Self-Organization by Optimizing Free-Energy

J.J. Verbeek , N. Vlassis , B.J.A. Kröse

University of Amsterdam, Informatics Institute
Kruislaan 403, 1098 SJ Amsterdam, The Netherlands

Abstract. We present a variational Expectation-Maximization algorithm to learn probabilistic mixture models. The algorithm is similar to Kohonen's Self-Organizing Map algorithm and not limited to Gaussian mixtures. We maximize the variational free-energy that sums data log-likelihood and Kullback-Leibler divergence between a normalized neighborhood function and the posterior distribution on the components, given data. We illustrate the algorithm with an application on word clustering.

Keywords: self-organizing map, mixture modeling, variational EM.

1 Introduction

Kohonen's Self-Organizing Map (SOM) [5] is a data analysis method that combines vector quantization with topology preservation. With each quantizer we associate a *fixed* location in a 'latent' space. The latent space is of much lower dimension (typically just two) than the data space. The SOM algorithm finds locations for the quantizers in the data space such that the summed squared distance from data to closest node is small and simultaneously topology is preserved. Topology preservation means that nearby components in latent space are also nearby in the data space. As a consequence, data points associated with nearby components in latent space are coming from similar locations in the data space. Hence, to check if data points are 'close' in the data space, we can check whether their associated components are close in the latent space. Topology preservation makes SOMs useful for visualization of high dimensional data since it reduces the data dimensionality while preserving small distances.

We present a constrained or variational Expectation-Maximization (EM) learning algorithm to learn probabilistic mixture models similar to SOM. The algorithm applies to a wide class of mixture models.

In the next section we discuss a Gaussian mixture model that will serve as the example mixture model throughout the paper and also briefly discuss the EM algorithm. In section 3 we present our Self-Organizing Mixtures algorithm. We compare our algorithm to several closely related algorithms and present our conclusions in Section 4.

2 A simple generative model and EM

As generative model consider the Mixture of Gaussians (MoG), given by :

$$p(\mathbf{x}) = \frac{1}{k}\sum_{s=1}^{k} p(\mathbf{x} \mid s) \text{ for } \mathbf{x} \in \mathrm{I\!R}^{\mathrm{D}},$$

where $p(\mathbf{x}|s)$ is an isotropic Gaussian with inverse variance β and mean $\boldsymbol{\mu}_s$. We use this simple MoG to illustrate the similarity with SOM and because it keeps notation simple. Given data $\mathbf{x} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$, and an initial parameter vector $\boldsymbol{\theta} = \{\beta, \boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_k\}$ the EM algorithm finds a local maximizer $\boldsymbol{\theta}$ of the log-likelihood $\mathcal{L}(\boldsymbol{\theta})$. Assuming data are independent and identically distributed:

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{n=1}^{N} \log p(\mathbf{x}_n; \boldsymbol{\theta}) = \sum_n \log \frac{1}{k} \sum_s p(\mathbf{x}_n \mid s).$$

Learning is facilitated by introducing N *hidden* variables. Each hidden variable indicates which of the k mixture components generated the corresponding data point. The EM algorithm maximizes the negative free-energy [6]:

$$F(Q, \boldsymbol{\theta}) = \mathrm{E}_Q \log p(\mathbf{x}, s; \boldsymbol{\theta}) + \mathcal{H}(Q) = \mathcal{L}(\boldsymbol{\theta}) - D_{KL}(Q \parallel p(s \mid \mathbf{x}; \boldsymbol{\theta})), \quad (1)$$

where we used $\mathcal{H}$ and D_{KL} to denote respectively entropy and Kullback-Leibler (KL) divergence $Q = \prod_n q_n$ is a distribution over the hidden variables, q_n is a distribution on the mixture components for data point n. Note that, due to the non-negativity of the KL-divergence, for *any* Q we have: $F(Q, \boldsymbol{\theta}) \leq \mathcal{L}(\boldsymbol{\theta})$.

The two forms in which we expanded F are associated with the M-step and the E-step of EM. In the M-step we change $\boldsymbol{\theta}$ as to maximize, or at least increase, $F(Q, \boldsymbol{\theta})$. The first decomposition includes $\mathcal{H}(Q)$ which is a constant w.r.t. $\boldsymbol{\theta}$, so we essentially maximize the joint log-likelihood in the M-step. In the E-step we maximize F w.r.t. Q. Since the second decomposition includes $\mathcal{L}(\boldsymbol{\theta})$ which is constant w.r.t. Q, what remains is a KL divergence which is the effective objective function in the E-step of EM.

In standard applications of EM (e.g. mixture modeling) Q is unconstrained, which results in setting $q_n = p(s \mid \mathbf{x}_n; \boldsymbol{\theta})$ in the E-step since the non-negative KL divergence equals zero if and only if both arguments are equal. Therefore, after each E-step $F(Q, \boldsymbol{\theta}) = \mathcal{L}(\boldsymbol{\theta})$. Variational methods are for example used when optimization over the unconstrained Q is intractable, in which case Q is restricted to a certain class $\mathcal{Q}$ of distributions allowing for tractable computations. Variational EM maximizes F instead of $\mathcal{L}$, the objective sums log-likelihood and a penalty which is high if the true posterior is far from any member of $\mathcal{Q}$.

3 Using free-energy for self-organization

By appropriately constraining Q we can enforce a topological ordering of the mixture components, as we explain below. By using the right class $\mathcal{Q}$, the penalty term measures topological ordering. This is much like the approach taken in [7], where constraints on Q are used to globally align the local coordinate systems of the components of a probabilistic mixture of factor analyzers.

We associate with each mixture component s a latent coordinate $\mathbf{g}_s$. It is convenient to take the components as located on a regular grid in the latent space. We then restrict the distributions q_n to be discretized isotropic Gaussians in the latent space, centered on one of the components. The distributions q_n can be regarded as normalized *neighborhood function* of Kohonen's SOM. We thus put the constraint that $q_n \in \mathcal{Q} = \{p_1, \ldots, p_k\}$, where:

$$p_r(s) = \frac{\exp\left(-\lambda \parallel \mathbf{g}_s - \mathbf{g}_r \parallel^2\right)}{\sum_t \exp\left(-\lambda \parallel \mathbf{g}_t - \mathbf{g}_r \parallel^2\right)}.$$

A small λ corresponds to a broad distribution, and for large λ the distribution p_r becomes more peaked. For fixed λ we can always increase or keep constant the objective by performing EM steps, since in the E-step we can always keep Q fixed or change Q if this increases the objective function.

Since the objective function might have local optima and EM is only guaranteed to give locally optimal solutions, good initialization of the parameters of the mixture model is essential to finding a good solution. Analogous to the method of shrinking the extent of the neighborhood function with the SOM, we can start with a small λ (broad neighborhood function) and increase it iteratively until a desired value is reached. In implementations we started with λ such that the p_r are close to uniform over the components, then we run the EM algorithm until convergence. Note that if the p_r are almost uniform the initialization of $\boldsymbol{\theta}$ becomes irrelevant. After convergence we set $\lambda^{new} \leftarrow \eta\lambda^{old}$ with $\eta > 1$ (typically η is close to unity). In order to initialize the EM procedure with λ^{new}, we initialize $\boldsymbol{\theta}$ with the value found in running EM with λ^{old}.

Using $q_{ns} = q_n(s)$, for our MoG case F can be rewritten as:

$$F(Q, \boldsymbol{\theta}) = \frac{1}{2} ND \log\beta - \sum_{ns} q_{ns} \Big[\beta \parallel \mathbf{x}_n - \boldsymbol{\mu}_s \parallel^2 /2 + \log q_{ns}\Big].$$

For fixed λ an EM algorithm can be derived by differentiation:

- **E:** Determine (by means of exhaustive or sparse search in $\mathcal{Q}$, see below) for each $\mathbf{x}_n$ the distribution $p_{r^*} \in \mathcal{Q}$ that maximizes F, set $q_n = p_{r^*}$.
- **M:** Set: $\boldsymbol{\mu}_s = \sum_n q_{ns}\mathbf{x}_n / \sum_n q_{ns}$ and $\beta = ND / \sum_{ns} q_{ns} \parallel \mathbf{x}_n - \boldsymbol{\mu}_s \parallel^2$.

The component r^* on which p_{r^*} is centered for data point n, is referred to as the 'winner' for $\mathbf{x}_n$.

The computational cost of the E-step is $O(Nk^2)$, a factor k slower than SOM and prohibitive in large-scale applications. However, by restricting the search for a winner in the E-step to a limited number of candidate winners we can obtain an $O(Nk)$ algorithm. A straightforward choice is to use the l components with the largest joint likelihood $p(\mathbf{x}, s)$ as candidates, corresponding for our MoG to smallest Euclidean distance to the data point. If none of the candidates yields a higher value of $F(Q, \boldsymbol{\theta})$ we keep the winner of the previous step, in this way we are guaranteed to increase the objective in every step. We found $l = 1$ to work well and fast in practice, in this case we only check whether the winner from the previous round should be replaced with the closest node.

Let us consider why our algorithm yields topology preservation. Consider the second decomposition of the objective function (1), and two mixtures, parameterized by $\boldsymbol{\theta}$ and $\boldsymbol{\theta}'$, that yield equal data log-likelihood $\mathcal{L}(\boldsymbol{\theta}) = \mathcal{L}(\boldsymbol{\theta}')$. The objective function prefers the mixture for which the KL-divergences are the smallest, i.e. the mixture for which the posterior on the mixture components looks like a discretized Gaussian in the latent space. This implies that nearby components in the latent space model similar data. Moreover, we see the same effect as in SOM: for each data item a winning mixture component is selected and next the winning component and its neighbors in the latent space (to a lesser degree) are tuned toward that data item. Note that these observations hold for any mixture model and that the algorithm is readily generalized to any mixture model for which we have a maximum-likelihood EM algorithm.

4 Discussion, illustration, and conclusions

Discussion. Our algorithm is very close to SOM when applied on the simple MoG. If we use only $l = 1$ candidate winner in the E-step the difference with Kohonen's winner selection is that *we only accept the closest node as a winner when it increases the energy and keep the previous winner otherwise*. Our M-step coincides exactly with the update rule of the batch SOM.

In [3, 4] another SOM-like algorithm is proposed with objective function:

$$-\sum_{ns} q_{ns}[\beta \sum_{r} h_{sr} \parallel \mathbf{x}_n - \boldsymbol{\mu}_r \parallel /2 + \log q_{ns}].$$

There, the neighborhood function, implemented by the h_{sr}, is fixed, but the winner assignment is soft. Instead of selecting one 'winner', an *unconstrained* distribution q_{ns} over the components is used. The β is used for annealing: for very small β the entropy term, with only one *global* optimum, becomes dominant, whereas for large β the quantization error, with many local optima, becomes dominant. By gradually increasing β more and more structure is added to the objective function.

Our work differs in several ways. We use *one localized distribution* in the latent space to constrain q_n, as opposed to a *mixture of localized distributions* in the latent space. As a consequence the easy speed-up of our algorithm does not apply to the algorithms in [3, 4]. We use the neighborhood function width (controlled by λ) for annealing as opposed to using β. Note that although both λ and β can be used for annealing, only β can be optimized efficiently as a free parameter. Both our objective function and the one of [3, 4] can be interpreted as a log-likelihood plus a penalty term for non-topology preserving configurations. We believe the interpretation provided in this work is more direct and applies trivially to any mixture model, whereas for the interpretation of [3, 4] it is not clear whether it applies to any mixture model.

Another similar model is presented in [1]. There, in the E-step we set:

$$q_n = \arg\max_r \sum_s p_r(s)p(\mathbf{x}_n|s).$$

The M-step finds a new parameter vector that maximizes:

$$\sum_n \log \sum_s q_n(s)p(\mathbf{x}_n|s).$$

This algorithm does not optimize a single likelihood function, even if we keep the neighborhood function width fixed, since the mixing weights vary for each data item and change throughout the iterations of the algorithm. The algorithm has run-time $O(nk^2)$, but can benefit from the same speed-up used here.

The Generative Topographic Map (GTM) [2] achieves topographic organization in a quite different manner. In GTM mixture components are parameterized by a linear combination of nonlinear basis functions of the locations of the components in the latent space. The parameters of the linear map are learned from data by maximizing the data log-likelihood. The number of basis functions and their smoothness have to be fixed in advance or are found by model selection procedures. Other maximum-likelihood approaches achieve topographic organization by a smoothness prior [8] on the parameters. However, for this method is it not clear whether it generalizes directly any mixture model and whether speed-ups can be applied to avoid $O(nk^2)$ run-time.

Illustration. Figure 1 shows results of modeling word occurrence data on a part of the 20 newsgroup data set, word $\mathbf{x}_n$ is shown at $\mathbf{g}_n = \sum_s p(s|\mathbf{x}_n)\mathbf{g}_s$. Each of the 100 words is a data item with 16242 binary features indicating its occurrence in each of 16242 documents. We learned a mixture with our algorithm, where each mixture component is a product of Bernoulli distributions:

$$p(\mathbf{x}_n|s) = \prod_{i=1}^{16242} {p_{s,i}}^{x_n^i}(1-p_{s,i})^{(1-x_n^i)}.$$

Due to the huge dimensionality of this data, the posteriors are rather peaked, resulting in $\mathbf{g}_n$ almost equal to one of the $\mathbf{g}_s$. Therefore, we did not use the actual posterior but the distribution $p'(s|\mathbf{x}_n) \propto p(s|\mathbf{x}_n)^\alpha$ for $\alpha < 1$ such that the entropies of the $p'(s|\mathbf{x}_n)$ were close to two bits, giving a smoother visualization.

Conclusions. We presented a penalized log-likelihood probabilistic mixture modeling method, similar to Kohonen's SOM. The probabilistic formulation offers several benefits: (i) The method is directly applicable to any mixture model for i.i.d. data for which we can write down an EM algorithm. (ii) The generative model allows seamless embedding in larger probabilistic systems. It is for example straightforward to learn mixtures of Self-Organizing maps. (iii) There is a direct and clear link to the mixture model data log-likelihood.

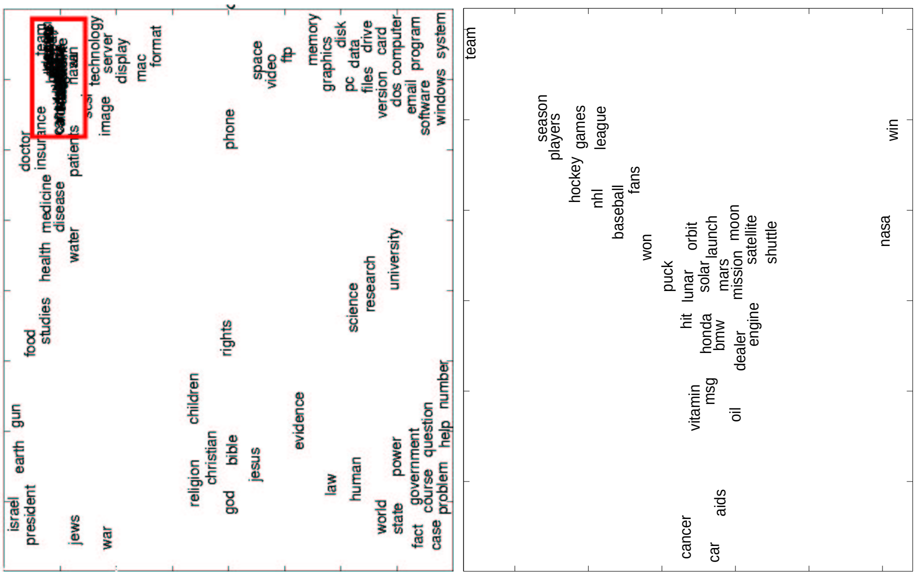

Figure 1: Self-organizing Bernoulli models, $k = 25$. Right plot zooms left.

References

[1] A. Anouar, F. Bedran, and S. Thiria. Probabilistic self organized map: application to classification. In M. Verleysen, editor, *Proceedings of European Symposium on Artificial Neural Networks*, Belgium, 1997. D-Facto.

[2] C. M. Bishop, M. Svensén, and C. K. I Williams. GTM: The generative topographic mapping. *Neural Computation*, 10:215–234, 1998.

[3] T. Graepel, M. Burger, and K. Obermayer. Self-organ. maps: generalizations and new optimiz. techniques. *Neurocomputing*, 21:173–190, 1998.

[4] T. Heskes. Self-organizing maps, vector quantization, and mixture modeling. *IEEE Transactions on Neural Networks*, 12:1299–1305, 2001.

[5] T. Kohonen. *Self-Organizing Maps*. Springer, 2001.

[6] R. M. Neal and G. E. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. In M.I. Jordan, editor, *Learning in Graphical Models*, pages 355–368. Kluwer, 1998.

[7] S.T. Roweis, L.K. Saul, and G.E. Hinton. Global coordination of local linear models. In T.G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*. MIT Press, 2002.

[8] A. Utsugi. Density estimation by mixture models with smoothing priors. *Neural Computation*, 10(8):2115–2135, 1998.