Kernel-based topographic map formation achieved with an information-theoretic approach

Marc M. Van Hulle
K.U.Leuven, Laboratorium voor Neuro- en Psychofysiologie
Campus Gasthuisberg, Herestraat, B-3000 Leuven, BELGIUM
Tel.: + 32 16 34 59 61, Fax: + 32 16 34 59 93
E-mail: marc@neuro.kuleuven.ac.be

May 13, 2002

2002 Special Issue of Neural Networks:

New Developments in Self-organizing Maps

Acknowledgments

M.M.V.H. is supported by research grants received from the Fund for Scientific Research (G.0185.96N), the National Lottery (Belgium) (9.0185.96), the Flemish Regional Ministry of Education (Belgium) (GOA 95/99-06; 2000/11), the Flemish Ministry for Science and Technology (VIS/98/012), and the European Commission, 5th framework programme (QLG3-CT-2000-30161 and IST-2001-32114).

Request for reprints

Marc M. VAN HULLE
K.U.Leuven
Laboratorium voor Neuro- en Psychofysiologie
Faculteit Geneeskunde
Campus Gasthuisberg
Herestraat
B-3000 Leuven
Belgium

Phone: + 32 16 34 59 61
Fax: + 32 16 34 59 60
E-mail: marc@neuro.kuleuven.ac.be

Running title

Kernel-based topographic map formation

Kernel-based topographic map formation achieved with an information-theoretic approach

Abstract

A new information-theoretic learning algorithm is introduced for kernel-based topographic map formation. The kernels are allowed to overlap and move freely in the input space, and to have differing kernel ranges. We start with Linsker's infomax principle and observe that it cannot be readily extended to our case, exactly due to the presence of kernels. We then consider Bell and Sejnowski's generalization of Linsker's infomax principle, which suggests differential entropy maximization, and add a second component to be optimized, namely, mutual information minimization between the kernel outputs, in order to take into account the kernel overlap, and thus the topographic map's output redundancy. The result is joint entropy maximization of the kernel outputs, which we adopt as our learning criterion. We derive a learning algorithm and verify its performance both for a synthetic example, for which the optimal result can be derived analytically, and for a classic real-world example.

Key words : self-organizing maps, kernel-based topographic maps, information-based learning, joint entropy maximization, incomplete gamma distribution kernel

1 Introduction

One way to improve the density estimation properties, the noise tolerance, or even the biological relevance of the Self-Organizing Map (SOM) algorithm (Kohonen, 1982,1995), is to equip the lattice neurons with local kernel functions, such as Gaussians, rather than Winner-Take-All (WTA) functions. We further call such lattices *kernel-based* topographic maps (Fig. 1). An early example is the elastic net of Durbin and Willshaw (1987), which can be viewed as an equal-variance Gaussian mixture density model, fitted to the data points by a penalized maximum likelihood term. In recent years, algorithms that adopt a probabilistic approach have been introduced by Bishop and co-workers (Bishop *et al.*, 1998) (Generative Topographic Map, based on constrained, equal-variance Gaussian mixture density modeling with equal mixings), Utsugi (1997) (also using equal mixings of equal-variance Gaussians), and Van Hulle (1998,2000) (equiprobabilistic maps using equal mixtures of Gaussians with differing variances). Furthermore, we should also mention the fuzzy membership in clusters approach of Graepel and co-workers (Graepel *et al.*, 1997), and the maximization of local correlations approach of Xu and co-workers (Sum *et al.*, 1997), both of which rely on equal-variance Gaussians. Graepel and co-workers (Graepel *et al.*, 1998) also proposed a still different approach to kernel-based topographic map formation by introducing a non-linear transformation that maps the data points to a high-dimensional "feature" space, and that, in addition, admits a kernel function, such as a Gaussian, with fixed vari-

ance, as in the (kernel-based) Support Vector Machines (SVMs) (Vapnik, 1995). This idea was recently taken up again by András (2001), but with the purpose of optimizing the map's classification performance, by individually adjusting the (Gaussian) kernel radii using a supervised learning algorithm. Another approach is offered by the Local Density Estimation (LDE) algorithm (Van Hulle, 2002), which individually adapts the centers and radii of the Gaussian kernels to the assumed Gaussian local input density. Finally, the original SOM algorithm itself has been regarded as an approximate way to perform equivariance Gaussian mixture density modeling by Utsugi (1997), Yin and Allinson (2001), Kostiainen and Lampinen (2002), and others.

Figure 1: about here.

The question is now: how to develop kernel-based topographic maps that optimize an *information-theoretic criterion*, such as Linsker's *infomax* principle?[1] The kernels should be allowed to move freely in the input space, possibly with differing kernel ranges, so as to model the local input density. The obvious answer is to express the average mutual information integral in terms of the kernel output densities, or probabilities when they are discretized, and adjust the kernel parameters so that the integral is maximized. However, such an approach rapidly becomes infeasible in practice: 1) topographic maps typically contain several tens to hundreds of neurons, and 2) several neurons in the map can be jointly active, due to their overlapping activation kernels. Indeed, with respect to the second argument, Linsker (1989) already needed to restrict himself to binary neurons (in a WTA network), in order to facilitate the computation of the average mutual information integral. A different information-theoretic approach is to minimize the Kullback-Leibler divergence (also called relative- or cross-entropy) between the true and the estimated input density, an idea that has been introduced for kernel-based topographic map formation by Benaim and Tomasini (1991), using equal-variance Gaussians, and extended more recently by Yin and Allinson (2001) to Gaussians with differing variances.

In this article, we will introduce a new learning algorithm for kernel-based topographic map formation that is aimed at maximizing the map's joint entropy. This is achieved by maximizing the (differential) entropies of the kernel outputs individually, which also maximizes the information transfer through each kernel, given that also the mutual information between the kernel outputs needs to be minimized. The latter is achieved heuristically by having a competitive stage in the learning process, the presence of which is, together with the neighborhood function, essential for topology-preserving map formation. The article is organized as follows. In section 2, we develop our information-theoretic approach to kernel-based

[1] Linsker proposed a principle of maximum information preservation (mutual information maximization between input and output – *infomax* for short) which was later adopted by the Independent Component Analysis (ICA) community for addressing the Blind Source Separation (BSS) problem in the zero-noise limit (Bell and Sejnowski, 1995; Amari *et al.*, 1996; Cardoso and Laheld, 1996).

topographic map formation. Then, in section 3, we introduce our self-organizing learning algorithm. In section 4, we discuss the correspondence with other algorithms for kernel-based topographic map formation. In section 5, we assess the density estimation and the joint entropy performances for two example cases: a synthetic example, for which the theoretically-optimal joint entropy result can be derived analytically, and a real-world example, for which the theoretically-optimal result is not known, but which is commonly considered for benchmarking in the density estimation literature. Finally, we conclude the article in section 6.

2 Information-theoretic approach

As pointed out in the introduction, maximizing the average mutual information integral is not a feasible option for kernel-based topographic map formation. Another possibility is the generalization, introduced by Bell and Sejnowski (1995), of Linsker's infomax principle to nonlinear (*i.e.*, sigmoidal) output functions (in the zero-noise limit, *i.e.*, uncorrupted by any known noise source). They showed that the mutual information between input and output is maximized when the output entropy is maximized. The entropy is called "differential" entropy since it only holds with respect to some reference such as the discretization accuracy of the inputs and the outputs. The mutual information between input and output is maximized by maximizing the differential entropy only, which will be achieved when the output activity distribution is uniform (we assume that the output has bounded support). As a result of maximizing the differential entropy, when assuming an unimodal input density, for the argument's sake, the high density part of the input matches the sloping part of the neuron's output function $h(v)$. In other words, the nonlinear transformation $h(v)$, $v \in V \subseteq \Re$, that makes the output distribution uniform is the cumulative distribution function (repartition function):

$$h(v) = \int_{-\infty}^{v} p(x)\, dx, \tag{1}$$

with $p(.)$ the input density. Note that the derivative of $h(v)$ corresponds to the input density distribution.

It is clear that this principle cannot directly be applied to kernel-based output functions. Indeed, $h(v)$ in eq. (1) is a *global* output function: it increases monotonously when in the input v goes from $-\infty$ to ∞. What we need is a *localized* output function, a kernel function, such as a Gaussian. In addition, since we have multiple neurons in a topographic map, the kernels are likely to overlap, even if they would have bounded supports in the input space. The overlap makes the neural outputs statistically dependent and, hence, the lattice output distribution redundant. We can formulate this dependency in information-theoretic terms as the mutual information between the neural outputs. Consider a lattice consisting of two neurons with outputs y_1 and y_2. Their joint entropy can be written as:

$$H(y_1, y_2) = H(y_1) + H(y_2) - I(y_1, y_2), \tag{2}$$

with $H(y_1)$ and $H(y_2)$ the differential entropies of the neuron outputs, and $I(y_1, y_2)$

the mutual information between them. Maximizing the joint entropy $H(y_1, y_2)$ consists of maximizing the differential entropies while jointly minimizing $I(y_1, y_2)$. Indeed, when the differential entropies would be maximized individually, then the converged kernels will coincide completely, and the mutual information will be maximal. Conversely, the mutual information will be minimal when the neurons are inactive for every sample drawn from the input distribution, but then the differential entropies will also be minimal. Hence, the learning algorithm needs to find a trade-off between the two contributing factors in eq. (2).

2.1 Kernel definition

As motivated by Bell and Sejnowski (1995), the mutual information that the output y_i of neuron i contains about its input $\mathbf{v}$, $I(y_i, \mathbf{v})$, $\mathbf{v} \in V \subseteq \Re^d$, is maximal when the differential entropy of y_i, $H(y_i)$, is maximal. When assuming that the kernel output has bounded support, the differential entropy $H(y_i)$ will be maximal when the output distribution is uniform. This is achieved when the kernel function corresponds to the cumulative distribution function (repartition function) of the kernel's input density.

Assume that the input density distribution is a d-dimensional, radially-symmetrical Gaussian with mean $[\mu_1, \ldots, \mu_d]$, and standard deviation σ. The distribution of the Euclidean distances to the mean of the Gaussian can be obtained as follows. For a unit-variance Gaussian, $x \stackrel{\Delta}{=} \sum_{j=1}^{d} (v_j - \mu_j)^2$, $\mathbf{v} = [v_j]$, is known to obey the chi-squared distribution with $\theta = 2$ and $\alpha = \frac{d}{2}$ degrees of freedom (Weisstein, 1999):

$$p_{\chi^2}(x) = \frac{x^{\frac{d}{2}-1} \exp -\frac{x}{2}}{2^{\frac{d}{2}} \Gamma(\frac{d}{2})}, \tag{3}$$

for $0 \leq x < \infty$, and with $\Gamma(.)$ the gamma distribution. Let $r = \sqrt{x}$, then $p(r) = 2\, r\, p_{\chi^2}(r^2)$, following the fundamental law of probabilities. After some algebraic manipulations, we can write the distribution of the Euclidean distances as follows:

$$p(r) = \frac{2r^{d-1} \exp -\frac{r^2}{2}}{2^{\frac{d}{2}} \Gamma(\frac{d}{2})}, \tag{4}$$

or when the input Gaussian's standard deviation is σ:

$$p(r) = \frac{2(\frac{r}{\sigma})^{d-1} \exp\left(-\frac{(\frac{r}{\sigma})^2}{2}\right)}{2^{\frac{d}{2}} \Gamma(\frac{d}{2})}. \tag{5}$$

The distribution is exemplified in Fig. 2 (thick and thin continuous lines). The mean of r equals $\mu_r = \frac{\sqrt{2}\sigma\Gamma(\frac{d+1}{2})}{\Gamma(\frac{d}{2})}$ which can be approximated as $\sqrt{d}\sigma$ for d large, using the approximation for the ratio of the gamma functions by Graham and co-workers (1994); the second moment around zero equals $d\sigma^2$.

Finally, the kernel is defined in accordance with the cumulative distribution of $p(r)$, which is the (complement of the) incomplete gamma distribution:

$$y_i = K(r, \mathbf{w}_i, \sigma_i) = P(\frac{d}{2}, \frac{r^2}{2\sigma_i^2}) \equiv \frac{\Gamma(\frac{d}{2}, \frac{R^2}{2\sigma_i^2})}{\Gamma(\frac{d}{2})}. \tag{6}$$

The kernel functions for $d = 1, \ldots, 10$ are plotted in Fig. 2 (thick and thin dashed lines). When evaluating y_i in terms of the input $\mathbf{v}$, a radially-symmetrical kernel is obtained, which gradually becomes bell-shaped, when d increases (*cf.*, rightmost thin dashed line in Fig. 2). It should be noted that this kernel can be used for regression purposes, *e.g.*, as in a Radial Basis Function (RBF) type of network, since it is capable of "universal approximation", in the sense of Park and Sandberg (1993) (see Appendix). We will not further consider regression in this article, but rather concentrate on density estimation.

Figure 2: about here.

---

3 Self-organizing learning algorithm

Consider a lattice A of N neurons and corresponding Gaussian kernels $K(\mathbf{v}, \mathbf{w}_i, \sigma_i)$, $i = 1, \ldots, N$. Consider first the adaptation of the kernel center. The entropy of the kernel output of neuron i can be written as:

$$H(y_i) = -\int_0^\infty p_{y_i}(x) \ln p_{y_i}(x)\ dx, \tag{7}$$

with $p_{y_i}(.)$ the kernel output density which, in turn, can be written as a function of the distribution of the kernel input r:

$$p_{y_i}(y_i) = \frac{p_r(r)}{|\frac{\partial y_i}{\partial r}|}. \tag{8}$$

After substitution of the latter into eq. (7), we obtain:

$$H(y_i) = -\int_0^\infty p_r(r)\ \ln p_r(r)\ dr + \int_0^\infty p_r(r)\ \ln |\frac{\partial y_i(r)}{\partial r}|\ dr. \tag{9}$$

Since the first term on the right hand side does not depend on the kernel center, we only need to further concentrate on the second term, which in fact corresponds to the expected value of its ln component. Our on-line stochastic gradient ascent learning rule for the kernel center is then derived as follows:

$$\Delta \mathbf{w}_i = \eta_w \frac{\partial H}{\partial r} \frac{\partial r}{\partial \mathbf{w}_i} = \eta_w \left(\frac{\partial y_i}{\partial r} \right)^{-1} \frac{\partial}{\partial \mathbf{w}_i} \left(\frac{\partial y_i}{\partial r} \right), \tag{10}$$

with η_w the learning rate, and with:

$$\frac{\partial y_i}{\partial r} = \frac{-2}{\Gamma(\frac{d}{2})} \frac{r^{d-1}}{(\sqrt{2}\sigma_i)^{d-1}} \exp\left(-\frac{(\frac{r}{\sigma})^2}{2}\right), \tag{11}$$

of which also the derivative with respect to $\mathbf{w}_i$ is needed. Substitution leads to the following learning rule:

$$\Delta \mathbf{w}_i = \eta_w \frac{\mathbf{v} - \mathbf{w}_i}{\sigma_i^2}, \tag{12}$$

where we have omitted the additional term $(d-1)\frac{\mathbf{v}-\mathbf{w}_i}{\|\mathbf{v}-\mathbf{w}_i\|^2}$, in order to simplify the rule, since it converges towards the same result. The learning rule for the kernel radius σ_i can be derived directly (thus, without simplifications):

$$\Delta \sigma_i = \eta_\sigma \frac{\partial H}{\partial \sigma_i} = \eta_\sigma \frac{1}{\sigma_i} \left(\frac{\|\mathbf{v} - \mathbf{w}_i\|^2}{d\sigma_i^2} - 1 \right), \tag{13}$$

with η_σ the learning rate.

These learning rules are not yet complete: indeed, since they are identical for all neurons, we will obtain identical solutions for the kernel centers and radii. The differential entropy of each kernel's output will be maximal, but since the kernels are further identical, their overlap will be maximal and their outputs maximally statistically dependent. In order to reduce the statistical dependency, and to let the kernels span the input space, we need an additional component in the learning process. We can formulate statistical dependency in information-theoretic terms as the mutual information *between* kernel outputs. Hence, we maximize the differential entropy given that we also need to minimize the mutual information, in order to cope with statistical dependency. This dual goal is captured by maximizing the joint entropy of the kernel outputs: $H(y_1, y_2, \ldots, y_N) = \sum_{i=1}^{N} H(y_i) - I(y_1, y_2, \ldots, y_N)$.

We will perform mutual information minimization heuristically by putting kernel adaptation in a competitive learning context. We opt for an *activity-based* competition between the lattice neurons: $i^* = \arg\max_{\forall i \in A} y_i$, ("winner-takes-all", WTA)[2]. In this way, the winning neuron's kernel will decrease its range – in particular when it is strongly active – and, thus, decrease its overlap with the surrounding kernels. We will verify later whether such an heuristic is reasonable, by verifying the algorithm's performance for a case for which the theoretically-optimal result can be derived analytically.

Finally, we need to supply topological information to the learning process. We will do this in the traditional way by not only updating the kernel parameters of the winning neuron, but also of its lattice neighbors (cooperative stage). Hence, the complete learning algorithm becomes:

$$\Delta \mathbf{w}_i = \eta_w \, \Lambda(i, i^*, \sigma_\Lambda) \frac{\mathbf{v} - \mathbf{w}_i}{\sigma_i^2}, \tag{14}$$

$$\Delta \sigma_i = \eta_\sigma \, \Lambda(i, i^*, \sigma_\Lambda) \frac{1}{\sigma_i} \left(\frac{\|\mathbf{v} - \mathbf{w}_i\|^2}{d\sigma_i^2} - 1 \right), \quad \forall i, \tag{15}$$

[2] Note that this definition of the winning neuron differs from the one resulting from the (minimum) Euclidean distance rule ($i^* = \arg\min_i \|\mathbf{w}_i - \mathbf{v}\|$), which is more commonly used in topographic map formation algorithms.

with Λ the neighborhood function, a monotonous decreasing function of the lattice distance from the winner, *e.g.*, a Gaussian:

$$\Lambda(i, i^*, \sigma_\Lambda) = \exp\left(-\frac{\|\mathbf{r}_i - \mathbf{r}_{i^*}\|^2}{2\sigma_\Lambda^2}\right), \tag{16}$$

with σ_Λ the neighborhood function range, and $\mathbf{r}_i$ neuron i's lattice coordinate (we assume a discrete lattice with a rectangular topology). We adopt the following neighborhood cooling scheme:

$$\sigma_\Lambda(t) \;=\; \sigma_{\Lambda 0} \exp\left(-2\sigma_{\Lambda 0}\frac{t}{t_{max}}\right), \tag{17}$$

with t the present time step, t_{max} the maximum number of time steps, and $\sigma_{\Lambda 0}$ the range spanned by the neighborhood function at $t = 0$.

3.1 Lattice-disentangling dynamics

In order to show the algorithm's lattice-disentangling dynamics, we consider the standard case of a square lattice and a square uniform input density. We take a 24×24 planar lattice and a two-dimensional uniform input density $[0, 1]^2$. The initial weights are randomly chosen from the same input density. The radii are initialized randomly by sampling the uniform distribution $(0, 0.1]$. We take $t_{max} = 2,000,000$ and $\sigma_{\Lambda 0} = 12$ and keep the learning rate fixed at $\eta_w = 0.01$ and $\eta_\sigma = 10^{-4}\eta_w$. The results are shown in Fig. 3 for the weights and in Fig. 4 for the radii.

Figure 3: about here.

Figure 4: about here.

---

4 Correspondence with other algorithms for kernel-based topographic map formation

In this section, we will relate our algorithm to other algorithms for kernel-based topographic map formation, provided that they develop their kernels in the input space directly. The Soft Topographic Vector Quantization (STVQ) algorithm (Graepel *et al.*, 1997) is a fixed point iteration algorithm (iterative contraction mapping) that performs a fuzzy assignment of data points to clusters, whereby each cluster corresponds to a single neuron. The weight vectors represent the cluster centers, and they are determined by iterating the following equilibrium equation (put into our format):

$$\mathbf{w}_i = \frac{\int_V \mathbf{v} \sum_j \Lambda(i,j) P(\mathbf{v} \in \mathcal{C}_j)\; p(\mathbf{v})\; d\mathbf{v}}{\int_V \sum_j \Lambda(i,j) P(\mathbf{v} \in \mathcal{C}_j)\; p(\mathbf{v})\; d\mathbf{v}}, \quad \forall i \in A, \tag{18}$$

with $P(\mathbf{v} \in \mathcal{C}_j)$ the assignment probability of data point $\mathbf{v}$ to cluster $\mathcal{C}_j$ (*i.e.*, the probability of "activating" neuron j), which is given by:

$$P(\mathbf{v} \in \mathcal{C}_j) = \frac{\exp\left(-\frac{\beta}{2} \sum_k \Lambda(j,k) \ \|\mathbf{v} - \mathbf{w}_k\|^2\right)}{\sum_l \exp\left(-\frac{\beta}{2} \sum_k \Lambda(l,k) \ \|\mathbf{v} - \mathbf{w}_k\|^2\right)}, \tag{19}$$

with β the inverse temperature parameter, and $\Lambda(i,j)$ the transition probability of the noise-induced change of data point $\mathbf{v}$ from cluster $\mathcal{C}_i$ to $\mathcal{C}_j$. The algorithm can be regarded as a general model for probabilistic, SOM-based topographic map formation by putting $\beta \to \infty$ in eq. (19), and $\Lambda(i,j) = \delta_{ij}$ in eq. (19) and/or eq. (18). For example, the Soft-SOM (SSOM) algorithm (Graepel *et al.*, 1997) is obtained by putting $\Lambda(i,j) = \delta_{ij}$ in eq. (19), but not in eq. (18). Kohonen's Batch Map version (Kohonen, 1995) is obtained for $\beta \to \infty$ and $\Lambda(i,j) = \delta_{ij}$ in eq. (19), but not in eq. (18), and for $i^* = \arg\min_j \|\mathbf{v} - \mathbf{w}_j\|^2$ (*i.e.*, distance-based "winner-takes-all" rule).

Our joint entropy maximization algorithm differs from the STVQ algorithm in at least three ways. First, the STVQ kernel represents a *fuzzy membership (in clusters) function*, *i.e.*, the softmax function, normalized with respect to the other lattice neurons. In our case, the kernel represents a cumulative distribution function, which operates in the input space, and determines the winning neuron. Second, instead of using kernels with equal radii, as in the STVQ algorithm, our radii are individually adapted. Third, the kernels also differ conceptually since in the STVQ algorithm, the kernel radii are related to the magnitude of the noise-induced change in the cluster assignment (thus, in lattice space), whereas in our case they are related to the radii of the incomplete gamma distribution kernels and, by consequence, to the standard deviations of the assumed Gaussian local input densities (thus, in input space).

In the kernel-based Maximum Entropy learning Rule (kMER) (Van Hulle, 1998), the kernel outputs are thresholded (0/1 activations) and, depending on these binary activations, the kernel centers and radii are adapted. In the joint entropy maximization algorithm, both the activation states as well as the definition of the winning neuron i^* in the learning rules eqs. (14,15) depend on the continuously-graded kernel outputs.

Our joint entropy maximization algorithm is also different from the Local Density Estimation (LDE) algorithm (Van Hulle, 2002). The LDE algorithm updates the kernel centers as well as the kernel radii. The learning rule for the kernel centers is:

$$\Delta \mathbf{w}_i = \eta_w \ \Lambda(i, i^*, \sigma_\Lambda) \ \frac{(\mathbf{v} - \mathbf{w}_i)}{\sigma_i^2} K(\mathbf{v}, \mathbf{w}_i, \sigma_i), \tag{20}$$

with $K(.)$ a Gaussian kernel, and that for the kernel radii:

$$\Delta \sigma_i = \eta_\sigma \ \Lambda(i, i^*, \sigma_\Lambda) \ \frac{1}{\sigma_i} \left(\frac{\|\mathbf{v} - \mathbf{w}_i\|^2}{\sigma_i^2} - \rho d \right) K(\mathbf{v}, \mathbf{w}_i, \sigma_i), \tag{21}$$

with ρ a scale factor, and with the winning neuron defined as $i^* =$

$\arg\max_{\forall i \in A}(K(\mathbf{v}, \mathbf{w}_i, \sigma_i))$. Hence, not only the learning rules of the two algorithms differ, but also their kernel definitions.

Finally, there is an interesting point of correspondence with the learning algorithm suggested by Yin and Allinson (2001). They started with Benaim and Tomasini's idea of adapting the kernel centers in such a manner that the Kullback-Leibler divergence of the map's output is minimized (Benaim and Tomasini, 1991), and extended it by also adapting the kernel radii. When assuming equal mixings of radially-symmetrical Gaussians, their learning rules for the kernel centers and radii become (using their simplifications):

$$\Delta \mathbf{w}_i = \eta_w \, \widehat{P}(\mathbf{v}|i) \, (\mathbf{v} - \mathbf{w}_i), \tag{22}$$

$$\Delta \sigma_i = \eta_\sigma \, \widehat{P}(\mathbf{v}|i) \, \left(\|\mathbf{v} - \mathbf{w}_i\|^2 - d\sigma_i^2 \right), \quad \forall i, \tag{23}$$

which look structurally similar to eqs. (14,15), respectively, on condition that $\widehat{P}(\mathbf{v}|i)$ is regarded as a neighborhood function. In fact, this is exactly what Yin and Allinson consider as the role of $\widehat{P}(\mathbf{v}|i)$ (p. 408). Following their derivation, $\widehat{P}(\mathbf{v}|i)$ represents neuron i's posterior probability of activation, which is a function of the neurons' Gaussian kernels, hence, their neighborhood function is expressed in input space coordinates. There are three marked distinctions with our learning rules. First, in our case, $\widehat{P}(\mathbf{v}|i) = \delta_{i\,i^*}$, with i^* the neuron that wins the competition. Second, we still have a separate term $\Lambda(.)$ for our neighborhood function, eq. (16), which is expressed in lattice space coordinates (thus, not in input space coordinates). Third, in Yin and Allinson's case, the winning neuron i^* is the one for which $\widehat{P}(\mathbf{v}|i)$ is maximal, $i^* = \arg\max_{\forall i \in A} \widehat{P}(\mathbf{v}|i)$, thus a density-based rule, instead of an activity-based rule, as in our case. In the next section, we will show by simulations that these distinctions lead to quite different kernel distributions.

5 Simulations

5.1 Theoretically-optimal joint entropy performance

What is now the joint entropy maximization performance of our algorithm? And what is the effect of the heuristic we have adopted for minimizing the mutual information? In order to address these questions, we consider the standard normal distribution, since for this case the theoretically-optimal joint entropy JE and mutual information MI can be expressed analytically:

$$\begin{aligned} \mathrm{MI} &= \log_2 2N + 2\left(\frac{N-1}{N}\log_2\frac{N}{N-1} + \frac{1}{N}\right) + \log_2\frac{1}{2N} \\ &\quad + (N-2)\left(\frac{3}{2N}\log_2\frac{2N}{3} + \frac{2N-3}{2N}\log_2\frac{2N}{2N-3}\right) \ \text{(bits)}, \end{aligned} \tag{24}$$

$$\mathrm{JE} = \log_2 2N \ \text{(bits)}, \tag{25}$$

with the second term in MI being present for $N > 2$ only, and with k the number of intervals (bins) into which we uniformly quantize the kernel outputs. Figure 5 shows JE and MI as a function N, parameterized with respect to k; the plots for

the $k > 2$ cases are determined by optimizing the kernel parameters numerically for a Gaussian distribution centered at the origin and with unit variance, $\mathrm{G}(0,1)$. We observe that MI continues to increase when N increases, and quickly becomes larger than JE.

We run our joint entropy maximization algorithm 20 times for each (N,k) combination shown in Fig. 5, given the $\mathrm{G}(0,1)$ input distribution, and plot the mean and standard deviations for the JE and MI obtained (Fig. 5, thin continuous lines). We observe that the difference between the theoretical and the obtained JE curves is small (the curves almost completely coincide). We also observe that there is a larger spread on the MI results than on the JE results, but the difference between the average and the theoretical MI curves is again small.

Figure 5: about here.

5.2 Density estimation performance

The next question is the density estimation performance of our algorithm. Since each kernel is adapted under the assumption that the input density is locally Gaussian, with mean the corresponding neuron's kernel center, and standard deviation the neuron's kernel radius, we can estimate the input density $p(\mathbf{v})$ in a way similar to the classic *Variable Kernel* density estimation method (Silverman, 1992):

$$\widehat{p}(\mathbf{v}) = \sum_{i=1}^{N} \frac{\exp(-\frac{\|\mathbf{v}-\mathbf{w}_i\|^2}{2(\sigma_i)^2})}{N(2\pi)^{\frac{d}{2}}\sigma_i^d}. \tag{26}$$

In order to assess the density estimation performance of our algorithm, we will consider a synthetic and a real-world example. The synthetic example is the aforementioned standard normal distribution example; the real-world example is a classic benchmark for density estimation algorithms. Furthermore, we will also run the other kernel-based topographic map formation algorithms mentioned in section 4: the algorithm of Yin and Allinson (2001), the STVQ and SSOM algorithms (Graepel *et al.*, 1997), the kMER algorithm (Van Hulle, 1998), and the LDE algorithm (Van Hulle, 2002).

5.2.1 Synthetic example

We train a 9 neuron, one-dimensional lattice in one-dimensional space and take $\eta_w = 0.001$, $\eta_\sigma = 0.1\eta_w$, $t_{max} = 1,000,000$ and $\sigma_{\Lambda 0} = 4.5$. Note that, albeit the input is Gaussian, the distribution of inputs that activate a given kernel will be non-Gaussian, due to our activity-based competition rule. The density estimate $\widehat{p}(\mathbf{v})$ and the original Gaussian distribution $p(\mathbf{v})$ are shown in Fig. 6A. We observe a good correspondence between $\widehat{p}(\mathbf{v})$ and $p(\mathbf{v})$. The mean squared error (MSE) is $9.35\ 10^{-5}$ when calculated for 100 uniform positions in the range shown. When we quantize the kernel outputs into $k = 2$ intervals, then the joint entropy is 4.14 bits and the mutual information 1.32 bits.

In the case of Yin and Allinson's learning algorithm, we use the same learning rate, cooling scheme, and number of iterations as in our joint entropy maximization algorithm. The resulting density estimate $\widehat{p}(\mathbf{v})$ is shown in Fig. 6B. The MSE equals 1.09 10^{-3}, the joint entropy 3.18 bits, and the mutual information 0.56 bits (also for $k = 2$ intervals). The MSE performance improves if the neighborhood range does not vanish (p. 407 in Yin and Allinson, 2001). If we optimize for this range, then we obtain a MSE of 1.18 10^{-4}, with 3.85 bits joint entropy and 3.67 bits mutual information. We observe that the increase in joint entropy goes at the expense of a more rapid increase in mutual information, as expected.

In the case of the other algorithms mentioned in section 4, we run all simulations on the same input data, using the cooling scheme of our joint entropy maximization algorithm, except for the SSOM and STVQ algorithms. For the STVQ algorithm, we take for the neighborhood function radius $\sigma_\Lambda = 0.5$, and for the equal and constant kernel radii $\frac{1}{\beta} = 0.01$, as suggested in (Graepel *et al.*, 1997). We also adopt these parameter values for the SSOM algorithm, since it is in fact a limiting case of the STVQ algorithm. Furthermore, again for the SSOM and STVQ algorithms, since they do not adapt their kernel radii, we look for the (common) kernel radius that optimizes the MSE between the estimated and the theoretical distributions. In this way, we at least know that a better MSE result cannot be obtained. We also optimize the ρ_s parameter of kMER in this way, for the same reason. The joint entropy, mutual information and MSE results are all summarized in Table 1, together with the algorithms' parameters. Note that, in the case of the Yin and Allinson algorithm, we list the range-optimized MSE result mentioned earlier.

Finally, for comparison's sake, we also list the theoretical-optimal joint entropy and mutual information performances (row labeled "theoretical"), which were discussed in the previous section. We observe in Table 1 that our joint entropy maximization algorithm ("max(JE)") not only yields the highest joint entropy value, but also achieves it with the lowest mutual information value, thus with the lowest degree of kernel overlap or map output redundancy.

Figure 6: about here.

Table 1: about here.

5.2.2 Real-world example

The second example is a classic benchmark in the density estimation literature. The data set consists of 222 observations of eruption lengths of the Old Faithful geyser in Yellowstone National Park, and was compiled by Weisberg (see Silverman, 1992). We repeat the previous simulations, using the same parameter settings and lattice sizes, except for the following modifications. For the Yin and Allinson algorithm, when optimizing for the neighborhood range, we aim at maximizing

the joint entropy, instead of minimizing the MSE, since we do not dispose of the theoretical density distribution. The common kernel radii of the SSOM and STVQ algorithms are, for the same reason, also optimized with respect to the joint entropy. In this way, we at least know that better joint entropy results cannot be obtained with these algorithms. The results are summarized in Table 2. We observe that our joint entropy maximization algorithm performs best in terms of the joint entropy, but also, that it performs best in terms mutual information.

The density estimate obtained with our joint entropy maximization algorithm is shown in Fig. 7A (thick continuous line). In order to show the dependence of the estimate on the number of kernels used, we also plot the result for $N = 25$ kernels (Fig. 7B). The joint entropy and mutual information are now 4.57 and 6.06 bits, respectively (again, for $k = 2$ intervals). Finally, for comparison's sake, we also consider a more traditional variable kernel density estimation method, namely, the adaptive unbiased cross-validation (adaptive UCV) method (Sain and Scott, 1996) which allocates a kernel at each data point. The adaptive UCV method has been shown to yield particularly good results, compared to other methods, on the Old Faithful geyser data set (Sain and Scott, 1996). The adaptive UCV result is superimposed on the density estimates shown in Fig. 7A,B (thick dashed lines). We observe that the density estimate for $N = 25$ more closely approximates the UCV result. However, one should be aware of the fact that the UCV result is also an approximation of the unknown true density distribution.

Figure 7: about here.

Table 2: about here.

---

6 Conclusion

We have introduced a new learning algorithm for kernel-based topographic map formation that is aimed at maximizing the map's joint entropy. The kernel parameters are adjusted so as to maximize the differential entropies of the kernel outputs and, at the same time, to minimize the mutual information between these outputs. The former is achieved by maximizing the information transfer through each kernel individually, and the latter, in a heuristic sense, by having a competitive stage in the learning process. The kernel output functions approximate the cumulative distributions of the assumed Gaussian local input densities. In the near future, we intend to explore how the density estimate formed by the maps can be used for clustering purposes.

References

Amari, S.-I., Cichocki, A., and Yang, H. (1996). A new learning algorithm for blind signal separation. In *Advances in Neural Processing Systems* (Vol. 8), D.S. Touretzky, M. Mozer and M. Hasselmo (Eds.), pp. 757-763, Cambridge, MA: MIT Press.

András, P. (2001). Kernel-Kohonen networks. *Int. J. Neural Systems*, submitted.

Bell A.J., and Sejnowski, T.J. (1995). An information-maximization approach to blind separation and blind deconvolution. *Neural Computat.*, **7**, 1129-1159.

Benaim, M., and Tomasini, L. (1991). Competitive and self-organizing algorithms based on the minimization of an information criterion. *Proc. ICANN'91*, pp. 391-396.

Bishop, C.M., Svensén, M., and Williams, C.K.I. (1998). GTM: The generative topographic mapping. *Neural Computat.*, **10**, 215-234.

Cardoso, J.-F., and Laheld, B. (1996). Equivariant adaptive source separation. *IEEE Trans. on Signal Processing*, **44**(12), 3017-3030.

Durbin, R., and Willshaw, D. (1987). An analogue approach to the travelling salesman problem using an elastic net method. *Nature*, **326**, 689-691.

Graepel, T., Burger, M., and Obermayer, K. (1997). Phase transitions in stochastic self-organizing maps. *Physical Rev. E*, **56**(4), 3876-3890.

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

Graham, R.L., Knuth, D.E., and Patashnik, O. (1994). Answer to problem 9.60 in *Concrete Mathematics: A Foundation for Computer Science.* Reading, MA: Addison-Wesley.

Kohonen, T. (1982). Self-organized formation of topologically correct feature maps. *Biol. Cybern.*, **43**, 59-69.

Kohonen, T. (1995). *Self-organizing maps.* Heidelberg: Springer.

Kostiainen, T., and Lampinen, J. (2002). Generative probability density model in the self-organizing map. In *Self-organizing neural networks: Recent advances and applications*, U. Seiffert & L. Jain (Eds.), pp. 75-94. Heidelberg: Physica Verlag.

Linsker, R. (1989). How to generate ordered maps by maximizing the mutual information between input and output signals. *Neural Computat.*, **1**, 402-411.

Park, J., and Sandberg, I.W. (1993). Approximation and radial-basis function networks. *Neural Comput.*, **5**, 305-316.

Sain, S.R., and Scott, D.W. (1996). On Locally Adaptive Density Estimation. *Journal of the American Statistical Association*, **91**, 1525-1534.

Silverman, B.W. (1992). *Density estimation for statistics and data analysis.* London: Chapman and Hall.

Sum, J., Leung, C.-S., Chan, L.-W., and Xu, L. (1997). Yet another algorithm which can generate topography map. *IEEE Trans. Neural Networks*, **8**(5), 1204-1207.

Utsugi, A. (1997). Hyperparameter selection for self-organizing maps. *Neural Computat.*, **9**, 623-635.

Vapnik, V.N. (1995). *The nature of statistical learning theory.* New York: Springer-Verlag.

Van Hulle, M.M. (1998). Kernel-based equiprobabilistic topographic map formation. *Neural Computat.*, **10**(7), 1847-1871.

Van Hulle, M.M. (2000). *Faithful Representations and topographic maps: From distortion- to information-based self-organization*, New York: Wiley.

Van Hulle, M.M. (2002). Kernel-based topographic map formation by local density modeling. *Neural Computat.*, **14**, in press.

Weisstein, E.W. (1999). *CRC Concise Encyclpdia od Mathematics.* London: Chapman and Hall.

Yin, H., and Allinson, N.M. (2001). Self-organizing mixture networks for probability density estimation. *IEEE Trans. Neural Networks*, **12**, 405-411.

Appendix: Kernel definition and universal approximation

Consider the following two-layered network for regression purposes:

$$O(\mathbf{v}) = \sum_{i=1}^{N} W_i \; K(\frac{\mathbf{v} - \mathbf{w}_i}{\sigma_i}), \tag{27}$$

with $O(.)$ the output of the network's single output node, $\mathbf{v}$ the network's input, with $\mathbf{v} \in \mathcal{R}^d$, N the number of kernel nodes in the hidden layer, $\mathbf{w}_i$ and σ_i the center and radius of the ith kernel node, respectively, and W_i the (scalar) weight from the ith kernel node to the output node. Note that we will only consider networks with single outputs and scalar function approximation; the extension to multiple outputs and multidimensional function approximation is evident. Furthermore, we will restrict ourselves to approximation in L_2-space and assume gradient descent learning (on the squared regression error) for obtaining the network parameters.

The question is now: is the network supplemented with our incomplete gamma distribution kernel capable of approximating arbitrary well any function? Or, in other words, is such a network broad enough for "universal approximation"?

According to Proposition 1 in (Park and Sandberg, 1993), our kernel should satisfy the following two conditions in L_2-space:

$$\begin{aligned} &\textstyle\int_{\mathcal{R}^d} K(\mathbf{v}) \; d\mathbf{v} \neq 0, \\ &\textstyle\int_{\mathcal{R}^d} |K(\mathbf{v})|^2 \; d\mathbf{v} < \infty. \end{aligned} \tag{28}$$

Proof:

Since we are only considering kernels individually, and since each kernel is radially-symmetrical around its center, we simplify the notation of our kernel as $K(\mathbf{v})$, instead of $K(\mathbf{v}, \mathbf{w}_i, \sigma_i)$, or even as $K(z)$, with $z = \frac{|v - w_i|}{\sqrt{2}\sigma_i}$.

Condition 1:

Since $K(\mathbf{v}) > 0$, $\forall \mathbf{v} \in \mathcal{R}^d$, $\int_{\mathcal{R}^d} K(\mathbf{v}) \; d\mathbf{v} > 0$. Hence, condition 1 is satisfied.

Condition 2:

1-dimensional case

For the $d = 1$ case, we have that $K(z) = P(\frac{1}{2}, z^2) = \text{erfc}(z) = 1 - \text{erf}(z)$.

Since $K(v) \leq 1 \; \forall v \in \mathcal{R}$, we have that $|K(v)|^2 \leq K(v)$, $\forall v \in \mathcal{R}$, and hence, $\int_{\mathcal{R}} |K(v)|^2 \; dv \leq \int_{\mathcal{R}} K(v) \; dv$. Furthermore, since our kernel is radially symmetric: $K^2(z) < K(z)$, $\forall z \in (0, \infty)$.

We have the following upper bound for the erf function (Weisstein, 1999, p. 561):

$$\exp(z^2) \int_z^\infty \exp(-t^2) \; dt \;\; \leq \;\; \frac{1}{z + \sqrt{z^2 + \frac{4}{\pi}}}, \quad \forall z \in [0, \infty). \tag{29}$$

Now since $\mathrm{erf}(z) = 1 - \frac{2}{\sqrt{\pi}} \int_z^\infty \exp(-t^2)\, dt$, we can re-write the previous equation as follows:

$$\mathrm{erf}(z) \;\geq\; 1 - \frac{2}{\sqrt{\pi}} \frac{\exp(-z^2)}{z + \sqrt{z^2 + \frac{4}{\pi}}}, \quad \forall z \in [0, \infty). \tag{30}$$

Since $K(z) = erfc(z) = 1 - \mathrm{erf}(z)$, we obtain the following upper bound, after some algebraic manipulations:

$$K(z) \leq \frac{2}{\sqrt{\pi}} \frac{\exp(-z^2)}{z + \sqrt{z^2 + \frac{4}{\pi}}} \leq \exp(-z^2), \quad \forall z \in [0, \infty). \tag{31}$$

Hence, since $|K(z)|^2 \leq K(z)$, $\forall z$, we have that:

$$\int_0^\infty |K(z)|^2\, dz \leq \int_0^\infty K(z)\, dz < \int_0^\infty \exp(-z^2)\, dz, \tag{32}$$

and, since:

$$\exp(-z^2) = (\pi)^{d/2}\, \mathrm{G}(0, \frac{1}{\sqrt{2}}), \tag{33}$$

with $\mathrm{G}(0, \frac{1}{\sqrt{2}})$ the d-dimensional, radially-symmetrical Gaussian with zero-mean and standard deviation $\frac{1}{\sqrt{2}}$, we have that:

$$\int_0^\infty |K(z)|^2\, dz < \frac{1}{2}(\pi)^{d/2} < \infty. \tag{34}$$

d-dimensional case

We construct the following upper bound for our kernel in the d-dimensional case, $K(z) = P(\frac{d}{2}, z^2)$, with $z = \frac{\|\mathbf{v} - \mathbf{w}_i\|}{\sqrt{2}\sigma_i}$: we take $K^{\mathrm{up}}(z) \triangleq 1$, for $z \in [0, a)$, and $K^{\mathrm{up}}(z) \triangleq \mathrm{erfc}(z - a)$ for $z \in [a, \infty)$, *i.e.*, a shifted erfc function, shifted over a distance $a \geq 0$.

It is clear that, when $K^{\mathrm{up}}(z) \geq K(z)$, for $z \in [0, \infty)$, and when $\int_0^\infty K^{\mathrm{up}}(z)$ is finite, then $\int_0^\infty K(z)$ will also be finite. This is already satisfied for $z \in [0, a)$ since $K^{\mathrm{up}}(z) = 1 \geq K(z)$. Hence, we only have to show that there exists a positive, finite shift a so that $K^{\mathrm{up}}(z) \geq K(z)$ for $z \in [a, \infty)$.

We have that:

$$\begin{aligned} K^{\mathrm{up}}(z) &= \mathrm{erfc}(z - a) = \frac{2}{\sqrt{\pi}} \int_z^\infty \exp^{-(t-a)^2}\, dt \\ &= \frac{2}{\sqrt{\pi}} \int_z^\infty \exp^{2ta - a^2} \cdot \exp^{-t^2}\, dt. \end{aligned} \tag{35}$$

For our kernel $K(z)$, we rewrite the incomplete gamma distribution as follows:

$$\begin{aligned} P(\frac{d}{2}, z^2) &= \frac{\Gamma(\frac{d}{2}, z^2)}{\Gamma(\frac{d}{2})} = \frac{1}{\Gamma(\frac{d}{2})} \int_{z^2}^\infty s^{\frac{d}{2}-1} \cdot \exp^{-s}\, ds \\ &= \frac{2}{\Gamma(\frac{d}{2})} \int_z^\infty t^{d-1} \cdot \exp^{-t^2}\, dt, \end{aligned} \tag{36}$$

where we have substituted s for t^2.

We will achieve $K^{\text{up}}(z) \geq K(z)$, when the non-common arguments in the integrals eqs. (35,36) satisfy: $\frac{\exp^{2ta-a^2}}{\sqrt{\pi}} \geq \frac{t^{d-1}}{\Gamma(\frac{d}{2})}$. Based on the series expansion of exp we have that, in conservative sense: $\frac{(2ta-a^2)^{d-1}}{(d-1)!\sqrt{\pi}} > \frac{t^{d-1}}{\Gamma(\frac{d}{2})}$. Since these arguments represent straight lines, as a function of t, the one on the left-hand side will remain above the one on the right-hand side when $a > \left(\frac{\sqrt{\pi}(d-1)!}{\Gamma(\frac{d}{2})}\right)^{\frac{1}{d-1}}$, *i.e.*, a positive and finite shift of the erfc function in $K^{\text{up}}(z)$ for $z \in [a, \infty)$.

QED

Nomenclature

A : discrete lattice consisting of N formal neurons

d : number of dimensions in V-space

$\Delta \mathbf{w}_i$: update of kernel center of neuron i

$\Delta \sigma_i$: update of kernel radius of neuron i

η_σ: learning rate for kernel radii

η_w: learning rate for kernel centers ("weights")

$h(v)$: (sigmoidal) output function

$\Gamma(.)$: gamma distribution

$H(y_1)$: differential entropy of y_1

$H(y_1, y_2)$: joint entropy of y_1 and y_2

i : a neuron's label

i^* : neuron winning the competition

$I(y_1, y_2)$: mutual information between y_1 and y_2

JE: joint entropy

k: number of quantization intervals (of kernel output)

$K(.)$: kernel output function

Λ: neighborhood function

MI: mutual information

MSE : mean squared error

N : number of lattice neurons

$p(\mathbf{v})$: input density distribution

$p_{\chi^2}(.)$: chi-squared distribution

$p_{y_i}(.)$: density of kernel output y_i of neuron i

$P(\alpha, x)$: (complement of the) incomplete gamma distribution of x with α degrees of freedom

$\widehat{P}(\mathbf{v}|i)$: neuron i's posterior probability of activation

$\mathbf{r}_i$: neuron i's lattice coordinate

σ: standard deviation (of Gaussian distribution)

σ_i: kernel radius neuron i

σ_Λ: neighborhood function range

$\sigma_{\Lambda 0}$: initial neighborhood range

SOM : Kohonen's Self-Organizing (feature) Map

t : simulation time (time step)

t_{max} : maximum number of time steps

V : d-dimensional space of input signals

$\mathbf{v} = [v_1, ..., v_d]$: input vector in d-dimensions

WTA : winner-takes-all

$\mathbf{w}_i = [w_{i1}, ..., w_{id}]$: kernel center ("weight vector") of i-th neuron in d dimensions

y_i: kernel output of neuron i

Table Legends

Table 1: Performance of our joint entropy maximization algorithm ("max(JE)"), and that of five other kernel-based topographic map algorithms (see text), when $N = 9$ kernels are developed on a standard normal input distribution. The parameter settings of the algorithms are listed. The performance is measured in terms of the joint entropy (JE) and the mutual information (MI) of the kernel outputs ($k = 2$ quantization levels), and the mean squared error (MSE) between the obtained and the theoretical density distributions. Both the JE and MI are expressed in bits.

Table 2: Performance in the case of the Old Faithful geyser eruption lengths data set. The performance is measured in terms of the joint entropy (JE) and the mutual information (MI), for $k = 2$ quantization levels, and are expressed in bits.

Algorithm	Parameters	JE	MI	MSE
max(JE)	$\eta_w = 0.001$, $\eta_\sigma = 0.1\eta_w$	4.14	1.32	$9.35\ 10^{-5}$
Yin & Allinson	$\eta_w = 0.001$, $\eta_\sigma = 0.1\eta_w$	3.85	3.67	$1.18\ 10^{-4}$
STVQ	$\frac{1}{\beta} = 0.01$, $\sigma_\Lambda = 0.5$, $\sigma_{\text{opt}} = 0.555$	3.59	2.42	$1.99\ 10^{-3}$
SSOM	$\frac{1}{\beta} = 0.01$, $\sigma_\Lambda = 0.5$, $\sigma_{\text{opt}} = 0.520$	3.78	2.03	$1.99\ 10^{-3}$
LDE	$\eta_w = 0.01$, $\eta_\sigma = 0.1\eta_w$	3.90	3.15	$7.05\ 10^{-4}$
kMER	$\eta_w = 0.001$, $\rho_r = 1$, $\rho_s = 4.58$	4.01	4.01	$1.75\ 10^{-4}$
theoretical	–	4.17	1.39	–

Table 1

Algorithm	Parameters	JE	MI
max(JE)	$\eta_w = 0.001$, $\eta_\sigma = 0.1\eta_w$	4.02	1.36
Yin & Allinson	$\eta_w = 0.001$, $\eta_\sigma = 0.1\eta_w$	3.90	1.55
STVQ	$\frac{1}{\beta} = 0.01$, $\sigma_\Lambda = 0.5$, $\sigma_{\text{opt}} = 0.350$	3.50	1.92
SSOM	$\frac{1}{\beta} = 0.01$, $\sigma_\Lambda = 0.5$, $\sigma_{\text{opt}} = 0.350$	3.84	2.20
LDE	$\eta_w = 0.01$, $\eta_\sigma = 0.1\eta_w$	3.66	2.33
kMER	$\eta_w = 0.001$, $\rho_r = 1$, $\rho_s = 2.20$	3.72	2.25

Table 2

Figure legends

Figure 1: Kernel-based topographic maps. Example of a 2×2 map (*cf.* rectangle in V-space) for which each neuron has a Gaussian kernel as output function. For each neuron, a circle is drawn with center the neuron weight vector and radius the kernel range.

Figure 2: Distribution functions of the Euclidean distance r from the mean of a unit-variance, radially-symmetrical Gaussian input density eq. (5), parameterized with respect to the dimensionality d (continuous lines), and the corresponding complements of the cumulative distribution functions eq. (6) (dashed lines). The functions are plotted for $d = 1, \ldots, 10$ (from left to right); the thick lines correspond to the $d = 1$ case.

Figure 3: Evolution of a 24×24 lattice with a rectangular topology as a function of time. The outer squares outline the uniform input distribution $[0, 1]^2$. The values given below the squares represent time.

Figure 4: Evolution of the kernel radii corresponding to Fig. 3.

Figure 5: Theoretically-optimal joint entropy (JE) eq. (25) (thick continuous lines) and mutual information (MI) eq. (24) (thick dashed lines) for the case of a one-dimensional, univariate Gaussian plotted as a function of the number of neurons N. The JE and MI are plotted for $k = 2, 4, 8, 16, 32$ quantization levels, starting with $k = 2$ curves at the bottom and moving upwards. The thin continuous lines and their error bars correspond to the mean and the standard deviations of the mutual information found in 20 simulation runs. The mean joint entropy curves coincide with the theoretically-expected curves; the error bars are not shown since they are everywhere smaller than $2.2\ 10^{-2}$.

Figure 6: One-dimensional, unit-variance Gaussian input density (thick dashed line) and the estimates (thick continuous lines) obtained with our learning algorithm (A) and Yin and Allinson's (B) when 9 kernels are used (thin continuous lines). Abbreviation: pd = probability density.

Figure 7: Density estimation in the case of the Old Faithful geyser eruption lengths data set. Density estimates obtained with our learning algorithm (thick continuous lines), using $N = 9$ kernels (A) and $N = 25$ kernels (B), and with the adaptive UCV method (thick dashed lines). The individual kernels of our algorithm are also shown (thin continuous lines).

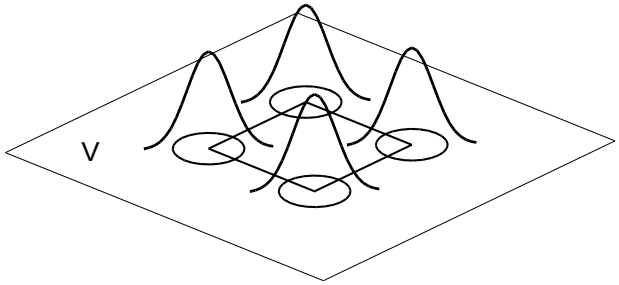

Figure 1

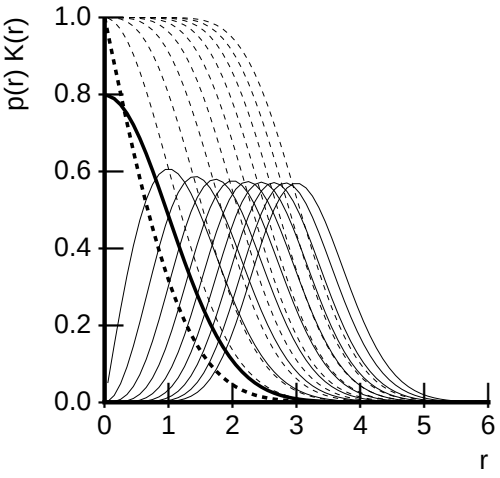

Figure 2

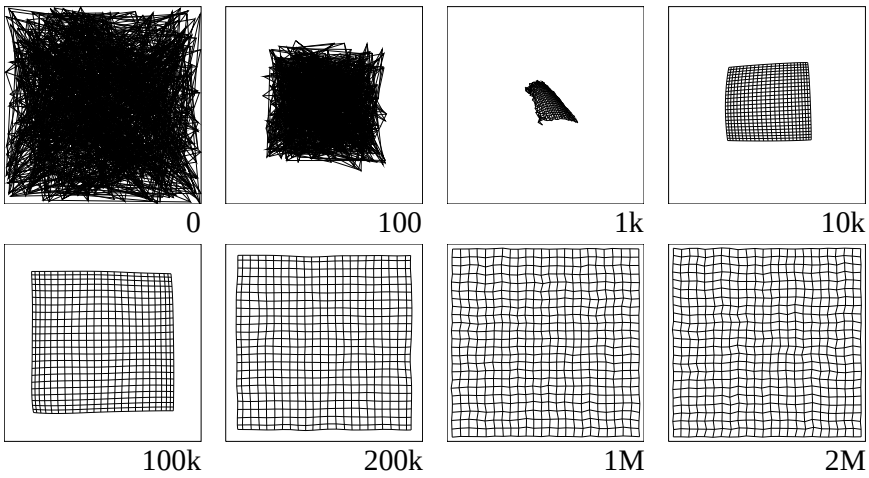

Figure 3

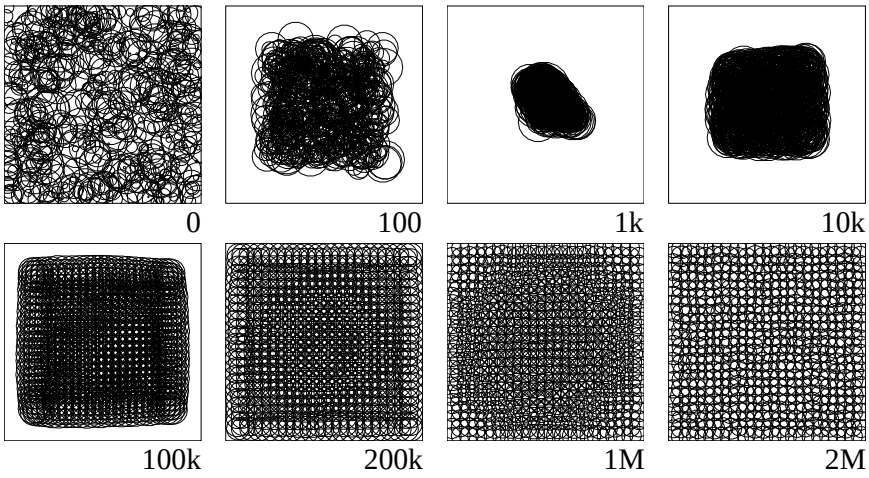

Figure 4

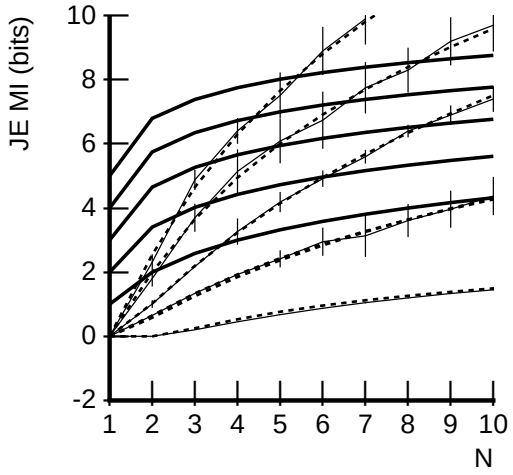

Figure 5

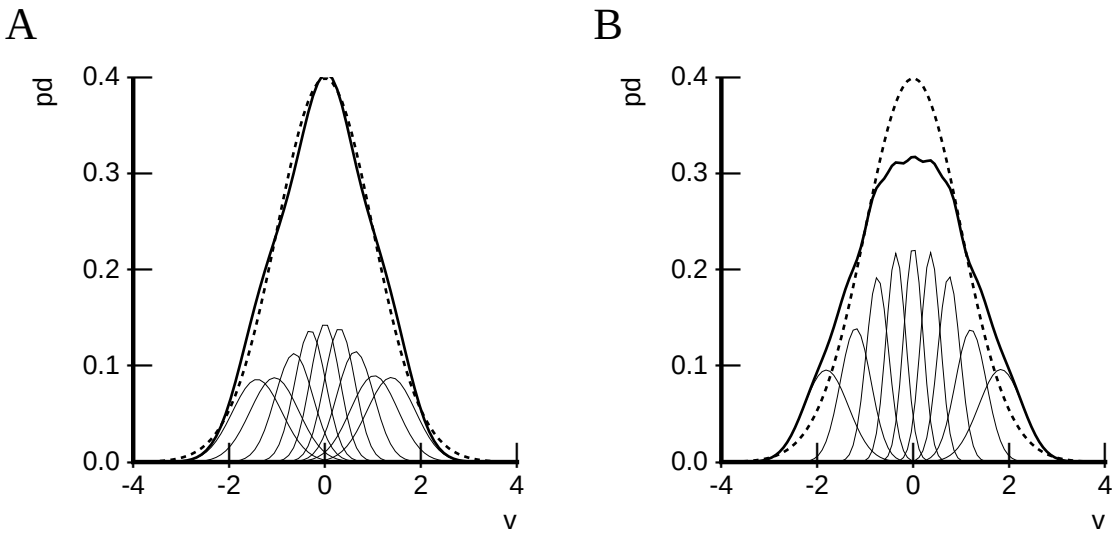

Figure 6

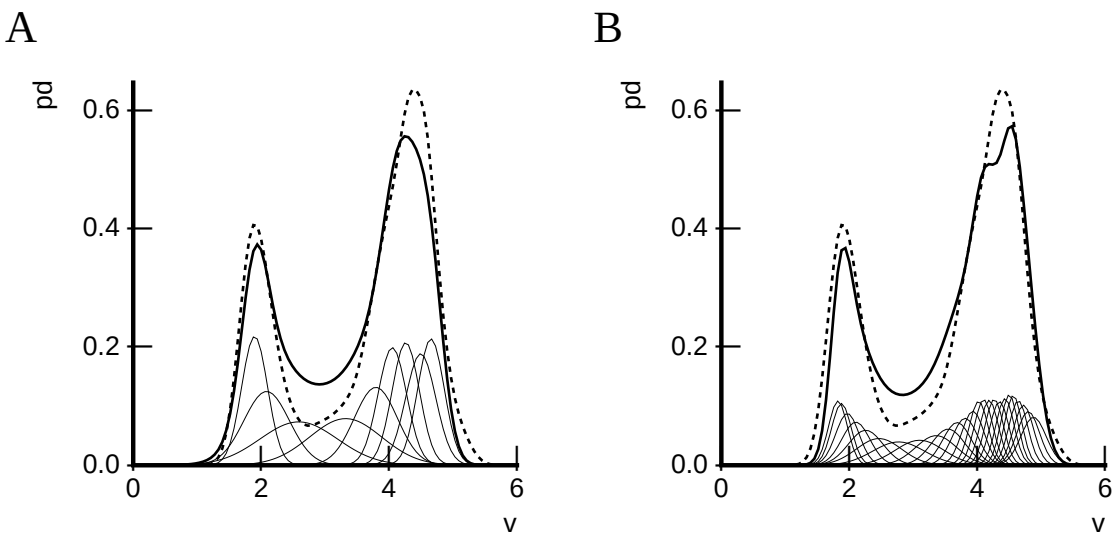

Figure 7