Coupled Principal Component Analysis
Ralf Möller, Axel Könies

Abstract—A framework for a class of coupled principal component learning rules is presented. In coupled rules, eigenvectors and eigenvalues of a covariance matrix are simultaneously estimated in coupled equations. Coupled rules can mitigate the stability-speed problem affecting non-coupled learning rules, since the convergence speed in all eigendirections of the Jacobian becomes widely independent of the eigenvalues of the covariance matrix. A number of coupled learning rule systems for principal component analysis, two of them new, is derived by applying Newton’s method to an information criterion. The relation to other systems of this class, the adaptive learning algorithm (ALA), the robust recursive least squares algorithm (RRLSA), and a rule with explicit renormalization of the weight vector length, are established.

Index Terms—Principal component analysis, minor component analysis, neural networks, speed-stability trade-off, Newton’s method, Oja’s rule, adaptive learning algorithm, robust recursive least squares learning algorithm

I. INTRODUCTION

Principal component analysis (PCA) of high-dimensional data is an ingredient of many signal processing applications. PCA strives to extract the “principal” directions in the data space where the variance of the data is maximal, thus paving the way for dimension reduction and data compression. Other applications rely on minor component analysis (MCA) which finds the directions of minimal variance. Neural network approaches to PCA (MCA) pursue an “online” approach where an estimate of the principal (minor) directions is updated after each presentation of a data point. Therefore these methods are especially suited for high-dimensional data, since the computation of the large covariance matrix can be avoided, and for the tracking of non-stationary data, where the covariance matrix slowly changes over time.

Many neural principal component analyzers have been suggested in the literature, with important initial contributions by Oja [1] and Sanger [2]; an overview can be found in [3]. The number of neural networks for minor component analysis is somewhat smaller; a recent review is given in [4]. Although the field has been active for two decades now, the attempts to improve the methods and suggest new approaches and information criteria are continuing.

For a single PCA (MCA) neuron, the vector of synaptic weights is modified by a learning rule in a way that it converges towards that eigenvector of the covariance matrix of the data distribution which has the largest (smallest) corresponding eigenvalue; this eigenvector has the desired principal (minor) direction. With multiple PCA (MCA) neurons, the weight vectors should converge to the orthogonal set of eigenvectors of the covariance matrix with maximal (minimal) eigenvalues. Since generally only the direction of eigenvectors is defined but not their length, neural PCA/MCA rules often constrain the length of the weight vectors, some by converging towards a defined length (usually unit length [1] or a length related to the eigenvalue [5]), others by keeping the weight vector length constant over time [4].

In this paper, we provide a framework for a special class of learning rules where eigenvectors and eigenvalues are simultaneously estimated in coupled update equations. Most previously suggested rules did not consider eigenvalue estimates in the update equations of the weights, an exception being attempts to control the learning rate based on the eigenvalue estimates [6]. We prove that coupled learning rules are a solution for the speed-stability problem that plagues most non-coupled learning rules. The speed of convergence of a system depends on the eigenvalues of its Jacobian. In non-coupled PCA/MCA rules, these vary with the eigenvalues of the covariance matrix.

We show that, in non-coupled PCA rules, the eigen-motion in all directions mainly depends on the principal eigenvalue of the covariance matrix. Numerical stability and fast convergence can only be achieved by guessing this eigenvalue in advance. Especially for chains of principal component analyzers which simultaneously estimate the first few principal eigenvectors [2], choosing the right learning rates for all stages may prove difficult: For a single principal component analyzer, the normalization of the data or the adjustment of the learning rate to the maximal variance over all directions may be sufficient to achieve fast convergence. The same holds for chains that are trained sequentially, stage-by-stage, since the learning rate of a stage can be adjusted to the deflated data set at its input. However, if chains of principal component analyzers are trained simultaneously (this being a case where neural network models can unfold their advantages), the second and all following stages would require previous knowledge about the eigenvalues.

The problem is even more severe for MCA rules. These rules exhibit a wide range of convergence speeds in the different eigendirections, since the eigenvalues of the Jacobian cover approximately the same (relative) range as the eigenvalues of the covariance matrix. With learning rates small enough to still guarantee the stability of the numerical procedure, non-coupled MCA rules may converge only slowly.

Coupled learning rules can be derived by applying Newton’s method to a common information criterion. A Newton descent yields learning rules with approximately equal convergence speed in all eigendirections of the system. Moreover, all eigenvalues of the Jacobian of such a system are approximately...
-1, thus the dependence on the eigenvalues of the covariance matrix can be eliminated. We show that with respect to averaged differential equations (with right-hand sides depending on the covariance matrix), this approach solves the speed-stability problem for both PCA and MCA rules. However, these differential equations can only be turned into the above-mentioned online rules for the PCA but not for the MCA case, leaving the more severe MCA stability problem still unresolved.

In section II, we derive an averaged PCA learning rule and an averaged MCA learning rule by applying Newton’s method to a common information criterion. The analysis of the eigen-motion of the two averaged systems of differential equations in section III reveals that their speed of convergence is optimal. For comparison, we also analyze the eigen-motion of two instances of non-coupled learning rules. In section IV it is shown that the online version of the PCA learning rule could in principle show fluctuations in the length of the eigenvector estimate; however, our simulations show that this is not the case.

Another approximation (introduced in section V) yields a PCA learning rule that is still close to the optimum with respect to the speed of convergence, but at the same time avoids fluctuations in the weight vector length; this rule is identical to the “adaptive learning algorithm for principal component analysis” (ALA) [6]. The MCA rule derived from the Newton approach is analyzed in section VI. We discuss why coupled learning equations are only beneficial for PCA, but not for MCA rules, and explain why complementing a non-coupled MCA rule with learning rate control in the style of ALA cannot improve convergence speed.

In section VII we show that the presented framework covers a number of other learning rules. Further approximation from ALA leads to another, novel PCA rule which, however, is prone to fluctuations in the weight vector length (which was confirmed in simulations). A discrete learning rule with explicit weight vector normalization can be derived from this rule which mends the problem of length fluctuations. We establish relations between the rule with explicit normalization, ALA, and the “robust recursive least squares learning algorithm” (RRLSA) [5], a rule derived from a recursive least squares framework. All rules have been tested for synthetic three-dimensional distributions and for 64-dimensional distributions obtained from natural image data; in addition, we apply one of the rules to very high-dimensional image data. Since ALA, RRLSA, and the rule with explicit normalization have been investigated and partially compared in simulations before, we do only present selected simulation results in section VIII. The theoretical insights gained are summarized and discussed in section IX.

II. DERIVATION OF COUPLED LEARNING RULE SYSTEMS

Learning rules for principal component analysis are often derived by optimization of some information criterion, e.g. by maximization of the variance of the projected data or by minimization of the reconstruction error (overviews: [3], [7, ch.6]). However, as pointed out in [7, p.137], any criterion may be used in which the maximum (possibly under a constraint) coincides with the desired principal directions or principal subspace. The freedom of choosing an information criterion is even greater if Newton’s method is applied: in this case, it suffices to find a criterion where the stationary points coincide with the desired solutions. As shown in appendix I, all stationary points become attractors in a Newton descent.

The information criterion used as the starting point for our analysis is

\[ p = w^T C w \lambda^{-1} - w^T w + \ln \lambda. \]  

(1)

Here \( w \) denotes the \( n \)-dimensional weight vector, i.e. the estimate of the eigenvector, \( \lambda \) the eigenvalue estimate, and \( C = E \{ x x^T \} \) the \( n \times n \) covariance matrix of the data. This criterion is related to the normalized Mahalanobis measure, which is the negative logarithm of the best-fitting multivariate Gaussian for a distribution with the covariance matrix \( C \) [16, sec. 4.2]; (1) is a special case with only one eigenvector. We see that

\[ \frac{\partial p}{\partial w} = 2Cw\lambda^{-1} - 2w \]  

(2)

\[ \frac{\partial p}{\partial \lambda} = -w^T Cw \lambda^{-2} + \lambda^{-1}, \]  

(3)

so that the stationary points \((\hat{w}, \hat{\lambda})\) are defined by

\[ C\hat{w} = \hat{w} \hat{\lambda} \]  

(4)

\[ w^T \hat{w} = \hat{\lambda}, \]  

(5)

from which we also conclude that \( w^T w = 1 \). Thus the criterion (1) fulfills the above-mentioned requirement: the stationary points include all associated eigenvectors and eigenvalues of \( C \). The Hessian of the criterion is

\[ H(w, \lambda) = 2 \begin{pmatrix} C\lambda^{-1} - I & -Cw\lambda^{-2} \\ -w^T C\lambda^{-2} & w^T Cw \lambda^{-3} \end{pmatrix}. \]  

(6)

Newton’s method as given in equation (55) requires an inversion of the Hessian. In appendix II, approximations of the inverted Hessian are determined in the vicinity of two different stationary points \((w_1, \lambda_1)\), \((w_j, \lambda_j)\), the first being the principal solution (PCA), where \( \lambda \approx \lambda_1 \gg \lambda_2, \ldots, \lambda_n \), the second being the minor solution (MCA), where \( \lambda \approx \lambda_1 \ll \lambda_2, \ldots, \lambda_n \):

\[ H_{-1}^{-1} \approx \frac{1}{2} \begin{pmatrix} C^{-1} - \frac{3}{2} w^T \lambda - w^T w & -w \lambda \\ -w^T \lambda & 0 \end{pmatrix}. \]  

(7)

With the gradient from (2, 3) and a Newton descent according to (55), we get one coupled system of differential equations for the PCA case

\[ \dot{w} = Cw\lambda^{-1} - w^T Cw \lambda^{-1} - \frac{1}{2} w(1 - w^T w) \]  

(9)

\[ \dot{\lambda} = w^T Cw - w^T w \lambda, \]  

(10)

and another for the MCA case

\[ \dot{w} = C^{-1} w\lambda + w^T Cw \lambda^{-1} - \frac{1}{2} w(1 + 3w^T w) \]  

(11)

\[ \dot{\lambda} = w^T Cw - w^T w \lambda. \]  

(12)

In this paper, we will relate to (9, 10) and its online version as “nPCA”, and to (11, 12) as “nMCA”.

\[ \end{raw}]
III. ANALYSIS OF EIGEN-MOTION

A. Coupled PCA system

It is straightforward to show that the stationary points of nPCA (9, 10) coincide with those of the criterion (1). Stability is determined by computing the Jacobian

$$J(w, \lambda) = \begin{pmatrix} \frac{\partial \mathcal{J}}{\partial w} & \frac{\partial \mathcal{J}}{\partial \lambda} \end{pmatrix}$$

for a stationary point \((w_1, \lambda_1)\):

$$J(w_1, \lambda_1) = \begin{pmatrix} \mu \lambda_1^{-1} - \mathbf{w}_1^{T} \mathbf{w}_1 & 0 \\ 0 & -1 \end{pmatrix}.$$  \hspace{1cm} (13)

By orthogonal transformation with \(U\) from (58) we obtain a transformed Jacobian \(\mathbf{J}' = U^{T}JU\) with the same eigenvalues as \(\mathbf{J}\)

$$\mathbf{J}'(w_1, \lambda_1) = \begin{pmatrix} \tilde{\lambda} \lambda_1^{-1} - \mathbf{w}_1^{T} \mathbf{w}_1 & 0 \\ 0 & -1 \end{pmatrix}.$$  \hspace{1cm} (14)

Here, \(\tilde{\lambda}\) is a \(n \times n\) diagonal matrix containing all eigenvalues of \(\mathcal{C}\). The eigenvalues \(\alpha\) determined from\( \det(\mathbf{J}' - \alpha \mathbf{I}) = 0\) are

$$\alpha_1 = \alpha_{n+1} = -1, \alpha_j = \frac{\lambda_j}{\lambda_1} - 1, j = 2, \ldots, n.$$  \hspace{1cm} (16)

Since stability requires \(\alpha < 0\) and thus \(\lambda_j < \lambda_1, j = 2, \ldots, n\), we find that only principal eigenvector-eigenvalue pairs are stable stationary points, while all other stationary points are saddles or repellers. Moreover we see that, if we assume \(\lambda_j < \lambda_1\), all eigenvalues are \(\alpha \approx -1\), so the system converges with approximately equal speed in all its eigendirections, and this speed is widely independent of the eigenvalues \(\lambda_j\) of the covariance matrix.

B. Coupled MCA system

Also in the nMCA system (11, 12), the stationary points are the same as for the criterion (1). As above we analyze the stability by finding the eigenvalues of

$$\mathbf{J}'(w_1, \lambda_1) = \begin{pmatrix} \tilde{\lambda} \lambda_1^{-1} - \mathbf{w}_1^{T} \mathbf{w}_1 & 0 \\ 0 & -1 \end{pmatrix},$$

which are

$$\alpha_1 = \alpha_{n+1} = -1, \alpha_j = \frac{\lambda_j}{\lambda_1} - 1, j = 2, \ldots, n.$$  \hspace{1cm} (18)

We obtain an MCA condition \(\lambda_1 < \lambda_j, j = 2, \ldots, n\), and see that for \(\lambda_1 \ll \lambda_j\) all eigenvalues are \(\alpha \approx -1\), thus the convergence speed is again about equal in all eigendirections and independent of the eigenvalues of \(\mathcal{C}\).

C. Non-coupled Systems

To pinpoint the difference between the eigen-motion in coupled and in non-coupled systems, we analyze two instances of non-coupled rules, Oja’s PCA learning rule [1], in averaged form

$$\dot{\mathbf{w}} = \mathbf{C}^{T}\mathbf{w} - \mathbf{w}\mathbf{C}^{T}\mathbf{w},$$

and the MCA rule suggested by Feng et al. [8], in averaged form

$$\dot{\mathbf{w}} = -\mu(\mathbf{C}^{T}\mathbf{w})\mathbf{C}^{T}\mathbf{w} + \mathbf{w}.$$  \hspace{1cm} (20)

The PCA rule (19) has stationary points defined by \(\mathbf{C}^{T}\mathbf{w}_1 = \mathbf{w}_1\lambda_1, \mathbf{w}_1^{T}\mathbf{w}_1 = 1\). The Jacobian in the stationary points is \(\mathbf{J}(\mathbf{w}_1) = \frac{\partial \mathcal{J}}{\partial \mathbf{w}_1}\mathbf{C} = \lambda_1 - 2\mathbf{w}_1^{T}\mathbf{w}_1\); \(\lambda_1\). The eigenvalues of the transformed Jacobian \(\mathbf{J}'(\mathbf{w}_1) = \mathbf{I} - \lambda_1 - 2\mathbf{e}_1\mathbf{e}_1^{T}\lambda_1\) are

$$\alpha_1 = -2\lambda_1, \alpha_j = \lambda_j - \lambda_1 \approx -\lambda_1, j = 2, \ldots, n.$$  \hspace{1cm} (21)

Obviously, the speed of convergence depends on the principal eigenvalue \(\lambda_1\), with \(\lambda_1 > \lambda_j, j = 2, \ldots, n\). If a numerical procedure (like Euler’s method) is applied for the solution, the learning rate (step size) has to be chosen in accordance with an estimate of \(\lambda_1\).

For the MCA rule (20), we have again stationary points in the eigenvectors of \(\mathcal{C}\), but here \(\mathbf{w}_1^{T}\mathbf{w}_1 = \lambda_1^{-1}\). The eigenvalues in the stationary points is \(\mathbf{J}(\mathbf{w}_1) = -\mathbf{C}^{T}\mathbf{w}_1 - \lambda_1\); \(\lambda_1\). The eigenvalues of the transformed Jacobian \(\mathbf{J}'(\mathbf{w}_1) = -\mu\lambda_1^{-1} + 2\mathbf{e}_1\mathbf{e}_1^{T}\lambda_1\) are

$$\alpha_1 = -2, \alpha_j = -\frac{\lambda_j}{\lambda_1} + 1 \approx -\frac{\lambda_j}{\lambda_1}, j = 2, \ldots, n.$$  \hspace{1cm} (22)

Here we see that the speed-stability problem is even more severe than in the PCA rule. The speed of convergence does not only depend on the minor eigenvalue \(\lambda_1\), with \(\lambda_1 \ll \lambda_j, j = 2, \ldots, n\), but also on all other eigenvalues \(\lambda_j, j = 2, \ldots, n\) of the covariance matrix. If these extend over a large interval, no suitable learning rate may be found for a numerical solution that still guarantees stability (upper limit defined by the largest eigenvalue) but also ensures a sufficient speed of convergence in all eigendirections.

IV. ONLINE PCA SYSTEM

We first focus on the nPCA rule, and return to the nMCA rule in section VI. The averaged differential equations of nPCA (9, 10) can be turned into an online learning rule by informally approximating \(\mathcal{C} \approx \mathbf{x}\mathbf{x}^{T}\), where \(\mathbf{x}\) is a data vector drawn from the distribution. According to stochastic approximation theory, the resulting stochastic differential equation has the same convergence goal as the deterministic averaged equation if certain conditions are fulfilled, the most important being a learning rate \(\gamma\) that decreases to zero over time (see e.g. [9] or textbooks [3, 7]). The online rule can be understood as a learning rule for the weight vector \(\mathbf{w}\) of a linear neuron which computes its output \(y\) from the scalar product of weight vector and input vector, \(y = \mathbf{w}^{T}\mathbf{x}\). We obtain

$$\dot{w} = \gamma(y\lambda^{-1}(\mathbf{x} - \mathbf{w}y) - \frac{1}{2} w(1 - w^{T}\mathbf{w}))$$  \hspace{1cm} (23)

$$\dot{\lambda} = \gamma(y^{2} - \mathbf{w}^{T}\mathbf{w}).$$  \hspace{1cm} (24)

Critical for the behavior of the online rule is the temporal derivative of the (squared) weight vector length, \(\dot{\mathbf{w}}^{T}\mathbf{w}/dt = \mathbf{w}^{T}\dot{\mathbf{w}}\); see the discussion in [4]. Here we obtain

$$\dot{\mathbf{w}}^{T}\mathbf{w} = \gamma(y^{2}\lambda^{-1} - \frac{1}{2} w^{T}(1 - w^{T}\mathbf{w})).$$  \hspace{1cm} (25)
We see that the sign of this expression does not only depend on $w^T w$, but also on the random variable $y$, therefore the weight vector length may in principle be fluctuating and only converging when the learning rate $\gamma$ becomes sufficiently small. For comparison, Oja’s rule (19) yields

$$w^T \dot{w} = \gamma y^2 (1 - w^T w), \quad (26)$$

thus every deviation of $w^T w$ from unity is immediately compensated, and only the absolute value of the temporal derivative depends on $y$, but not its sign. Fluctuations are therefore avoided (in a numerical procedure at least to the first order in $\gamma$, see [4]). Simulations show, however, that fluctuations in nPCA (23, 24) are actually small; see section VIII. This may be due to the fact that the first factor is on average positive with higher probability than negative, so that the second factor often determines the sign of the product. A further approximation, however, produces an online system for which fluctuations in the weight vector length can also be analytically excluded, as will be shown in the next section.

V. ANOTHER PCA SYSTEM: ALA

If we further approximate $w^T w \approx 1$ (which is fulfilled in the vicinity of the stationary points) in the averaged system nPCA (9, 10), we obtain the system

$$\dot{w} = Cw\lambda^{-1} - w w^T Cw\lambda^{-1} \quad (27)$$
$$\dot{\lambda} = w^T Cw - \lambda. \quad (28)$$

This learning rule system is known as ALA [6]. It has the stationary points defined by $Cw = w_1 \lambda_1$, with $w_1^T w_1 = 1$. The eigenvalues of the system’s Jacobian,

$$\alpha_1 = -2, \quad \alpha_{n+1} = -1, \quad \alpha_j = \frac{\lambda_j}{\lambda_1} - 1, \quad j = 2, \ldots, n, \quad (29)$$

do slightly deviate from the close-to-optimal solution found in (9, 10), but are still approximately equal and widely independent of the eigenvalues of the covariance matrix. The corresponding online system is given by

$$\dot{w} = \gamma y \lambda^{-1} (x - wy) \quad (30)$$
$$\dot{\lambda} = \gamma (g^2 - \lambda). \quad (31)$$

We see that ALA can be interpreted as an instance of Oja’s PCA rule (19), with a learning rate scaled by the current eigenvalue estimate $\lambda^{-1}$. Accordingly, we obtain

$$w^T \dot{w} = \gamma y^2 \lambda^{-1} (1 - w^T w), \quad (32)$$

i.e., this rule shares the beneficial property of a non-fluctuating weight vector length with Oja’s rule; compare with equation (26) and (25). In simulations we find only small differences in the overall behavior of ALA and nPCA (23, 24); see section VIII.

It is known [6] that in ALA the weight vector length may diverge if the eigenvalue estimate is too small. The same holds for the nPCA system (23, 24). ALA therefore contains a renormalization step for the weight vector length. We found that choosing the learning rate somewhat smaller and starting with higher initial eigenvalue estimates usually avoided divergence in our tests.

VI. ONLINE MCA SYSTEM

Newton’s method leads to the nMCA system (11, 12). It was shown in section III to have a Jacobian with eigenvalues that are equal and widely independent of the eigenvalues of the covariance matrix, and thus appears to be a solution for the speed-stability problem discussed in section I. However, when attempting to turn this system into an online rule, we encounter the problem to replace the inverse covariance matrix $C^{-1}$ by an expression containing the input vector $x$. An averaged equation depending on $C$ has the form $\dot{w} = f(C, w) = f(E\{xx^T\}, w) = E\{f(xx^T, w)\}$. In an online rule, the expectation of the gradient is approximated by slowly following $\dot{w} = \gamma f(xx^T, w)$ for subsequent observations of $x$. This transition is obviously not possible if the equation contains $C^{-1}$. Thus there is no online version of the MCA system (11, 12).

Nevertheless, also for MCA rules we could pursue the approach to normalize the learning rate to an estimate of the eigenvalue, as it was suggested for the ALA PCA rule (30, 31). Assume that we have a non-coupled MCA rule with eigenvalues of the Jacobian

$$\alpha_j = \lambda_1 - \lambda_j \approx -\lambda_j, \quad j = 2, \ldots, n, \quad (33)$$

which have the opposite sign to the eigenvalues of Oja’s PCA rule; see equation (21). This would, for example, be the case for the AMEX MCA rule [10]. Normalizing the learning rate would turn the eigenvalues into

$$\alpha_j = 1 - \frac{\lambda_j}{\lambda_1} \approx \frac{\lambda_j}{\lambda_1}, \quad j = 2, \ldots, n. \quad (34)$$

Despite this ALA-style normalization, the speed of convergence in the different eigendirections still depends on the entire range of eigenvalues of the covariance matrix, so that the speed-stability problem is not mended by this modification. Instead, it would be necessary to normalize by $\lambda_j$ instead of $\lambda_1$. This is actually the role played by the inverse covariance matrix $C^{-1}$ in (11), but it prevents the derivation of an online rule.

VII. RELATION TO OTHER PCA RULES

A. Further Approximation of ALA

By one further step of approximation from ALA we arrive at an interesting coupled PCA rule which, unlike ALA, has no counterpart in a non-coupled rule. If in (27) we approximate $w^T Cw \approx \lambda$ (fulfilled in the vicinity of the stationary points), we obtain the averaged system

$$\dot{w} = Cw\lambda^{-1} - w \quad (35)$$
$$\dot{\lambda} = w^T Cw - \lambda, \quad (36)$$

in the following called “cPCA”. Note that this system is closely related to the definition of eigenvectors and eigenvalues $Cw = \lambda w$. Obviously the stationary points $(w_1, \lambda_1)$ are given by $Cw_1 = w_1 \lambda_1$ and $w_1^T w_1 = 1$. The transformed Jacobian

$$J^*(w_1, \lambda_1) = \left( \begin{array}{c} \Lambda \lambda_1^{-1} - I \\ 2e_i^T \lambda_1 \end{array} \right) \left( \begin{array}{c} -\epsilon_1 \lambda_1^{-1} \\ -1 \end{array} \right). \quad (37)$$
has the eigenvalues
\[ \alpha_{j,n} = -\frac{1}{2} \pm \frac{1}{2} \sqrt{\lambda_j^2 - 1}, \quad j = 2, \ldots, n. \] (38)

The system is a stable principal component analyzer, but two of the eigendirections have imaginary eigenvalues, indicating oscillations. The corresponding online system of cPCA is given by
\[ \dot{\mathbf{w}} = \gamma (\mathbf{x}_t \lambda^{-1} - \mathbf{w}) \]
(39)
The temporal derivative of the weight vector length is
\[ \mathbf{w}^T \dot{\mathbf{w}} = \gamma (y^2 - \lambda). \]
(40)
indicating that the weight vector length fluctuates. Stability and fluctuations are confirmed by simulations; see section VIII.

C. Relation to RRLSA

The unnormalized weight vector \( \tilde{\mathbf{w}}_t \) incorporates both the eigenvector estimate (in its direction) and the eigenvalue estimate (in its length), and thus fuses (42) and (43) into the single update equation (49). This equation is closely related to RRLSA, where the analogous equation is
\[ \dot{\mathbf{w}}_{t+1} = \beta_t \mathbf{w}_t + \mathbf{x}_t y_t \]
(51)
(notation different from [5]). Equations (49) and (51) are exchangeable for \( \mathbf{w}_t = \tilde{\mathbf{w}}_t / \gamma_t \) and \( \beta_t = 1 - \gamma_t \) (with \( \gamma_t \) and \( \beta_t \) constant). Note that in both coupled PCA with explicit normalization (42) and in RRLSA (51), a normalization step is required for the computation of \( y_t \) (50).

The RRLSA learning rule can alternatively be derived by multiplying the left side of the weight update equation of ALA (44) with \( \lambda_{t+1} \), and the right side of (44) with the identical expression obtained from the right side of (45); second-order terms of \( \gamma_t \) are omitted and (50) establishes the relation to the normalized weight vector \( \mathbf{w}_t \).

VIII. SIMULATIONS

A. Parameter Adjustment of the Coupled Rules

For the coupled learning rules, an exponential decay of the learning rate from \( \gamma_0 = 0.01 \) to \( \gamma_{\text{max}} = 0.0001 \) was experimentally found to be appropriate to ensure both fast convergence and stability. The training time \( t_{\text{max}} \) influences the precision of the final eigenvectors — if \( t_{\text{max}} \) is too small, it is possible that the eigenvectors did not fully converge and are still related to their initial values in the end. Moreover, with large data sets, the training time should be chosen high enough so that each training sample is likely to be selected multiple times in the training process. The initial eigenvalue estimates \( \lambda_0 \) should be chosen above the range of expected eigenvalues; if they are too small, the coupled methods may diverge.

B. Low-Dimensional Synthetic Data

We first test the different learning rules (in their online forms with Euler’s method applied) with low-dimensional, synthetic data. In the three-dimensional space \( (n = 3) \), 1000 data points were randomly generated from a uniform distribution inside a cuboid of size \( 500 \times 100 \times 200 \) which was rotated around the \( x, y, \) and \( z \) axis by angles of \( 20^\circ, 10^\circ, \) and \( 15^\circ \), in that order. The distribution’s covariance matrix has the eigenvalues 20452, 3173, 863. We used \( t_{\text{max}} = 10000 \) training steps, an initial eigenvalue estimate of \( \lambda_0 = 1000 \), random orthonormal
initial weight vectors, and an exponential decay of the learning rate from $\gamma_0 = 0.01$ to $\gamma_{t_{\text{max}}} = 0.0001$. The same distribution, same initialization, and same presentation order were used for all experiments.

Figure 1 shows the time course of the projection of the eigenvector estimate (weight vector) onto the real principal eigenvector of the distribution. We see that in all rules the projection converges towards unity, thus the weight vector aligns with the real eigenvector. An example was chosen where the initial projection was close to zero, so the weight vector was initially pointing in a roughly perpendicular direction. The diagram shows a similar time course for nPCA and ALA, as well as for rPCA and RRLSA; in all these rules, the convergence time was approximately the same. Slower convergence is found for cPCA. With respect to the eigenvalue estimate, all rules except cPCA have approximately the same time course; see figure 2. While in cPCA the eigenvalue estimate oscillates around its true value, the other rules converge from one direction. As was pointed out in section VII-A, cPCA is suffering from fluctuations in the weight vector length. This is visible in figure 3. In nPCA and ALA, the weight vector length only slightly deviates from unity, although theoretically nPCA could show fluctuations as cPCA does. Figure 4, however, shows that the time course of the weight vector length for nPCA and ALA is almost identical.

C. Medium-Dimensional Image Data

The learning rules were also applied to a medium-dimensional data distribution ($n = 64$) formed by 30,000 windows of size $8 \times 8$ extracted from a gray-scale image of a natural scene (with gray values in $[0, 1]$), as done in [2]. Chains of $m$ single-neuron principal component analyzers were formed following the deflation principle [2]. Here we used $m = 64$, $t_{\text{max}} = 30,000$, $\lambda_0 = 1$, and random orthogonal initial weight vectors. Again, the same distribution, same initialization, and same presentation order were used.

We focus on a comparison between the coupled nPCA rule and Oja’s non-coupled rule; ALA, rPCA, and RRLSA produced results very similar to nPCA, while cPCA was not
suitable for this task. For nPCA, the learning rate was decaying from $\gamma_0 = 0.01$ to $\gamma_{t_{\text{max}}} = 0.0001$, and we used the deflation principle (79, 80); see appendix IV. For Oja’s rule [1], we selected a higher learning rate ($\gamma_0 = 0.1$, $\gamma_{t_{\text{max}}} = 0.001$) and used the deflation principle (77, 78). In both experiments, we selected the deflation principle that produced the best results. The initial learning rates were chosen as high as stability allowed. All units have the same learning rate, we do not assume to have previous knowledge on the eigenvalues. Note that all stages of the chain are trained simultaneously, not sequentially.

The eigenvalues of the 64 units at the end of the training are plotted in figure 5; it is apparent that nPCA robustly converges to eigenvalues ranging over four orders of magnitude, while the eigenvalues did not fully converge with Oja’s rule. The corresponding eigenvectors are visualized in figure 6, showing the typical result of image frequencies increasing with $k$. It is visible that the minor eigenvectors are better structured for nPCA, while they appear to have random values for Oja’s rule. To compare the quality of the eigenvector and eigenvalue estimates, figure 7 depicts the “eigen error” $\varepsilon(w, \lambda) = \|Cw\lambda^{-1} - w\|^2$ for both nPCA and Oja’s rule. In both cases, the later stages in the chain have larger errors, but we see that the error is around one order of magnitude smaller for nPCA.

D. High-Dimensional Image Data

In addition, we demonstrate that coupled learning rules are also applicable to very high-dimensional data, in this case $n = 16384$. The training data are 72 gray-scale images with $128 \times 128$ pixels of an object rotating around the vertical axis. No windows were extracted, but the entire images were used as data points. The data were taken from the COIL-20 database [12]; we show the results for object no. 2 (figure 8, left).

$m = 10$ eigenvectors were extracted in a chain of principal component analyzers using the nPCA rule (23, 24) and the deflation principle (79, 80). The learning rate was decreasing from $\gamma_0 = 0.01$ to $\gamma_{t_{\text{max}}} = 0.0001$ over $t_{\text{max}} = 3000$ learning steps, with an initial eigenvalue estimate $\lambda_0 = 10.0$. For comparison, we computed the same number of eigenvectors with the program eigen from the SLAM software library [13],
which uses an algorithm based on singular value decomposition along with a conjugate gradient algorithm [14].

The principal eigenvectors of both methods are shown in figure 8, the result from SLAM in the top row, the result from nPCA in the bottom row. The eigenvectors are very similar to each other; inverted colors result from eigenvectors with opposite sign. Also the eigenvalues (not shown) of both methods are closely corresponding. In terms of computation speed, our method cannot compete with the SLAM program (factor $1:100$); reasonably good eigenvector estimates are, however, already available earlier in the learning process.

IX. DISCUSSION

We can summarize the theoretical insights gained in this paper as follows.

1) Non-coupled PCA/MCA rules suffer from a stability-speed problem, since the eigen-motion depends on the eigenvalues of the covariance matrix. In PCA rules, the eigen-motion depends on the principal eigenvalue, in MCA rules on all eigenvalues of the covariance matrix (section III).

2) Newton’s method yields averaged systems with identical speed of convergence in all eigendirections. We apply Newton’s method to an information criterion that explicitly depends on eigenvector and eigenvalue estimates and has stationary points in the eigenvectors and eigenvalues of the covariance matrix. We obtain averaged learning rule systems for PCA and MCA (nPCA, nMCA) with coupled differential equations for eigenvector and eigenvalue estimates. The eigen-motion of these systems is widely independent of the eigenvalues of the covariance matrix, thus solving the stability-speed problem for averaged systems (section III).

3) Averaged coupled PCA systems obtained from Newton’s method can be turned into online equations (section IV), but MCA systems can not. In the MCA case, equal convergence speed in all eigendirections requires averaged equations containing $C^{-1}$, but this prevents the derivation of online forms. Normalizing the learning rate in MCA equations as suggested for ALA does not solve the stability-speed problem (section III).

4) The additional effort for coupled learning rules in comparison to non-coupled rules — an equation for the estimation of the eigenvalue and the consideration of the eigenvalue estimate in the weight learning rule — is small.

5) The online form (23, 24) of the nPCA rule may theoretically show fluctuations in the weight vector length (section IV), although in simulations this was not found to be the case.

6) Different approximations of the nPCA rule yield the ALA PCA learning rule (30, 31) — a form of Oja’s PCA rule —, and the novel coupled cPCA rule (39, 40). ALA mends the potential problem of fluctuating weight vector length at the price of somewhat less optimal eigenvalues of the Jacobian. The cPCA rule, however, suffers from fluctuations (sections V and VII-A).

7) As was shown before, ALA (30, 31) may diverge and requires either smaller learning rates or a special mechanism to confine the weight vector length [5] when it exceeds a threshold. The same holds for nPCA (23, 24) and for cPCA (39, 40). Obviously, a Newton descent only guarantees equal convergence speeds in the vicinity of the stationary point, while in larger distance from the stationary point, the speed of convergence may differ between the different eigendirections, thus entailing stability problems in the numerical solution.

8) Explicit normalization of the weight vector length turns the cPCA system (39, 40) into a robust principal component analyzer, rPCA (42, 43), that is not affected by divergence and can operate with higher learning rates. An approximation of the explicit normalization for small learning rates yields the Euler solution (44, 45) for ALA (section VII-B).

9) A different approximation of the rPCA rule (42, 43) leads to the RRLSA PCA method. RRLSA can also be obtained by an approximation of ALA. RRLSA was originally derived from a recursive least squares approach [15], [5], but, as the other rules presented here, also fits into the integrative Newton framework (section VII-C).

10) Simulations confirm that coupled PCA learning rules applied in chains of simultaneously trained stages lead to improved accuracy of the eigenvectors and eigenvalues (section VIII).

11) The novel cPCA rule (39, 40) has no counterpart in a non-coupled rule. As this example demonstrates, the coupled systems approach may generally open the way to new forms of learning rules which are impossible in
non-coupled systems, thus going beyond mere learning rate control as in ALA.

12) When Newton’s method is applied, the freedom of choosing an information criterion is increased. The criterion just has to have stationary points in the desired solutions. Our criterion (1) is related to the “normalized Mahalanobis” measure [16, sec. 4.2].

**APPENDIX I**

**PROPERTIES OF NEWTON’S METHOD**

We show that applying Newton’s method leads to a system which (i) has the same convergence speed in all directions, (ii) has a Jacobian with eigenvalues of -1, and (iii) turns all stationary points into attractors. Let \( f(x) \) be a function which can be written as Taylor expansion around \( x_0 \), with \( \xi = x - x_0 \):

\[
f(x) = f(x_0) + \frac{\partial f}{\partial x} \xi + \frac{1}{2} \xi^T \frac{\partial^2 f}{\partial x^2} \xi + \ldots,
\]

where “…” indicates higher-order terms in \( \xi \). If \( x_0 \) is a stationary point of \( f \), we get

\[
f(x) = f(x_0) + \frac{1}{2} \xi^T \frac{\partial^2 f}{\partial x^2} \xi + \ldots
\]

The matrix \( H(x) = \frac{\partial^2 f}{\partial x^2} \) is the Hessian of \( f \). We obtain an approximation of the gradient

\[
\frac{\partial f(x)}{\partial x} \approx H(x_0) : \xi.
\]

The Newton descent in \( f \) is defined as

\[
x = -H^{-1}(x) \frac{\partial f(x)}{\partial x},
\]

and combined with the above approximation we obtain

\[
\dot{x} \approx -H^{-1}(x_0) \cdot H(x_0) \cdot (x - x_0).
\]

We can also approximate \( H^{-1}(x) \) with a Taylor expansion as

\[
H^{-1}(x) = H^{-1}(x_0) + \ldots
\]

and obtain a first-order approximation

\[
\dot{x} \approx -H^{-1}(x_0) \cdot H(x_0) \cdot (x - x_0).
\]

This system has a stable attractor in \( x_0 \). It convergences with the same speed from all directions. The eigenvalues of the Jacobian of the system are -1.

**APPENDIX II**

**INVERSION OF THE HESSIAN**

The inversion of the Hessian \( H \) from equation (6) is simplified by an orthogonal transformation with

\[
U = \begin{pmatrix} \mathbf{W} & 0^T \\ 0 & 1 \end{pmatrix},
\]

where \( \mathbf{W} \) contains all eigenvectors of \( \mathbf{C} \) in its columns. We get the transformed Hessian \( H^* = U^T H U \)

\[
H^* = 2 \begin{pmatrix} \Lambda_\lambda^{-1} - I & -e_1 \lambda^{-2} \\ -e_1 \lambda^{-2} & w^T C \lambda^{-3} - \frac{1}{2} \lambda^{-2} \end{pmatrix},
\]

where \( \Lambda \) is a diagonal matrix containing all eigenvalues of \( \mathbf{C} \). In the vicinity of a stationary point \( (w_1, \lambda_1) \) we approximate \( w \approx w_1, \lambda \approx \lambda_1 \), \( w^T w \approx e_1 = (1, 0, \ldots, 0) \), and \( w^T C w \approx \lambda_1 \), and obtain

\[
H^* \approx 2 \begin{pmatrix} \Lambda_\lambda^{-1} - I & -e_1 \lambda^{-1} \\ -e_1 \lambda^{-1} & \frac{1}{2} \lambda^{-2} \end{pmatrix}.
\]

Using the ansatz

\[
H^* = 2 \begin{pmatrix} A \lambda^{-1} - I & -e_1 \lambda^{-1} \\ -e_1 \lambda^{-1} & \frac{1}{2} \lambda^{-2} \end{pmatrix}.
\]

we get the following equations from \( H^* H^{-1} = I \):

\[
\begin{align*}
A &= (\Lambda_\lambda^{-1} - I - 2e_1 e_1^T)^{-1} \\
B &= 2\Lambda_\lambda^{-1} A \\
c &= 2\Lambda_\lambda^{-1} - 2e_1 e_1^T e_1 \\
d &= 2\lambda^2 + 2\lambda e_1^T c.
\end{align*}
\]

The matrix \( \Lambda_\lambda^{-1} - I - 2e_1 e_1^T \) has different approximations for principal component analysis, where \( \lambda \approx \lambda_1 \), \( j = 2, \ldots, n \), and for minor component analysis, where \( \lambda \approx \lambda_j \), \( j = 2, \ldots, n \), and we obtain:

\[
\begin{align*}
A_{\text{PCA}} &= \frac{1}{2} e_1 e_1^T - I \\
A_{\text{MCA}} &= \Lambda^{-1} - \frac{3}{2} e_1 e_1^T.
\end{align*}
\]

The remaining elements as determined from (63), (64), and (65) are equal for the two cases, \( b^T \approx -e_1^2 \lambda, c \approx -e_1 \lambda, d \approx 0, \) and with \( H^{-1} = U H^* U^{-1} \) we get

\[
\begin{align*}
H_{\text{PCA}}^{-1} &= \frac{1}{2} \begin{pmatrix} \frac{1}{2} w^T w - I & -w \lambda \\ -w \lambda & 0 \end{pmatrix} \\
H_{\text{MCA}}^{-1} &= \frac{1}{2} \begin{pmatrix} C^{-1} \lambda - \frac{3}{2} w^T w & -w \lambda \\ -w \lambda & 0 \end{pmatrix}.
\end{align*}
\]

**APPENDIX III**

**APPROXIMATION OF EXPLICIT NORMALIZATION**

A time-discrete system with explicit normalization determines the next weight vector \( \mathbf{w}' \) from the previous weight vector \( \mathbf{w} \) by

\[
\mathbf{w}' = \frac{\mathbf{w} + \gamma \mathbf{w}}{\| \mathbf{w} + \gamma \mathbf{w} \|}
\]

In such a system, \( \mathbf{w}^T \mathbf{w} = 1 \) is guaranteed all the time. For small \( \gamma \), we can approximate the denominator by

\[
[[\mathbf{w}^T + \gamma \mathbf{w}]/(\mathbf{w} + \gamma \mathbf{w})]^{-\frac{1}{2}}
\]

\[
= [\mathbf{w}^T \mathbf{w} + 2\mathbf{w}^T \mathbf{w} + \mathcal{O}(\gamma^2)]^{-\frac{1}{2}}
\]

\[
\approx 1 - \gamma \mathbf{w}^T \mathbf{w}.
\]

We obtain

\[
\mathbf{w}' = (\mathbf{w} + \gamma \mathbf{w})(1 - \gamma \mathbf{w}^T \mathbf{w})
\]

\[
= \mathbf{w} + \gamma (\mathbf{w} - \mathbf{w}^T \mathbf{w}) + \mathcal{O}(\gamma^2)
\]

\[
\approx \mathbf{w} + \gamma (\mathbf{w} - \mathbf{w}^T \mathbf{w}).
\]
APPENDIX IV

DEFLATION IN MULTI-NEURON NETWORKS

All single-neuron PCA networks can be concatenated in chains that would extract the first \( m \) principal eigenvectors and eigenvalues by “deflation”. In the original approach by Sanger [2], neuron \( k \) sees the deflated input vector \( \mathbf{x}_{k,t} \), which is recursively computed from

\[
\mathbf{x}_{k,t} = \begin{cases} \mathbf{x}_t & k = 1 \\ \mathbf{x}_{k-1,t} - \mathbf{w}_{k-1,t} \mathbf{y}_{k-1,t} & k = 2, \ldots, m, \end{cases} \tag{77}
\]

and determines its output from

\[
\mathbf{y}_{k,t} = \mathbf{w}_{k,t}^T \mathbf{x}_{k,t}. \tag{78}
\]

In coupled rules, we observed an improvement of performance with a modified deflation principle. There, the deflated input vector \( \mathbf{x}_{k,t} \) is recursively computed from

\[
\tilde{\mathbf{x}}_{k,t} = \begin{cases} \mathbf{x}_t & k = 1 \\ \tilde{\mathbf{x}}_{k-1,t} - \mathbf{w}_{k-1,t} \mathbf{y}_{k-1,t} & k = 2, \ldots, m, \end{cases} \tag{79}
\]

and determines its output from

\[
\tilde{\mathbf{y}}_{k,t} = \mathbf{w}_{k,t}^T \tilde{\mathbf{x}}_{k,t}. \tag{80}
\]

Note that the outputs are computed from the deflated input vectors, not from the original input vector as in [2]. Convergence is improved, if the eigenvalue estimate is updated using this output value. We decided to also use this value for the weight update and for the recursive deflation. A theoretical analysis of the differences between the two deflation methods still has to be done.

ACKNOWLEDGMENT

The authors would like to thank Bärbel Herrnberger and the two anonymous reviewers for helpful comments on the first version of this manuscript.

REFERENCES