Neural networks

Doc. RNDr. Iveta Mrázová, CSc.

Department of Theoretical Computer Science and Mathematical Logic

Faculty of Mathematics and Physics

Charles University in Prague

Neural networks

– Self-organization –

Doc. RNDr. Iveta Mrázová, CSc.

Department of Theoretical Computer Science and Mathematical Logic

Faculty of Mathematics and Physics

Charles University in Prague

Self-organization

- **Unsupervised training:**
  - Self-organization and clustering
- **Motivation:**
  - The network decides by itself what response fits best for the presented input pattern and adjusts its weights accordingly
- **Problem:**
  - Determine the number and location of clusters present in the feature space

Self-organization (2)

It is good to know that the truth is with us, otherwise I would feel quite afraid.

Self-organization (3)

There was in fact no girl´s war, just a few minor arguments.

Self-organization (4)

Competitive learning:

- Compete for the „right to represent the patterns“
- **Winner - takes - all** rule (WTA)
- **Inhibition** of opponents
- Network **plasticity**
- Learning with conscience

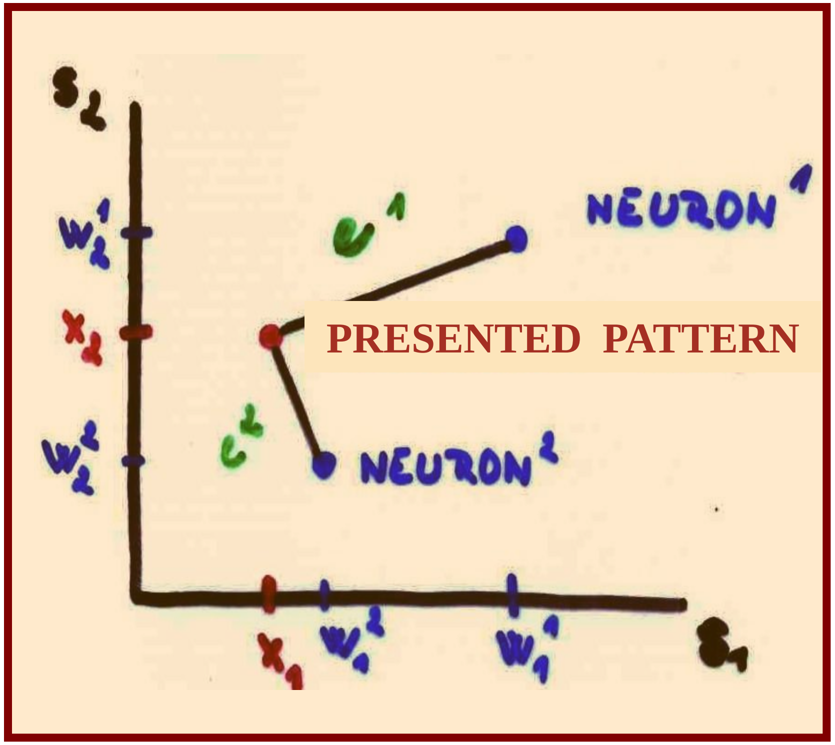
Reinforcement learning:

- Emphasizes the best reproduction of inputs

Unsupervised competitive learning

- ***n*** – dimensional input patterns are processed by such a number of neurons, that corresponds to the (assumed) number of clusters
- In such a case, the neurons compute the (Euclidean) **distance** between the presented pattern and their weight vector

Unsupervised competitive learning (2)



- Competition „is won" by the neuron, situated the closest to the presented pattern
- The winning neuron becomes the most active one and will **inhibit** the activity of other neurons

Unsupervised competitive learning (2a)

- Inhibition by means of „lateral connections“

  ==> **lateral inhibition**

- Global information about the state of all the neurons in the network is necessary to decide whether a neuron will be active or not

- The activity of a neuron signalls the membership of the presented input to the cluster of vectors represented by this neuron

Unsupervised competitive learning (3)

- The winning neuron adjusts its weights towards the presented pattern:

$$\Delta \vec{w} = \alpha \cdot (\vec{x} - \vec{w})$$

network plasticity (decays slowly during training)

Our objective:

- Position the neurons into the cluster centers
- Keep the already formed network structure

Unsupervised competitive learning (4)

- **Strategies speeding-up the training:**
  - An appropriate weight initialization, e.g., according to randomly selected patterns
- **Problems:**
  - Dead (never used) neurons
    - A grid in the Kohonen layer
    - Topological neighborhood of neurons
    - Controlled competition and the mechanism of conscience

Unsupervised competitive learning (5)

- During training, the weights of the neurons should be set in such a way that correspond to the „centers of gravity of the respective clusters“
- The energy function of a set of ***n***–dimensional normalized input patterns, $X = \{\vec{x}_1, \ldots, \vec{x}_m\}; (n \geq 2)$ is given for ***1*** neuron with the weight vector $\vec{w}$ by means of:

$$E_X(\vec{w}) = \sum_{i=1}^{m} \left\| \vec{x}_i - \vec{w} \right\|^2 ; \quad \vec{w} \in R^n$$

Unsupervised competitive learning (6)

==> in the optimum case, the weight vector is located in the center of the input pattern cluster

$$E_X(\vec{w}) = \sum_{i=1}^{m} \left\| \vec{x}_i - \vec{w} \right\|^2 = \sum_{i=1}^{m} \sum_{j=1}^{n} \left(x_{ij} - w_j \right)^2 =$$

$$= \sum_{i=1}^{m} \sum_{j=1}^{n} \left(x_{ij}^2 - 2 x_{ij} w_j + w_j^2 \right) =$$

$$= m \left(\sum_{j=1}^{n} w_j^2 - \frac{2}{m} \sum_{j=1}^{n} w_j \left(\sum_{i=1}^{m} x_{ij} \right) \right) + \sum_{i=1}^{m} \sum_{j=1}^{n} x_{ij}^2 =$$

Unsupervised competitive learning (7)

$$E_X(\vec{w}) = m\sum_{j=1}^{n}\left(w_j^2 - \frac{2}{m}w_j\left(\sum_{i=1}^{m}x_{ij}\right) + \frac{1}{m^2}\left(\sum_{i=1}^{m}x_{ij}\right)\left(\sum_{i=1}^{m}x_{ij}\right)\right) -$$

$$\underbrace{-\frac{1}{m}\sum_{j=1}^{n}\left(\sum_{i=1}^{m}x_{ij}\right)\left(\sum_{i=1}^{m}x_{ij}\right) + \sum_{i=1}^{m}\sum_{j=1}^{n}x_{ij}^2}_{=\,K} =$$

$$= m\left(\sum_{j=1}^{n}\left(w_j - \frac{1}{m}\sum_{i=1}^{m}x_{ij}\right)^2\right) + K =$$

$$= m\left\|\vec{w} - \vec{x}^*\right\|^2 + K$$

Unsupervised competitive learning (8)

→ the vector $\vec{x}^*$ is the centroid of the cluster $\{\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_m\}$ and K is a constant

→ the energy function has a global minimum at $\vec{x}^*$

Unsupervised competitive learning (9)

Clustering methods for empirical multidimensional data:

- Two basic approaches:
  - ***k*** nearest neighbors
  - ***k*** – means algorithm
- ***k* nearest neighbors** (supervised training)
  - The training patterns are stored and classified into one of ***l*** different classes
  - A new input vector is classified into the class that contains the majority of its ***k*** nearest neighbors (from the stored set)

Unsupervised competitive learning (10)

- ***k*-means clustering algorithm**
  - Unsupervised learning
  - Input vectors are classified into ***k*** different clusters

    (at the beginning, each cluster contains exactly ***1*** vector)
  - A new vector $\vec{x}$ is assigned to the cluster ***k***, the centroid $\vec{c}_k$ of which lies the closest to this pattern

Unsupervised competitive learning (11)

- ***k*-means clustering algorithm** (continue)
  - The centroid $\vec{c}_k$ is then adjusted by means of:

$$\vec{c}_k(new) = \vec{c}_k(old) + \frac{1}{n_k}\left(\vec{x} - \vec{c}_k(old)\right)$$

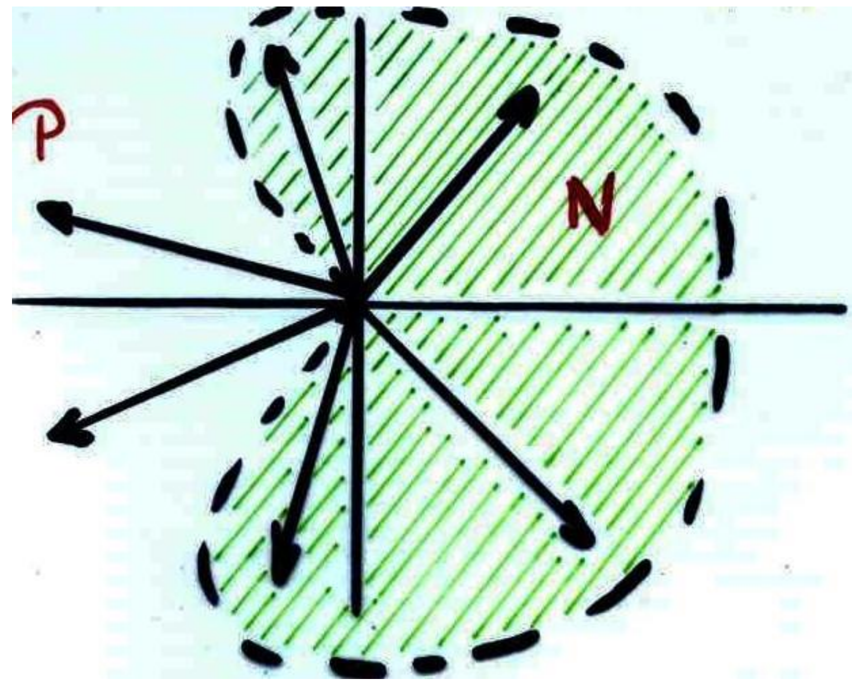
n_k …. the number of vectors already assigned to cluster k

  - This procedure is iteratively repeated for the entire data set (its structure is then captured by the „weight vectors“ $\vec{c}_i$; $i = 1, \ldots, k$)

→ **Vector quantization**

Clustering problem

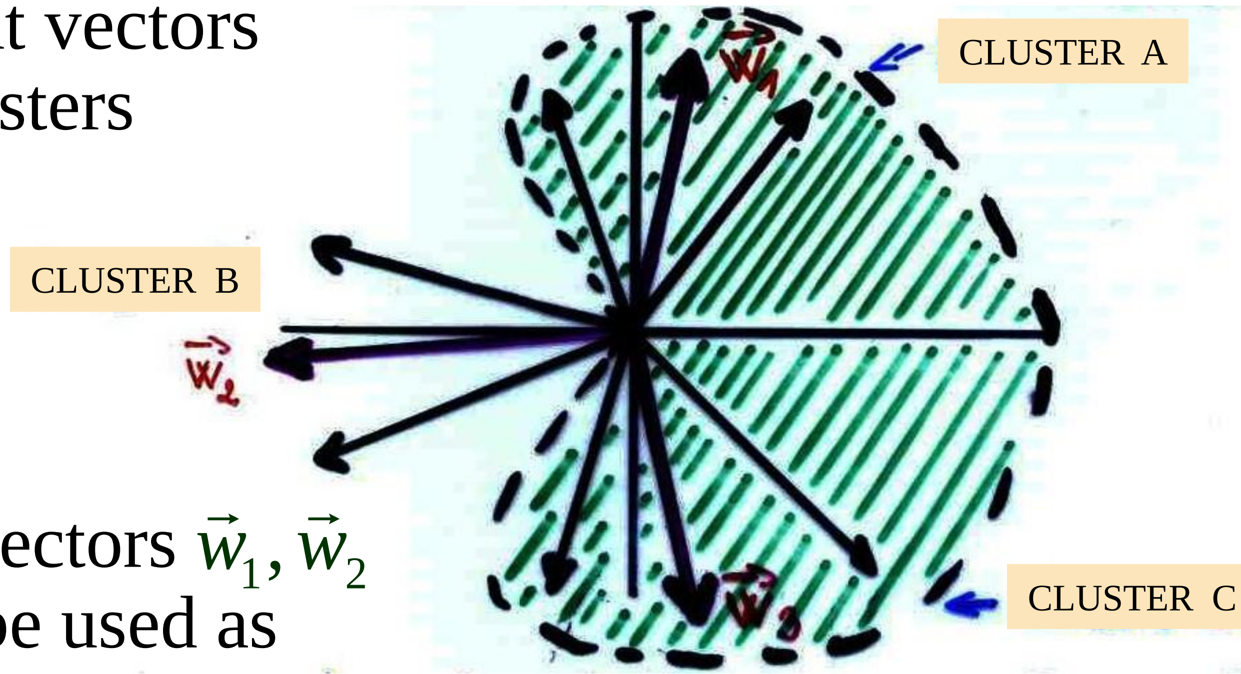
- Two sets of vectors:
  - ***P*** and ***N***
- Difficult to „separate“ the clusters by means of a simple perceptron such that:



$$\vec{w}\cdot\vec{p} \geq 0 \quad \forall \vec{p} \in P \quad \wedge \quad \vec{w}\cdot\vec{n} < 0 \quad \forall \vec{n} \in N$$

Clustering problem (2)

- Three weight vectors for three clusters



- 3 different vectors $\vec{w}_1, \vec{w}_2$ and $\vec{w}_3$ can be used as „representants“ of the respective clusters ***A*** , ***B*** a ***C***

Clustering problem (3)

- Each one of these vectors is „relatively close“ to any vector from the respective cluster
- Each weight vector corresponds to a single neuron which is active only if the input vector is close enough to its own weight vector

==> **How could we determine the number and distribution of clusters?**

Competitive learning - algorithm

- Let $X = \{\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_l\}$ be a set of normalized input vectors in $\boldsymbol{n}$ – dimensional space which we want to classify into $\boldsymbol{k}$ different clusters
- The neural network consists of $\boldsymbol{k}$ neurons, each of which has $\boldsymbol{n}$ inputs and zero threshold

Initialization:

- The normalized weight vectors $\vec{w}_1, \ldots, \vec{w}_k$ are generated randomly

Competitive learning – algorithm (2)

Test:

- Select randomly a vector $\vec{x}_j \in X$
- Compute $\vec{w}_i \cdot \vec{x}_j$ for $\boldsymbol{i = 1, \ldots, k}$
- Select $\vec{w}_m$ such that $\vec{w}_m \cdot \vec{x}_j \geq \vec{w}_i \cdot \vec{x}_j \; (\forall i = 1, \ldots, k)$
- Continue with Update

Update:

- Substitute $\vec{w}_m(new)$ with $\vec{w}_m(old) + \vec{x}_j$ and normalize
- Continue with Test

Competitive learning – algorithm (3)

- The algorithm can be stopped after a pre-determined number of steps
- The weight vectors of the k neurons are „attracted“ towards the centers of the respective clusters in the input space
- The algorithm is based on the principle known as **„winner-takes-all“**

Competitive learning – algorithm (4)

- Normalized vectors prevent the weight vectors from becoming so large that they would win the competition too often
  - Other neurons would then never be updated and would remain useless $\rightarrow$ **„dead neurons“**
- Since both the input and weight vectors are normalized, the scalar product $\vec{w}_i \cdot \vec{x}_j$ of a weight and input vector is equal to the cosine of the angle between these two vectors

Competitive learning – algorithm (5)

- The selection rule guarantees that the weight vector $\vec{w}_m$ of the cluster that is updated is the one that lies closest to the tested input vector
- The update rule rotates the weight vector $\vec{w}_m$ towards $\vec{x}_j$

Different learning rules:

- **Update with a learning constant**

$$\Delta\vec{w}_m = \eta\,\vec{x}_j \qquad ;\eta \in (0,1) \text{ decays slowly in time}$$

→ **network plasticity**

Competitive learning – algorithm (6)

Different learning rules (continue):

- **Difference update**

$$\Delta\vec{w}_m = \eta\left(\vec{x}_j - \vec{w}_m\right)$$

  - „correction“ proportional to the difference of both vectors

- **Batch update** $\rightarrow$ a more stable learning process
  - Weight „corrections“ are computed for each respective pattern and then cumulated
  - After a number of iterations the weight corrections are added to the weights at once

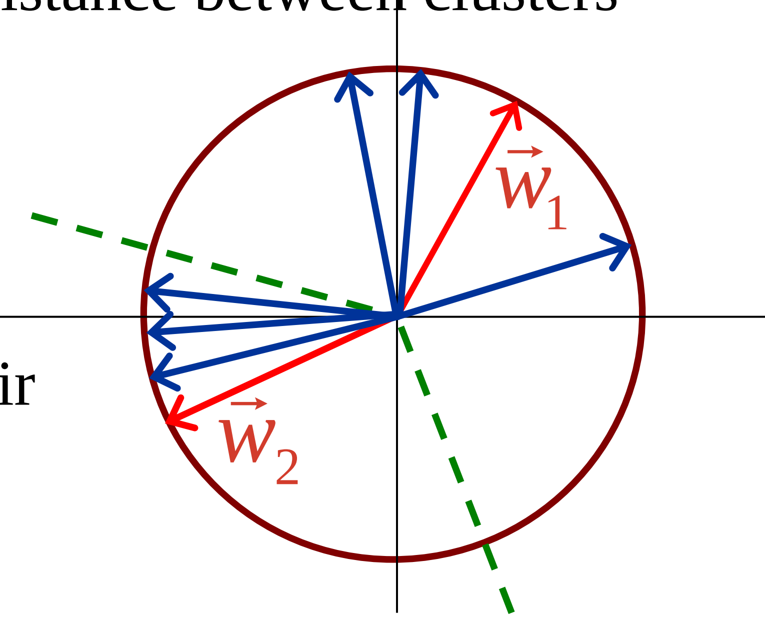
Competitive learning – algorithm (7)

Stability of the solutions

- Necessity of a suitable measure for a „good clustering“

  → **a simple approach:** find the distance between clusters

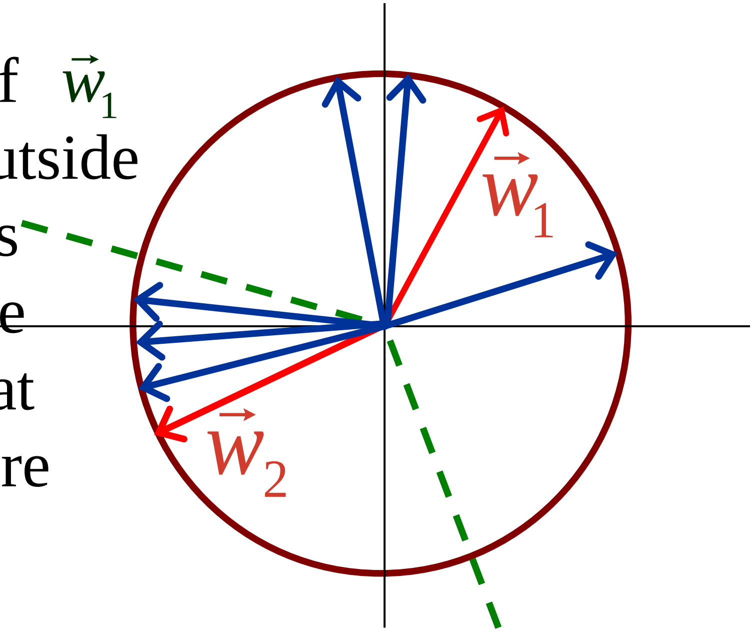
- Two clusters of vectors and two „representative weight vectors“:
  - Both „representative vectors“ lie close to the vectors from their respective cluster



Competitive learning – algorithm (8)

Stability of the solutions (continue):

- $\vec{w}_1$ lies inside a „cone“ defined by the vectors from its cluster
- $\vec{w}_2$ lies outside of the „cone“ of $\vec{w}_1$
- The vector $\vec{w}_1$ will not jump outside of its „cone“ in future iterations
- The vector $\vec{w}_2$ will jump inside the cone defined by its cluster at some point and will remain there



$\rightarrow$ such a solution will be stable

Competitive learning – algorithm (9)

Solution in stable equilibrium:

- **Intuitive idea:**
  - **stable equilibrium requires clearly delimited clusters**
- If the clusters overlap or are very extended, it can be the case that no stable solution can be found

  ==> **unstable equilibrium**

Stability of the solutions - analysis

Definition:

Let $\boldsymbol{P}$ denote the set $\{\vec{p}_1, \ldots, \vec{p}_m\}$ of $\boldsymbol{n}$ – dimensional ($\boldsymbol{n} \geq \boldsymbol{2}$) vectors located in the same half-space (~ a formal restriction of the cluster size).

The cone $\boldsymbol{K}$ defined by $\boldsymbol{P}$ is the set of all vectors $\vec{\boldsymbol{x}}$ of the form $\vec{x} = \alpha_1 \vec{p}_1 + \ldots + \alpha_m \vec{p}_m$, where $\boldsymbol{\alpha_1, \ldots, \alpha_m}$ are positive real numbers.

- The cone of a cluster contains all vectors „within“ the cluster
- The diameter of a cone defined by normalized vectors is proportional to the maximum possible angle between two vectors in the cluster

Stability of the solutions – analysis (2)

Definition:

The (angular) diameter φ of a cone K defined by normalized vectors $\{\vec{p}_1, \ldots, \vec{p}_m\}$ corresponds to:

$$\varphi = \sup \left\{ \arccos\left(\vec{a} \cdot \vec{b}\right) \middle| \forall \vec{a}, \vec{b} \in K ; \|\vec{a}\| = \|\vec{b}\| = 1 \right\}$$

where $0 \le \arccos\left(\vec{a} \cdot \vec{b}\right) \le \pi$

- A sufficient condition for stable equilibrium is that the angular diameter of the cluster´s cone must be smaller than the distance between clusters.

Stability of the solutions – analysis (3)

Definition:

Let $P = \{\vec{p}_1, \ldots, \vec{p}_m\}$ and $N = \{\vec{n}_1, \ldots, \vec{n}_k\}$ be two non-void sets of normalized vectors in an ***n***–dimensional space ($\boldsymbol{n \geq 2}$), that define the cones $\boldsymbol{K_P}$ and $\boldsymbol{K_N}$

- If the intersectionof the two cones is void, the (angular) distance between $\boldsymbol{K_N}$ and $\boldsymbol{K_P}$ is given by:

  $$\psi = \inf \left\{ \arccos(\vec{p} \cdot \vec{n}); \; \vec{p} \in K_P, \; \vec{n} \in K_N \text{ and } \|\vec{p}\| = \|\vec{n}\| = 1 \right\}$$

  where $0 \leq \arccos(\vec{p} \cdot \vec{n}) \leq \pi$

- If the two cones $\boldsymbol{K_P}$ and $\boldsymbol{K_N}$ intersect, $\boldsymbol{\psi_{P,N} = 0}$

Stability of the solutions – analysis (4)

- If angular distance between clusters is greater than angular diameter of the clusters, a stable solution exists
  - The weight vectors will lie inside their respective cluster cones
  - Once inside their respective cluster cones, ‚the weight vectors will not leave them

Clustering quality control:

- A smaller number of „more compact“ clusters is usually preferred
- A cost function penalizing a too big number of clusters

PCA – Principal Component Analysis

~ reduces the dimensionality of the input data

→ use less features without losing essential information

→ selection of the most important features

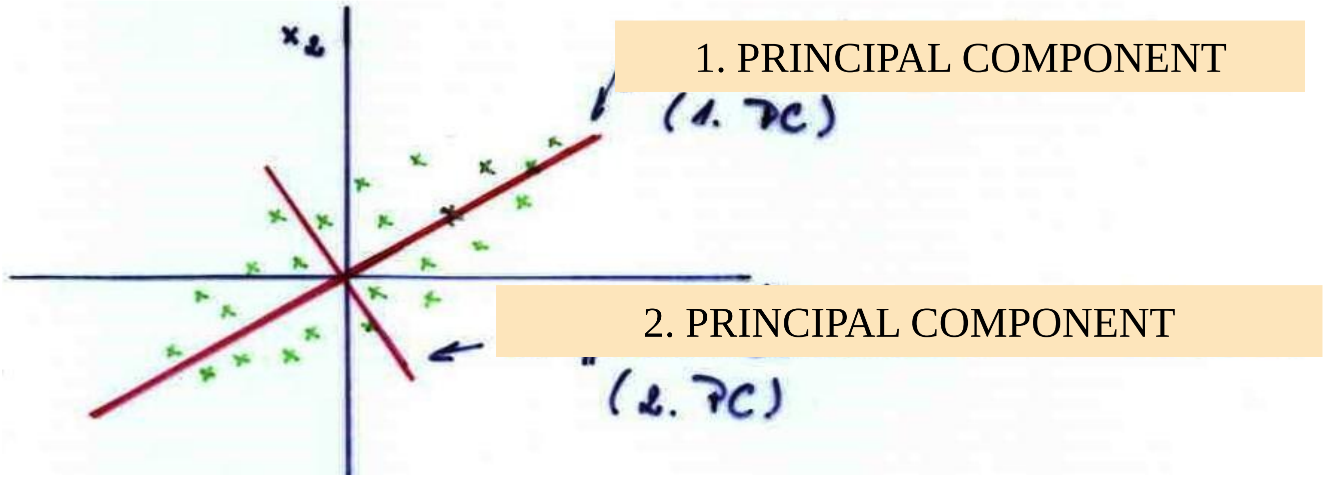
~ a set of m n – dimensional vectors is given:

$$\{\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_m\}$$

~ the first principal component of this set of vectors is a vector $\vec{w}$, which maximizes the expression

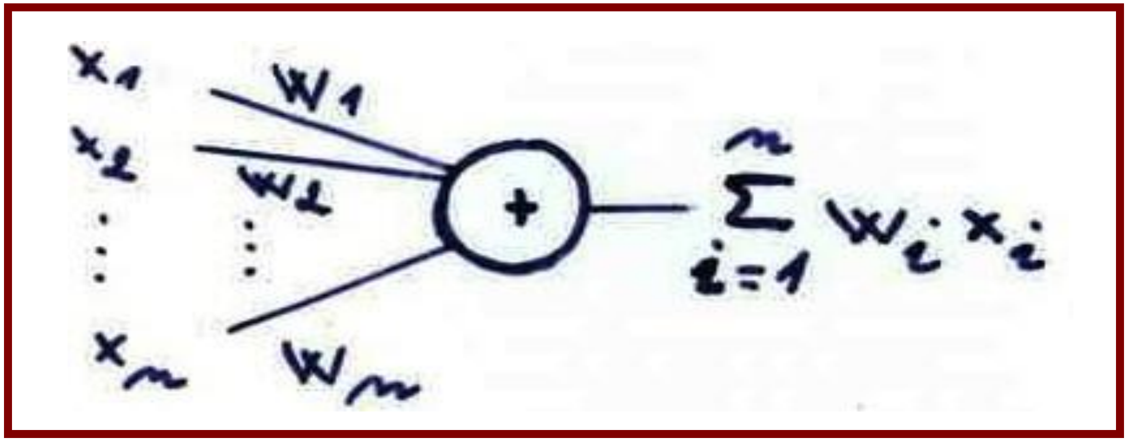
$$\frac{1}{m} \sum_{i=1}^{m} \| \vec{w} \cdot \vec{x}_i \|^2$$

PCA – Principal Component Analysis (2)



- Distribution of the input data:
  - 1. PC: direction of maximum variance in the data
  - 2. PC: orthogonal to 1. PC and maximum variance (~ subtract from $\vec{x}$ its orthogonal projection on 1. PC)

PCA – Principal Component Analysis (3)



- **The applied model:**
  - Linear associator
    - outputs the weighted input as a result
  - Unsupervised reinforcement learning – Oja´s algorithm

Computation of the principal components with artificial neural networks

Oja´s learning algorithm (E. Oja, 1982)
(for the computation of the first principal component)

Assumption:

- The centroid of the input data is located at the origin

Start:

- Let X be a set of n – dimensional vectors
- The vector $\vec{w}$ is initialized randomly ($\vec{w} \neq 0$)
- A learning constant γ with $0 < \gamma \leq 1$ is selected

Computation of the principal components with artificial neural networks (2)

Oja´s learning algorithm (continue):

Update:

- From the set $\boldsymbol{X}$ a vector $\vec{x}$ is randomly selected
- The scalar product $\Phi = \vec{x} \cdot \vec{w}$ is computed
- The new weight vector is $\vec{w} + \gamma\, \Phi \left(\vec{x} - \Phi\, \vec{w} \right)$
- Make γ smaller and go to „Update“

<u>Stopping condition for update</u>

- e.g., a predetermined number of iterations

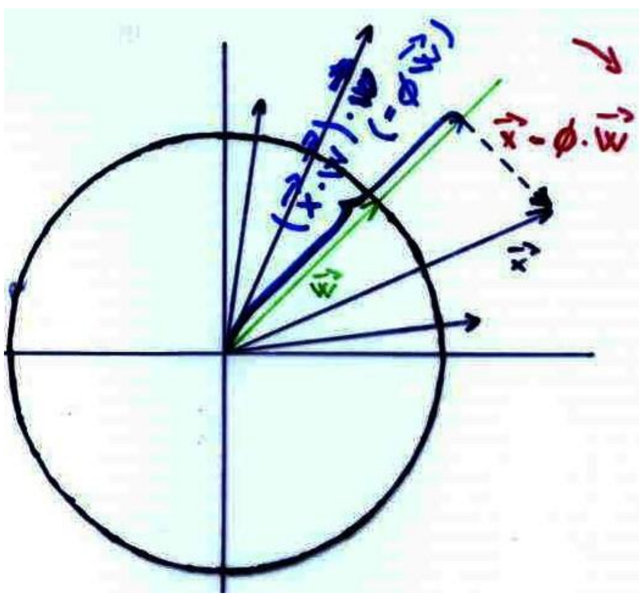
Computation of the principal compo-nents with artificial neural networks (3)

- **Learning constant γ**
  - The learning constant must be chosen small enough to guarantee adequate weight updates (limit big oscillations)
- **„Automatic normalization“ of the weight vector**
  - A global information about all patterns is not necessery
  - A local information about the updated weight, input and scalar product of the associator is sufficient
- **The first principal component is equivalent to the direction of the longest eigenvector of the correlation matrix of the considered input vectors**

Convergence of Oja´s algorithm

When a unique solution to the task exists, Oja´s algorithm will converge:

Idea of the proof:



Update of $\vec{w}$ towards $\vec{x}$

-) if Oja´s algorithm is started a weight vector inside a cone, it will oscillate in it, but will not leave it

-) for $\| \vec{w} \| = 1$ the scalar product $\Phi = \vec{x} \cdot \vec{w}$ corresponds to the length of projection of $\vec{x}$ on $\vec{w}$

Convergence of Oja´s algorithm (2)

<u>Idea of the proof</u> <u>(continue):</u>

-) the vector $\vec{x} - \Phi\,\vec{w}$ is orthogonal to $\vec{w}$
-) an iteration of Oja´s algorithm attracts $\vec{w}$ to the vectors from the cluster ***X***
-) if the length of $\vec{w}$ remains equal to ***1*** (or close to ***1***), $\vec{w}$ will be brought into the middle of the cluster
-) further, it is necessary to show that the vector $\vec{w}$ is automatically normalized by the Oja´s learning algorithm:

<u>Idea:</u> a) the length of the vector $\vec{w}$ is bigger than ***1***
b) the length of the vector $\vec{w}$ is smaller than ***1***

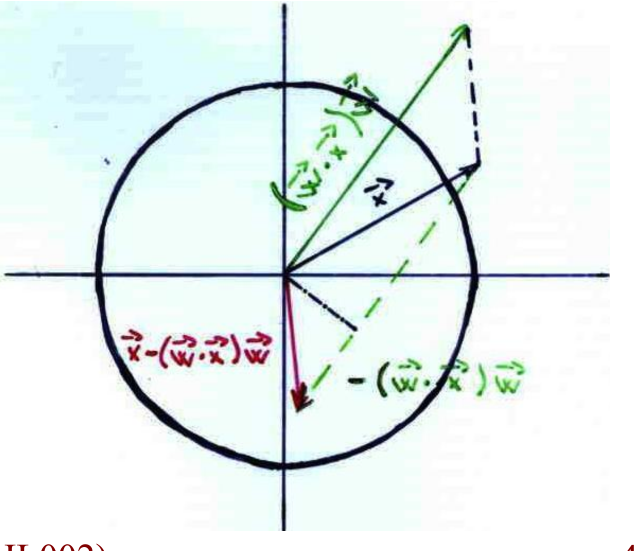
Convergence of Oja´s algorithm (3)

a) the length of the vector $\vec{w}$ is bigger than **1**

- The length of the vector $(\vec{x} \cdot \vec{w})\vec{w}$ is bigger than the length of the orthogonal projection of $\vec{x}$ on $\vec{w}$
- Further, assume that $\vec{x} \cdot \vec{w} > 0$, i.e., that the vectors $\vec{x}$ and $\vec{w}$ are not too far away one from the other
- Vector $\vec{x} - (\vec{x} \cdot \vec{w})\vec{w}$ has a negative projection on $\vec{w}$, as

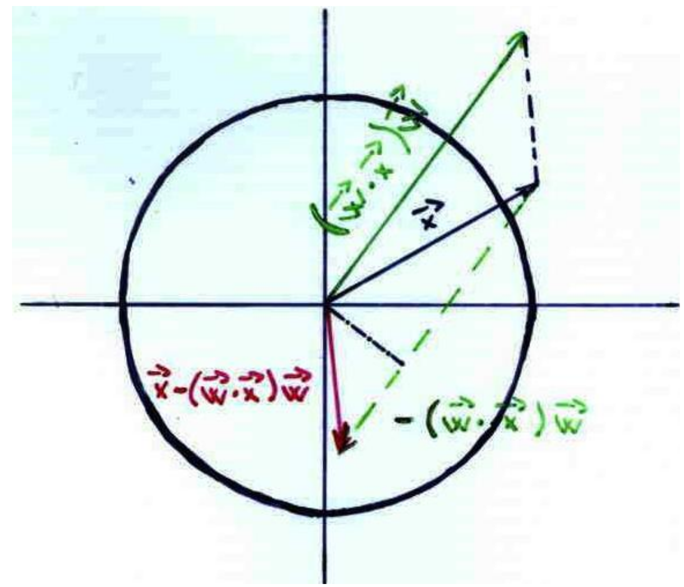
$$\left(\vec{x} - (\vec{x} \cdot \vec{w})\vec{w}\right) \cdot \vec{w} =$$

$$= \vec{x} \cdot \vec{w} - \|\vec{w}\|^2 \, \vec{x} \cdot \vec{w} < 0$$



Convergence of Oja´s algorithm (4)

- **The result of many iterations:**
  - (the vector $\vec{x} - (\vec{x} \cdot \vec{w})\vec{w}$ has one component normal to $\vec{w}$ and another one with the opposite direction of $\vec{w}$)
  - $\vec{w}$ will be brought into the middle of the cluster of vectors (and the normal component cancels in average)
  - $\vec{w}$ will become smaller with the growing number of iterations of this type (!avoid making $\vec{w}$ too small or reversing its direction in an iteration)



Convergence of Oja´s algorithm (5)

- **A suitable choice of the learning parameter γ and normalization of the training vectors:**
  - if the vector $\vec{x}$ has a positive scalar product Φ with $\vec{w}$, then this should hold also for the new weight vector:
  - It should thus hold: $\vec{x}\cdot\left(\vec{w}+\gamma\,\Phi\left(\vec{x}-\Phi\,\vec{w}\right)\right)>0$

$$\Phi+\gamma\,\Phi\left\|\vec{x}\right\|^2-\gamma\,\Phi\,\Phi^2>0$$

$$\underbrace{\Phi}_{>0}\;\underbrace{\left(1+\gamma\left(\left\|\vec{x}\right\|^2-\Phi^2\right)\right)}_{\Longrightarrow\;\gamma\left(\left\|x^2\right\|-\Phi^2\right)>-1}>0$$

  - For positive small enough γ is always satisfied

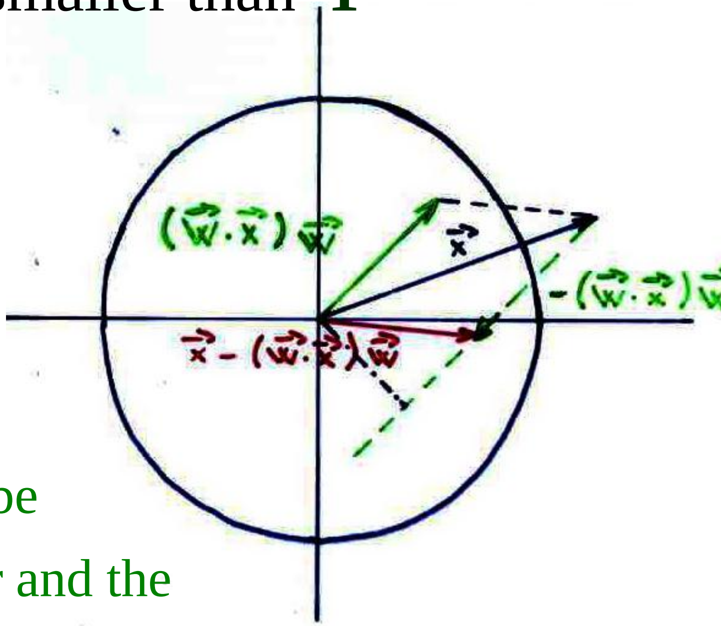
Convergence of Oja´s algorithm (6)

b) the length of the vector $\vec{w}$ is smaller than **1**

(similarly to a))

→ The vector $\vec{x} - (\vec{x} \cdot \vec{w})\vec{w}$ has a positive projection on $\vec{w}$

→ growing of $\vec{w}$



- **Combining a) and b)** $\Rightarrow$ $\vec{w}$ will be brought into the middle of the cluster and the length of $\vec{w}$ will oscillate around 1 (for a small enough γ)
- **<u>Problems:</u>** „sparse“ clusters

Oja´s learning algorithm: problems and generalization

<u>Problems:</u> -) „sparse“ clusters

-) big differences in the length of input vectors

Computation of more principal components: