Hollmén,J., V.Tresp, and O.Simula (2000). A learning vector quantization algorithm for probabilistic models. In Proceedings of EUSIPCO 2000 - X European Signal Processing Conference ,Volume II,pp.721 –724.

A LEARNING VECTOR QUANTIZATION ALGORITHM FOR PROBABILISTIC MODELS

Jaakko Hollmén †, *Volker Tresp* ‡ *and Olli Simula* †
† Helsinki University of Technology, Laboratory of Computer and Information Science
P.O. Box 5400, 02015 HUT, Finland, e:mail: [Jaakko.Hollmen,Olli.Simula]@hut.fi
‡ Siemens AG, Corporate Technology, Information and Communications
81730 Munich, Germany, e-mail: Volker.Tresp@mchp.siemens.de

ABSTRACT

In classification problems, it is preferred to attack the discrimination problem directly rather than indirectly by first estimating the class densities and by then estimating the discrimination function from the generative models through Bayes's rule. Sometimes, however, it is convenient to express the models as probabilistic models, since they are generative in nature and can handle the representation of high-dimensional data like time-series. In this paper, we derive a discriminative training procedure based on Learning Vector Quantization (LVQ) where the codebook is expressed in terms of probabilistic models. The likelihood-based distance measure is justified using the Kullback-Leibler distance. In updating the winner unit, a gradient learning step is taken with regard to the parameters of the probabilistic model. The method essentially departs from a prototypical representation and incorporates learning in the parameter space of generative models. As an illustration, we present experiments in the fraud detection domain, where models of calling behavior are used to classify mobile phone subscribers to normal and fraudulent users. This is an extension of our earlier work in clustering probabilistic models with the Self-Organizing Map (SOM) algorithm to the classification domain.

1 INTRODUCTION

Classification is inherently a discrimination problem. Contrary to the real nature of the problem, the classical approach to classification solves the problem by first estimating the class specific densities and by then calculating the posterior probabilities of classes, which in turn define the discrimination function [3, 5]. Recent research [8] shows that for problems where discrimination is the main concern, attacking discrimination problems by density estimation may be inferior to more direct approaches. Still, it may be desirable to formulate the models in terms of generative, probabilistic models, while the learning procedure aims at being able to discriminate well. In this paper, we consider learning codebook-based classifiers, where the codebook is expressed in terms of probabilistic models, but where the training procedure is discriminative in nature. The training algorithm is derived from the LVQ algorithm [10], which learns to classify data samples by nearest neighbor (in the Euclidean distance sense) classification with regard to labeled codebook vectors. We depart from the prototype representation and express the codebook in terms of generative probabilistic models. These models are associated with the input data through likelihood, which measures the *generative* probability of data. The use of likelihood-based distance measure is derived from the Kullback-Leibler distance between unknown probability density of a data sample and a model stored in the codebook. The iterative update is based on the gradients of the model with regard to the parameters. The classifier may be used as a maximum likelihood classifier, and the resulting codebook may be used as a fixed kernel base used for further training as suggested in [8]. We also derive the form of the gradient update for models with hidden variables, which enables us to use finite mixture models [6, 1] as the models in the codebook. The work is an extension of our earlier work to the classification domain. In [7], we introduced a Self-Organizing Map algorithm for clustering probabilistic models. This work was motivated with a general user profiling problem, where a large body of data must be clustered, but where it is desirable to formulate the cluster models with probabilistic models rather than with prototypes of data, as is usually done with the Self-Organizing Map algorithm. In the experiments, we present a time-series classification problem. Behavior of mobile phone users is collectively described by call data. Data is labeled to belong to classes *fraud* and *normal* and is represented as a binary time series indicating air time of mobile phone users. Calling activity of mobile phone users is modeled using dynamic, probabilistic models, which express the transition probabilities in time between states no-calling and calling. In all, this work presents a promising approach to learning unsupervised models for discrimination based on supervised learning.

2 LVQ ALGORITHM

The Learning Vector Quantization (LVQ) is an algorithm for learning classifiers from labeled data samples. Instead of modeling the class densities, it models the discrimination function defined by the set of labeled codebook vectors and the nearest neighborhood search between the codebook and data. In classification, a data point x_i is assigned to a class according to the class label of the closest codebook vector. The training algorithm involves an iterative gradient update of the winner unit. The winner unit m^c is defined by

$$c = \arg\min_k \|x_i - m^k\|. \tag{1}$$

The direction of the gradient update depends on the the correctness of the classification using a nearest-neighborhood rule in Euclidean space. If a data sample is correctly classified (the labels of the winner unit and the data sample are the same), the model vector closest to the data sample is attracted towards the sample; if incorrectly classified, the data sample has a repulsive effect on the model vector. The update equation for the winner unit m^c defined by the nearest-neighbor rule and a data sample $x(t)$ are

$$m^c(t+1) := m^c(t) \pm \alpha(t)[x(t) - m^c(t)] \tag{2}$$

where the sign depends on whether the data sample is correctly classified $(+)$ or misclassified $(-)$. The learning rate $\alpha(t) \in]0,1[$ must decrease monotonically in time. For different picks of data samples from our training set, this procedure is repeated iteratively until convergence. Kohonen also presents optimized learning-rate LVQ, where the learning-rate is optimized for each codebook individually. For further variations, see [10, 11].

3 LVQ FOR PROBABILISTIC MODELS

Our new classifier consists of a labeled codebook of probabilistic models. The conceptual difference to standard LVQ is that the codebook models are no longer in the space of input data, but rather in the space defined by the parameters of models.

3.1 Justification for the Distance Measure

In order to associate data with models, we need to define a distance measure between models and data. In the following, a likelihood-based distance metric is justified by a derivation from the Kullback-Leibler distance [1], which relates two probability densities. Thinking of data as samples originating from an unknown distribution $p(x)$ and having a model $q(x;\theta)$ with parameterization θ, the distance relating $p(x)$ and $q(x;\theta)$ is the Kullback-Leibler distance

$$KL(p \parallel q) = -\int p(x) \log \frac{q(x\,;\theta^k)}{p(x)} dx. \tag{3}$$

The true distribution $p(x)$ is unknown but can be approximated by a Dirac unit impulse at the available data sample by $p(x) \approx \delta(x - x_i)$, which after substitution to Equation (3) gives us for the Kullback-Leibler distance the negative log likelihood of our data with our empirical model

$$= -\log q(x_i\,;\theta^k).$$

Thus, minimizing the Kullback-Leibler distance between the unknown true distribution that generated the data and our empirical model leads to minimizing the negative logarithm of the likelihood of the data with our empirical model. This justifies the use of this probability measure as a distance measure between models and data. The justification is the same as used in our earlier work [7].

3.2 Winner Search and Gradient Update

In contrast to the standard LVQ, where the winner unit is defined with a nearest-neighbor rule in the Euclidean space, we now have a winner unit which minimizes the negative log likelihood of data. Equivalently, this is a *maximum likelihood unit* m^c defined by

$$c = \arg\max_k q(x_i\,;\theta^k). \tag{4}$$

The update equation is defined as a gradient step in the space of models. The gradient is taken with regard to the model parameters θ^c of the winner unit. The update equation is

$$\theta^c(t+1) := \theta^c(t) \pm \alpha(t) \frac{\partial \log q(x(t);\theta^c)}{\partial \theta^c}. \tag{5}$$

To contrast the new approach with the standard LVQ, the winner search defined by Equation (1) is replaced by Equation (4) and the update rule in Equation (2) is replaced by Equation (5). Similar analogy applies to the choice of the sign.

3.3 Derivatives of Hidden Variable Models

Although the form of Equation 5 is valid for all probabilistic models, the expressions are particularly simple to calculate for models where all variables are observed. Models involving hidden variables such as finite mixture models [6] or hidden Markov models [14] provide an elegant framework for modeling many interesting domains. Therefore, we consider the gradient update in the case of hidden variable models, although they will be not used in the experiments. Let us assume a model with observed variables $Y = \{y_1, \ldots, y_n\}$ and hidden variables $S = \{s_1, \ldots, s_m\}$. The derivatives can readily be expressed by considering the likelihood of observed data which in turn can be expressed in terms of the joint distribution of hidden variables and observed data, marginalized over the hidden variables.

$$\frac{\partial \log P(Y;\theta)}{\partial \theta} = \frac{\partial}{\partial \theta} \log \sum_S P(Y,S;\theta)$$

This can be further manipulated by taking the derivative of a log of a function and interchanging the order of the derivative and the sum. The multiplicative factor is then the likelihood of observed data. The derivative of the joint probability can then be expressed by applying the rule $\frac{\partial}{\partial x} f(x) = f(x) \frac{\partial}{\partial x} \log f(x)$ derived from the rule of derivation for the logarithm function which gives us

$$\frac{\partial \log P(Y;\theta)}{\partial \theta} = \frac{1}{P(Y;\theta)} \sum_S P(Y,S;\theta) \frac{\partial}{\partial \theta} \log P(Y,S;\theta)$$

and finally, by taking the likelihood of observed data within the sum, we have the following expression for the derivatives of the likelihood

$$\frac{\partial \log P(Y;\theta)}{\partial \theta} = \sum_S P(S|Y;\theta) \frac{\partial}{\partial \theta} \log P(Y,S;\theta). \quad (6)$$

The first term is calculated through an inference procedure and the second term is easily calculated from the usual definition of the probabilistic models. Note, that the logarithm decomposes the product of probabilities into a sum. In case of a large codebook, complicated models and data, this calculation may become prohibitive, since the it must be repeated for every iteration in the algorithm.

4 EXPERIMENTS

The aim of fraud detection is to discriminate between normal and fraudulent behavior. In telecommunications, the calling behavior is collectively described by call data which is observed. From the call data, we may learn models which can be used for detecting fraud. Since detection is inherently a task of discrimination, it is interesting to consider discriminative training procedures where the models are expressed as generative models. Our data consists of a time-series of zeros and ones, whether a mobile phone user is idle ($x_t = 0$) or calling ($x_t = 1$) during a particular minute. The four possibilities of adjacent measurements are 00 (ongoing silence), 01 (start of a call), 11 (ongoing call), 10 (end of a call). We model this behavior with a dynamic, probabilistic model expressing the Markov transitions $P(x_t = j | x_{t-1} = i) = \theta_{ij}$. The time-series describing the calling behavior are labeled and belong to classes *fraud* and *normal behavior*. The likelihood of a time series $x_0, \ldots, x_T$ becomes then

$$P(x_0, \ldots, x_T) = P(x_0) \prod_{i=1}^{T} P(x_t = j | x_{t-1} = i; \theta).$$

The winner search is based on the parameters θ_{ij}^k (k is the codebook index) of the generative models and the sufficient statistics calculated of the time-series $\hat{p}_{ij} = n_{ij}/T$, where n_{ij} is the count of the joint occurrence $x_{t-1} = i, x_t = j$. The winner unit indexed by c is

$$c = \arg\min_k [- \sum_{i,j=0}^{1} \hat{p}_{ij} \log \theta_{ij}^k]. \quad (7)$$

Since the parameters are constrained ($\theta_{ij} \in [0,1]$ and $\sum_j \theta_{ij} = 1$), we must enforce these constraints during learning. As in our earlier work, we introduced a softmax layer [1] as $o_{ij} = \theta_{ij}^k = \exp(w_{ij}^k) / \sum_j \exp(w_{ij}^k)$ to map unconstrained parameters w_{ij} to the space of constrained parameters θ_{ij}. In deriving the update rule for the winner unit, we must use the chain rule of differentiation. For a closer derivation, see [7]. The update rule for the winner unit is then

$$\theta_{ij}^c(t+1) := \theta_{ij}^c(t) \pm \alpha(t)(\hat{p}_{ij}(1 - o_{ij}) - \hat{p}_{i\neg j} o_{ij}). \quad (8)$$

We trained a classifier with calling data from 140 normal and 70 fraudulent mobile phone users. The calling activity spanned a period of 49 days in the normal cases and 92 days in the fraudulent cases. After initialization, we classified correctly 87.6 % of the training set data with the maximum likelihood classification rule. After training, we classified correctly 97.1 % of the training data, and 95.8 % of the testing data. Figure 1 shows the codebook with 4 models for each class.

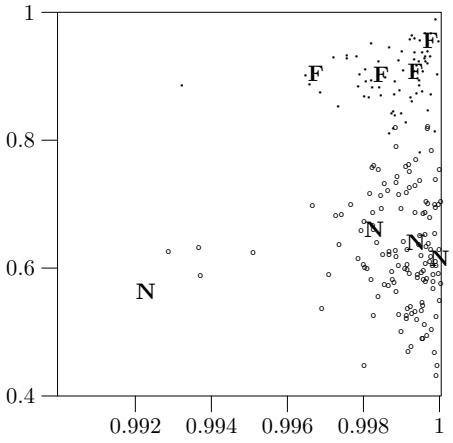

Figure 1: The resulting codebook after training. Parameter values of the codebook entries θ_{00}^k (infrequency of calls) are plotted on the horizontal axis, θ_{11}^k (lengths of the calls) on the vertical axis. The models corresponding to normal calling behavior are marked with a letter N and the ones corresponding to fraudulent behavior with a letter F. The empirical estimates for the parameter values are shown with dots for the fraud data and circles for the normal data.

As pointed out in [10, 11], careful initialization is needed to avoid situations when the repulsion from the wrong class pushes the codebook entries away from its own class. Kohonen devises many ways to guarantee a good initialization, that is, placement of class-specific codebook vectors on the right side of the decision boundary. In the new approach, the codebook is expressed in terms of parameters of probabilistic models whereas the classes are defined by the data samples. We store the unconstrained parameters w in the codebook, the constrained θ are calculated with a softmax layer [1, 7]. To

initialize the codebook, we need to estimate θ from our data x and map them back to w by fixing one of the w and solving for the other. This procedure ensures that the initialization is a reasonable one.

5 DISCUSSION

The methods described in this paper enable the learning of class specific codebooks expressed through probabilistic models. Since the elementary models are probabilistic in nature, it would be interesting to consider the codebook as a mixture model with equal mixing weights. It is worth noting that the training procedure circumvents the mixture training problem usually considered in the framework of EM algorithm [4, 13] by considering training only the winner unit. Another interesting possibility is to use the codebook in a way to define a kernel base as suggested by Jaakkola et al [8]. Contrary to their approach, we could have a limited base to define a base concentrated near the decision boundary, similarly to support vector machines [2]. These approaches would output continuous measures of the degree of class membership which would further make the Receiver Operating Characteristic (ROC) analysis possible.

We have recently become aware of the work on discriminative learning in the context of tuning generative speech recognition systems reported in [9], which is closely related to LVQ2. Similarly to their work we allow the likelihood to be used as a discriminant function. Using LVQ in discriminative tuning of hidden Markov models has also been reported by [12].

6 SUMMARY

We presented a Learning Vector Quantization (LVQ) algorithm for learning a classifier defined by a codebook of probabilistic models. The models implicitly define a discrimination function in the input data space through maximum likelihood search. The prototypical codebook vectors were replaced by generative, probabilistic models and the LVQ learning rules were modified accordingly. The likelihood-based distance was justified by a derivation form the Kullback-Leibler distance. The conceptual difference to conventional training of probabilistic models is the use of supervised, gradient based learning instead of maximum likelihood estimation. This specifically tunes the models for discrimination. The algorithm may also be used in post-processing to enhance the discriminative aspect of generative density models earlier trained with the EM algorithm.

References

[1] Chris Bishop. *Neural Networks in Pattern Recognition.* Oxford Press, 1996.

[2] Christopher J.C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[3] Vladimir Cherkassky and Filip Mulier. *Learning from data: Concepts, Theory and Methods.* John Wiley & Sons, 1998.

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

[5] Richard O. Duda and Peter E. Hart. *Pattern Recognition and Scene Analysis.* John Wiley & Sons, 1973.

[6] B.S. Everitt and D.J. Hand. *Finite Mixture Distributions.* Monographs on Applied Probability and Statistics. Chapman and Hall, 1981.

[7] Jaakko Hollmén, Volker Tresp, and Olli Simula. A self-organizing map algorithm for clustering probabilistic models. In *Proceedings of the Ninth International Conference on Artificial Neural Networks (ICANN'99)*, volume 2, pages 946–951. IEE, 1999.

[8] Tommi Jaakkola and David Haussler. Exploiting generative models in discriminative classifiers. In M. Kearns, S. Solla, and D.A. Cone, editors, *Advances in Neural Information Processing Systems: Proceedings of the 1998 Conference (NIPS'11)*, pages 487–493. MIT Press, 1999.

[9] Biing-Hwang Juang and Shigeru Katagiri. Discriminative learning for minimum error classification. *IEEE Transactions on Signal Processing*, 40(12):3043–3054, 1992.

[10] Teuvo Kohonen. *Self-Organizing Maps.* Springer-Verlag, 1995.

[11] Teuvo Kohonen, Jussi Hynninen, Jari Kangas, Jorma Laaksonen, and Kari Torkkola. LVQ_PAK: The learning vector quantization package. Technical Report A30, Helsinki University of Technology, Laboratory of Computer and Information Science, 1996.

[12] Mikko Kurimo. *Using Self-Organizing Maps and Learning Vector Quantization for Mixture Density Hidden Markov Models.* PhD thesis, Helsinki University of Technology, 1997.

[13] Geoffrey J. McLahlan. *The EM Algorithm and Extensions.* Wiley & Sons, 1996.

[14] Alan B. Poritz. Hidden markov models: A guided tour. In *Proceedings of the IEEE International conference of Acoustics, Speech and Signal Processing (ICASSP'88)*, pages 7–13, 1988.