Covariance and Correlation Matrix

Given sample $\{\mathbf{x}_n\}_1^N$, where $\mathbf{x} \in \mathbb{R}^d$, $\mathbf{x}_n = \begin{bmatrix} x_{1n} \\ x_{2n} \\ \vdots \\ x_{dn} \end{bmatrix}$

- sample mean $\overline{\mathbf{x}} = \frac{1}{N}\sum_{n=1}^{N} \mathbf{x}_n$, and entries of sample mean are $\overline{x}_i = \frac{1}{N}\sum_{n=1}^{N} x_{in}$
- sample covariance matrix is a $d \times d$ matrix $\mathbf{Z}$ with entries $\mathbf{Z}_{ij} = \frac{1}{N-1}\sum_{n=1}^{N}(x_{in} - \overline{x}_i)(x_{jn} - \overline{x}_j)$
- sample correlation matrix is a $d \times d$ matrix $\mathbf{C}$ with entries $\mathbf{C}_{ij} = \frac{1}{N-1}\frac{\sum_{n=1}^{N}(x_{in}-\overline{x}_i)(x_{jn}-\overline{x}_j)}{\sigma_{x_i}\sigma_{x_j}}$, where σ_{x_i} and σ_{x_j} are the sample standard deviations

Covariance and Correlation Matrix Example

Given sample:

$$\begin{bmatrix} 1.2 \\ 0.9 \end{bmatrix}, \begin{bmatrix} 2.5 \\ 3.9 \end{bmatrix}, \begin{bmatrix} 0.7 \\ 0.4 \end{bmatrix}, \begin{bmatrix} 4.2 \\ 5.8 \end{bmatrix}, \qquad \overline{\mathbf{x}} = \begin{bmatrix} 2.15 \\ 2.75 \end{bmatrix}$$

$$\mathbf{Z} = \begin{bmatrix} 2.443333 & 3.940000 \\ 3.940000 & 6.523333 \end{bmatrix}$$

$$\mathbf{C} = \begin{bmatrix} \frac{2.443333}{1.563117 \cdot 1.563117} & \frac{3.940000}{1.563117 \cdot 2.554082} \\ \frac{3.940000}{2.554082 \cdot 1.563117} & \frac{6.523333}{2.554082 \cdot 2.554082} \end{bmatrix} = \begin{bmatrix} 1.000000 & 0.986893 \\ 0.986893 & 1.000000 \end{bmatrix}$$

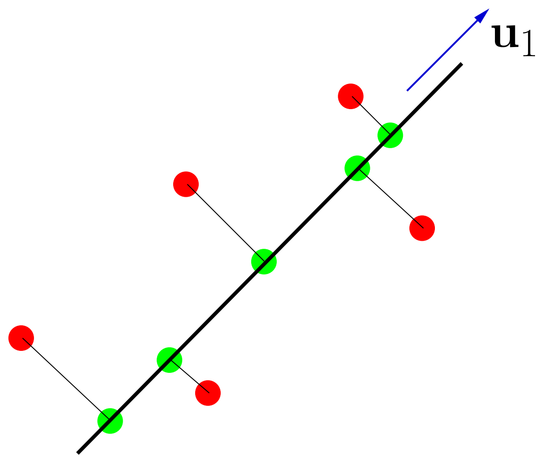
Observe, if sample is z-normalized ($x_{ij}^{\text{new}} = \frac{x_{ij} - \overline{x}_i}{\sigma_{x_i}}$, mean 0, standard deviation 1) then $\mathbf{C}$ equals $\mathbf{Z}$.
See `cov()`, `cor()`, `scale()` in R.

Principal Component Analysis with NN

Principal Component Analysis (PCA) is a technique for

- dimensionality reduction
- lossy data compression
- feature extraction
- data visualization

Idea: orthogonal projection of the data onto a lower dimensional linear space, such that the variance of the projected data is maximized.



Maximize Variance of Projected Data

Given data $\{\mathbf{x}_n\}_1^N$ where $\mathbf{x}_n$ has dimensionality d.

- Goal: project data onto a space having dimensionality $m < d$ while maximizing the variance of the projected data.

Let us consider the projection onto one-dimensional space $(m = 1)$.

- Define direction of this space using a d-dimensional vector $\mathbf{u}_1$.

Mean of the projected data is $\mathbf{u}_1^T \overline{\mathbf{x}}$, where $\overline{\mathbf{x}}$ is sample mean

$$\overline{\mathbf{x}} = \frac{1}{N} \sum_{n=1}^{N} \mathbf{x}_n$$

Maximize Variance of Projected Data (cont.)

Variance of projected data is given by

$$\frac{1}{N}\sum_{n=1}^{N}\left(\mathbf{u}_1^T\mathbf{x}_n - \mathbf{u}_1^T\overline{\mathbf{x}}\right)^2 = \mathbf{u}_1^T\mathbf{S}\mathbf{u}_1$$

where $\mathbf{S}$ is the data covariance matrix defined by

$$\mathbf{S} = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n - \overline{\mathbf{x}})(\mathbf{x}_n - \overline{\mathbf{x}})^T$$

Goal: maximize the projected variance $\mathbf{u}_1^T\mathbf{S}\mathbf{u}_1$ with respect to $\mathbf{u}_1$. Prevent $\|\mathbf{u}_1\|$ growing to infinity, use constrain $\mathbf{u}_1^T\mathbf{u}_1 = 1$, gives optimization problem:

$$\begin{array}{ll} \text{maximize} & \mathbf{u}_1^T\mathbf{S}\mathbf{u}_1 \\ \text{subject to} & \mathbf{u}_1^T\mathbf{u}_1 = 1 \end{array}$$

Maximize Variance of Projected Data (cont.)

Lagrangian form (one Lagrange multiplier λ_1):

$$L(\mathbf{u}_1, \lambda_1) = \mathbf{u}_1^T \mathbf{S} \mathbf{u}_1 - \lambda_1(\mathbf{u}_1^T \mathbf{u}_1 - 1)$$

Set derivative with respect to $\mathbf{u}_1$ to zero,

$$\frac{\partial L(\mathbf{u}_1, \lambda_1)}{\partial \mathbf{u}_1} = \mathbf{0} \quad \text{gives} \quad \mathbf{S}\mathbf{u}_1 = \lambda_1 \mathbf{u}_1$$

last term says that $\mathbf{u}_1$ must be an eigenvector of $\mathbf{S}$. Finally by left-multiplying by $\mathbf{u}_1^T$ and making use of $\mathbf{u}_1^T \mathbf{u}_1 = 1$ one can see that the variance is given by

$$\mathbf{u}_1^T \mathbf{S} \mathbf{u}_1 = \lambda_1.$$

Observe, that variance is maximized when $\mathbf{u}_1$ equals to the eigenvector having largest eigenvalue λ_1.

Second Principal Component

Second eigenvector $\mathbf{u}_2$ should also be of unit length and orthogonal to $\mathbf{u}_1$ (after projection uncorrelated to $\mathbf{u}_1^T\mathbf{x}$).

$$\begin{array}{rl} \text{maximize} & \mathbf{u}_2^T\mathbf{S}\mathbf{u}_2 \\ \text{subject to} & \mathbf{u}_2^T\mathbf{u}_2 = 1, \quad \mathbf{u}_2^T\mathbf{u}_1 = 0 \end{array}$$

Lagrangian form (two Lagrange multipliers λ_1, λ_2):

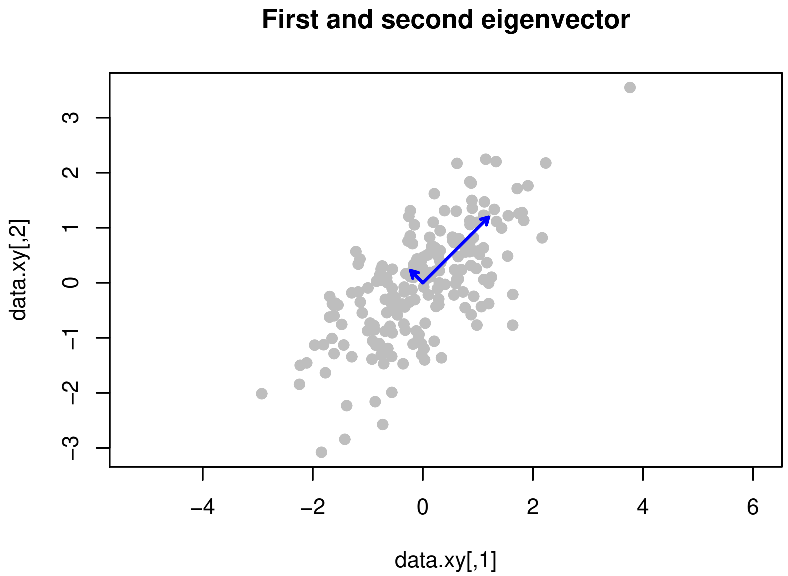
$$L(\mathbf{u}_2, \lambda_1, \lambda_2) = \mathbf{u}_2^T\mathbf{S}\mathbf{u}_2 - \lambda_2(\mathbf{u}_2^T\mathbf{u}_2 - 1) - \lambda_1(\mathbf{u}_2^T\mathbf{u}_1 - 0)$$

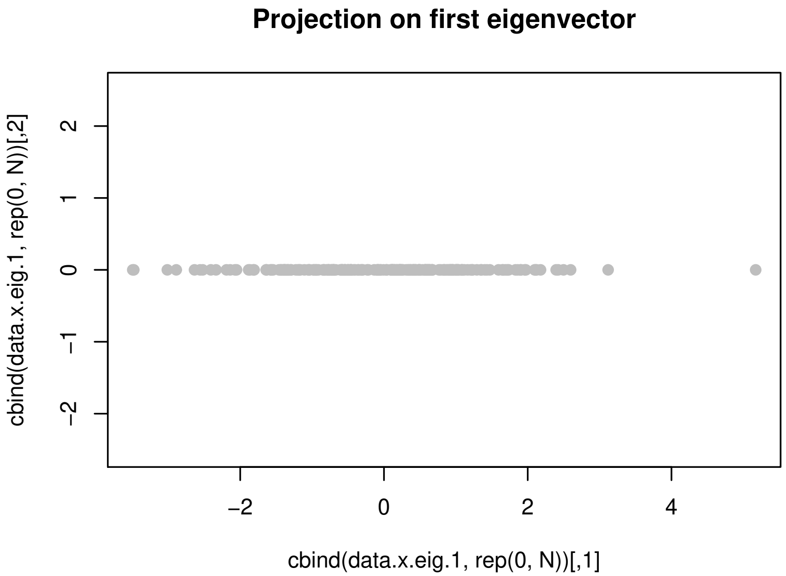
This gives solution

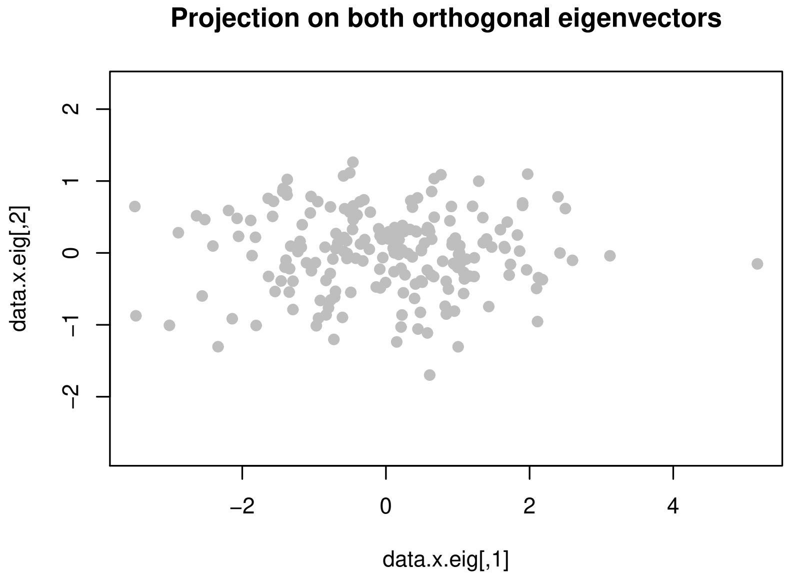
$$\mathbf{u}_2^T\mathbf{S}\mathbf{u}_2 = \lambda_2$$

which implies that $\mathbf{u}_2$ should be eigenvector of $\mathbf{S}$ with second largest eigenvalue λ_2. Other dimensions are given by the eigenvectors with decreasing eigenvalues.

PCA Example


First and second eigenvector
data.xy[,2]
data.xy[,1]
3
2
1
0
−1
−2
−3
−4
−2
0
2
4
6


Projection on first eigenvector
cbind(data.x.eig.1, rep(0, N))[,2]
cbind(data.x.eig.1, rep(0, N))[,1]
2
1
0
−1
−2
−2
0
2
4

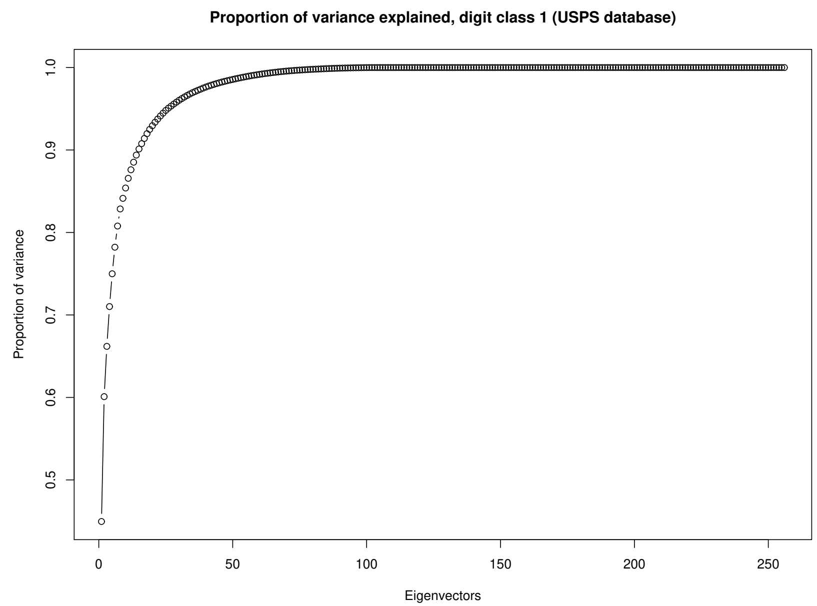

Projection on both orthogonal eigenvectors
data.x.eig[,2]
data.x.eig[,1]
2
1
0
−1
−2
−2
0
2
4

Proportion of Variance

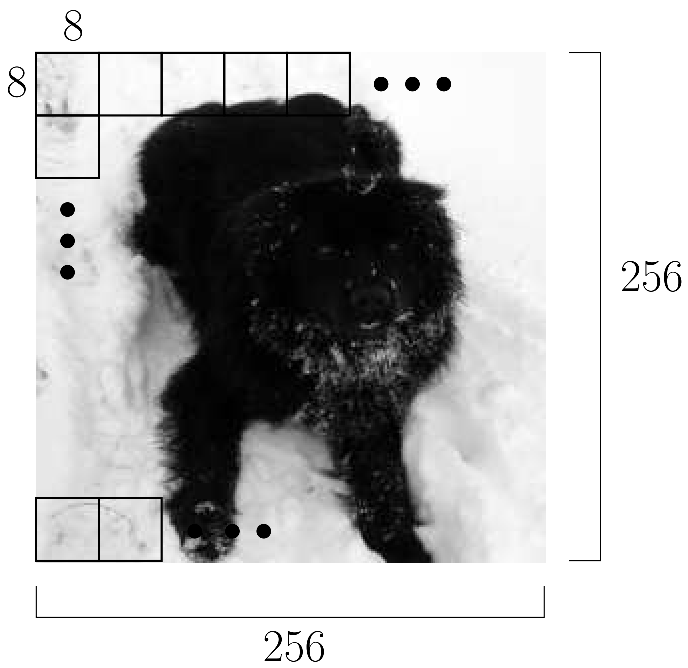
- In image and speech processing problems the inputs are usually highly correlated.

If dimensions are highly correlated, then there will be *small* number of eigenvectors with large eigenvalues ($m \ll d$). As a result, a large reduction in dimensionality can be attained.

$$\frac{\lambda_1 + \lambda_2 + \ldots + \lambda_m}{\lambda_1 + \lambda_2 + \ldots + \lambda_m + \ldots + \lambda_d}$$



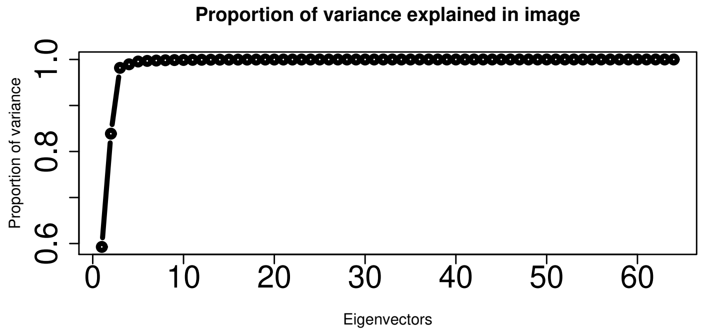
PCA Second Example

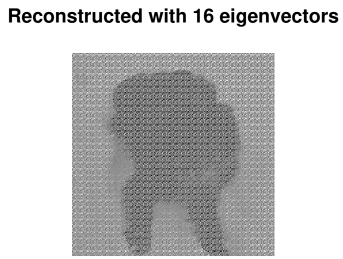


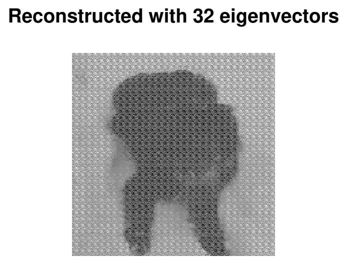
- Segment image in $32 \cdot 32 = 1024$ image pieces of size $8 \times 8 \equiv 1 \times 64$: $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{1024} \in \mathbb{R}^{64}$
- Determine mean: $\bar{\mathbf{x}} = \frac{1}{1024} \sum_{n=1}^{1024} \mathbf{x}_i$
- Determine covariance matrix $\mathbf{S}$ and the m eigenvectors $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_m$ having the largest corresponding eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_m$
- Create eigenvector matrix $\mathbf{U}$, where $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_m$ are column vectors
- Project image pieces $\mathbf{x}_i$ into subspace as follows: $\mathbf{z}_i^T = \mathbf{U}^T(\mathbf{x}_i^T - \bar{\mathbf{x}}^T)$

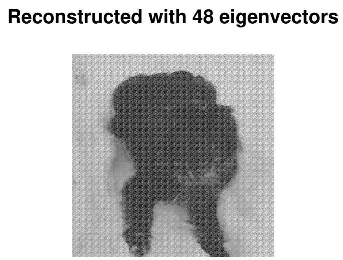
PCA Second Example (cont.)

- Reconstruct image pieces by back-projecting it to the original space as $\widetilde{\mathbf{x}}_i^T = \mathbf{U}\mathbf{z}_i^T + \overline{\mathbf{x}}$. Note, mean is added (substracted step before) because data is not normalized

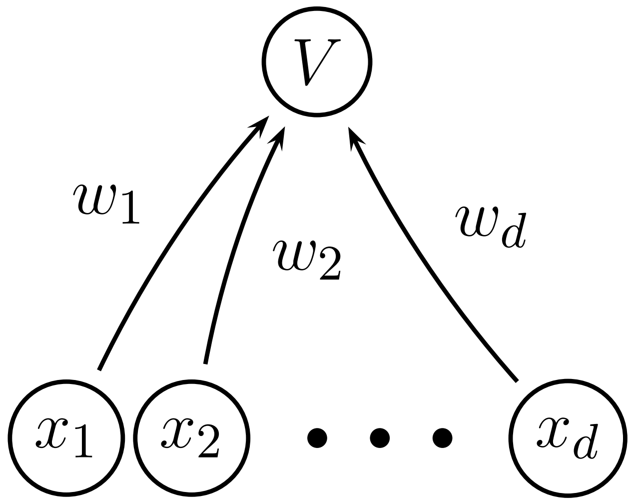








PCA with a Neural Network



$$V = \mathbf{w}^T \mathbf{x} = \sum_{j=1}^{d} w_j x_j$$

Apply Hebbian learning rule

$$\Delta w_i = \eta V x_i,$$

such that after some update steps weight vector $\mathbf{w}$ should point in direction of maximum variance.

PCA with a Neural Network (cont.)

Suppose that there is a stable equilibrium point for $\mathbf{w}$ such that the *average* weight change is zero

$$0 = \langle \Delta w_i \rangle = \langle V x_i \rangle = \langle V \sum_j w_j x_j x_i \rangle = \sum_j \mathbf{C}_{ij} w_j = \mathbf{C}\mathbf{w}.$$

Angle brackets $\langle \cdot \rangle$ indicates an average over the input distribution $P(\mathbf{x})$ and $\mathbf{C}$ denotes the *correlation matrix* with

$$\mathbf{C}_{ij} \equiv \langle x_i \, x_j \rangle, \qquad \text{or} \qquad \mathbf{C} \equiv \langle \mathbf{x} \, \mathbf{x}^T \rangle$$

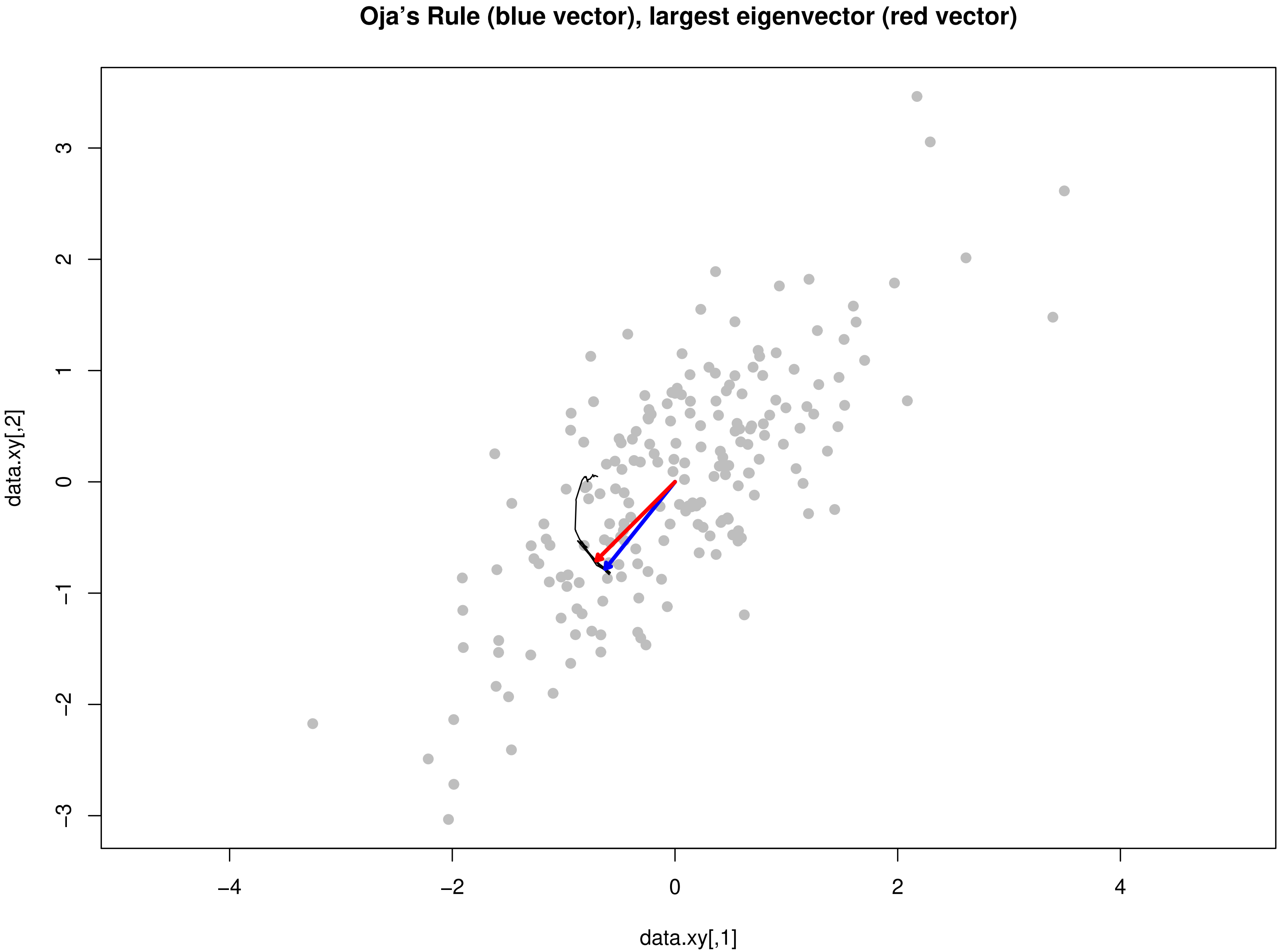
Note, $\mathbf{C}$ is symmetric ($\mathbf{C}_{ij} = \mathbf{C}_{ji}$) and positive semi-definite which implies that its eigenvalues are positive or zero and eigenvectors can be taken as orthogonal.

PCA with a Neural Network (cont.)

- At our hypothetical equilibrium point, $\mathbf{w}$ is an eigenvector of $\mathbf{C}$ with eigenvalue 0
- Never stable, because $\mathbf{C}$ has some pos. eigenvalues and some corresponding eigenvector would grow exponentially
- constrain the growth of $\mathbf{w}$, e.g. renormalization ($\|\mathbf{w}\| = 1$) after each update step
- more elegant idea: adding a weight decay proportional to V^2 to Hebbian learning rule (Oja's Rule)

$$\Delta w_i = \eta V (x_i - V w_i)$$

PCA with a Neural Network Example



Some insights into Oja's Rule

Oja's rule converges to a weight vector $\mathbf{w}$ with following properties:

- $\|\mathbf{w}\| = 1$ (unit length),
- eigenvector direction: $\mathbf{w}$ lies in a maximal eigenvector direction of $\mathbf{C}$,
- variance maximization: $\mathbf{w}$ lies in a direction that maximizes $\langle V^2 \rangle$

Oja's learning rule is still limited, because we can construct only the first principal component of the z-normalized data.

Construct the first m principal components

- Single-layer network with the i-th output V_i given by $V_i = \sum_j w_{ij} x_j = \mathbf{w}_i^T \mathbf{x}$, $\mathbf{w}_i$ is the weight vector for the i-th output
- Oja's m-unit learning rule

$$\Delta w_{ij} = \eta V_i (x_j - \sum_{k=1}^{d} V_k w_{kj})$$

- Sanger's learning rule

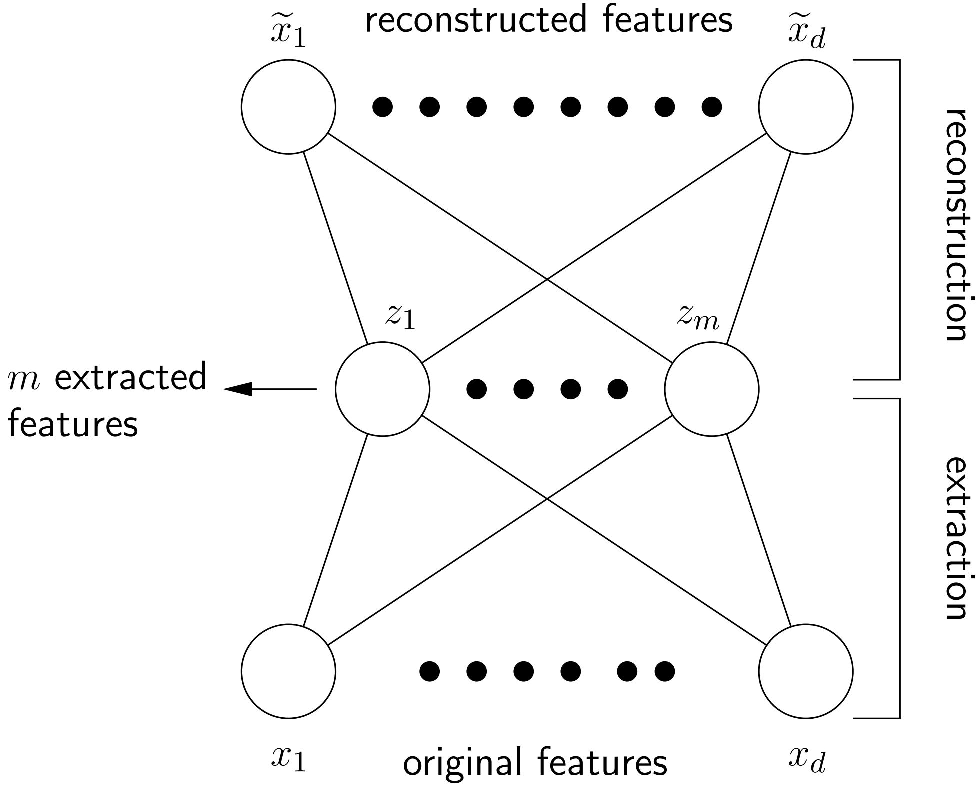
$$\Delta w_{ij} = \eta V_i (x_j - \sum_{k=1}^{i} V_k w_{kj})$$

- Both rules reduce to Oja's 1-unit rule for the $m = 1$ and $i = 1$ case

Oja's and Sanger's Rule

- In both cases the $\mathbf{w}_i$ vectors converge to orthogonal unit vectors
- Weight vectors become in Sanger's rule exactly the first m principal components, in order $\mathbf{w}_i = \pm\mathbf{c}^i$, where $\mathbf{c}^i$ is normalized eigenvector of the correlation matrix $\mathbf{C}$ belonging to the i-th largest eigenvalue λ_i
- Oja's m-unit rule converges to the m weight vectors that span the same subspace as the first m eigenvectors, but do not find the eigenvector directions themselves

Linear Auto-Associative Network



- Network is training to perform identity mapping
- Idea: bottleneck units represents significant features in the input data
- Train network by minimizing the sum-of-square error

$\frac{1}{2} \sum_{n=1}^{N} \sum_{k=1}^{d} \left(y_k(\mathbf{x}^{(n)}) - x_k^{(n)}) \right)^2$

Linear Auto-Associative Network (cont.)

- Equivalent to the Oja's/Sanger's update rule, this type of learning can be considered as unsupervised learning, since no independent target data is provided
- Error function has a unique global minimum when hidden units have linear activations functions
- At this minimum the network performs a projection onto the m-dimensional sub-space which is spanned by the first m principal components of the data
- Note, however, that these vectors need not to be orthogonal or normalized

Non-Linear Auto-Associative Network