DATA ANALYSIS, VISUALIZATION, AND HIDDEN FACTOR DISCOVERY BY UNSUPERVISED LEARNING

Erkki Oja and Kimmo Kiviluoto

Helsinki University of Technology,
Laboratory of Computer and Information Science,
P.O.B. 2200, 02015 HUT, Finland
Email: `Erkki.Oja@hut.fi, Kimmo.Kiviluoto@hut.fi`

ABSTRACT

With the continuous increase in computing power, it has become possible to process and classify masses of natural data, such as statistical information, images, speech, as well as other kinds of signals and measurements coming from very different sources. Many problems occur in industry, finance, remote sensing, medicine, and natural sciences, to mention only a few main fields, in which one needs efficient tools for visualization, prediction, clustering, and profiling.

Often the explicit modelling of the processes underlying the measurements is very hard and so inferences from the measurement data must be made by learning methods. A widely used class of learning algorithms are the neural learning paradigms. In this paper, emphasis is on unsupervised neural learning. Especially the techniques of Self-Organizing Maps and Independent Component Analysis are reviewed and shown to be useful in this context. Some examples are shown on applications of these techniques on financial data analysis.

1. INTRODUCTION

A large majority of the present-day neural network research and applications have focused on *supervised learning* in feedforward networks like the Multi-Layer Perceptron network, the Radial Basis Function network, or the LVQ network [9]. Together with their powerful training algorithms, these networks provide highly efficient model-free methods for designing non-linear mappings between inputs and outputs using a database of training samples. Prominent examples are pattern recognition, optical character readers, industrial diagnostics, condition monitoring, modelling complex black box systems for control, and time series analysis and forecasting. There are a great deal of practical sample cases showing the power of supervised neural networks, and their relations to statistical methods like regression and Bayesian inference are well understood [6].

Unsupervised learning algorithms are the other important subclass of neural learning. The characteristic feature of unsupervised neural learning is that the training set only contains input samples. No desired outputs or target outputs are available at all. Basically, these algorithms fall into one of two categories [9]: first, extensions of the linear transform coding methods of statistics like Principal Component Analysis, and second, learning vector coding methods that are based on competitive learning.

Of the two main types of unsupervised learning outlined above, we review in this paper two algorithms: the *Self-Organizing Map*, which is a competitive learning method related to but not equivalent to vector coding; and *Independent Component Analysis*, which can be seen as an extension of PCA. Examples of these two methods on financial data analysis, visualization, and hidden factor recovery are given.

2. THE SELF-ORGANIZING MAP

One of the best-known neural networks in the unsupervised category is the Self-Organizing Map (SOM) introduced by Kohonen [17]. 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 real space. The input space is represented by a training sample $(\mathbf{x}_1, ..., \mathbf{x}_T)$. For example, the inputs can be measurements from a machine or a chemical process, or financial data describing a company or a customer. Typically, the codewords are found to correspond to relevant clusters among the input training data, e.g. customer groups or typical process states, and they can be efficiently used to cluster new inputs.

One way to understand the SOM [17], [18] is to consider it as a neural network implementation of vector coding: each codeword is the weight vector of a neural unit. However, there is an essential extra feature in the SOM. The neurons are arranged to a 1-, 2- or multi-dimensional lattice such that each neuron has a set of neighbors. The goal of learning is not only to find the most representative code vectors for the input training set in the mean square sense, but at the same time to build a topological mapping from the input space to the grid of neurons.

Mathematically, this can be defined as follows. For any data point $\mathbf{x}$ in the input space, one of the codewords is closest to it. Assume that $\mathbf{w}_i$ is the closest among all codewords:

$$\|\mathbf{x} - \mathbf{w}_i\| = min\|\mathbf{x} - \mathbf{w}_j\|, j = 1, ..., N \qquad (1)$$

The unit i having the weight vector $\mathbf{w}_i$ is then called the *best-matching unit* (BMU) for vector $\mathbf{x}$, and index $i = i(\mathbf{x})$ can be considered as the *output* of the map. Note that for fixed $\mathbf{x}$, Eq. (1) defines the index i of the BMU, and for fixed i, Eq. (1) defines the Voronoi set of unit i as the set of points $\mathbf{x}$ that satisfy (1). By the above relation, the input space is mapped to the discrete set of neurons.

By a topological mapping we mean that if an arbitrary point $\mathbf{x}$ is mapped to unit i, then all points in neighborhoods of $\mathbf{x}$ are mapped either to i itself or to one of the units in the neighborhood of i in the lattice. This implies that if i and j are two neighboring units on the lattice, then their Voronoi sets in the input space have a common boundary. Whether the topological property can hold for all units, however, depends on the dimensionalities of the input space and the neuron lattice: because no topological maps between two spaces of different dimensions can exist in the strict mathematical sense, a two-dimensional neural layer can only follow locally two dimensions of the multidimensional input space. Usually the input space has a much higher dimension, but the data cloud $(\mathbf{x}_1, ..., \mathbf{x}_T)$ used in training may be roughly concentrated on a lower-dimensional manifold, which the map is able to follow at least approximately [18].

The well-known Kohonen algorithm for self-organization of the code vectors is as follows [18]:

1. Choose initial values randomly for the weight vectors $\mathbf{w}_i$ of the units i

2. Repeat Steps 3,4 until the algorithm has converged:

3. Draw a sample $\mathbf{x}$ from the training set and find the best matching unit i according to Eq. (1)

4. Adjust the weight vectors of all units by

$$\mathbf{w}_j := \mathbf{w}_j + \gamma * h_r * (\mathbf{x} - \mathbf{w}_j) \qquad (2)$$

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.

(In the original version [17], the neighborhood function h_r was equal to 1 for a certain neighborhood of i, and 0 elsewhere. The neighborhood and the gain γ should slowly decrease in time).

The convergence and the mathematical properties of this algorithm have been considered by several authors, e.g. [18], [20].

3. SOM IN CORPORATE BANKRUPTCY PREDICTION

The Self-Organizing Map has been found to be a valuable tool for analyzing financial statements. In a number of earlier studies [2, 14, 21, 22], the SOM has been used for data visualization, and in some cases also for classification of companies into healthy and bankruptcy-prone ones.

The material used in our recent studies [15] consists of Finnish small and medium-sized enterprises, using the line of business, age, and size as the selection criteria. It was also required that the history and state of the enterprise be known well enough: if there were no data available for a longer period than two years before the bankruptcy, or if the last known financial statements were very poor, the company was rejected from the sample. In the final sample, there were 8 484 financial statements given by 2 116 companies, of which 568 have gone bankrupt.

The financial indicators used for training the SOM are three commonly used ratios that measure the profitability and solidity of a company: (i) operating margin and rents, (ii) net income, and (iii) equity ratio. In addition to these, (iv) net income of the previous year is also included; together with the net income of the present year, it reflects the change in profitability. Before training, the indicators were first preprocessed using histogram equalization separately for each indicator.

The SOM after training is shown in figure 1. In each subfigure, one component of the weight vectors associated with the map units is displayed. The gray-level coloring shows the relative values of that component in different parts of the map, black corresponding to "bad" values – either low values of financial ratios, or high proportion of bankruptcies. Note how the profitability generally increases when going downwards, while solidity generally increases to the right. The bankruptcies are concentrated in the upper left-hand corner of the map, where both profitability and solidity are low.

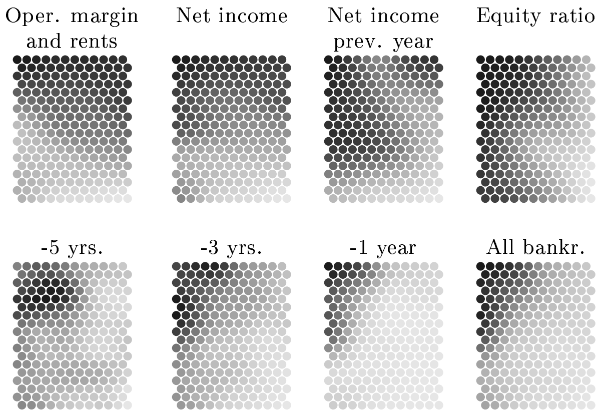

Figure 1: *The first-level SOM. Shown are the relative values of the financial indicators, proportion of enterprises with varying number of years to bankruptcy, and proportion of all enterprises that went bankrupt within five years after giving the financial statement.*

Examples of trajectories formed by an enterprise during several consecutive years are depicted in figure 2. Generally, the trajectories tend to move counterclockwise: a decrease in profitability, which shows as an upward movement, eventually results in a decrease in solidity, thus producing a leftward movement as well. Exceptions to this rule indicate abnormalities, such as sudden changes in the capital structure of the enterprise.

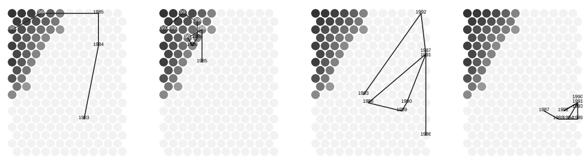

Figure 2: *Trajectories of four companies on the first-level SOM – the two companies on the left went eventually bankrupt. The year the enterprise was mapped to each trajectory point is plotted next to the trajectory; the area with a high bankruptcy risk is marked with a (thresholded) darker shade.*

It seems plausible that this methodology could be extended also to quantitative analysis. The first step could be the classical "bankruptcy prediction problem", in which one tries to answer the question whether or not some company will go bankrupt, given its financial statements. This has become something of a benchmark problem and has been widely studied, so it could be used to get an idea of the performance level of the methodology introduced here.

4. INDEPENDENT COMPONENT ANALYSIS

Recently, the method of Independent Component Analysis (ICA) has attracted considerable interest in the signal processing and neural network communities [1, 4, 5, 7, 12, 13, 19]. Consider a set of parallel signals or time series $x_i(t)$, with i indexing the individual time series and t denoting discrete time. In the basic ICA, a generative model is assumed, by which the original signals $x_i(t)$ are instantaneous linear mixtures of independent source signals $s_j(t)$, with some unknown mixture coefficients $a_{i,j}$:

$$x_i(t) = \sum_j a_{i,j} s_j(t) \tag{3}$$

for each signal $x_i(t)$. Such a model may not be unrealistic in some sensor array applications in which a number of independent signals s_i arrive at a number of sensors but are weighted and superimposed due to the different locations of the sensors. Also, in the case of financial time series, there may be some underlying factors like seasonal variations or economic events that affect a number of simultaneous time series but can be assumed to be quite independent.

If we go to vector-matrix formulation, defining a vector-valued time series $\mathbf{x}(t)$ with elements $x_i(t)$, a vector-valued source signal $\mathbf{s}(t)$ with elements $s_j(t)$, and a matrix $\mathbf{A} = (a_{i,j})$, then we can write

$$\mathbf{x}(t) = \mathbf{A}\mathbf{s}(t). \tag{4}$$

Matrix $\mathbf{A}$ is called the mixing matrix. The problem is to invert this model and find out the vector $\mathbf{s}(t)$, given sequential observations on $\mathbf{x}(t)$, but without knowing the mixing matrix $\mathbf{A}$.

The fundamental restriction of the ICA model is that we can only estimate non-Gaussian independent components (except if just one of the independent components is Gaussian). Moreover, neither the energies nor the signs of the independent components $s_i(t)$ can be estimated, because any constant multiplying an independent component in eq. (4) could be cancelled by dividing the corresponding column of the mixing matrix $\mathbf{A}$ by the same constant. For mathematical convenience, we define here that the amplitudes of the independent components $s_i(t)$ have unit variance. This makes the (non-Gaussian) independent components unique, up to their signs. Note that no order is defined between the independent components.

In the basic approach to solving the ICA problem, the solution is sought in the form

$$\hat{\mathbf{s}} = \mathbf{y} = \mathbf{W}\mathbf{x}. \tag{5}$$

The goal is now to find a matrix $\mathbf{W}$ that makes the elements of $\mathbf{y}$ statistically independent. We call such a matrix a separating matrix. A recent review of various information theoretic contrast functions for solving $\mathbf{W}$, like mutual information, negentropy, maximum entropy, and infomax, as well as the maximum likelihood approach, is given by Cardoso [5], who also discusses the numerical problems in minimizing / maximizing such contrast functions.

The problem of solving the separating matrix $\mathbf{W}$ is somewhat simplified if we consider only one of the source signals at a time. From eq. (5) it follows

$$\hat{s}_i = y_i = \mathbf{w}_i^T \mathbf{x} \tag{6}$$

with $\mathbf{w}_i$ the i-th row of $\mathbf{W}$. We have earlier suggested and analyzed neural type one-unit learning rules [11] that give as solutions one row $\mathbf{w}_i$ of the separating matrix. A condition of local convergence to a correct solution was given. The condition is very robust and shows that a wide range of nonlinear functions in our learning rules are possible.

The problem is further simplified by performing a preliminary sphering or prewhitening of the data $\mathbf{x}$: the observed vector $\mathbf{x}$ is first linearly transformed to another vector whose elements are mutually uncorrelated and all have unit variances. This transformation is always possible and can be accomplished by classical Principal Component Analysis. At the same time, the dimensionality of the data should be reduced so that the dimension of the transformed data vector equals n, the number of independent components. This also has the effect of reducing noise. It can be shown that after this preprocessing, $\mathbf{W}$ will be an orthogonal matrix.

As an example of contrast functions, consider the simple case of maximizing the kurtosis $E\{y_i^4\} - 3E\{y_i^2\}$ of the estimated signals y_i; but because we assumed that the estimated signals have unit variance, this reduces to maximizing the fourth moment $E\{y_i^4\}$. Its gradient with respect to $\mathbf{w}_i$ (see Eq. (6)) is $4E\{(\mathbf{w}_i^T\mathbf{x})^3\mathbf{x}\}$. In a gradient type learning rule, the row $\mathbf{w}_i$ of the separating matrix $\mathbf{W}$ would be sought using an instantaneous version of this gradient, in which the expectation is dropped and the gradient is computed separately for each input vector $\mathbf{x}$. In addition, a normalization term would be needed that keeps the norm of $\mathbf{w}_i$ equal to one – remember that our $\mathbf{W}$ matrix must be orthogonal due to the prewhitening of the data $\mathbf{x}$.

Another, much more efficient algorithm is the following fixed point iteration [10]:

1. Take a random initial vector $\mathbf{w}(0)$ of norm 1. Let $k = 1$.
2. Let $\mathbf{w}(k) = E\{\mathbf{x}(\mathbf{w}(k-1)^T\mathbf{x})^3\} - 3\mathbf{w}(k-1)$. The expectation can be estimated using a large sample of $\mathbf{x}$ vectors (say, 1,000 points).
3. Divide $\mathbf{w}(k)$ by its norm.
4. If $|\mathbf{w}(k)^T\mathbf{w}(k-1)|$ is not close enough to 1, let $k = k+1$ and go back to step 2. Otherwise, output the vector $\mathbf{w}(k)$.

The final vector $\mathbf{w}(k)$ given by the algorithm equals one of the rows of the (orthogonal) separating matrix $\mathbf{W}$.

To estimate n independent components, we run this algorithm n times. To ensure that we estimate each time a different independent component, we only need to add a simple orthogonalizing projection inside the loop. Recall that the rows of the separating matrix $\mathbf{W}$ are orthonormal because of the sphering. Thus we can estimate the independent components one by one by projecting the current solution $\mathbf{w}(k)$ on the space orthogonal to the rows of the separating matrix $\mathbf{W}$ previously found. Also a symmetrical orthogonalization is possible.

This algorithm, with the preliminary whitening and several extensions, is implemented in the FastICA package available through the WWW [8]. A remarkable property of the FastICA algorithm is that a very small number of iterations, usually 5-10, seems to be enough to obtain the maximal accuracy allowed by the sample data. This is due to the cubic convergence shown in [10].

5. APPLICATION: FINDING HIDDEN FACTORS IN FINANCIAL DATA

It is a tempting alternative to try ICA on financial data. There are many situations in that application domain in which parallel time series are available, such as currency exchange rates or daily returns of stocks, that may have some common underlying factors. ICA might reveal some driving mechanisms that otherwise remain hidden. In a recent study of a stock portfolio [3], it was found that ICA is a complementary tool to PCA, allowing the underlying structure of the data to be more readily observed.

In [16], we applied ICA on a different problem: the cashflow of several stores belonging to the same retail

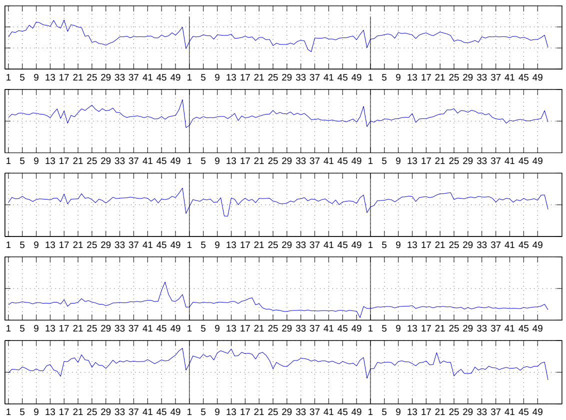

Figure 3: (from [16]). *Five samples of the original cashflow time series. Horizontal axis: time in weeks.*

chain, trying to find the fundamental factors common to all stores that affect the cashflow data. Thus, the cashflow effect of the factors specific to any particular store, i.e., the effect of the actions taken at the individual stores and in its local environment could be analyzed.

The assumption of having some underlying independent components in this specific application may not be unrealistic. For example, factors like seasonal variations due to holidays and annual variations, and factors having a sudden effect on the purchasing power of the customers like prize changes of various commodities, can be expected to have an effect on all the retail stores, and such factors can be assumed to be roughly independent of each other. Yet, depending on the policy and skills of the individual manager like e.g. advertising efforts, the effect of the factors on the cash flow of specific retail outlets are slightly different. By ICA, it is possible to isolate both the underlying factors and the effect weights, thus also making it possible to group the stores on the basis of their managerial policies using only the cash flow time series data.

The data consisted of the weekly cash flow in 40 stores that belong to the same retail chain; the cash flow measurements cover 3 years or 156 weeks. Some examples of the original data $x_i(t)$ are shown in Fig. 3.

The prewhitening was performed so that the original signal vectors were projected to the subspace spanned by their first four principal components and the variances were normalized to 1. Thus the dimension of the signal space was decreased from 40 to 5. Using the FastICA algorithm, four IC's $s_i(t),\ i = 1, ..., 5$ were estimated. As depicted in Fig. 4, the FastICA algorithm has found several clearly different fundamental factors hidden in the original data.

The factors have clearly different interpretations. The upmost two factors follow the sudden changes that are caused by holidays etc.; the most prominent example is the Christmas time. The factor on the bottom row, on the other hand, reflects the slower seasonal variation, with the effect of the summer holidays clearly visible. The factor on the third row could represent a still slower variation, something resembling a trend. The last factor, on the second row, is different from the others; it might be that this factor follows mostly the relative competitive position of the retail chain with respect to its competitors, but other interpretations are also possible.

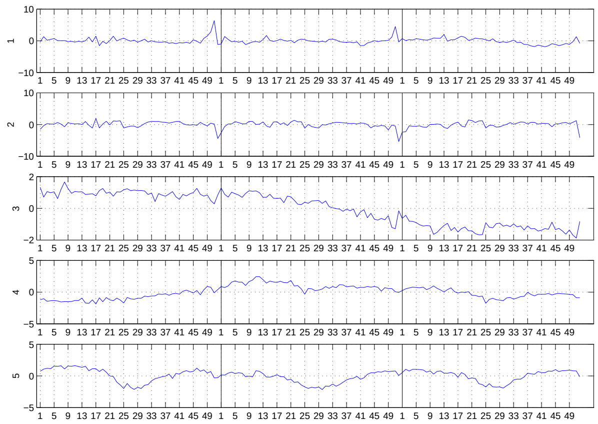

Figure 4: (from [16]). *Five independent components or fundamental factors found from the cashflow data.*

More details on the experiments and their interpretation can be found in [16].

6. REFERENCES

[1] S. Amari, A. Cichocki, and H.H. Yang. A new learning algorithm for blind source separation. In *Advances in Neural Information Processing 8*, Cambridge, MA: MIT Press, 757 - 763, 1996.

[2] B. Back, G. Oosterom, K. Sere, and M. van Wezel. A comparative study of neural networks in bankruptcy prediction. In *Multiple Paradigms for Artificial Intelligence* (C. Carlsson, T. Jarvi, and T. Reponen, eds.), Helsinki, Finland: Finnish Artificial Intelligence Society, 140 - 148, 1994.

[3] A. Back and A. Weigend. First application of Independent Component Analysis to extracting structure from stock returns. *Int. J. Neural Systems 8*, 473-484, 1997.

[4] A. J. Bell and T. J. Sejnowski. An information-maximization approach to blind separation and blind deconvolution. *Neural Computation 7*, 1129–1159, 1995.

[5] J.-F. Cardoso. Blind signal separation: statistical principles. *Proc. IEEE 86*, 2009 - 2025, 1998.

[6] V. Cherkassky and F. Mulier, *Learning from Data - Concepts, Theory, and Methods*. New York: Wiley, 1998.

[7] A. Cichocki and R. Unbehauen. Robust neural networks with on-line learning for blind identification and blind separation of sources. *IEEE Trans. CS I 43*, 894 - 906, 1996.

[8] The FastICA public domain package available at `http://www.cis.hut.fi/projects/ica/fastica/`.

[9] S. Haykin, *Neural Networks - a Comprehensive Foundation*. New York: MacMillan College Publ. Co., 1994.

[10] A. Hyvärinen and E. Oja. A fast fixed-point algorithm for Independent Component Analysis. *Neural Computation 9*, 1483 - 1492, 1997.

[11] A. Hyvärinen and E. Oja, E. Independent Component Analysis by general non-linear Hebbian-like learning rules. *Signal Processing 64*, 301 - 313, 1998.

[12] C. Jutten and J. Herault. Independent component analysis (INCA) versus independent component analysis. In *Signal Processing IV: Theories and Applications* (J. Lacoume et al, eds.), Amsterdam: Elsevier, 643 - 646, 1988.

[13] J. Karhunen, E. Oja, L. Wang, R. Vigario, and J. Joutsensalo. A class of neural networks for Independent Component Analysis. *IEEE Trans. on Neural Networks 8*, 486 - 504, 1997.

[14] K. Kiviluoto and P. Bergius. Analyzing financial statements with the self-organizing map. In *Proc. of the workshop on self-organizing maps (WSOM'97)*, Espoo, Finland, June 1997, 362–367.

[15] K. Kiviluoto and P. Bergius. Exploring corporate bankruptcy with two-level self-organizing maps. In *Proc. of the Fifth Int. Conf. on Computational Finance*, Boston, MA, Dec. 1998, 373–380.

[16] K. Kiviluoto and E. Oja. Independent component analysis for parallel financial time series. In *Proc. ICONIP'98*, Kitakyushu, Japan, Oct. 1998, 895–898.

[17] T. Kohonen. Self-organized formation of topologically correct feature maps. *Biol. Cyb. 43*, 59 - 69, 1982.

[18] T. Kohonen, *Self-Organizing Maps.* Berlin: Springer, 1995.

[19] E. Oja. The nonlinear PCA learning rule in independent component analysis. *Neurocomputing 17*, 25 - 45, 1997.

[20] H. Ritter, T. Martinetz, and K. Schulten, *Neural Computation and Self-Organizing Maps: an Introduction*. Reading: Addison-Wesley, 1992.

[21] B. Martín-del-Brío and C. Serrano-Cinca. Self-organizing neural networks for the analysis and representation of data: some financial cases. *Neural Computing & Applications 1*, 193–206, 1993.

[22] S. Shumsky and A. Yarovoy. Neural network analysis of Russian banks. In *Proc. of the workshop on self-organizing maps (WSOM'97)*, Espoo, Finland, June 1997, 351 - 355.