An Extension of Neural Gas to Local PCA

Ralf Möller, Heiko Hoffmann

Cognitive Robotics, Max Planck Institute for Psychological Research, Amalienstr. 33, D-80799 Munich, Germany

Abstract

We suggest an extension of the Neural Gas vector quantization method to local principal component analysis. The distance measure for the competition between local units combines a normalized Mahalanobis distance in the principal subspace and the squared reconstruction error, with the weighting of both measures depending on the residual variance in the minor subspace. A recursive least squares method performs the local principal component analysis. The method is tested on synthetic two- and three-dimensional data and on the recognition of handwritten digits.

Key words: unsupervised learning, local principal component analysis, vector quantization, Neural Gas, handwritten digit recognition

1 Introduction

Principal component analysis (PCA) is a method for dimension reduction of multivariate data and a standard tool for data compression, pattern recognition, and statistical analysis. Through PCA, a high-dimensional pattern space is approximated by a subspace of lower dimension which is spanned by the first principal eigenvectors of the data covariance matrix. In that way, low-variance components of the data distribution are eliminated, resulting in a more parsimonious description. Not only can single data points be represented by fewer components under minimal error in the reconstruction of the original data, but the entire distribution can be represented by the principal eigenvectors and their corresponding eigenvalues (variance in the eigendirection).

However, PCA is a *linear* technique where data points are described by their projections onto a hyperplane embedded in the original data space. Therefore,

[1] *E-mail addresses:* moeller@psy.mpg.de, hoffmann@psy.mpg.de

globally applying PCA to a distribution will be inefficient, if the components have non-linear dependencies [1]. Several extensions to PCA have been suggested to overcome this problem, e.g. non-linear expansion and principal curves (overviews: [1; 2]). An alternative to these global non-linear approaches are *local PCA* [2] methods, where a distribution is represented by a collection of locally linear descriptions, the latter being obtained by applying PCA to a local region of the data distribution. Thus, local PCA can be used to produce compact models of non-linear data distributions [2].

The model of a distribution can be applied for detection, classification, completion of partially defined data, and data compression. In *detection*, samples from a data stream are characterized as being part of the modeled distribution or not (e.g. face detection in images [4]). In *classification*, detection is performed with respect to a number of different modeled distributions (e.g. handwritten digit recognition [5]). In *completion* of partially defined samples, some components of the sample vector are specified (input variables), and the remaining components (output variables) are determined in a way that the resulting vector would with high probability be part of the modeled distribution. In *compression*, each sample would be characterized by the index of the sub-model and by the coordinates within the corresponding local linear subspace.

Several methods for local PCA (which is closely related to "Gaussian mixture models") have been suggested; see the discussion in section 5.2. Here we introduce a method which was derived from a deterministic framework (vector quantization, PCA, and annealing). It is based on the soft competition scheme of the Neural Gas vector quantization method [6] and uses an online regime with an update of centers and subspaces after the presentation of each single data point. In soft competition methods, a data point is not exclusively assigned to one unit, but shared between multiple units according to a set of weighting factors (see section 5.2). Soft competition is generally favorable since it prevents the method from getting caught in sub-optimal solutions [2], and especially the ranking method adopted from Neural Gas [6] is known to converge fast to near-optimal solutions. Instead of the Euclidean distance measure used by Neural Gas, a combination of a normalized Mahalanobis distance with the squared reconstruction error guides the competition between the units. The weighting between the two measures is determined from the residual variance in the minor subspace of each sub-model. The unit centers are updated as in Neural Gas, while the subspace learning is based on an on-

[2] The term "local PCA" is also used to describe local Hebbian type learning algorithms where "local" refers to the fact that the weight modification in a single PCA network only depends on the corresponding pre- and postsynaptic activities [3]. We thank one of the anonymous reviewers for pointing out this possible source of confusion.

line neural principal component analyzer, the "robust recursive least squares learning algorithm" (RRLSA) [7].

In section 2, we briefly recapitulate the Neural Gas vector quantization approach. Our extension of this method to local PCA is introduced in section 3. The method is tested on low-dimensional synthetic data and on a high-dimensional problem of handwritten digit recognition in section 4. We discuss the distance measure used for the competition in section 5.1 and the relation to other local PCA methods in section 5.2. The PCA method is described in appendix A, appendix B provides an analysis of the extrema in the potential function that is underlying the competition in our model, appendix C describes the computation of the log-likelihood that was used to evaluate the experimental results, and appendix D describes the models of the synthetic distributions.

2 Neural Gas

Vector quantization methods encode a set of data points in n-dimensional space with a smaller set of reference or center vectors $\mathbf{c}_k$; $k = 1, \ldots, N$. Each data vector is represented by the closest reference vector. The reference vectors are determined such that the expected Euclidean distance between all data vectors and their corresponding reference vectors becomes minimal. Neural Gas [6] is a vector quantization technique with soft competition between the units. The reference vectors $\mathbf{c}_k$ are initialized by randomly assigning vectors from the training set. In each training step, the squared Euclidean distances

$$d_k(\mathbf{x}) = \|\mathbf{x} - \mathbf{c}_k\|^2 = (\mathbf{x} - \mathbf{c}_k)^T(\mathbf{x} - \mathbf{c}_k) \tag{1}$$

between a randomly selected input vector $\mathbf{x}$ from the training set and all reference vectors $\mathbf{c}_k$ are computed; the vector of these distances is $\mathbf{d}$. Each reference vector k is assigned a rank $r_k(\mathbf{d}) = 0, \ldots, N-1$, where a rank of 0 indicates the closest and a rank of $N-1$ the most distant reference vector to $\mathbf{x}$. The reference vectors are then adjusted according to

$$\mathbf{c}_k \leftarrow \mathbf{c}_k + \varepsilon \cdot h_\varrho[r_k(\mathbf{d})] \cdot (\mathbf{x} - \mathbf{c}_k). \tag{2}$$

The function $h_\varrho(r) = e^{-r/\varrho}$ implements the soft competition method: not only the best-matching reference vector is adapted (with $h_\varrho(0) = 1$), but also all other reference vectors, with a factor that is exponentially decreasing with their rank. The width of this influence is determined by the neighborhood range ϱ. In addition, the update of all reference vectors is affected by a global learning rate ε. Over the course of learning, both ϱ and ε decrease exponentially from an initial positive value ($\varrho(0)$, $\varepsilon(0)$) to a smaller final positive value ($\varrho(T)$, $\varepsilon(T)$) according to $\varrho(t) = \varrho(0)[\varrho(T)/\varrho(0)]^{(t/T)}$ and

$\varepsilon(t) = \varepsilon(0)[\varepsilon(T)/\varepsilon(0)]^{(t/T)}$. Here t denotes the time step and T the total number of training steps. Thus, over time, modifications become more local and convergence is enforced.

3 Extension of Neural Gas to Local PCA

In contrast to plain vector quantization, local PCA techniques do not only partition the data space into a number of disjunctive regions, but also approximate the data points within a region by their projections into the local system of the principal eigenvectors (i.e., the eigenvectors with the largest associated eigenvalues). Thus, the distribution is represented by a collection or a mixture of local linear sub-models (referred to as "units" in the following) [2].

In our extension of the Neural Gas method to local PCA, each of the N units is described by a tuple $\{\mathbf{c}_k, \mathbf{W}_k, \mathbf{\Lambda}_k, \lambda_k^*\}$, with $k = 1, \ldots, N$ denoting the unit index. The number of units N is fixed and must be chosen before the training according to the expected complexity of the data distribution and to the computational effort that can be invested in the training; increasing N will usually improve the approximation quality. The vector $\mathbf{c}_k$ is the reference or center vector as in Neural Gas. The $n \times m$ matrix $\mathbf{W}_k$ encodes the first m principal eigenvector estimates $\mathbf{w}_{ki}$ of the local distribution in its columns $i = 1, \ldots, m$. The diagonal $m \times m$ matrix $\mathbf{\Lambda}_k$ contains the corresponding first m principal eigenvalue estimates λ_{ki}, and the parameter λ_k^* represents an estimate of the eigenvalue in each of the remaining $n-m$ minor eigendirections. In geometrical terms, each unit k now encodes a hyper-ellipsoid centered at $\mathbf{c}_k$. Directions and lengths of the first m half-axes of the hyper-ellipsoid are specified by the unit-length axis vectors $\mathbf{w}_{ki}$ and the square root of the diagonal entries λ_{ki}, respectively. In the remaining $n-m$ dimensions, the hyper-ellipsoid is spherical with a radius $\sqrt{\lambda_k^*}$; see also [5].

Instead of the Euclidean distance (1), the soft competition between the units uses the distance measure

$$d_k(\mathbf{x}) = \mathbf{y}_k^T \mathbf{\Lambda}_k^{-1} \mathbf{y}_k + \frac{1}{\lambda_k^*}(\boldsymbol{\xi}_k^T \boldsymbol{\xi}_k - \mathbf{y}_k^T \mathbf{y}_k) + \ln |\mathbf{\Lambda}_k| + (n-m) \ln \lambda_k^*, \qquad (3)$$

where $\boldsymbol{\xi}_k = \mathbf{x} - \mathbf{c}_k$ is the deviation between the input vector $\mathbf{x}$ and the center of the unit $\mathbf{c}_k$, and the vector $\mathbf{y}_k = \mathbf{W}_k^T \boldsymbol{\xi}_k$ contains the coordinates of $\boldsymbol{\xi}_k$ in the system of the first m eigenvectors $\mathbf{w}_{ki}$. $|\mathbf{\Lambda}_k|$ denotes the determinant of $\mathbf{\Lambda}_k$. We provide a motivation for this distance measure in section 5.1.

For each presented input vector $\mathbf{x}$, the rank $r_k(\mathbf{d})$ of unit k is determined from the distance measure (3) as described above. Equation (2) is used for the update of the centers. Simultaneously, the shape of the hyper-ellipsoid in

the first m dimensions, encoded by $\mathbf{W}_k$ and $\mathbf{\Lambda}_k$, is updated through an online PCA method. In our implementation this is RRLSA, a neural method for principal component analysis based on the recursive least squares method [7], but other methods could be used in its place. We selected RRLSA since it is robust and converges fast even if the eigenvalues extend over several orders of magnitude. The method was slightly modified by introducing a unit-specific learning rate $\alpha_k = \varepsilon \cdot h_\varrho[r_k(\mathbf{d})]$. We abstractly write the update as

$$\mathbf{W}_k, \mathbf{\Lambda}_k \leftarrow \mathrm{PCA}\{\mathbf{W}_k, \mathbf{\Lambda}_k, \boldsymbol{\xi}_k, \alpha_k\}, \tag{4}$$

and provide the details of the modified method in appendix A. An estimate λ_k^* of the eigenvalues in the $n-m$ minor directions is obtained by determining the residual variance σ_k^2 in these directions from

$$\sigma_k^2 \leftarrow \sigma_k^2 + \alpha_k \cdot (\boldsymbol{\xi}_k^T \boldsymbol{\xi}_k - \mathbf{y}_k^T \mathbf{y}_k - \sigma_k^2), \tag{5}$$

and distributing the residual variance evenly to all minor directions gives

$$\lambda_k^* = \frac{\sigma_k^2}{n-m}. \tag{6}$$

The neighborhood range ϱ and the learning rate ε decrease exponentially as in the Neural Gas method. The initial and final values of neighborhood range and learning rate need to be adjusted by experimentation. The parameter sets given in this paper can be used as orientation for the selection.

The unit centers $\mathbf{c}_k$ are initialized as in Neural Gas, the eigenvector estimates $\mathbf{w}_{ki}$ of each unit k are initially set to a random orthonormal system, the m eigenvalue estimates λ_{ki} are assigned identical values $\lambda(0)$ in all units and dimensions, and the residual variances σ_k^2 are initially set to a value $\sigma^2(0)$. The choice of $\lambda(0)$ and $\sigma^2(0)$ is not critical.

In the following, our learning system is referred to as a "network". Although we do not provide a network schematic with neurons and synaptic connections here, we assume that for most of the mechanisms of our learning system (PCA units, competition between units) strictly connectionist counterparts can be found.

4 Simulations

We test the suggested local PCA method on synthetic data in two and three dimensions (section 4.1) and on high-dimensional data in a pattern recognition task (section 4.2).

4.1 Synthetic Data

4.1.1 Data Generation and Analysis

Training data were synthesized from given distribution models (see appendix D). The following standard parameter set was used in the training (unless stated otherwise): number of training steps $T = 30\,000$, final neighborhood range $\varrho(T) = 0.01$, initial global learning rate $\varepsilon(0) = 0.5$, final global learning rate $\varepsilon(T) = 0.05$. The initial neighborhood range $\varrho(0)$ was varied with the number N of units and is specified for each experiment. The initial eigenvalue $\lambda(0)$ and the initial residual variance estimate $\sigma^2(0)$ were chosen according to the spatial extension of the distribution and are also specified for each experiment.

In figures 1 to 5 and figure 11, training vectors are depicted as small gray squares. The training results are visualized as a set of Mahalanobis ellipsoids centered at $\mathbf{c}_k$. For graphical reasons, the length of the half-axis i of a unit k is shown as $1.5\sqrt{\lambda_{ki}}$. The principal eigenvector is visualized by a thick arrow, the second principal eigenvector by a thinner arrow. For experiments with three-dimensional data (figures 4 and 5), the projections of training vectors and ellipsoids onto two coordinate planes are shown (the coordinate systems are indicated in the diagrams).

For each result, we specify the log-likelihood per pattern ($\mathcal{L}$, see appendix C). Note that the absolute value of $\mathcal{L}$ depends on the scaling of the distribution and is therefore not of importance. Moreover, as visible in figure 1, the absolute change of $\mathcal{L}$ from an unordered to the final well-ordered state is small. We therefore only provide $\mathcal{L}$ with the intention to demonstrate an improvement in the approximation of the data points over training, and to give a basis for the comparison between different methods.

After training, each pattern can be assigned to one of the units according to the minimal distance (3). To evaluate if all units are used to approximate the distribution, we count the number of patterns (ν_k) assigned to each unit and provide the mean and standard deviation of this number over all units.

4.1.2 Vortex-like Distribution

Figure 1 shows the network state at different time steps in the training process for a two-dimensional, vortex-like distribution. The log-likelihood increases during training, and the improvement of the model quality is clearly visible. After the final step, each unit is assigned approximately the same number of patterns, but the size of the ellipsoids in the center of the vortex is smaller since there the density of the data points is higher.

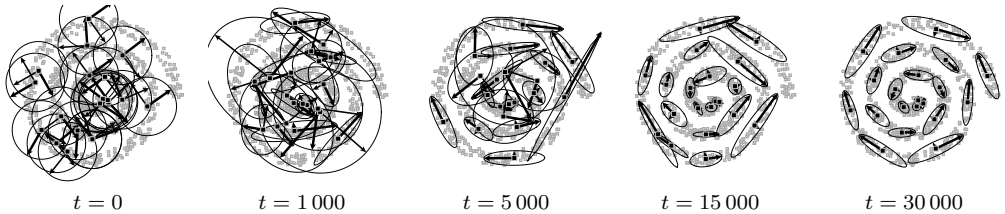


Fig. 1. Snapshots of the training process at different time steps t for a vortex-like distribution with 1 000 points ($n = m = 2$, $N = 20$, $\varrho(0) = 2$, $\lambda(0) = \sigma^2(0) = 1\,000.0$). Final assignment: $\nu_k = 50 \pm 9.7$ patterns per unit.

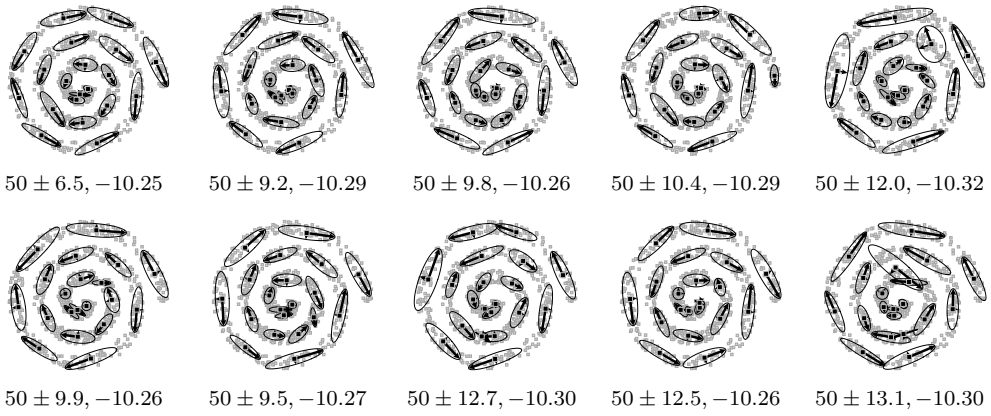
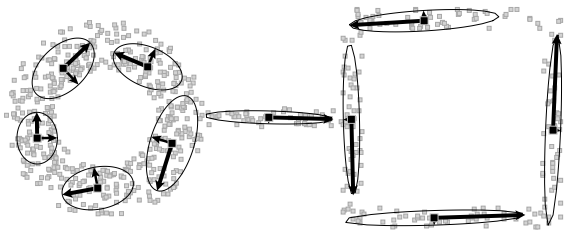


Fig. 2. Final network states for a vortex-like distribution with 1 000 points ($n = m = 2$, $N = 20$, $\varrho(0) = 2$, $\lambda(0) = \sigma^2(0) = 1\,000.0$), obtained from 10 runs with different random initialization and different random presentation order of the data points. Mean and standard deviation of ν_k and the log-likelihood $\mathcal{L}$ are given for each final state.

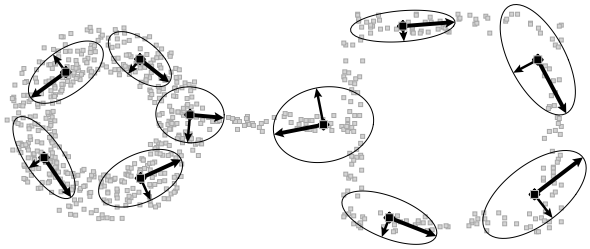
In order to test the robustness of the method, we repeated the training process 10 times with the same distribution, but different random initialization and different random presentation order of the data points. The final network states are shown in figure 2. Visual inspection reveals small defects in only two of the experiments (rightmost runs in the top and bottom row); in both cases, one of the ellipses is extending over the gaps in the distribution. Except for these two runs, the final approximations are qualitatively the same.

4.1.3 Ring-line-square Distribution

Ten learning runs with the the ring-line-square distribution in figure 3 (A) produced qualitatively comparable results (not shown) with log-likelihoods of the final approximation ranging from -10.91 to -10.85. A comparison of figure 3 A and B illustrates the advantage of our method, where competition and PCA are integrated (A), over a two-stage method (B). In the two-stage method, which is the simplest form of local PCA, the partitioning is accomplished by vector quantization (here Neural Gas), and PCA is subsequently



A



B

Fig. 3. Results of the training for a two-dimensional distribution with 500 data points in the ring, 50 data points in the line, and 200 data points in the square part ($n = m = 2$, $N = 10$, $\varrho(0) = 1$, $\lambda(0) = \sigma^2(0) = 1\,000.0$). A: Training process as described in section 3 ($\mathcal{L} = -10.85$, $\nu_k = 75 \pm 26.6$). B: Local regions are obtained from a Neural Gas step, then PCA is executed in each region separately ($\mathcal{L} = -11.09$, $\nu_k = 75 \pm 33.6$).

applied within each of the local regions defined by the vector quantization. In the PCA phase, there is no competition between the regions, and the quantization itself remains unchanged. For each region, 3 000 steps with the same PCA method (see appendix A) were executed. The log-likelihood values show that the approximation is better with the integrated method, which is also obvious from the diagrams. The centers found by the Neural Gas method (B) differ from those found by the local PCA method (A), especially in the square-shaped part of the distribution.

4.1.4 Spiral Distribution 1

Results with a three-dimensional spiral-shaped distribution are shown in figures 4 and 5. In this example, uniformly distributed noise in the directions of the three axes was used to generate the noisy spiral (see appendix D). Full PCA ($n = m$) was used in figure 4, while in figure 5, the same distribution was approximated by units with a single principal eigenvector ($m = 1$) and estimation of λ_k^* from the residual variance. Obviously the method found reasonable values for λ_k^* (indicated by circles in figure 5).

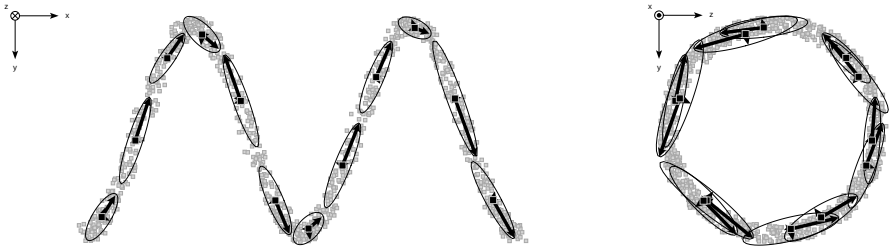


Fig. 4. Results of the training process for a three-dimensional spiral-shaped distribution ($n = m = 3$, $N = 12$, $\varrho(0) = 1$). $\mathcal{L} = -14.06$, $\nu_k = 83.3 \pm 10.6$.

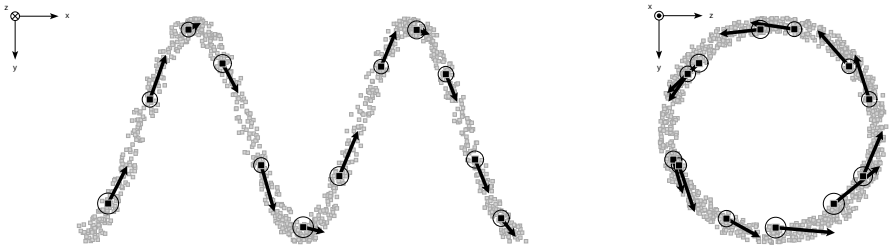


Fig. 5. Results for the same distribution as in figure 4, but with dimension reduction ($n = 3$, $m = 1$, $N = 12$, $\varrho(0) = 1$). The radii of the circles have the values $1.5\sqrt{\lambda_k^*}$. $\mathcal{L} = -14.11$, $\nu_k = 83.3 \pm 12.7$.

4.1.5 Spiral Distribution 2

For a similar spiral-shaped distribution, we compared the performance of our method with two of the most recent local PCA methods, the Mixture of Probabilistic Principal Component Analyzers ("Mixture-of-PPCA") method [2] and the "Generalized Radial Basis Function Network" [8]. Spiral data (500 points with Gaussian noise) were generated with the same Matlab program as used in [2]. In all experiments, $N = 8$ units with a single principal eigenvector ($m = 1$) were used.

For the standard parameter set with $\varrho(0) = 1.0$ and $\lambda(0) = \sigma^2(0) = 1.0$, our method produces a model with $\mathcal{L} = 1.510$ (mean over 5 values ranging from 1.479 to 1.538). The final result can be slightly improved without qualitative differences by lowering both the final neighborhood range and the final learning rate. With the modified parameter set $T = 30\,000$, $\varrho(0) = 1.0$, $\varrho(T) = 0.005$, $\varepsilon(0) = 0.5$, $\varepsilon(T) = 0.005$, $\lambda(0) = \sigma^2(0) = 1.0$, we obtain $\mathcal{L} = -1.444$ (mean over 5 values ranging from -1.425 to -1.470). The Netlab implementation [9] of the Mixture-of-PPCA method [2] achieves $\mathcal{L} = -1.483$ (for exactly the same data set), and a value of $\mathcal{L} = -1.426$ is given in the paper [2] for a different data set generated from the same model distribution with the above-mentioned program. In the Generalized Radial Basis Function Network (used as described in section 3.1.3 in [8]), the mean log-likelihood was $\mathcal{L} = -1.480$, obtained from 5 values ranging from -1.464 to -1.507. Note that the differences in the magnitude of $\mathcal{L}$ to the spiral distributions in figures 4 and 5 are due to

differences in the scaling of the distributions which affects the log-likelihood.

4.2 Recognition of Handwritten Digits

In this paper, we use handwritten digit recognition to test the generalization abilities of our local PCA method on high-dimensional data (here $n = 784$). Our goal is not to compete with other pattern recognition methods (which in their performance are often superior to our results [10]), but to compare our local PCA approach with the performance of similar unsupervised approaches. Each digit class is modeled by a separate network. In the training process, the different networks do not interact. For the classification of a digit, the distances to all units of all networks are computed, and the digit is classified as part of the network model containing the unit with minimal distance value.

We used 70 000 digits from the handwritten digit database MNIST [11] which is a subset of the NIST database produced by the U.S. National Institute of Standards and Technology. The database is separated into 60 000 digits for training and 10 000 digits for testing. The gray-scale images (scaled to pixel values in the interval $[0, 1]$) are centered in a 28×28 pixel grid. Without any preprocessing they were transformed into feature vectors with $28 \times 28 = 784$ dimensions and compiled in 10 training sets.

Each of the 10 networks contains $N = 10$ units with $m = 10$ eigenvector directions. The eigenvector estimates were initialized with random orthonormal vectors, and the initial values $\lambda(0) = \sigma^2(0) = 0.1$ were chosen. The network was trained using the parameter set $T = 30\,000$, $\varrho(0) = 2$, $\varrho(T) = 0.002$, $\varepsilon(0) = 0.5$, $\varepsilon(T) = 0.0002$. The training of all 10 networks took about 36 minutes on an Athlon XP 2200+ with 1 GB RAM. In order to analyze whether the shape of the hyper-ellipsoids is actually of importance for the classification, we compare the distance measure (3) — a combination of a normalized Mahalanobis distance with the squared reconstruction error (section 5.1) — with the Euclidean distance (1). To give a comparison with a vector quantization approach, we also trained and tested with the standard Neural Gas method using the same training parameters (training time about 6 minutes). As with the synthetic data (see figure 3, B), we applied the two-stage learning method where the data set is partitioned by Neural Gas and PCA is performed separately in each region without competition between the units (PCA stage: 3 000 training steps, training time about 2 minutes).

Table 1 summarizes the results. The error rates given for the extension of Neural Gas are the averages over 3 training processes, each with different random initialization and different random presentation order of the data points. The differences between minimal and maximal error rates were 0.26%, 0.10%, and

Table 1
Error rates of the digit classification (image size 28×28). The percentage of misclassified digits is given for different training methods, different distance measures in the classification, and for the classification of the training or the test set.

training method	distance measure	data set	error rate
extension of Neural Gas	normal. Mahalanobis + reconstr.	test	2.79 %
	normal. Mahalanobis + reconstr.	training	1.97 %
	Euclidean	test	8.23 %
Neural Gas	Euclidean	test	7.61 %
Neural Gas + PCA	normal. Mahalanobis + reconstr.	test	3.03 %

0.38% for the error values given in rows 1 to 3. Error rates achieved with the Euclidean distance measure (1) are considerably higher than with the normalized Mahalanobis plus reconstruction distance (3), both for the centers found by our method and for the centers found by Neural Gas. This clearly shows that indeed the shapes of the hyper-ellipsoids are essential in the classification. The result of the Neural Gas algorithm is slightly better than the Euclidean measure applied to our method, since the optimization in Neural Gas directly relates to this distance measure. Surprisingly, the simple two-stage procedure (Neural Gas plus PCA) yields almost as good results as our method and even performs slightly better than previously suggested local PCA methods (see below). Apparently, handwritten digit data are not a test case were an integrated local PCA method like ours shows a pronounced advantage over a sequential method (as it is visible in figure 3).

In the following, we will take a closer look at the structure of the trained model. The patterns of the training set were about evenly assigned to the units ($\nu_k = 600 \pm 92$ over all digits). The centers of the 10 hyper-ellipsoids of each network are shown in figure 6. They are typical representatives of the 10 digits with variations in the style of writing (different slants and sizes, digit "2" with or without loop, digit "7" with or without cross-bar). Figure 7 shows the eigenvectors $\mathbf{w}_{ki}$ which determine the orientation of one hyper-ellipsoid in the network of the digit "2". The eigenvectors encode the variations around the digit represented by the center vector. To illustrate this further, we added multiples of one of the eigenvectors (here the principal eigenvector) to the center and plotted the resulting image; see figure 8. Comparing the images in the two opposite directions along one eigenvector reveals a change of the digit's size and of the slant of the lower stroke.

Figure 9 shows a sample of images mis-classified by our method together with the center vectors of the unit to which they were assigned. Misclassified images mostly seem to be extreme variations lying far from the majority of other digits of their class and are thus not covered by the describing hyper-ellipsoids.

Fig. 6. The centers of the 10 hyper-ellipsoids of each network after training.

Fig. 7. Center (left image) and eigenvectors (in deflation order from left to right) of one unit of the network for the digit "2". In the eigenvector diagrams, white and black indicate positive and negative components, respectively.

Fig. 8. The variation of a digit by adding multiples of the principal eigenvector $\mathbf{w}_{k1}$ to the center $\mathbf{c}_k$. The center image $\mathbf{c}_k$ is marked by a frame, the eigenvector $\mathbf{w}_{k1}$ is depicted on the right side. From the center to the left, $-0.5\sqrt{\lambda_{k1}}\mathbf{w}_{k1}$ is added to each picture. Thus, the picture on the very left deviates by $-2\sqrt{\lambda_{k1}}\mathbf{w}_{k1}$ from the center. From the center image to the right, the vector $0.5\sqrt{\lambda_{k1}}\mathbf{w}_{k1}$ is added. The principal eigenvalue was $\lambda_{k1} = 5.50$.

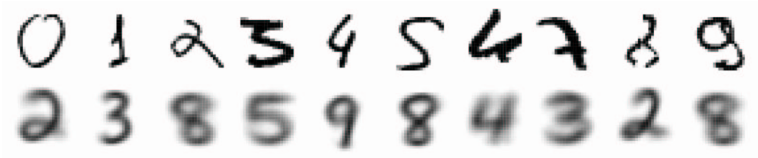


Fig. 9. A sample of the mis-classified digits. The first mis-classified digit of each class is shown (top row, class "0" to "9" from left to right) together with the center vector of the unit to which the pattern was assigned (bottom row).

Table 2
The classification performance of our model (first two rows) compared to others. The database, the number of training patterns, the image size, the number of units, and the number of principal components used are given. The classification error on the test set is shown in the last column.

training method	database	patterns	image size	units	PCs	error
extension of NG	MNIST	60 000	28×28	10	10	2.79%
extension of NG	MNIST	60 000	8×8	10	10	3.11%
Hinton *et al.* [5]	CEDAR	11 000	8×8	10	10	4.68%
Tipping *et al.* [2]	CEDAR	11 000	8×8	10	10	4.64%
Meinicke *et al.* [12]	MNIST	60 000	8×8	8	var.	2.91%
Zhang *et al.* [14]	NIST	10 000	25×20	20	1	4.58%
Zhang *et al.* [14]	NIST	10 000	25×20	40	1	2.20%

The proposed model performs similarly to other modular models applied to the same task. In order to make the results more comparable, we repeated the training with a database of image size 8×8. This database was obtained from the original database with image size 28×28 in the following way: A margin of 4 pixels was removed from the original images, so that the digits in the resulting 20×20 image are tightly fitting into the frame. Each of the pixels of the final 8×8 image was produced by a weighted summation over a local region with a Gaussian of half-width 1.25 pixels (in the original image). The same training parameter set as for the 28×28 pixel database was used.

Table 2 presents the results of our method with image size 28×28 and 8×8, together with the results obtained from previous publications. The results for our network are averages over 3 training processes (as described above). Hinton *et al.* [5] use a mixture of locally linear models, Tipping and Bishop [2] use the Mixture-of-PPCA method. However, both studies used a different database. Meinicke and Ritter [12] used the same database as we did, sampled down to 8×8 pixels. Their method is a Gaussian mixture model with an additional annealing process, where the dimensionality of the major subspace increased during training depending on the data set. The method of Zhang *et al.* [13; 14] is briefly described in section 5.2; we report the error rates for their best network with the smallest and with the largest number of units.

5 Discussion

5.1 Distance Measure

The core of our extension of Neural Gas to local PCA is the distance measure (3) which guides the competition between the units. So far we did not succeed in deriving the method from a common error function, as it was accomplished for Neural Gas and for local PCA methods based on maximizing the log-likelihood. We can, however, motivate the distance measure by the following analysis. Here we focus on one local region and therefore omit centers and unit indices in the equations. We start with a distance measure $d(\boldsymbol{\xi})$ and its expectation D for all data within the region (subsequently also referred to as potential function):

$$d(\boldsymbol{\xi}) = \boldsymbol{\xi}^T \mathbf{W} \boldsymbol{\Lambda}^{-1} \mathbf{W}^T \boldsymbol{\xi} - \mathrm{tr}(\mathbf{W}^T \mathbf{W}) + \ln |\boldsymbol{\Lambda}| \tag{7}$$

$$D = E\{d(\boldsymbol{\xi})\} = \mathrm{tr}(\mathbf{W}^T \mathbf{C} \mathbf{W} \boldsymbol{\Lambda}^{-1}) - \mathrm{tr}(\mathbf{W}^T \mathbf{W}) + \ln |\boldsymbol{\Lambda}|. \tag{8}$$

$\mathbf{C} = E\{\boldsymbol{\xi}\boldsymbol{\xi}^T\}$ is the covariance matrix of the data assigned to the region, $\mathbf{W}$ is an $n \times m$ matrix ($m \leq n$) the columns of which are the m estimated eigenvectors of dimension n, and $\boldsymbol{\Lambda}$ is a diagonal $m \times m$ matrix containing the estimated eigenvalues.

D has its candidate extreme points in eigenvectors and eigenvalues $(\bar{\mathbf{W}}, \bar{\boldsymbol{\Lambda}})$ of $\mathbf{C}$. In appendix B we prove that under the orthonormality constraint $\mathbf{W}^T \mathbf{W} = \mathbf{I}$ only those candidate points are strict local minima, where $\bar{\mathbf{W}}$ and $\bar{\boldsymbol{\Lambda}}$ contain the *minor* eigenvectors and eigenvalues, respectively, while all other candidate points are saddle points. PCA will therefore minimize (8) only if $n = m$ (determining *all* minor eigenvectors also gives all principal eigenvectors), but not in the case of dimension reduction ($m < n$). We assume it to be a common principle underlying both Neural Gas and our method that the potential function minimized by the learning rules, here D, is the averaged version of the distance measure guiding the competition between the units, here $d(\boldsymbol{\xi})$. The condition $n = m$ must therefore also hold for the distance measure, i.e. a full principal component analysis yielding all n eigenvectors and eigenvalues is required.

However, there are a number of arguments why full PCA should be avoided. Firstly, for high-dimensional data full PCA would be forbiddingly costly. Secondly, most natural data sets (like image data, see e.g. [15]) have large eigenvalues only in few eigendirections, while the variances in the remaining eigendirections are small and can be considered as noise. Thirdly, in many real-world applications the size of the data set will be limited, resulting in only a small number of directions with non-zero variance — trying to deter-

mine more eigendirections than that will produce numerical problems [4]. We therefore strive to reduce the number of eigenvectors to be determined, while at the same time fulfilling the condition $n = m$.

This can be accomplished by approximating all eigenvalues in the remaining $n-m$ minor directions with an identical estimate λ^*. We assume $\mathbf{W}$ and $\mathbf{\Lambda}$ to be of dimension $n \times n$. $\mathbf{W}$ is split into $\mathbf{W} = (\hat{\mathbf{W}}, \check{\mathbf{W}})$, with $\hat{\mathbf{W}}$ containing in its columns the m principal eigenvectors to be determined, and $\check{\mathbf{W}}$ containing the $n - m$ minor eigenvectors. $\mathbf{\Lambda}$ is composed of $\hat{\mathbf{\Lambda}}$ in the upper part, with $\hat{\mathbf{\Lambda}}$ containing the m principal eigenvalues, and $\lambda^*\mathbf{I}$ in the lower part ($\mathbf{I}$ is a unit matrix of size $n - m$). From (7) we obtain

$$
\begin{aligned}
d(\boldsymbol{\xi}) &= \boldsymbol{\xi}^T \mathbf{W} \mathbf{\Lambda}^{-1} \mathbf{W}^T \boldsymbol{\xi} - m + \ln|\mathbf{\Lambda}| && (9)\\
&= \boldsymbol{\xi}^T \hat{\mathbf{W}} \hat{\mathbf{\Lambda}}^{-1} \hat{\mathbf{W}}^T \boldsymbol{\xi} + \frac{1}{\lambda^*} \boldsymbol{\xi}^T \check{\mathbf{W}} \check{\mathbf{W}}^T \boldsymbol{\xi} - m + \ln|\hat{\mathbf{\Lambda}}| + (n-m)\ln\lambda^* && (10)\\
&= \boldsymbol{\xi}^T \hat{\mathbf{W}} \hat{\mathbf{\Lambda}}^{-1} \hat{\mathbf{W}}^T \boldsymbol{\xi} + \frac{1}{\lambda^*} \boldsymbol{\xi}^T (\mathbf{I} - \hat{\mathbf{W}} \hat{\mathbf{W}}^T) \boldsymbol{\xi} - m + \ln|\hat{\mathbf{\Lambda}}| + (n-m)\ln\lambda^* && (11)\\
&= \hat{\mathbf{y}}^T \hat{\mathbf{\Lambda}}^{-1} \hat{\mathbf{y}} + \frac{1}{\lambda^*} (\boldsymbol{\xi}^T \boldsymbol{\xi} - \hat{\mathbf{y}}^T \hat{\mathbf{y}}) - m + \ln|\hat{\mathbf{\Lambda}}| + (n-m)\ln\lambda^*. && (12)
\end{aligned}
$$

Under the orthonormality constraint, the second term in equation (7) becomes $\mathrm{tr}(\mathbf{W}^T\mathbf{W}) = m$. The transition from equation (10) to (11) exploits the orthonormality condition $\mathbf{W}\mathbf{W}^T = \mathbf{I}$. In equation (12) we introduced the transformed coordinates $\hat{\mathbf{y}} = \hat{\mathbf{W}}^T \boldsymbol{\xi}$, from which we obtain an expression for the squared reconstruction error

$$
e^2 = \boldsymbol{\xi}^T \boldsymbol{\xi} - \hat{\mathbf{y}}^T \hat{\mathbf{y}} = \|\boldsymbol{\xi} - \hat{\mathbf{W}} \hat{\mathbf{y}}\|^2. \qquad (13)
$$

The distance measure in equation (12) corresponds to $d_k(\mathbf{x})$ in equation (3); only the constant third term $(-m)$ was omitted since it does not affect the competition. An estimate of the expectation of e^2 — the residual variance $\sigma^2 = E\{e^2\}$ in the remaining $n - m$ directions — is obtained from equation (5), and evenly distributing the residual variance to all minor directions gives λ^* from equation (6).

The distance measure (9) is identical (up to a constant offset and factor) to the "normalized Mahalanobis distance" used by Sung and Poggio [4]

$$
d_M(\boldsymbol{\xi}) = \tfrac{1}{2}(n \ln 2\pi + \boldsymbol{\xi}^T \mathbf{C}^{-1} \boldsymbol{\xi} + \ln|\mathbf{C}|) \qquad (14)
$$

which is the negative logarithm of the best-fitting multi-variate Gaussian for a distribution with the covariance matrix $\mathbf{C}$. A normalized Mahalanobis distance combines a Mahalanobis distance (first term in (9), second term in (14)) and an expression related to the volume of the corresponding hyper-ellipsoid (third term in (9), third term in (14)). This volume term cannot be omitted since

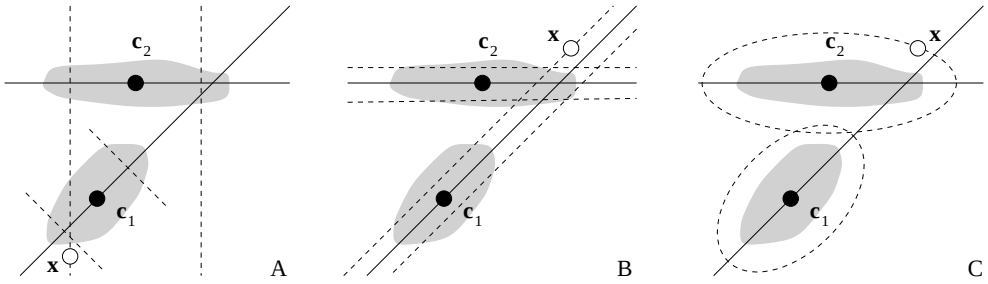


Fig. 10. Comparison between different distance measures ($n = 2$, $m = 1$). A: Normalized Mahalanobis distance in the principal subspace. B: Squared reconstruction error. C: Combination of the two measures. Filled circles depict the centers $\mathbf{c}_1$ and $\mathbf{c}_2$ of two local PCA units, the solid lines show their principal eigendirections, and the dashed curves visualize iso-distance curves. The two units are assumed to approximate two distributions indicated by the shaded regions. Open circles depict data vectors $\mathbf{x}$. The data vectors shown would be assigned to unit 2 in (A), to unit 1 in (B), and to unit 2 in (C).

minimizing a plain Mahalanobis distance would lead to an unlimited growth of the size of the hyper-ellipsoid.

Sung and Poggio [4] compute the normalized Mahalanobis distance for the m principal directions (first and fourth term of (12)), and the squared reconstruction error (13), but keep both measures separate in a two-value distance metric. Hinton *et al.* [5] add both measures as in (12), but with a relative weighting adjusted by a fixed parameter, whereas in our method the weighting depends on the current estimate of λ^*. The last term in (12) would be constant and identical for all units in the approach by Hinton *et al.* and can thus be omitted there, but has to be included here, since λ^* takes different values for each unit.

Geometrically, combining the normalized Mahalanobis distance in the principal subspace with the squared reconstruction error corresponds to complementing a hyper-ellipsoid in the principal subspace with a hyper-sphere in the minor subspace [5; 4]. Each measure alone would not be sufficient for the competition between the units, as is shown in figure 10 for an example with $n = 2$ and $m = 1$. If the normalized Mahalanobis distance is only computed in the principal subspace, iso-distance hyper-curves degenerate from hyper-ellipsoids to hyper-elliptic cylinders since the distance measure remains constant when moving in the *minor* subspace. In the two-dimensional case shown in figure 10 (A), the iso-distance curves are parallel lines (dashed). It is obvious from the figure that a data vector $\mathbf{x}$ (open circle) in large Euclidean distance from the center of a unit may unwantedly yield a small normalized Mahalanobis distance for this unit and therefore is wrongly assigned to that unit. The same holds if the reconstruction error alone is used as distance measure (B), with the difference that here the distance value stays constant when moving in the *principal* subspace. Only a combination of the two measures or a full normal-

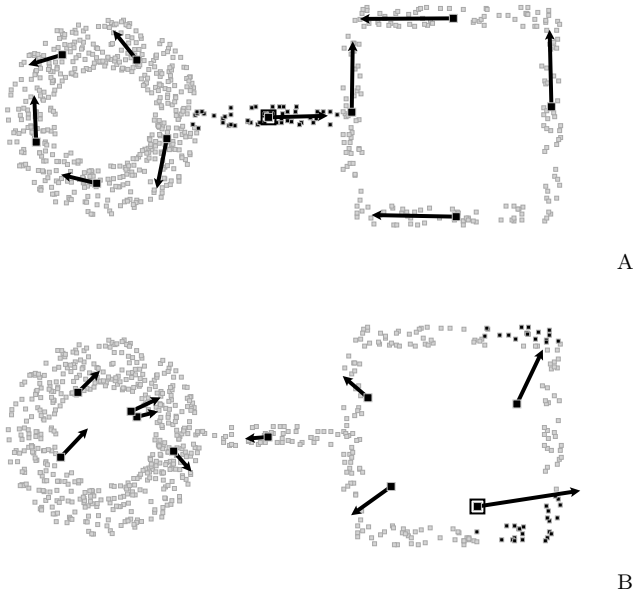


Fig. 11. Result of the training ($n = 2$, $m = 1$) for different distance measures used in the competition. A: Combination of normalized Mahalanobis distance and reconstruction error. B: Normalized Mahalanobis distance in the principal subspace alone. Arrows represent the principal eigenvector of each unit. Data points shown in black have the minimal distance to the unit with the center enclosed in the square.

ized Mahalanobis measure (C) will ensure a proper assignment to the units. This holds for both the competition in the training phase and for the application of the model. Figure 11 shows that using the normalized Mahalanobis distance in the principal subspace alone causes our method to fail: In diagram B, many unit centers lie outside the distribution, the principal eigenvectors are not aligned with the data, and distant patterns are assigned to the same unit.

5.2 Relation to Other Local PCA Methods

Local PCA methods previously suggested in the literature can be classified according to different criteria. A first dichotomy can be found between *block and online methods*. Block methods execute alternating steps of pattern assignment and local PCA (including the update of the centers), where the PCA step considers *all* patterns that were previously assigned to a region [5; 1; 4; 2]. Online methods work in a pattern-by-pattern regime: after the presentation of a single pattern, the pattern is assigned to the units, and the centers as well as the eigenvector and eigenvalue estimates are updated immediately [16; 13; 14]. A second dichotomy is given between *hard and soft assignment* (or competition). A data vector can be assigned exclusively to one unit or a set of units (hard assignment, [16; 5; 1; 4]), or the responsibility for a data vector can

be shared between all units according to weighting factors (soft assignment, [5; 2; 13; 8; 14]). A third dichotomy distinguishes between approaches derived from or motivated by a *probabilistic framework*, often based on "Expectation Maximization" (EM) [17] of Gaussian mixture models [18; 19; 20; 5; 21; 2; 12], and approaches based on a *deterministic framework*, usually a combination of vector quantization methods, PCA, and annealing techniques [16; 1; 4; 13; 14]. There is, however, a smooth transition and partially an equivalence between these two classes [21].

The method suggested here is an *online method* with *soft assignment* derived from a *deterministic framework*. In contrast, the majority of the methods, specifically those based on EM, are block methods. Clear advantages of either block or online methods cannot be accounted for so far. Soft assignment should definitely be preferred, since hard assignment is prone to premature convergence to sub-optimal solutions [6; 2]. The selection of a deterministic framework was mainly motivated by the advantageous properties of the Neural Gas vector quantization approach, specifically the robustness of its soft competition scheme, and by the existence of fast and robust neural principal component analyzers with online regime like RRLSA.

The method of Zhang *et al.* [13; 14] is like ours classified as online, soft, and deterministic, and also uses the soft competition scheme of Neural Gas. The method is, however, restricted to a special case where all local principal subspaces go through the origin, and can therefore not be used to approximate arbitrary distributions. An application to the synthetic test cases presented in section 4.1 would therefore fail. In the digit recognition task (section 4.2), Zhang's method produces good results, probably because the single principal eigenvector points towards those corners of the data hypercube, where the digits of the corresponding class are located. In Zhang's method, the ranking is based on the reconstruction error, and the update of the local principal subspaces is accomplished by a non-linear version of Oja's subspace rule [22]. With regard to the analysis in section 5.1 we assume that using the reconstruction error alone as distance measure is justified in their method since the subspaces go through the origin and an erroneous assignment like the one depicted in figure 10 (B) cannot occur.

6 Conclusion

The method presented in this work can be interpreted as an extension of the Neural Gas vector quantization method [6] to local PCA. It is an instance of a local PCA method with online learning. The robust soft competition mechanism of Neural Gas is used to avoid premature convergence. Competition between the units is guided by a weighted combination of the normalized

Mahalanobis distance in the major subspace and the reconstruction error, as was previously suggested [5]. A novel method was introduced which allows to estimate the weighting factor from the residual variance in the minor subspace. The performance of the method crucially depends on the core principal component analyzer which has to work robustly over several orders of magnitude in the eigenvalues. RRLSA [7] — a method derived from a recursive least squares approach — proved to be a good choice. Since all eigenvector and eigenvalue estimates are updated simultaneously, the PCA method had to be combined with a subsequent Gram-Schmidt step to avoid a collapse of orthogonality. Gram-Schmidt orthonormalization is costly, but interlocking of learning and orthonormalization can reduce the total computational effort [23] (see appendix A). In tests with low- and high-dimensional data our method proved to work robustly, and the classification quality in the digit recognition task is comparable to other local PCA methods. Besides the number of units and the number of principal eigenvectors, only the number of training steps, four training parameters (initial and final values of neighborhood range and global learning rate), and the initial values for the eigenvalue estimates and for the estimate of the residual variance have to be chosen. The distance measure could be motivated by both analytical and geometrical arguments. Ongoing research focuses on an application of the method to the learning of input-output relationships, specifically for kinematic robot arm models.

Acknowledgements

M.E. Tipping kindly provided the Matlab program used for the generation of the spiral data. The authors are grateful to P. Tavan for making available the code of the Generalized Radial Basis Function Network. R.M. thanks B. Herrnberger for comments on the manuscript and A. Könies for countless fruitful discussions.

A PCA Method

Local PCA in the presented approach is based on the "Robust Recursive Least Squares Learning Algorithm" (RRLSA) [7]. Least-squares methods for PCA converge faster than gradient-based algorithms, since they are not suffering from accuracy-speed trade-offs [24; 25]. RRLSA is a sequential network of single-neuron principal component analyzers based on deflation of the input vector [26]. In the following, we describe the update mechanism of a slightly modified version of RRLSA, written in equation (4) as $\mathbf{W}, \mathbf{\Lambda} \leftarrow \mathrm{PCA}\{\mathbf{W}, \mathbf{\Lambda}, \boldsymbol{\xi}, \alpha\}$ (unit indices k are omitted here). Note that for the RRLSA

method the eigenvalue estimates are not state variables, thus the update formally simplifies to $\mathbf{W}, \mathbf{\Lambda} \leftarrow \mathrm{PCA}\{\mathbf{W}, \boldsymbol{\xi}, \alpha\}$.

Internally, RRLSA uses an unnormalized version $\tilde{\mathbf{W}}$ of the weight matrix which is initialized with small random values. For each data vector $\boldsymbol{\xi}$, the output signals of all neurons are computed with the normalized matrix $\mathbf{W}$ (with unit-length column vectors $\mathbf{w}_i$, see below) from $\mathbf{y} = \mathbf{W}^T\boldsymbol{\xi}$. The update of the unnormalized weight vectors is done according to

$$\tilde{\mathbf{w}}_i \leftarrow \tilde{\mathbf{w}}_i + \alpha \cdot (\boldsymbol{\xi}^{(i)} y_i - \tilde{\mathbf{w}}_i), \quad i = 1, \ldots, m, \tag{A.1}$$

where m is the number of analyzer neurons, α is a learning rate, and $\boldsymbol{\xi}^{(i)}$ denotes the deflated input vector seen by neuron i. The deflated input vector is computed recursively from

$$\boldsymbol{\xi}^{(i+1)} = \boldsymbol{\xi}^{(i)} - \mathbf{w}_i y_i \quad \text{with} \quad \boldsymbol{\xi}^{(1)} = \boldsymbol{\xi}. \tag{A.2}$$

The eigenvalue and eigenvector estimates are then obtained from

$$\lambda_i = \|\tilde{\mathbf{w}}_i\|, \quad \mathbf{w}_i = \frac{\tilde{\mathbf{w}}_i}{\|\tilde{\mathbf{w}}_i\|}, \quad i = 1, \ldots, m. \tag{A.3}$$

For the application of RRLSA in the local PCA method, a subsequent orthogonalization of $\tilde{\mathbf{W}}$ based on the Gram-Schmidt procedure is needed. Since all eigenvectors have to be estimated simultaneously in each step, deflation is not sufficient to maintain the orthogonality of the weight matrix. If the previous eigenvector estimates are not exactly perpendicular, the deflated vector in the next network stage will have components in the subspace of these eigenvectors which may be larger than the components in the corresponding orthogonal subspace. The resulting loss of orthogonality causes the local PCA method to fail, since the distance measure relies on orthonormal weight vectors in $\mathbf{W}$. Gram-Schmidt orthonormalization is costly ($2nm^2$ operations), but when learning and orthonormalization are interlocked, the computational effort reduces by a factor of two [23]. All results presented in this paper were produced with the interlocked method. Since, in principle, orthonormality could deteriorate over time in the interlocked method (although this was only observed in pathological cases), an orthonormalization step using a Modified Gram-Schmidt procedure [27] was inserted after every 1 000th learning step.

B Proof of Minimum Property

The potential (8) can be written as

$$p(\mathbf{W}, \mathbf{\Lambda}) = \sum_{i=1}^{m} \mathbf{w}_i^T \mathbf{C} \mathbf{w}_i \lambda_i^{-1} - \sum_{i=1}^{m} \mathbf{w}_i^T \mathbf{w}_i + \sum_{i=1}^{m} \ln \lambda_i, \tag{B.1}$$

where $\mathbf{W}$ is an $n \times m$ matrix the columns of which are the m principal eigenvector estimates, and λ_i are the principal eigenvalue estimates. The candidate extreme points of this potential fulfill

$$\mathbf{C} \bar{\mathbf{w}}_i = \bar{\mathbf{w}}_i \bar{\lambda}_i \quad \forall\, i = 1, \ldots, n, \tag{B.2}$$

where $\bar{\mathbf{W}}$ contains m eigenvectors of $\mathbf{C}$ as its columns and $\bar{\mathbf{\Lambda}}$ is a diagonal matrix with m eigenvalues of $\mathbf{C}$ in the diagonal. In the following, we tacitly assume that all eigenvalues of $\mathbf{C}$ are pairwise different. We study the potential function on the orthonormality constraint $\mathbf{w}_i^T \mathbf{w}_j = \delta_{ij}$. Let $\mathbf{v}_i$ be a small deviation from $\bar{\mathbf{w}}_i$. With $\mathbf{w}_i = \bar{\mathbf{w}}_i + \mathbf{v}_i$, and $\bar{\mathbf{w}}_i^T \bar{\mathbf{w}}_j = \delta_{ij}$, the constraint can be written as

$$\bar{\mathbf{w}}_i^T \mathbf{v}_j + \bar{\mathbf{w}}_j^T \mathbf{v}_i + \mathbf{v}_i^T \mathbf{v}_j = \mathbf{0} \quad \forall\, i, j = 1, \ldots, m. \tag{B.3}$$

Moreover, let l_i be a small deviation from $\bar{\lambda}_i$, so $\lambda_i = \bar{\lambda}_i + l_i$. Considering the constraint $\mathbf{w}_i^T \mathbf{w}_j = \delta_{ij}$, the potential in one of the candidate points is $p(\bar{\mathbf{W}}, \bar{\mathbf{\Lambda}}) = \sum_{i=1}^{m} \ln \bar{\lambda}_i$, while the potential in an arbitrary other point defined by $\mathbf{v}_i$ and l_i is

$$p(\mathbf{W}, \mathbf{\Lambda}) = \sum_{i=1}^{m} (\bar{\mathbf{w}}_i + \mathbf{v}_i)^T \mathbf{C} (\bar{\mathbf{w}}_i + \mathbf{v}_i)(\bar{\lambda}_i + l_i)^{-1} - m + \sum_{i=1}^{m} \ln(\bar{\lambda}_i + l_i) \tag{B.4}$$

$$= \sum_{i=1}^{m} (\bar{\lambda}_i - \bar{\lambda}_i \mathbf{v}_i^T \mathbf{v}_i + \mathbf{v}_i^T \mathbf{C} \mathbf{v}_i)(\bar{\lambda}_i + l_i)^{-1} - m + \sum_{i=1}^{m} \ln(\bar{\lambda}_i + l_i), \tag{B.5}$$

where (B.2) and (B.3) were applied. From a Taylor expansion up to second-order terms we obtain the approximations $(\bar{\lambda}_i + l_i)^{-1} \approx \bar{\lambda}_i^{-1}(1 - \bar{\lambda}_i^{-1} l_i + \bar{\lambda}_i^{-2} l_i^2)$ and $\ln(\bar{\lambda}_i + l_i) \approx \ln \bar{\lambda}_i + \bar{\lambda}_i^{-1} l_i - \frac{1}{2} \bar{\lambda}_i^{-2} l_i^2$. We insert these into (B.5), omit third- and higher-order terms of the small deviations, and find the following expression for the potential difference

$$\Delta p = p(\mathbf{W}, \mathbf{\Lambda}) - p(\bar{\mathbf{W}}, \bar{\mathbf{\Lambda}}) = \underbrace{\sum_{i=1}^{m} \tfrac{1}{2} \bar{\lambda}_i^{-2} l_i^2}_{L} + \underbrace{\sum_{i=1}^{m} \left(\mathbf{v}_i^T \mathbf{C} \mathbf{v}_i \bar{\lambda}_i^{-1} - \mathbf{v}_i^T \mathbf{v}_i \right)}_{S}. \tag{B.6}$$

L is always non-negative, so we analyze the sign of S. After inserting the spectral expansion $\mathbf{C} = \sum_{j=1}^{n} \bar{\lambda}_j \bar{\mathbf{w}}_j \bar{\mathbf{w}}_j^T$ and $\mathbf{I} = \sum_{j=1}^{n} \bar{\mathbf{w}}_j \bar{\mathbf{w}}_j^T$ we obtain

$$S = \sum_{i=1}^{m} \sum_{j=1}^{n} (\mathbf{v}_i^T \bar{\mathbf{w}}_j)^2 (\bar{\lambda}_j \bar{\lambda}_i^{-1} - 1). \tag{B.7}$$

We split S into two sums

$$S = \underbrace{\sum_{i=1}^{m}\sum_{j=1}^{m}(\mathbf{v}_i^T\bar{\mathbf{w}}_j)^2(\bar{\lambda}_j\bar{\lambda}_i^{-1}-1)}_{T} + \underbrace{\sum_{i=1}^{m}\sum_{j=m+1}^{n}(\mathbf{v}_i^T\bar{\mathbf{w}}_j)^2(\bar{\lambda}_j\bar{\lambda}_i^{-1}-1)}_{U}, \tag{B.8}$$

where T is again split into

$$T = \sum_{i=1}^{m-1}\sum_{j=i+1}^{m}(\mathbf{v}_i^T\bar{\mathbf{w}}_j)^2(\bar{\lambda}_j\bar{\lambda}_i^{-1}-1) + \sum_{j=1}^{m-1}\sum_{i=j+1}^{m}(\mathbf{v}_i^T\bar{\mathbf{w}}_j)^2(\bar{\lambda}_j\bar{\lambda}_i^{-1}-1). \tag{B.9}$$

For small deviations $\mathbf{v}_i$ we can use the following first order approximation derived from (B.3):

$$\bar{\mathbf{w}}_i^T\mathbf{v}_j \approx -\bar{\mathbf{w}}_j^T\mathbf{v}_i. \tag{B.10}$$

After exchanging indices in the second sum of (B.9), inserting (B.10), and fusing both sums, it turns into

$$T \approx \sum_{i=1}^{m-1}\sum_{j=i+1}^{m}(\mathbf{v}_i^T\bar{\mathbf{w}}_j)^2\frac{(\bar{\lambda}_i-\bar{\lambda}_j)^2}{\bar{\lambda}_i\bar{\lambda}_j} \geq 0. \tag{B.11}$$

T is always non-negative, and U is non-negative for all deviations if and only if

$$\bar{\lambda}_j > \bar{\lambda}_i \quad \forall\, j = m+1,\ldots,n,\ i = 1,\ldots,m, \tag{B.12}$$

thus the potential has a minimum in the minor eigenvectors and eigenvalues of $\mathbf{C}$.

Under (B.12), this minimum is *strict*, since all non-vanishing deviations from the candidate extreme point entail $\Delta p > 0$. First case: For any deviation in λ_i, i.e. $l_i \neq 0$, we have $L > 0$ and thus $\Delta p > 0$. Second case: Let us assume that $l_i = 0\ \forall i$, for which $\Delta p = S$. We show by contradiction that $S \neq 0$. By combining (B.8) with (B.11), (B.12), and (B.3) we conclude that the case $S = 0$ requires $\mathbf{v}_i^T\bar{\mathbf{w}}_j = 0\ \forall j = 1,\ldots,n,\ i = 1,\ldots,m$. Since the $\bar{\mathbf{w}}_i$ span the entire $\mathbb{R}^n$, and each non-vanishing $\mathbf{v}_i$ would have to be perpendicular to this space, this can only mean $\mathbf{v}_i = \mathbf{0}\ \forall i$. This contradiction proves $\Delta p > 0$ also for the second case.

If (B.12) is violated, so that $\exists\, j^* \in \{m+1,\ldots,n\},\ i^* \in \{1,\ldots,m\} : \bar{\lambda}_{j^*} < \bar{\lambda}_{i^*}$, we can choose the deviations $l_i = 0\ \forall i$, and $\mathbf{v}_i$ so that $\mathbf{v}_{i^*}^T\bar{\mathbf{w}}_{j^*} \neq 0$, while $\mathbf{v}_i^T\bar{\mathbf{w}}_j = 0$ for all other i, j. In this case we get $\Delta p < 0$ for some deviations, so the candidate extreme point turns out to be a saddle.

C Computation of the log-likelihood

From our distance measure (3) we obtain a Gaussian probability density function $p_k(\mathbf{x}) = (2\pi)^{-\frac{n}{2}} \exp\{-\frac{1}{2} d_k(\mathbf{x})\}$ for each unit k. The prior probability or mixing parameter [21] of each unit is estimated from $P_k = \nu_k/\nu$, where ν_k is the number of training patterns assigned to unit k, and ν is the total number of training patterns. The probability density function of a Gaussian mixture model with N units is $p(\mathbf{x}) = \sum_{k=1}^{N} P_k\, p_k(\mathbf{x})$. The log-likelihood per pattern $\mathcal{L}$ is obtained from $\mathcal{L} = \frac{1}{\nu} \sum_{i=1}^{\nu} \log p(\mathbf{x}_i)$.

D Synthetic distributions

Each call to $u(l, r)$ draws a random number from a uniform distribution between l and r. Each call to $g(m, d)$ draws a random number from a normal distribution with mean m and standard deviation d. Each call to $b(M)$ returns a number from the discrete set M with equal probability. The coordinates of the generated data point are denoted by (x, y) (2D) or (x, y, z) (3D).

Vortex distribution *Parameter:* maximal angle $\hat{\alpha} = 6\pi$, maximal radius $\hat{r} = 120$, noise $n = 20$. *Data point generation:* angle $\alpha = u(0, \hat{\alpha})$, radius $r = \hat{r} * \alpha/\hat{\alpha} + u(0, n)$, coordinates $x = r\cos\alpha$, $y = r\sin\alpha$.

Ring-line-square distribution *Parameter:* Ring: inner radius $\check{r} = 50$, outer radius $\hat{r} = 100$. Line: length $l = 150$, noise $n = 20$. Square: side length $s = 200$, noise $n = 20$. *Data point generation:* Ring: radius $r = u(\check{r}, \hat{r})$, angle $\alpha = u(0, 2\pi)$, coordinates $x = r\cos\alpha$, $y = r\sin\alpha$. Line: coordinates $x = \hat{r} + u(0, l)$, $y = u(-n/2, n/2)$. Square: $v_1 = b(\{0, 1\}) \cdot s + u(-n/2, n/2)$, $v_2 = u(0, s)$, if $b(\{0, 1\}) = 0$ the coordinates are $x = \hat{r} + l + v_1$, $y = -\hat{r} + v_2$, otherwise $x = \hat{r} + l + v_2$, $y = -\hat{r} + v_1$.

Spiral distribution 1 *Parameter:* maximal angle $\hat{\alpha} = 2\pi$, radius $r = 120$, length $l = 500$, noise $n = 20$. *Data point generation:* point index $k = 0, \ldots, K-1$, total number of points K, $\tilde{x} = k \cdot l/K$, $\tilde{\alpha} = k \cdot \hat{\alpha}/K$, coordinates $x = \tilde{x} + u(-n/2, n/2)$, $y = r\cos\tilde{\alpha} + u(-n/2, n/2)$, $z = r\sin\tilde{\alpha} + u(-n/2, n/2)$.

Spiral distribution 2 *Parameter:* noise $n = 0.1$. *Data point generation:* $x = u(0, 1) \cdot 4\pi + g(0, n)$, $y = \sin x + g(0, n)$, $z = \cos x + g(0, n)$.

References

[1] N. Kambhatla, T. K. Leen, Dimension reduction by local principal component analysis, Neural Computation 9 (7) (1997) 1493–1516.

[2] M. E. Tipping, C. M. Bishop, Mixtures of probabilistic principal component analyzers, Neural Computation 11 (2) (1999) 443–482.

[3] A. Weingessel, K. Hornik, Local PCA algorithms, IEEE Transactions on Neural Networks 11 (6) (2000) 1242–1250.
[4] K. K. Sung, T. Poggio, Example-based learning for view-based human face detection, IEEE Transactions on Pattern Analysis and Machine Intelligence 20 (1) (1998) 39–51.
[5] G. E. Hinton, P. Dayan, M. Revow, Modeling the manifolds of images of handwritten digits, IEEE Transactions on Neural Networks 8 (1) (1997) 65–74.
[6] T. M. Martinetz, S. G. Berkovich, K. J. Schulten, "Neural-gas" network for vector quantization and its application to time-series prediction, IEEE Transactions on Neural Networks 4 (4) (1993) 558–569.
[7] S. Ouyang, Z. Bao, G.-S. Liao, Robust recursive least squares algorithm for principal component analysis, IEEE Transactions on Neural Networks 11 (1) (2000) 215–221.
[8] S. Albrecht, J. Busch, M. Kloppenburg, F. Metze, P. Tavan, Generalized radial basis function networks for classification and novelty detection: self-organization of optimal Bayesian decision, Neural Networks 13 (10) (2000) 1075–1093.
[9] I. Nabney, C. Bishop, Netlab neural network software, Neural Computing Research Group, Aston University, http://www.ncrg.aston.ac.uk/netlab/ (2002).
[10] Y. LeCun, L. Jackel, L. Bottou, A. Brunot, C. Cortes, J. Denker, H. Drucker, I. Guyon, U. Müller, E. Säckinger, P. Simard, V. Vapnik, Comparison of learning algorithms for handwritten digit recognition, in: F. Fogelman, P. Gallinari (Eds.), Proceedings of the International Conference on Artificial Neural Networks, 1995, pp. 53–60.
[11] Y. LeCun, The MNIST database of handwritten digits, NEC Research Institute, http://yann.lecun.com/exdb/mnist/index.html (1998).
[12] P. Meinicke, H. Ritter, Resolution-based complexity control for Gaussian mixture models, Neural Computation 13 (2) (2001) 453–475.
[13] B. Zhang, M. Fu, H. Yan, Handwritten digit recognition by a mixture of local principal component analysis, Neural Processing Letters 8 (3) (1998) 241–252.
[14] B. Zhang, M. Fu, H. Yan, A nonlinear neural network model of mixture of local principal component analysis: application to handwritten digits recognition, Pattern Recognition 34 (2) (2001) 203–214.
[15] P. J. B. Hancock, R. J. Baddeley, L. S. Smith, The principal components of natural images, Network 3 (1) (1992) 61–70.
[16] R. D. Dony, S. Haykin, Image segmentation using a mixture of principal components representation, IEE Proceedings — Vision, Image, Signal Processing 144 (2) (1997) 73–80.
[17] A. P. Dempster, N. M. Laird, D. B. Rubin, Maximum likelihood from incomplete data via EM algorithm, Journal of the Royal Statistical Society Series B 39 (1) (1977) 1–38.
[18] C. Bregler, S. M. Omohundro, Nonlinear image interpolation using mani-

fold learning, in: G. Tesauro, D. Touretzky, T. Leen (Eds.), Advances in Neural Information Processing Systems 7, MIT Press, Cambridge, MA, 1995, pp. 973–980.
[19] M. I. Jordan, L. Xu, Convergence results for the EM approach to mixtures of experts architectures, Neural Networks 8 (9) (1995) 1409–1431.
[20] L. Xu, M. I. Jordan, On convergence properties of the EM algorithm for Gaussian mixtures, Neural Computation 8 (1) (1996) 129–151.
[21] E. Alpaydin, Soft vector quantization and the EM algorithm, Neural Networks 11 (3) (1998) 467–477.
[22] E. Oja, Neural networks, principal components, and subspaces, International Journal of Neural Systems 1 (1) (1989) 61–68.
[23] R. Möller, Interlocking of learning and orthonormalization in RRLSA, Neurocomputing 49 (1-4) (2002) 429–433.
[24] S. Bannour, M. R. Azimi-Sadjadi, Principal component extraction using recursive least squares learning, IEEE Transactions on Neural Networks 6 (2) (1995) 457–469.
[25] K. I. Diamantaras, S. Y. Kung, Principal Component Neural Networks. Theory and Applications, John Wiley & Sons, 1996.
[26] T. D. Sanger, Optimal unsupervised learning in a single-layer linear feedforward neural network, Neural Networks 2 (6) (1989) 459–473.
[27] G. H. Golub, C. F. van Loan, Matrix Computations, 3rd Edition, Johns Hopkins University Press, Baltimore and London, 1996.

Ralf Möller received a Ph.D. in electrical engineering from the Technical University of Ilmenau, Germany, and a Venia legendi in computer science from the University of Zurich, Switzerland. He is heading the Cognitive Robotics group at the Max Planck Institute for Psychological Research. His research interests include behavior-based approaches to visual cognition, visual navigation, biorobotics, neuromorphic systems, and parallel computation.

Heiko Hoffmann is currently a Ph.D. student in the Cognitive Robotics group at the Max Planck Institute for Psychological Research. He received his diploma degree in physics from the University of Heidelberg. His research interests include recurrent neural networks, unsupervised learning, and vision motor interaction.