Probabilistic Principal Component Analysis Based on JoyStick Probability Selector

Marko V. Jankovic, *Senior Member, IEEE,* and Masashi Sugiyama

***Abstract*— Principal component analysis (PCA) is a commonly applied technique for data analysis and processing, e.g. compression or clustering. In this paper we propose a probabilistic PCA model based on the Born rule. In off-line realization it can be seen as a successive optimization problem. In the on-line realization it will be solved by introduction of two different time scales. It will be shown that recently proposed time oriented hierarchical method, used for realization of biologically plausible PCA neural networks, represents a special case of the proposed model. The proposed model gives a general framework for creating different PCA realizations/algorithms. A particular realization can optimize locality of calculation, convergence speed, preciseness or some other parameter of interest. We will present some experimental results to illustrate effectiveness of the proposed model.**

I. Introduction

Principal component analysis (PCA) (e.g. [18]) is a well established technique for dimensionality reduction, and chapters on the subject may be found in numerous texts on multivariate data analysis. Examples of its many applications include data compression, image processing, visualization, exploratory data analysis, pattern recognition, and time series prediction.

The most common derivation of PCA is in terms of a standardized linear projection which maximizes the variance in the projected space [11]. For a set of observed N-dimensional data vectors $\{\boldsymbol{x}_m\}$, $m \in \{1, \ldots, M\}$ the K principal axes $\boldsymbol{w}_k$, $k \in \{1,\ldots, K\}$, are those orthonormal axes onto which retained variance under projection is maximal. It can be shown that the vectors $\boldsymbol{w}_k$ are defined by the K dominant eigenvectors $\boldsymbol{u}_k$ (i.e. those with the largest associated eigenvalues λ_k) of the sample covariance matrix $\boldsymbol{C}$, such that $\boldsymbol{C}\boldsymbol{u}_k=\lambda_k\boldsymbol{u}_k$. The K principal components of the observed vector $\boldsymbol{x}_k$ are given by the vector $\boldsymbol{y}_k=\boldsymbol{U}^{\mathrm{T}}(\boldsymbol{x}_k-\boldsymbol{x}_{\mathrm{mean}})$, where $\boldsymbol{U}=(\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_K)$ and $\boldsymbol{x}_{\mathrm{mean}}$ is the data sample mean. Matrix $\boldsymbol{W}$ can be expressed as $\boldsymbol{W}=\boldsymbol{U}\boldsymbol{R}$, where $\boldsymbol{R}$ is a rotation matrix of appropriate dimension. If the variables $\boldsymbol{y}_k$ are uncorrelated such that the output covariance matrix is diagonal with elements λ_k, then matrix $\boldsymbol{W}$ can be expressed as $\boldsymbol{W}=\boldsymbol{U}\boldsymbol{P}$, where $\boldsymbol{P}$ is a permutation matrix.

M. V. Jankovic is with the Department of Computer Science, Tokyo Institute of Technology, Ookayama 2-12-1, Meguro-ku, Tokyo, 152-8552, Japan, on leave from the EE Institute "Nikola Tesla", Belgrade, Serbia (phone: +81-3-7734-2699; fax: +81-3-7734-2699; e-mail: marko@ sg.cs.titech.ac.jp).

M. Sugiyama, is with the Department of Computer Science, Tokyo Institute of Technology, Ookayama 2-12-1, Meguro-ku, Tokyo, 152-8552, Japan, (e-mail: sugi@cs.titech.ac.jp).

Artificial neurons and neural networks have been shown to perform PCA when gradient ascent (descent) learning rules are used, which is related to the constrained maximization (minimization) of some objective functions [2, 6, 14, 22]. Due to their low complexity, such algorithms and their implementation in neural networks are potentially useful for tracking of slow changes of correlations in the input data or in updating eigenvectors with new samples.

In recent years, many Principal Subspace Analysis (PSA) algorithms have been proposed and studied in the literature [2, 5, 9, 12, 14, 22]. Some of these PSA algorithms were modified in order to derive parallel PCA algorithms. Usually, the modification was performed by introduction of some asymmetry (inhomogeneity) or nonlinearity in the original PSA algorithm that is not considered desirable from the point of view of the implementation of those algorithms in parallel hardware. Rare examples of a fully homogeneous algorithm are the bigradient algorithm proposed in [19], and the time-oriented hierarchical method proposed in [12, 15]. For a comprehensive review of known parallel, as well as sequential PCA algorithms see e.g. [6].

The Time-Oriented Hierarchical Method (TOHM) is a recently proposed cooperative competitive concept that transforms the PSA algorithms in to PCA algorithms (it can also be used to transform minor subspace algorithms (MSA) into minor component algorithms (MCA)). By implementation of the TOHM, we can create a fully homogeneous learning rule in terms of individual neurons if the initial PSA algorithm is fully homogeneous. The method uses two distinct time scales. A given PSA algorithm is responsible, on a faster time scale, for the behavior of all output neurons. At this scale, a principal subspace is obtained. On a slower time scale, output neurons compete to fulfill their "own interests". At this scale, basis vectors in the principal subspace are rotated toward the principal eigenvectors. To the best of our knowledge, the bigradient algorithm is the only other known algorithm that uses a similar concept. The difference between the TOHM and the bigradient algorithm has been investigated in [13, 15].

In this paper we will give a probabilistic view of the PCA problem and show that the TOHM is just one possible way to implement PCA on-line. The probabilistic view is based on a, new geometric interpretation of probability based on the Born rule [27]. We will explain that PCA can be seen as a successive optimization problem if we are implementing it off-line. In the case of on-line implementation, we will use constraint optimization on two time scales, for instance as done in the TOHM method. From the aspect of artificial neural networks, the choice of different realization concepts

has direct impact on the algorithm's convergence speed, preciseness, complexity of plausible hardware realization or biological plausibility. In Section II, the Born rule will be introduced. Geometric interpretation of the Born rule will be given in Section III. Section IV contains introduction of new probabilistic PCA. Some practical aspects of implementation will be discussed in Section V. Experimental results are presented in Section VI. Section VII gives conclusion remarks.

II. Quantum Probability Model and Born Rule

In quantum mechanics the transition from a deterministic description to a probabilistic one is done using a simple rule termed the Born rule. This rule states that the probability of an outcome ($\boldsymbol{a}$) given a state ($\boldsymbol{\Psi}$) is the square of their inner product $(\boldsymbol{a}^{\mathrm{T}}\boldsymbol{\Psi})^2$. This section is based on a similar section in [27].

In quantum mechanics the Born rule is usually taken as one of the axioms. However, this rule has well established foundations. Gleason's theorem [10] states that the Born rule is the only consistent probability distribution for a Hilbert space structure. Wooters [28] has shown that by using the Born rule as a probability rule, the natural Euclidean metrics on a Hilbert space coincides with a natural notion of a statistical distance. Short review for some other justifications of the Born rule can be seen in [27].

The quantum probability model takes place in a Hilbert space H of finite or infinite dimension. A state is represented by a positive semidefinite linear mapping (a matrix ρ) from this space to itself, with a trace of 1, i.e. $\forall \boldsymbol{\Psi} \in \mathrm{H}\ \ \boldsymbol{\Psi}^{\mathrm{T}}\boldsymbol{\rho}\boldsymbol{\Psi}\geq 0$, $\mathrm{Tr}(\boldsymbol{\rho})=1$. Such $\boldsymbol{\rho}$ is self adjoint and is called a density matrix.

Since $\boldsymbol{\rho}$ is self adjoint, its eigenvectors $\boldsymbol{\Phi}_i$ are orthonormal and since it is positive semidefinite its eigenvalues p_i are real and nonnegative $p_i \geq 0$. The trace of a matrix is equal to the sum of its eigenvalues, therefore $\sum_i p_i = 1$.

The equality $\boldsymbol{\rho} = \sum_i p_i \boldsymbol{\Phi}_i \boldsymbol{\Phi}_i^{\mathrm{T}}$ is interpreted as "the system is in state $\boldsymbol{\Phi}_i$ with probability p_i". The state $\boldsymbol{\rho}$ is called the pure state if $\exists i$ s.t. $p_i = 1$. In this case, $\boldsymbol{\rho}=\boldsymbol{\Psi}\boldsymbol{\Psi}^{\mathrm{T}}$ for some normalized state vector $\boldsymbol{\Psi}$, and the system is said to be in state $\boldsymbol{\Psi}$.

A measurement M with an outcome $\boldsymbol{x}$ in some set $\boldsymbol{X}$ is represented by a collection of positive definite matrices $\{\boldsymbol{m}_x\}_{x\in X}$ such that $\sum_{x\in X} \boldsymbol{m}_x = \mathbf{1}$ ($\mathbf{1}$ is being the identity matrix in H). Applying measurement M to state $\boldsymbol{\rho}$ produces outcome $\boldsymbol{x}$ with probability

$$p_x(\boldsymbol{\rho})=\mathrm{trace}(\boldsymbol{\rho}\boldsymbol{m}_x) \qquad (1)$$

This is the Born rule. Most quantum models deal with a more restrictive type of measurement called the von Neumann measurement, which involves a set of projection operators $\boldsymbol{m}_a=\boldsymbol{a}\boldsymbol{a}^{\mathrm{T}}$ for which $\boldsymbol{a}^{\mathrm{T}}\boldsymbol{a}'=\delta_{aa'}$. In a modern language, von Neumann's measurement is a conditional expectation onto a maximal Abelian subalgebra of the algebra of all bounded operators acting on the given Hilbert space. As before, $\sum_{\boldsymbol{a}\in \mathrm{M}} \boldsymbol{a}\boldsymbol{a}^{\mathrm{T}} = 1$. For this type of measurement the Born rule takes a simpler form: $p_a(\boldsymbol{\rho})=\boldsymbol{a}^{\mathrm{T}}\boldsymbol{\rho}\boldsymbol{a}$. Assuming $\boldsymbol{\rho}$ is a pure state this can be simplified further to

$$p_a(\boldsymbol{\rho}) = (\boldsymbol{a}^{\mathrm{T}}\boldsymbol{\Psi})^2. \qquad (2)$$

So, we can see that the probability of the outcome of the measurement will be $\boldsymbol{a}$, if the state is $\boldsymbol{\rho}$, is actually cosine square of the angle between vectors $\boldsymbol{a}$ and $\boldsymbol{\Psi}$, or $p_a(\boldsymbol{\rho})=\cos^2(\boldsymbol{a},\boldsymbol{\Psi})$.

III. JoyStick Probability Selector

In this section we will give one novel but simple interpretation of the probability that is related to the Born rule. Here we will assume that we are dealing with finite dimensional discrete variable.

For the moment, let's assume that we are dealing with discrete two dimensional variable. It can associate us with a coin tossing. Assume further that two possible outcomes of our experiment are represented by the dummy variables {01} and {10}. If we represent our coin as a unit norm vector in the two dimensional space (we will call that vector JoyStick Probability Selector or JSPS), then we can have the following simple geometric interpretation given in Fig. 1.

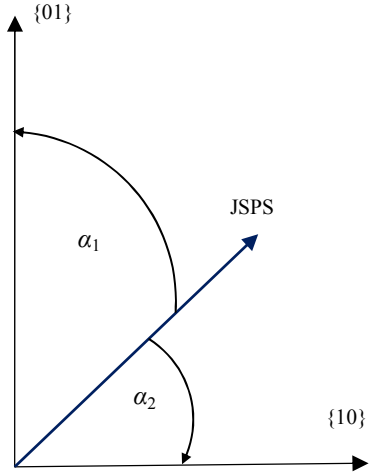

Fig. 1. JoyStick Probability Selector

Now, we will suggest that the probability of outcome {01} is equal to cosine square of angle α_1, while the probability of outcome {10} is equal to cosine square of angle α_2. It is not difficult to see that $\cos^2(\alpha_1) + \cos^2(\alpha_2) = 1$. We can see that the probability of the particular outcome of the experiment (in this case coin toss) is equal to the inner product of the unit norm JSPS and the unit norm vector which represents that outcome. Then, it is easy to see that this coincides to the Born rule interpretation for the case of a pure state and von Neumann measurement system. It is easy to see that when state (JSPS) vector collapses to one of the states, it is not possible to give information of the other state, which is consistent with some quantum mechanics results.

We can check what will happen if our discrete variable is of the dimension 3. In that case our system can be represented by Fig. 2. Now, we have three possible outcomes of the experiment that are represented by dummy variables {001}, {010} and {100}. Again we have the JSPS vector which

represents the status of our variable before we perform the measurement. Again, the probability of the outcome is given by the cosine square of the angle between JSPS and particular

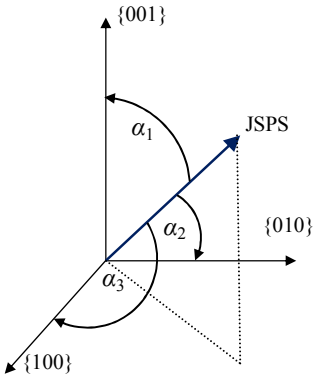

Fig. 2. A 3-D example

outcome vector. It is not difficult to check that

$$\sum_{i=1}^{3} \cos^2(\alpha_i) = 1. \quad (3)$$

It can be easily concluded that it follows from Pythagorean Theorem. For any 3-D vector whose norm is r we have

$$\sum_{i=1}^{3} r^2 \cos^2(\alpha_i) = r^2. \quad (4)$$

It can be easily concluded that this way of reasoning can be extended to any finite dimension D.

It is interesting to notice that this representation is actually reasonable for unimodal data (data that can be represented by a pure state). If the data is multi modal we need more complex model which is not going to be analyzed here.

IV. Probabilistic PCA

PCA also has another convenient interpretation that is perhaps less well known. In this probabilistic interpretation [24], each point $\boldsymbol{x}_i$ is thought of as a random draw from some unknown distribution $P_{\theta i}$, where $P_{\boldsymbol{\theta}}$ denotes a unit Gaussian with mean $\boldsymbol{\theta} \in \mathrm{R}^N$. Then, the purpose of PCA is to find the set of parameters $\boldsymbol{\theta}_1,\ldots, \boldsymbol{\theta}_m$ that maximize the likelihood of the data, subject to the condition that these parameters all lie in a low-dimensional subspace. In other words, $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m$ are considered to be a noise-corrupted version of some true points $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_m$ which lie in the subspace; the goal is to find these true points, and the main assumption is that the noise is Gaussian. The equivalence of this interpretation to the ones based on variance maximization follows simply from the fact that negative log likelihood under this Gaussian model is equal (ignoring constants) to the sum of the squared distances from the data points $\boldsymbol{x}_i$ to their projections $\boldsymbol{\theta}_i$ in the subspace. The Gaussian assumption may be inappropriate, for instance, if the data is binary-valued, or integer-valued, or is nonnegative [7]. In fact, the Gaussian is only one of the canonical distributions that make up the exponential family, and it is a distribution tailored to real-valued data. The Poisson is better suited to integer data and Bernoulli to binary data. It seems natural to consider variants of PCA which are founded upon these other distributions in place of the Gaussian.

So, in [7] PCA was extended to the rest of the exponential family. Let $\{P_{\boldsymbol{\theta}}\}$ be any parameterized set of distributions from the exponential family, where $\boldsymbol{\theta}$ is the natural parameter of a distribution. Given data $\boldsymbol{x}_1,\ldots,\boldsymbol{x}_m \in \mathrm{R}^N$, the goal is now to find parameters $\boldsymbol{\theta}_1,\ldots, \boldsymbol{\theta}_m$ which lie in a low dimensional subspace and for which the log-likelihood is maximized.

That approach effortlessly permits hybrid dimensionality reduction schemes in which different types of distributions can be used for different attributes of the data. If the data $\boldsymbol{x}_i$ have few binary attributes and a few integer-valued attributes, then some coordinates of the corresponding $\boldsymbol{\theta}_i$ can be the parameters of binomial distributions while others are the parameters of Poisson distributions.

In [7] was pointed out that dimensionality reduction schemes for non-Gaussian distributions are substantially different from PCA. For instance, in PCA the parameters $\boldsymbol{\theta}_i$, which are means of Gaussians, lie in a space which coincides with that of the data $\boldsymbol{x}_i$. This is not the case in general, and therefore, although the parameters $\boldsymbol{\theta}_i$ lie in a linear subspace, they typically correspond to a nonlinear surface in the space of the data.

Here, we have to point out two things when we are considering the existing probabilistic PCA approaches:

- First, it is frequently named probabilistic PCA although it should be correctly named probabilistic PSA. Although, PSA and PCA are similar techniques they should be considered as different. First of all, result of PCA takes a solution from a discrete set while the result of PSA is taken from a continuous set. Another important difference is that the input data covariance matrix is diagonal only in the coordinate system which is defined by eigenvectors themselves and not in any other rotated system.
- The second problem is that proposed probabilistic models are more related to factor analysis [7] and make strong assumptions about the distribution of the noise such as Gaussian or in more general case exponential family. It seems that those assumptions are too strong and as we are going to see, not really necessary.

Now, we will give a new definition of PCA that is different from already mentioned definitions that actually specify PSA. The definition is based on modification of the Hotteling definition [11]:

Definition 1: PCA can be defined as a problem of two successive optimization processes. In the first step, PSA is performed – which means that we are going to find orthogonal projection of the data onto lower dimensional linear space, known as principal subspace, such that the projected data is maximized. (In the text that follows, this step will have a different interpretation based on the Born rule, but we will avoid it here in order to simplify the definition.) In the second step the data projected in the principal subspace will be optimized in the sense that the entropy of the data is

minimized. In the second step, we perform learning on Grassman Principal Submanifold [13].

This formulation can be related to the recently proposed Time Oriented Hierarchical Method (TOHM) [12, 15] which represents a cooperative-competitive concept and is known as a general method that transforms PSA/MSA algorithms into PCA/MCA algorithms. The TOHM enables on-line realization in a single step (which means the data can be used only once; the problem is solved by introduction of multiple time scales) and will be presented later in the paper.

Formulation can also be seen as an approach similar to the one proposed in [8]. In this case first we are looking for a pool of the possible solutions (hyperensemble) and after we define it we are looking for the solution (ensemble) that fulfils the additional criterion. In other words, we are breaking one difficult problem into two simpler problems.

It is not difficult to see that the probabilistic MCA can be defined analogously as a two step successive optimization process. In the first step, the data is projected in the minor subspace, and in the second step entropy of the projected data should be minimized.

In the text that follows, we will concentrate on symmetric cost functions in the sense that possible artificial neural network realization is fully symmetric form the point of view of an individual neuron. Of course, it is not very difficult to give definitions that are more general so that they can include nonsymmetric cost functions (e.g. to cover the cases proposed in [1] or [16]), but it will not be discussed in this paper. In this paper, we consider nonsymmetric realizations as not completely successful realizations of the fully symmetric case, which is opposite to the point of view in which the fully symmetric realization represents a special case of nonsymmetric realizations.

Based on the Definition 1 and quantum probabilistic model we are going to give definition of probabilistic model for PCA. In order to do that, first we are going to give a new probabilistic approach to PSA.

A. Probabilistic PSA

Without a loss of generality, let's assume that we are dealing with a random variable $\boldsymbol{x}$ that take realizations from a set of observed N-dimensional zero-mean data vectors $\{\boldsymbol{x}_i\}$, $i \in \{1, \ldots, N_{\text{sample}}\}$ which are sampled from some distribution in time instants $t = iT$ where i is already defined and T represents the sampling period. We will denote $\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_K$ the column vectors of orthonormal matrix $\boldsymbol{W}$ ($K \leq N$). Now, we are going to define probabilistic PSA. In order to do that, we are going to specify the following cost function

$$\text{JS1}(\boldsymbol{W}) = \text{E}\big(p(\boldsymbol{W}, \boldsymbol{x})\big) = \sum_i p(\boldsymbol{W}, \boldsymbol{x}_i), \tag{5}$$

where E represents the empirical expectation operator, $\boldsymbol{W}$ is orthogonal matrix and $p(\boldsymbol{W}, \boldsymbol{x})$ is defined in the text that follows. This formulation represents only a class of algorithms that explicitly require orthonormality of matrix $\boldsymbol{W}$. For the cases where explicit orthonormality is not desirable constraint (e.g. in biologically plausible neural networks – since the explicit orthonormality usually requires global computation) we can formulate a different cost function which requires explicitly only unit norm of columns of the matrix $\boldsymbol{W}$ (that can be obtained by local calculations). This is the case for MH(O) type learning algorithms [14]. Here it will not be done due to space limitation.

We can write

$$p(\boldsymbol{W}, \boldsymbol{x}_i) = p(\boldsymbol{W}|\boldsymbol{x}_i)p(\boldsymbol{x}_i), \tag{6}$$

where we define

$$p(\boldsymbol{W}|\boldsymbol{x}_i) = \sum_{k=1}^{K} p(\boldsymbol{w}_k|\boldsymbol{x}_i). \tag{7}$$

Now, we will define $p(\boldsymbol{w}_k|\boldsymbol{x}_i)$ by usage of the Born rule. In that case we have

$$p(\boldsymbol{w}_k|\boldsymbol{x}_i) = \frac{\left(\boldsymbol{w}_k^{\mathrm{T}}\boldsymbol{x}_i\right)^2}{\|\boldsymbol{x}_i\|^2}. \tag{8}$$

In other words $p(\boldsymbol{w}_k|\boldsymbol{x}_i)=\cos^2(\boldsymbol{w}_k,\boldsymbol{x}_i)$. Generally speaking, we consider the data as a state vectors in the von Neuman measurement system which is defined by the columns of the matrix $\boldsymbol{W}$.

Also, we are going to define $p(\boldsymbol{x} = \boldsymbol{x}_i \mid t=iT)$ as

$$p(\boldsymbol{x} = \boldsymbol{x}_i) = \frac{\|\boldsymbol{x}_i\|^2}{\sum_{n=1}^{N_{sample}} \|\boldsymbol{x}_n\|^2}, \tag{9}$$

where N_{sample} represents the overall number of samples that are going to be analyzed. It is interesting to note that the only thing that we can conclude about the $p(\boldsymbol{x}_s)$ is that it is proportional to $\|\boldsymbol{x}_s\|^2$. The sum in the denominator represents the energy of samples that are going to be analyzed – we actually do not know the value of that sum in any, but the final moment. However, we know that it represents some constant. We can easily see that the adopted probability measure fulfils the two conditions that are required for probability function $f(\boldsymbol{x})$ (in our case $p(\boldsymbol{x})$) to be considered as a modified generalized probability measure [26]:

1. For each $\boldsymbol{x}$, $0 \leq f(\boldsymbol{x}) \leq 1$,
2. $\sum_i f(\boldsymbol{x}_i) = 1$.

If we want to calculate marginal probability $p(\boldsymbol{x}=\boldsymbol{x}_s)$ (i.e. probability of observing a certain value $\boldsymbol{x}_s$ no matter what is the time instant i) we can do it as

$$p(\boldsymbol{x} = \boldsymbol{x}_s) = \sum_{i=1}^{N_{sample}} \delta(\boldsymbol{x}_i, \boldsymbol{x}_s)p(\boldsymbol{x}_i), \tag{10}$$

where δ represents Kronecker δ function. In some sense, our approach is similar to the Dirichlet process model which is usually used to model a stream of symbols where the vocabulary of symbols is not limited [20]. In the Dirichlet process case we have one parameter α which is constant. In our case, it is not constant but it is defined by the energy of the new incoming symbol (sample). Also, in our case every incoming symbol is treated as a new one – so initially we do not consider that the same samples in different moments are the same (e.g. we can make analogy to spin of particle). That

means that our input space is initially considered as an augmented space of dimension N_{sample}. When we observe all samples, we can switch to the "deformed" space of dimension N_d, which is defined by the number of different energy levels d.

We can see that (5) can be written as

$$\mathrm{JS1}(\boldsymbol{W}) = \mathrm{E}(\boldsymbol{W}, \boldsymbol{x}) = \sum_i p(\boldsymbol{W}, \boldsymbol{x}_i) = \frac{1}{\sum_{j=1}^{N_{sample}} \|\mathrm{x_j}\|^2} \left(\sum_{k=1}^{K} \sum_{i=1}^{N_{sample}} \boldsymbol{w}_k^{\mathrm{T}} \boldsymbol{x}_i \boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{w}_k \right). \quad (11)$$

Now, we define probabilistic PSA as a constrained maximization problem

$$\max_{\boldsymbol{W}} \mathrm{JS1}(\boldsymbol{W}), \quad (12)$$

under the constraint that $\boldsymbol{W}$ is an orthonormal matrix. So, we are looking for $\boldsymbol{W}$ which maximizes joint probability $p(\boldsymbol{W}, \boldsymbol{x})$ on the average. Solution will be obtained for matrix $\boldsymbol{W}=\boldsymbol{W}^{\mathrm{JS1}}$ which represents the rotated matrix of matrix $\boldsymbol{U}$ which consists of eigenvectors $\boldsymbol{u}_i$ as columns, where $\boldsymbol{u}_i$ represent K dominant eigenvectors of the covariance matrix $\boldsymbol{C} = \mathrm{E}(\boldsymbol{x}\boldsymbol{x}^{\mathrm{T}})$. Again, the matrix $\boldsymbol{W}$ is selected in such a way that we maximize, on the average, joint probability $p(\boldsymbol{W}, \boldsymbol{x})$, where the columns of $\boldsymbol{W}$ represent measurement system with $\boldsymbol{x}$ as a state vectors. It is not difficult to see that with this definition, PSA also can be extended to a clustering problem – see e.g. [17, 27]. Also, it is easy to see that application of gradient ascent to JS1 maximization will lead to Hebb learning rule.

B. Probabilistic PCA

So, as a result of PSA we will obtain a "pool" or "hyperensemble" of matrices $\boldsymbol{W}^{\mathrm{JS1}}$ that span the subspace of the empirical covariance matrix of zero-mean input samples $\boldsymbol{x}$. Now, in order to create PCA, we are going to do the second step. In this step we are going to select an ensemble of solutions that fulfill additional criterion, once the matrix $\boldsymbol{W}$ satisfies the solution of PSA problem.

Let's assume that the matrix $\boldsymbol{W}=\boldsymbol{W}^{\mathrm{JS1}}$ is an orthonormal matrix and that it spans the principal subspace. Then, we are going to analyze the following cost function

$$\mathrm{JS2}(\boldsymbol{W}^{\mathrm{JS1}}) = \mathrm{E}\left(\mathrm{S}\left(p(\boldsymbol{W}^{\mathrm{JS1}}, \boldsymbol{x}) \right) \right) \quad (13)$$

where S represents the entropy (see next paragraph). If we chose the Tsallis entropy [25], (13) can be written as

$$\mathrm{JS2}(\boldsymbol{W}^{\mathrm{JS1}}) = \mathrm{E}\left(\frac{1 - \sum_{k=1}^{K} p^q(\boldsymbol{w}_k^{\mathrm{JS1}}, \boldsymbol{x})}{q-1} \right), \quad (14)$$

where q is a nonnegative real value, different from 1, and again E represents empirical expectation. In the case $q \to 1$, we will have the Shannon entropy $\mathrm{S}=\mathrm{S_S}$, so our cost function is defined as

$$\mathrm{JS2}(\boldsymbol{W}^{\mathrm{JS1}}) = \mathrm{E}\left(\sum_{k=1}^{K} p(\boldsymbol{w}_k^{\mathrm{JS1}}, \boldsymbol{x}) \log \left(\frac{1}{p(\boldsymbol{w}_k^{\mathrm{JS1}}, \boldsymbol{x})} \right) \right). \quad (15)$$

We can, also, use some other definition of entropy. Now, probabilistic PCA can be defined as the following constrained minimization problem

$$\min_{\boldsymbol{W}^{\mathrm{JS1}}} \mathrm{JS2}(\boldsymbol{W}^{\mathrm{JS1}}) \ , \quad (16)$$

under constraint that $\boldsymbol{W}^{\mathrm{JS1}}$ is orthonormal and spans the principal subspace of matrix $\boldsymbol{C}$. It can be shown that the solution of this optimization process will be $\boldsymbol{W}^{\mathrm{JS1}}= \boldsymbol{U}\boldsymbol{P}$, where, as before, $\boldsymbol{U}$ consists of K principal eigenvectors $\boldsymbol{u}_i$ as columns, $\boldsymbol{u}_i$ represent principal eigenvectors of the covariance matrix $\boldsymbol{C} = \mathrm{E}(\boldsymbol{x}\boldsymbol{x}^{\mathrm{T}})$, and $\boldsymbol{P}$ is a permutation matrix. This can be seen from section 11.3 in [21] for the Shannon entropy and can be done in a similar way for other types of entropy (proof is omitted here, due to space limitation).

We will have to emphasize that, strictly speaking, proposed equations are correct only in the case $K=N$, since in the case $K < N$, $\sum_{k=1}^{K} p(\boldsymbol{w}_k^{\mathrm{JS1}}, \boldsymbol{x}) < 1$. In the case $K < N$ we will define the term "principal subspace" entropy, $\mathrm{S^{PS}}$. Here, we will analyze only the Shannon entropy case, but similar can be done for Tsallis or other types of entropy. We define

$$\mathrm{S^{PS}}(\boldsymbol{W}^{\mathrm{JS1}}) = \sum_{k=1}^{K} p(\boldsymbol{w}_k^{\mathrm{JS1}}, \boldsymbol{x}) \log \left(\frac{1}{p(\boldsymbol{w}_k^{\mathrm{JS1}}, \boldsymbol{x})} \right). \quad (17)$$

Correct definition of Shannon entropy in the principal subspace can be done as

$$\mathrm{S_S}(\boldsymbol{W}^{\mathrm{JS1}}) = \sum_{k=1}^{K} p^{\mathrm{PS}}(\boldsymbol{w}_k^{\mathrm{JS1}}, \boldsymbol{x}) \log \left(\frac{1}{p^{\mathrm{PS}}(\boldsymbol{w}_k^{\mathrm{JS1}}, \boldsymbol{x})} \right) \quad (18)$$

where

$$p^{\mathrm{PS}}(\boldsymbol{w}_k^{\mathrm{JS1}}, \boldsymbol{x}) = \frac{p(\boldsymbol{w}_k^{\mathrm{JS1}}, \boldsymbol{x})}{\sum_{i=1}^{K} p(\boldsymbol{w}_i^{\mathrm{JS1}}, \boldsymbol{x})} = \frac{p(\boldsymbol{w}_k^{\mathrm{JS1}}, \boldsymbol{x})}{Cl} \ . \quad (19)$$

Definition of constant Cl is obvious from (19), and now we have $\sum_{i=1}^{K} p^{PS}(\boldsymbol{w}_i^{\mathrm{JS1}}, \boldsymbol{x}) = 1$. Now, we can write (18) as

$$\mathrm{S_S}(\boldsymbol{W}^{\mathrm{JS1}}) = \log(Cl) + \frac{1}{Cl} \mathrm{S^{SP}}(\boldsymbol{W}^{\mathrm{JS1}}) \ . \quad (20)$$

From (20) can be seen that minimization of $\mathrm{S_S}$ ($\boldsymbol{W}^{\mathrm{JS1}}$) is equivalent to minimization of $\mathrm{S^{SP}}(\boldsymbol{W}^{\mathrm{JS1}})$, so equations (14) and (15) can be formally used even in the cases when $K<N$. Other way to show this easily, would be to project input data into principal subspace and then to use such data as an input data in minimization of JS2.

Equation (13) can be made more general if we use a monotonic increasing function of entropy instead of the entropy itself.

C. Minimization of approximate Tsallis entropy

In this section we will give a very simple derivation of the special case of the proposed PCA method, which will be also useful when we, later, discuss how PCA can be realized on-line. In a special case, PCA can be seen as a minimization of the approximate Tsallis entropy, $S_T(q \approx 1)$.

Let's consider the Tsallis entropy with parameter $q = 1+\varepsilon$, where ε is very small. In that case we have

$$\mathrm{E}\big(\mathrm{S_T}(\boldsymbol{W}, q)\big) = \mathrm{E}\left(\frac{1-\sum_{k=1}^{K} p^q(\boldsymbol{w}_k, \boldsymbol{x})}{q-1}\right). \quad (21)$$

When q is close to 1 we can have the following approximation

$$p^{q-1}(\boldsymbol{w}_k, \boldsymbol{x}) = \mathrm{e}^{(q-1)\ln\left(p(\mathbf{w}_k, \boldsymbol{x})\right)} \approx 1 + (q-1)\ln\big(p(\boldsymbol{w}_k, \boldsymbol{x})\big). \quad (22)$$

Then, we can write

$$\mathrm{E}\big(\mathrm{S_T}(\boldsymbol{W}, q)\big) \approx \mathrm{E}\left(\frac{1-\sum_{k=1}^{K} p(\boldsymbol{w}_k, \boldsymbol{x})}{q-1} - \sum_{k=1}^{K} p(\boldsymbol{w}_k, \boldsymbol{x})\ln\big(p(\boldsymbol{w}_k, \boldsymbol{x})\big)\right), \quad (23)$$

or

$$\mathrm{E}\big(\mathrm{S_T}(\boldsymbol{W}, q)\big) \approx \mathrm{E}\left(\frac{1-\sum_{k=1}^{K} p(\boldsymbol{w}_k, \boldsymbol{x})}{q-1}\right) + \mathrm{E}\big(\mathrm{S_S}(\boldsymbol{W})\big). \quad (24)$$

So, in this case, when q is close to 1, the Tsallis entropy can be approximately expressed as the sum of two terms – one that depends on joint probabilities and the other that represents the Shannon's entropy. The first term is multiplied by some factor $(1/(q-1))$ which is much bigger then 1, like it is calculated on a faster time-scale than the second term that represents entropy. Now, based on discussion in previous two subsections, we can conclude that minimization of the approximate Tsallis entropy will lead toward PCA. It is interesting to notice that in the case $N=K$ we only need to minimize the Shannon entropy term since the $\Sigma_j\, p(\boldsymbol{w}_j, \boldsymbol{x}) = 1$. Also, we point out that in on-line learning algorithms, where the matrix $\boldsymbol{W}$ is only approximately orthonormal (since the norms of the columns of the matrix $\boldsymbol{W}$ are slightly different from 1), $\Sigma_j\, p(\boldsymbol{w}_j, \boldsymbol{x})$ can be understood more precisely as $\Sigma_j\, p(\boldsymbol{w}_j, \boldsymbol{x})^a$, for some a that is not exactly one (so the effective exponent is actually a little bit different from 1). In the case of strongly orthonormally constrained (SOC) algorithms, which means that $\boldsymbol{W}$ is orthonormal in all learning steps, effective exponent $a = 1$.

V. Practical Aspects of Implementation

In this section we will discuss some practical aspects of the implementation of the proposed method. We will discuss off-line implementation and on-line implementation (samples can be used only once) cases separately.

First, based on discussion in the previous section, we can sketch a following algorithm:

Algorithm for off-line calculation of PCA:

In: Matrix $\boldsymbol{X}$ of size $N \times N_{sample}$
In: Desired number of principle components
Computation:
Step 1:
Compute $\mathbf{W}^{\mathbf{JS1}} = \max_{\mathrm{W}} \mathrm{JS1(W)}$ where

$$\mathrm{JS1(W)} = \mathrm{E}(\boldsymbol{W}, \boldsymbol{x}) = \sum_i p(\boldsymbol{W}, \boldsymbol{x}_i) = \frac{1}{\sum_{j=1}^{N_{sample}} \|\boldsymbol{x}_j\|^2}\left(\sum_{k=1}^{K}\sum_{i=1}^{N_{sample}} \boldsymbol{w}_k^{\mathrm{T}} \boldsymbol{x}_i \boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{w}_k\right)$$

Step 2:

Compute $\boldsymbol{W}^{\mathbf{JS2}} = \min_{\boldsymbol{W}^{\mathrm{JS1}}} \mathrm{JS2}(\boldsymbol{W}^{\mathrm{JS1}})$ where

$$\mathrm{JS2}(\boldsymbol{W}^{\mathrm{JS1}}) = \mathrm{E}\left(\mathrm{S}\left(p(\boldsymbol{W}^{\mathrm{JS1}}, \boldsymbol{x})\right)\right).$$

From the implementation point of view, this realization is not optimal because we should find the pool of the matrices $\boldsymbol{W}^{\mathrm{JS1}}$ that minimize function JS1. This is impossible if we want to find all of them and can be pretty demanding from the point of view of calculation time and storage capacity if we want to have a big, but finite number of such solutions. Instead, it is much better to find one solution $\boldsymbol{W}^{\mathrm{JS1}*}$, and to use it to project input data $\boldsymbol{X}$ to $\boldsymbol{X}^{\mathrm{PSA}}$ that lies in the principal subspace. In the first step we can use any of the available algorithms for PSA calculation. Then we can use the $\boldsymbol{X}^{\mathrm{PSA}}$ as an input to the Step 2 of the algorithm, and find a solution $\boldsymbol{W}^{\mathrm{JS2}}$ that minimizes the cost function $\mathrm{JS2}(\boldsymbol{W}^{\mathrm{JS1}*})$. In that case the algorithm looks like:

Improved algorithm for off-line calculation of PCA:

Input:
In: Matrix $\boldsymbol{X}$ of size $N \times N_{sample}$
In: Desired number of principle components K
Computation:
Step 1:
Compute $\boldsymbol{W}^{\mathrm{JS1}}$ that spans the principal subspace of the empirical covariance matrix $\boldsymbol{C}=\boldsymbol{X}*\boldsymbol{X}^{\mathrm{T}}$. Then, project $\boldsymbol{X}$ to that subspace and get $\boldsymbol{X}^{\mathrm{PSA}}=\boldsymbol{W}^{\mathrm{JS1T}}*\boldsymbol{X}$.

Step 2:
Use $\boldsymbol{X}^{\mathrm{PSA}}$ as an input matrix and select some entropy measure S.
Compute $\boldsymbol{W}^{\mathrm{JS2}} = \min_{\boldsymbol{W}^{\mathrm{JS1}}} \mathrm{JS2}(\boldsymbol{W}^{\mathrm{JS1}})$ where

$$\mathrm{JS2}(\boldsymbol{W}^{\mathrm{JS1}}) = \mathrm{E}\left(\mathrm{S}\left(p(\boldsymbol{W}^{\mathrm{JS1}}, \boldsymbol{x}^{\mathrm{PSA}})\right)\right).$$

In realization of step 1, we can use any existing off-line PSA learning algorithm, while in step 2 we can use any algorithm that minimizes selected entropy function.

In on-line realization, we can use every sample only once – which means we cannot use the successive optimization

technique that is proposed. The solution can be based on minimization of approximate the Tsallis entropy already explained in subsection IV.C. Here it we will be introduced as a cooperative-competitive concept that was used in artificial neural networks and which is called the time oriented hierarchical learning (TOHM). In this approach, two time scales are proposed (it can be easily modified for the case of multiple-time scales). The idea of multiple time scales is used in physics for simulation purposes [4, 23], and, also, represents the major idea that support the superstatistics concept [3].

In TOHM on a faster time-scale we calculate parameters of global interest (principal subspace), while on the slower time scale, we calculate parameters that are of local interest (principal eigenvectors). Concrete realization is based on Lagrangian multiplier idea – solution that should be obtained on the faster time scale is used as a constraint on the slower time-scale. This can be formally represented as the maximization of the following cost function

$$\mathrm{JS}(\boldsymbol{W}) = \mathrm{JS1}(\boldsymbol{W}) - \mu\,\mathrm{JS2}(\boldsymbol{W}) \qquad (25)$$

where μ is a real number which is by absolute value smaller than one (since the JS2 should be realized on a slower time-scale). We can see here that Lagrangian multiplier is used in an unusual way – it does not multiply the constraint part, but the objective one. Anyway, it will not affect the solution since it just represents the weight ratio of the different parts of the learning rule.

Several algorithms that perform PCA, and that are based on the TOHM method are proposed in the literature [12], [13], [15] and [16]. Without going into the details, in all those approaches the PCA algorithms are obtained from some known PSA algorithms together with minimization of the Tsallis entropy, $S_T(q \approx 1)$. Value of μ was selected in all algorithms independently, after stability analysis for every particular choice of JS1 and JS2.

VI. Experimental Results

In this section, we will present some small scale simulation results. We will consider a single layer linear artificial neural network with four inputs and two outputs. In all experiments, the adopted PSA algorithm was as in [22] and PCA parts of algorithms were obtained by minimization of four different entropy functions, or more precisely, the principal subspace entropy functions. Selected entropy functions were Tsallis entropy with q=0.5, q≈1 and q=2 as well as the Shannon entropy function. In figures that follow, Error was defined as

$$\mathrm{Error} = \log\big(\mathrm{abs}(\mathrm{abs}(\cos(\boldsymbol{v}_1, \boldsymbol{u}_1)) - 1)\big)$$

where $\boldsymbol{v}_1$ represents principal eigenvector calculated by neural network, while $\boldsymbol{u}_1$ represents numerically calculated principal eigenvector of input covariance matrix. In all experiments, the learning factors (~1/t) and the initial value of the weight matrix were the same.

In Fig. 3, the input signals were generated as a linear mixture of random Gaussian noise. In Fig. 4 input signals were generated as linear mixture of random uniform noise. Sinusoidal signals were used as input in Fig. 5. Input signals in Fig. 6 were generated as a mixture random uniform noise but learning rate is decreased 10 times.

We can see that, depending on the type of input signals, we have different behaviors of algorithms. Although we cannot say that the choice of the particular entropy measure will

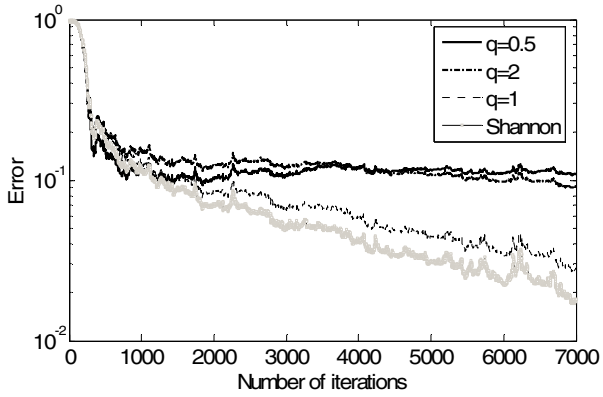

Fig. 3. Input is linear mixture of random Gaussians signals

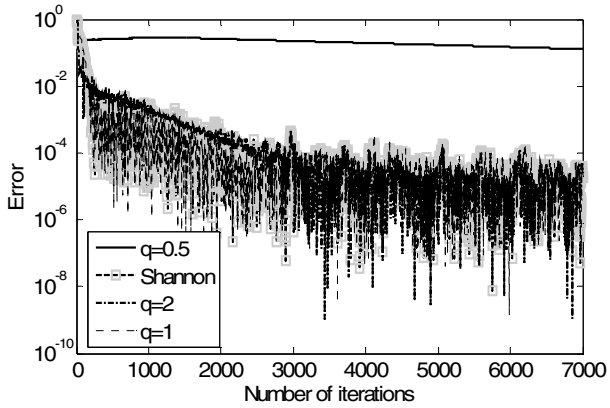

Fig. 4. Input is a linear mixture of random uniformly distributed signals

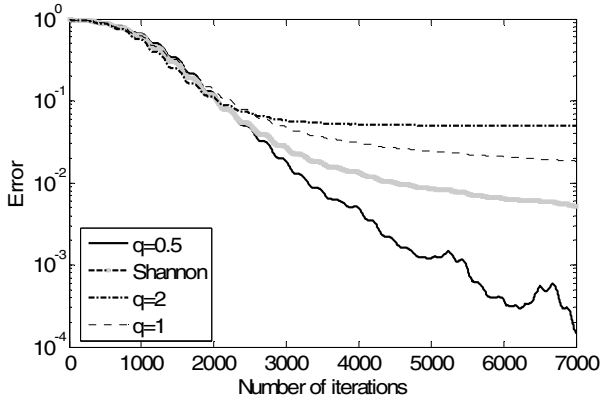

Fig. 5. Input is represented by sinusoidal signals

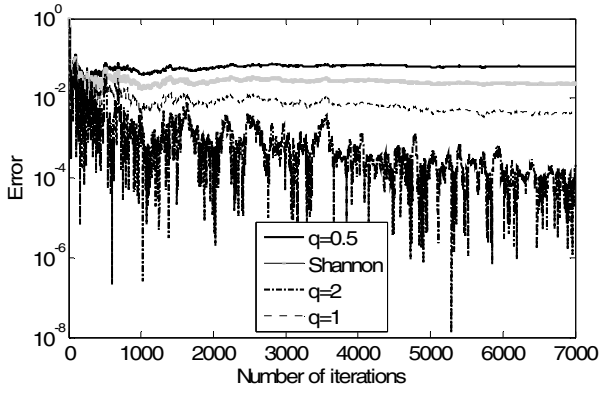

Fig. 6. Input is same as in a picture 4, but learning rate is decreased 10 times

always lead to fastest convergence and highest precision, based on presented experimental results, we can see that selection of the Shannon entropy and the Tsallis entropy with $q \approx 1$, will mainly result in a satisfactory performance. In some sense, those two choices represent robust/general entropy measures, while the two others represent entropy measures suitable for signals or networks with specific features.

VII. Conclusion

In this paper we proposed a probabilistic principal component analysis based on the Born rule and its new, simple geometrical representation, named JoyStick Probability Selector. Proposed probabilistic PCA can be solved off-line as a successive optimization problem. In on-line implementation, the problem is solved by the introduction of two distinct time-scales. By selection of different PSA algorithms and different entropy functions, it is possible to create a big number of algorithms that can be used for PCA. In that way, it is possible to create algorithms that could be potentially optimal from the point of view of convergence speed, preciseness, complexity of hardware implementation, locality of calculations etc. A few simulation results were used to illustrate how the choice of entropy function affects the convergence speed and preciseness of the PCA algorithm implemented in an artificial neural network setup.

Proposed probabilistic PSA was based on existence of explicit orthonormality constraint for weight matrix. In the papers that follow, it will be explained how proposed model can be modified to include the cases where the orhtonormality constraint is implicit (e.g. MH(O) type algorithms).

As it is briefly mentioned, the proposed approach can be easily modified to give definition of probabilistic MCA.

It is possible to show that similar way of reasoning can lead to a new, probabilistic definition of independent component analysis (ICA). By doing this, it is possible to give different interpretations of some existing ICA solutions and to propose new ones. This is currently under investigation.

Acknowledgment

The authors are grateful to Dr. Ryota Tomioka for useful discussions which improved paper's quality and readability.

References

[1] J.-H. Ahn, J.-H. Oh and S. Choi, "Learning principal directions: Integrated-Squared-Error minimization", *Neurocomputing*, vol. 70, pp. 1372- 1381, 2007.

[2] P. Baldi and K. Hornik, "Learning in linear neural networks: A survey", *IEEE Trans. Neural Networks*, vol. 6, pp. 837-858, 1995.

[3] C. Beck, "Superstatistics: theory and applications", *Continuum Mech. Thermodyn.*, vol. 16, pp. 293-304, 2004.

[4] C.M. Bishop, *Pattern Recognition and Machine Learning*. New York: Springer Science+Business Media, LLC, 2006.

[5] T.-P. Chen and S.-I Amari, "Unified stabilization approach to principal and minor components", *Neural Networks,* vol. 14, pp. 1377-1387, 2001.

[6] A. Cichocki and S.-I. Amari, *Adaptive Blind Signal and Image Processing – Learning Algorithms and Applications*. John Wiley and Sons, 2003.

[7] M. Collins, S. Dasgupta, and R. E. Schapire, "A generalization of principal component analysis to the exponential family", *Advances in Neural Information Processing Systems*, vol. 14, pp. 617-624, 2002.

[8] G. E. Crooks, "Beyond Boltzmann-Gibbs statistics: Maximum entropy hyperensembles out of equilibrium", *Physical Review*, E 75: 041119, 2007 .

[9] P. Földiák, "Adaptive network for optimal linear feature extraction", *IJCNN'89*, pp. 401-405, 1989.

[10] A. M. Gleason, "Measure on the closed subspaces of a Hilbert space", *J. Math. & Mechanics,* vol. 6, pp. 885-893, 1957.

[11] H. Hotteling, "Analysis of a complex of statistical variables into principal components", *Journal of Educational Psychology*, vol. 24, pp. 417-441, 1933.

[12] M. Jankovic and H. Ogawa, "Time-Oriented hierarchical method for computation of principal components using subspace learning algorithm", *Int. J. Neural Systems*, vol. 14, pp. 313-324, 2004.

[13] M. Jankovic and B. Reljin "Neural learning on Grassman/Stiefel principal/minor submanifold", EUROCON 2005, pp. 249-252, 2005.

[14] M. Jankovic and H. Ogawa, "Modulated Hebb-Oja learning rule – A method for principal subspace analysis", *IEEE Trans. on Neural Networks,* vol. 17, pp. 345-356, 2006.

[15] M. Jankovic and B. Reljin, "General stochastic approximation and neural learning on principal Stiefel submanifold", *WSEAS Trans. on Information Science and Applications*, vol. 3, pp. 1195-1201, 2006.

[16] M. Jankovic and M. Sugiyama, "A multipurpose linear component analysis method based on modulated Hebb Oja learning rule", *IEEE Signal Processing Letters*, vol. 15, pp. 677-680, 2008.

[17] M. Jankovic, M. Sugiyama and B. Reljin, "Tensor based image segmentation", in *Proc. 9th Symposium on Neural Network Applications in Electrical Engineering*, Belgrade, 2008, pp. 67-73.

[18] I. T. Jolliffe, *Principal Component Analysis,* 2nd ed., New York: Springer –Verlag, 2002.

[19] J. Karhunen, E. Oja, L. Wang, R. Vigario and J. Joutensalo, "A class of neural networks for independent component analysis", *IEEE Trans. on Neural Networks*, vol. 8, pp. 486-504, 1997.

[20] D. J. C. MacKay, *Information Theory, Inference and Learning Algorithms.* Cambridge University Press, 2003.

[21] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information.* Cambridge University Press, 2000.

[22] *E. Oja, "Neural Networks, principal components, and subspaces", Int. J. Neural Syst.*, vol. 1, pp. 61–68, 1989.

[23] J. C. Sexton and D. H. Weingarten, "Hamiltonian evolution for the Hybrid Monte Carlo algorithm", *Nuclear Physics B*, vol. 380, pp. 665-677, 1992.

[24] M. Tipping and C. M. Bishop, "Probabilistic Principal Component Analysis", *Journal of the Royal Statistical Society, Series B*, vol.61, part 3, pp. 611-622, 1999.

[25] C. Tsallis, "Possible generalization of Boltzmann-Gibbs statistics", *J. Statistical Physics,* vol. 52, pp. 479-478, 1988.

[26] M. K. Warmuth and D. Kuzmin, "A Bayesian probability calculus for density matrices", in *Proc. 22nd Conference on Uncertainty in Artificial Intelligence*, 2006, pp. 503-511.

[27] L. Wolf, "Learning using the Born rule", Tech. Rep. MIT-CSAIL-TR-2006-036, 2006.

[28] W. K. Wooters, "Statistical distance and Hilbert space", *Physical Review D.*, vol. 23, pp. 357-362, 1981.