, for some applications it is useful to be able to pre-fix the condensed set size.

To look for a point distribution, imagine a problem of two Gaussian distribution classes, in the euclidean space. The best condensed set would be the mean of the Gauss function in each class. Therefore, let us suppose a class distribution that could be modelled by a mixture of gaussians. The centre of these gaussians could become the set of prototypes representing the class distribution. That is, the condensed set.

3.1 Initialisation

The EM algorithm for finite mixture fitting has several drawbacks [9]: it is a local method, thus it is sensitive to initialisation because the likelihood function of a mixture model is not unimodal. Another important issue in mixture modelling is the selection of the number of components. With too many components, the mixture may over-fit the data, while a mixture with too few components may not be flexible enough to approximate the true underlying model.

We assume that all the components have the same functional form. For example, they are all *d*-variate gaussians, each one being thus fully characterised by the parameter vector Θ_m.

Taking into account these facts, we have to provide a number M of gaussians to represent each class distribution; this number will correspond to the number of prototypes in each class in the condensed set. At the initial stage, all the gaussians belonging to a class are located on the centroid of the prototypes from that class. Adding random disturbances to the centroid of a class, we obtain a mixture of gaussians that represents the class distribution. For each gaussian, the initial variance is 1/10 of the range of each dimension.

3.2 Optimisation

After the initialisation stage, an iterative optimisation process, based on the EM algorithm, is carried out in order to find an optimal location of the mixtures of gaussians. This iterative procedure converges to a *maximum likelihood* estimate of the mixture parameters. Accordingly, to fit each class distribution to a mixture of gaussians, we iterate on the following steps:

E-step Compute the contributions of prototypes $\mathbf{x}^t$ in the class c_j, to belong to the set $\{\mathbf{x}^1, \ldots, \mathbf{x}^{N_j}\}$, where N_j is the number of elements to c_j. The conditional *pdf* for the m^{th} mode is:

$$P_m(\mathbf{x}^t|c_j) = \frac{\alpha_{m|j} \prod_{k=1}^{d} N(x_k^t : \mu_{m|kj}, \sigma_{m|kj})}{\sum_{l=1}^{M} \alpha_{l|j} \prod_{k=1}^{d} N(x_k^t : \mu_{l|kj}, \sigma_{l|kj})} \tag{3}$$

This function is normalised such $\sum_{m=1}^{M} \alpha_{m|j} = 1$. We represent the multidimensional gaussians as a product of unidimensional normal distributions, with $\mu_{m|kj}$ and $\sigma_{m|kj}$ as the means and standard deviations.

M-step Compute the parameters of the modes m^{th} for each value x^t that exists in the class c_j.

$$\alpha_{m|j} = \frac{1}{N_j} \sum_{t=1}^{N_j} P_m(\mathbf{x}^t|c_j) \qquad \mu_{m|kj} = \frac{\sum_{t=1}^{N_j} P_m(\mathbf{x}^t|c_j) x_k^t}{\sum_{t=1}^{N_j} P_m(\mathbf{x}^t|c_j)} \tag{4}$$

$$\sigma_{m|kj} = \frac{\sum_{t=1}^{N_j} P_m(\mathbf{x}^t|c_j)(x_k^t - \mu_{m|kj})^2}{\sum_{t=1}^{N_j} P_m(\mathbf{x}^t|c_j)} \tag{5}$$

3.3 Stopping Criterion

The iterative optimisation process just described can cause an overlapping between the gaussians from different classes, which will produce a deterioration in classification accuracy. Therefore, this process should stop when no class obtains an increase in performance with respect to a previous iteration.

To this end, after each iteration, the consistency criterion is estimated by calculating the 1-NN classification error over the TS using the centroids of the gaussians in their current locations. Movement of the gaussians is carried out while any class obtains a classification rate higher than that of the previous step.

3.4 MixtGauss Algorithm

The process introduced in the previous sections can be viewed as an adaptive condensing scheme, in which the points of the resulting set will correspond to the centroids of the gaussians. Algorithmically, it can be summarised as it is shown in Algorithm 1.

4 Description of Databases and Experiments

Eleven real datasets (Table 1) have been taken from the UCI Repository [17] to assess the behaviour of the algorithm here introduced. The experiments have been conducted to compare *MixtGauss* algorithm to *Chen*'s scheme, in terms of both TS size reduction and accuracy of the condensed 1-NN classification rule.

The algorithm proposed here as in the case of *Chen*'s, needs to be applied to overlap-free (no overlapping among different class regions) datasets. Thus, as a general rule, and according to previously published results [2, 21], the Wilson's editing has been considered to properly remove overlapping between classes. The parameter involved (k) has been obtained in our experiments by performing a five-fold cross-validation experiment using only the TS and

Algorithm 1: *MixtGauss*

```
(* initialisation *)
**for** $c = 1..number\_of\_classes$ **do**
   $mean[c] = CalculateMean(c, TS)$
   **for** $g = 1..number\_of\_gaussians\_per\_class$ **do**
      $gaussians[c, g] = mean[c] + Random_Disturbance()$
   **end for**
**end for**
(* optimisation *)
**repeat**
   $previous\_gaussians[c, g] = gaussians[c, g]$
   $gaussians[c, g] = EMstep()$
   $previous\_accuracy = current\_accuracy$
   $current\_accuracy = CalculateAccuracy()$
   $classes\_improve = 0$
   **for** $c = 1..number\_of\_classes$ **do**
      **if** $current\_accuracy[c] > previous\_accuracy[c]$ **then**
         $classes\_improve = classes\_improve + 1$
      **else**
         **if** $current\_accuracy[c]! = previous\_accuracy[c]$ **then**
            **for** $g = 1..number\_of\_gaussians\_per\_class$ **do**
               $gaussians[c, g] = previous\_gaussians[c, g]$
            **end for**
         **end if**
      **end if**
   **end for**
**until** $classes\_improve == 0$
```

computing the average classification accuracies for different values of k. The best edited set (including the non-edited TS) is thus selected as input for the different condensing schemes.

Data set	No. classes	No. features	TS size	Test set size
Cancer	2	9	546	137
Pima	2	6	614	154
Glass	6	9	171	43
Heart	2	13	216	54
Liver	2	6	276	69
Vehicle	4	18	677	169
Vowel	11	10	422	106
Wine	3	13	142	36
Phoneme	2	5	4323	1080
Satimage	6	36	5148	1287
Texture	11	40	4400	1100

Table 1: Datasets used in the experiments.

The two algorithms compared here have some different characteristics that make them not exactly with the same conditions for comparison. In *MixtGauss* algorithm, a number M of gaussians per class has to be given, which results in a number of prototypes in the condensed set equal to $M \cdot J$ (J denotes the number of classes). On the other hand, *Chen*'s scheme requires the size of the resulting condensed set, which will be defined as $M \cdot J$ in each experiment, thus obtaining the same reduction rate in both schemes.

Another important difference between *MixtGauss* and *Chen*'s condensing algorithms refers to the fact that in the approach proposed here, each class is represented by the same number of prototypes, while in *Chen*'s procedure, the size can be different for each class.

5 Experimental Results and Discussion

Tables 2 and 3 report the 1-NN accuracy (standard deviation in brackets) and the reduction rate with respect to the edited TS when using a condensed set size of $3 \cdot J$ and $5 \cdot J$, respectively. Results corresponding to the edited set without condensing are also included in Table 2 for comparison purposes.

	Edited Acc.%	Edited Size	Chen Acc.%	MixtGauss Acc.%	Size Reduc.%
Cancer	95.75 (2.50)	538	96.78 (1.11)	95.32 (4.21)	98.88
Pima	73.05 (3.57)	468	65.89 (1.84)	69.27 (3.66)	98.72
Glass	71.40 (3.78)	171	56.04 (10.02)	59.46 (15.54)	89.49
Heart	65.88 (4.69)	155	64.07 (5.91)	64.07 (7.45)	96.13
Liver	65.85 (5.25)	203	55.69 (3.06)	60.01 (2.28)	97.04
Vehicle	64.41 (2.11)	677	46.66 (4.91)	49.72 (7.33)	98.23
Vowel	97.90 (1.23)	422	42.71 (7.76)	65.44 (4.18)	92.19
Wine	71.35 (4.47)	109	72.37 (8.42)	73.56 (10.14)	91.77
Phoneme	72.05 (8.58)	3,945	69.45 (4.09)	75.27 (15.07)	99.85
Satimage	82.99 (12.50)	4,975	65.59 (9.60)	74.76 (13.66)	99.64
Texture	98.96 (0.38)	4,400	50.76 (4.29)	79.71 (0.83)	99.25
Average	78.15		62.36	69.69	96.47

Table 2: Results for the edited TS, and for the *Chen*'s and *MixtGauss* condensed sets with $3 \cdot J$ prototypes.

Several comments can be made from the results in these tables. As expected, classification accuracy strongly depends on the number of prototypes in the condensed set. Correspondingly, the classification percentages obtained when using $5 \cdot J$ prototypes (65.91% and 71.21% in average) are higher than those corresponding to $3 \cdot J$ in most cases (62.36% and 69.69% in average). When comparing these accuracies with those provided by the edited set, one can see that differences are not very important taking into account that the set size has been reduced in more than 90%.

MixtGauss algorithm yields higher classification accuracy than *Chen*'s scheme in both experiments. Differences between them are more significant when using a smaller number of resulting prototypes, thus indicating that *MixtGauss* algorithm is able to fit the class distributions in a more appropriate way than *Chen*'s procedure.

In order to emphasise the results just commented, in Fig. 1 we show the averaged accuracy obtained with the resulting condensed sets for the *Vowel* database, for different sizes from 11 to 209 prototypes (that is, 50% of the edited set size: a smaller reduction seems to do not make sense). As can be seen, differences between both methods are much more important as a higher reduction has been applied. When the reduction level is close to 50%, the classification accuracies are similar enough and they are very close to that of the edited set.

	Chen Accuracy %	MixtGauss Accuracy %	Size Reduction %
Cancer	96.34 (2.60)	94.30 (6.39)	98.14
Pima	69.54 (3.12)	68.88 (3.97)	97.87
Glass	63.99 (6.65)	61.34 (8.19)	82.48
Heart	65.15 (3.30)	63.33 (6.03)	93.56
Liver	57.12 (3.26)	56.86 (5.00)	95.06
Vehicle	50.20 (5.75)	54.96 (1.55)	97.04
Vowel	48.89 (5.67)	82.28 (2.94)	86.98
Wine	69.62 (7.30)	73.56 (10.14)	86.29
Phoneme	69.36 (3.99)	71.11 (19.77)	99.75
Satimage	70.87 (7.86)	77.01 (12.36)	99.40
Texture	63.92 (2.96)	79.73 (0.73)	98.75
Average	65.91	71.21	94.12

Table 3: Results for the *Chen*'s and *MixtGauss* condensed sets with $5 \cdot J$ prototypes.

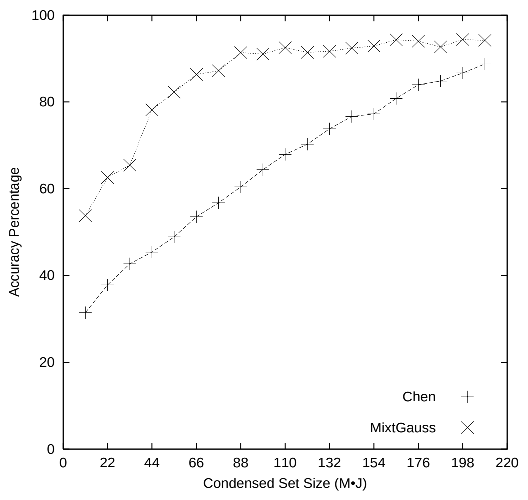

Figure 1: Accuracy when varying the number of prototypes in the *Vowel* database.

6 Concluding Remarks

In this paper, a new adaptive approach to TS size reduction have been introduced. This algorithm primarily consists of replacing each class of the original set by a given number of new prototypes. The position of these new prototypes are estimated by using the EM algorithm to optimise the location of a number of gaussians.

From the experiments carried out, it seems that *MixtGauss* algorithm provides good results, higher in accuracy rate than those obtained by other adaptive algorithms, while the size reduction is the same. On the other hand, differences between them are more significant in the case of a higher reduction in size.

Future work is now directed to get an independent number of gaussians for each class, according to their spatial distribution. On the other hand, it is also necessary to conduct a more extensive comparison of *MixtGauss* with other adaptive methods present in the literature.

Acknowledgements

This work has been supported by grants TIC2003-08496 and CPI2001-2956-C02-02 from CICYT MCT and projects IST-2001-37306 from the European Union and P1-1B2002-07 from *Fundació Caixa Castelló-Bancaixa*.

References

[1] D.W. Aha and D. Kibler and M.K.Albert, Instance-based learning algorithms. Machine Learning **6**(1), (1991) 37–66

[2] M.C. Ainslie and J.S. Sánchez, Space partitioning for instance reduction in lazy learning algorithms. 2nd Workshop on Integration and Collaboration Aspects of Data Mining, Decision Support and Meta-Learning (2002) 13–18

[3] C.L. Chang, Finding prototypes for nearest neighbor classifiers. IEEE Trans. on Computers, **23** (1974) 1179–1184

[4] C.H. Chen and A. Józwik, A sample set condensation algorithm for the class sensitive artificial neural network. Pattern Recognition Letters, **17** (1996) 819–823

[5] B.V. Dasarathy, Nearest neighbor (NN) norms: NN pattern classification techniques. IEEE Computer Society Press, Los Alamitos, CA (1990)

[6] B.V. Dasarathy, Minimal consistent subset (MCS) identification for optimal nearest neighbor decision systems design. IEEE Trans. on Systems, Man and Cybernetics **24** (1994) 511-517

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

[8] R. Duda and P. Hart, Pattern classification and scene analysis. NY, J. Wiley (1991)

[9] M.A.T. Figueiredo and A.K. Jain, Unsupervised learning of finite mixture models. IEEE Trans. on Pattern Analysis Machine Intelligence **24** (2002) 381–396

[10] P. Hart, The condensed nearest neighbor rule. IEEE Trans. on Information Theory **14** (1968) 505–516

[11] T. Hastie and R. Tibshirani, Discriminant analysis by Gaussian mixtures. Journal of the Royal Statistical Society (B) **58** (1996) 155–176

[12] G. Hinton and P. Dayan and M. Revow, Modeling the manifolds of images of handwritten digits. IEEE Trans. on Neural Networks **8** (1997) 65–74

[13] T. Kohonen, Self-organizing maps. Springer-Verlag: Berlin Heidelberg, Germany (1995)

[14] L.I. Kuncheva and J.C. Bezdek, Presupervised and postsupervised prototype classifier design. IEEE Trans. on Neural Networks **10** (5) (1999) 1142–1152

[15] G. McLachlan and K. Basford, Mixture models: inference and application to clustering. New York: Marcel Dekker (1988)

[16] G. McLachlan and T. Krishnan, The EM algorithm and extensions. New York: John Wiley & Sons (1997)

[17] C.J. Merz and P.M. Murphy, UCI repository of machine learning DB. Dept. Information and Comp. Sci., U. California (1998). http://www.ics.uci.edu/˜mlearn.

[18] J. Novovicova and P. Pudil and J. Kittler, Divergence based feature selection for multimodal class densities. IEEE Trans. on Pattern Analysis and Machine Intelligence. **18** (1996) 218–223

[19] I. Tomek, Two modifications of CNN. IEEE Trans. on Systems, Man and Cybernetics **6** (1976) 769-772

[20] G.T. Toussaint and B.K. Bhattacharya and R.S. Poulsen, The application of Voronoi diagrams to non-parametric decision rules. Computer Science and Statistics: The Interface, L. Billard, Elsevier Science, North-Holland, Amsterdam (1985)

[21] D.R. Wilson and T.R. Martinez, Reduction techniques for instance-based learning algorithms. Machine Learning. **38** (2000) 257–286