Machine Learning - II

Methods of Knowledge Based Software Development

S. Nõmm

[1]Department of Software Science, Tallinn University of Technology

08.12.2017

In continuation of the previous lecture

References:

- The Elements of Statistical Learning (Second Edition) by T. Hastie, R. Tibishirani, J. Friedman. Springer 2011.
- Data Mining The TextBook by C.Aggarwal. Springer 2015.
- Machine Learning A Probabilistic Approach by K.Murphy. MIT Press 2012.

Due to the fact that Home assignment early deadline is was agreed to be December the 22 It is necessary to explain two formalism required to start preparations for the home work. This lecture contains two different parts.

- Part I Support Vector Machines
- Part II Neural Networks (Introduction)

- Part I Support Vector Machines

In continuation of the previous lecture

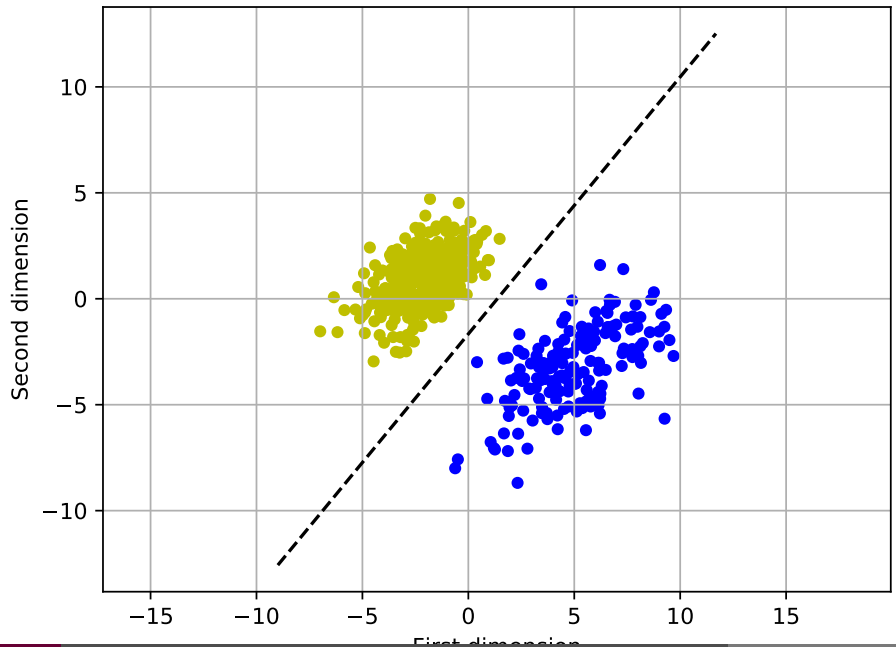
- Linear regression

$$\hat{y} = a_1 x_1 + \ldots a_n x_n + b = x^T a + b$$

- Logistic regression (case of two classes)

$$C(x) = \text{sign}[x^T a + b]$$

One may think about trend line (it is not any more a trendline) as of hypersurface (decision boundary) separating space into subspaces corresponding to each class.



Separability

Linear separability

Two classes are said to be linearly separable in $\mathbb{R}^n$ -if there exists a hyperplane dividing the space into two subspaces such that all the elements of the first class belong to one subspace and the elements of the second class belong to the other subspace.
Or
-if there exist n - dimensional vector a and scalar b such that for the elements of one class $x^T a > b$ and for the elements of the second class $x^T a < b$

Separability

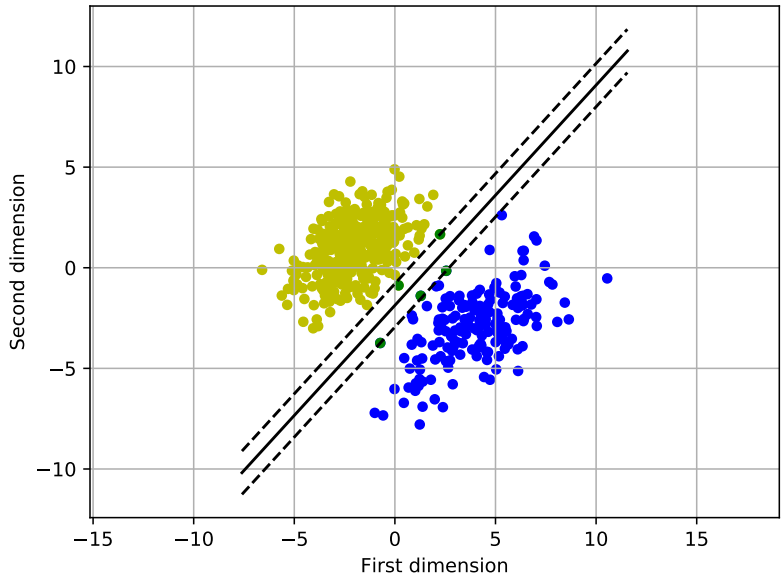
- If two classes are linearly separable it is possible to construct two hyperplanes, parallel to the "separating" hyperplane, such that first hyperplane would contain at least one point of the first class and second hyperplane will contain at least one point of the second class.
- The training data points belonging to these hyperplanes are referred as *support vectors* and the distance between the hyperplanes is referred as *margin*.
- In order to determine maximum margin hyperplane nonlinear programming optimization is required. First margin is expressed as the function of the coefficients of separating hyperplane. Second optimization problem is solved.

Maximum margin hyperplane

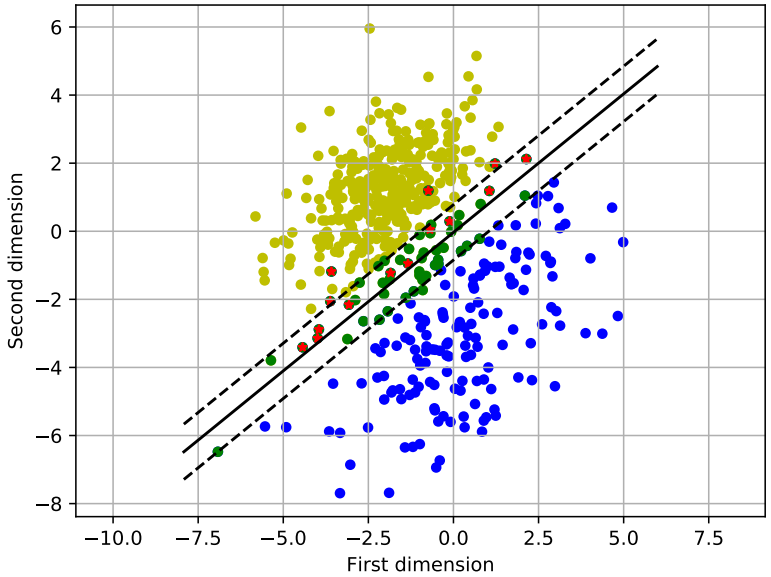
- For the hyperplane $x^T a + b$ vector $a = (a_1, \ldots, a_n)$ is n - dimensional vector representing the normal direction to the hyperplane.
- Then the distance (margin) from the separating hyperplane to the hyperplanes containing points of each class (see previous slide) would be $M = ||a||^{-1}$.
- The optimization problem may be stated in terms of finding vector a that would maximize margin

Hard and soft separation cases

The case when classes are linearly separated is rather rare.



Hard separation case (classes are linearly separable)



Soft separation case, violations are present

Support Vector Machines: linear case

- In the case of hard separability support vector machine constructs the hyperplanes maximizing the margin.
- In the case of soft separability the idea of maximizing the margin remains the same but violations are allowed. "Small" number of points located on the "wrong" side of the hyperplane. These points are referred as *violations* or violation points . Denote $\xi = (\xi_1, \dots \xi_N)$ the distances from the corresponding hyperplanes. Then Support vector classifier is required to

$$\min ||a|| \quad \text{subject} \quad \text{to} \quad \begin{cases} y_i(x_i^T a + b) \geq 1 - \xi_i, \forall i, \\ \xi_i \geq 0, \sum \xi_i \leq C \end{cases} .$$

where C is a constant - cost parameter.

Support Vector Machines: nonlinear case

- Let the decision boundary between two classes is given by

$$8(x_1 - 1)^2 + 50(x_2 - 2) = 1$$

  may be easily rewritten as

$$8x_1^2 - 16x_1 + 50x_2^2 - 200x_2 + 207 = 0$$

- It may be expressed linearly in terms of four variables $z_1 = x_1^2$, $z_2 = x_1$, $z_3 = x_2^2$ and $z_4 = x_2$ as

$$8z_1 - 16z_2 + 50z_3 - 200z_4 + 207 = 0$$

- The only drawback of this approach is that it requires to perform transformation explicitly.
- As a small exercise one may simplify initial equation in to the form where the problem will be transformed from 2D to the 3D case. In this case it is possible to visualize not only initial decision boundary but also one in a higher dimensional space. Also it would be able observe an effect of changing numeric values of the parameters.

The kernel trick

- Solution of the optimization problem 8 required to construct the hyperplane (decision boundary) is based on maximization of the Lagrangian (Wolfe) dual objective function:

$$L_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j x_i^T X_j.$$

- Observe that this problem may be solved not in terms of data points but in terms of their dot products. This leads the idea to define similarity function on the transformed representation with use of so called kernel function.

$$K(x_i, x_j) = \Phi(x_i) \cdot \Phi(x_j)$$

  In this case Lagrangian (Wolfe) dual objective function will take shape

$$L_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j K(x_i, x_j).$$

- Corresponding equation of the decision boundary $\Phi(x)^T a + b$

The kernel trick: most popular kernel functions

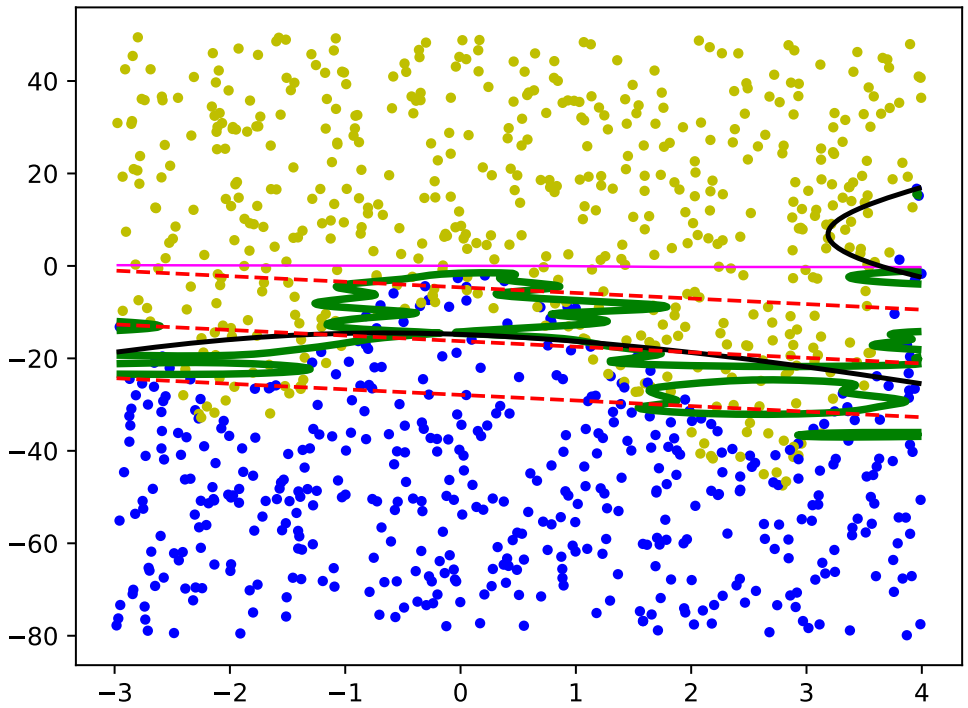
K should be a symmetric positive define function.

- n -th Degree polynomial $K(x_i, x_j) = (1 + x_i \cdot x_j)^n$.
- Radial basis $K(x_i, x_j) = \exp(-\gamma ||x_i - x_j||^2)$.
- Sigmoid kernel $K(x_i, x_j) = \tanh(k_1 x_i \cdot x_j + k_2)$.

A note on the implementation in Python

Take time to observe all the methods associated with the classifier!

```
from sklearn.svm import SVC
clf = SVC(kernel='linear')
clf = SVC(kernel='poly',degree=3)
clf = SVC(kernel='rbf')
clf_fit(D_ini, D_init_labels)
Z = clf.predict(D_valid)
```

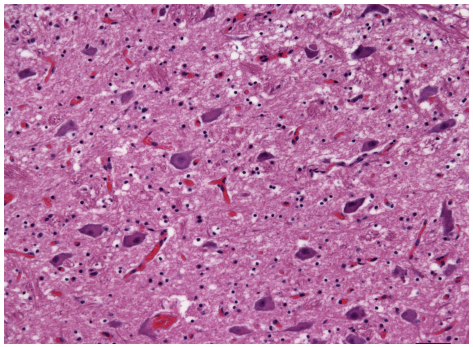


- Part II Introduction to Neural Networks

What is *Artificial Neural Network ?*

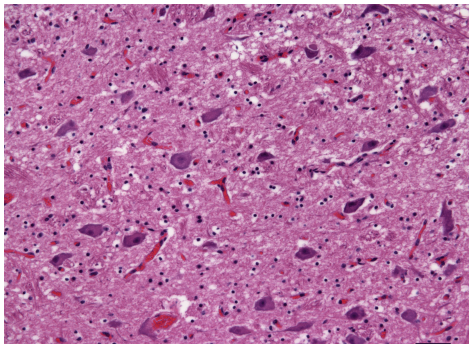
Biology inspired mathematical abstraction

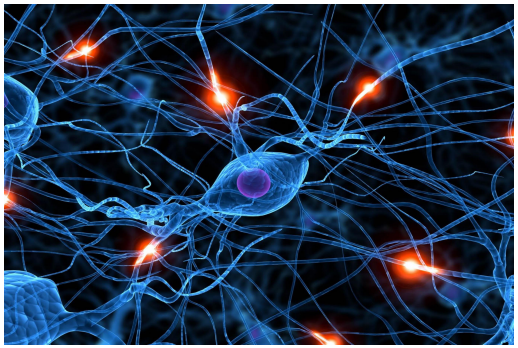
What is *Artificial Neural Network ?*



Biology inspired mathematical abstraction

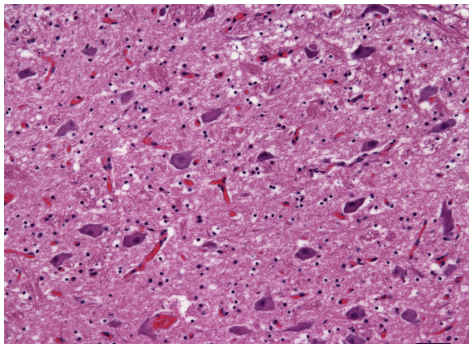
What is *Artificial Neural Network ?*

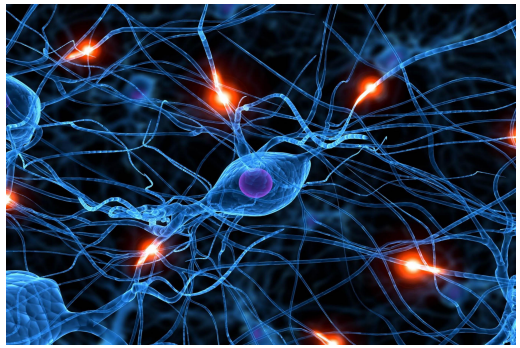




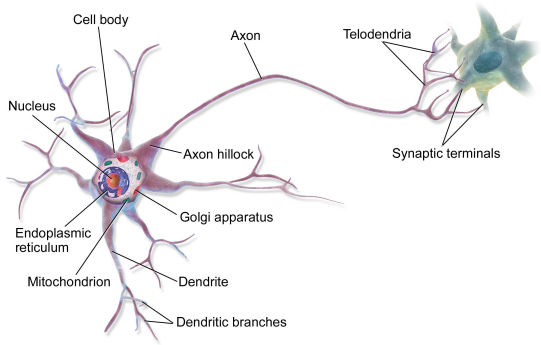
Biology inspired mathematical abstraction

What is *Artificial Neural Network ?*

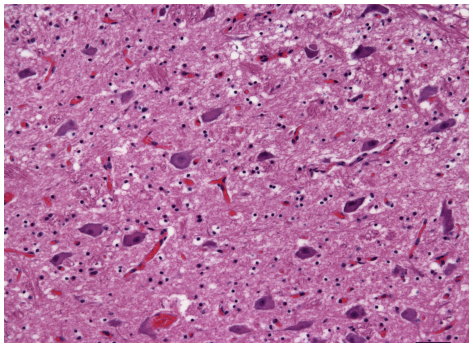


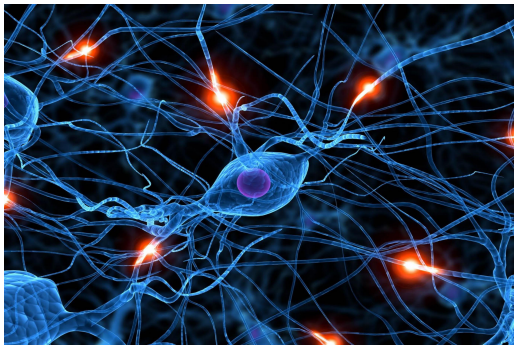


Biology inspired mathematical abstraction

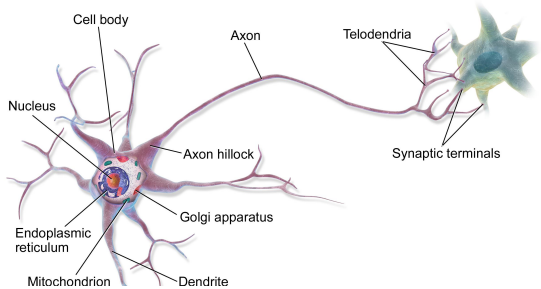


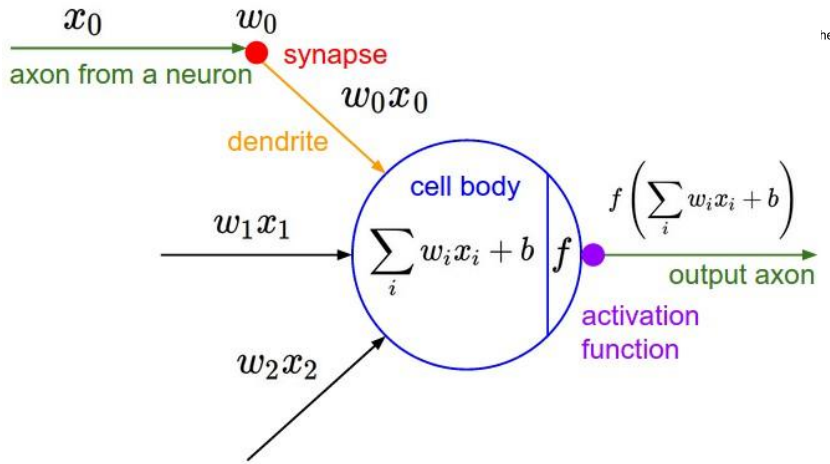
What is *Artificial Neural Network ?*



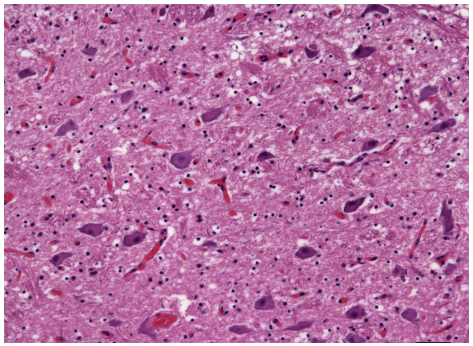


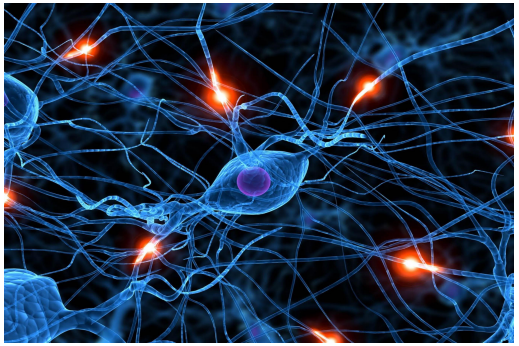
Biology inspired mathematical abstraction



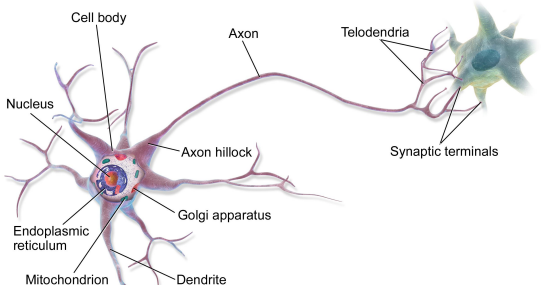


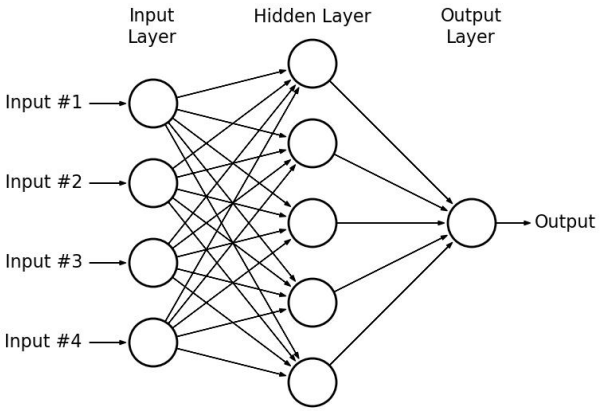
What is *Artificial Neural Network ?*

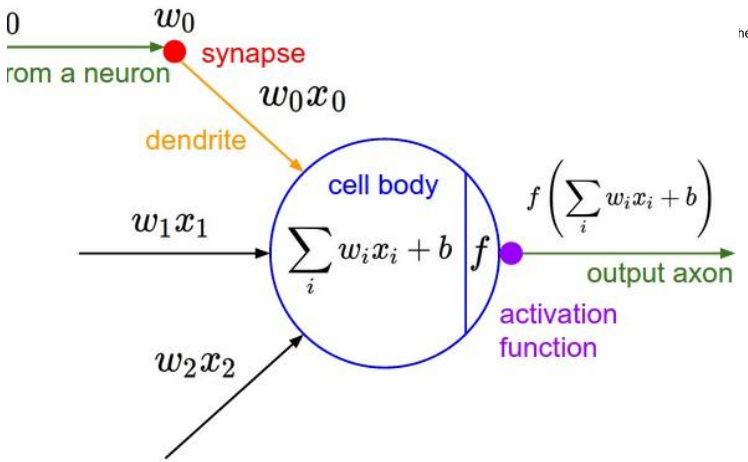




Biology inspired mathematical abstraction







Artificial Neural Network

- Brain consists of neurons.
- Basic idea of the artificial neural network is to combine models of the single neurons.
- Mathematical model of the single neuron is called *perceptron*.
- Perceptron has a number of drawbacks motivating the idea of artificial neural networks.

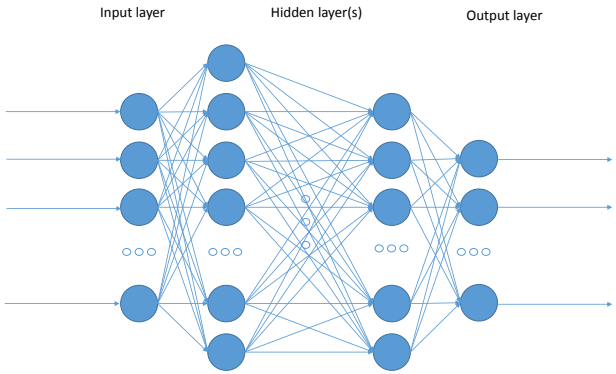
The model of a single neuron

$$y = f\left(\sum_{j=1}^{d} w_j x_j\right) = f\left(w^T x\right)$$

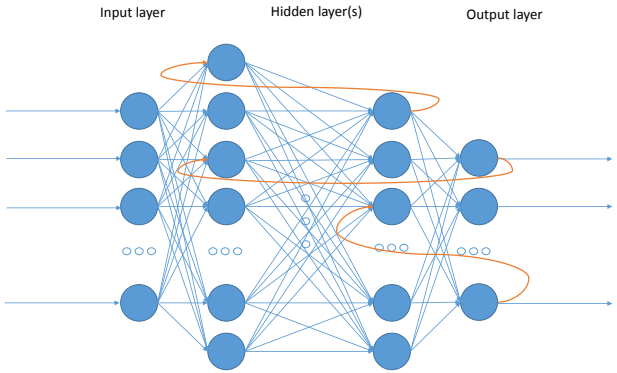
- d is the dimension of the input vector and number of *weights* w_j.
- $x \in \mathbb{R}^d$ - *input vector*.
- $f(\cdot)$ is a smooth nonlinear function referred as *activation function*. Most popular activation functions are:
  - Log-Sigmoid function $logsig(x) = 1/(1 + e^{-x})$.
  - Tan-Sigmoid function $tansig(x) = 2/(1 + e^{(-2*n)}) - 1$
  - Linear transfer function
- In case of Log -Sigmoid activation function the model of the neuron takes form:

$$y = \frac{1}{1 + e^{-\sum_{j=1}^{d} w_j x_j}}$$

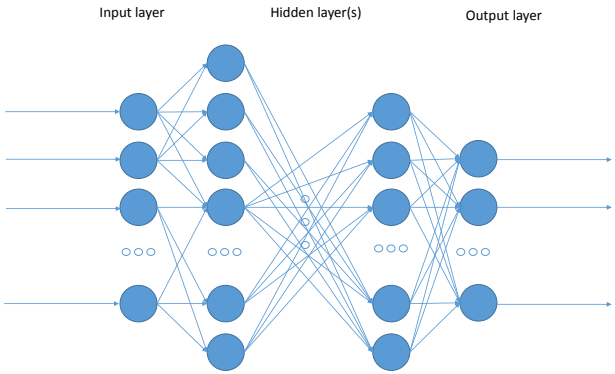
Different topologies of the ANN



Fully connected feed forward neural network



Recurrent neural network



Restricted connectivity feed forward neural network

ANN

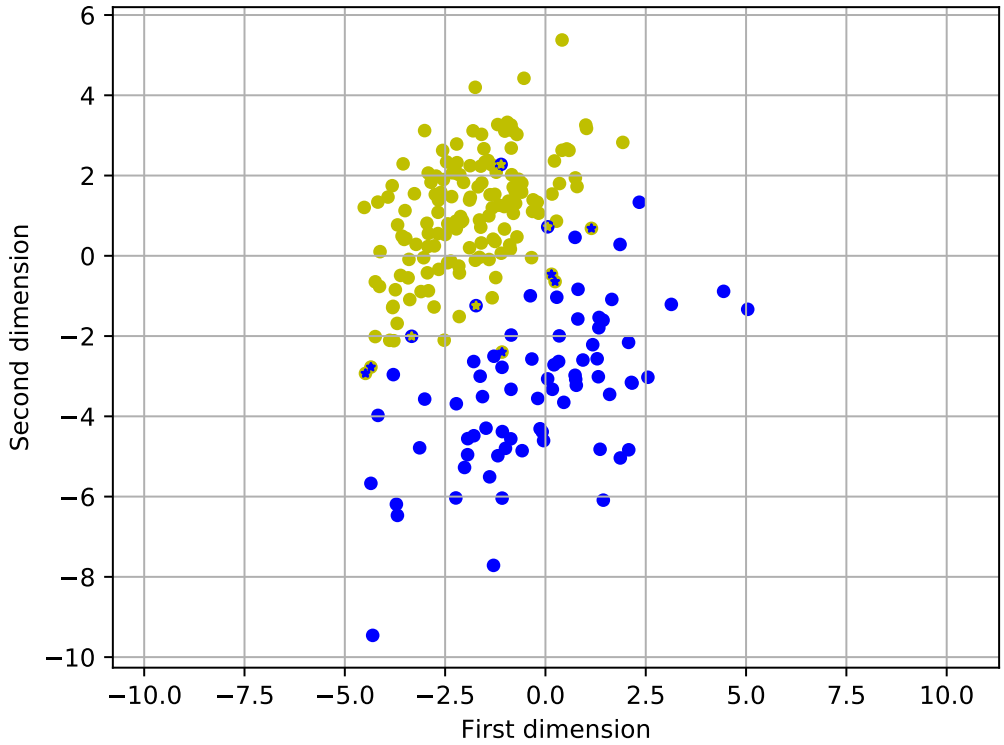
- How to choose optimal topology of the net? Nr of layers? Nr of neurons on hidden layer etc?
- Universal approximation theorem:
  - G.Cybenko (1989) Single hidden layer with finite number of neurons (multilayer perceptron) can approximate continuous functions on a compact subsets of $\mathbb{R}^n$. Some weak assumptions about activation function were made. Algorithmic aspects were not touched.
  - K.Hornik (1991) Demonstrated importance of the choice of architecture over the choice of activation function.

Training methods

- Initialization of the network.
- Training algorithms.
  - Levenberg-Marquardt backpropagation.
  - Bayesian regularization backpropagation.
  - Scaled conjugate gradient backpropagation.
  - Resilent backpropagation.
- Stopping criteria.
- Epoch - is the measure of the number of times all the training vectors are used to update the weights.

Simple example

Very simple classification example: Network with two hidden layers, with 5 and 2 neurons correspondingly. For training and verification used the same set as in previous part.



Home Assignment

Formal statement of the home Assignment will be made available before the practice 14:00.