A Self-Organizing Map for Clustering Probabilistic Models

Jaakko Hollmén †, Volker Tresp ‡ and Olli Simula †

Helsinki University of Technology †
Lab. of Comp. and Information Science
P.O. Box 5400, 02015 HUT, Finland
[Jaakko.Hollmen,Olli.Simula]@hut.fi

Siemens AG, Corporate Technology ‡
Information and Communications
81730 Munich, Germany
Volker.Tresp@mchp.siemens.de

Abstract

We present a general framework for Self-Organizing Maps, which store probabilistic models in map units. We introduce the negative log probability of the data sample as the error function and motivate its use by showing its correspondence to the Kullback-Leibler distance between the unknown true distribution of data and our empirical models. We present a general winner search procedure based on this probability measure and an update step based on its gradients. As an application, we derive the learning rules for a particular probabilistic model that is used in user profiling in mobile communications network. Due to the constrained nature of the parameters of our probabilistic model, we introduce a new parameter space, in which the gradient update step is performed. In the experiments, we show clustering of user profiles using calling data involving normal users of mobile phones and users that are known to be victims of fraud. In the summary, we discuss further applications of the approach.

1 Introduction

The Self-Organizing Map [1] [2] is one of the most popular neural network models for unsupervised learning. It has been successfully applied in clustering and visualization of high dimensional data. However, application of the standard Self-Organizing Map algorithm relies on the validity of the squared Euclidean distance between the map units and data as an error function. This condition is fulfilled for most cases involving feature vectors, but is not valid for parameters of probabilistic models, which are the focus of this paper.

In this paper, we derive a general framework for learning Self-Organizing Maps, which store probabilistic models in the map units. We show that minimizing the Kullback-Leibler distance between the unknown true generator of data and our empirical models leads to minimizing the negative log probability of the data with our empirical model. The winner search is based on this probability measure. The adaption step is based on the gradients of this probability measure with regard to the parameters of the probabilistic models.

We present an application of this framework to learning multiple user profiles in an unsupervised fashion from calling data from a mobile communications network. The models are generative probability density models, which express dynamic transitions of an observed variable *call*. Since the models are conditional densities, the parameters are constrained in a way which makes the direct application of gradient update unsuitable. Therefore, we introduce a new parameter space, in which the gradient update step takes place. The constrained parameters can be calculated with the softmax function. The learning process reinforces the constraints during the learning, so that the models are guaranteed to be proper.

A novel aspect of our work is the application of the Self-Organizing Map to dynamic probabilistic models. Previously, Kohonen had stated that implementation of selective responses to dynamic phenomena using the Self-Organizing Map is one of the more difficult problems. He defines a framework for such implementations by defining operator maps [2], where cells are tunable operators, usually filters for dynamic patterns. Kohonen sees as the central problem the tuning of such operators to an input sequence. Within this framework falls the work of Lampinen and Oja [3], who introduced a Self-Organizing Map algorithm for autoregressive models, where the map units consist of linear prediction coefficients of linear autoregressive models. Kangas [4] reports work with operator maps for analyzing speech data. Joutsensalo [5] combines the Self-Organizing Map learning rule with Oja's subspace rule for data compression and representing non-linear data distributions or manifolds. The latest work in this line is the Adaptive-Subspace SOM of Kohonen [6], which — with its modular architecture — learns to identify input patterns subject to some simple transformations. The common thread of these works is that models are stored in the map units of Self-Organizing Maps and that learning integrates adapting these models to data. All these papers use the Euclidean distance function as the error measure. Contrary to this, we develop a suitable error measure to be used with probabilistic models. Since we are learning conditional probability densities, the work involves constraints on the parameters of the probabilistic models, which must be reinforced during learning.

In Section 2, we briefly review the standard Self-Organizing Map algorithm and in Section 3 introduce the the general framework for learning Self-Organizing Maps, which have probabilistic models as map units. Also, a derivation of the learning rules on a user profiling problem is presented. We present experimental results with our approach in clustering dynamical probabilistic models. A Self-Organizing Map is used to model calling behavior in a mobile communications network. The clusters of models can be used in fraud detection [7], each model being a prototype of behavioral dynamics. In the Section 4, we summarize and present possible further applications.

2 The SOM Algorithm

The Self-Organizing Map is a neural network model for the analysis and visualization of high-dimensional data. It maps nonlinear statistical relationships between high-dimensional input data into simple geometric relationships on a regular, usually two-dimensional grid. It has been successfully applied for the analysis of industrial processes [8], [9], for example. For a bibliography on published papers, see [10].

The Self-Organizing Map is a collection of prototype vectors, between which a neighborhood relation is defined. This neighborhood relation defines a structured lattice, usually a two-dimensional, rectangular or hexagonal lattice of map units. Training a Self-Organizing Map from data is divided into two steps, which are applied alternately. First, a winner unit is searched, which minimizes the following Euclidean distance measure between data samples x and the map units m^k

$$c = \arg\min_k ||x - m^k||.$$

Then, the map units are updated in the *topological* neighborhood of the winner unit. The topological neighborhood is defined in terms of the lattice structure, not according to the distances between data samples and map units. The update step can be performed by applying

$$m^k(t+1) := m^k(t) + \alpha(t)h^c(t,k)[x(t) - m^k(t)]$$

where the last term in the square brackets is proportional to the gradient of the squared Euclidean distance $d(x, m^k) = ||x - m^k||^2$. The learning rate $\alpha(t) \in [0,1]$ must be decreasing function of time and the neighborhood function $h^c(t,k)$ is non-increasing function around the winner unit defined in the topological lattice of map units. A good candidate is a Gaussian around the winner unit defined in terms of the coordinates $\mathbf{r}$ in the lattice of neurons

$$h^c(t,k) = \exp(-\frac{||\mathbf{r}^k - \mathbf{r}^c||^2}{2\sigma(t)^2}).$$

During learning, the learning rate and the width of the neighborhood function are decreased, typically in a linear fashion. The map then tends to converge to a stationary distribution, which approximates the probability density of data.

The Self-Organizing Map may be visualized by using a unified distance matrix representation [11], where the clustering of the SOM is visualized by calculating distances between the map units locally and representing these with gray levels. Another choice for visualization is the Sammon's mapping [12], which projects the high-dimensional map units on a plane by minimizing the global distortion of inter point distances when applying the mapping.

3 Self-Organizing Map for Clustering Probabilistic Models

In our approach the map unit k stores the empirically estimated parameter vector θ^k with an associated probabilistic model $q(x\,;\theta^k)$. For implementing a Self-Organizing Map Algorithm, we need to define a distance between the map units (i.e. the θ^k) and data generating models. Euclidean distance between parameters of the probabilistic models is obviously a bad choice, since models which are very close in parameter space[1] might encode very different probabilistic models. For example, a prior probability of $P = 0$ for a disease might mean something dramatically different for a patient than a probability of $P = 0.01$. Also the distance between θ and a data point itself cannot be defined in a Euclidean way since they have different dimensionality. Therefore the natural choice for a distance measure in parameter space should be derived from the associated probabilistic models. The most common distance measure for this case is the Kullback-Leibler distance [13, 14]

$$KL(p\,||\,q) = -\int p(x)\log\frac{q(x\,;\theta^k)}{p(x)}dx. \quad (1)$$

The true distribution $p(x)$ is unknown but can be approximated by a Dirac unit impulse at the available data sample $p(x) \approx \delta(\mathbf{x}_i)$, which after substitution to Equation (1) gives us for the Kullback-Leibler distance the expression

$$-\log q(\mathbf{x}_i\,;\theta^k).$$

which is the negative log probability of data for our empirical model. 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 probability of the data with our empirical model. This justifies the use of this probability measure as a distance measure between models and data. In light of this derivation, we can derive a Self-Organizing Map algorithm for parametric probabilistic models. To illustrate the idea, we derive an algorithm for a specific case of user profiling in mobile phone networks.

3.1 Clustering User Profiles

In this paper we apply our approach to clustering user profiles described by calling behavior in a mobile communications network. The equation

$$P(x_t = j|x_{t-1} = i;\theta) = \theta_{ij}$$

denotes the transition of observed variables from one time step to next. The cases are the beginning of call ($x_{t-1} = 0, x_t = 1$), the end of the call ($x_{t-1} = 1, x_t = 0$), on-going call ($x_{t-1} = 1$, $x_t = 1$), and on-going silence ($x_{t-1} = 0, x_t = 0$). The calling pattern of a particular user is described by the parameter vector θ. The four parameters θ_{ij} have constraints $\theta_{ij} \in [0,1]$ and $\sum_j \theta_{ij} = 1$. The user profiles are learned in an unsupervised fashion from data.

3.1.1 Log Probability-based Winner Search

As shown earlier, minimizing the negative logarithm of the probability minimizes the Kullback-Leibler distance between the empirical and the true distribution. We define the winner search in terms of this probability measure, which gives us in our specific case the equation

$$\begin{aligned}-\log P(\mathbf{x}\,;\theta^k) &= -\frac{1}{T}[\log\prod_{t=1}^{T} P(x_t|x_{t-1};\theta^k)] \\ &= -\frac{1}{T}[\sum_{t=1}^{T}\log\,P(x_t|x_{t-1};\theta^k)].\end{aligned}$$

For distributions in the exponential family, the likelihood can be expressed in terms of the sufficient statistics. We may express the likelihood measure with sufficient statistics, which for a time-series $\mathbf{x} = \{x_1,\dots,x_T\}$ and our model, are the counts n_{ij} of the cases $x_t = j$ and $x_{t-1} = i$. To obtain invariance to the length of the time-series, we scale the n_{ij} to get the ratios $p_{ij} = \frac{n_{ij}}{T}$. This formulation allows us to estimate the p_{ij} in an on-line fashion, if necessary. The likelihood expression can be further manipulated by considering the sufficient statistics n_{ij} and the ratios p_{ij}

$$\begin{aligned}-\log P(\mathbf{x}\,;\theta^k) &= -\frac{1}{T}[\sum_{i,j=0}^{1} n_{ij}\log\,P(x_t|x_{t-1};\theta^k)] \\ &= -\sum_{i,j=0}^{1} p_{ij}\log\,\theta^k_{ij}.\end{aligned}$$

The winner search procedure looks for the best-matching map unit $P(x_t|x_{t-1};\theta^c)$ among all the map units, in our approach the best statistical fit of the model to data. The winner unit may

[1]In Euclidean sense

be defined with the equation

$$c = \arg\min_k [- \sum_{i,j=0}^{1} p_{ij} \log \theta^k_{ij}]. \qquad (2)$$

Conditions on the p_{ij} and proper θ^k_{ij} ensure that the distance is always greater than or equal to zero.

3.1.2 Constrained Update Rules

The form of update step is a gradient update of the form

$$\theta^k(t+1) := \theta^k(t) + \alpha(t) h^c(t,k) \frac{\partial \log P(\mathbf{x}(t); \theta^k)}{\partial \theta^k}$$

for the map units in the topological neighborhood of the winner unit. Since the models are conditional densities and involve constraints discussed earlier, the direct application of gradient descent procedure on updating the parameters does not enforce the constraints. If the constraints are not satisfied during learning, correct calculation of likelihood is not possible. Therefore, we need an update rule that takes into account the constraints between the parameters. To solve this problem, we introduce unconstrained parameters w^k_{ij}, which map to our constrained parameter space θ^k_{ij} by applying the softmax layer. The softmax layer [13, 14] is defined as

$$\theta^k_{ij} = \frac{\exp(w^k_{ij})}{\sum_j \exp(w^k_{ij})}. \qquad (3)$$

The gradients of the negative log probability with regard to the free parameters can be calculated with the chain rule of differentiation. After algebraic manipulation we get for our probability-based error and the our model[2]

$$\begin{aligned} \frac{\partial \log P(\mathbf{x}(t); \theta^k)}{\partial w^k_{ij}} &= \sum_l \frac{\partial \log P(\mathbf{x}(t); \theta^k)}{\partial \theta^k_{il}} \frac{\partial \theta^k_{il}}{\partial w^k_{ij}} \\ &= p_{ij}(1 - o_{ij}) - p_{i\neg j} o_{ij} \end{aligned}$$

where the o_{ij} is the output of the softmax layer as a function of the free parameters $w_{ij}, j = \{0,1\}$. Note that the update of the parameter w_{ij} involves also counts p_{ij} and $p_{i\neg j}$.

The update term for the winner unit and its topological neighbors is now

$$\Delta w^k_{ij}(t) = \alpha(t) h^c(t,k) \sum_l \frac{\partial \log P(\mathbf{x}(t); \theta^k)}{\partial \theta^k_{il}} \frac{\partial \theta^k_{il}}{\partial w^k_{ij}}.$$

[2] $\neg$ is the negation operator, $\neg 1 = 0$, $\neg 0 = 1$

The constrained parameters can then readily be calculated by applying the softmax function to the free parameters.

3.2 Experiments

Hollmén and Tresp used a hierarchical regime-switching model to detect fraud in a mobile communications network [7]. Their model involved a probability density for the observed time-series as presented in the previous section. Their regime-switching model, however, involved only one probability density for each type of behavior (fraud and non-fraud). The probability density model for the observed variable *call* models the transitions from the observed state of no calling and calling to the same state at the next time step.

The methods presented in this paper enable learning multiple models of user behavior in an unsupervised fashion from data. In the experiments, we model the different dynamics of calling behavior by learning a Self-Organizing Map that stores these models in the map units.

The data are call detail records from calls made with mobile phones. The data set contains 600 normal users, each with calling data for 49 days and 302 fraudulent users, each with calling data for 92 days. The data set was divided into two sets for training and testing. The data are represented in the form of a time series, each value indicating whether a mobile phone was used or not during a particular minute. The data was sampled with a rate of one minute. Sufficient statistics n_{ij} and the ratios p_{ij} were counted for the time series. An example time series is presented in Figure 1.

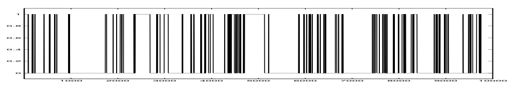

Figure 1: The data for one user is shown. The time period on the horizontal axis is one week. The estimated models express the probability of transitions between the states of no calling (zero) and calling (one).

With this data, we trained a Self-Organizing Map of 80 units (8 $\times$ 10 map units). The topology between the map units was chosen to be rectangular. The neighborhood function used in the experiments was Gaussian. The map was trained on the sufficient statistics of the time series, which can be used to calculate the negative log probabilities.

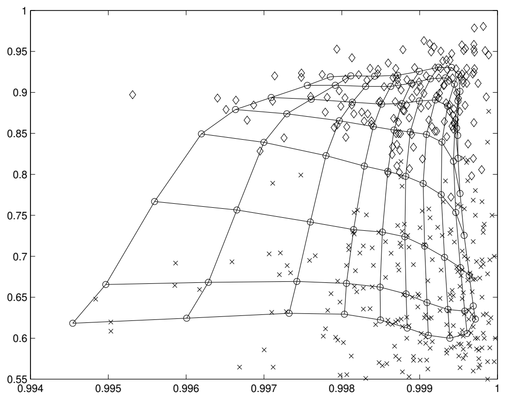

Figure 2: Two parameters θ_{00} (horizontal axis), θ_{11} (vertical axis) of the models stored in map units of the Self-Organizing Map are shown with circles. The rectangular lattice structure is shown with lines between map units. The parameters for training data involving fraud users are shown with diamonds, normal users with cross marks. Parameters in the upper areas of the figure correspond to long calls, while the parameters in the rightmost area correspond to infrequent calls.

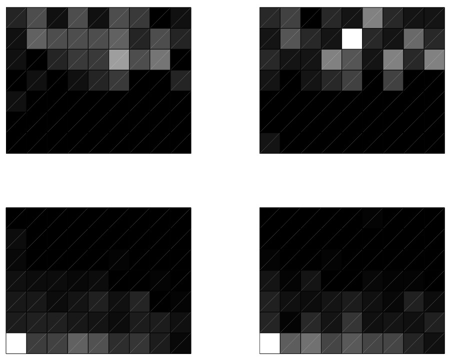

Figure 3: The training set (left column) and testing set (right column) have been mapped to map units and are represented as hit counts. The lighter the map unit, the more data samples were mapped to it by the winner search procedure. In the upper row are the fraud users and in the lower row the normal users. We can see that the class-specific distributions are well separated by the map and that therefore the map can be the basis for the analysis of future calling patterns.

The training of the map was performed in two training steps, first a coarse ordering[3] and thereafter fine-tuning[4]. For both training rounds, the learning rate was linearly decreased to zero during learning. The σ of the Gaussian neighborhood function was decreased linearly to one. Two parameters of the trained map can be seen in Figure 2. The response of the map to class-specific distributions is visualized in Figure 3. The figure shows that the upper parts of the map have become sensitive to fraud users, whereas normal users are mapped to lower parts of the map by the winner search.

4 Summary

We presented a general framework for clustering probabilistic models with the Self-Organizing Map algorithm. The standard Euclidean distance in the winner search was replaced with a probability measure, which was shown to correspond to the Kullback-Leibler distance between the true and the empirical model. In the update step, the gradients of the negative log probability measure are used. As an application, we derived the learning algorithms for specific probability distribution functions to be used in user profiling of mobile phone users. Due to the constrained form of the parameters, additional parameterization and a softmax function were introduced to reinforce the constraints during learning.

The ideas presented here may be used for deriving a modified version of the Learning Vector Quantization (LVQ) algorithm [2], which could be used to learn a time-series classifier in a supervised fashion. Similarly, the ideas concerning the constraints on the parameters may be applied in situations, where the features have intrinsic constraints and are represented as proportions. Such a development could be coined as Self-Organizing Map of pie charts. Additionally, in order to visualize the clusters of probabilistic models with Sammon's mapping [12], the discrepancy measure should be modified to account for the Kullback-Leibler -based distances in the original space and the squared Euclidean distance in the low-dimensional display. Furthermore, it would be interesting to cluster models involving latent variables, such as mixture models [15] or hidden Markov models [16] or to train their component densities with the Self-Organizing Map algorithm presented in this paper and compare the results with the standard mixture density approach and the use of the EM algorithm [17]. It would be in-

[3]$\alpha(0) = 0.9$, $\sigma(0) = 5$, 10000 iterations
[4]$\alpha(0) = 0.1$, $\sigma(0) = 2$, 10^6 iterations

teresting to consider the possible relationships of the resulting set of parameters and samples from a Bayesian posterior distribution.

Acknowledgments

The first author was funded by Siemens AG. He wishes to thank Professor Olli Simula for supervising the thesis work and Professor Bernd Schürmann and Michiaki Taniguchi, both at Siemens AG, Corporate Technology for continuing support. The first author also thanks Suvi and Risto for support.

References

[1] Teuvo Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, September 1990.

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

[3] J. Lampinen and E. Oja. Self-organizing maps for spatial and temporal AR models. In *Proc. 6 SCIA, Scand. Conf. on Image Analysis*, pages 120–127, 1989.

[4] Jari Kangas. *On the Analysis of Pattern Sequences by Self-Organizing Maps*. PhD thesis, Helsinki University of Technology, 1994.

[5] Jyrki Joutsensalo. Nonlinear data compression and representation by combining self-organizing map and subspace rule. In *Proc. ICNN'94, Int. Conf. on Neural Networks*, pages 637–640, Piscataway, NJ, 1994. IEEE Service Center.

[6] Teuvo Kohonen, Samuel Kaski, and Harri Lappalainen. Self-organized formation of various invariant-feature filters in the adaptive-subspace SOM. *Neural Computation*, 9:1321–1344, 1997.

[7] Jaakko Hollmén and Volker Tresp. Call-based fraud detection in mobile communications networks using a hierarchical regime-switching model. In M. Kearns, S. Solla, and D.A. Cone, editors, *Advances in Neural Information Processing Systems: Proceedings of the 1998 Conference (NIPS'11)*, 1999.

[8] T. Kohonen, E. Oja, O. Simula, A. Visa, and J. Kangas. Engineering applications of the self-organizing map. *Proceedings of the IEEE*, 84(10):1358–84, 1996.

[9] Esa Alhoniemi, Jaakko Hollmén, Olli Simula, and Juha Vesanto. Process monitoring and modeling using the self-organizing map. *Integrated Computer Aided Engineering*, 6(1):3–14, 1999.

[10] Samuel Kaski, Jari Kangas, and Teuvo Kohonen. Bibliography of self-organizing map (SOM) papers: 1981-1997. *Neural Computing Surveys*, 1:102–350, 1998.

[11] A. Ultsch and H. Siemon. Kohonen's self-organizing maps for exploratory data analysis. In *Proceedings of the International Neural network Conference (INNC'90)*, pages 305–308. Kluwer, 1990.

[12] John W. Sammon Jr. A nonlinear mapping for data structure analysis. *IEEE Transactions on Computers*, C-18(5):401–409, May 1969.

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

[14] Brian D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, 1996.

[15] R.A. Redner and H.F. Walker. Mixture densities, maximum likelihood and the EM algorithm. *SIAM Review*, 26(2):195–234, 1984.

[16] 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.

[17] 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.