Probabilistic measures for responses of Self-Organizing Map units

Esa Alhoniemi, Johan Himberg, and Juha Vesanto
Helsinki University of Technology
Laboratory of Computer and Information Science
P.O. Box 5400, FIN-02015 HUT, Finland
email {Esa.Alhoniemi,Johan.Himberg,Juha.Vesanto}@hut.fi

Abstract

The Self-Organizing Map (SOM) is a widely used data visualization tool in engineering applications. The algorithm performs a non-linear mapping from a high-dimensional data space to a low-dimensional space, which is typically a two-dimensional, rectangular grid. This makes it possible to present multidimensional data in two dimensions.

Often the model vectors of the SOM and a new data sample need to be compared. The SOM, however, gives no probability measures to determine if the sample belongs to data sets determined by map units. For this purpose a modified batch version of reduced kernel density estimator (RKDE) was tested. The results were compared with Gaussian Mixture Model (GMM) and S-Map.

1 Introduction

The SOM algorithm [6] produces a topology preserving mapping from high-dimensional data space to a low-dimensional grid of map units. It also carries out vector quantization by representing the input vectors using model vectors of the map units.

The SOM network can be used to investigate and to visualize the structure and dependencies in multivariate data, for example process states [1, 7, 9]. In many applications the best-matching unit (BMU), the map unit with model vector closest to a sample vector, is important. However, following problems exist:

- the sample may be far from the closest model vector (problem of novelty detection) or
- there may exist several (almost) equidistant model vectors.

These problems may be avoided by using *response*, a function of the distance between model vector and data sample $\mathbf{x}$. A suitable functional form for the response r_i of unit i is for example

$$r_i = \frac{1}{1 + ||\mathbf{x} - \mathbf{m}_i||^2}, \; i = 1, \ldots, N, \qquad (1)$$

where $\mathbf{m}_i$ is the model vector of unit i and N is the number of units. This formula has some beneficial properties, e.g. $r_i \in [0, 1]$. Thus r_i can be considered as a "fuzzy indicator" of the data sample hit for the map unit i.

However, probabilistic interpretation of the response in Eq. 1 is not clear. The response may alternatively be defined, using the Bayes' theorem, as conditional probability of unit i given data sample $\mathbf{x}$:

$$r_i = P(i|\mathbf{x}) = \frac{p(\mathbf{x}|i)P(i)}{p(\mathbf{x})} = \frac{p(\mathbf{x}|i)P(i)}{\sum_i p(\mathbf{x}|i)P(i)}. \qquad (2)$$

The priors $P(i)$ and the conditional densities $p(\mathbf{x}|i)$ need to be estimated. For estimating the conditional densities we adopt the idea that the model vectors generate the data distribution. This leads to the batch version of RKDE using SOM model vectors as kernel centers.

2 Description of the methods

2.1 Gaussian mixture model

In Gaussian Mixture Model (GMM), the modeled data is assumed to be generated by a set of Gaussian distributions. The parameters of the Gaussians are determined using the Expectation-Maximization (EM) algorithm [3].

Initial guesses for the distribution centers were in our tests computed using the K-means algorithm. The simplest way to form a GMM is to compute the covariance matrix for each cluster without any iterations of the EM algorithm, which is fast to compute but typically produces poor results.

In kernel estimation, diagonal forms of the covariance matrix of each cluster are often used, because inside a cluster the variables can be assumed to be independent. Another reason for using diagonal form — even though variables would not be totally independent — is that estimation of full covariance matrix requires lots of data which are not in many cases available.

In the GMM simulations, Netlab software package by Nabney and Bishop [8] was used.

2.2 Self-organizing reduced kernel density estimator

A straightforward idea to estimate the probability density using the SOM is to build a RKDE model using the map model vectors as kernel centers. The method was proposed by Hämäläinen [4]. We used a modified batch version of the method.

A simple way to estimate the prior probability for map unit i is ratio

$$P(i) = \frac{\#(\mathbf{x}_n \in V_i)}{\#\mathbf{x}_n},\ n = 1, ...M, \tag{3}$$

where V_i is the Voronoi set of unit i (set of data vectors whose closest model vector is $\mathbf{m}_i$) and M is the number of samples.

The density functions $p(\mathbf{x}|i)$ for each map unit are estimated using corresponding Voronoi sets V_i. Unfortunately this way a biased data set is obtained because no samples can lie outside the Voronoi region. The parameter estimation may be adjusted in several ways. Due to the topological ordering of the SOM, the neighborhood function can be used to get a weighted contribution of data from the neighboring units; also the priors $P(i)$ may be computed this way. In the distribution modeling following assumptions are made:

- variables in the sets V_i, $i \in 1, \dots, N$ (and in the Voronoi sets of neighbors of i) are independent and
- distributions $p(x^l|i)$, $l = 1, \dots, d$ (d is the data dimension) are Gaussian.

This approach differs from the method by Hämäläinen in two ways: in his work, the priors (kernel weights) were not adjusted and the kernel width was determined in a different way.

In the SOM experiments the training of the network was carried out using SOM Toolbox[1].

2.3 S-Map

Kiviluoto and Oja have recently proposed the S-Map algorithm [5], which has an inherently probabilistic background. In S-Map the softmax activations of the Generative Topographic Mapping (GTM) [2] and the learning algorithm of the SOM are combined. According to tests by Kiviluoto and Oja the S-Map seems to have better self-organizing capability than GTM and it is computationally lighter.

3 Experimental results

3.1 Test data

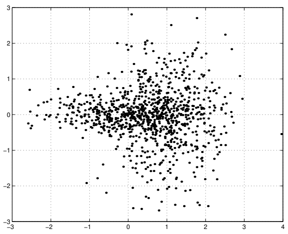
Three data sets for testing the algorithms were generated. The parameters of the sets are shown in Table 1. Because we wanted to reconstruct the distributions using kernel estimators and compare them with the original ones, the data sets were generated using known distributions.

Set #	Dim	Centers	Covariance matrices
I	2	(0,0) (1,0)	diag(1,0.1) diag(1,0.1)
II	3	(0,0,0) (1,0,0) (2,2,2)	diag(1,0.1,0.5) diag(0.5,1,1) diag(0.1,0.1,0.1)
III	4	(0,0,0,0) (1,1,0,0) (2,0,2,0) (-1,-1,0,-1) (0,0,2,0) (-1,2,0,1)	diag(1,0.1,0.5,1) diag(0.5,1,1,0.5) diag(0.1,0.1,0.1,0.1) diag(1,1,0.1,0.1) diag(0.1,1,0.1,0.5) diag(0.5,0.1,1,0.1)

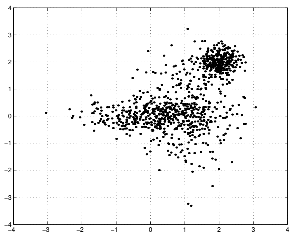
Table 1: The parameters of the distributions. Data set I had two, set II three and set III six Gaussian kernels. Notation "diag($\sigma_1^2, \dots, \sigma_N^2$)" refers to diagonal covariance matrix with variances $\sigma_1^2, \dots, \sigma_N^2$.

[1] Freely available at URL `http://www.cis.hut.fi/projects/somtoolbox/`

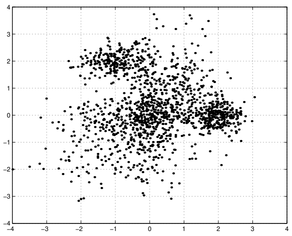
Plots of 1000 randomly sampled points from the distributions I–III are presented in Figures 1 (a)–(c). The distributions II and III are projected in two dimensions using the principal component analysis (PCA).



(a)



(b)



(c)

Figure 1: Plots of randomly sampled points from the distributions used in generation of data set I (a), II (b) and III (c).

3.2 Test procedure

Seven different kernel estimation methods were compared. Three of them were based on the SOM, three used the GMM and the last one was the S-Map algorithm. Descriptions of the methods are in Table 2.

All the SOM based kernel estimates were built by first computing the kernel centers (model vectors) using the SOM algorithm. In the simulations, the Gaussian neighborhood kernel was used. Then, for each kernel, the diagonal covariance matrix with different variances was estimated based on the data sets described in Table 2.

In the GMMs, the initial guesses for kernel centers were computed using K-means algorithm. In the first case, the variances of diagonal covariance matrix were determined using Voronoi sets of these centers. In the last two cases, the elements of diagonal covariance matrix were computed using the EM algorithm. In the former case the variances were constrained to be equal, in the latter they were allowed to be different.

The S-Map was trained using the S-Map algorithm which computes the kernel centers and kernel width (variance) parameter for all units during the network training.

Method #	**Method description**
1	SOM and all Voronoi sets weighted by the neighborhood function
2	SOM and the Voronoi sets of the units and their six closest units
3	SOM and the Voronoi sets of the units and their six 1-neighbors
4	GMM and EM, initialization only
5	GMM and EM, variances equal, 20 iterations
6	GMM and EM, variances different, 20 iterations
7	S-Map

Table 2: Kernel estimation methods used in the tests.

All the methods were tested for the data sets I–III. Two versions of each data set were used: 400 and 4000 randomly chosen data vectors. The number of kernels was 40 in all tests. After kernel estimation, each distribution was reconstructed using kernel estimates. The obtained distribution q was then compared to the original distribution p using Kullback-Leibler divergence:

$$d = \sum_i p_i \log(p_i/q_i) + q_i \log(q_i/p_i). \qquad (4)$$

3.3 Results

The results for 400 samples (10 samples/kernel) are presented in Figure 2 (a) and for 4000 samples (100 samples/kernel) in Figure 2 (b). The test error was computed using Eq. 4. All results are averages over 100 test runs.

Four examples of data set I distributions vs. reconstructed ones are illustrated in Figures 3 (a)–(d).

4 Discussion

The GMM with 20 iterations of EM algorithm gave the best results. The initialized version of the algorithm produced the weakest results, which is not very surprising.

The SOM-based kernel estimates did quite well except for the one using Voronoi sets of the BMU and its topological neighbors. This is probably due to the fact that neighboring units are in some parts of the SOM quite different. Taking into account that the method requires no iterations and it didn't move the kernel centers like GMM, the results were good.

Even though the results of the S-Map were not very good, it succeeded in the comparison quite well in proportion to the number of estimated parameters. It had only one common variance parameter, the kernel width, whereas all the other methods had several dozens of parameters. It should also be noted that the results could probably be improved by more careful selection of training parameters.

Our primary goal was to find out if the SOM unit activations could be given probabilistic interpretation using a RKDE even if the number of samples for each quantization unit is small. In the light of our results this appears to be roughly possible.

5 Acknowledgments

The authors wish to thank Mr. Kimmo Kiviluoto for program code and expertise in S-Map training.

References

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

[2] C. M. Bishop, M. Svensen, and C. K. I. Williams. GTM: A principled alternative to the self-organizing map. In *Advances in Neural Information Processing Systems*, volume 9, pages 354–360, 1997.

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

[4] A. Hämäläinen. *Self-organizing Map and Reduced Kernel Density Estimation.* PhD thesis, University of Jyväskylä, 1995.

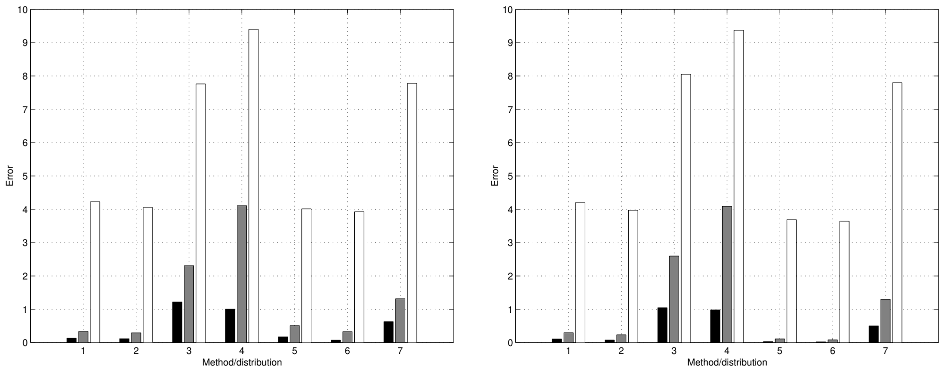
[5] K. Kiviluoto and E. Oja. S-Map: A network with a simple self-organization algorithm for generative topographic mappings. In M. I. Jordan, M. J. Kearns, and S. A. Solla, editors, *Advances in Neural Processing Systems 10*, pages 549–555. MIT Press, 1997.

[6] T. Kohonen. *Self-Organizing Maps.* Springer, Berlin, Heidelberg, New York, 1995.

[7] 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 – 1384, October 1996.

[8] I. Nabney and C. M. Bishop. Netlab. URL http://www.ncrg.aston.ac.uk/netlab/, 1998.

[9] V. Tryba and K. Goser. Self-Organizing Feature Maps for process control in chemistry. In T. Kohonen, K. Mäkisara, O. Simula, and J. Kangas, editors, *Artificial Neural Networks*, pages 847–852, Amsterdam, Netherlands, 1991. North-Holland.



(a) (b)

Figure 2: Test results for (a) 400, (b) 4000 samples data sets. The results of data set I are denoted by black, data set II by grey and data set III by white color.

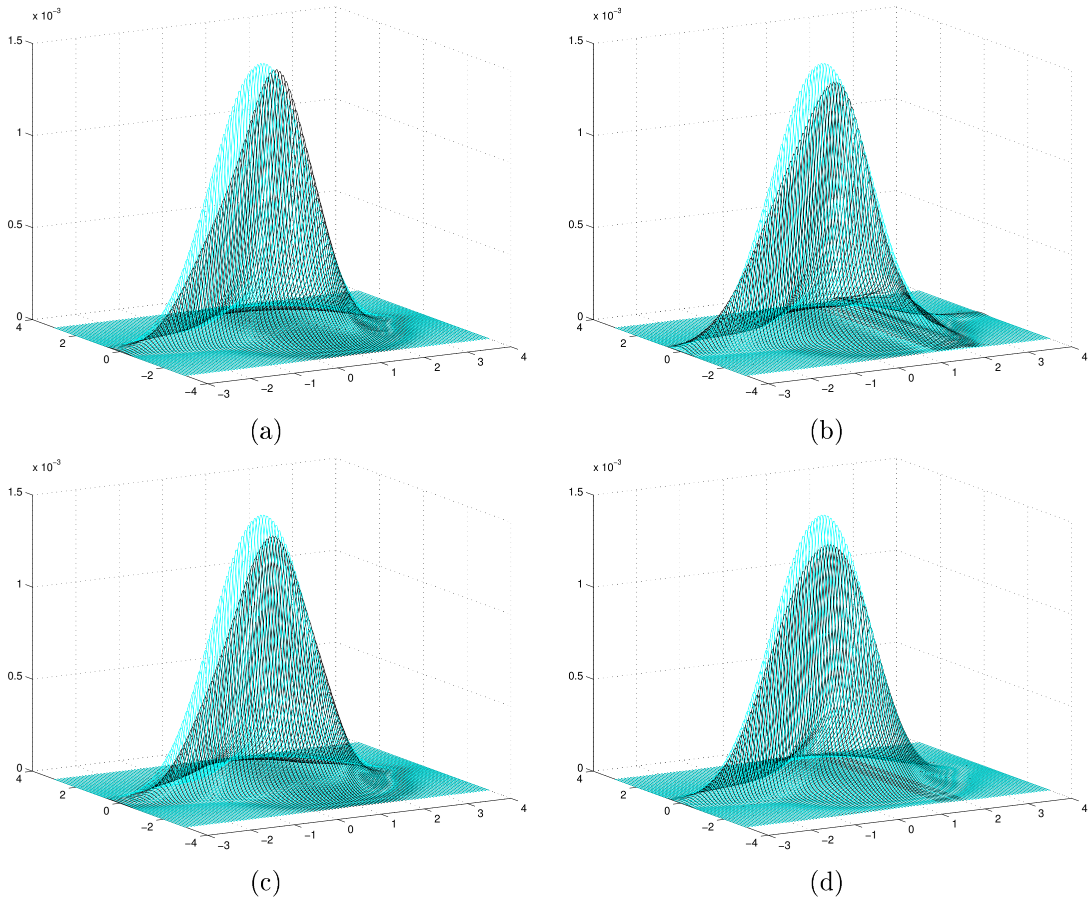


Figure 3: Reconstructed vs. original distributions: results with 400 samples using method 1 (a), 6 (b) and 4000 samples using method 1 (c), 6 (d). The original distribution is denoted by gray and the reconstructed one by black color.