Classification with Multiple Latent Variable Models using Maximum Entropy Discrimination

Machiel Westerdijk MACHIEL@MBFYS.KUN.NL
Wim Wiegerinck WIMW@MBFYS.KUN.NL
Foundation for Neural Networks, University of Nijmegen, P.O. Box 9101, 6500 HB Nijmegen, The Netherlands

Abstract

In this paper we propose a method to use multiple generative models for classification tasks. The standard approach to use generative models for classification is to train a separate model for each class. A novel data point is then classified by the model that gives it the highest probability. The algorithm we propose alters the parameters of the models to improve the classification accuracy. The algorithm is based on the maximum entropy discrimination method, recently introduced by T. Jaakkola. Our approach is made computationally tractable by assuming that each of the models is deterministic, by which we mean that a data-point is associated to only a single latent or hidden state. The resulting algorithm is an interesting variant of the support vector machine learning algorithm. We show and compare test results on a handwritten digit recognition problem.

1. Introduction

Probabilistic graphical models, such as hidden Markov models (Baum et al., 1970) , sigmoid belief networks (Neal, 1991) and hierarchical mixtures of experts (Jordan & Jacobs, 1994) are excellently suited to discover hidden structures in complex data. In particular if prior knowledge is available about the sources that generate the observed data, such *generative models* can provide a compact representation of data that reflects its underlying mechanisms. However, in their standard use, generative models are less suited for classification tasks, since they have been optimized for likelihood estimation, and are not optimally tuned to model the classification boundary. On the other hand, models that are optimized for classification, such as support vector machines (Vapnik, 1995), often operate like black-boxes and lack insight into underlying generative mechanisms.

Recently, however, Jaakkola et al. (1999) proposed a general framework in which they tune a given set of probabilistic models such that classification error on training data is minimized. The method is based on the maximum entropy principle. As a special case, this framework includes support vector machines.

However, application of this framework on a set of probabilistic generative models with a layer of latent variables is computationally intractable. Variational techniques which can be used to optimize the log-likelihood of such models are not applicable, since in this scheme they do not provide a lower bound of the objective function any more. In this paper we deal with this problem by using *deterministic* generative models, which have been introduced recently (Westerdijk et al., 1999b; Westerdijk et al., 1999a). In these models, the layer of latent variables is represented by a single state, which simplifies the framework considerably, while the principle of hidden explanatory features of the data is retained.

The paper is organized as follows. In Section 2 we briefly review Jaakkola's framework. In Section 3 we review the deterministic generative vector quantization (GVQ) models. In Section 4 we describe how Jaakkola's framework can be applied to GVQ models. In addition we describe the incremental algorithm to optimize the GVQ models. In Section 5 we describe simulation results. We end with a discussion in Section 6.

2. Maximum Entropy Discrimination

In this section we briefly discuss the maximum entropy classification framework which was originally formulated in Jaakkola et al. (1999). As in the original paper, we will restrict ourselves to two-class classification problems with class labels $c \in \{-1, 1\}$.

Assume that for a given class c we have a probability

distribution $p(\mathbf{x}|F^c)$ of patterns $\mathbf{x}$. The distribution is determined by class specific parameters F^c. The decision rule to classify a data point $\mathbf{x}$ with a given parameter setting F is given by the sign of the discriminant function

$$\mathcal{L}(\mathbf{x}|F) = \log \frac{p(\mathbf{x}|F^1)}{p(\mathbf{x}|F^{-1})}. \quad (1)$$

Classification on basis of the whole distribution $p(F)$ is done using the average value of the discriminant function at point $\mathbf{x}$,

$$\int_F p(F)\mathcal{L}(\mathbf{x}|F). \quad (2)$$

The parameter distribution $p(F)$ that makes the least assumptions about the choice of the parameters F while obeying the constraint that the training data are separated well can be found with the maximum entropy method. More precisely: Suppose we have a set of training data $\{\mathbf{x}_t\}_{t=1}^P$ with corresponding class labels $\{y_t\}_{t=1}^P$. The maximum entropy method maximizes the entropy of the distribution $p(F)$ subject to the classification constraints

$$\int_F p(F) y_t \mathcal{L}(\mathbf{x}_t|F) \geq \gamma_t, \quad (3)$$

where γ_t are the desired classification margins.

Jaakkola et al. (1999) suggest an improvement to this approach by including prior information about the parameter values F and margin values $\gamma = (\gamma_1, \ldots, \gamma_P)$. This is done by incorporating a prior distribution $p_0(F, \gamma)$ into the maximum entropy framework. The objective of this *minimum relative entropy principle* is to minimize the relative entropy between the prior distribution $p_0(F)$ and the new distribution $p(F)$

$$D(p\|p_0) = \int_F p(F,\gamma) \log \frac{p(F,\gamma)}{p_0(F,\gamma)}, \quad (4)$$

subject to the classification constraints:

$$\int_{F,\gamma} p(F,\gamma)\,[y_t \mathcal{L}(\mathbf{x}_t|F) - \gamma_t] \geq 0 \quad \forall t. \quad (5)$$

The decision rule for a data point $\mathbf{x}$ is

$$y = \text{sign}\left(\int_F p(F)\mathcal{L}(\mathbf{x}|F)\right) \quad (6)$$

The solution to this problem for the 'posterior' distribution $p(F, \gamma)$ has the form

$$p(F,\gamma) = \frac{1}{Z(\lambda)} p_0(F,\gamma) e^{\sum_t \lambda_t [y_t \mathcal{L}(\mathbf{x}_t) - \gamma_t]}, \quad (7)$$

where

$$Z(\lambda) = \int_{F,\gamma} p_0(F,\gamma) e^{\sum_t \lambda_t [y_t \mathcal{L}(\mathbf{x}_t) - \gamma_t]}, \quad (8)$$

is the partition function which normalizes the distribution $p(F, \gamma)$. The vector of Lagrange multipliers $\lambda = (\lambda_1, \ldots, \lambda_P)$ is given by the (unique) maximum of

$$J(\lambda) = -\log Z(\lambda), \quad (9)$$

under the constraint $\lambda_t \geq 0\ \forall t$

3. Generative Vector Quantization

As a generative model for the class specific probability distribution $p(\mathbf{x}|F^c)$ of the data we will use the Generative Vector Quantization representation discussed in Westerdijk et al. (1999b). In this model, data points are explained in terms of combinations of features. This section reviews the basic idea of this technique. For notational convenience we will omit in this section the class labels c since here we are concerned with finding a probability model for a single class only.

Consider a generative model with one hidden (latent) layer of n binary units with states $\mathbf{s} \in \{0,1\}^n$ and a continuous visible layer (the layer corresponding to the data) with values $\mathbf{x} = (x_1, \ldots, x_d, \ldots, x_D) \in \Re^D$. For a given set of model parameters F such a generative model has the form

$$p(\mathbf{x}|F) = \sum_{\mathbf{s}} p(\mathbf{x}|\mathbf{s}, F) p(\mathbf{s}). \quad (10)$$

In generative vector quantization the distribution $p(\mathbf{x}|F)$ is parameterized by a real $D \times n$ matrix F with columns $\mathbf{f}_1, \ldots \mathbf{f}_n$. For a given state $\mathbf{s}$ the distribution of $\mathbf{x}$ is Gaussian with mean $F\mathbf{s} = \sum_{i=1}^n \mathbf{f}_i s_i$ and variance σ

$$p(\mathbf{x}|\mathbf{s}, F) = (2\pi\sigma^2)^{-D/2} e^{-\frac{1}{2\sigma^2}\|\mathbf{x} - F\mathbf{s}\|^2}. \quad (11)$$

Each binary state $\mathbf{s}$ therefore corresponds to a cluster center located at $F\mathbf{s}$. The vectors $\mathbf{f}_i$ corresponding to the columns of F will be called *features*.

An example of a set of clusters generated by three features in a 2-dimensional space is shown in Figure 1. The data points for which this clustering is found are plotted with small dots. Note that the number of features n is not related to the dimensionality of the data space. Hence, considered as a basis, the feature set may be under or over complete.

We will consider a special deterministic variant of the model defined by (10) and (11). In the deterministic case each data point $\mathbf{x}$ is associated with only one

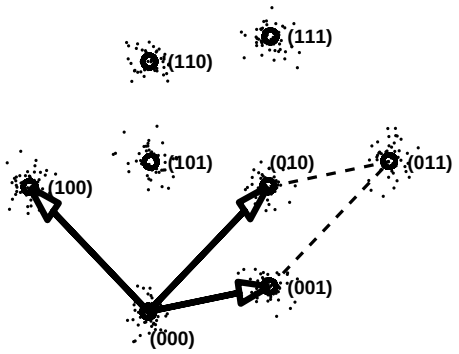

Figure 1. The cluster (circles) are generated by a small set of basic features $\mathbf{f}_1, \mathbf{f}_2$ and $\mathbf{f}_3$, which correspond to the states (100), (010) and (001), respectively. The cluster corresponding to the state (011), for example, is given by the sum of the features corresponding to the two states (001) and 010 (see broken lines).

binary feature combination (cluster) $F\mathbf{s}$, namely the one which is closest to $\mathbf{x}$ in the Euclidian sense. In more technical terms, the distribution $p(\mathbf{s}|\mathbf{x})$ of binary states $\mathbf{s}$ for a given data point $\mathbf{x}$ is $p(\mathbf{s}|\mathbf{x}) = \delta(\mathbf{s} - \mathbf{s}_\mathbf{x})$ which is equal to 1 if $\mathbf{s} = \mathbf{s}_\mathbf{x}$ and equal to 0 otherwise. The state $\mathbf{s}_\mathbf{x}$ is that state $\mathbf{s}$ for which $\|\mathbf{x} - F\mathbf{s}\|^2$ is minimal. One can also view this deterministic model as the Gaussian mixture model (10) with $\sigma \to 0$.

This deterministic model is closely related to the representation used in standard vector quantization, see for example Gray (1984).In standard vector quantization too a data point is associated with only a single cluster or codebook vector. The difference is that in the generative model, considered here, the clusters (codebook vectors) are constructed out of a smaller set of basic features $\mathbf{f}_i$.

Because of its connection to standard vector quantization we use the name *generative vector quantization (GVQ)* for the $\sigma \to 0$ generative model.

3.1 GVQ Learning

In GVQ each data point $\mathbf{x}$ is associated with a particular codebook vector corresponding to a unique binary state $\mathbf{s}_\mathbf{x}$. The squared Euclidian distance between the whole data set $\mathcal{D} = \{\mathbf{x}^t | t = 1, \ldots, P\}$ and its cluster representation, $\{F\mathbf{s}_{\mathbf{x}^t} | t = 1, \ldots, P\}$, is

$$E = \sum_{t=1}^{P} \|\mathbf{x}^t - F\mathbf{s}_t\|^2 \,, \tag{12}$$

where $\mathbf{s}_t$ is shorthand notation for $\mathbf{s}_{\mathbf{x}^t}$. The task is, therefore, to find both the optimal associations of data points to cluster centers, and the best feature vectors in order to minimize E. Since the associations between data points and cluster centers will change if the feature matrix F is changed, minimizing (12) directly with respect to F and the associations is not practical. For this reason a two-step iteration procedure is used.

After initialization of the features F the GVQ learning algorithm iterates between an association step 1 which finds, for each data-point, the single most nearby cluster center and a minimization step 2 which finds the optimal feature configuration for the given association:

1. For $t = 1, \ldots, P$

$$\mathbf{s}_t \leftarrow \underset{\mathbf{s}}{\text{argmin}} \|\mathbf{x}^t - F\mathbf{s}\|^2, \tag{13}$$

2.

$$F \leftarrow \underset{\tilde{F}}{\text{argmin}} \sum_t \|\mathbf{x}^t - \tilde{F}\mathbf{s}_t\|^2 \tag{14}$$

The first step, (13), is computationally difficult since, in principle, it involves a search through all 2^n binary states $\mathbf{s}$. There are different methods to find approximate solutions to (13). In a recent paper (Westerdijk et al., 1999a) we compared different search methods on a large variety of problems. The methods considered are the variational mean-field approximation and methods from Bayesian inference: Belief Propagation and Belief Revision. Depending on the structure of the problem (*e.g.,* correlations between binary units, sparsity of the feature matrix F, etc.) a different method is preferred. In most situations it is possible to find a good solution to (13) within reasonable time even for large numbers of features.

The second step (14) is computationally straightforward since it involves an unconstrained minimization of a simple quadratic form.

4. Maximum Entropy Discrimination with Multiple GVQ Models

Suppose we have trained a separate GVQ feature model F^c on each subset of the training data in which all data points have the same class label $y_t = c$. For all data points $\mathbf{x}_t$ (irrespective of its class) each GVQ model F^c has a closest cluster center. This cluster center corresponds to the binary state $\mathbf{s}^c_{\mathbf{x}_t}$. A default strategy to classify a novel example $\mathbf{x}$ is to classify the example with class label c corresponding to that GVQ model with the smallest distance $\|\mathbf{x} - F^c\mathbf{s}^c_\mathbf{x}\|^2$ to $\mathbf{x}$. However, the objective function with which we found the feature representations (12) is not optimally tuned for finding the best decision boundary.

Our aim is to apply the Maximum Entropy Discrimination method to tune the features F^c of each model to improve the decision boundary. As we will see shortly, however, direct application of the Maximum Entropy Discrimination method is intractable. Therefore we propose a two-step incremental procedure, similar to GVQ learning. In the first step, the associations are computed given the feature matrices F^c (see (13)). In the second step, the associations are hold constant and the (prior) features of each model are slightly updated $F_0^c \rightarrow F_0^c + \Delta F^c$. Under the assumption is that the associations computed with F_0^c and $F_0^c + \Delta F^c$ do not differ significantly, the iterative procedure gradually improves the classification boundary. In this section we will elaborate the update ΔF^c of the feature matrix.

4.1 The Discriminant Function

The full GVQ model F consists of a class 1 feature matrix F^1 and a class -1 matrix F^{-1} *i.e.,* $F = \{F^{-1}, F^1\}$. Data points are classified according to their distance to the GVQ models or equivalently to the sign of the discriminant function:

$$\mathcal{L}(\mathbf{x}|F) = \beta\|\mathbf{x} - F^{-1}\mathbf{s}_{\mathbf{x}}^{-1}\|^2 - \beta\|\mathbf{x} - F^1\mathbf{s}_{\mathbf{x}}^1\|^2, \tag{15}$$

where $\beta = 1/(2\sigma^2)$ (see (11)) and $\mathbf{s}^{-1}$ resp. $\mathbf{s}^1$ are the binary states corresponding to those class -1 and class 1 cluster centers which are closest to data point $\mathbf{x}$.

4.2 Prior Distributions

As explained in section 2 we need to specify a prior probability $p_0(F, \gamma)$ distribution over the features F and margins γ. We take a distribution which factorizes over the classes and margins

$$p_0(F, \gamma) = p_0(F^1)p_0(F^{-1})\prod_t p_0(\gamma_t). \tag{16}$$

We assign the prior distribution over the features of each class in terms of initial models F_0^c

$$p_0^c(F) = \frac{1}{Z_0^c} e^{-\nu\|F - F_0^c\|^2}. \tag{17}$$

A sensible choice for the prior distribution of the margins as proposed by Jaakkola et al. (1999) is given by

$$p_0(\gamma_t) = Ke^{-K(\gamma_0 - \gamma_t)} \quad \text{for} \quad \gamma_t \leq \gamma_0, \tag{18}$$

and $p_0(\gamma_t) = 0$ for $\gamma_t > \gamma_0$. With this choice for the prior, negative margin values γ_t have a small but finite probability. This makes it possible to find decision boundaries also if the data is non-separable, for example due to intrinsic noise. In that case a small fraction, controlled by K, of the training data is allowed to be at the wrong side of the decision boundary. In section 4.3 we will show that K becomes an upper bound for the solution of the Lagrange multipliers λ.

4.3 The Objective Function for the Lagrange Multipliers λ

As explained in section 2 the distribution $p(F)$ with which we construct the new improved decision rule (7) is specified in terms of a set of Lagrange multipliers $\lambda = (\lambda_1, \ldots, \lambda_P)$. To construct the objective function (9) for λ we first need to compute the partition function (8). The partition function $Z(\lambda)$ for the 'posterior' distribution over features F and margins γ is factorized as follows

$$Z(\lambda) \equiv Z^1(\lambda)Z^{-1}(\lambda)Z^\gamma(\lambda). \tag{19}$$

The partition function for the margins is

$$Z^\gamma(\lambda) = \prod_t \frac{1}{1 - \lambda_t/K} e^{-\lambda_t}. \tag{20}$$

The partition functions over features is more complicated,

$$Z^c(\lambda) = \frac{1}{Z_0^c}\int_F e^{-\nu\|F - F_0^c\|^2} e^{-c\beta\sum_t \lambda_t y_t \|\mathbf{x}_t - F\mathbf{s}_t^c\|^2} \tag{21}$$

where $c \in \{-1, 1\}$ is the class index. Since the associations $\mathbf{s}_t^c$ are in general functions of F, the integrand in (21) is a very complicated function of F. However, by keeping the associations fixed in the state defined by F_0^c, (21) reduces to a Gaussian integral which can be easily computed. This approach, however, can only make sense if the correct associations in the posterior distribution of F are not too much different from the associations defined by F_0^c. Therefore, the posterior distribution should differ only slightly from the prior distribution. To achieve this, the parameter settings should be such that $\nu \gg 1$.

With constant associations, (21) can be computed by Gaussian integration,

$$Z^c(\lambda) = \frac{|\pi\Lambda^{c\,-1}|^{\frac{D}{2}}}{Z_0^c}\, e^{\sum_d \left[\frac{1}{4}\mathbf{b}_d^{c\,T}\Lambda^{c\,-1}\mathbf{b}_d^c - C_d^c\right]}, \tag{22}$$

where

$$\Lambda^c = \nu I + c\beta\sum_t \lambda_t y_t \mathbf{s}_t^c \mathbf{s}_t^{c\,T}. \tag{23}$$

Note that Λ^c is defined in terms of direct products $\mathbf{s}_t^c \mathbf{s}_t^{cT}$ defining a matrix with elements $s_i^c s_j^c$. The vector $\mathbf{b}_d$ in (22) is given by

$$\mathbf{b}_d^c = -2c\beta \sum_t \lambda_t y_t x_{dt}^c \mathbf{s}_t^c \tag{24}$$

and the scalar C_d^c is

$$C_d^c = c\beta \sum_t \lambda_t y_t {x_{dt}^c}^2. \tag{25}$$

The training points $\mathbf{x}_t^c$ are given relative to the cluster centers $\mathbf{x}_t^c = \mathbf{x}_t - F_0^c \mathbf{s}_t^c$.

Now we are able to compute the objective function $J(\lambda) = -\sum_c \log Z^c(\lambda)$ which consists of terms

$$-\log Z^c(\lambda) = K^c - \sum_d \frac{1}{4} \left(\mathbf{b}_d^c\right)^T \left(\Lambda^c\right)^{-1} \mathbf{b}_d^c + \sum_d C_d^c, \tag{26}$$

where K^c is a constant which does not depend on λ. Because of the matrix Λ^c and its inverse, $J(\lambda)$ has a complicated dependency on the Lagrange multipliers λ. However, since we assumed $\nu \gg 1$, the off-diagonal elements of Λ in (23) are small compared to its diagonal elements and Λ. This motivates us to make the following approximation

$$\Lambda_{ij}^{-1} \approx \frac{1}{\nu} \delta_{ij}. \tag{27}$$

With this approximation the augmented objective function $J'(\lambda)$ (omitting the constant terms) becomes

$$J'(\lambda) = \sum_t \log(1 - \lambda_t / K) + L^T \lambda - \lambda^T Q \lambda, \tag{28}$$

where

$$L_t = \gamma_0 + y_t \beta \sum_c c \sum_d {x_{dt}^c}^2, \tag{29}$$

and

$$Q_{t,t'} = \frac{\beta^2}{\nu} y_t y_{t'} \sum_c \mathbf{s}_t^c \mathbf{s}_{t'}^c \sum_d x_{dt}^c x_{dt'}^c. \tag{30}$$

To find the Lagrange multipliers λ, (28) needs to be maximized under the constraints $\lambda_t \geq 0$. The logarithmic term in (28) ensures that the solution will satisfy $\lambda_t \leq K$ *i.e.,* the constant K is an upper bound for the Lagrange multipliers. As an approximation we discard the logarithmic terms from (28) and instead optimize

$$J^*(\lambda) = L^T \lambda - \lambda^T Q \lambda \tag{31}$$

under the constraints $0 \leq \lambda_t < K$.

4.4 Computation of the Classifier

Having obtained the Lagrange multipliers by maximizing (31) we are able to compute the augmented classifier (6)

$$y(\mathbf{x}) = \text{sign}\left(\int_{F^1, F^{-1}} p_1(F^1) p_{-1}(F^{-1}) \mathcal{L}(\mathbf{x}) \right) \tag{32}$$

To evaluate (32) we have to compute integrals of the form:

$$\int_F \|\mathbf{x} - F\mathbf{s}_x\|^2 p(F). \tag{33}$$

This amounts to evaluating first and second moments of a Gaussian distribution, the result of which is

$$\int_F \|\mathbf{x} - F\mathbf{s}_x\|^2 p(F) = \|\mathbf{x} - M\mathbf{s}_\mathbf{x}\|^2 + \frac{D}{2} \mathbf{s}_\mathbf{x}^T \Lambda^{-1} \mathbf{s}_\mathbf{x}, \tag{34}$$

where $M = F_0 + c\beta \sum_t \lambda_t y_t \Lambda^{-1} \mathbf{x}_t (\mathbf{s}_t)^T$ is the *augmented feature matrix.* Again, the direct product $\mathbf{x}_t (\mathbf{s}_t)^T$ in M defines a matrix with elements $x_{ti} s_{tj}$. With the approximation (27) the classifier then becomes

$$y(\mathbf{x}) = \text{sign}\left(\|\mathbf{x} - M^{-1}\mathbf{s}_\mathbf{x}^{-1}\|^2 - \|\mathbf{x} - M^1 \mathbf{s}_\mathbf{x}^1\|^2 + \frac{D}{\nu} \Delta S^2 \right), \tag{35}$$

where

$$M^c \approx F_0^c + \Delta F_0^x \tag{36}$$

with

$$\Delta F_0^x = c \frac{\beta}{\nu} \sum_t \lambda_t y_t \mathbf{x}_t^c \mathbf{s}_t^c \tag{37}$$

Furthermore, ΔS^2 in (35) is defined as

$$\Delta S^2 = \frac{1}{2} \left(\|\mathbf{s}_\mathbf{x}^{-1}\|^2 - \|\mathbf{s}_\mathbf{x}^1\|^2 \right) \tag{38}$$

The parameter β/ν in (37) behaves as a learning parameter. In the limit $\beta/\nu \to 0$ we retain the original discriminant function (15) defined by F_0.

As an illustration, consider the special case that each feature set consists of only one feature $\mathbf{f}_0$. In that case $\Delta S^2 = 0$ and the new class $c = 1$ feature $\mathbf{f}_{\text{new}}^1$ is

$$\mathbf{f}_{\text{new}}^1 = \mathbf{f}_0^1 + \frac{\beta}{\nu} \sum_t \lambda_t y_t (\mathbf{x}_t - \mathbf{f}_0^1) \tag{39}$$

From this expression we see that the features are pulled towards the center of gravity of patterns (weighted with the positive values λ) with the correct label $y_t = 1$ and are pushed away from patterns with the opposite class label $y_t = -1$.

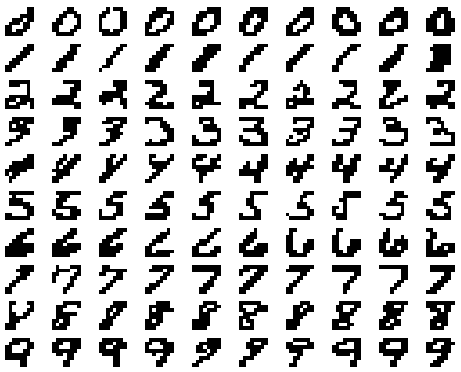

Figure 2. A sample of binary images of handwritten digits.

4.5 Practical Implementation

Our method for doing the constraint maximization of (31) is related to the Sequential Minimal Optimization algorithm (Platt, 1999) which is developed for optimizing Support Vector Machines (SVMs). At each stage in this algorithm two Lagrange Multipliers λ_t are selected for optimization while holding the others fixed. This is repeated until the error is within a ϵ distance from the (unique) minimum. The difference with our implementation is that we do not have the additional SVM constraint $\sum_t y_t \lambda_t = 0$. Therefore we can maximize $J^*(\lambda)$ by doing 1-dimensional optimizations (iteratively updating a single λ_t at a time) which can be done analytically making the optimization very fast.

5. Application to Handwritten Digit Recognition

The data set we used to test our method consisted of 11000 handwritten digits compiled by the U.S. Postal Service Office of Advanced Technology. We used the same preprocessed data as Saul et al. (1996) and Sallans et al. (1998). Each processed image is built up out of 8×8 black and white pixels. A data sample of each digit class is shown in Figure 2. We used the same partitioning of the data into a training set and a test set as Saul et al. (1996) and Hinton et al. (1995). The training set consisted of 700 examples of each digit and the test set consisted of 400 examples of each digit.

The multi-class classification problem was split up into 10 binary classification problems for which we trained 20 models in total. For each digit, we constructed a classification boundary between the training examples of that digit and the examples of the other digits. To do this we used the iterative Maximum Entropy procedure to train two GVQ models for each digit class, one for the positive training examples (the 700 examples corresponding to the digit) and one for the negative training examples (the $9 \times 700 = 6300$ examples corresponding to the remaining digits).[1]

Cross-validation within the training set, where we varied the number of latent states from 1 to 8, revealed that we needed to choose at least 8 hidden units for each model. To restrict the computational overhead we did not investigate the performance of larger numbers of units. The reason for this is that, for the purpose of this paper, we did not incorporate any approximating techniques to find the associations $\mathbf{s}$ for each data pattern. As explained in section 3.1 the exact algorithm scales exponentially with the number of features. As mentioned in the same section, there exist different accurate algorithms (such as mean-field) to speed up the association step. With the 8 latent unit configuration the CPU time needed for a 500 MHz Pentium III to fit a positive and negative model for one digit was 1.4 hours.

As a parameter setting we used $\epsilon \equiv \beta/\nu = 0.001$, and $K = 1$ for the upper bound. Following Jaakkola et al. (1999), we chose $\gamma_0 = 1$ for the position where the margin prior, $p_0(\gamma)$, peaks. These choices gave good convergence behavior *i.e.,* the training score slowly increases to a maximum without oscillations.

Figure 3 shows typical digit representations of each positive and negative digit model. The positive models (top row) clearly correspond to the class they were trained on. In some cases the negative models resemble a specific digit. For example, the negative model of digit 2 (third column, second row) looks like a 9. This indicates that the digit with which a 2 is mostly confused is a 9 and therefore the negative model emphasis on the 9 to separate it from the 2.

At each iteration we computed the training set scores on the binary classification problems. As can be seen in the left subplot of Figure 4 the training score reaches 100% correct after about 10 iterations. The change in the feature values $\|\Delta F_0\|$ at each stage is plotted in the right subplot.

At each iteration only a tiny fraction (≈ 1 out of 100) of the training set patterns $\mathbf{x}_t$ had a corresponding Lagrange multiplier $\lambda_t > 0$. Hence, the change in the features ΔF (37) is specified in terms of only a few

[1] As explained in Jaakkola et al. (1999),the maximum entropy framework can also be formulated for multi-class classification problems in a direct way. On the other hand, there are indications (Weston & Watkins, 1998) that, for practical purposes, the multi-class formulation does not have a great advantage over the multiple binary approach.

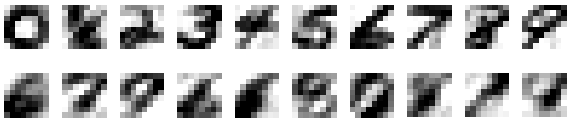

Figure 3. Typical GVQ representations that were learned with the maximum entropy procedure. In each column, the upper image corresponds to the GVQ representation of the 'positive' class and lower image corresponds to the 'negative' class.

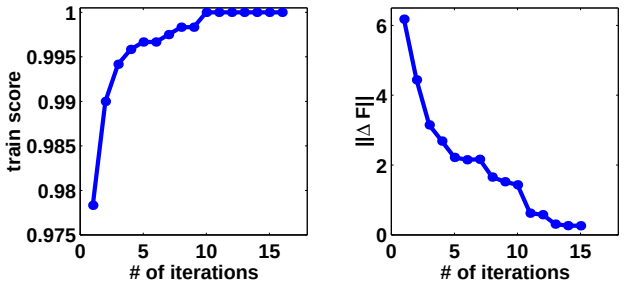

Figure 4. Left subplot: The training set score after each stage of adjusting the features $F_0^{\text{new}} \leftarrow F_0 + \Delta F$. Right subplot: The change in the feature values $\|\Delta F\|$ at each stage.

training patterns. This is analogous to what we see in support vector machines where the decision boundary is defined by a only few data points (the support vectors).

To classify a test example $\mathbf{x}$ we computed its distance $E_i^c(\mathbf{x}) = \min_{\mathbf{s}} ||\mathbf{x} - M_{(i)}^c \mathbf{s}||$ to both the positive $c = 1$ and negative $c = -1$ models of each class i. We then computed, for each digit i, the difference ΔE_i between the distances to the positive and negative model, *i.e.*, $\Delta E_i = E_i^1(\mathbf{x}) - E_i^{-1}(\mathbf{x})$. The class i for which ΔE_i was minimal was then chosen to label the test example $\mathbf{x}$. This procedure corresponds to the intuitive notion that an example belonging to a certain class should be close to the model corresponding to that class and far away from the negative model of that class.

The prior GVQ models (corresponding to F_0 in (17)) misclassified 7.8% of the test patterns. After application of the iterative Maximum entropy procedure the test error reduced with 3% to 4.8%. A component wise representation of this error is shown in Table 1 where we printed the confusion matrix. The entry on the ith row and jth column specifies how many times digit i was classified as digit j.

Using exactly the same partitioning of the data set into

Table 1. Confusion matrix obtained with the GVQ model after optimization with the maximum entropy procedure.

	0	1	2	3	4	5	6	7	8	9
0	391	2	1	0	2	0	2	0	2	0
1	0	395	0	0	0	1	1	0	3	0
2	1	6	374	2	3	2	2	2	7	1
3	0	0	5	364	0	18	0	2	7	4
4	0	0	1	0	383	1	0	3	1	11
5	1	5	0	8	0	379	3	1	3	0
6	1	4	2	0	0	3	388	0	2	0
7	0	1	2	0	1	0	0	386	4	6
8	1	14	2	2	1	12	2	5	357	4
9	1	2	0	0	1	0	0	5	2	389

Table 2. Test error rates on the digit recognition problem

GVQ prior	nearest neighbor	back prop.	wake-sleep	GVQ-ME	mean field
7.8%	6.7%	5.6%	4.8 %	4.8%	4.6%

train- and test-set, Hinton et al. (1995) reported test error rates of 6.7%, 5.6% and 4.8% obtained with nearest neighbor classification, a back-propagation multilayer perceptron and generative models trained with the wake-sleep algorithm, respectively. Again using the same data partitioning, Saul et al. (1996) obtained a slightly smaller error rate of 4.6% with sigmoid belief networks. In that case a single network was trained for each digit using standard (unconstrained) maximum likelihood optimization. Each network consisted of an 8×8 grid of visible units, a middle layer of 24 binary hidden units and a top layer of 8 binary units. An overview of the test error results is presented in Table 2.

6. Conclusion

Generative models provide a way to model the distribution of complex structures within data. In addition, the description of the data in terms of a small set of elementary components may lead to a lucent representation which is important in many data exploration tasks.

In the standard approach generative models are optimized for maximum likelihood estimation and are therefore not directly optimized for the task of classification. In this paper we proposed a method to adjust the models to improve their performance as a combined classifier. The basic idea of our method is to use a deterministic approximation to the distribution of the latent states within each model. While fixing this distribution we use the maximum entropy method to

slightly adjust the models to improve the separation of the training data. Iterating further in this manner leads to a new set of models which is better suited for the purpose of classification.

The effectivity of this procedure is demonstrated by the test error results we obtained for the digit recognition problem: Initially the classification score of the GVQ models was much worse than that of standard classification methods such as nearest neighbor classification and back-propagation neural networks. After adjusting the features with the iterative maximum entropy procedure the GVQ models outperform these standard methods and the performance is comparable to the performance of advanced techniques such as wake-sleep and mean-field sigmoid belief networks.

In this paper the iterative maximum entropy scheme was applied to GVQ. In GVQ the visible states $\mathbf{x}$ depend linearly on the features values F. This enabled us to analytically compute the partition function $\log Z(\lambda)$, needed to construct the iterative maximum entropy scheme. In other linear models, such as principal component analysis and factor analysis, the linearity can be exploited in a similar way and iterative maximum entropy procedures can be constructed analogously.

In non-linear models the visible states depend non-linearly on the feature values. Examples of non-linear models are models with sigmoid transfer functions (for binary inputs) and soft-max functions (for nominal inputs). For non-linear models the partition function cannot be computed analytically. However, if the prior feature distribution is sharply peaked, the computation of the partition function effectively involves an integration over a small region in feature space. In that case a linear expansion could be sensible. With such an approximation the procedure can be extended to include non-linear models. Iterative maximum entropy schemes for non-linear models are currently under study.

References

Baum, L., Petrie, T., Soules, G., & Weiss, N. (1970). A maximization technique occuring in the statistical analysis of probabilistic functions of markov chains. *The Annals of Mathematical Statistics*, *41*, 164–171.

Gray, R. (1984). Vector quantisation. *IEEE ASSP Magazine*, 4–29.

Hinton, G., Dayan, P., Frey, B., & Neal, R. (1995). The wake-sleep algorithm for unsupervised neural networks. *Science*, 1158–1161.

Jaakkola, T., Meila, M., & Jebara, T. (1999). *Maximum entropy discrimination* (Technical Report MIT-AITR-1668). MIT AI Lab.

Jordan, M. I., & Jacobs, R. (1994). Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, *6* ,181–214.

Neal, R. M. (1991). Connectionist learning of belief networks. *Artificial Intelligence*, *56*, 71–113.

Platt, J. (1999). chapter Fast training of support vector machines using sequential minimal optimization. *Advances in kernel methods - support vector learning*, Cambridge, MA: MIT press.

Sallans, B., Hinton, G., & Ghahramani, Z. (1998). A hierchical community of experts. In C. M. Bishop (Ed.), *Neural networks and machine learning*, NATO ASI Series F, 269–284. Springer-Verlag.

Saul, L. K., Jaakkola, T., & Jordan, M. I. (1996). Mean field theory for sigmoid belief networks. *Journal of Artificial Intelligence Research*, *4*, 61–76.

Vapnik, V. (1995). *The nature of statistical learning theory.* New York: Springer-Verlag.

Westerdijk, M., Barber, D., & Wiegerinck, W. (1999a). Deterministic generative models for fast feature discovery. Unpublished manuscript. Dept. of Medical Physics and Biophysics, University of Nijmegen, The Netherlands.

Westerdijk, M., Barber, D., & Wiegerinck, W. (1999b). Generative vector quantisation. *Proceedings of the International Conference on Artificial Neural Networks 1999* (pp. 934–939).

Weston, J., & Watkins, C. (1998). *Multi-class support vector machines* (Technical Report CSD-TR-98-04). University of London, Dept. of Computer Science.