Discriminative Gaussian Mixture Models: A Comparison with Kernel Classifiers

Aldebaro Klautau A.KLAUTAU@IEEE.ORG
Nikola Jevtić NIKOLA@CWC.UCSD.EDU
Alon Orlitsky ALON@ECE.UCSD.EDU
ECE Department, UCSD, 9500 Gilman Drive, La Jolla, CA 92093, USA

Abstract

We show that a classifier based on Gaussian mixture models (GMM) can be trained discriminatively to improve accuracy. We describe a training procedure based on the extended Baum-Welch algorithm used in speech recognition. We also compare the accuracy and degree of sparsity of the new discriminative GMM classifier with those of generative GMM classifiers, and of kernel classifiers, such as support vector machines (SVM) and relevance vector machines (RVM).

1. Introduction

GMMs have been applied in many classification tasks, e.g., (Hastie & Tibshirani, 1996). Conventionally, the expectation-maximization (EM) algorithm (Dempster et al., 1977) is used to train the mixtures, leading to a *generative* (see, e.g., Rubinstein & Hastie, 1997) classifier.

Recently, several new *discriminative* learning methods have been proposed. They include SVM (Cortes & Vapnik, 1995), proximal SVM (PSVM) (Fung & Mangasarian, 2001; Rifkin, 2002), RVM (Tipping, 2001), and informative vector machine (IVM) (Lawrence et al., 2002). These discriminative methods were successfully used in a variety of classification problems.

In this paper we derive an algorithm for discriminative training of GMMs based on the *extended* Baum-Welch algorithm, which is used to train hidden Markov models (HMM). It incorporates several speed-ups, and has a very low memory footprint, even for multiclass problems.

We also compare the accuracy and sparsity of the proposed classifier to those of the conventional, generative, GMM classifier, and of the four discriminative classifiers mentioned above when constrained to using Gaussian kernels. We show that the proposed discriminative GMM classifier (DGMM) achieves, for example, a result close to the Bayes error (14%) for the *waveform* dataset (Breiman et al., 1984), while the generative GMM, SVM, PSVM, RVM and IVM achieve 17.4, 15.5, 15.4, 16.0, 17.2%, respectively. In a vowel classification task (Klautau, 2002), DGMM achieves the best error using 20 Gaussians (with diagonal covariances), while the five classifiers listed above achieve their best result using 40 (Gaussians), 413, 599, 112 and 134 support vectors, respectively.

The paper is organized as follows. In Section 2 we establish the notation. In Section 3 we present DGMM as a special case of a Bayes classifier, and discuss a training procedure. Most of the material in Section 3 can be found in the speech recognition literature, but we present it in the classification framework to avoid issues specific to HMMs. In Section 4 we briefly describe the kernel classifiers that will be compared to DGMM. Section 5 presents the simulations results and is followed by our conclusions.

2. Classification

We deal with supervised classification problems, where one is given a *training set* $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$ containing N *examples*, which are independently and identically distributed (iid) samples from an unknown but fixed distribution $P(\mathbf{x}, y)$. Each example $(\mathbf{x}, y)$ consists of an instance $\mathbf{x} \in \mathcal{X}$ and a label $y \in \{1, \ldots, Y\}$. The input $\mathbf{x}$ is a vector of dimension K. If the input is discrete, we use x instead of $\mathbf{x}$. A *classifier* is a mapping $f : \mathcal{X} \to \{1, \ldots, Y\}$. Of special interest are binary classifiers, for which $Y = 2$, and for mathematical convenience, sometimes the labels are

Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003), Washington DC, 2003.

$y \in \{-1, 1\}$ (e.g., SVM) or $y \in \{0, 1\}$ (e.g., RVM).

2.1. Bayes classifiers

We adopt the nomenclature used in (Duda et al., 2001), where[1] $P(y|\mathrm{x})$, $P(\mathrm{x}|y)$, $P(y)$ and $P(\mathrm{x})$ are called *posterior*, *likelihood*, *prior* and *evidence*, respectively, and are related through Bayes' rule

$$P(y|\mathrm{x}) = \frac{P(\mathrm{x}|y)P(y)}{P(\mathrm{x})}. \tag{1}$$

Bayes classifiers attempt to select the label

$$\arg\max_{y=1,\dots,Y} P(\mathrm{x}|y)P(y),$$

which maximizes the posterior probability. However, neither $P(y)$, nor $P(\mathrm{x}|y)$ is known, hence the classifiers use estimates $\hat{P}(y)$ and $\hat{P}(\mathrm{x}|y)$ and maximize

$$f(\mathrm{x}) = \arg\max_{y=1,\dots,Y} \hat{P}(\mathrm{x}|y)\hat{P}(y). \tag{2}$$

We assume here that the prior $P(y)$ can be reliably estimated by counting the labels in the training set, i.e., $\hat{P}(y) = P(y)$. Estimating $\hat{P}(\mathrm{x}|y)$ is often the most difficult task. Hence, Bayes classifiers typically assume a parametric distribution $\hat{P}(\mathrm{x}|y) = \hat{P}_{\Theta_y}(\mathrm{x}|y)$ where Θ_y describes the distribution's parameters that need to be determined (e.g., mean and covariance matrix if the likelihood model is a Gaussian distribution).

If $\hat{P}(\mathrm{x}, y) = P(\mathrm{x}, y)$, the Bayes classifier achieves the optimal (Bayes) error (Duda et al., 2001). However, with limited data, one has to carefully choose the model assumed for the likelihood and the algorithm for their estimation. We are interested in two different ways of learning Bayes classifiers, which correspond to two ways of estimating Θ. They are described in the next subsection.

2.2. Generative vs. discriminative training

The conventional way of estimating Θ is through maximum likelihood estimation (MLE). MLE classifiers seek $\Theta_g = \arg\max_\Theta R_g(\Theta)$, where

$$R_g(\Theta) = \prod_{n=1}^{N} \hat{P}(\mathrm{x}_n|y_n),$$

Such classifiers are called *generative* (Ng & Jordan, 2002) or *informative* (Rubinstein & Hastie, 1997).

The term generative is used because if the estimated $\hat{P}(\mathrm{x}, y)$ is "close" to the true distribution $P(\mathrm{x}, y)$, we could use $\hat{P}(\mathrm{x}, y)$ to generate samples with statistics similar to the ones of our original training set. However, for the sake of classification, we do not need to keep Θ. For example, one cannot generate samples out of a *linear discriminant analysis* (LDA) classifier after simplifying the expressions (Rubinstein & Hastie, 1997) that define f. In such cases, the term informative seems more appropriate.

By contrast, discriminative Bayes classifiers seek $\Theta_d = \arg\max_\Theta R_d(\Theta)$, where

$$R_d(\Theta) = \prod_{n=1}^{N} \hat{P}(y|\mathrm{x}).$$

Note that

$$\begin{aligned} R_d(\Theta) &= \prod_{n=1}^{N} \frac{\hat{P}(\mathrm{x}_n|y_n)\hat{P}(y_n)}{\hat{P}(\mathrm{x}_n)} \\ &= \prod_{n=1}^{N} \left(1 + \frac{\sum_{j \neq y_n} \hat{P}(\mathrm{x}_n|j)\hat{P}(j)}{\hat{P}(\mathrm{x}_n|y_n)\hat{P}(y_n)}\right)^{-1}. \end{aligned}$$

It follows that discriminative procedures try not only to maximize the likelihood of examples (x, y), but, at the same time, minimize the likelihood of competing classes $j \neq y$. As for other generative-discriminative pairs of classifiers, training a discriminative Bayes classifier is harder than a generative. There are no closed-form solutions and iterative optimization algorithms are needed.

Sometimes a generative classifier is more appropriate, specially when the likelihood model is correct and $\hat{P}(y) = P(y)$. However, in many practical cases, the discriminative classifier performs better (Nádas et al., 1988; Rubinstein & Hastie, 1997) unless training data is scarce (Ng & Jordan, 2002).

3. DGMM

GMM classifiers adopt a mixture of M_y multivariate Gaussians $\mathcal{N}(\mathrm{x}|\mu, \Sigma)$ for modeling class y, namely

$$\hat{P}(\mathrm{x}|y) = \sum_{m=1}^{M_y} w_{ym} \mathcal{N}(\mathrm{x}|\mu_{ym}, \Sigma_{ym}). \tag{3}$$

In this work we assume diagonal covariance matrices Σ_{ym} for computational reasons. We call GMM the classifier that uses MLE to learn the mixtures, and DGMM the one that adopts Eq. (3) as likelihood model and is trained through maximizing $R_d(\Theta)$.

We note that other likelihood models have been adopted for generative classifiers. Assuming $\hat{P}(\mathrm{x}|y) =$

[1] We use P to denote both probability mass functions and densities.

$\mathcal{N}(\mathbf{x}|\mu_y, \Sigma)$ has only one Gaussian, where Σ is a full covariance matrix shared by all Y likelihoods leads to the LDA classifier (homoscedastic model) (McLachlan, 1992). If the (full) covariance matrices are different, namely, $\hat{P}(\mathbf{x}|y) = \mathcal{N}(\mathbf{x}|\mu_y, \Sigma_y)$, one has the quadratic DA (QDA) classifier (heteroscedastic model). Restricting QDA to use diagonal covariance matrices corresponds to the so-called naïve Bayes classifier. The mixture DA (MDA) classifier (Hastie & Tibshirani, 1996) adopts mixtures of Gaussians with Σ being a full matrix that is shared by all distributions.

The next subsections describe DGMM training. For historical reasons, we start by making a connection to HMM-based speech recognition. Then, the last two subsections discuss an algorithm for discriminatively reestimating discrete distributions and its modification to support GMMs.

3.1. MMIE

A speech recognition system is typically based on HMMs. These systems do not fall in our definition of classifier, especially because their input $\mathbf{x}$ (associated with, e.g, a phrase or word) has variable-length. But if we assume that a class y can be represented by an HMM that provides $P(\mathbf{x}|y)$, most of our discussion in this section is valid for both HMM-based systems and classifiers f.

Estimating Θ_d is known in speech recognition as *maximum mutual information estimation* (MMIE) because, assuming the input $x \in \{1, \ldots, X\}$ is discrete, the conditional entropy of label $\mathbf{Y}$ given $\mathbf{X}$ is

$$H(\mathbf{Y}|\mathbf{X}) = -\mathbf{E}[\log P(y|x)] = -\sum_{x,y} P(x,y) \log P(y|x).$$

As we do not know $P(x, y)$, we use the sample average

$$\hat{H}(\mathbf{Y}|\mathbf{X}) = -\frac{1}{N} \sum_{n=1}^{N} \log \hat{P}(y_n|x_n) = -\frac{1}{N} \log R_d(\Theta).$$

By maximizing $R_d(\Theta)$, we minimize $\hat{H}(\mathbf{Y}|\mathbf{X})$ and, consequently, maximize the estimated mutual information

$$\hat{I}(\mathbf{X}; \mathbf{Y}) = \hat{H}(\mathbf{Y}) - \hat{H}(\mathbf{Y}|\mathbf{X})$$

because $\hat{H}(\mathbf{Y})$ is fixed.

We note that MMIE is strongly related to other *maximum conditional likelihood* estimation techniques, e.g., (Nádas et al., 1988; Jebara & Pentland, 1998).

MMIE was developed by the IBM speech group (Bahl et al., 1986). More precisely, according to (Nádas et al., 1988), MMIE was proposed by Robert Mercer. In speech recognition, the most popular algorithm for MMIE is the extended Baum-Welch, e.g., (Woodland & Povey, 2002), which has outperformed some standard optimization methods (Valtchev, 1995). The extended Baum-Welch algorithm was proposed in (Gopalakrishnan et al., 1991) for reestimating discrete HMMs, and extended in (Normandin, 1991) to continuous HMMs with mixtures of Gaussians as output distributions.

3.2. Extended Baum-Welch

Using EM to learn a GMM classifier is equivalent to using Baum-Welch for HMMs with only one state, where the α and β variables (Rabiner, 1989) are not needed. Similarly, our method for training DGMM corresponds to applying the extended Baum-Welch algorithm to GMM classifiers, and we call it *discriminative* EM (DEM). The base of both DEM and extended Baum-Welch algorithms is the following theorem.

Theorem 1 (Baum & Eagon, 1967) Let $S(\Theta) = S(\{\theta_{ji}\})$ be a homogeneous degree d polynomial with nonnegative coefficients. Let $\overline{\Theta} = \{\overline{\theta}_{ji}\}$ be any point of the domain $\mathcal{D} : \theta_{ji} \geq 0$, $\sum_{i=1}^{q_j} \theta_{ji} = 1$, $j = 1, \ldots, p$ such that, $\forall j$,

$$\sum_{i=1}^{q_j} \left(\overline{\theta}_{ji} \frac{\partial S}{\partial \theta_{ji}} (\overline{\Theta}) \right) \neq 0,$$

where $\frac{\partial S}{\partial \theta}(\overline{\Theta})$ denotes the value of $\frac{\partial S(\theta)}{\partial \theta}$ at $\overline{\Theta}$. Let $\tilde{\Theta} = T(\overline{\Theta}) = T(\{\overline{\theta}_{ji}\})$ denote the point of $\mathcal{D}$ whose j, i coordinate is

$$\tilde{\theta}_{ji} = \frac{\overline{\theta}_{ji} \frac{\partial S}{\partial \theta_{ji}} (\overline{\Theta})}{\sum_{i=1}^{q_j} \left(\overline{\theta}_{ji} \frac{\partial S}{\partial \theta_{ji}} (\overline{\Theta}) \right)}. \tag{4}$$

Then, $S(T(\overline{\Theta})) > S(\overline{\Theta})$ unless $T(\overline{\Theta}) = \overline{\Theta}$. □

The denominator in Eq. (4) guarantees the *growth transformation* T leads to distributions. We illustrate the use of Theorem 1 with the following simple example.

Example 1 Assume the likelihood of class y is a probability mass distribution over $\{1, \ldots, X\}$ given by $\hat{P}(x|y) = \theta_{yx}$, with $\sum_1^X \theta_{yi} = 1, \forall y$. Let N_{ji} be the number of occurrences of example (i, j), such that

$$R_g(\Theta) = \prod_{j=1}^{Y} \prod_{i=1}^{X} {\theta_{ji}}^{N_{ji}} \quad \text{and} \quad \frac{\partial R_g}{\partial \theta_{ji}} (\overline{\Theta}) = R_g(\overline{\Theta}) \frac{N_{ji}}{\overline{\theta}_{ji}}.$$

Applying Theorem 1 to maximize the polynomial $S(\Theta) = R_g(\Theta)$ leads to the MLE solution

$$\tilde{\theta}_{ji} = \frac{N_{ji}}{\sum_{i=1}^{X} N_{ji}}$$

Initialization: pick $\overline{\Theta}$
Repeat until convergence:

- Calculate the value of $R_d(\overline{\Theta})$
- Create the polynomials

$$\begin{aligned} Q(\Theta) &= \mathrm{Num}(\Theta) - R_d(\overline{\Theta})\mathrm{Den}(\Theta) \\ S(\Theta) &= Q(\Theta) + C(\Theta), \end{aligned}$$

  where

$$C(\Theta) = c\left(\sum_{j,i}\theta_{ji} + 1\right)^d,$$

  c is the smallest magnitude that cancels all the negative coefficient of $Q(\Theta)$ ($c = 0$ if no negative coefficient exists) and d is the degree of $Q(\Theta)$.

- Use Theorem 1 on $S(\Theta)$ to find $\tilde{\Theta} = T(\overline{\Theta})$
- Update for next iteration $\overline{\Theta} \leftarrow \tilde{\Theta}$

Figure 1. Algorithm proposed in (Gopalakrishnan et al., 1991).

in the first iteration, independently of $\overline{\Theta}$. □

Note that we want to use T to iteratively maximize R_d, which is not a polynomial, but a rational function

$$R_d(\Theta) = \frac{\mathrm{Num}(\Theta)}{\mathrm{Den}(\Theta)}.$$

Gopalakrishnan et al. (1991) studied how to apply the growth transformation T to rational functions. As an intermediate result, they showed that Theorem 1 also applies when $S(\Theta)$ is inhomogeneous. Then, they proved that the algorithm in Figure 1 locally maximizes R_d.

To understand the role of the polynomials Q and C, let us consider that we could apply the transformation T of Theorem 1 to Q, obtaining $Q(\hat{\Theta}) > Q(\overline{\Theta})$. This may not be possible because $Q(\Theta)$ can have negative coefficients, which would violate an assumption in Theorem 1. Ignoring this for a moment, we note that $Q(\overline{\Theta}) = 0$, because $Q(\Theta)$ is defined using $R_d(\overline{\Theta})$. Hence, if $Q(\hat{\Theta}) > Q(\overline{\Theta})$, we have $Q(\hat{\Theta}) = \mathrm{Num}(\hat{\Theta}) - R_d(\overline{\Theta})\mathrm{Den}(\hat{\Theta}) > 0$ and $R_d(\hat{\Theta}) > R_d(\overline{\Theta})$, as desired. To overcome the negative coefficients of Q, we use the polynomial C. Every possible monomial of $Q(\Theta)$ occurs in $C(\Theta)$ and the coefficients of $S(\Theta) = Q(\Theta) + C(\Theta)$ are nonnegative. Note that $\sum_{j,i}\theta_{ji} = Y$ and $C(\Theta)$ is effectively a constant that is independent of θ_{ji}. Therefore, the values $\{\theta_{ji}\}$ that maximize $S(\Theta)$ also maximize $Q(\Theta)$. In fact, (Gopalakrishnan et al., 1991) suggest for c the largest negative coefficient magnitude in $Q(\Theta)$, which is an upper-bound for canceling negative terms. We use the following example to illustrate this algorithm.

Example 2 Assume a Bayes classifier with the same likelihood as in Example 1, but trained according to the algorithm of (Gopalakrishnan et al., 1991). Also, assume that $X = Y = 2$ and let $N_{ji} = \mathbf{M}(j, i)$, where $\mathbf{M} = \begin{bmatrix} 4 & 5 \\ 6 & 2 \end{bmatrix}$. We use π_y for prior probability $P(y)$. In this case,

$$R_d(\Theta) = \frac{k\theta_{11}^4\theta_{12}^5\theta_{21}^6\theta_{22}^2}{(\pi_1\theta_{11} + \pi_2\theta_{21})^{10}(\pi_1\theta_{12} + \pi_2\theta_{22})^7},$$

where $k = \pi_1^9\pi_2^8 \approx 8 \times 10^{-6}$. At each iteration we form

$$\begin{aligned} S(\Theta) \;=\; & C(\Theta) + k\theta_{11}^4\theta_{12}^5\theta_{21}^6\theta_{22}^2 - \\ & R_d(\overline{\Theta})\left((\pi_1\theta_{11} + \pi_2\theta_{21})^{10}(\pi_1\theta_{12} + \pi_2\theta_{22})^7\right). \end{aligned}$$

In this example, $Q(\Theta)$ is homogenous, so we can use $C(\Theta) = c\left(\sum_{i,j}\theta_{ij}\right)^{17}$, and the optimal constant is given by $c = R_d(\overline{\Theta})\frac{(\max_j \pi_j)^{17}}{\binom{17}{10}}$. It is interesting that the optimal constant is easy to obtain in this example, as opposed to the magnitude of the highest negative coefficient in $Q(\Theta)$ which would require numerical optimization.

In this case, if we estimate the priors π_y from the training set using MLE, the same set of parameters Θ maximizes both R_g and R_d because (if the priors are estimated with MLE) maximizing both R_d and R_g lead to the MLE of the joint distribution $P(x, y)$. □

Note that we can use

$$\frac{\partial S(\Theta)}{\partial\theta_{ji}} = \frac{\partial\mathrm{Num}(\Theta)}{\partial\theta_{ji}} - R_d(\overline{\Theta})\frac{\partial\mathrm{Den}(\Theta)}{\partial\theta_{ji}} + D',$$

and the algorithm still has guaranteed convergence as long as $D' \geq cd(Y + 1)^{d-1}$. Also note that

$$\frac{\partial\log R_d}{\partial\theta_{ji}}(\overline{\Theta}) = \frac{1}{\mathrm{Num}(\overline{\Theta})}\left(\frac{\partial\mathrm{Num}}{\partial\theta_{ji}}(\overline{\Theta}) - R_d(\overline{\Theta})\frac{\partial\mathrm{Den}}{\partial\theta_{ji}}(\overline{\Theta})\right),$$

hence

$$\frac{\partial S(\Theta)}{\partial\theta_{ji}} = \mathrm{Num}(\overline{\Theta})\left(\frac{\partial\log R_d}{\partial\theta_{ji}}(\overline{\Theta}) + \frac{D'}{\mathrm{Num}(\overline{\Theta})}\right). \quad (5)$$

In practical implementations, we want to adopt some value $D < D'/\mathrm{Num}(\overline{\Theta})$ for faster (but not guaranteed) convergence.[2] Hence, we can choose an empirical value D according to some heuristic, and substituting Eq. (5) in Eq. (4) (the term $\mathrm{Num}(\overline{\Theta})$ cancels) we get

$$\tilde{\theta}_{ji} = \frac{\overline{\theta}_{ji}\left(\frac{\partial \log R_d}{\partial \theta_{ji}}(\overline{\Theta}) + D\right)}{\sum_{i=1}^{q_j}\left(\overline{\theta}_{ji}\frac{\partial \log R_d}{\partial \theta_{ji}}(\overline{\Theta})\right) + D}, \qquad (6)$$

where D should be large enough for convergence. One can observe from this equation that, the greater D, the less the new parameters will differ from $\overline{\Theta}$, which implies in more iterations (i.e., slow convergence).

3.3. Extension to mixture of Gaussians

In this subsection we assume the likelihoods are mixtures of Gaussians with diagonal covariance matrices. We have to learn the parameters $\Theta = \{\mu_{jmk}, \sigma_{jmk}, w_{jm}\}$, where $|\Theta| = 2M(K+1)$. The indices (j, m, k) specify the class, the mixture component and the dimension, respectively. It is convenient to create, for each iteration of the algorithm, a model that would output the evidence $\hat{P}(\mathbf{x})$ of Eq. (1), which is associated with the denominator of R_d. In other words, we conceptually treat the marginal distribution

$$P(\mathbf{x}) = \sum_{y=1}^{Y} P(\mathbf{x}|y)P(y)$$

as a *denominator* model. Here it has a mixture with all $M = \sum_{y=1}^{Y} M_y$ Gaussians, namely

$$P(\mathbf{x}) = \sum_{y=1}^{Y}\sum_{m=1}^{M_y} \mathcal{N}(\mathbf{x}|\mu_{ym}, \Sigma_y)w'_{ym},$$

where $w'_{ym} = P(y)w_{ym}$ is the original weight scaled by the associated prior.

We define the notation for this subsection by briefly describing the EM algorithm, as applied to GMM estimation. For each class j, the "expectation" step of EM passes through all N_j examples of this class, calculating

$$P_j(m|\mathbf{x}_n) = \frac{\mathcal{N}(\mathbf{x}_n|\mu_{ym}, \Sigma_y)w_{jm}}{\sum_{m=1}^{M_j}\mathcal{N}(\mathbf{x}_n|\mu_{ym}, \Sigma_y)w_{jm}},$$

and accumulating the intermediate results needed to compute the *occupation counts* $\Lambda = \{\nu_{jmk}, \zeta_{jmk}, \gamma_{jm}\}$, where $|\Lambda| = |\Theta|$,

$$\nu_{jmk} = \sum_{n:y_n=j} P_j(m|\mathbf{x}_n)\mathbf{x}_n(k),$$

$$\zeta_{jmk} = \sum_{n:y_n=j} P_j(m|\mathbf{x}_n)\mathbf{x}_n^2(k),$$

$$\gamma_{jm} = \sum_{n:y_n=j} P_j(m|\mathbf{x}_n),$$

and $\mathbf{x}(k)$ is the k-th component of $\mathbf{x}$. These counts allow for calculating a new set of parameters in the "maximization" step as, e.g.:

$$\tilde{w}_{jm} = \frac{\gamma_{jm}}{\sum_{m=1}^{M_j}\gamma_{jm}} \quad\text{and}\quad \tilde{\mu}_{jmk} = \frac{\nu_{jmk}}{\gamma_{jm}}.$$

The DEM algorithm uses two set of occupation counts: $\Lambda^{num} = \{\nu_{jmk}^{num}, \zeta_{jmk}^{num}, \gamma_{jm}^{num}\}$ and $\Lambda^{den} = \{\nu_{jmk}^{den}, \zeta_{jmk}^{den}, \gamma_{jm}^{den}\}$. In the expectation step, each example is used to update the counts in Λ^{den}, independent of the value of y_n, and the counts in Λ^{num} that correspond to the correct class y_n. In other words, we compute $P(m|\mathbf{x}_n)$ for the denominator model using all M Gaussians, while $P_{y_n}(m|\mathbf{x}_n)$ associated with the numerator is computed using only the Gaussians of the correct class y_n. Hence, in the end of each epoch, the counts in Λ^{num} are exactly the same as the ones that would be obtained with EM.

In the maximization step, the component weights are reestimated according to Eq. (6), namely

$$\tilde{w}_{jm} = \frac{\gamma_{jm}^{num} - \gamma_{jm}^{den} + Dw_{jmk}}{\sum_{m'=1}^{M_j}\left(\gamma_{jm'}^{num} - \gamma_{jm'}^{den}\right) + D}.$$

The means and variances are reestimated according to the equations derived in (Normandin, 1991):

$$\tilde{\mu}_{jmk} = \frac{\nu_{jmk}^{\mathrm{num}} - \nu_{jmk}^{\mathrm{den}} + D\overline{\mu}_{jmk}}{\gamma_{jm}^{\mathrm{num}} - \gamma_{jm}^{\mathrm{den}} + D}$$

and

$$\tilde{\sigma}_{jmk}^2 = \frac{\zeta_{jmk}^{\mathrm{num}} - \zeta_{jmk}^{\mathrm{den}} + D(\overline{\sigma}_{jmk}^2 + \overline{\mu}_{jmk}^2)}{\gamma_{jm}^{\mathrm{num}} - \gamma_{jm}^{\mathrm{den}} + D} - \tilde{\mu}_{jm}^2.$$

In our implementation we use a different D_{jm} for each Gaussian, which depend on a learning rate η that varies over time. We also reduce training time by using a K-d tree to avoid calculating the value of all M Gaussians for each training example. More details can be found in (Klautau, 2003) and Java code is available at *www.deec.ufpa.br/aldebaro/dgmm.*

[2] In Example 2, the smallest constant that guaranteed convergence was $D = \frac{17(9/17)^{17}2^{16}}{\binom{17}{10}\mathrm{Den}(\overline{\Theta})}$, which is a relatively large number. If we start the estimation from a random point, $\mathrm{Den}(\overline{\Theta})$ is going to be a poor description of the data, and D will be large, slowing the convergence.

4. Kernel classifiers

Here we briefly describe the kernel classifiers used in the experiments in Section 5. We note that GMM can be seen as a "kernel" method (see, e.g., page 188 in Hastie et al., 2001). However, by kernel classifier we mean the ones obtained through kernel learning, as defined, e.g., in (Scholkopf & Smola, 2002).

We assume a Gaussian radial-basis function (RBF) kernel

$$\mathcal{K}(\mathbf{x}, \mathbf{x}') = e^{-\gamma \|\mathbf{x}-\mathbf{x}'\|^2},$$

which allows for a more direct comparison with DGMM. Two of the kernel classifiers (RVM and IVM) that we consider are based on Bayesian learning, while the others (SVM and PSVM) are non-probabilistic.

4.1. Non-probabilistic kernel classifiers

SVM and other kernel methods can be related to regularized function estimation in a reproducing kernel Hilbert space (RKHS) (Tikhonov & Arsenin, 1977). One wants to find the function f that minimizes

$$\frac{1}{N}\sum_{n=1}^{N} L(f(\mathbf{x}_n), y_n) + \lambda \|f\|^2_{\mathcal{H}_\mathcal{K}} \qquad (7)$$

where $\mathcal{H}_\mathcal{K}$ is the RKHS generated by the kernel $\mathcal{K}$, $f = h + b$, $h \in \mathcal{H}_\mathcal{K}$, $b \in \mathbb{R}$ and $L(f(\mathbf{x}_n), y_n)$ is a loss function. The solution to this problem, as given by the *representer* theorem (Kimeldorf & Wahba, 1971), is

$$f(\mathbf{x}) = \sum_{n=1}^{N} \omega_n \mathcal{K}(\mathbf{x}, \mathbf{x}_n) + b. \qquad (8)$$

This expression indicates that SVM and related classifiers are *example-based* (Scholkopf & Smola, 2002). In other words, assuming a Gaussian kernel, the mean of a Gaussian is restricted to be a training example $\mathbf{x}_n$.

Some examples $\mathbf{x}_n$ may not be used (e.g., the learning procedure may have assigned $\omega_n = 0$). We call *support vectors* the examples that are actually used in the final solution, even for classifiers other than SVM. For saving computations in the test stage, it is convenient to learn a sparse f, with few support vectors. We can obtain sparse solutions only if the loss function L is zero over an interval (see, e.g., problem 4.6 in Scholkopf & Smola, 2002). SVM achieves a sparse solution (at some degree) by choosing the loss

$$L(f(\mathbf{x}_n), y_n) = (1 - f(\mathbf{x}_n) y_n)_+,$$

where $(z)_+ = \max\{0, z\}$. The PSVM classifier is obtained by choosing

$$L(f(\mathbf{x}_n), y_n) = (f(\mathbf{x}_n) - y_n)^2$$

and, consequently, ends up using all the training set as support vectors. Motivated by the discussion in (Rifkin, 2002), hereafter we refer to PSVM as *regularized least-square classifier* (RLSC).

4.2. Probabilistic kernel classifiers

Both RVM and IVM are sparse Bayesian learning methods that can be related to the regularized risk functional of Eq. (7) (see, e.g., Scholkopf & Smola, 2002). They adopt a model for the posterior given by $g(y_n f(\mathbf{x}_n))$, where f is given by Eq (8) and g is a sigmoid link function that converts scores into probabilities.

The framework described in (Tipping, 2001) allows for, e.g., solving multiclass problems and estimating the variances of each Gaussian. However, the computational cost is very high. IVM is an alternative with a fast training procedure.

4.3. DGMM vs. kernel classifiers

Here, we briefly discuss some characteristics of DGMM and kernel classifiers. Kernel classifiers do not require that the coefficients ω lead to distributions, as DGMM. But the Gaussians of kernel classifiers are restricted to share the same spherical covariance matrix $\Sigma = \gamma^{-1}\mathbf{I}$.

DGMM, as RVM and IVM, provides an indication of class membership probability. Besides, it can deal with multiclass problems, while the other classifiers are either limited to binary problems by construction, or have to decompose the multiclass into binary problems due to the high computational cost of solving the multiclass case directly (Tipping, 2001).

The training time of DGMM scales with $\mathcal{O}(NM)$, where M is the number of Gaussians (note the actual time depends on the number of iterations), while for SVM, RLSC, RVM and IVM, it scales with $\mathcal{O}(N^2)$, $\mathcal{O}(N^3)$, $\mathcal{O}(N^3)$ and $\mathcal{O}(NM^2)$, respectively. DGMM has the smallest memory footprint (assuming SVM uses a cache of reasonable size), but IVM are often the fastest to train. Note that computing the value of a Gaussian with diagonal covariance matrix for DGMM, takes K more multiplications than computing the kernel.

In terms of model selection, γ must be selected for all kernel classifiers. Unless there are convergence problems, the specification of the kernel is all that RVM requires. Both DGMM and IVM require the number of Gaussians to be specified. IVM also requires selecting the bias b and a scalar to be added to the the kernel matrix (Lawrence et al., 2002). For DGMM, we also select a floor value for the variances, which can be seen

Name	train	test	classes (Y)	attributes (K)
synth	250	1000	2	2
waveform	400	4600	3	40
pima	200	332	2	7
pbvowel	599	600	10	2
0-6 (mnist)	11841	1938	2	256
7-9 (mnist)	12214	2037	2	256
d-t (timit)	6380	300	2	118
iy-ih (timit)	8874	446	2	118

Table 1. Description of the datasets.

	DGMM	SVM	IVM
0-6	0 / 1.0 / 80	0.02 / 0.8 / 734	0.4 / 0.6 / 200
7-9	0.8 / 2.7 / 30	0 / 1.8 / 2130	0.2 / 2.0 / 1000
d-t	12.9 / 11.0 / 4	12.1 / 13.3 / 2268	9.9 / 13.3 / 700
iy-ih	9.4 / 10.1 / 4	6.5 / 7.8 / 2087	6.9 / 11.9 / 800

Table 2. Results for the four largest datasets. The three entries are training error, test error and number of Gaussians (support vectors).

as a crude regularization procedure. SVM requires selecting the constant C. In the next section we report the results achieved by these classifiers.

5. Simulation results

We evaluated the performance of different classifiers using the eight standard datasets listed in Table 1. The datasets *pima* and *synth* were made available by B. Ripley[3]. The waveform dataset is described in (Breiman et al., 1984). The *pbvowel* dataset corresponds to a version of the Peterson and Barney's vowel data described in (Klautau, 2002). Two binary problems (digits 0 vs. 6 and 7 vs. 9) were extracted[4] from MNIST, which is a dataset of handwritten digits available from Y. LeCun. Two other binary problems (phones d vs. t and iy vs. ih) were extracted from the TIMIT speech dataset[5]. We converted each occurrence of these phones into fixed-length vectors ($K = 118$) using a linear warping procedure (see, Klautau, 2003). The datasets synth and waveform are toy examples, for which we know the Bayes errors are 8% and 14%, respectively.

For the two multiclass datasets, we used all-pairs ECOC with Hamming decoding (Allwein et al., 2000). For classifiers other than GMM and DGMM, we standardized each attribute to have mean 0 and variance 1. We use a simple k-means algorithm to initialize DGMM with the same number of Gaussians per class, while other methods could give better performance (Normandin, 1995).

All parameters are selected using ten-fold cross-validation on the training set. Instead of searching over a fixed range (as for the other parameters), γ and, for SVM, C were selected as follows. We chose γ by running We start with $\gamma = 1$, then increase (or decrease) γ by multiplying by 2 (or 0.5) until we do not get improvements for three consecutive times (we initialize the search with different γ if it does not converge). For SVM, after choosing γ with $C = 1$, we optimize C using the same search procedure. This model selection procedure is computationally intensive (especially for RVM). Because of that, we used the four largest datasets only with DGMM, SVM and IVM.

Table 2 shows the results for the full versions of the four largest (and binary) datasets. In these experiments, the maximum number of Gaussians for IVM and DGMM was set to 1000. Note that these are binary, while multiclass problems can favor DGMM.

We created reduced versions of the largest datasets in Table 2, by keeping only 500 examples of each. These smaller versions are called s0-6, s0-7, sd-t and siy-ih. Table 3 shows the results for these and the other small datasets. In this case, the maximum number of Gaussians for IVM and DGMM was set to 100.

These preliminary results show that DGMM outperforms GMM in terms of accuracy and can be competitive with kernel classifiers.

6. Conclusions

We described DGMM and a training method, based on the extended Baum-Welch algorithm. The experimental results indicate that DGMM is competitive with kernel classifiers in terms of accuracy and sparsity. DGMM can deal with multiclass problems and provides a probabilistic output.

We note that the DEM algorithm used to train DGMMs can substitute EM in finding Gaussians for RBF networks. Preliminary experiments showed that this can improve the accuracy of RBF networks but did not bring improvements over DGMM.

Future work includes a thorough evaluation of multiclass problems and an investigation of a principled way to perform model selection and regularization for DGMM.

[3] http://www.stats.ox.ac.uk/pub/PRNN.

[4] We subsampled MNIST to obtain images with 16 x 16 pixels using Matlab's function imresize.

[5] http://www.ldc.upenn.edu/.

References

Allwein, E., Schapire, R., & Singer, Y. (2000). Reducing multiclass to binary: A unifying approach for margin classifiers. *Journal of Machine Learning Research*, 113–

	GMM	DGMM	SVM	RLSC	RVM	IVM
synth	11.6 / 8.2 / 4	11.8 / 8.1 / 4	10.8 / 8.9 / 119	14 / 10.3 / -	10.0 / 10.3 / 18	12.4 / 9.2 / 54
waveform	13.2 / 17.4 / 6	9.2 / 14.4 / 30	7.4 / 15.5 / 336	0.8 / 15.4 / -	7.8 / 16.0 / 7	3.6 / 17.2 / 100
pima	23.5 / 21.4 / 8	4.5 / 28.6 / 48	22 / 20.8 / 132	13.5 / 23.5 / -	23.0 / 20.48 / 3	21.5 / 21.1 / 82
s0-6	0 / 2.4 / 56	0 / 2.4 / 56	0 / 2.4 / 132	0 / 2.6 / -	0 / 1.96 / 233	0.2 / 1.3 / 56
s7-9	0.8 / 9.5 / 68	0 / 8.9 / 32	0.8 / 5.0 / 185	1.0 / 6.2 / -	0.59 / 6.73 / 30	4.1 / 7.2 / 96
sd-t	8.8 / 17.3 / 8	2.1 / 16.7 / 22	10.6 / 17.3 / 251	0 / 20 / -	10.59 / 17.0 / 19	13.0 / 19.3 / 92
siy-ih	8.3 / 17.0 / 8	3.2 / 12.6 / 16	5.9 / 11.4 / 254	8.3 / 14.6 / -	0.81 / 15.02 / 27	4.1 / 12.1 / 96
pbvowel	16.7 / 21.7 / 40	16.7 / 18.7 / 20	15.5 / 20.3 / 413	17.4 / 23 / -	17.03 / 18.83 / 112	24.5 / 27.2 / 134

Table 3. Results for small datasets (less than 600 training examples). The three entries are training error, test error and number of Gaussians (support vectors). For multiclass datasets and kernel classifiers, we count the number of distinct support vectors. RLSC uses all training set as support vectors.

141.

Bahl, L., Brown, P., de Souza, P., & (1986)., R. M. (1986). Maximum mutual information estimation of hidden Markov model parameters for speech recognition. *ICASSP* (pp. 49-52).

Baum, L., & Eagon, J. (1967). An inequality with applications to statistical estimation for probabilistic functions of Markov processes and to a model for ecology. *Bulletin of the AMS*, *73*, 360-363.

Breiman, L., Friedman, J., Olshen, R., & Stone, C. (1984). *Classification and regression trees.* Wadsworth.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, *20*, 273-297.

Dempster, A., Laird, N., & Rubin, D. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society (B)*, *39*, pp. 1-22.

Duda, R., Hart, P., & Stork, D. (2001). *Pattern classification.* Wiley.

Fung, G., & Mangasarian, O. (2001). Proximal support vector classifiers. *KDD* (pp. 77-86).

Gopalakrishnan, P., Kanevsky, D., Nádas, A., & Nahamoo, D. (1991). An inequality for rational functions with applications to some statistical estimation problems. *IEEE Transactions on Information Theory*, *37*, 107-113.

Hastie, T., & Tibshirani, R. (1996). Discriminant analysis by Gaussian mixtures. *Journal of the Royal Statistical Society series B*, *58*, 158-176.

Hastie, T., Tibshirani, R., & Friedman, J. (2001). *The elements of statistical learning.* Springer Verlag.

Jebara, T., & Pentland, A. (1998). Maximum conditional likelihood via bound maximization and the cem algorithm. *Neural Information Processing Systems 11.*

Kimeldorf, G., & Wahba, G. (1971). Some results on Tchebychean spline functions. *J. Math. Anal. Applic.*, *33*, 82-95.

Klautau, A. (2002). *Classification of Peterson and Barney's vowels using Weka* (Technical Report). UFPA, *http://www.deec.ufpa.br/tr.*

Klautau, A. (2003). *Speech recognition using discriminative classifiers.* Doctoral dissertation, UCSD.

Lawrence, N., Seeger, M., & Herbrich, R. (2002). Fast sparse Gaussian process methods: The informative vector machine. *Neural Information Processing Systems 15.*

McLachlan, G. (1992). *Discriminant analysis and statistical pattern recognition.* Wiley.

Nádas, A., Nahamoo, D., & Picheny, M. (1988). On a model-robust training method for speech recognition. *IEEE Trans. on ASSP*, *36*, 1432-6.

Ng, A., & Jordan, M. (2002). On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. *NIPS.*

Normandin, Y. (1991). *Hidden Markov models, maximum mutual information estimation and the speech recognition problem.* Doctoral dissertation, McGill University.

Normandin, Y. (1995). Optimal splitting of HMM Gaussian mixture components with MMIE training. *ICASSP* (pp. 449-52).

Rabiner, L. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, *77*, 257-86.

Rifkin, R. (2002). *Everything old is new again: A fresh look at historical approaches in machine learning.* Doctoral dissertation, MIT.

Rubinstein, Y., & Hastie, T. (1997). Discriminative vs informative learning. *Knowledge Discovery and Data Mining* (pp. 49-53).

Scholkopf, B., & Smola, A. (2002). *Learning with kernels.* MIT Press.

Tikhonov, A., & Arsenin, V. (1977). *Solutions of ill-posed problems.* Winston.

Tipping, M. (2001). Sparse bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, *1*, 211-244.

Valtchev, V. (1995). *Discriminative methods in HMM-based speech recognition.* Doctoral dissertation, Cambridge University.

Woodland, P., & Povey, D. (2002). Large scale discriminative training of hidden Markov models for speech recognition. *Computer Speech and Language*, *16*, 25-47.