Available online at www.sciencedirect.com
SCIENCE DIRECT
Neural Networks 19 (2006) 780–784
Neural Networks
www.elsevier.com/locate/neunet

2006 Special Issue

On the equivalence between kernel self-organising maps and self-organising mixture density networks

Hujun Yin*

School of Electrical and Electronic Engineering, University of Manchester, Manchester, M60 1QD, United Kingdom

Abstract

The kernel method has become a useful trick and has been widely applied to various learning models to extend their nonlinear approximation and classification capabilities. Such extensions have also recently occurred to the Self-Organising Map (SOM). In this paper, two recently proposed kernel SOMs are reviewed, together with their link to an energy function. The Self-Organising Mixture Network is an extension of the SOM for mixture density modelling. This paper shows that with an isotropic, density-type kernel function, the kernel SOM is equivalent to a homoscedastic Self-Organising Mixture Network, an entropy-based density estimator. This revelation on the one hand explains that kernelising SOM can improve classification performance by acquiring better probability models of the data; but on the other hand it also explains that the SOM already naturally approximates the kernel method.

Keywords: SOM; Kernel method; Kernel SOM; Mixture models; Bayes classifier

1. Introduction

The Self-Organising Map (SOM) (Kohonen, 1999) is one of the most popular and widely applied neural network models owing to its several distinct properties over other neural networks such as nonlinear mapping of input to output space and topological preserving on the map. The SOM has been studied, applied and extended extensively in the last couple of decades and many insights have been gained since its introduction. However numerous new developments and applications continue to emerge (e.g. Allinson, Obermayer, & Yin, 2002; Allinson, Yin, Allinson, & Slack, 2001; Ishikawa, Miikulainen, & Ritter, 2004).

Kernel methods have received a great deal of attention in the past few years, especially in the supervised learning community (Shawe-Taylor & Cristianini, 2004). By applying a nonlinear mapping function to the input space and with the help of kernel operations on the mapped space, a nonlinear, complex problem may become linear in the high dimensional feature space (Aizerman, Braverman, & Rozonoer, 1964). Typical examples are the Support Vector Machines (Cortes & Vapnik, 1995). Kernel methods have also been applied to unsupervised learning models such as principal component analysis (Schölkopf, Smola, & Müller, 1998), principal factor analysis, projection pursuit and canonical correlation analysis (Fyfe, MacDonald, & Charles, 2000). Two kernel variants of the SOM have been proposed recently. MacDonald and Fyfe (2000) derived a kernel SOM from kernelising the k-means clustering algorithm with added neighbourhood learning. Andras (2002) and Pan, Chen, and Zhang (2004) have also proposed a kernel SOM by transforming the input space to a feature space and then applying nonlinear kernel functions to the mapped data and obtained improved classification results.

The objectives of these kernel SOMs are different from some earlier approaches such as Graepel, Burger, and Obermayer (1998) and Yin and Allinson (2001) and van Hulle (2002), which aim at optimising the topographic mapping and approximating data distribution respectively. In Graepel et al. (1998), kernel functions were applied to transform the input to high dimensional space, thus transforming the distance metric to nonlinear and adding more flexibility in vector-quantising and capturing the data structures. In Yin and Allinson (2001) and van Hulle (2002), neurons in the SOM were treated as Gaussian (or other) kernels; the resulting map approximates a mixture of Gaussian (or other) distributions of the data

* Tel.: +44 (0)161 306 8714; fax: +44 (0)161 306 4784.
E-mail address: h.yin@manchester.ac.uk.

0893-6080/$ - see front matter © 2006 Elsevier Ltd. All rights reserved.
doi:10.1016/j.neunet.2006.05.007

by minimising the Kullback–Leibler divergence between the neural model and the data. It further establishes a link between the mixture model and the self-organisation process (Yin & Allinson, 2001). Hegde, Erdogmus, Lehn-Schioler, Rao, and Principe (2004) has also applied such an information theoretical approach to vector quantisation, though without the neighbourhood learning.

This paper provides a formal link between the kernel SOMs and the Self-Organising Mixture Networks (SOMN) (Yin & Allinson, 2001), as well as explaining that the SOM implicitly approximates the kernel methods. Establishing the connection between the kernel approach and a probabilistic model reveals why the kernel SOMs can perform better for classification than the standard SOM in certain situations. The paper is organised as follows. In Section 2 two recent kernel SOMs are reviewed and it is further shown that both can be derived from an energy function of the SOM (Heskes, 1999; Lampinen & Oja, 1992). Furthermore, we show that the kernel SOMs can be performed in the transformed space completely. This link unifies the kernel approaches to the SOM. In Section 3, a direct relationship between the kernel SOM and the SOMN is revealed. The relation further explains in Section 4 that the kernel SOM is also an underlying mixture model and that the SOM is an approximation of a kernel method. Conclusions are given in Section 5.

2. Kernel self-organising maps

A kernel is a function $\kappa: X \times X \in \Re$, where X is the input space. This function is a dot product of the mapping function $\phi(\mathbf{x})$, i.e. $\kappa(\mathbf{x}; \mathbf{y}) = \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle$, where $\phi: X \to F$, F is a high dimensional inner product feature space. The mapping function $\phi(\mathbf{x})$ is often nonlinear and not known. All the operations are defined in terms of the kernel function instead of the mapping function.

2.1. Type I kernel SOM

Following the kernel PCA (Schölkopf et al., 1998), a k-means based kernel SOM (termed here the *type I kernel SOM*) has been proposed by MacDonald and Fyfe (2000). Each data point $\mathbf{x}$ is mapped to the feature space via a (unknown or imaginary) nonlinear function $\phi(\mathbf{x})$. In principle each mean can be described as a weighted sum of the observations in the feature space, $\mathbf{m}_i = \sum_n \gamma_{i,n} \phi(\mathbf{x}_n)$, where $\{\gamma_{i,n}\}$ are the constructing coefficients. The algorithm then selects a mean or assigns a data point with the minimum distance between the mapped point and the mean,

$$\begin{aligned} \|\phi(\mathbf{x}) - \mathbf{m}_i\|^2 &= \left\| \phi(\mathbf{x}) - \sum_n \gamma_{i,n} \phi(\mathbf{x}_n) \right\|^2 \\ &= \kappa(\mathbf{x}, \mathbf{x}) - 2 \sum_n \gamma_{i,n} \kappa(\mathbf{x}, \mathbf{x}_n) + \sum_{n,m} \gamma_{m,n} \kappa(\mathbf{x}_n, \mathbf{x}_m). \end{aligned} \quad (1)$$

The update of the mean is based on a soft learning algorithm,

$$\mathbf{m}_i(t+1) = \mathbf{m}_i(t) + \Lambda[\phi(\mathbf{x}) - \mathbf{m}_i(t)] \quad (2)$$

where Λ is the normalised winning frequency of the i-th mean and is defined as

$$\Lambda = \frac{\zeta_{i(\mathbf{x}),j}}{\sum_{n=1}^{t+1} \zeta_{i,n}} \quad (3)$$

where ζ is the winning counter and is often defined as a Gaussian function between the indexes of the two neurons.

As the mapping function $\phi(\mathbf{x})$ is not known, the updating rule Eq. (2) is further elaborated and leads to the following updating rules for the constructing coefficients of the means (MacDonald & Fyfe, 2000):

$$\gamma_{i,n}(t+1) = \begin{cases} \gamma_{i,n}(t)(1-\zeta), & \text{for } n \neq t+1 \\ \zeta, & \text{for } n = t+1. \end{cases} \quad (4)$$

Note that these constructing coefficients, $\{\gamma_{i,n}\}$, together with the kernel function, effectively define the kernel SOM in the feature space. The winner selection, i.e. Eq. (1), operates on these coefficients and the kernel function. No explicit mapping function $\phi(\mathbf{x})$ is required. The exact means or neuron weights, $\{\mathbf{m}_i\}$, are not required.

2.2. Type II kernel SOM

There is another, direct way to kernelise the SOM (termed here the *type II kernel SOM*) by mapping the data points and neuron weights, both defined in the input space, to a feature space, then applying the SOM in the mapped dot product space. The winning rules of this second type of kernel SOM have been proposed as follows either in the input space (Pan et al., 2004):

$$v = \arg\min_i \|\mathbf{x} - \mathbf{m}_i\|, \quad (5)$$

or in the feature space (Andras, 2002):

$$v = \arg\min_i \|\phi(\mathbf{x}) - \phi(\mathbf{m}_i)\|. \quad (6)$$

It will soon become clear that these two rules are equivalent for certain kernels such as Gaussian ones.

The weight updating rule is proposed as (Andras, 2002)

$$\mathbf{m}_i(t+1) = \mathbf{m}_i(t) + \alpha(t) h(v(\mathbf{x}), i) \nabla J(\mathbf{x}, \mathbf{m}_j) \quad (7)$$

where $J(\mathbf{x}, \mathbf{m}_i) = \|\phi(\mathbf{x}) - \phi(\mathbf{m}_i)\|^2$ is the distance function in the feature space or the proposed instantaneous or sample objective function. $\alpha(t)$ and $h(v(\mathbf{x}), i)$ are the learning rate and neighbourhood function respectively.

Note that

$$\begin{aligned} J(\mathbf{x}, \mathbf{m}_i) &= \|\phi(\mathbf{x}) - \phi(\mathbf{m}_i)\|^2 \\ &= \kappa(\mathbf{x}, \mathbf{x}) + \kappa(\mathbf{m}_i, \mathbf{m}_i) - 2\kappa(\mathbf{x}, \mathbf{m}_i) \end{aligned} \quad (8)$$

and

$$\nabla J(\mathbf{x}, \mathbf{m}_i) = \frac{\partial \kappa(\mathbf{m}_i, \mathbf{m}_i)}{\partial \mathbf{m}_i} - 2 \frac{\partial \kappa(\mathbf{x}, \mathbf{m}_i)}{\partial \mathbf{m}_i}. \quad (9)$$

Therefore this kernel SOM can also be operated entirely in the feature space with the kernel function. As the weights of the neurons are defined in the input space, they can be explicitly resolved, unlike in the type I kernel SOM.

2.3. Link to an energy function

From the energy function point of view, the SOM minimises the following energy (Lampinen & Oja, 1992; Heskes, 1999), at least for the discrete case,[1]

$$E = \sum_i \int_{V_i} \sum_j h(i,j)\|\mathbf{x}-\mathbf{m}_j\|^2 p(\mathbf{x})\mathrm{d}\mathbf{x} \tag{10}$$

where V_i is the Voronoi region of neuron i.

The extension of this energy function in the feature space is

$$E_F = \sum_i \int_{V_i} \sum_j h(i,j)\|\phi(\mathbf{x})-\phi(\mathbf{m}_j)\|^2 p(\mathbf{x})\,\mathrm{d}\mathbf{x}. \tag{11}$$

The kernel SOM can be seen as a result of directly minimising this transformed energy stochastically, i.e., using the sample gradient on $\sum_j h(v(\mathbf{x}),j)\|\phi(\mathbf{x})-\phi(\mathbf{m}_j)\|^2$:

$$\begin{aligned}\frac{\partial \hat{E}_F}{\partial \mathbf{m}_i} &= \frac{\partial}{\partial \mathbf{m}_j}\sum_j h(v(\mathbf{x}),j)\|\phi(\mathbf{x})-\phi(\mathbf{m}_j)\|^2 \\ &= -2h(v(\mathbf{x}),i)\nabla J(\mathbf{x},\mathbf{m}_j).\end{aligned} \tag{12}$$

This leads to the same weight updating rule of the type II kernel SOM as to Eq. (7).

Various kernel functions such as Gaussian (or radial basis function), Cauchy and polynomial ones are readily applicable to the kernel SOM (Lau, Yin, & Hubbard, in press). For example, for a Gaussian kernel, the winning and weight updating rules are

$$\begin{aligned}v &= \arg\min_i J(\mathbf{x},\mathbf{m}_i) = \arg\min_i[-2\kappa(\mathbf{x},\mathbf{m}_i)] \\ &= \arg\min_i\left[-\exp\left(-\frac{\|\mathbf{x}-\mathbf{m}_i\|^2}{2\sigma^2}\right)\right]\end{aligned} \tag{13}$$

$$\begin{aligned}\mathbf{m}_i(t+1) &= \mathbf{m}_i(t)+\alpha(t)h(v(\mathbf{x}),i)\frac{1}{2\sigma^2} \\ &\quad\times\exp\left(-\frac{\|\mathbf{x}-\mathbf{m}_i\|^2}{2\sigma^2}\right)(\mathbf{x}-\mathbf{m}_i)\end{aligned} \tag{14}$$

respectively. Please note that for Gaussian kernel functions, although the winning rule Eq. (13) is derived from the feature space, it is equivalent to Eq. (5), the winning rule conducted in the input space.

For the type I kernel SOM, as the neuron weights are defined in the feature space, one can define a corresponding distance between a neuron weight and a mapped input in the feature space as, although the explicit mapping function is unknown,

$$\begin{aligned}J'(\mathbf{x},\mathbf{m}_i) &= \|\phi(\mathbf{x})-\mathbf{m}_i\|^2 \\ &= \kappa(\mathbf{x},\mathbf{x})+\mathbf{m}_i\cdot\mathbf{m}_i-2\phi(\mathbf{x})\cdot\mathbf{m}_i.\end{aligned} \tag{15}$$

Then,

$$\nabla J'(\mathbf{x},\mathbf{m}_i) = 2\mathbf{m}_i - 2\phi(\mathbf{x}). \tag{16}$$

[1] The use of this energy function may imply a difference winning rule, $v = \arg\min_i \sum_j h(i,j)\|\mathbf{x}-\mathbf{m}_i\|^2$. For simplicity we still use the simple winning rule, Eq. (5) or (6). When the number of neurons is large and the neighbourhood function is symmetric, the two become similar.

Replacing the distance measure and its derivative in the above energy function (11) and its differentiation (12) and using the steepest descent method we obtain the updating rule for the neuron weights as

$$\mathbf{m}_i(t+1) = \mathbf{m}_i(t) + 4\alpha(t)h(v(\mathbf{x}),i)(\phi(\mathbf{x})-\mathbf{m}_i). \tag{17}$$

As can be seen, it is equivalent to Eq. (2), the underlying weight updating rule of the type I kernel SOM. Therefore the type I kernel SOM also minimises this energy function in the feature space.

3. Self-organising mixture network and kernel SOMs

The self-organising mixture (density) network (SOMN) (Yin & Allinson, 2001) extends and adapts the SOM to a mixture density model, in which each node characterises a conditional probability distribution. The joint probability density of the data (or the network) is described by a mixture distribution,

$$p(\mathbf{x}\mid\Theta) = \sum_{i=1}^{K} p_i(\mathbf{x}\mid\theta_i)P_i \tag{18}$$

where $p_i(\mathbf{x}\mid\theta_i)$ is the i-th component-conditional density, and θ_i is the parameter for the i-th conditional density, $i = 1,2,\ldots K$, $\Theta = (\theta_1,\theta_2,\ldots,\theta_K)^T$, and P_i is the prior probability of the i-th component or node and is also called the mixing weights. For example, a Gaussian mixture has the following conditional densities respectively:

$$\begin{aligned}p_i(\mathbf{x}\mid\theta_i) &= \frac{1}{(2\pi)^{d/2}|\Sigma_i|^{1/2}} \\ &\quad\times\exp\left[-\frac{1}{2}(\mathbf{x}-\mathbf{m}_i)^T\Sigma_i^{-1}(\mathbf{x}-\mathbf{m}_i)\right]\end{aligned} \tag{19}$$

where $\theta_i = \{\mathbf{m}_i, \Sigma_i\}$ are the mean vector and covariance matrix respectively.

Suppose that the true environmental data density function and the estimated one are $p(\mathbf{x})$ and $\hat{p}(\mathbf{x})$ respectively. The Kullback–Leibler information distance measures the divergence between these two, and is defined as

$$\mathbf{I} = -\int \log\frac{\hat{p}(\mathbf{x})}{p(\mathbf{x})}p(\mathbf{x})\,\mathrm{d}\mathbf{x}. \tag{20}$$

It is always positive and is equal to zero only when the two densities are identical.

When the estimated density is modelled as a mixture distribution, i.e. a function of various sub-densities and their parameters, one can seek the optimal estimate of these parameters by minimising the Kullback–Leibler divergence via its partial differentials with respect to every model parameter, i.e.

$$\frac{\partial \mathbf{I}}{\partial\theta_i} = -\int\left[\frac{1}{\hat{p}(\mathbf{x}\mid\hat{\Theta})}\frac{\partial\hat{p}(\mathbf{x}\mid\hat{\Theta})}{\partial\theta_i}\right]p(\mathbf{x})\,\mathrm{d}\mathbf{x}, i = 1,2,\ldots,K. \tag{21}$$

As the true data density is not known, the stochastic gradient is used for solving these non-directly solvable equations.

This results in the following adaptive updating rules for the parameters and priors Yin and Allinson (2001)[2]:

$$\begin{aligned}\hat{\theta}_i(t+1) &= \hat{\theta}_i(t) + \alpha(t)h(v(\mathbf{x}), i)\left[\frac{1}{\hat{p}(\mathbf{x} \mid \hat{\Theta})}\frac{\partial \hat{p}(\mathbf{x} \mid \hat{\Theta})}{\partial \theta_i}\right] \\ &= \hat{\theta}_i(t) + \alpha(t)h(v(\mathbf{x}), i) \\ &\quad \times \left[\frac{\hat{P}_i(t)}{\sum_j \hat{P}_i(t)\hat{p}_j(\mathbf{x} \mid \theta_j)}\frac{\partial \hat{p}_i(\mathbf{x} \mid \hat{\theta}_i)}{\partial \theta_i}\right] \qquad (22)\end{aligned}$$

$$\begin{aligned}\hat{P}_i(t+1) &= \hat{P}_i(t) + \alpha(t)\left[\frac{\hat{p}_i(\mathbf{x} \mid \hat{\theta}_i)\hat{P}_i(t)}{\hat{p}(\mathbf{x} \mid \Theta)} - \hat{P}_i(t)\right] \\ &= \hat{P}_i(t) - \alpha(t)h(v(\mathbf{x}), i)[\hat{P}(i \mid \mathbf{x}) - \hat{P}_i(t)] \qquad (23)\end{aligned}$$

where $\alpha(t)$ is the learning coefficient or rate at time step t, and $0 < \alpha(t) < 1$ and decreases monotonically. The neighbourhood function $h(v(\mathbf{x}), i)$ is further introduced to restrict the learning in a neighbourhood of the winner, which is found via the maximum (estimated) posterior probability of the node,

$$\hat{P}(i \mid \mathbf{x}) = \frac{\hat{P}_i\hat{p}_i(\mathbf{x} \mid \hat{\theta}_i)}{\hat{p}(\mathbf{x} \mid \hat{\Theta})}. \qquad (24)$$

When the SOMN is limited to the homoscedastic case, i.e. equal variances and equal priors (non-informative priors) for all components, only the means are the learning variables. The above winner rule becomes

$$v = \arg\max_i \frac{\hat{p}_i(\mathbf{x} \mid \theta_i)}{\sum_j \hat{p}_j(\mathbf{x} \mid \theta_j)}. \qquad (25)$$

Eq. (25) is then equivalent to rule Eq. (5) or (6) or (13) when the conditional density function is isotropic or symmetric or is a function of $\|\mathbf{x} - \mathbf{m}\|$.

The corresponding weight updating rule is

$$\begin{aligned}\mathbf{m}_i(t+1) &= \mathbf{m}_i(t) + \alpha(t)h(v(\mathbf{x}), i) \\ &\quad \times \frac{1}{\sum_j p_j(\mathbf{x} \mid \theta_j)}\frac{\partial p_i(\mathbf{x} \mid \theta_i)}{\partial \mathbf{m}_i}. \qquad (26)\end{aligned}$$

It can be seen that Eq. (26) bears a similarity to Eq. (7). Again if the conditional density function is of a kernel type and isotropic (e.g. Gaussian functions), the above rule leads to the same result as Eq. (7), with an additional normalising factor, $\sum_j \hat{p}_j(\mathbf{x} \mid \theta_j) = p(\mathbf{x})$. When the data density is relatively smooth, this factor is only a (common) scalar value for all neurons.

For example, for a Gaussian mixture with equal variance and prior for all nodes, it is easy to show that the winning and mean updating rules become

$$v = \arg\max_i \left[\exp\left(-\frac{\|\mathbf{x} - \mathbf{m}_i\|^2}{2\sigma^2}\right)\right] \qquad (27)$$

$$\begin{aligned}\mathbf{m}_i(t+1) &= \mathbf{m}_i(t) + \alpha(t)h(v(\mathbf{x}), i)\frac{1}{2\sigma^2} \\ &\quad \times \frac{1}{\sum_j p_j(\mathbf{x} \mid \theta_j)}\exp\left(-\frac{\|\mathbf{x} - \mathbf{m}_i\|^2}{2\sigma^2}\right)(\mathbf{x} - \mathbf{m}_i). \qquad (28)\end{aligned}$$

They are equivalent to those of the kernel SOM with Gaussian kernels, i.e., Eqs. (13) and (14).

The revealing of the equivalence between the kernel SOM and the SOMN explains why kernel SOM can provide better classification results than the standard SOM as reported in (MacDonald & Fyfe, 2000; Andras, 2002; Pan et al., 2004). It shows that the kernel SOM has better abilities to model the data density and thus yields improved classifications.

4. The SOM and kernel methods

The kernel SOMs on one hand explain that the kernel SOM is approximating a mixture density model using the kernel function as the prototype conditional density. On the other hand, as the SOM is a special case of the SOMN when equal variance and equal priors are used for all the nodes and when the number of nodes is large, the SOM is approximately a kernel method.

Experimental results on several benchmark data sets have shown that although the kernel SOM can produce better classification performance when the kernel parameters are optimised (often empirically) in some cases, we are short of evidence to indicate that the kernel SOM will always outperform the standard SOM (Lau et al., in press). Typical results are given in Tables 1 and 2. The performance of the kernel SOMs depends on the type of kernel functions, the parameters of the kernel function, as well as the labelling (classifying) methods for the neurons. Generally the kernel SOMs are much more computationally demanding than the standard SOM. The standard SOM can provide equivalent performance at a much lower cost.

As often the kernel SOM uses a single kernel function, i.e. the same parameter, e.g. width, throughout, this implies that the kernel SOM assumes the same variance for all its conditional densities in its SOMN counterpart. In other words, the kernel SOM uses the simplest case of the SOMN. One can

Table 1
Classification errors of kernel SOMs with $\sigma = 1.0$ on the iris data set

Kernel	Type I kernel SOM			Type II kernel SOM			Standard SOM		
	M	*A*	*V*	*M*	*A*	*V*	*M*	*A*	*V*
Gaussian	3.8	5.3	3.7	4.1	6.9	6.9			
Cauchy	3.8	4.3	3.4	3.6	6.1	6.1	4.1	6.9	3.9
Log[a]	11.8	38.7	40.2	3.9	6.8	6.8			

M, *A* and *V* denote the minimum distance, average distance and majority voting methods for labelling the nodes (Lau et al., in press).

[a] Note that the log function does not satisfy Mercer's theorem and thus is a non-Mercer kernel.

[2] A common neighbourhood function, $h(v(\mathbf{x}), i)$, is added for forming topology as in the SOM.

Table 2
Classification errors on the UCI colon cancer data set

Kernel	Type I kernel SOM			Type II kernel SOM			Standard SOM		
	M	*A*	*V*	*M*	*A*	*V*	*M*	*A*	*V*
Gaussian	5.6	5.8	5.6	5.3	5.3	4.7			
Cauchy	5.5	5.6	5.5	5.5	5.5	4.8	4.3	7.0	3.8
Log[a]	4.6	4.6	4.6	5.2	5.2	4.6			

M, *A* and *V* denote the minimum distance, average distance and majority voting methods for labelling the nodes (Lau, Yin, & Hubbard, in press).

[a] Note that the log function does not satisfy Mercer's theorem and thus is a non-Mercer kernel.

argue that like in a general SOMN, different parameters can be employed for different neurons for an optimised performance. However unless there exist methods for optimising the kernel parameters (or conditional densities), they can only be obtained empirically or through an exhaustive search, which often takes a long time and adds more computation on top of an already computationally intensive method. As the standard SOM can be regarded as a special case of the SOMN with the same width for all neurons when the number of neurons is large, the SOM actually approximates the kernel SOM already. In practice, when the data density is smooth and the number of neurons is large, the SOM will yield a similar performance to the kernel SOM for classification, but at a fraction of the computational cost.

5. Conclusions

In this paper, the relationship between the kernel SOM and the self-organising mixture network (SOMN) has been established. When the conditional density function is of a kernel type or the kernel function is of density type, and both are isotropic or symmetric, then the two methods are equivalent. Then the kernel SOM can be understood as an entropy optimised mixture density learner and thus can provide improved classification results. As the SOM is a special case of the SOMN, this in turn explains why the SOM approximates the kernel method naturally. In other words, further kernelising SOM may not be necessary.

References

Aizerman, M., Braverman, E., & Rozonoer, L. (1964). Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, *25*, 821–837.

Allinson, N., Obermayer, K., & Yin, H. (Eds.) (2002). New developments on self-organising maps [Special issue]. *Neural Networks*, *15*(8–9), 943–1151).

Allinson, N., Yin, H., Allinson, L., & Slack, J. (Eds.) (2001). *Advances in self-organising maps*. London: Springer.

Andras, P. (2002). Kernel-Kohonen networks. *International Journal of Neural Systems*, *12*, 117–135.

Cortes, C., & Vapnik, V. (1995). Support vector networks. *Machine Learning*, *20*, 273–297.

Fyfe, C., MacDonald, D., & Charles, D. (2000). Unsupervised learning using radial kernels. In R. J. Howlett (Ed.), *Recent advances in radial basis networks*.

Graepel, R. J. T., Burger, M., & Obermayer, K. (1998). Self-organizing maps: Generalization and new optimization techniques. *Neurocomputing*, *21*, 173–190.

Hegde, A., Erdogmus, D., Lehn-Schioler, T., Rao, Y. N., & Principe, J. C. (2004) Vector- quantization by density matching in the minimum Kullback–Leibler divergence sense. In *Proc IJCNN'04* (pp. 1362–1366).

Heskes, T. (1999). Energy functions for self-organizing maps. In E. Oja, & S. Kaski (Eds.), *Kohonen Maps*. Elsevier.

Ishikawa, M., Miikulainen, R., & Ritter, H. (Eds.) (2004). New developments on self-organising systems [Special issue]. *Neural Networks*, *17*, 1037–1389.

Kohonen, T. (1999). *Self-organising maps*. Springer.

Lampinen, J., & Oja, E. (1992). Clustering properties of hierarchical self-organizing maps. *Journal of Mathematical Imaging and Vision*, *2*, 261–272.

Lau, K. W., Yin, H., & Hubbard, S. (2006) Kernel self-organising maps for classification, *Neurocomputing* (in press).

MacDonald, D., & Fyfe, C. (2000). The kernel self organising map. In *Proc. 4th int. conf. on knowledge-based intelligence engineering systems and applied technologies* (pp. 317–320).

Pan, Z. S., Chen, S. C., & Zhang, D. Q. (2004). A kernel-base SOM classifier in input space. *Acta Electronica Sinica*, *32*, 227–231 (in Chinese).

Schölkopf, B., Smola, A., & Müller, K. R. (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, *10*, 1299–1319.

Shawe-Taylor, J., & Cristianini, N. (2004). *Kernel methods for pattern analysis*. Cambridge University Press.

van Hulle, M. (2002). Kernel-based topographic map formation achieved with an information- theoretic approach. *Neural Networks*, *15*, 1029–1039.

Yin, H., & Allinson, N. (2001). Self-organising mixture networks for probability density estimation. *IEEE Transactions on Neural Networks*, *12*, 405–411.