To appear in International Journal of Knowledge-Based Intelligent Engineering Systems

Mixture Density from Autonomous Experts

Jarkko Salojärvi and Samuel Kaski
Neural Networks Research Centre
Helsinki University of Technology
P.O. Box 5400, FIN-02015 HUT, Finland
{jarkko.salojarvi,samuel.kaski}@hut.fi

Abstract. *Generative models of probability density are common in descriptive modeling of data, i.e., summarizing large data sets and exploring the dependencies between their variables. Mixture models are a class of generative models particularly suitable for clustering. If it is assumed that each data sample stems from one of a set of sources that generate data independently, it is desirable to model the data with a mixture of "experts" corresponding to the sources. The experts cannot learn autonomously, since each of them needs to know the contribution of the others. In this paper we introduce a mixture model of autonomous experts that approximates the plain mixture density model by a set of experts that learn autonomously. Each expert stores simplified representations of the other experts and uses them to judge the contribution of the others. The representations are updated intermittently. During learning the modeling task becomes divided and conquered automatically.*

1 Introduction

Generative modeling of probability distributions is a well-established field currently under intensive investigation and having a wide spectrum of applications. As is well known, probabilistic models are advantageous in that all of the machinery of statistical inference is applicable to them.

Here we consider one specific application domain, viz. modeling of the probability density of data. Its practical applications reside particularly in clustering and modeling of the dependencies between data variables.

Mixture density models are a commonly used category of probability density estimators (see [10]). They have become almost the de facto standard in clustering applications. In mixture models it is assumed that each data sample is generated by one of a set of independently operating mixture components, and that the data set is a mixture of the samples from them. The simplest generators have normally distributed densities but more complex models can be incorporated as well, for example Hidden Markov Models in sequence analyses.

There exist good expectation maximization (EM)-based algorithms for fitting the mixture models to data, the models have lots of applications, and the models have interesting variants. For a recent overview, see [13]. Also Self-Organizing Map-type topographic mappings from latent spaces to the input space can be formulated within the mixture model framework [1, 15].

With generative models, the modeling task is effectively broken into partially overlapping subtasks assigned to different relatively simpler experts. Such division of labor increases analytical tractability and makes the results more easily understandable. The latter point is particularly important in data analysis and descriptive modeling tasks where the goal is usually to summarize large data sets and to gain understanding of their essential characteristics. The model need not necessarily correspond to the "true" process that generates the data. It is more important that it is understandable and yet flexible enough to model the desired kinds of statistical relationships inherent in the data. Each expert may then be a relatively simple model, and their mixture can be used in a data-driven fashion to explore complex data sets.

Although the experts are independent data generators, experts within a mixture cannot be fitted autonomously to data. The reason is (details are given below) that when fitting each expert the contributions of all the other experts must be known. In this paper we introduce a variant of the mixture model in which the experts can learn autonomously. The experts can then be considered as simple "intelligent agents" which can be implemented as, for example, separate processes in different computers in a distributed fashion. Each agent tries to represent certain kinds of data, and the agents collaborate in order to jointly represent the whole data set.

While learning each expert would in principle need to know what kinds of data the other experts represent. Our experts solve this problem by maintaining simplified, periodically updated representations of the other experts. Since the models are simplified, it is feasible to store them with each expert. In this paper we formulate the principle in detail and study its performance empirically for the simplest possible experts, i.e. Gaussian mixtures. The results will later be generalized to other kinds of models.

Our work is related to two topics that have been studied intensively during recent years: the mixture of experts model [5, 6] and committee machines [12]. Committees can be used to overcome the limitations of single models by combining the predictions of different experts, for example by majority voting in classification tasks. Mixtures of experts, on the other hand, are probabilistic models that combine the probability distributions generated by different experts. The experts model the conditional probability of the dependent variable given the independent variables, and learn to specialize on different domains of the independent variables.

Both of these models are usually applied to regression and classification tasks, in which the goal is to predict the value of a random variable given the value(s) of the other variable(s). The problem of modeling the dependent variable is somewhat more complicated than the density estimation problem considered in this paper. In principle extensions of our autonomous experts should be applicable to modeling conditional distributions as well.

2 Mixtures of Autonomous Models

In this Section the mixture of autonomous experts model will be introduced in its general form that is applicable to any kinds of experts. In the next Section the details will be illustrated for the simplest kinds of experts: mixtures of Gaussian distributions.

The goal is to model a data set $\{\mathbf{x}_k\}$, $k = 1, \ldots, N$, where the samples may in principle be any kinds of entities. In the applications of this paper they will be real-valued vectors, $\mathbf{x}_k \in \mathbb{R}^n$, but they could be for example symbol sequences such as the genetic base pair sequences or the amino acid sequences of proteins.

Denote the prior probability that the jth expert generates a data point by $p(M_j)$, and the probability density function of the samples that the expert generates by $p(\mathbf{x}|M_j)$. The mixture model consisting of K experts then generates the probability distribution

$$p(\mathbf{x}) = \sum_{j=1}^{K} p(\mathbf{x}|M_j) p(M_j) \ . \qquad (1)$$

For the real vector-valued data the component densities $p(\mathbf{x}|M_j)$ are typically simple parametric density functions such as the Gaussians in Section 3. For the genetic and protein symbol sequences Hidden Markov Models are a common choice. Other kinds of structured models such as probabilistic grammars could be useful for some applications as well.

The model is fitted to the data by finding optimal values for its parameters, i.e. the $p(M_j)$ and the parameters of the component densities $p(\mathbf{x}|M_j)$. In this paper we will use use the common maximum likelihood criterion. The logarithm of the likelihood of the (independently sampled) data, given the model, is

$$E = \sum_{k=1}^{N} \log p(\mathbf{x}_k) = \sum_{k} \log \sum_{j=1}^{K} p(\mathbf{x}|M_j) p(M_j) \ . \qquad (2)$$

The maximum likelihood estimate for the parameters is obtained by maximizing E with respect to them.

Let us next show more specifically why each expert needs to know the contributions of the other experts while learning. We will consider gradient-based learning for simplicity; similar conclusions hold for EM-based learning which will be discussed for Gaussian mixture models in Section 3.

Denote the parameters of $p(\mathbf{x}|M_j)$ by $\boldsymbol{\theta}_j$. The gradient is then

$$\frac{\partial}{\partial \boldsymbol{\theta}_j} E = \sum_k p(\mathbf{x}_k)^{-1} p(M_j) \frac{\partial}{\partial \boldsymbol{\theta}_j} p(\mathbf{x}_k|M_j) \ , \qquad (3)$$

where $p(\mathbf{x}_k)$ is given by (1). The gradient depends on the contributions of all the experts to the probability density, and hence of all the parameters of all of the experts, through the term $p(\mathbf{x}_k)^{-1}$. The same holds for the gradient of $p(M_j)$ (not shown for brevity).

If the experts are to learn autonomously they need an estimate for the probability $p(\mathbf{x}_k)$. Our solution is to incorporate, in each autonomous expert, a representation of the densities generated by the other experts. Denote the representation of expert j for the density of expert i by $\hat{p}^{(j)}(\mathbf{x}|M_i)$, and for the prior probability by $\hat{p}^{(j)}(M_i)$. The expert j can then approximate the mixture of all experts by

$$\hat{p}^{(j)}(\mathbf{x}) = p(\mathbf{x}|M_j) p(M_j) + \sum_{l \neq j} \hat{p}^{(j)}(\mathbf{x}|M_l) \hat{p}^{(j)}(M_l) \ . \qquad (4)$$

Gradient-based learning for general mixture models can then be carried out by adapting the parameters in short steps toward the negative gradient (3), where $p(\mathbf{x}_k)$ is approximated by $\hat{p}^{(j)}(\mathbf{x}_k)$ given in (4). The representations the experts store of each other should be updated periodically, which adds a small computational cost to the algorithm.

By assigning the autonomously operating experts to different processors or computers it is possible to distribute the computational load. If the representations the experts store of each other are considerably simpler than the original representations the experts hold of themselves, then the time and memory complexity of each expert is considerably lower than that of the whole mixture.

The feasibility of the autonomous experts model depends on whether the representations of the other experts are simple enough to reduce the complexity yet accurate enough to permit the experts to function well together as a mixture. Empirical tests for Gaussian mixtures will be presented on Section 4.

Note that it may be beneficial in distributed computing to fix the prior probabilities of the experts $p(M_j)$ to equal values. The experts then learn a solution in which each one of them is responsible of approximately the same proportion of the data.

3 Gaussian Mixture Model

In this Section the autonomous learning procedure is formulated in detail for Gaussian mixture models. The Gaussian mixture is relatively simple and used widely for density estimation and clustering (see [10, 13]), and therefore a natural candidate for a case study of autonomous learning.

In the plain Gaussian mixture model each expert generates a Gaussian probability density. This is one option in the autonomous expert model as well, but it is of course possible to group several Gaussians to the same expert. Denote the autonomous experts by M_j as in (1). We will assign a set of K_j Gaussians, denoted by $m_i^{(j)}$ to the autonomous expert j. The expert then generates the density

$$p(\mathbf{x}|M_j) = \sum_{i=1}^{K_j} p(\mathbf{x}|M_j, m_i^{(j)}) p(m_i^{(j)}|M_j) \ , \quad (5)$$

and the mixture of experts generates the density

$$p(\mathbf{x}) = \sum_{j=1}^{K} p(M_j) \sum_{i=1}^{K_j} p(\mathbf{x}|M_j, m_i^{(j)}) p(m_i^{(j)}|M_j) \ . \quad (6)$$

The mixture can be considered as a kind of a hierarchical Gaussian mixture model. All Gaussian distributions, generated by the models $m_i^{(j)}$, are assumed to be symmetrical with their means denoted by $\boldsymbol{\mu}_i^{(j)}$ and variances by $(\sigma_i^{(j)})^2$. For simplicity, the variance is here the same in all dimensions.

The maximum (log) likelihood solution for the parameters can be sought for by the gradient approach outlined in Section 2, or by the expectation maximization (EM) algorithm [4]. In the case studies of this paper the EM algorithm will be used.

The EM algorithm is a general iterative method for finding missing values. In mixture models the missing values indicate which generator has produced each data sample. The algorithm can be divided into two iteratively applied steps, the expectation and maximization step. It is straightforward to generalize the solution to our hierarchical mixture, resulting in the following two formulas for the E-step:

$$\begin{aligned} p(M_j, m_i^{(j)}|\mathbf{x}_k) &= \frac{p(\mathbf{x}_k|M_j, m_i^{(j)}) p(m_i^{(j)}|M_j) p(M_j)}{p(\mathbf{x}_k)} \\ p(M_j|\mathbf{x}_k) &= \sum_i p(M_j, m_i^{(j)}|\mathbf{x}_k). \end{aligned}$$

and the following update formulas for the M-step, in which the values obtained in the E-step are utilized:

$$\begin{aligned} \boldsymbol{\mu}_i^{(j)} &= \frac{\sum_k p(M_j, m_i^{(j)}|\mathbf{x}_k)\mathbf{x}_k}{\sum_k p(M_j, m_i^{(j)}|\mathbf{x}_k)} \\ (\sigma_i^{(j)})^2 &= \frac{\sum_{k,i} p(M_j, m_i^{(j)}|\mathbf{x}_k)\|\mathbf{x}_k - \boldsymbol{\mu}_i^{(j)}\|^2}{n \sum_k p(M_j|\mathbf{x}_k)}. \\ p(m_i^{(j)}|M_j) &= \frac{\sum_k p(M_j, m_i^{(j)}|\mathbf{x}_k)}{\sum_k p(M_j|\mathbf{x}_k)} \\ p(M_j) &= \frac{1}{N} \sum_k p(M_j|\mathbf{x}_k), \end{aligned}$$

where n is the dimensionality of the data, and N the number of data samples.

When the expert M_j is updated, the other experts $l \neq j$ are needed during the expectation step for the computation of the numerator $p(\mathbf{x}_k)$.

In this paper we will test the feasibility of the autonomous computing scheme by approximating the other experts l by single Gaussians. This approximation has the nice consequence that the optimal solution can be easily obtained in closed form. The optimal solution in the sense of the Kullback-Leibler divergence from the true distribution of the expert l,

$$\int p(\mathbf{x}|M_l) \log \frac{p(\mathbf{x}|M_l)}{\hat{p}^{(j)}(\mathbf{x}|M_l)} d\mathbf{x} \ , \quad (7)$$

is obtained by minimizing (7) with respect to the parameters of the Gaussian approximation. This relatively straightforward computation is omitted for brevity.

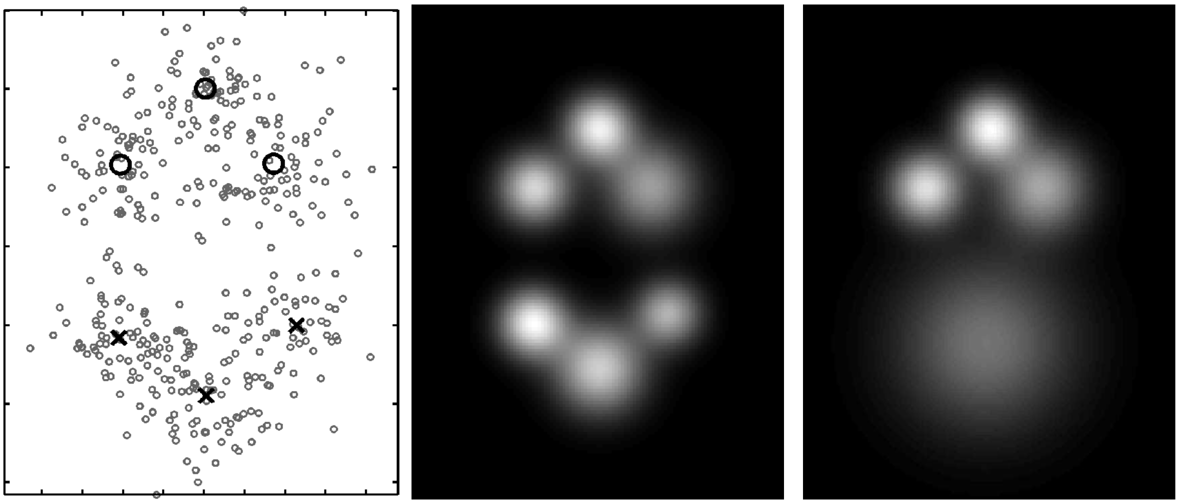

Figure 1: *A toy example that demonstrates the autonomous experts. The data (the small circles in the leftmost figure) consisted of a total of 400 samples generated from six Gaussian kernels. There were two autonomous experts, each consisting of a mixture of three Gaussians. The means of the Gaussians of the first expert are shown with circles and the means of the second expert with crosses, respectively. The density generated by the mixture of the experts after 54 EM iterations is shown in the center image. The density plot on the right shows how the first experts sees the data. The large Gaussian at the bottom represents the other expert.*

Note that it may not be easy to obtain closed-form solutions in general, for more flexible approximations than the single Gaussian used here. The approximator can then be updated by generating data from the expert and fitting it using Markov Chain Monte Carlo methods [11] or plain maximum likelihood.

The autonomous mixture model is demonstrated with toy data in Figure 1.

4 Experiments

In order to demonstrate and verify the principle of autonomous experts we modeled three real-world data sets with mixture models. We used the simplest possible experts, i.e. Gaussian mixtures, to present a proof of concept of the new principle. Same kinds of experiments will be carried out for different kinds of models later on.

If the autonomous experts were perfect their mixture would be equal to an ordinary mixture of non-autonomous experts. Hence, comparison with plain mixtures of Gaussians of the same size provides us with a yardstick of the quality. We have used two measures: The likelihood of test data not used for learning, and the classification accuracy. Note that neither model aims at minimizing classification errors; the classification accuracy is an indirect measure that provides an intuitive sense of the quality of the approximations. It is easier to interpret the classification accuracy than the abstract likelihood values.

In order to simulate the setting in which each autonomous expert is run in a separate computer, and hence it is desirable to balance their workload, the prior values $p(M_j)$ were set to equal values and fixed. The priors $p(M_j, m_i^{(j)})$ within the experts were, however, allowed to vary. The updating of the experts was asynchronized by choosing randomly one expert to be updated during each EM iteration. Since only one expert was updated in each iteration, the number of iterations, compared to the plain Gaussian mixture, was multiplied by the number of autonomous experts. This has been compensated for in the figures.

Note that with this experimental setup, also the plain Gaussian mixture model will be realized: if the number K of the experts M_j is one, the model reduces to the plain mixture of Gaussians. Since the prior probabilities $p(M_j)$ of the experts were fixed to equal values, also a Gaussian mixture model with fixed priors will be realized by setting the number K_j of the models $m_i^{(j)}$ to one for each j.

The autonomous experts updated their estimates of each other after every 10 epochs (updating after every 40 epochs was tried in some of the experiments with similar results).

All models were initialized with the K-means algorithm, after which the variances of the Gaussian kernels were initialized to unity. Finally, all models were fitted to the data by maximizing their log-likelihood with the EM algorithm.

Information of the classes of the data samples was not utilized during learning. After learning the classification accuracy was measured by first estimating, based on the learning data, the probabilities $p(c|M_j, m_i^{(j)})$ that the Gaussian kernel i in expert j has generated a sample in class c. The classification of each sample $\mathbf{x}_k$ in the test set was then obtained by finding the maximum of $p(c|\mathbf{x}_k) = \sum_{j,i} p(c|M_j, m_i^{(j)}) p(M_j, m_i^j|\mathbf{x}_k)$ over the classes c. If the classification was different than the correct class the classification error was increased.

4.1 Data

The experiments were carried out on three different real-world datasets, with increasing number of samples and dimensionality.

The first set was from the machine learning data repository at the University of California at Irvine [2]. The 690 samples were obtained from Australian credit card applications which were classified to two groups: approved or rejected. The samples consisted of 14 attributes, eight of which were categorical; all attribute names had been changed to meaningless symbols to protect confidentiality of the data. Categorical values were encoded with 1-of-c coding. All data attributes were normalized to unit variance and zero mean before learning, because in its raw form the data was essentially one-dimensional (the first principal component explained over 99.9% of the variance).

The second dataset [7] consisted of 1550 samples of 20 cepstral-coefficient values of six Finnish vowels (/a/, /e/, /i/, /o/, /u/, /y/) picked up from continuous speech of one speaker. There were no missing values and we did not preprocess the data further.

The third dataset consisted of 5964 patent abstracts from four classes: optical computing devices, electrical digital data processing, analogue computers, and hybrid arrangements. The document vectors were formed using the standard vector space model [14] with additional weighting of the words according to an entropy-based measure of their importance (cf. [8]). Each one of the most frequently occurring words (in this case 957 words each of which occurred 50 times or more in the document database) was allocated one dimension of the data vector, and the value of the slot was set equal to the number of occurrences of the word in the document, times the weight of the word. The resulting vectors were normalized and the dimensionality of 957 was reduced to 50 using Latent Semantic Indexing (LSI) [3], also a standard procedure in the Information Retrieval community. In the LSI method a singular value decomposition of the data set is computed, and then the projections of the data on (in our case) the 50 most important singular vectors are formed.

4.2 Results

In the experiments each dataset was randomly partitioned into a training set and a test set with 90 % of the data used for training and 10% for testing. The process was repeated 10 times with 10 different choices of the training and test set (10-fold cross-validation). The reported results are the averages of these ten runs with 95% confidence intervals, computed assuming the values are normally distributed.

We compared the mixtures of autonomous experts with plain hierarchical Gaussian mixture models of the same size. The classification accuracy and (log) likelihood for the test set were computed as a function of the number of experts, while the total number of Gaussian kernels within the models remained the same. The 95% confidence intervals of the results were computed in order to assess the significance of the differences.

The results of the comparisons for the three data sets are shown in Figures 2, 3, and 4, respectively. For two of the data sets (Figs. 3 and 4) the likelihoods are clearly worse for the autonomous mixture, and for one (Fig. 2) the differences are within the confidence intervals. When measured by the classification accuracy the autonomous model is, however, about as good as the plain mixture model: For one of the data sets (Fig. 4) it is about as good, for one (Fig. 3) slightly worse, and for one (Fig. 2) better. All differences are within the confidence intervals, however.

We studied additionally the convergence properties of the models for the Australian credit card database. The log-likelihood for the training data was monitored as a function of the number of iterations of the EM algorithm for four experts having three Gaussian mixture components each. The means of the experts were initialized randomly: three samples were generated from each of the four randomly chosen Gaussians that corresponded to the autonomous experts. The variances in the model were initialized to unity. The results shown in Figure 5 suggest that both models converge about as quickly. The likelihood is better for the plain mixture of Gaussians but the classification accuracy is about the same, or even better for the autonomous mixture.

Finally, the effect of different schedules for updating the approximations the autonomous experts maintain of each other was studied. The

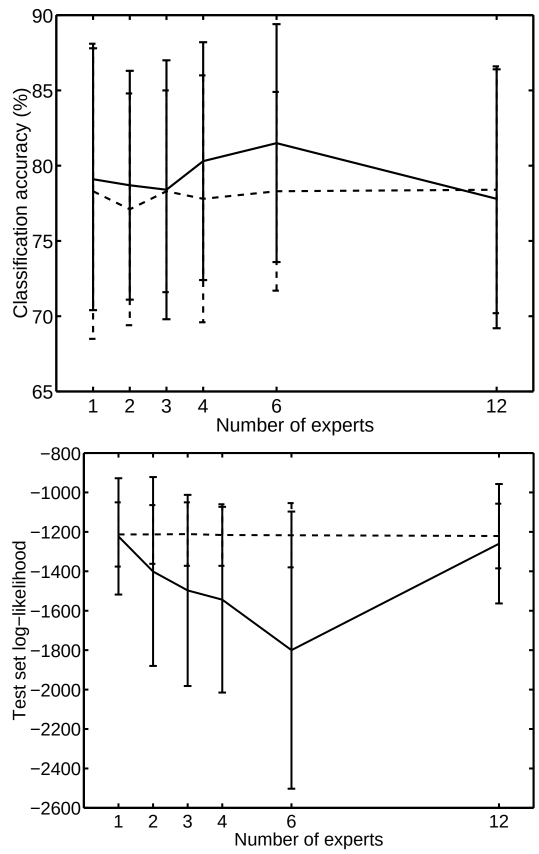

Figure 2: Comparison of plain Gaussian mixtures and mixtures of autonomous experts for the Australian credit card database. Top: Classification accuracy of the test set as a function of the number of experts. Bottom: (log) likelihood of the test set. The overall number of Gaussian kernels was fixed to 12, and the number of EM iterations was 120. Solid line: mixture of autonomous experts, dashed line: plain mixture of Gaussians. The bars show the 95% confidence intervals.

results are also shown in Figure 5, and they suggest that even updates after every 40 epochs are sufficient for convergence.

5 Conclusions

In this paper we introduced the principle of autonomous experts for mixture density modeling. Maximum likelihood learning procedures were derived for certain kinds of simple experts, mixtures of Gaussian densities. The mixtures approximate the ordinary mixture models in which the learning is not autonomous, and it was demonstrated in several experiments that the result was close to the “optimal” original model. More experiments are, however, needed to clarify the reasons for the differences in the results for different data sets.

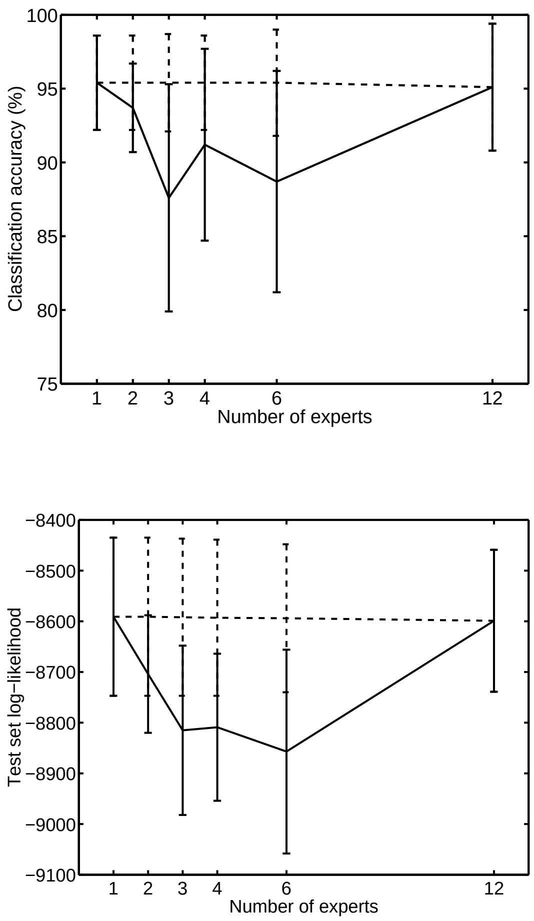

Figure 3: Comparison of plain Gaussian mixtures and mixtures of autonomous experts for the wowel database. The number of EM iterations was 24. For more details see Figure 2.

The model can be relatively easily generalized to other kinds of experts; the goodness of the resulting approximation depends on how well each expert can be approximated with simpler models maintained by the autonomous experts of each other.

It may also be possible to generalize the model to mixtures of predictors or classifiers that would be autonomous variants of the mixture of experts models [5, 6]. Finally, instead of the maximum likelihood learning procedure a Bayesian procedure that helps in model complexity selection could be utilized.

The learning procedure has an interesting interpretation that is perhaps most clearly visible in the stochastic gradient-based learning rule of the simplest Gaussian mixture model in which the densities $p(\mathbf{x}|M_j)$ in (1) are single Gaussians. Assume that the mixture learns with stochastic gradient descent, i.e., by moving in short steps towards the instantaneous gradient obtained for a single data sample $\mathbf{x}$. Then it is straightforward to show that the learning rule for the

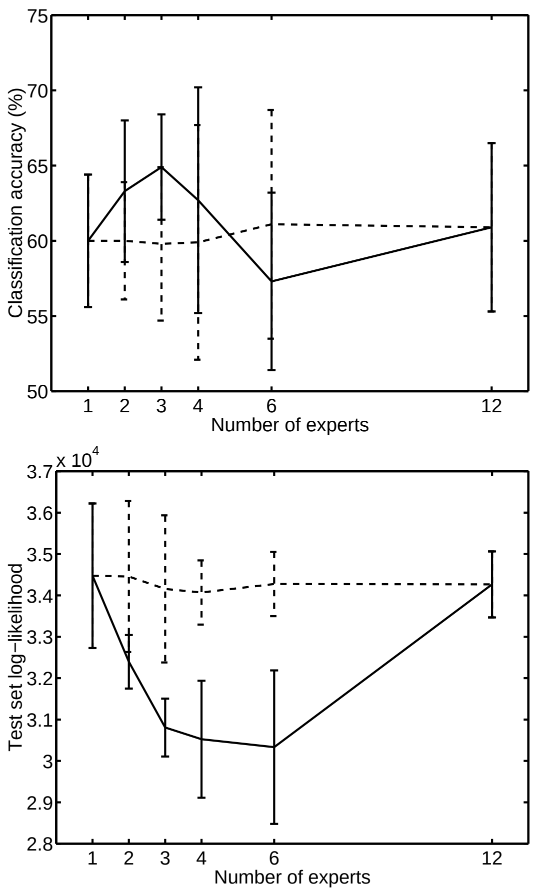

Figure 4: Comparison of plain Gaussian mixtures and mixtures of autonomous experts for the computer patent database. The number of EM iterations was 24. For more details see Figure 2.

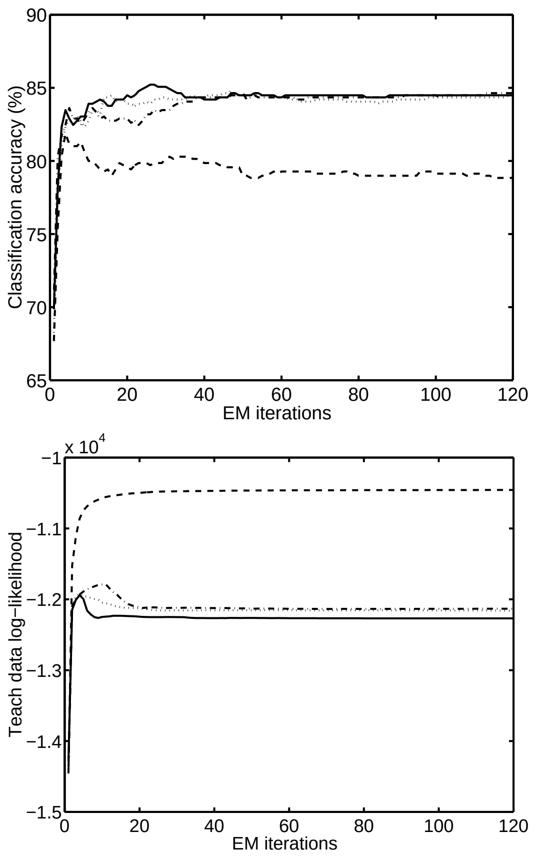

Figure 5: Convergence rates of the models. Top: classification accuracy for test data as a function of the number of EM iterations. Bottom: log likelihood for teaching data during learning. Dotted line: autonomous experts with the approximations of each other updated after every iteration. Solid line: updated every 10 iterations. Dash-dotted line: updated every 40 iterations. Dashed line: plain mixture of Gaussians.

means of the Gaussians becomes

$$\boldsymbol{\mu}_j(t+1) = \boldsymbol{\mu}_j(t) + \frac{\alpha(t)}{\sigma_j^2} p(M_j|\mathbf{x})(\mathbf{x} - \boldsymbol{\mu}_j(t)) \ , \quad (8)$$

where $\alpha(t)$ is the learning rate that decreases gradually according to the schedules of the stochastic approximation theory [9].

The update rule (8) can be interpreted so that the jth expert modulates its learning, i.e. movement towards the data, by its estimate of the probability $p(M_j|\mathbf{x})$ of being responsible for the data. It is intuitively appealing that the group of autonomous experts learns best if each expert evaluates how good it is in a task and modulates its learning according to the evaluation. The result can be considered as a generalization of competitive learning to a group of autonomous experts.

References

[1] C. M. Bishop, M. Svensén, and C. K. I. Williams. GTM: The generative topographic mapping. *Neural Computation*, 10(1):215–235, 1996.

[2] C. Blake and C. Merz. UCI Repository of machine learning databases. [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Irvine, CA: University of California, Department of Information and Computer Science, 1998.

[3] S. Deerwester, S. T. Dumais, G. W. Furnas, and T. K. Landauer. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41:391–407, 1990.

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

[5] R. Jacobs, M. Jordan, S. Nowlan, and G. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 1991.

[6] M. Jordan and R. Jacobs. Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, 6(2):181–214, 1994.

[7] T. Kohonen, J. Hynninen, J. Kangas, J. Laaksonen, and K. Torkkola. The learning vector quantization program package. Technical Report A30, Helsinki University of Technology, Laboratory of Computer and Information Science, Espoo, Finland, 1996.

[8] T. Kohonen, S. Kaski, K. Lagus, J. Salojärvi, J. Honkela, V. Paatero, and A. Saarela. Self organization of a massive document collection. *IEEE Transactions on Neural Networks*, 11:574–585, 2000.

[9] H. J. Kushner and G. G. Yin. *Stochastic Approximation Algorithms and Applications.* Springer, New York, 1997.

[10] G. J. McLachlan and K. E. Basford. *Mixture Models. Inference and Applications to Clustering.* Marcel Dekker, New York, NY, 1988.

[11] R. Neal. Probabilistic inference using markov chain monte carlo methods. Technical report, CRG-TR-93-1, 1993.

[12] M. Perrone and L. Cooper. When networks disagree : ensemble methods for hybrid neural networks. In R. Mammone, editor, *Artificial Neural Networks for Speech and Vision*, pages 126–142. Chapman & Hall, London, 1993.

[13] S. Roweis and Z. Ghahramani. A unifying review of linear gaussian models. *Neural Computation*, 11(2):305–345, 1999.

[14] G. Salton and M. J. McGill. *Introduction to modern information retrieval.* McGraw-Hill, New York, 1983.

[15] A. Utsugi. Hyperparameter selection for self-organizing maps. *Neural Computation*, 9(3):623–635, 1997.