CSE-E4810 Machine Learning and Neural Networks (5 cr)

Lecture 3: Data Preprocessing and Principal Component Analysis

Prof. Juha Karhunen

https://mycourses.aalto.fi/course/view.php?id=13086

Hebbian learning

- Hebbian learning is based on neurobiological considerations, originating from the pioneering work of D. Hebb (1949).
- It is discussed in Du's and Swamy's book in subsection 12.1.1.
- In its simplest form, Hebbian learning can be formulated as follows.
- Denote again by $x_j(k)$ the j:th input signal to a neuron, and by $y(k)$ the neuron's output at the same (discrete) time instant k.
- When the input and output signal are:
  - positively correlated, the strength w_j of their synaptic connection should be increased;
  - negatively correlated, w_j should be decreased.
- This principle yields the **simple Hebbian learning rule**

$$w_j(k+1) = w_j(k) + \mu_k y(k) x_j(k) \tag{1}$$

or in the vector form (recalling that $y(k) = \mathbf{w}(k)^T \mathbf{x}(k)$)

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \mu_k y(k) \mathbf{x}(k) \tag{2}$$

- Again, the learning parameter $\mu_k > 0$ determines how large an update is made on iteration k.
- The simple Hebbian learning rule (1) is not yet mathematically satisfactory.
- This is because it leads to unlimited growth of the synaptic weight vector $\mathbf{w}(k)$ when $k \to \infty$.
- To prevent this, some kind of normalization (or saturation) of the weight vector $\mathbf{w}(k)$ is needed.
- **Anti-Hebbian learning** (Section 12.5) is just the reverse of Hebbian learning, with μ_k replaced by $-\mu_k$ in (1) and (2).

Oja's learning rule

- Oja's learning rule demonstrates the usefulness of Hebbian learning.
- It was developed by E. Oja in our laboratory in 1982.
- We discuss it here in more detail than in subsection 12.1.2 in Du's and Swamy's book.
- Oja's rule will be discussed further in the exercises.
- Oja's rule uses a **single linear neuron** with the input vector $\mathbf{x}$.
- The output of this neuron is simply $y = v = \mathbf{w}^T \mathbf{x}$, where $\mathbf{w}$ is the weight vector of the neuron and v its linear response.
- The goal is to find a single direction $\mathbf{w}$ that would best describe the data.

- The approximation of $\mathbf{x}$ in this direction is

$$\hat{\mathbf{x}} = \mathbf{w}y = (\mathbf{w}^T\mathbf{x})\mathbf{w} \tag{3}$$

- If the weight vector $\mathbf{w}$ is normalized to unity: $\mathbf{w}^T\mathbf{w} = 1$, $\hat{\mathbf{x}}$ in (3) is the projection of $\mathbf{x}$ in the direction of $\mathbf{w}$.
- Anyway, the approximation error is

$$\mathbf{e} = \mathbf{x} - \hat{\mathbf{x}} = \mathbf{x} - \mathbf{w}y \tag{4}$$

- The goal in Oja's algorithm is to minimize the mean-square approximation error

$$J(\mathbf{w}) = \frac{1}{2}\mathrm{E}\{\| \mathbf{e} \|^2\} \tag{5}$$

- Quite similarly to the LMS algorithm, the mean-square error $J(\mathbf{w})$ is

in Oja's rule replaced by the instantaneous squared error

$$\mathcal{E}(\mathbf{w}) = \frac{1}{2} \| \mathbf{e} \|^2 \tag{6}$$

- This is because instantaneous stochastic gradients based on the current data vector $\mathbf{x}(\mathbf{k})$ only are used in Oja's rule.
- However, when the number k of iterations increases, Oja's rule has used once all the k input (data) vectors $\mathbf{x}(1), \mathbf{x}(2), \ldots, \mathbf{x}(k)$ presented thus far.
- After initial convergence has taken place, it then approximately minimizes the MSE error minimizes the mean-square error (5).
- Inserting $\mathbf{e}$ from (4) into the criterion (6) yields

$$\mathcal{E}(\mathbf{w}) = \frac{1}{2}(\mathbf{x}^T\mathbf{x} - 2\mathbf{w}^T\mathbf{x}y + \mathbf{w}^T\mathbf{w}y^2) \tag{7}$$

- The gradient of $\mathcal{E}(\mathbf{w})$ with respect to the weight vector $\mathbf{w}$ is

$$\frac{\partial \mathcal{E}(\mathbf{w})}{\partial \mathbf{w}} = -\mathbf{x}y + \mathbf{w}y^2 \tag{8}$$

- Just like the LMS rule, Oja's rule tries to minimize the error criterion (5) by proceeding into the direction of the negative instantaneous gradient:

$$\mathbf{w}(k+1) = \mathbf{w}(k) - \frac{\partial \mathcal{E}(\mathbf{w}(k))}{\partial \mathbf{w}(k)} \tag{9}$$

- Inserting (8) into (9) yields the **Oja's rule**

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \mu_k y(k)[\mathbf{x}(k) - \mathbf{w}(k)y(k)] \tag{10}$$

for updating the weight vector of the linear neuron.

- The first updating term $\mu_k y(k)\mathbf{x}(k)$ in Oja's rule (10), where $y(k) = \mathbf{w}^T(k)\mathbf{x}(k)$, realizes Hebbian learning.

- While the second updating term $-\mu_k \mathbf{w}(k)[y(k)]^2$ keeps the norm of the weight vector $\mathbf{w}(k)$ close to unity.
- Note that no explicit normalization is needed here.
- This is because it can be shown that the criteria (5) or (6) force the normalization of $\mathbf{w}$.
- The output $y(k) = \mathbf{w}^T(k)\mathbf{x}(k)$ of Oja's linear neuron can be shown to converge to the first **principal component** of the input data $\mathbf{x}$ using suitable assumptions.
- The weight vector $\mathbf{w}(k)$ converges to the principal eigenvector of the data covariance matrix $\mathbf{C}_x = \mathrm{E}\{\mathbf{x}\mathbf{x}^T\}$.
- This is the single direction describing best the data.
- We discuss principal component analysis (PCA) soon in more detail.
- See also our course T-61.3050 Machine Learning: Basic Principles.

- Oja's neuron can be extended to a network of several linear neurons.
- And Oja's learning rule can be extended for estimating several principal components (e.g. Du and Swamy, Section 12.3).
- Theoretically, it is highly interesting that Hebbian learning used in simple linear single-layer networks can provide such results.
- PCA is widely used standard preprocessing technique.
- In practice, it is usually advisable to use standard numerical algorithms for estimating the principal components.
- They are much more accurate and efficient than the relatively slowly converging stochastic gradient algorithms.
- However, the PASTd algorithm (Section 12.4) can useful in updating the principal components with new incoming samples.
- But we skip adaptive PCA algorithms in Sections 12.3-12.5.

Data preprocessing

- Preprocessing of the training data appropriately is important.
- It can affect strongly the quality of learning and final performance of a neural network.
- Consider for example the symmetric sigmoid (tanh) activation function used by the multilayer perceptron (MLP) networks.
- This activation function is defined by

$$y = \tanh(\alpha v) = \frac{e^{\alpha v} - e^{-\alpha v}}{e^{\alpha v} + e^{-\alpha v}} = \frac{1 - e^{-2\alpha v}}{1 + e^{-2\alpha v}} \qquad (11)$$

- There y is the output of a neuron, $v = \mathbf{w}^T\mathbf{x}$ its linear response, $\mathbf{w}$ the weight vector of the neuron, and $\mathbf{x}$ its input vector.
- The function is depicted in Figure 1 for three values of the slope parameter α.

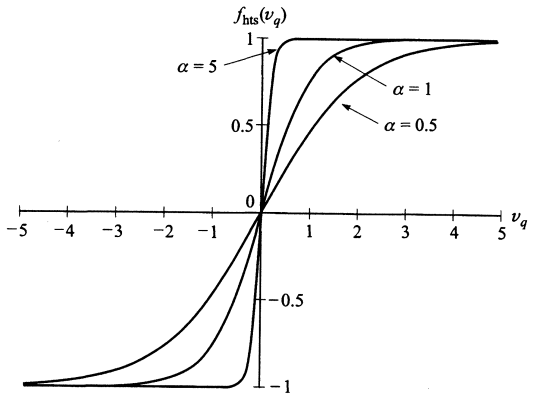

Figure 1: The $\tanh(\alpha v)$ activation function for 3 values of the slope parameter α.

- From the figure we can see that if the linear response $v \gg 0$, the output y is very close to the saturated value $+1$ of the function (11).
- Similarly if $v \ll 0$, $y \approx -1$.
- When the activation function operates on its saturated area, the neuron does not learn.
- On the other hand, it is not good if most of the values of v are too small positive or negative numbers.
- This is because the activation function (11) then operates on its linear part in the vicinity of origin.
- The network learns best when the values of the linear response v hit often to the nonlinear "shoulders" of the sigmoidal nonlinearities.

Scaling

- The training data is often **amplitude scaled** so that the values of the components of the data vectors lie between -1 and $+1$.
- This is achieved easily by dividing the components of the data vectors by the maximum absolute value of all the components.
- Another form of scaling is **mean centering.**
- There the training data $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_K$ is made zero mean.
- This takes place by subtracting from the data vectors $\mathbf{x}(k)$ their estimated mean vector

$$\hat{\mathbf{m}} = \frac{1}{K} \sum_{k=1}^{K} \mathbf{x}(k) \tag{12}$$

- Subtraction of $\hat{\mathbf{m}}$ normalizes the data $\mathbf{x}$ with respect to their first order statistics, that is mean.

- Mean centering is often accompanied with **variance scaling.**
- There the variances σ_i^2 of each component x_i of the data vectors $\mathbf{x}$ are first estimated from the formula

$$\hat{\sigma}_i^2 = \frac{1}{K-1} \sum_{k=1}^{K} [x_i(k) - \hat{m}_i]^2 \tag{13}$$

  where $\hat{m}_i$ is the i:th component of the estimated mean vector $\hat{\mathbf{m}}$.
- The components $x_i(k)$ of the data vectors $\mathbf{x}(k)$ are then divided by their estimated standard deviations $\hat{\sigma}_i$.
- This normalizes their variances to unity.
- Variance scaling is advisable if the components of the training data are measured with different units.
- If the training data $\mathbf{x}$ is mean centered or variance scaled, these operations should be done for the respective desired values $\mathbf{d}$, too.

Whitening

- Instead of mere variance scaling, the data is often whitened.
- In **whitening or sphering** , a linear transformation $\mathbf{T}$ is applied to the mean centered data vectors $\mathbf{x}$:

$$\tilde{\mathbf{x}} = \mathbf{T}\mathbf{x} \tag{14}$$

- After whitening, the transformed data vectors $\tilde{\mathbf{x}}$ have unit covariance matrix:

$$\mathbf{C} = \mathrm{E}\{\tilde{\mathbf{x}}\tilde{\mathbf{x}}^T\} = \mathbf{I} \tag{15}$$

- This means that different components of $\tilde{\mathbf{x}}$ are uncorrelated: $\mathrm{E}\{\tilde{x}_i \tilde{x}_j\} = 0$, $i \neq j$.
- And each component has unit variance: $\mathrm{E}\{\tilde{x}_i^2\} = 1$ for all i.
- Whitening can be achieved in several ways.

- For example using principal component analysis (PCA) to be discussed soon.
- Whitening is needed as preprocessing in several independent component analysis methods.
- It is useful also in learning of multilayer perceptron networks.

Preprocessing example: Old Faithful data

- Consider a data set gathered from the geyser Old Faithful in Wyoming, USA.
- The data set comprises 272 observations, each representing one eruption of the geyser.
- Each of the observations has two components:
  1. The first component is the duration of the eruption in minutes;
  2. The second one is the time until the next eruption, also in

minutes.

- The data is plotted in Figure 2; the horizontal axis shows the first component and the vertical one the second component.
- Figure 3 shows the results of standardizing the individual variables to zero mean and unit variance.
- Also shown in red are the principal axes of this normalized data set.
- The principal axes correspond to the PCA eigenvectors; they are discussed later on during this lecture and in the exercises.
- The principal axes are plotted over the range $\pm\sqrt{\lambda_i}$ where λ_i is the corresponding PCA eigenvalue.
- Finally, Figure 4 shows the result of whitening the Old Faithful data.
- After whitening, the data has zero mean and unit covariance matrix.

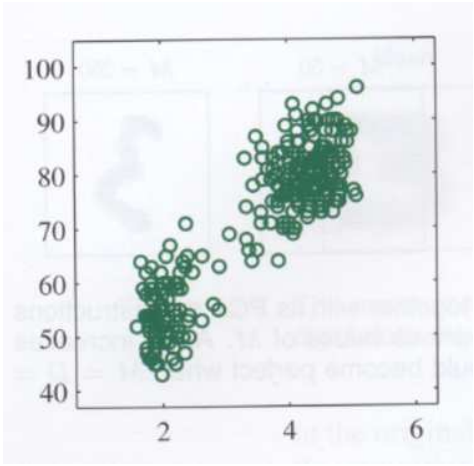

Figure 2: The original old Faithful data.

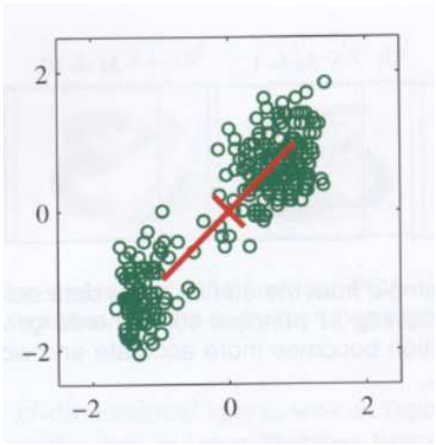

Figure 3: The Old Faithful data after mean and variance scaling and the principal axes of this normalized data set.

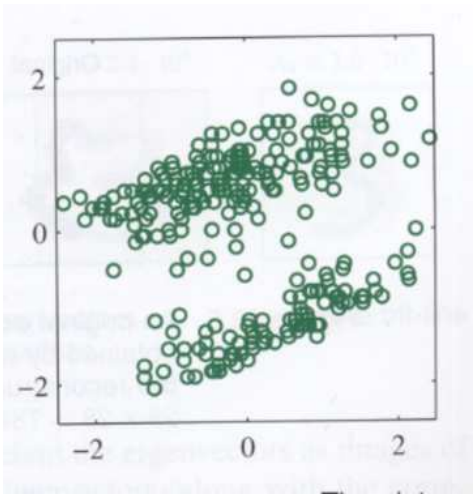

Figure 4: The Old Faithful data after whitening

Principal component analysis (PCA)

- Principal component analysis (PCA) is discussed briefly in Section 12.2 in Du's and Swamy's book.
- We consider it somewhat more in detail in the following.
- But we skip the remainder of Chapter 12 in Du's and Swamy's book.
- There sections 12.6 - 12.14 describe various extensions and modifications of PCA often somewhat superficially.
- And Sections 12.3 -12.5 adaptive (iterative) PCA algorithms.

Minimization of mean-square approximation error

- Recall that in Oja's rule the data vectors $\mathbf{x}$ were approximated by a single weight vector $\mathbf{w}$ in (3).
- Such an approximation is naturally too crude in most instances.

- But we can generalize (3) so that the approximation $\hat{\mathbf{x}}$ uses m weight vectors $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_m$:

$$\hat{\mathbf{x}} = \mathbf{W}\mathbf{y} = \mathbf{W}\mathbf{W}^T\mathbf{x} = \sum_{j=1}^{m} (\mathbf{w}_j^T\mathbf{x})\mathbf{w}_j \tag{16}$$

where the transformed vector

$$\mathbf{y} = \mathbf{W}^T\mathbf{x} = [\mathbf{w}_1^T\mathbf{x}, \mathbf{w}_2^T\mathbf{x}, \ldots, \mathbf{w}_m^T\mathbf{x}] \tag{17}$$

and the transformation matrix

$$\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_m] \tag{18}$$

- Assume that the number m of the weight vectors $\mathbf{w}_j$ is smaller than the dimensionality n of the data vectors $\mathbf{x}$: $m < n$.
- Assume further for simplicity that the n-dimensional weight vectors

$\mathbf{w}_j$ constitute an orthonormal basis:

$$\mathbf{W}^T\mathbf{W} = \mathbf{I} \tag{19}$$

- $\hat{\mathbf{x}}$ is the projection of $\mathbf{x}$ onto the subspace spanned by this basis.
- Assuming that $\mathbf{W}$ is known, we need to know for each data vector $\mathbf{x}$ only the corresponding m-dimensional feature vector $\mathbf{y}$.
- Essentially, this means that the data is compressed from original n-dimensional space to a smaller dimensional space m.
- Again, the basis vectors $\mathbf{w}_j$ are determined so that they minimize the mean-square approximation error

$$J(\mathbf{w}) = \frac{1}{2}\mathrm{E}\{\| \mathbf{e} \|^2\} = \frac{1}{2}\mathrm{E}\{\| \mathbf{x} - \hat{\mathbf{x}} \|^2\} = \frac{1}{2}\mathrm{E}\{\| \mathbf{x} - \mathbf{W}\mathbf{W}^T\mathbf{x} \|^2\} \tag{20}$$

- One can show that the optimal basis vectors $\mathbf{w}_j$ are the principal

eigenvectors $\mathbf{e}_j$ of the data covariance matrix $\mathbf{C} = \mathrm{E}\{\mathbf{x}\mathbf{x}^T\}$.

- The data $\mathbf{x}$ is assumed to have zero mean.
- If not, mean centering is used, and the mean can be added back to the results.
- The j:th principal eigenvectors $\mathbf{e}_j$ of $\mathbf{C}$ is the normalized eigenvector corresponding to the j:th largest eigenvalue λ_j of $\mathbf{C}$:

$$\mathbf{C}\mathbf{e}_j = \lambda_j\mathbf{e}_j, \qquad \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_m \tag{21}$$

- Note that all the eigenvalues of the data covariance matrix $\mathbf{C}$ are real and positive because $\mathbf{C}$ is symmetric and positive definite.
- In fact, the orthonormality assumption (19) is not needed, but it emerges from the minimization of the MSE error (20).
- The principal eigenvectors $\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_j$ minimize the approximation error (20) for every number j of principal components.

A simple illustration

- Figure 5 illustrates principal component analysis in a simple two-dimensional example.
- The are four data points, shown as red dots.
- The vector $\mathbf{u}_1$, shown by magenta line, points to the direction of the first principal eigenvector.
- The direction of this vector is the one in which the data has maximal variance.
- It is also the direction that minimizes squared projection errors, indicated by blue lines in Figure 5.
- The green dots show the projections of the data points onto the direction of the first principal eigenvector.

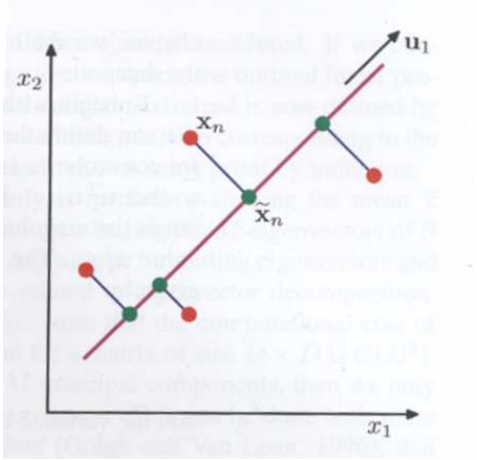

Figure 5: An illustration of PCA in two dimensions

Maximization of variance

- PCA can be derived from variance maximation criterion as well.
- The criterion for a single weight vector $\mathbf{w}$ is to maximize

$$J(\mathbf{w}) = \frac{1}{2}\mathsf{E}\{(\mathbf{w}^T\mathbf{x})^2\} \tag{22}$$

  under the normalization constraint $\| \mathbf{w} \|^2 = 1$.
- This is needed, because otherwise $J(\mathbf{w}) \to \infty$ with $\| \mathbf{w} \| \to \infty$.
- The optimal weight vector $\mathbf{w}$ is again the first principal eigenvector $\mathbf{e}_1$ in (21).
- The variance maximization problem can be generalized by looking for the m:th weight vector $\mathbf{w}_m$ which:
  - Is orthogonal to the previously found weight vectors $\mathbf{w}_j$, $j = 1, 2, \ldots, m-1$;

  - Maximizes the variance of $\mathbf{x}$ in such directions;
  - And has unit norm: $\| \mathbf{w}_m \|^2 = 1$.
- Mathematically, one should maximize the criterion

$$J(\mathbf{W}) = \frac{1}{2}\mathsf{E}\{\| \mathbf{W}^T\mathbf{x} \|^2\} \tag{23}$$

  under the orthonormality contraints $\mathbf{W}^T\mathbf{W} = \mathbf{I}$.
- Similarly as before, the optimal weight matrix (18) consists of the principal eigenvectors defined in (21): $\mathbf{w}_j = \mathbf{e}_j$.
- The criteria (20) and (23) can be optimized directly for the $n \times m$ weight matrix (18) without requiring their optimality for smaller numbers $j = 1, 2, \ldots, m-1$ of weight vectors $\mathbf{w}_j$.
- Then any orthonormal basis of the m-dimensional **PCA subspace** provides also an optimal solution.

- This PCA subspace is spanned by the m principal eigenvectors $\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_m$.

Data compression using PCA

- PCA is used widely for compressing high-dimensional data $\mathbf{x}$ to a smaller dimensional PCA subspace.
- After the mapping (17), the computational load of many methods is much smaller.
- The total variance (power) of the zero mean data $\mathbf{x}$ is $\mathrm{E}\{\| \mathbf{x} \|^2\} = \sum_{j=1}^{n} \lambda_j$.
- The MSE approximation error (20) (omitting the factor $1/2$) equals to the sum $\sum_{j=m+1}^{n} \lambda_j$ of discarded eigenvalues.
- The variance (23) (omitting the factor $1/2$) retained by PCA mapping is $\sum_{j=1}^{m} \lambda_j$.

- Often one can compress the data using PCA quite significantly while keeping the MSE error (20) relatively small.
- This is because the largest eigenvalues are often orders of magnitude larger than the smallest ones.
- The components discarded correspond often mainly to noise, carrying little relevant information on the data.
- Assume that we know K data vectors $\mathbf{x}(1), \mathbf{x}(2), \ldots, \mathbf{x}(K)$.
- In practice, one usually first estimates (assuming mean centering) the covariance matrix $\mathrm{E}\{\mathbf{x}\mathbf{x}^T\}$ from this data:

$$\hat{\mathbf{C}} = \frac{1}{K} \sum_{i=1}^{K} \mathbf{x}(i)\mathbf{x}(i)^T \tag{24}$$

- The eigenvector of $\hat{\mathbf{C}}$ are then computed using standard numerical methods and used in PCA.

Example: Compression of handwritten digits

- Figure 6 shows an example of using PCA in compression of digital images.
- The images of handwritten digits in the leftmost column of Figure 6 are represented rather coarsely using a 32×32 grid only.
- When scanned rowwise, they form 1024-dimensional data vectors.
- For each digit class, about 1700 handwritten samples were collected.
- The second column in Figure 6 shows their sample means.
- For each digit class, 64 first PCA eigenvectors were then estimated.
- The remaining columns (from 3rd to 8th) show the reconstructions of each digit using 1, 2, 5, 16, 32, and 64 principal components.
- Note how a relatively small percentage of the total 1024 principal components produces reasonably good reconstructions.

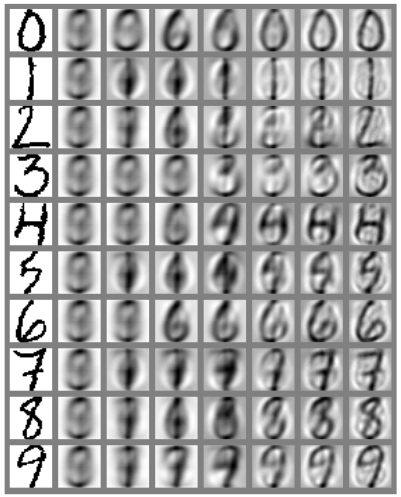

Figure 6: Compression of handwritten digit images using PCA.

Limitations of PCA

- PCA is an optimal linear technique for Gaussian distributed data.
- In fact, PCA can be derived as an optimal maximum likelihood solution from a probabilistic data model where all the random variables are Gaussian.
- See for example Section 12.2 in the book C. Bishop, "Pattern Recognition and Machine Learning", Springer 2006.
- But if the data is nonlinearly distributed or non-Gaussian, PCA is no longer optimal.
- Neural networks techniques provide then often better results.
- PCA uses second-order statistics (covariances) only.
- This suffices for Gaussian data $\mathbf{x}$, because it is completely defined by:

- Its first-order statistics, the mean vector $\mathbf{m}_x = \mathrm{E}(\mathbf{x})$;
- And second-order statistics, the covariance matrix $\mathbf{C}_x = \mathrm{E}[(\mathbf{x} - \mathbf{m}_x)(\mathbf{x} - \mathbf{m}_x)^T]$.

- Recall multivariate Gaussian distribution of vector $\mathbf{x}$:

$$p_x(\mathbf{x}) = \frac{1}{\kappa}\exp\{-\frac{1}{2}(\mathbf{x} - \mathbf{m}_x)^T\mathbf{C}_x^{-1}(\mathbf{x} - \mathbf{m}_x)\} \qquad (25)$$

- The scaling constant $\kappa = (2\pi)^{n/2}(\det(\mathbf{C}_x))^{1/2}$, where n is the dimension of $\mathbf{x}$ and $\det(\mathbf{C}_x)$ is the determinant of $\mathbf{C}_x$.
- Non-Gaussian data carries often a lot of extra information in its higher-order statistics.
- Independent component analysis (ICA) utilizes the higher-order statistics in non-Gaussian data.
- ICA will be discussed later on in our course.