

Finding Clusters and Components by Unsupervised Learning

Erkki Oja*

Helsinki University of Technology, Neural Networks Research Centre
P.O.Box 5400, 02015 HUT, Finland
erkki.oja@hut.fi

Abstract. We present a tutorial survey on some recent approaches to unsupervised machine learning in the context of statistical pattern recognition. In statistical PR, there are two classical categories for unsupervised learning methods and models: first, variations of Principal Component Analysis and Factor Analysis, and second, learning vector coding or clustering methods. These are the starting-point in this article. The more recent trend in unsupervised learning is to consider this problem in the framework of probabilistic generative models. If it is possible to build and estimate a model that explains the data in terms of some latent variables, key insights may be obtained into the true nature and structure of the data. This approach is also reviewed, with examples such as linear and nonlinear independent component analysis and topological maps.

1 Introduction: Supervised and Unsupervised Learning from Data

In statistical pattern recognition, machine learning from a training set is an essential technique. If the classes of the training vectors are known, supervised methods are used to build the classifiers. If class information does not exist, one has to resort to unsupervised methods. Also in the preliminary stage of feature extraction unsupervised methods are mostly used [13, 30].

The optimality of supervised classifiers is given by the theoretical limit of the Bayes decision rule. For unsupervised methods, no such clear optimality criterion exists. Usually, the result of unsupervised learning is a new explanation or representation of the observation data, which will then lead to improved future decisions. In statistical pattern recognition, the representation may be a clustering of the data, a discrete map, or a continuous lower-dimensional manifold in the vector space of observations, which explains their structure and may reveal their underlying causes [13].

Unsupervised learning seems to be the basic mechanism for sensory adaptation in the animal brain, e.g. in the visual pathway [4]. In pattern recognition, it is a highly powerful and promising approach to some practical problems like

* This work was supported by the Academy of Finland as part of its Center of Excellence project “New Information Processing Principles”.

data mining and knowledge discovery from very large databases, or new modes of human-computer interactions in which the software adapts to the requirements and habits of the human user by observing her behaviour. For an excellent collection of recent articles on unsupervised learning, see [23].

The current trend in unsupervised learning is to consider this problem in the framework of probabilistic generative models. The concept of a generative model is very general and potentially powerful. In fact, as discussed by Roweis and Ghahramani [48], a large number of central techniques like FA, PCA, mixtures of Gaussians, vector quantization, and also dynamical models like Kalman filters or Hidden Markov Models, can be presented in a unified framework of unsupervised learning under a single basic generative model. If it is possible to build and estimate a model that explains the data in terms of some latent variables, key insights may be obtained into the true nature and structure of the data. Operations like prediction and compression become easier and rigorously justifiable. In this paper, we take a brief look at such models, which reveal the structure of the data by projections on linear or nonlinear structures, spanned by components or clusters hidden in the data.

The first class of unsupervised learning methods we consider in Section 2 is motivated by standard statistical methods like PCA or FA, which give a reduced subset of linear combinations of the original input variables. Also nonlinear variants have been suggested, such as autoassociative neural networks, kernel PCA, principal curves and surfaces, and mixtures of local PCA's. A more recent model in this category is that of independent components, which would maximally reduce the redundancy between the latent variables even in the case that gaussianity does not hold. This leads to the techniques of Independent Component Analysis (ICA) and Blind Source Separation (BSS) [27]. In the latter technique, a set of parallel time signals such as speech waveforms, electromagnetic measurements from the brain, or financial time series, are assumed to be linear combinations of underlying independent latent variables. The variables, called independent components, are found by efficient ICA learning rules. ICA is a linear technique, but nonlinear variants have been proposed recently, and some approaches along Nonlinear ICA or Nonlinear FA are also pointed out in Section 2.

The second class of methods is close to clustering or visualization by projecting the data on a nonlinear low-dimensional grid. A typical application is data mining or profiling from massive databases. It is of interest to find out what kind of typical clusters there are among the data records, and what is the relation between the clusters. A competitive learning algorithm gives an efficient solution to this problem. Section 3 briefly reviews a well-known competitive learning system, the Self-Organizing Map (SOM) [36], and a related generative latent variable model GTM [7].

2 Finding Independent Components

2.1 Principal Component Analysis

Principal component analysis (PCA) and the closely related Karhunen-Loève Transform, or the Hotelling Transform, as well as Factor Analysis (FA), are clas-

sical techniques in statistical data analysis, feature extraction, and data compression [14, 40, 61]. Given a set of multivariate measurement vectors $\mathbf{x}(1), \dots, \mathbf{x}(T)$, the purpose is to find a smaller set of variables with less redundancy, that would give as good a representation as possible. The redundancy is measured through second-order statistics only and is removed by decorrelation. This means rotating the data into a new coordinate system given by the eigenvectors of the data covariance matrix.

It is not always feasible to solve the eigenvectors by standard numerical methods. In an on-line data compression application like image or speech coding, the data samples $\mathbf{x}(t)$ arrive at high speed, and it may not be possible to estimate the covariance matrix and solve the eigenvector-eigenvalue problem once and for all.

An alternative is to derive gradient ascent algorithms or other on-line methods for PCA. The algorithms will then converge to the solution of the problem, that is, to the eigenvectors. The advantage of this approach is that such algorithms work on-line, using each input vector $\mathbf{x}(t)$ once as it becomes available and making an incremental change to the eigenvector estimates, without computing the covariance matrix at all. This approach is the basis of the PCA neural network learning rules introduced by the author [39, 42]. Other related on-line algorithms have been introduced in [16, 49, 14, 60]. Some of them, like the APEX algorithm by Diamantaras and Kung [14], are based on a feedback neural network. Also minor components defined by the eigenvectors corresponding to the smallest eigenvalues can be computed by similar algorithms [42].

Another possibility for PCA computation in neural networks is the Multi-Layer Perceptron network, which learns using the back-propagation algorithm (see [20]) in unsupervised autoassociative mode. In autoassociative mode, the same vectors \mathbf{x} are used both as inputs and as desired outputs in back-propagation learning. This network with nonlinear hidden layer was suggested for data compression by [11], and it was shown to be closely connected to the theoretical PCA by [8]. It is not equivalent to PCA, however, as shown by [31], unless the hidden layer is linear. A much more powerful network is obtained if more hidden layers are added. For instance, a 5 - layer autoassociative MLP is able to compute in principle any smooth nonlinear mapping between the inputs and the central hidden layer, and another mapping between the central hidden layer and the outputs. This is due to the two extra nonlinear hidden layers; see e.g. [41]. This network is one way to compute a nonlinear PCA expansion.

Other prominent approaches to extend PCA to nonlinearities are the kernel PCA [52] and the method of Principal Curves [19]. PCA can be “kernelized” because it is a second-order statistical technique. Yet another approach to construct nonlinear PCA manifolds is to combine the two major unsupervised learning paradigms - PCA and vector coding (VQ) - using mixtures of local linear models, for example PCA’s, in which the data cloud is first clustered or parcelled using VQ, and then a separate linear model is fitted to each of the clusters around the code vector. This notion has been formalized by several authors [22, 47, 48, 55, 62]. It is closely related to the conventional technique of semipara-

metric density estimation, the Mixture of Gaussians (MoG) model widely used in clustering and data modelling. However, instead of using full covariance matrices for the component gaussians, the local linear models constrain the covariances in a natural and easily adjustable way.

Another linear projection technique is Factor Analysis (FA) [18], in which a generative latent variable model is assumed for \mathbf{x} :

$$\mathbf{x} = \mathbf{A}\mathbf{y} + \mathbf{n}. \quad (1)$$

With certain assumptions on the additive noise, FA and PCA produce the same solution. PCA, too, can be derived from a generative model in the technique called Probabilistic PCA [55] or Principal Factor Analysis [18].

2.2 Independent Component Analysis

In Independent Component Analysis (ICA) [1, 5, 9, 10, 25, 27, 32, 34, 43] the same model (1) is assumed, but now the assumption on y_i is much stronger: we require that they are statistically independent and nongaussian. Interestingly, then the ambiguity in Factor Analysis disappears and the solution, if we can find one, is (almost) unique.

In the simplest form of ICA, the additive noise \mathbf{n} is not included and the standard notation for the independent components or sources is s_i ; thus the ICA model for observation vectors \mathbf{x} is

$$\mathbf{x} = \mathbf{A}\mathbf{s}. \quad (2)$$

It is assumed that both \mathbf{x} and \mathbf{s} are zero mean. The observations x_i are now linear combinations or mixtures of the sources s_j . The matrix \mathbf{A} is called in ICA the mixing matrix. The model looks deceptively simple but is not, because both \mathbf{A} and \mathbf{s} are unknown and must be estimated from a sample of the observations \mathbf{x} .

We may further assume that the dimensions of \mathbf{x} and \mathbf{s} are the same. If originally $\dim \mathbf{x} < \dim \mathbf{s}$, or there are more sources than observed variables, then the problem becomes quite difficult - see [27]. If, on the other hand, $m = \dim \mathbf{x} > \dim \mathbf{s} = n$, then model (2) implies that there is redundancy in \mathbf{x} which is revealed and can be removed by performing PCA on \mathbf{x} . This is done as follows.

We can write the $m \times m$ covariance matrix of \mathbf{x} as

$$\mathbf{C}_{\mathbf{x}} = \mathbf{A}\mathbf{E}\{\mathbf{s}\mathbf{s}^T\}\mathbf{A}^T = \mathbf{A}\mathbf{A}^T. \quad (3)$$

We have used the knowledge that matrix $\mathbf{E}\{\mathbf{s}\mathbf{s}^T\}$ is diagonal, due to the fact that the elements of \mathbf{s} are zero mean and independent; if we further absorb their variances to matrix \mathbf{A} and assume that $\mathbf{E}\{s_i^2\} = 1$, then it holds $\mathbf{E}\{\mathbf{s}\mathbf{s}^T\} = \mathbf{I}$. Now, matrix \mathbf{A} is an $m \times n$ matrix and so matrix $\mathbf{C}_{\mathbf{x}} = \mathbf{A}\mathbf{A}^T$ is an $m \times m$ matrix with rank n . It will have only n nonzero eigenvalues. Let us denote the diagonal $n \times n$ matrix of the nonzero eigenvalues of $\mathbf{C}_{\mathbf{x}}$ by \mathbf{D} , the orthonormal eigenvectors of $\mathbf{C}_{\mathbf{x}}$ by $\mathbf{e}_1, \dots, \mathbf{e}_m$, and the orthogonal matrix that has the n first

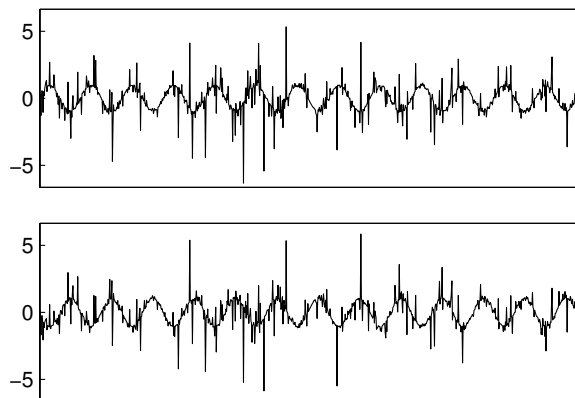


Fig. 1. Mixed signals

ones as columns by \mathbf{E} . Thus \mathbf{E} is $m \times n$. Make now a linear transformation for the m - dimensional observation vectors \mathbf{x} :

$$\mathbf{x}' = \mathbf{D}^{-1/2} \mathbf{E}^T \mathbf{x}. \quad (4)$$

For the covariance matrix of the transformed n - dimensional vector \mathbf{x}' it holds:

$$\mathbb{E}\{\mathbf{x}' \mathbf{x}'^T\} = \mathbf{D}^{-1/2} \mathbf{E}^T \mathbf{C}_{\mathbf{x}} \mathbf{E} \mathbf{D}^{-1/2} = \mathbf{D}^{-1/2} \mathbf{E}^T \mathbf{E} \mathbf{D} \mathbf{D}^{-1/2} = \mathbf{I}. \quad (5)$$

This transformation is called whitening. Let us assume in the following that whitening has always been performed in the model, and denote simply by \mathbf{x} the whitened observation vector whose dimension is the same as that of the source vector \mathbf{s} .

Whitening has another desirable side-effect, which can be seen by noting from eq. (3) that now $\mathbf{A} \mathbf{A}^T = \mathbf{I}$. But this means that matrix \mathbf{A} is an orthogonal matrix, for which $\mathbf{A}^{-1} = \mathbf{A}^T$. So, if we knew matrix \mathbf{A} , we could directly solve the unknown source vector \mathbf{s} from the model by

$$\mathbf{s} = \mathbf{A}^T \mathbf{x}.$$

It is an interesting finding that very few assumptions suffice for solving the mixing matrix and, hence, the sources. All we need is the assumption that the sources s_i are statistically independent and nongaussian. Consider the following simple example: we have two signals, shown in Fig. 1, that are linear combinations or mixtures of two underlying independent nongaussian source signals. This example is related to model (2) in such a way that the elements x_1, x_2 of the random vector \mathbf{x} in (2) are the amplitudes of the two signals in Fig. 1. The signals provide a sample $\mathbf{x}(1), \dots, \mathbf{x}(T)$ from this two-dimensional random vector. The joint histogram of the sample vectors is plotted in Fig. 2; each point in the scatter plot corresponds to one time point in Fig. 1. The vector \mathbf{x} is now white in the sense that x_1 and x_2 are zero mean, uncorrelated, and have unit variance.

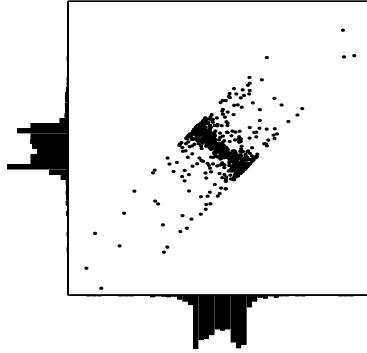


Fig. 2. Histogram of the two amplitudes of the mixed signals x_1, x_2

This may not be apparent from the histogram but can be verified by estimating the covariance matrix of all the points.

The example suggests a method that in fact is highly useful and forms the basis of some practical ICA algorithms. Consider a line passing through the origin at the center of the data cloud in Fig. 2. Denote a unit vector defining the direction of the line by \mathbf{w} . Then the projection of a data point \mathbf{x} on the line is given by $y = \mathbf{w}^T \mathbf{x}$. This can be considered as a random variable whose density is approximated by the histogram of the projections of all the data points in the cloud on this line. No matter what is the orientation of the line, it always holds that y has zero mean and unit variance. The unit variance is due to $E\{y^2\} = E\{(\mathbf{w}^T \mathbf{x})^2\} = \mathbf{w}^T E\{\mathbf{x}\mathbf{x}^T\} \mathbf{w} = \mathbf{w}^T \mathbf{w} = 1$ where we have used the facts that \mathbf{x} is white and \mathbf{w} has unit norm.

However, it is easy to see from Fig. 2 that the density of y will certainly vary as the orientation of the line varies, meaning that all the moments of y cannot stay constant. In fact, any other moment than the first and second ones is not constant. What is most important is that any such moment, say, $E\{y^3\}$ or $E\{y^4\}$ or in fact $E\{G(y)\}$, with $G(y)$ a nonlinear and non-quadratic function, will attain a number of maxima and minima when the orientation of the line goes full circle, and some of these extrema coincide with orientations in which the 2-dimensional density factorizes into the product of its marginal densities - meaning independence.

In Fig. 3, the coordinate system has been rotated so that the fourth moment $E\{y^4\}$ is maximal in the vertical direction and minimal in the horizontal direction. We have found two new variables $y_1 = \mathbf{w}_1^T \mathbf{x}$ and $y_2 = \mathbf{w}_2^T \mathbf{x}$, with $\mathbf{w}_1, \mathbf{w}_2$ orthonormal, that satisfy

$$p(y_1, y_2) = p(y_1)p(y_2)$$

with $p(\cdot)$ the appropriate probability densities. The variables are thus independent and it holds

$$\mathbf{y} = \mathbf{W}\mathbf{x}$$

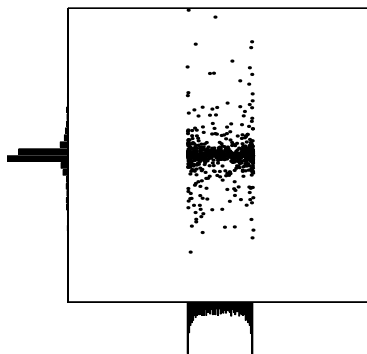


Fig. 3. Histogram of the two amplitudes of the separated signals y_1, y_2

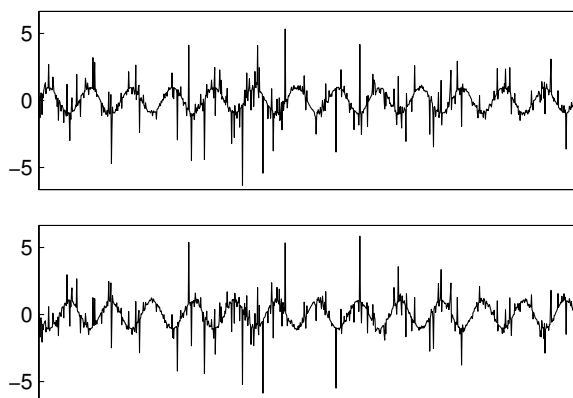


Fig. 4. Separated signals

where $\mathbf{W} = (\mathbf{w}_1 \mathbf{w}_2)^T$. We have solved the inverse of the model (2) and obviously found the mixing matrix: $\mathbf{A} = \mathbf{W}^T$.

Fig. 4 shows y_1, y_2 again arranged in their correct time order. It is seen that they form two signals, one a random nongaussian noise and the other one a deterministic sinusoid. These were in fact the original signals that were used to make the artificial mixtures in Fig. 1. In the context of separating time series or signals, the ICA technique is an example of blind signal separation.

The above illustrative example can be formalized to an efficient mathematical algorithm. What we need is a numerical method to maximize, say, the fourth moment $E\{y^4\}$ in terms of a unit norm weight vector \mathbf{w} . A possibility is gradient ascent: the gradient of $E\{y^4\}$ with respect to \mathbf{w} is $4E\{y^3 \mathbf{x}\} = 4E\{(\mathbf{w}^T \mathbf{x})^3 \mathbf{x}\}$. However, gradient methods are notoriously slow. A better idea is a fast algorithm with higher-order convergence speed. Such a method is provided by the FAsTICA algorithm. For finding one independent component (one weight vector \mathbf{w}), the algorithm is as follows:

1. Choose the initial value randomly for the weight vector \mathbf{w} .
2. Repeat Steps 3,4 until the algorithm has converged:
3. Normalize \mathbf{w} to unit norm.
4. Update \mathbf{w} by

$$\mathbf{w} \leftarrow \mathbf{E}\{(\mathbf{w}^T \mathbf{x})^3 \mathbf{x}\} - 3\mathbf{w} \quad (6)$$

This algorithm was introduced in [25] and further extended and analyzed in [26]; for a detailed review, see [27]. The FastICA algorithm is available in public-domain software [15] from the author's web pages. The algorithm can be run either in a deflation mode, in which the orthogonal weight vectors (columns of the mixing matrix \mathbf{A}) can be found one at a time, or in a parallel mode, in which all the independent components and the whole matrix \mathbf{A} are solved in one iteration.

An analysis of the local maxima and minima of a general cost function $\mathbf{E}\{G(y)\} = \mathbf{E}\{G(\mathbf{w}^T \mathbf{x})\}$ over the unit sphere $\|\mathbf{w}\| = 1$ was made by the author in [45]. The result is

Theorem. Under the linear mixing model $\mathbf{x} = \mathbf{A}\mathbf{s}$, with whitened \mathbf{x} (hence: orthogonal \mathbf{A}), the local maxima (resp. minima) of $\mathbf{E}\{G(\mathbf{w}^T \mathbf{x})\}$ under the constraint $\|\mathbf{w}\| = 1$ include those columns \mathbf{a}_i of the mixing matrix \mathbf{A} such that the corresponding sources s_i satisfy

$$\mathbf{E}\{s_i g(s_i) - g'(s_i)\} > 0 \text{ (resp. } < 0) \quad (7)$$

where $g(\cdot)$ is the derivative of $G(\cdot)$.

The Theorem essentially says that all the columns of the mixing matrix will be among the local minima or maxima of $\mathbf{E}\{G(\mathbf{w}^T \mathbf{x})\}$, but there may be also other extrema. The condition (7) states that some columns (and the corresponding sources) are found by minimizing, others by maximizing. For the case $G(y) = y^4$, (7) becomes

$$\mathbf{E}\{s_i^4 - 3\} > 0$$

(note that the sources have unit variances). The term on the left hand side is the kurtosis of s_i . Thus, the positively kurtotic sources are found at the local maxima of $\mathbf{E}\{(\mathbf{w}^T \mathbf{x})^4\}$ and vice versa. For other cost functions $G(y)$, the condition (7) always splits the sources in two groups, too.

In [27], the above method of fourth order moment maximization is shown to be an example of a powerful criterion of finding maximally nongaussian orthogonal directions through the multidimensional density $p(\mathbf{x})$. Cost functions like maximum likelihood or minimal mutual information are shown to be intimately related to this basic criterion. Other algorithms to solving the basic linear ICA model have been reported e.g. by [1, 5, 9, 10, 32], as reviewed in [27]. Especially, if the sources are actually signals with time structure, not just samples of random variables, then blind separation can be achieved using either temporal correlations [6] or nonstationarity [46].

2.3 Nonlinear Factor and Independent Component Analysis

The model (2) is extremely simple and can be extended in several directions. If the additive noise cannot be assumed to be zero, we have the noisy ICA model, also termed independent factor analysis [2]. This is due to the fact that it is otherwise similar to the factor analysis model (1), with the difference that the factors y_i are not uncorrelated (thus independent) gaussians, but rather independent nongaussians. Some solution methods are reviewed in [27].

Another extension is nonlinear ICA and FA. Instead of the linear models (2),(1), consider

$$\mathbf{x} = \mathbf{f}(\mathbf{y}, \mathbf{M}) + \mathbf{n} \quad (8)$$

with \mathbf{f} a nonlinearity parameterized by an array of parameters \mathbf{M} . Vector \mathbf{y} gives a number of latent variables and \mathbf{n} is again gaussian noise. If we assume that the prior $p(\mathbf{y})$ for \mathbf{y} is gaussian with unit (or diagonal) covariance, making the elements y_i independent, the model (14) may be called nonlinear factor analysis. A further extension would be $p(\mathbf{y})$ that is nongaussian but factorizable so that the y_i are independent; then the model becomes nonlinear independent component analysis.

Several authors have attacked this problem. The baseline is that the problem is ill-defined. Under very general assumptions, a random vector can be transformed nonlinearly into another random vector that has independent elements [28], but there is no guarantee that the independent elements are the original sources. Therefore, the solution can only be sought with restrictions that somehow regularize the problem. Typical such restrictions are post-nonlinear mixtures and some special cases that can be reduced to linear mixtures with simple mappings. For general nonlinearities, there are a variety of methods, some of them rather ad hoc; for a review, see [33].

Recently, Valpola [56] used an approximation for the nonlinear function $\mathbf{f}(\mathbf{y}, \mathbf{M})$ in the model, that was based on a Multilayer Perceptron (MLP) network with one hidden layer. It is well-known [24, 17] that this function can approximate uniformly any continuous functions on compact input domains and it is therefore suitable for this task. Then the model becomes

$$\mathbf{x} = \mathbf{B}\phi(\mathbf{A}\mathbf{y} + \mathbf{a}) + \mathbf{b} + \mathbf{n} \quad (9)$$

where \mathbf{A}, \mathbf{a} are the weight matrix and offset vector of the hidden layer, ϕ is the sigmoidal nonlinearity, typically a \tanh or \sinh^{-1} function, and \mathbf{B}, \mathbf{b} are the weight matrix and offset vector of the linear output layer. It is understood that ϕ is applied to its argument vector element by element. In practice, there is a training sample $\mathbf{x}(1), \dots, \mathbf{x}(T)$, and we wish to solve from the model the corresponding source or factor vectors $\mathbf{y}(1), \dots, \mathbf{y}(T)$.

The problem now is that, contrary to the usual supervised learning situations, the inputs to the MLP are not known and therefore back-propagation type of learning rules cannot be used for finding the unknown parameters. The idea in [56] is to use a purely Bayesian approach called ensemble learning. The cost function is the Kullback - Leibler divergence between the true posterior

probability for the parameters, given the observations, and an approximation of that density. Denote the set of all the unknown parameters by $\Theta = \{\mathbf{Y}, \mathbf{M}\}$. There the vector \mathbf{Y} contains all the unknown source vectors $\mathbf{y}(1), \dots, \mathbf{y}(T)$, while \mathbf{M} contains the weights of the MLP network that define the unknown nonlinear function \mathbf{f} , and also the parameters of the gaussian noise \mathbf{n} . In addition, because this is a Bayesian model, it includes hyperparameters defining the distributions of the weights. Denote the sample of observations by $\mathbf{X} = \mathbf{x}(1), \dots, \mathbf{x}(T)$.

We can write for the posterior density of the parameters

$$p(\Theta|\mathbf{X}) = p(\mathbf{Y}, \mathbf{M}|\mathbf{X}) = \frac{p(\mathbf{X}|\mathbf{Y}, \mathbf{M})p(\mathbf{Y}|\mathbf{M})p(\mathbf{M})}{p(\mathbf{X})}. \quad (10)$$

The first term $p(\mathbf{X}|\mathbf{Y}, \mathbf{M})$ is obtained from the data model (9); it is simply a product of gaussians with means $\mathbf{B}\phi(\mathbf{A}\mathbf{y}(t) + \mathbf{a}) + \mathbf{b}$. Likewise, the terms $p(\mathbf{Y}|\mathbf{M})$ and $p(\mathbf{M})$ are obtained as products of gaussians, when we assume mutually independent gaussian priors for all the parameters. The term $p(\mathbf{X})$ does not contain any unknown parameters and can be omitted.

This density is now approximated by another density $q(\Theta)$ - the ensemble - that has a simple form [56]: it is a gaussian with diagonal covariance. Then the KL divergence

$$C_{KL} = \int d\Theta q(\Theta) \log \frac{q(\Theta)}{p(\Theta|\mathbf{X})} \quad (11)$$

also obtains a relatively simple form, splitting into the expectations of many simple terms. It can be minimized by a suitable numerical method.

In [56], several applications with real data are shown. The model is also extended to a dynamical model, similar to an extended Kalman filter but with unknown parameters, and very promising results are obtained in case studies [57, 29].

3 The Self-organizing Map

3.1 The Basic SOM

One of the best-known learning systems in the unsupervised category is the Self-Organizing Map (SOM) introduced by Kohonen [36]. It belongs to the class of vector coding algorithms. In vector coding, the problem is to place a fixed number of vectors, called codewords, into the input space which is usually a high-dimensional vector space. The dimension of the data vectors is determined by the problem and can be very large. In the WEBSOM system [37] for organizing collections of text documents, the dimensionality of the data in the largest applications is about $n = 50,000$ and the size of the training sample is about $T = 7,000,000$.

A well-known method for vector coding is the Linde-Buzo-Gray (LBG) algorithm, which is very similar to the k - means clustering algorithm [13]. Assume a set of nodes which are numbered by index $i = 1, \dots, k$, and assume that each

node i has a weight vector \mathbf{w}_i that has the same dimension as the input vectors \mathbf{x} that we wish to cluster. In k - means clustering, the goal is to place the weight vectors (codewords) into the input space in such a way that the average squared distance from each \mathbf{x} to its closest codeword is minimized. In the Self - Organizing Map (SOM), there is an extra feature compared to mere clustering: nodes are spatially arranged to a 1-, 2- or multidimensional lattice, such that each node has a set of neighbors. The goal of SOM learning is not only to find the most representative code vectors for the input training set in the sense of minimum distance, but at the same time to form a topological mapping from the input space to the grid of nodes. This idea originally stems from the modelling of the topographic maps on the sensory cortical areas of the brain. A related early work in neural modelling is [38].

For any data point \mathbf{x} in the input space, one or several of the codewords are closest to it. Assume that \mathbf{w}_i is the closest among all codewords:

$$\|\mathbf{x} - \mathbf{w}_i\| = \min\|\mathbf{x} - \mathbf{w}_j\|, j = 1, \dots, k \quad (12)$$

The unit i having the weight vector \mathbf{w}_i is then called the best-matching unit (BMU) for vector \mathbf{x} . The well-known Kohonen algorithm for self-organization of the code vectors is as follows [36]:

1. Choose initial values for the weight vectors \mathbf{w}_i .
2. Repeat Steps 3,4 until the algorithm has converged:
3. Draw a sample vector \mathbf{x} from the training set and find the best matching unit $i = i(\mathbf{x})$ according to Eq. (12).
4. Adjust the weight vectors of all units by

$$\mathbf{w}_j \leftarrow \mathbf{w}_j + \gamma * h_r * (\mathbf{x} - \mathbf{w}_j) \quad (13)$$

where γ is a gain factor and h_r is a function of the distance $r = \|i - j\|$ of units i and j measured along the lattice.

There are several choices for the initial values and for the neighborhood function h_r ; these, as well as the convergence and the mathematical properties of this algorithm have been considered by several authors, e.g. [36, 47, 44, 58]. For SOM learning, topology preservation, and its relation to a cost function, see [59, 12, 21]. A more efficient learning rule for the SOM is the batch algorithm, covered e.g. in [36]. The 2-dimensional map is also a powerful tool for data visualization: e.g., a color code can be used in which each unit has its own characteristic color. For a public domain software implementation of the SOM, with various graphical tools for map presentations as well as with preprocessing methods, see [54]. A database of well over four thousand applications of SOM is given by [53].

3.2 The Generative Topographic Map

There is a probabilistic generative model that is close to the SOM, the Generative Topographic Map (GTM) [7], in which the vectors \mathbf{x} are expressed in terms of

a number of latent variables, which are defined on a similar lattice or grid as the nodes in the SOM. Assume a grid with dimension l (usually, this would be equal to 2, at most 3), and assume there are k nodes \mathbf{y}_i , $i = 1, \dots, k$ on the grid. Assume a random latent variable \mathbf{y} , whose values are concentrated at these nodes. Let us make a nonlinear mapping from the l -dimensional random variable \mathbf{y} to the original n -dimensional vectors \mathbf{x} :

$$\mathbf{x} = \mathbf{f}(\mathbf{y}, \mathbf{M}) + \mathbf{n} \quad (14)$$

where \mathbf{M} is an array of parameters of the nonlinear function \mathbf{f} , and \mathbf{n} is additive noise. The form of the function \mathbf{f} is assumed to be determined except for the unknown parameters. The model (14) is the generative latent variable model of the GTM method. It means that the data \mathbf{x} are basically concentrated on an l -dimensional nonlinear manifold in the data space, except for the additive noise. The k vectors $\mathbf{w}_i = \mathbf{f}(\mathbf{y}_i, \mathbf{M})$ that are the images of the node points \mathbf{y}_i are analogous to the weight vectors or codewords of the SOM. If \mathbf{f} is smooth, a topographic ordering for the codewords is automatically guaranteed, because such an ordering is valid for the points \mathbf{y}_i . The GTM also has the advantage that it postulates a smooth manifold that naturally interpolates between the code vectors \mathbf{w}_i .

If we assume that the noise has a radially symmetrical gaussian density, then the density of \mathbf{x} , given \mathbf{y} , becomes a mixture of gaussians, having a separate gaussian density around each of the code vectors $\mathbf{w}_i = \mathbf{f}(\mathbf{y}_i, \mathbf{M})$. From this, the likelihood function for the parameters \mathbf{M}, β follows immediately. The EM algorithm can now be used to numerically solve the parameters by maximum likelihood, due to the mixture of gaussians form of the density - for details, see [7]. The reference also discusses the similarities and differences between GTM and SOM.

4 Conclusions

The two main paradigms of unsupervised machine learning in statistical pattern recognition have been reviewed: the extensions to the Principal Component Analysis technique, and the clustering, vector coding, and topological mapping technique. The first class of methods form a continuous linear or nonlinear transformation of the original input vectors to feature vectors of lower dimensionality, and are especially useful in feature extraction. The reduced representation given by the feature vectors would typically be input to a classifier.

The second class of methods are able to map highly nonlinear input data manifolds onto low dimensional lattices, preserving optimally the mutual topological relations of input vectors. Thus these methods, notably the Self-Organizing Map (SOM), are suitable for data clustering and visualization. The applications range from industrial quality control to financial data mining. Also generative latent variable versions for these basic models and their combinations were reviewed.

This paper was a review of the essential principles and theory underlying unsupervised learning, with some central references cited. It is not possible here

to give even a rudimentary list of applications of these techniques. There are available good text-books that cover some of the major approaches [23, 36, 14, 27].

References

1. Amari, S.-I., Cichocki, A. and Yang, H., "A new learning algorithm for blind source separation". In *Advances in Neural Information Processing Systems 8*, Cambridge: MIT Press, 1996, pp. 757 - 763.
2. Attias, H., "Independent factor analysis", *Neural Computation 11 (4)*, 1999, pp. 803 - 851.
3. Baldi, P. and Hornik, K., "Learning in linear neural networks: a survey", *IEEE Trans. Neural Networks 6 (4)*, 1995, pp. 837 - 858.
4. Barlow, H., "Unsupervised learning", *Neural Computation 1*, 1989, pp. 295 - 311.
5. Bell, A. and Sejnowski, T., "An information-maximization approach to blind separation and blind deconvolution", *Neural Computation 7*, 1995, pp. 1129 - 1159.
6. Belouchrani, A., Meraim, K., Cardoso, J-F., and Moulines, E., "A blind source separation technique based on second order statistics", *IEEE Trans. Signal Proc. 45*, 1997, pp. 434 - 444.
7. Bishop, C., Svensen, M and Williams, C., "GTM: the generative topographic mapping", *Neural Computation 10*, 1998, pp. 215 - 234.
8. Bourlard, H. and Kamp, Y., "Auto-association by multilayer Perceptrons and singular value decomposition", *Biol. Cybernetics 59*, 1988, pp. 291 - 294.
9. Cardoso, J.- F., "Blind signal separation: statistical principles", *Proc. of the IEEE 9 (10)*, 1998, pp. 2009 - 2025.
10. Cichocki, A. and Unbehauen, R., "Robust neural networks with on-line learning for blind identification and blind separation of sources", *IEEE Trans. on Circuits and Systems 43 (11)*, 1996, pp. 894 - 906.
11. Cottrell, G., Munro, P. and Zipser, D., "Learning internal representations from gray-scale images: an example of extensional programming", *Proc. 9th Ann. Conf. of the Cognitive Science Society*, 1987, pp. 462 - 473.
12. Der, R. and Herrmann, M., "Second-order learning in Self-Organizing Maps", in Oja, E. and Kaski, S. (Eds.), *Kohonen Maps*. Amsterdam: Elsevier, 1999, pp. 293 - 302.
13. De Vijver, P. and Kittler, J., *Pattern Recognition - a Statistical Approach*. Englewood Cliffs, NJ: Prentice-Hall, 1982.
14. Diamantaras, K. and Kung, S.: *Principal Component Neural Networks: Theory and Applications*. New York: J. Wiley & Sons, 1996.
15. The FastICA package. Available from www.cis.hut.fi/projects/ica/
16. Foldiak, P., "Adaptive network for optimal linear feature extraction", *Proc. Int. J. Conf. on Neural Networks*, Washington, DC, 1989, pp. 401 - 406.
17. Funahashi, K., "On the approximate realization of continuous mappings by neural networks", *Neural Networks 2*, 1989, pp. 183-192.
18. Harman, H.H., *Modern Factor Analysis*. Univ. of Chicago Press, 1967.
19. Hastie, T. and Stuetzle, W., "Principal curves", *J. Am. Statist. Assoc. 84*, 1989, pp. 502 - 516.
20. Haykin, S., *Neural Networks - a Comprehensive Foundation*. New York: MacMillan College Publ. Co., 1998.

21. Heskes, T., "Energy functions for Self-Organizing Maps", in Oja, E. and Kaski, S. (Eds.), *Kohonen Maps*. Amsterdam: Elsevier, 1999, pp. 303 - 316.
22. Hinton, G., Revow, M. and Dayan, P., "Recognizing handwritten digits using mixtures of linear models", in G. Tesauro, D. Touretzky, and T. Leen (Eds.), *Advances in Neural Information Processing Systems 6*, San Mateo: Kauffman, 1995, pp. 1015 - 1022.
23. Hinton, G. and Sejnowski, T.J., *Unsupervised Learning - Foundations of Neural Computation*. Cambridge, MA: MIT Press, 1999.
24. Hornik, M., Stinchcombe, M., and White, H., "Multilayer Feedforward Networks are Universal Approximators", *Neural Networks 2*, 1989, pp. 359-368.
25. Hyvärinen, A. and Oja, E., "A fast fixed-point algorithm for Independent Component Analysis", *Neural Computation 9 (7)*, 1997, pp. 1483 - 1492.
26. Hyvärinen, A., "Fast and robust fixed-point algorithms for Independent Component Analysis", *IEEE Trans. Neural Networks 10 (3)*, 1999, pp. 626 - 634.
27. Hyvärinen, A., Karhunen, J. and Oja, E., *Independent Component Analysis*. New York, Wiley, 2001.
28. Hyvärinen, A. and Pajunen, P., "Nonlinear independent component analysis: existence and uniqueness results", *Neural Networks 12*, 1999, pp. 429 - 439.
29. Ilin, A., Valpola, H. and Oja, E., "Nonlinear dynamical factor analysis for state change detection", *IEEE Trans. Neural Networks 15*, No. 3, May 2004.
30. Jain, A. K., Duin, P. W., and Mao, J., "Statistical pattern recognition: a review", *IEEE Trans. Pattern Analysis and Machine Intelligence 22*, 2000, pp. 4 - 37.
31. Japkowitz, N, Hanson, S and Gluck, A., "Nonlinear autoassociation is not equivalent to PCA", *Neural Computation 12 (3)*, 2000, pp. 531 - 545.
32. Jutten, C. and Herault, J., "Blind separation of sources, part I: an adaptive algorithm based on neuromimetic architecture", *Signal Processing 24*, 1991, pp. 1 - 10.
33. Jutten, C. and Karhunen, J., "Advances in nonlinear blind source separation", *Proc. 4th Int. Symp. on ICA and BSS*, Nara, Japan, April 1-4, 2003, pp. 245 - 256.
34. Karhunen, J., Oja, E., Wang, L., Vigario, R., and Joutsensalo, J., "A class of neural networks for independent component analysis", *IEEE Trans. on Neural Networks 8 (3)*, 1997, pp. 486 - 504.
35. Kendall, M and Stuart, A., *The Advanced Theory of Statistics, Vols. 1 - 3*. MacMillan, 1976 - 1979.
36. Kohonen, T., *Self-Organizing Maps*. Berlin: Springer, 1995. 3rd Edition, 2001.
37. Kohonen, T., Kaski, S., Lagus, K., Salojärvi, J., Paatero, V. and saarela, A., "Self organization of massive document collection", *IEEE Trans. Neural Networks 11 (3)*, 2000, pp. 574 - 585.
38. von der Malsburg, C., "Self-organization of orientation sensitive cells in the striate cortex", *Kybernetik 14*, 1973, pp. 85 - 100.
39. Oja, E. "A Simplified Neuron Model as a Principal Components Analyzer", *J. Math. Biol. 15*, 1982, pp. 267-273.
40. Oja, E., *Subspace Methods of Pattern Recognition*. Letchworth: RSP and J. Wiley, 1983.
41. Oja, E., "Data Compression, Feature Extraction, and Autoassociation in Feedforward Neural Networks", *Proc. ICANN-91*, Espoo, Finland, June 24 - 28, 1991, pp. 737 - 745.
42. Oja, E., "Principal Components, Minor Components, and Linear Neural Networks", *Neural Networks 5*, 1992, pp. 927 - 935.
43. Oja, E., "The nonlinear PCA learning rule in independent component analysis", *Neurocomputing 17 (1)*, 1997, 25 - 46.

44. Oja, E. and Kaski, S. (Eds.), *Kohonen Maps*. Amsterdam: Elsevier, 1999.
45. Oja, E. and Wang, L., "Neural fitting: robustness by anti-Hebbian learning", *Neurocomputing* 12, 1976, pp. 155 - 170.
46. Pham, D. and Cardoso, J-F., "Blind separation of instantaneous mixtures of non-stationary sources", *IEEE Trans. Signal Proc.* 49, 2001, pp. 1837 - 1848.
47. Ritter, H., Martinetz, T., and Schulten, K., *Neural Computation and Self-Organizing Maps: an Introduction*. Reading: Addison-Wesley, 1992.
48. Roweis, S. and Ghahramani, Z., "A unifying review of linear gaussian models", *Neural Computation* 11 (2), 1999, pp. 305 - 346.
49. Rubner, J. and Tavan, P., "A self-organizing network for Principal Component Analysis", *Europhysics Letters* 10 (7), 1989, pp. 693 - 698.
50. Sanger, T., "Optimal unsupervised learning in a single-layered linear feedforward network", *Neural Networks* 2, 1989, pp. 459 - 473.
51. Schalkoff, R., *Pattern Recognition - Statistical, Structural, and Neural Approaches*. J. Wiley, 1992.
52. Schölkopf, B., smola, A. J. and Müller, K.-R., "Nonlinear component analysis as a kernel eigenvalue problem", *Neural Comput.* 10, 1998, 1299 - 1319.
53. Bibliography of SOM papers: a reference list of over 4000 studies on the Self-Organizing Map. Available at www.cis.hut.fi/research/som-bibli/.
54. The SOM Toolbox for Matlab. Available at www.cis.hut.fi/projects/somtoolbox/.
55. Tipping, M. E. and Bishop, C. M., "Mixtures of probabilistic principal component analyzers", *Neural Computation* 11 (2), 1999, pp. 443 - 482.
56. Valpola, H., "Bayesian ensemble learning for nonlinear factor analysis", *Acta Polyt. Scand. Ma 108*, Espoo, 2000. D.Sc. Thesis, Helsinki University of Technology.
57. Valpola, H., Oja, E., Ilin, A., Honkela, A. and Karhunen, J., "Nonlinear blind source separation by variational Bayesian learning", *IEICE Trans. E86-A*, 2003, pp. 532 - 541.
58. VanHulle, M., *Faithful Representations and Topographic Maps*. New York: J. Wiley & Sons, 2000.
59. Villmann, T., "Topology preservation in Self-Organizing Maps". In Oja, E. and Kaski, S. (Eds.), *Kohonen Maps*. Amsterdam: Elsevier, 1999, pp. 267 - 292.
60. Wang, L. and Karhunen, J., "A unified neural bigradient algorithm for robust PCA and MCA", *Int. J. of Neural Systems* 7 (1), 1996, pp. 53 - 67.
61. Webb, A., *Statistical Pattern Recognition*. Arnold, 1999.
62. Xu, L., "Temporal BYY learning for state space approach, hidden Markov model, and blind source separation", *IEEE Trans. Signal Proc.* 48, 2000, pp. 2132 - 2144.