PRINCIPAL COMPONENT ANALYSIS WITH NEURAL NETWORK AND DNA SEQUENCE ANALYSIS

Marcus Rogério de OLIVEIRA*

ABSTRACT

The Artificial Neural Networks are versatile tools for a number of applications. With their many configurations of parameters and architectures, Artificial Neural Networks are used in Bioinformatics and other areas. This work presents an Artificial Neural Network that implements Principal Component Analysis (PCA) and performs DNA sequence analyses.

KEYWORDS: Artificial Neural Networks. Principal Component Analysis. Hebbian Neural Network Neuron.

INTRODUCTION

Nucleic acid and protein sequences contain information of interest to molecular biologists since the genome forms the blueprint of an organism (Cathy, 2000). As the molecular data continues to grow exponentially, computacional tools and techniques are needed to identify the features, functions and structure of genes.

Artificial Neural Networks are versatile tools for a number of applications. With their many features and capabilities for recognition, generalizations and classification, Artificial Neural Networks are used in Bioinformatics and other areas. Recently, Neural Networks have been used for statistical techniques that can be used in Genome studies (Oja, 1991). Principal Component Analysis (PCA) is one of them. In the ûeld of data analysis, it is important to reduce the dimensionality of data, because it will extract knowledge from the data. As a method of dimensionality reduction, PCA has been applied in various areas, such as data compression and pattern recognition (Hotelling, 1993).

The neuron model for PCA consists of a neuron derived from the original formulation of the Hebbian Neural Network Neuron. This model leads to a behaviour where the unit is able to extract from its inputs the statistically most significant factor. In fact, in PCA, n dimensional redundant data vectors, which are correlated to each other, are transformed into certain s dimensional data vectors ($s < n$) *orthogonal* to each other. These vectors give the principal directions along which the data can cloud mostly stretched. The principal components are the projection of the data set on eigenvectors of principal directions. The first principal components ranked with their eigenvalues in descending order explain the most variance of the data set and the last explains the least.

Principal Component Neural Networks (PCNN) are mainly used for classification and feature extraction. Because the PCA makes classification by extracting the most important information for classification and removing the correlation between the attributes, it will be used for Nucleid Acid Sequence comparison and analysis.

* Faculdade de Tecnologia de Taquaritinga – FATEC/TQ - Av. Dr. Flávio Henrique Lemos, 585 – Portal Itamaracá. 15900-000 -Taquaritinga – SP – Brasil. marcusrogerio@terra.com.br

Nucleic Acid Sequence Analysis

Proteins

A protein is typically built of a series of basic blocks called amino acids, chained together in a linear sequence of blocks. Amino acids may come in a variety of shapes and properties: they may be small or bulky, hidrophobic or hidrophyllic, electrically charged or neutral, etc. hence allowing for very complex shapes and interactions to be produced.

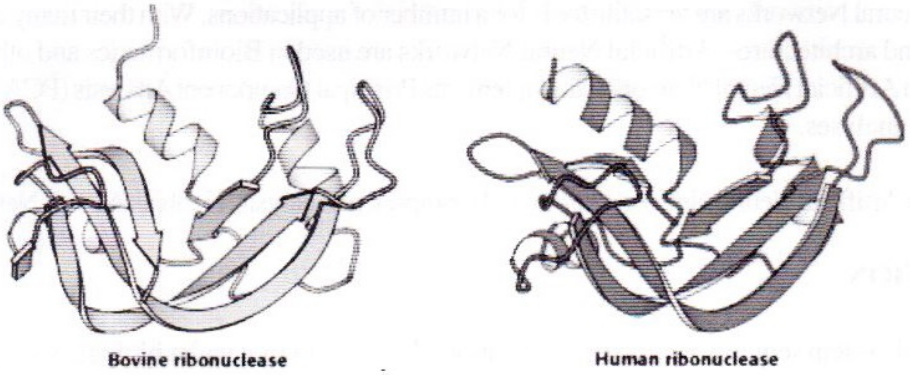

Figure 1. Proteins Structures

Amino acids are commonly referred to by name or by an abbreviation, usually in three or one letter. This allows for more efficient descriptions of how they are chained together to build a protein:

Table 1. Amino Acids

Neutral-Nonpolar	3-letter	1-letter
Glycine	Gly	G
L-Alanine	Ala	A
L-Valine	Val	V
L-Isoleucine	Ile	I
L-Leucine	Leu	L
L-Phenylalanine	Phe	F
L-Proline	Pro	P
L-Methionine	Met	M
Neutral-Polar		
L-Serine	Ser	S
L-Threonine	Thr	T
L-Tyrosine	Tyr	Y
L-Tryptophan	Trp	W
L-Asparagine	Asn	N
L-Glutamine	Gln	Q
L-Cysteine	Cys	C
Acidic		
L-Aspartic	Asp	D
L-Glutamic	Glu	E
Basic		
L-Lysine	Lys	K
L-Arginine	Arg	R
L-Histidine	His	H

Nucleic Acids

For Nucleic Acids, the number of basic building blocks is a lot smaller, each nucleic acid chain being composed of series of only four possible different nucleotides which furthermore provide for a very limited set of interactions.

Nucleic acids come in two types: DNA (DeoxyriboNucleic Acid) and RNA (RiboNucleic Acid). Both of them consist of a series of nucleotides that are glued one after the other to constitute the sequence of blocks that make up the functional chain.

Nucleotides are composed of a phosphate group, a sugar (ribose in RNA, and deoxyribose in DNA) and a base which marks the specific difference among nucleotides. The base may be one of guanine, cytosine, adenine and thymine in the case of DNA or guanine, cytosine, adenine or uracil for RNA. They can be referred to by their one letter abbreviations G, C, A, T and U. Interactions are mainly driven by the stablishment of hydrogen bonds, which can only be established among thymine (or uracil) and adenine (two hydrogen bonds) and cytosine and guanine (three hydrogen bonds).

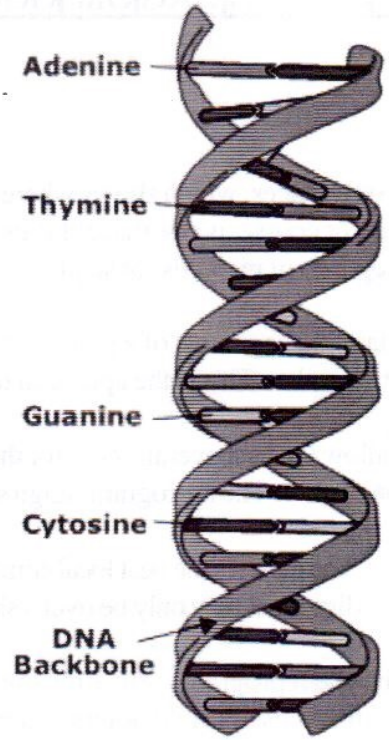

Figure 2. Helix Structure of DNA

The main role of nucleic acids is to convey all the genetic information needed to make proteins and control the building process. Protein sequences are coded by nucleic acids using groups three of nucleotides that code for a given amino acid: the code is more or less universal with little exceptions, and includes redundancy to increase the fidelity of the reading process when making duplicates or translating the information:

Table 2. A combination of 3 Nucleic Acids corresponds to 1 Amino Acid

UUU	Phe	UCU	Ser	UAU	Tyr	UGU	Cys
UUC	Phe	UCC	Ser	UAC	Tyr	UGC	Cys
UUA	Leu	UCA	Ser	UAA	Stop	UGA	Stop
UUG	Leu	UCG	Ser	UAG	Stop	UGG	Trp
CUU	Leu	CCU	Pro	CAU	His	CGU	Arg
CUC	Leu	CCC	Pro	CAC	His	CGC	Arg
CUA	Leu	CCA	Pro	CAA	Gln	CGA	Arg
CUG	Leu	CCG	Pro	CAG	Gln	CGG	Arg
AUU	Ile	ACU	Thr	AAU	Asn	AGU	Ser
AUC	Ile	ACC	Thr	AAC	Asn	AGC	Ser
AUA	Ile	ACA	Thr	AAA	Lys	AGA	Arg
AUG	Met	ACG	Thr	AAG	Lys	AGG	Arg
GUU	Val	GCU	Ala	GAU	Asp	GGU	Gly
GUC	Val	GCC	Ala	GAC	Asp	GGC	Gly
GUA	Val	GCA	Ala	GAA	Glu	GGA	Gly
GUG	Val*	GCG	Ala	GAG	Glu	GGG	Gly

Sequence Comparison

An alignment is an arrangement of two sequences, which shows where the two sequences are similar, and where they differ. An optimal alignment, of course, is one that exhibits the most similarities, and the least differences. Broadly, there are three categories of methods for sequence comparison:

• **Segment methods** compare all overlapping segments of a predetermined length (e.g., 10 amino acids) from one sequence to all segments from the other. This is the approach used in dotplots.

• **Optimal global alignment** methods allow the best overall score for the comparison of the two sequences to be obtained, including a consideration of gaps. These programs align sequences over their whole length.

• **Optimal local alignment** algorithms seek to identify the best local similarities between two sequences also including explicit consideration of gaps. Alignment may only be over a short span of sequence.

The basic idea behind the sequence alignment programs is to align the two sequences in such a way as to produce the highest score - a scoring matrix is used to add points to the score for each match and subtract them for each mismatch. The matrices commonly used for scoring protein alignments are more complex than the simple match/mismatch matrices used for DNA sequences such as the one we saw earlier; the scores that form the protein matrices are designed to reflect similarity between the different amino acids rather than simply scoring identities. Over time various mutations occur in sequences; the scoring matrices attempt to cope with mutations, but insertions and deletions require some extra parameters to allow the introduction of gaps in the alignment. There are penalties both for the creation of gaps and for the extension of existing ones; the default gap parameters given in alignment programs have been found to be empirically correct with test sequences but you should experiment with different gap penalties (Berg, 2002).

Principal Component Analysis (PCA)

PCA is a way of identifying patterns in data, and expressing the data in such a way as to highlight their similarities and differences. Since patterns in data can be hard to find in data of high dimension, where the facility of graphical representation is not available, PCA is a powerful tool for analysing data.

Principal component analysis (PCA) involves a mathematical procedure that transforms a number of (possibly) correlated variables into a (smaller) number of uncorrelated variables called *principal components* (Dunteman, 1989). The first principal component accounts for as much of the variability in the data as possible, and each succeeding component accounts for as much of the remaining variability as possible. Traditionally, principal component analysis is performed on a square symmetric matrix of type SSCP (pure sums of squares and cross products), Covariance (scaled sums of squares and cross products), or Correlation (sums of squares and cross products from standardized data). The analysis results for objects of type SSCP and Covariance do not differ, since these objects only differ in a global scaling factor. A Correlation object has to be used if the variances of individual variates differ much, or if the units of measurement of the individual variates differ.

The mathematical technique used in PCA is called eigen analysis: eigenvalues and eigenvectors of a square symmetric matrix with sums of squares and cross products. The eigenvector associated with the largest eigenvalue has the same direction as the first principal component. The eigenvector associated with the second largest eigenvalue determines the direction of the second principal component. The sum of the eigenvalues equals the trace of the square matrix and the maximum number of eigenvectors equals the number of rows (or columns) of this matrix.

Mathematical foundations

Principal component analysis is based on the statistical representation of a random variable. Suppose a random vector population **x**, where

$$\mathbf{x} = (x_1, \ldots, x_n)^T$$

and the mean of that population is denoted by

$$\mu_{\mathbf{x}} = E\{\mathbf{x}\}$$

and the covariance matrix of the same data set is

$$\mathbf{C}_x = E\{(\mathbf{x} - \mu_{\mathbf{x}})(\mathbf{x} - \mu_{\mathbf{x}})^T\}$$

The components of $\mathbf{C}_x$, denoted by c_{ij}, represent the covariances between the random variable components x_i and x_j. The component c_{ii} is the variance of the component x_i. The variance of a component indicates the spread of the component values around its mean value. If two components x_i and x_j of the data are uncorrelated, their covariance is zero $(c_{ij} = c_{ji} = 0)$. The covariance matrix is, by definition, always symmetric.

From a sample of vectors $\mathbf{x}_1, \ldots, \mathbf{x}_M$, PCA can calculate the sample mean and the sample covariance matrix as the estimates of the mean and the covariance matrix.

From a symmetric matrix such as the covariance matrix, PCA can calculate an orthogonal basis by finding its eigenvalues and eigenvectors. The eigenvectors $\mathbf{e}_i$ and the corresponding eigenvalues λ_i are the solutions of the equation

$$\mathbf{C}_x\mathbf{e}_i = \lambda_i\mathbf{e}_i, i = 1, ..., n$$

For simplicity it is assumed that the λ_i are distinct. These values can be found, for example, by finding the solutions of the characteristic equation

$$|\mathbf{C}_x - \lambda\mathbf{I}| = 0$$

where the $\mathbf{I}$ is the identity matrix having the same order than $\mathbf{C}_x$ and the the $|.|$ denotes the determinant of the matrix. If the data vector has *n* components, the characteristic equation becomes of order *n*. This is easy to solve only if *n* is small. Solving eigenvalues and corresponding eigenvectors is a non-trivial task, and many methods exist. One way to solve the eigenvalue problem is to use a neural solution to the problem (Oja, 1983). The data is fed as the input, and the network converges to the wanted solution. By ordering the eigenvectors in the order of descending eigenvalues (largest first), one can create an ordered orthogonal basis with the first eigenvector having the direction of largest variance of the data. In this way, we can find directions in which the data set has the most significant amounts of energy. Suppose one has a data set of which the sample mean and the covariance matrix have been calculated. Let $\mathbf{A}$ be a matrix consisting of eigenvectors of the covariance matrix as the row vectors.

By transforming a data vector $\mathbf{x}$, we get

$$\mathbf{y} = \mathbf{A}(\mathbf{x} - \mu_{\mathbf{x}})$$

which is a point in the orthogonal coordinate system defined by the eigenvectors. Components of $\mathbf{y}$ can be seen as the coordinates in the orthogonal base. It can reconstruct the original data vector $\mathbf{x}$ from $\mathbf{y}$ by

$$\mathbf{x} = \mathbf{A}^T\mathbf{y} + \mu_{\mathbf{x}}$$

using the property of an orthogonal matrix $\mathbf{A}^{-1} = \mathbf{A}^T$. The $\mathbf{A}^T$ is the transpose of a matrix $\mathbf{A}$. The original vector $\mathbf{X}$ was projected on the coordinate axes defined by the orthogonal basis. The original vector was then reconstructed by a linear combination of the orthogonal basis vectors. Instead of using all the eigenvectors of the covariance matrix, we may represent the data in terms of only a few basis vectors of the orthogonal basis. If it is denoted by the matrix having the K first eigenvectors as rows by $\mathbf{A}_K$, a similar transformation can be created as seen above

$$\mathbf{y} = \mathbf{A}_K(\mathbf{x} - \mu_{\mathbf{x}})$$

and

$$\mathbf{x} = \mathbf{A}_K^T\mathbf{y} + \mu_{\mathbf{x}}$$

This means that the original data vector can be projected on the coordinate axes having the dimension K and transforming the vector back by a linear combination of the basis vectors. This minimizes the mean-square error between the data and this representation with given number of eigenvectors. If the data is concentrated in a linear subspace, this provides a way to compress data without losing much information and simplifying the representation. By picking the eigenvectors having the largest eigenvalues we lose as little information as possible in the mean-square sense. One can e.g. choose a fixed number of eigenvectors and their respective eigenvalues and get a consistent representation, or abstraction of the data. This preserves a varying amount of energy of the original data. Alternatively, we can choose approximately the same amount of energy and a varying amount of eigenvectors and their respective eigenvalues. This would in turn give approximately consistent amount of information in the expense of varying representations with regard to the dimension of the subspace (Oja, 1983).

Neuron Model as a Principal Component Analyser

It is well known (Diamantras, 1986) that a 3 layer (including the input and the output layers) neural network model with linear transfer functions at the hidden layer has the rank reducing capability, where the specific rank is effected by the number of units at the hidden layer. This is a network version of reduced-rank (RR) regression (Anderson, 1993) which is also known as PCA of instrumental variables and redundancy analysis. The usual PCA follows when inputs and outputs coincide in RR regression analysis. The network version of PCA is not interesting in itself, because there are other more efficient and precise algorithms available (Takane, 1999). It becomes interesting when the model is extended to nonlinear PCA by including two additional hidden layers with nonlinear transfer functions, one between the input layer and the middle layer and the other between the middle layer and the output layer.

The neuron model considered here is as follow (Hotelling, 1993). The neuron receives a set of n scalar-valued inputs: $\xi_1, \ldots, \xi_n$ ch mey be assumed to represent firing frequencies in presynaptic fibers; in some models, the zero level is defined so that negative values for the effective inputs become possible) through n synaptic junctions with coupling strngths $\mu_1, \ldots, \mu_n$ The unit sends out an efferent signal η. According to many models of neuron networks, the input-output relationship is linearized to read

$$\eta = \sum_{i=1}^{n} \mu_i \xi_i .$$

In most recent models, the junction strengths μ_i have been assumed variable in time according to some version of the Hebbian hypothesis: the efficacies grow stronger when both the pre and postsynaptic signals are strong. However, as the basic Hebbian scheme would lead to unrealistic growth of the efficacies, a saturation or normalization has usually been assumed. This leads typically to the following type of learning equation:

$$\mu_i(t+1) = \frac{\mu_i(t) + \gamma\eta(t)\xi_i(t)}{\{\sum_{i=1}^{n} [\mu_i(t) + \gamma\eta(t)\xi_i(t)]^2\}^{1/2}}$$

where γ is a positive scalar.

The form of the denominator is due to the use of Euclidean vector norm. The sum of the squares of $\mu_i(t)$

remains equal to one. This particular form of normalization is very convenient from a mathematical point of view. The actual reason for a normalization would be the competition of the synapses of a given neuron over some limited resource factors that are essential in efficacy growth. It should be realized that the synaptic efficacies $\mu_i(t)$ need not be linearly related to euch resource factors and thus in the denominator of the equation above there is no reason to prefer e.g. the linear sum to some other form like the one suggested there. The equation above produces a very specific type of behaviour for the model neuron: if the input vectors $[\xi_1(t), \ldots, \xi_n(t)]^T$ for $t = 1, 2, \ldots$ are regarded as a vector-valued stochastic process, the the neural unit tends to become a principal component analyser for the input process.

Neural Network Design for PCA

Consider a network with n inputs, $h<n$ linear hidden units, and n linear output units. Given an input, the goal is to reproduce it at the output so the network acts as autoencoder, mapping an input pattern to itself.

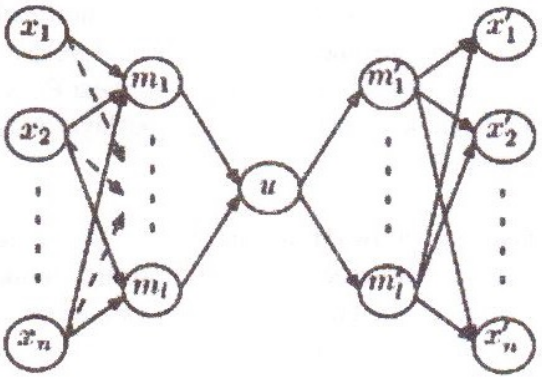

Figure 3. An autoencoder network maps an input vector to itself reducing dimensionality

The hidden layer acts as bottleneck tha forces the network to form a compressed representation for the data. Tha hidden layer activities are a linear functions of the inputs but the hidden çayer is smaller then the input dimension so some informations must necessarily be lost, in general. The best hidden layer representarion will be on that preserves as much information about the input as possible. Ideally, it will ignore nonessential noise and reproduce only the most significant features of the input pattern.

The optimal hidden unit weights are determined by a set of vectors spanning the singular value decomposition of the input data. That is, the ideal representation formed at the hidden layer span the same space as the h eigenvectors corresponding to the h largest eigenvalues of the covariance matriz of the training data. The network is linear so it cam be collapsed into a single linear transformation $x=Fx$. The rank of F is limited in the preceding by h, the dimension of the hidden layer.

Many papers have been written on the links between neural networks and principal components analysis and just as linear autoencoder implements a form of PCA, its has been shown that a single-hidden-layer nilar

network trained to perfoem classification with 1-of-N target representation implements a form of discriminate analysis .

Experiment

The nucleotides from a gene sequence where converted in real numbers between 0.2 and 0.7. From position 0, the real numbers representing the nucleotides were added by a factor of value 0.0002. This technic makes the position of the nucleotide in the sequence relevant.

Example of the data before the conversion:

atgccctcac aggaaacaga attcaaacac ttagacaatc aatataaaga tgaagtgaac gctcttaaag agaagttgga aaatttgcag gaacaaatca aagttcaaaa aaggatagaa gaacaagaaa aaccaagaaa atggtgggga ctatggcgaa aaatagatga ttcagttaaa aagaaagttc cagaattgcg atttaaagga tttacggatg attgggaaga gcgtaagttg ggagaattat ctaatattgt gggcggtgga acaccaagta catcgaactc tgaatactgg gacggtgata ttgactggta tgctccagct gaaattggag aacaaaggta tgttagtaaa agtaaaaaga ctattactga actaggtcta aagaagagct cagctagaat tttaccagta ggaactgtct tatttacttc tcgtgctggt atcggaaaca ccgctatatt aggtaaagaa gctacaacta accaagggtt tcaatcaatt gttcctaatc caaataaact tgatagttat tttatttatt caagaactaa tgaacttaag cggtatggtg aagtcaccgg tgcaggatct acttttgttg

Example of the data after the conversion:

0.02
0.025005
0.035014
0.030017999999999996
0.030023999999999995
0.030029999999999994
0.025029999999999997
0.030041999999999996
0.020031999999999998
0.030053999999999994
0.020043999999999996
0.035084
0.035090999999999999
0.020055999999999994

After the converstion, all the nucleotides are represented by real numbers in with the position is garanteed. So, the sequence can be submitted to the Neural Network.

The results can be observed by the follow program output:

Input file name: data1.dat
No. of rows, n = 1239
No. of cols, m = 2
Input data sample follows as a check, first 4 values.
value = 0.02
value = 0.02
value = 0.020004
value = 0.025005

Variable means:
0,0297 0,0297

Variable standard deviations:
0,0068 0,0068

SSCP or sums-of-squares and cross-products matrix:
(Note: correlations in this implementation)

1.00 0.10
0.10 1.00

Eigenvectors (leftmost col <—> largest eval):

0.7071 -0.7071
0.7071 0.7071

Eigenvalues and as cumulative percentages:
1,0991 0,9009

54,9565 100,0000

Row projections in new principal component space: (parcial)

-0.0574 0.0001
-0.0427 0.0148
0.0312 -0.0002
0.0017 -0.0001
-0.0131 0.0147
-0.0131 0.0147
-0.0426 0.0148
-0.0130 0.0147

Column projections in new principal. compomponent. space.
0,7413 -0,6711
0,7413 0,6711

The output above shows the result of the comparison of 2 genes of the same family organism. The column projections demonstrate the high degree of similarity. The eigevalues, the standard deviation and the mean are identical. This demonstrates that the Neural Network PCA can identify similarities between gene sequences.

After 10 experiments with the same family of genes, all of then converged to the same result: all the genes belong to the same organism family. With 10 experiments using genes sequence of different families of organisms, the NNPCA converged showing demonstrating that the genes are different. So, 100% of the experiment brings optimal results.

RESUMO

As Redes Neurais artificiais podem ser utilizadas para um grande número de aplicações das mais diferentes áreas. Variações de parâmetros e arquiteturas fazem com que uma rede neural responda de maneira diferente e específica. Por esse motivo, são utilizadas como solução em problemas diversos. Este trabalho apresenta um tipo especial de Rede Neural que implementa o método de Análise de Componentes Principais e efetua análises de seqüências DNA.

PALAVRAS-CHAVE: Redes Neurais Artificiais. Análise de Componentes Principais. Modelo de Neurônio de Hebbian.

BIBLIOGRAPHY

ANDERSON. T. W. *Neural Networks* Psychometrika, 1993.

BERG, G. et al. *Biochemistry: Fifht Edition*. W. H. Freeman and Company, 2002.

CATHY, W.H. et al. *Neural Networks and Genoma Informatics*. Methods in Computational Biology and Biochemistry. Volume 1. Ed. Elsevier, 2000.

DIAMANTRAS, K. I.. *Principal Component Neural Networks: Theory and Applications.* Wiley, Hardcover, Published February 1996.

DUNTEMAN, G. *Principal Components Analysis (Quantitative Applications in the Social Sciences).* SAGE Publications. May 1, 1989.

GRIFFITHS, A., et al. *An Introduction to Genetic Analysis.* Ed. W. H. Freeman and Company. 2000, hardcover.

HOTELLING, H. *Analysis of complex statistical variables into principal components.* Journal of Educational Psychology, 24,1993.

OJA, E. *A simplified Neuron Model as a Principal Component Analyser.* University of Kuopio. Institute of Mathematics, 1982.

OJA, E. *Subspace methods of pattern recognition.* Volume 6 of *Pattern recognition and image processing series*. John Wiley & Sons, 1983.

OJA, E. *Principal Components, Minor Components and Linear Neural Networks*. Lappeenranta University of Technology. March 1991.

TAKANE, Y. *Nonlinear PCA by Neural Network Models*. McGill University, 1999.

WEBB, A. A. R. et al. *The optimized internal representation of multilayer network performs nomlinear discriminant analysis*. Neural Network 3(4), 1991.