Probabilistic Approach to Neural Networks
Computation Based on Quantum Probability Model

Probabilistic Principal Subspace Analysis Example

Marko V. Jankovic, Institute of Electrical Engineering “Nikola Tesla”, Belgrade, Serbia

Abstract—In this paper, we introduce elements of probabilistic model that is suitable for modeling of learning algorithms in biologically plausible artificial neural networks framework. Model is based on two of the main concepts in quantum physics – a density matrix and the Born rule. As an example, we will show that proposed probabilistic interpretation is suitable for modeling of on-line learning algorithms for PSA, which are preferably realized by a parallel hardware based on very simple computational units. Proposed concept (model) can be used in the context of improving algorithm convergence speed, learning factor choice, or input signal scale robustness. We are going to see how the Born rule and the Hebbian learning rule are connected.

Keywords—Born rule, Hebbian learning rule, probabilistic PSA/PCA, von Neumann entropy.
Abstract— In this paper, we introduce probabilistic interpretation of the Principal Subspace Analysis (PSA) based on two of the main concepts in quantum physics – a density matrix and the Born rule. We analyze the proposed probabilistic model in the artificial neural network framework. In other words, we are looking for suitable probabilistic interpretation of on-line learning algorithms for PSA, which are preferably realized by a parallel hardware based on very simple computational units. First, we give a probabilistic interpretation of the Modulated Hebb-Oja learning method. Later, we show how the proposed method can be generalized in order to obtain a number of PSA algorithms. At the end, we present some experimental results to illustrate the usefulness of the proposed probabilistic PSA model.
Index Terms— Born rule, modulated Hebb-Oja learning rule, probabilistic PSA/PCA, time-oriented hierarchical learning, von Neumann entropy

I. INTRODUCTION

Principal subspace analysis (PSA) and principal component analysis (PCA) (e.g. [23]) are the well established techniques 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.

In literature, usually, we can find that definitions of PSA and PCA are not given separately. Here, we will give two most frequently cited definitions, and then we will give a short discussion:

PCA can be defined as the orthogonal projection of the data onto a lower dimensional linear space, known as the principal subspace, such that the variance of the projected data is maximized (Hotteling, 1933 [15]).

In the context of neural networks, this definition leads to the Subspace Learning Algorithm (SLA) [30] and its derivatives.

Equivalently, it can be defined as the linear projection that minimizes the average projection cost, defined as the mean square distance between the data points and their projections (Pearson, 1901 [31]).
In the context of neural networks, this definition represents the starting point of the Modulated Hebb-Oja algorithm [18] and its derivatives.

Here we will point out two things:

- In both cases definitions of PSA but not PCA are presented. Although PSA and PCA are similar techniques, they should be considered as different. First of all, the solution of PCA is represented by a discrete set (any permutation of eigenvectors of the input covariance matrix), while the solution of PSA is represented by 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, but not in any other rotated system. Also, if the data belongs to the low-dimensional subspace of the input space, it will have a sparse representation if the PCA is applied, while this, generally, will not be achieved if the PSA is applied. Here, it has to be stressed that this can be achieved only if the input data is unimodal. If the input data is multimodal, then PCA will not result in sparse representation either.

- In the second definition, we used a term equivalently – here, we want to stress out that definitions are equivalent form the mathematical point of view. However, from the realization point of view, these two definitions are quite different (see [18]). The first one requires a big number of global calculation circuits, while the second one requires only one or two global calculation circuits.
PSA also has another convenient interpretation that is perhaps less well known. It is the probabilistic interpretation given in [34]. Although it is called probabilistic PCA, it represents probabilistic PSA model. In order to achieve different objectives, it is further generalized in [3], [7], [10], [12] or [35]. All these models are based on some distribution of latent variable model, so they can be used also in the case when we have problem of missing data. One of the most powerful algorithms for iterative PSA calculation is EM type algorithm proposed in [34, 35]. The EM algorithm can be modified in such a way as to yield principal components, sorted in descending order of the corresponding eigenvalues, directly [1], or PCA method. This model can be linked to probabilistic coupled generative model [1]. Proposed models resulted in great number of useful applications in the context of clustering, density modeling, outlier detection etc. Common characteristic of all mentioned probabilistic models is that they are useful for generation of off-line algorithms (they require storage of all data or their covariance matrix). So, they usually require powerful hardware for their realization. In the case that we need tracking of slow changes of correlations in the input data or in updating eigenvectors with new samples these algorithms are computationally expensive. Also, they rely on the assumption that input data is sampled from some distribution (Gaussian, exponential family, t-distribution, or similar) or their mixtures. So, it is not clear how these probabilistic models can be used in the context of artificial neural networks (or on-line learning algorithms) for PSA/PCA calculation, and/or, could they be realized on a very simple computational structures like linear neural networks are.
In this paper, we will proposed a sort of “quantum” probabilistic PSA model which relies on a very small number of assumptions and that is suitable, as we are going to see, for modeling the on-line type learning algorithms for calculation of PSA/PCA. We will illustrate the usefulness of the proposed concept (model) in the context of algorithm convergence speed, learning factor choice, or input signal scale robustness.

Why are we still interested in standard linear neural networks approach? 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 [4, 9, 18, 29]. Due to their low complexity, such algorithms and their implementation in neural networks are potentially useful for the tracking of slow changes of correlations in the input data or in updating eigenvectors with new samples. They, also, could be successfully implemented on platforms like graphics processing unit (GPU). Furthermore, this kind of networks can be used for calculations of PCA and PSA for the higher dimensional systems than those solutions that require the storage of the whole covariance matrix. Another reason is that neural networks approach based on biologically plausible learning rules are still useful for the research in which the goal is to make computational models that emulate some of the brain circuitry.

In recent years, many Principal Subspace Analysis (PSA) algorithms have been proposed and studied in literature [4, 8, 13, 16, 18, 29]. Some of these PSA algorithms were modified in order to derive parallel PCA algorithms. Usually, the modification was performed by introduction of some asymmetry (inhomogenity) 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 [26], and the time-oriented hierarchical method (TOHM), proposed in [16, 19]. For a comprehensive review of the known parallel, as well as sequential PCA algorithms, see e.g. [9].

Recently, in [18], a new biologically inspired PSA method, named Modulated Hebb-Oja (MHO) learning rule, has been introduced. Major objectives for the method derivation were:

— to obtain a network which has a learning rule for individual synaptic efficacy that requires the least possible amount of explicit information about the other synaptic efficacies, especially those related to other neurons (in other words the locality of the calculation was considered very important);

— to minimize the neural hardware that is necessary for implementation of the proposed learning rule. Especially, the global calculation “neural circuitry” should be very simple, and the number of global calculation circuits should be as small as possible.

In this paper, we will give a probabilistic view of the PSA problem, starting with a probabilistic model of MHO learning algorithm. The probabilistic view is based on a new geometric interpretation of probability based on the Born rule [38]. Also, we will shortly introduce recently proposed probabilistic PCA model. In the case of on-line implementation, we will use constraint optimization on two time scales, as is done in the TOHM method, for instance. From the aspect of artificial neural networks, the choice of different realization concepts has a direct impact on the algorithm’s convergence speed, preciseness,
complexity of plausible hardware realization or biological plausibility. In an off-line context, probabilistic PCA can be seen as a successive optimization problem. In Section II, the Born rule will be introduced. Geometric interpretation of the Born rule will be given in Section III. The Born rule in the artificial neural networks framework is introduced in Section IV. In section V we introduce of new probabilistic PSA. In that section we will see what is the connection between the Born rule and the Hebbian learning rule. Experimental results are presented in Section VI. Section VII gives the conclusion remarks.

II. QUANTUM PROBABILITY MODEL AND BORN RULE

In this section, we give a short introduction of the quantum probability model and the Born rule, based on a similar section in [38]. A slightly different approach can be found in [37]. In quantum mechanics, the transition from a deterministic description to a probabilistic one is done using a simple rule, termed the Born rule.

In quantum mechanics, the Born rule is usually taken as one of the axioms. However, this rule has well established foundations. Gleason’s theorem [14] states that the Born rule is the only consistent probability distribution for a Hilbert space structure. Wooters [39] 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. A short review for some other justifications of the Born rule can be seen in [38].
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 $\rho$) from this space to itself, with a trace of 1, i.e. $\forall \Psi \in H \quad \Psi^T \rho \Psi \geq 0$, $\text{Tr}(\rho) = 1$. Such $\rho$ is self adjoint and is called a density matrix.

Since $\rho$ is self adjoint, its eigenvectors $\Phi_i$ 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 $\sum p_i = 1$.

The equation $\rho = \sum_i p_i \Phi_i \Phi_i^T$ is interpreted as “the system is in state $\Phi_i$ with probability $p_i$”. The state $\rho$ is called the pure state if $\exists i$ s.t. $p_i = 1$. In this case, $\rho = \Psi \Psi^T$ for some normalized state vector $\Psi$, and the system is said to be in state $\Psi$. So, the most general density operator is in the form $\rho = \sum_k p_k \Psi_k \Psi_k^T$ where the coefficients $p_k$ are nonnegative and add up to one, and $\Psi_k$ represent pure states. We can see that this decomposition is not unique.

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

$$p_z(\rho) = \text{trace}(\rho m_z) \quad (1)$$

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 $m_a = aa^T$, for which $a^T a' = \delta_{aa'}$. 
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, \( \sum_{a \in M} a a^\dagger = 1 \). For this type of measurement, the Born rule takes a simpler form: \( p_a(\rho) = a^\dagger \rho a \). Assuming \( \rho \) is a pure state this can be simplified further to

\[
p_a(\rho) = (a^\dagger \Psi)^2. \quad (2)
\]

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

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 a finite dimensional discrete variable.

For the moment, let’s assume that we are dealing with a discrete two dimensional variable. We can associate it 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.
Now, we will suggest that the probability of the outcome \{01\} is equal to the cosine square of angle \(\alpha_1\), while the probability of outcome \{10\} is equal to the cosine square of angle \(\alpha_2\). 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 the coin toss) is equal to the inner product of the unit norm JSPS and the unit norm vector which represents that outcome. Then, we can see that this coincides with the Born rule interpretation for the case of a pure state and the von Neuman measurement system. When the 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 as Fig. 2. Now, we have the 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 the particular outcome vector.

\[
\sum_{i=1}^{3} \cos^2(\alpha_i) = 1, \quad (3)
\]

which follows from generalized Pythagorean Theorem, or Parseval’s Theorem. For any 3-D vector whose norm is \(r\) we have
This way of reasoning can be extended to any finite dimension $D$. It can be extended, under some assumptions, to infinite dimensional cases (see e.g. [24] and [25]), but it will not be discussed here in detail.

From JSPS interpretation of the Born rule, we can see that every vector in finite dimensional Hilbert space represents a discrete distribution which is defined by unit norm vector in the same direction. Also, any discrete probability distribution can be represented as a vector in a Hilbert space of proper dimension. The norm of the vector can be interpreted in several ways (we will not discuss it here in details). An interpretation, that is going to be used in the text that follows, is that norm of the vector is related to the probability of the appearance of that vector.

IV. Born Rule in Neural Network Context

In this section, we will give some explanations and definitions that will be used in the following sections. It will be explained here, how some probabilistic concepts can be used in a neural network framework, or how neural networks can be used in a probabilistic framework that is basically based on the Born rule.
The basic single layer feedforward artificial neural network is depicted in Fig. 3. The output of the $n$-th output unit $y_n$ ($n=1, 2, \ldots, N$) of a layer of parallel linear artificial neurons is given as

$$y_n(i) = w_n(i)^T x(i),$$

with $x(i)$ denoting a $K$-dimensional zero-mean input vector of the network and $w_n(i)$ denoting a weight vector of the $n$-th output unit, and $i$ represents sampling instances $iT$, where $T$ is a sampling period. The output vector $y$ is defined as

$$y(i) = W(i)^T x(i). \quad (5)$$

Fig. 3. A layer of parallel linear artificial neurons
In the usual interpretation, based on specific requirements, e.g. minimization of some cost function, matrix $W$ is changed (trained) in the process of learning, according to some adopted learning rule.

Here we will give a slightly different interpretation. Again, we will consider a Hilbert space $H$ of a finite dimension. “State vectors” are defined by the input data vector $x_k$ and we can imagine that every vector $x_k$ is available in a big enough number of copies (clones), so we can perform as many simultaneous measurements as we want. A measurement $M$ with an outcome $w_n$ in some set $W$ is represented by a collection of positive definite matrices $\{m_{wn}\}_{w_n \in W}$ such that $m_{wn} = w_n w_n^T$, so $\sum_{w_n \in W} = W W^T$, which is not necessarily equal to the identity matrix on $H$. This means that the sum of the probabilities of the particular outcomes does not have to be equal to one – in other words, sometimes we will work with improper discrete distributions. Also, measures like entropy and divergence will be applied to improper probability distributions, or to a mixture of proper and improper probability distributions. In the following sections, we will point out that in the adopted framework, this will not affect the final result. Applying measurement $M$ to state $x_k$ produces outcome $w_n$ with the probability (the Born rule)

$$p(w_n \mid x_k) \overset{\text{def}}{=} \cos(w_n, x_k),$$

regardless of the norm of the vectors $w_n$ and $x_k$. In the following text, we will consider only vectors $w_n$ that have unit norms. This means
Also, if we apply $N$ simultaneous measurements $M^N$ to the state $x_k$ we obtain outcome $W$ with the probability

$$p(W | x_k) = \sum_{n=1}^{N} p(w_n | x_k).$$

Here, it is assumed that the outcome of each measurement is different. We define the joint probability of the state $x_k$ and outcome $W$ obtained by simultaneous multiple measurement $M^N$ on state $x_k$, $p(W, x_k)$ as

$$p(W, x_k) = p(W | x_k)p(x_k). \tag{6}$$

Now, without loss of generality, let’s assume that we are dealing with a random variable $x$ that takes realizations from a set of observed $K$-dimensional zero-mean data vectors $\{x_k\}, k \in \{1, \ldots, N_{\text{sample}}\}$, which are sampled from some distribution in time instants $t = kT$ where $k$ is already defined and $T$ represents the sampling period. Then, we can define $p(x=x_k | t=kT)$ as

$$p(x_k) = \frac{\|x_k\|^2}{\sum_{i=1}^{N_{\text{sample}}} \|x_i\|^2}, \tag{7}$$
where $N_{\text{sample}}$ 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 $p(x_k)$ is that it is proportional to $||x_k||^2$. 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 fulfills the two conditions that are required for the probability function $f(z)$ (in our case $p(z)$) to be considered as a modified generalized probability measure [37]:

1. For each $z$, $0 \leq f(z) \leq 1$,

2. $\sum_i f(z_i) = 1$.

In this definition, orthonormality is not explicitly required in order that the coefficients $f(z_i)$ sum up to one. However, from the JSPS introduction, we can see that it is always implicitly present. Also, we know from the definition of density operator, which represents a mixture of the pure states (see Section II), that the trace of the density matrix has to be 1. Here, we will consider all vectors as “oriented energies” or

$$x_k = \frac{x_k}{||x_k||} = x_n x_k,$$
where the norm of the vector $\|x_k\|$, represents the square root of the energy contained in the vector $x_k$, and the orientation represents some unit norm vector $x_n_k$, which represents some pure state. In that case, we can see that the statistical description of our system is represented by the density matrix $\rho$

$$\rho = \sum_k p_k x_n_k x_n_k^T,$$

as a statistical mixture of pure states $x_n_k$, and $p_k = p(x_k)$ are defined by (7). We can see that

$$\rho = \frac{N_{\text{sample}}}{\sum_{i=1}^{N_{\text{sample}}} \|x_i\|^2} C,$$

where $C$ is input signal covariance matrix. Obviously, the matrix $\rho$ and the matrix $C$ have the same eigenvectors. In some sense, proposed approach to input sample probability modeling, 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 [27]. In the Dirichlet process case, we have one parameter $\alpha$ 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 a particle). That means that our input space is initially considered as an augmented space of dimension $N_{\text{sample}}$. 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 \( d \), or some other “deformed” space which we want to analyze. If we want to
calculate “marginal” probability \( p(x=x_s) \) (i.e. probability of observing a certain value \( x \), no matter what the
time instant \( i \) is) we can do it as

\[
p(x = x_s) = \sum_{i=1}^{N_{\text{sample}}} \delta(x_i, x_s) p(x_i), \tag{8}
\]

where \( \delta \) represents the Kronecker \( \delta \) function.

In the proposed context, the learning algorithm applied to the neural network has a basic task to find the
measurement system in which input data is “best explained”, or have the features that are specified. As an
example, principal component analysis will search for the measurement (or we can say coordinate) system
in which the input data covariance matrix is diagonal. Or, using the quantum mechanics terms, we are
trying to find projective measurement which does not increase the von Neumann entropy of our density
matrix [28]. In that case, data that belongs to some low dimensional subspace have sparse representation.

V. PROBABILISTIC PSA

Now, we will give a new definition of probabilistic PSA (having in mind the definitions given in the
previous section):
**Definition 1:** PSA can be defined as a problem of minimization of some symmetric divergence/distance between the probability distribution of the input signal $p(x)$ and the joint probability distribution $p(W, x)$ of the input signal $x$ and the outcome $W$, obtained by the simultaneous multiple measurement $M^N$ on state (input signal) $x$.

In the next subsection we are going to explain one possible solution of the PSA problem, starting from Definition 1 and adopting a particular symmetric divergence measure.

**A. Probabilistic PSA**

Without loss of generality, let’s assume that we are dealing with a set of observed $K$-dimensional zero-mean data vectors $\{x_i\}$, $i \in \{1, \ldots, N_{\text{sample}}\}$ 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 $w_1, w_2, \ldots, w_N$ the unit norm column vectors of matrix $W$ ($N \leq K$). Now, we are going to define probabilistic PSA. In order to do that, we are going to define the symmetric distance, which we will call quadratic variational divergence

$$D_{qV}(p(x) \| p(W, x)) = E\left(\left(p(x) - p(W, x)\right)^2\right) = \sum_i \left(p(x_i) - p(W, x_i)\right)^2,$$

where $E$ represents the empirical expectation operator, and $W$ is a matrix whose columns have unit norms.

This formulation represents only a class of algorithms that do not require the explicit orthonormality of matrix $W$. Explicit orthonormality is sometimes not a desirable constraint, e.g. in biologically plausible
neural networks, since the explicit orthonormality usually requires global computation. Matrix $W$ with unit norm columns can be obtained by local calculations.

We can see that (9) can be written as

$$JS_1(W) = D_{qV}(p(x) \| p(W, x)) = E((p(x) - p(W, x))^2)$$

$$= \frac{1}{N_{\text{sample}}} \left( \sum_{k=1}^{N_{\text{sample}}} x_k^T x_k - \sum_{n=1}^{N_{\text{sample}}} w_n^T x_k w_k \right)^2, (10)$$

Now, we define the probabilistic PSA as a constrained minimization problem

$$\min_{W} (JS_1(W)), \quad (11)$$

under the constraint that $W$ consists of a unit norm columns. So, we are looking for $W$ which minimizes divergence $D_{qV}$. As it is shown later, the solution will be obtained for matrix $W=W^{JS_1}$, which represents the rotated matrix of matrix $U$, which consists of eigenvectors $u_i$ as columns, where $u_i$ represent $N$ dominant eigenvectors of the covariance matrix $C = E(xx^T)$. We can drop the constant that represents the square of overall energy of the input samples, so, the minimization of JS1 is equivalent to the minimization of the following function JS1m

$$JS_1m(W) = \sum_{k=1}^{N_{\text{sample}}} \left( x_k^T x_k - \sum_{n=1}^{N_{\text{sample}}} w_n^T x_k w_n \right)^2 = \sum_{k=1}^{N_{\text{sample}}} \left( x_k^T x_k - y_k^T y_k \right)^2$$
where \( y_k \) is defined by (5). Also, from (9) and (10) we can see that

\[
p(x_k) \propto x_k^T x_k \quad \text{and} \quad p(W, x_k) \propto y_k^T y_k. \quad (12)
\]

In the following text, we will use this fact, so we can make equations simpler. It was shown in [18] that the minimization of function JS1m by a gradient descent method (in the single layer neural network framework) will yield the principal subspace of the input covariance matrix. In [18] that algorithm was named the MH(O) algorithm and can be represented by the following learning rule (\( \delta \) represents the Kronecker \( \delta \) function):

\[
W(i + 1) = W(i) + \gamma(i) \left[ x(i)^T x(i) - y(i)^T y(i) \right] \left[ x(i) y(i)^T - (1 - \delta(K, N)) W(i) \text{diag}(y(i) y(i)^T) \right].
\]

We can use some other symmetric divergence/distance measures and create different types of algorithms. For instance we can use symmetric f-divergences. F-divergences were introduced by Csiszar [11] and Ali and Silvey [2]. Also, some other divergences/distances (or their linear combination) can be used. Some examples are given in the following Table I.
Table I

| Name                     | Equation                                           |
|--------------------------|----------------------------------------------------|
| Variational distance     | $D_V(P \parallel Q) = E[|p - q|]$                 |
| Quadratic variational divergence | $D_{qV}(P \parallel Q) = E[(p - q)^2]$             |
| Jeffrey’s J-divergence   | $D_J(P \parallel Q) = E\left[(p - q)\log\left(\frac{p}{q}\right)\right]$ |
| Jensen-Shannon divergence | $D_{JS}(P \parallel Q) = \frac{1}{2}E\left[p\log\left(\frac{2p}{p+q}\right) + q\log\left(\frac{2q}{p+q}\right)\right]$ |
| Hellinger distance       | $D_H(P \parallel Q) = \frac{1}{2}E\left(\sqrt{p} - \sqrt{q}\right)^2$ |

**Note 1:** A different class of probabilistic PSA algorithms can be obtained by the maximization of $JS_{1d}$ ($W$) defined in the following equation:

$$\max_W JS_{1d}(W) = \max_W E(p(W, x_k))$$

under the constraint that $W$ is orthonormal matrix. In that case the PSA problem is very similar to the clustering problem (see e.g. [21], [38]). This approach was analyzed in [22]. The Subspace Learning Algorithm (SLA) [30], can be seen as a particular realization of that approach.
**Note 2:** Since the definition of \( p(w_n|x_k) \) is based on the Born rule, we can see that any algorithm that is based on gradient minimization/maximization of the function which contains terms \( p(w_n|x_k) \) will result in the learning rule which contains the Hebbian learning rule part.

### B. Probabilistic PCA

In this section, we are going to give a definition of the probabilistic PCA that can be used for creation of symmetric PCA algorithms. There is also another possibility to create asymmetric algorithms as done in [20].

**Definition 2:** PCA can be defined as a problem of minimization of the entropy of the joint probability distribution \( p(W, x_k) \) of the input signal \( x \) and outcome \( W \), obtained by the simultaneous multiple measurement \( M^N \) on state (input signal) \( x \), under the constraint that the matrix \( W \) spans the principal subspace of the input signal covariance matrix \( C = E(xx^T) \). (Definition can be also based on terms like projective measurement and density matrix of input signal, but here we opted for simpler one.)

We can see that the proposed probabilistic definition of PCA, probabilistic PSA represents an integral part and it directly influences the characteristics of probabilistic PCA realizations, like algorithm convergence speed, locality of calculations, preciseness and so on. This probabilistic approach to PCA was recently proposed and analyzed in [22]. Here we will make only a brief review.

**Theorem:** (*Projective measurement cannot decrease entropy [28]*)
Suppose $P_k$ is a complete set of orthogonal projectors and $\rho$ is a density operator. Then the von Neumann entropy of the state $\rho' = \sum_k P_k \rho P_k$ of the system after the measurement is at least as great as the original entropy,

$$S(\rho') \geq S(\rho)$$

with equality if the $\rho = \rho'$ (which is only the case when projections matrices are defined by eigenvectors of $\rho$).

Von Neumann entropy is defined as: $S(\rho) = -\text{tr}(\rho \log \rho) = -\sum_i \lambda_i \log(\lambda_i)$, where $(\lambda_i)$ represent eigenvalues of matrix $\rho$.

Let’s assume that the matrix $W = W^{JS_1}$ is an orthonormal matrix and that it spans the principal subspace. Then, we are going to analyze the following cost function

$$JS_2(W^{JS_1}) = E[S(\rho(W^{JS_1}, x_k))], \quad (13)$$

where $S$ represents the “principal subspace” entropy (see the next paragraph). For instance, we can chose the Shannon entropy $S=S_S$, (although we chose some other definition of entropy, like the Tsallis [36] entropy, or their linear combination) so our cost function is expressed as
Now, the probabilistic PCA can be defined as the following constrained minimization problem

\[
\min_{W^{JS1}} \text{JS}2(W^{JS1}) \quad \text{(15)}
\]

under constraint that \(W^{JS1}\) is orthonormal and spans the principal subspace of matrix \(C\). It can be shown that the solution of this optimization problem will be \(W^{JS1} = UP\), where, as before, \(U\) consists of \(K\) principal eigenvectors \(u_i\) as columns, where \(u_i\) represent principal eigenvectors of the covariance matrix \(C = E(xx^T)\), and \(P\) is a permutation matrix. The proof is straightforward, and it is based on the Section III and the fact that projective measurement can increase but never decrease von Neumann entropy of a mixture state (see section 11.3 in [28], particularly, the equations (11.57) and (11.66)).

We will have to emphasize that, strictly speaking, equation (14) is correct only in the case \(N=K\), since in the case \(N < K\), \(\sum_n p(w_n^{JS1}, x_k) < 1\). In other words, in the case \(N < K\), \(p(w_n^{JS1}, x_k)\) does not represent a proper probability function since it is not normalized. In the case \(N < K\), we will define the term “principal subspace” entropy, \(S^{PS}\). Here, we will analyze only the Shannon entropy case, but something similar can be done for the Tsallis or other types of entropy. We define
The “correct” definition of the Shannon entropy in the principal subspace can be given as

$$S_{S}^{PS}(W_{JS1}) = \sum_{n=1}^{N} p_{n}^{PS}(w_{n}^{JS1}, x_{k}) \log \left( \frac{1}{p_{n}^{PS}(w_{n}^{JS1}, x_{k})} \right). \quad (17)$$

where

$$p_{PS}^{JS1}(w_{n}, x_{k}) = \frac{p(w_{n}^{JS1}, x_{k})}{\sum_{i=1}^{N} p(w_{i}^{JS1}, x_{k})} \frac{p(w_{n}^{JS1}, x_{k})}{V}. \quad (18)$$

The definition of the constant $V$ is obvious from (18), and we have $\sum_{i}^{N} p_{PS}^{JS1}(w_{i}^{JS1}, x_{k}) = 1$. Now, we can write (17) as

$$S_{S}(W_{JS1}) = \log(V) + \frac{1}{V} S^{SP}(W_{JS1}). \quad (19)$$
From (19), it can be seen that the minimization of $S_S(W^{JS_1})$ is equivalent to the minimization of $S^{SP}(W^{JS_1})$, so equation (14) can be formally used even in the cases when $N<K$, since $V$ represents the constant value that is controlled by the PSA part of the algorithm.

In the context of artificial neural network realization, usually we can use every sample only once – which means that we cannot use the successive optimization technique. The solution proposed here is based on a cooperative-competitive concept that is called the time oriented hierarchical method (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 [6, 32], and, also, in the superstatistics area [5].

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 the Lagrangian multiplier idea – a 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 minimization of the following cost function

$$JS(W) = JS_1(W) + \mu JS_2(W),$$  \hspace{1cm} (20)

where $\mu$ is a real number, which is by absolute value smaller than one (since the JS2 should be realized on a slower time-scale). Here, the Lagrangian multiplier does not multiply the constraint part, but the
objective one. It does not affect the solution since it 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 literature [16], [17], [19] and [20]. Without going into the details, in all those approaches the PCA algorithms are obtained from some known PSA algorithms together with the minimization of the Tsallis entropy, $S_T(q \approx 1)$. The value of $\mu$ was selected in all algorithms independently, after the stability analysis for every particular choice of JS1 and JS2.

We will also mention here that with the probabilistic definition given here, PCA can be used in a semi-supervised context, and can be successfully formulated for implementation on parallel computers. This is currently under investigation.
VI. EXPERIMENTAL RESULTS

In this section, we will present some results that illustrate the usefulness of the proposed concept. We are going to present some small scale experimental results that illustrate how the choice of the divergence/distance function influences the convergence speed and preciseness of the algorithm. Then, we will show some experimental results for scale robust PSA algorithm, based on BACH divergence. In the last part of this section, we are going to propose the BACH algorithm that represents modification of the single principal component analyzer, proposed by Oja [29].

A. Small scale simulation results

Now, we will present some small scale simulation results. Here, we will analyze the behavior of several PSA learning algorithms implemented in the feedforward, linear, single layer neural networks with four inputs \((K=4)\). We will consider two cases: first, we will consider the network with four outputs and after that we will consider the network with two outputs \((N=\{2,4\})\). We will assume that input samples are representatives of some set \(\Omega = \{x_1, x_2, \ldots\}\), and probability distributions \(P\) and \(Q\) are defined on \(\Omega\), as \(P = \{p(x) \mid x \in \Omega\}\) and \(Q = \{q(x) \mid x \in \Omega\}\). Let’s assume that \(p^*\) and \(q^*\) are defined as:

\[
p^*(x_k) = x_k^T x_k \propto p(x_k)\quad \text{and} \quad q^*(x_k) = y_k^T y_k \propto p(x_k)p(W \mid x_k),
\]
where $y_k=y(k)$ is defined by equation (5).

We will consider several divergence/distance functions that were given in the Table I, and respective learning algorithms obtained by the gradient descent method presented in Table II.

Table II

| Divergence/ Distance name (Algorithm abbreviation) | Learning algorithm |
|-------------------------------------------------|---------------------|
| Variational distance (A1)                       | $\Delta W \sim (x y^T - (1 - \delta(K, N)) W \text{diag}(y y^T)) \text{sign}(q^* - p^*)$ |
| Quadratic variational divergence (A2)            | $\Delta W \sim (x y^T - (1 - \delta(K, N)) W \text{diag}(y y^T)) (q^* - p^*)$ |
| Jefrrey’s J-divergence (A3)                      | $\Delta W \sim (x y^T - (1 - \delta(K, N)) W \text{diag}(y y^T)) \log \left( \frac{q^* + p^*}{2p^*} \right)$ |
| Jensen-Shannon divergence (A4)                   | $\Delta W \sim (x y^T - (1 - \delta(K, N)) W \text{diag}(y y^T)) \log \left( \sqrt{\frac{q^*}{p^*}} \right)$ |
| Hellinger distance (A5)                          | $\Delta W \sim (x y^T - (1 - \delta(K, N)) W \text{diag}(y y^T)) \log \left( \frac{\sqrt{q^*} - \sqrt{p^*}}{\sqrt{p^*}} \right)$ |

In Table II, $\delta(K, N)$ represent Kronecker $\delta$ and $\Delta W$ represents learning increment of the synaptic weight matrix $W$. Learning methods are obtained by application of gradient descent on a particular divergence/distance measure. We can see that algorithms A1-A5 require only two global calculation circuits for calculation of the input and output energy ($p^*$ and $q^*$).

a) Case $N=4$

In the first set of experiments, we will adopt $N=4$. That means that the number of inputs and outputs is the same. This case is useful, for instance, when the PCA algorithm is used in the prewhitening step of some
ICA algorithm (see e.g. [9]), or for matrix orthonormalization. The input signal in the \(i\)-th iteration \((x(i))\) is defined by the following equation (we use standard MATLAB notation)

\[
a = \text{diag}([1 0.8 0.6 0.4]) \text{rand}(4, 25000); \quad x(i) = 0.5(0.5 + 0.5 \text{rand}(1)) \frac{a(i)}{\text{norm}(a(i))} / pf;
\]

and in that way the energy of the input signal is kept in the range \((1:4, 1:1)\) if \(pf=1\). This range is chosen arbitrarily. Limitation of the input energy is quite natural in technical and biological systems, and here, has the additional purpose to limit learning increments, since several of adopted learning rates depend inversely on the output signal power. A change of the value of \(pf\) influences the energy level. In all experiments the initial value of the synaptic matrix \(W\) was chosen as

\[
W = -0.05 + 0.2 \text{rand}(4,4).
\]

The learning rates \(\gamma\) were constant and chosen to achieve almost maximal convergence speed (learning rates were chosen experimentally – they were gradually increased until the learning algorithms became unstable, and then, the learning rate was reduced to 85% of the value that caused instability). After 12000 iterations, the learning rate was decreased 16 times. In the case of algorithm A1, \(\gamma=l^*8*0.6\), in the case A2 \(\gamma=lf^*8*2.7\), for A3 \(\gamma=lf^*8*0.03\), for A4 \(\gamma = lf^*8*0.8\) and for A5 \(\gamma = lf^*8*0.96\). In the following figures, the behavior of the algorithms is depicted for several values of the parameter \(pf\). The \(pf\) was selected as 1, 2, 3,
4 and 5 respectively ($i=1$). Figs. 4-8 represent the behavior of algorithms, A1-A5. Fig. 9 represents the behavior of the Subspace Learning Algorithm (SLA) [30]. Implementation of the SLA algorithm requires $K$ global calculation circuits (where $K$ is the number of inputs). The error was defined as

$$\text{Error} = \log \left( \text{abs} \left( I - W^T W \right) \right),$$

where $I$ is $N$-dimensional identity matrix.

Fig. 4 Experimental results for (A1)

Fig. 5 Experimental results for (A2)
Fig. 6 Experimental results for (A3)

Fig. 7 Experimental results for (A4)

Fig. 8 Experimental results for (A5)
From the experiments, we can conclude that the SLA algorithm has smoother behavior and better preciseness than all other presented algorithms. However, we can notice that this algorithm is sensitive to the level of input energy, since the convergence speed and preciseness of the algorithm are affected significantly with the change of the input energy level. It seems that algorithms A3-A5 are more robust in that sense. Algorithm A1 is interesting, since its behavior is different from all other algorithms – preciseness is improved with the decrease of the input energy level, while convergence speed is not severely reduced. In the papers that follow, we will show how we can modify algorithm A1 to obtain scaling robust learning algorithm, and achieve good preciseness. It also can be noticed that reduction of the learning rate, after some time, improves the preciseness and stability for all algorithms.

b) Case $N=2$
In the following set of experiments, we will adopt $N=2$. In this case the input signal is compressed. The input signal is defined as in the previous set. Initial learning rates for all algorithms are decreased 8 times.

Variable parameter for experiments is learning rate that is held at the ratios $1:1/2:1/4:1/8:1/16$ ($lf=1$, $lf=1/2$, $lf=1/4$, $lf=1/8$, $lf=1/16$ respectively, and $pf=1$). Again, after 12000 iterations learning rates were decreased 16 times. Figures 10, 11, 12, 13 and 14 depict behavior of the algorithms A1-A5, while Fig. 15 represents behavior of SLA algorithm.

![Figure 10](image.png)

Fig. 10 Experimental results for (A1)
Fig. 11 Experimental results for (A2)

Fig. 12 Experimental results for (A3)
Fig. 13 Experimental results for (A4)

Fig. 14 Experimental results for (A5)
From this set of experiments, we can conclude, again, that the SLA algorithm is the most precise for the same convergence speed, but its convergence speed is significantly affected by the decrease of the learning rate. We can see that the convergence speed of algorithms A1, A2 and A5 is much less affected by the change of the learning rate, comparing to the SLA algorithm. However, they are less precise. On the other side, algorithms A1-A5 are much more suitable for realization in parallel hardware, since they require only 2 global calculation circuits, irrespectively of the number of output neurons.

It seems that with careful choice of the divergence/distance function we can achieve very similar convergence speed and preciseness as in the case of the SLA algorithm, and at the same time, still, have local implementation of the learning rule.
B. **BACH divergence based scale robust PSA learning algorithm**

Now, we are going to introduce the BACH divergence as

$$D_{BACH}(P \| Q) = E\left( \left( p^b - q^b \right)^2 \right),$$

where $b$ is a positive real number (in this section we are going to assume that $b$ is smaller than 1). We can see that the Hellinger distance represents special case where $b=0.5$. Here, we assume $b=0.025$. Then, analyzing neural network in the set-up that is explained in subsection A.a), we will obtain the result depicted in Fig. 16 (after implementation of gradient descent).

![Fig. 16. Experimental results for BACH divergence (4-dimensional case without compression)](image)

We can clearly see that speed and preciseness of the algorithm are almost not affected at all, despite the change of the energy in the input signal in the ratio 1 to 1/25. This could be seen as a good characteristic of the proposed algorithm, comparing it, for instance, to SLA algorithm.
In the two following pictures, we can see the behavior of the algorithm in the cases when the input signal
dimension is 100 (Fig. 17) and 1000 (Fig. 18), while the output dimension is 4. The proposed algorithm,
despite low memory requirements and small number of the global calculation circuits, is useful in the high
dimensional input case.

![Graph](image1.png)

**Fig. 17 Experimental results for the BACH divergence (100-dimensional input, 4-dimensional output)**

![Graph](image2.png)

**Fig. 18 Experimental results for the BACH divergence (1000-dimensional input, 4-dimensional output)**
C. BACH learning algorithm for single principal component extraction

In this section, we will propose a new algorithm for extraction of the principal component. We will call it the BACH algorithm. It is based on minimization of the BACH divergence. For illustration, we will assume that we have 16-dimensional input signal \( x \) sampled from uniform distribution. Our learning rule is given as

\[
\Delta w \sim (xy^T - wy^2)((x^T x)^b - (y^T y)^b)/(y^T y)^{1-b}
\]

where \( b \) is positive real number. Here, we select \( b=0.025 \). Without going into details, we can remove term \((x^T x)^b - (y^T y)^b\), so BACH learning rule can be represented as

\[
\Delta w \sim \frac{(xy^T - wy^2)}{(y^T y)^{1-b}},
\]

We can see that it represents modification of the famous Oja’s learning rule [29], which is, as we are going to see from the illustration example, robust to change of the input signal energy level, which significantly simplifies selection of the learning rate. In the Fig. 19, we can see behavior of the BACH (solid line) and Oja’s algorithm for the “nominal” energy level.
Fig. 19 Experimental results for BACH PCA algorithm (16-dimensional input, 4-dimensional output)

We can see that both algorithms behave similarly. However, if we decrease the input energy 100 times, and keeps learning rates unchanged, we can see (Fig. 20) that convergence speed of the BACH algorithm is only slightly affected by input signal energy change, while Oja’s algorithm becomes very slow. Although it is not going to be presented here, it can be said that BACH algorithm will not be significantly affected, even in the case when the input energy is decreased by a several order of magnitudes (e.g. billion times).
VII. CONCLUSION

In this paper, we proposed a probabilistic principal subspace analysis (starting from MHO algorithm), as well as brief introduction of probabilistic principal component analysis, both based on the Born rule. Here we only considered derivation of symmetric algorithms. A new simple geometrical representation of the Born rule, named JoyStick Probability Selector was presented, too. 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, selecting different divergence/distance functions, it is possible to create a big number of PSA algorithms. 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. By selection of different PSA algorithms and different entropy functions, it is possible to create an even bigger number of algorithms that can be used for PCA, and that have some specific features. With this, probabilistic definition, PCA can be used in semi-supervised context and can be successfully implemented in parallel computation machines. This is currently under investigation.

From the presented experimental results, we can see that it is possible to create the “optimal” divergence/distance function, which can result in good algorithm characteristics (e.g. convergence speed, preciseness, scaling robustness) and still have a realization which has only two (or less) global calculation circuits. Like illustration, we presented input signal scale robust principal subspace algorithm, and BACH algorithm. A small number of global calculation circuits is a very useful characteristic of the proposed algorithms for successful implementation in parallel hardware, or for simple implementation on GPU platforms.

The proposed approach can be easily modified to give the definition of the probabilistic minor subspace analysis (MSA) and the minor component analysis (MCA).

It is possible to show that a similar way of reasoning can lead to a new probabilistic definition of the independent component analysis (ICA). By doing this, it is possible to give different interpretations of some existing ICA solutions and to propose new ones. One possible way to do it has been proposed recently in [33].
REFERENCES

[1] J-H. Ahn, J-H. Oh and S. Choi. Learning principal directions: integrated-square-error minimization.

    *Neurocomputing*, 70: 1372-1381, 2007.

[2] S.M. Ali and S.D. Silvey. A general class of coefficients of divergence of the distribution from another.

    *J. Roy. Statistics Soc.* Ser B, 28: 131-142, 1966.

[3] C. Archambeau, N. Delanney and M. Verleysen. Robust probabilistic projection. In: Proc. 23rd

    *International Conference on Machine Learning*, Pittsburg, USA, 2006.

[4] P. Baldi and K. Hornik. Learning in linear neural networks: A survey. *IEEE Trans. Neural Networks*,

    6: 837-858, 1995.

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

[6] C.M. Bishop. Pattern Recognition and Machine Learning, Springer, 2006.

[7] T. Chen, E. Martin and G. Montague. Robust probabilistic PCA with missing data and contribution

    analysis fro outlier detection. *Computational Statistics and Data Analysis*, 53: 3706-3716, 2009.

[8] T.-P. Chen and S.-I Amari. Unified stabilization approach to principal and minor components. *Neural

    Networks*, 14: 1377-1387, 2001.

[9] A. Cichocki and S.-I. Amari. Adaptive Blind Signal and Image Processing – Learning Algorithms and

    Applications. John Wiley and Sons, 2003.

[10]M. Collins, S. Dasgupta, and R.E. Schapire. A generalization of principal component analysis to the

    exponential family. *Advances in Neural Information Processing Systems*, 14:617-624, 2002.
[11] I. Csiszár. A note on Jensen’s inequality. Studia. Sci. Math. Hungar., 1:185-188, 1966.

[12] Y. Fang and M.K. Jeong. Robust probabilistic multivariate calibration model. Technometrics, 50: 305-316, 2008.

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

[14] A. M. Gleason. Measure on the closed subspaces of a Hilbert space. J. Math. & Mechanics, 6: 885-893, 1957.

[15] H. Hotteling. Analysis of a complex of statistical variables into principal components. Journal of Educational Psychology, 24: 417-441, 1933.

[16] M. Jankovic and H. Ogawa. Time-Orinated hierarachical method for computation of principal components using subspace learning algorithm, Int. J. Neural Systems, 14: 313-324, 2004.

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

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

[19] M. Jankovic and B. Reljin. General stochastic approximation and neural learning on principal Stiefel submanifold”, WSEAS Transactions on Information Science and Applications, 3: 1195-1201, 2006.

[20] M. Jankovic and M. Sugiyama. A multipurpose linear component analysis method based on modulated Hebb Oja learning rule. IEEE Signal Processing Letters, 15: 677-680, 2008.
[21] M. Jankovic and M. Sugiyama. Tensor based image segmentation. *Ninth Symposium on Neural Network Applications in Electrical Engineering (NEUREL 2008),* 67-73, 2008.

[22] M. Jankovic and M. Sugiyama. Probabilistic principal component analysis based on joystick probability selector. *International Joint Conference on Neural Networks (IJCNN 2009)*, 1414-1421, 2009.

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

[24] R. V. Kadison. The Pythagorean Theorem: I. The finite case. *Proceedings of the National Academy of Sciences the United States of America*, vol. 99., no. 7, pp. 4178-4184, 2002.

[25] R. V. Kadison. The Pythagorean Theorem: II. The infinite discrete case. *Proceedings of the National Academy of Sciences the United States of America*, vol. 99., no. 8, pp. 5217-5222, 2002.

[26] 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*, 8: 486-504, 1997.

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

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

[29] E. Oja. A simplified neuron model as a principal component analyzer. *J. Math. Biol.*, 15:267-273, 1982.

[30] E. Oja. Neural networks, principal components, and subspaces. *Int. J. Neural Syst.*, 1: 61–68, 1989.
[31] K. Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2:559-572, 1901.

[32] J.C. Sexton and D.H. Weingarten. Hamiltonian evolution for the hybrid Monte Carlo algorithm, *Nuclear Physics B*, 380: 665-677, 1992.

[33] T. Suzuki and M. Sugiyama. Estimating squared-loss mutual information for independent component analysis. *Independent Component Analysis and Signal Separation, Lecture Notes in Computer Science*, 5441: 130-137, Berlin, Springer, 2009.

[34] M. Tipping and C.M. Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society, Series B*, 61, part 3: 611-622, 1999.

[35] M. Tipping and C.M. Bishop. Mixtures of probabilistic principal component analysers. *Neural Computation*, 11: 443-482, 1999.

[36] C. Tsallis. Possible generalization of Boltzmann-Gibbs statistics. *J. Statistical Physics*, 52: 479-478, 1988.

[37] M.K. Warmuth and D. Kuzmin. Bayesian generalized probability calculus for density matrices. *To appear in Machine Learning*, 2009.

[38] L. Wolf. Learning using the Born rule. Technical report *MIT-CSAIL- TR-2006-036*, 2006.

[39] W. K. Wootters. Statistical distance and Hilbert space. *Physical Review D.*, 23:357-362, 1981.
