Neural Networks: Learning Process

Prof. Sven Lončarić

sven.loncaric@fer.hr

http://www.fer.hr/ipg

Overview of topics

- Introduction
- Error-correction learning
- Hebb learning
- Competitive learning
- Credit-assignment problem
- Supervised learning
- Reinforcement learning
- Unsupervised learning

Introduction

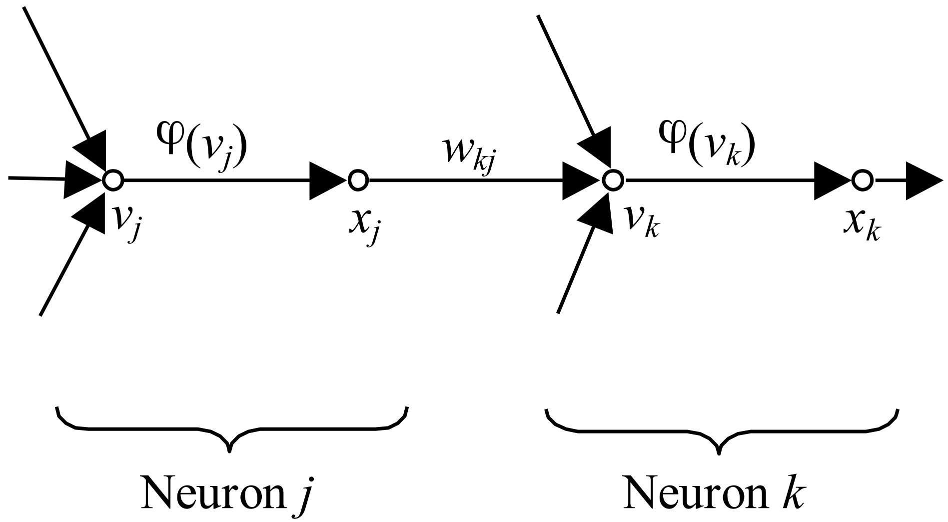
- One of the most important ANN features is ability to learn from the environment
- ANN learns through an iterative process of synaptic weights and threshold adaptation
- After each iteration ANN should have more knowledge about the environment

Definition of learning

- Definition of learning in the ANN context:
  - Learning is a process where unknown ANN parameters are adapted through continuous process of stimulation from the environment
  - Learning is determined by the way how the change of parameters takes place
- This definition implies the following events:
  - The environment stimulates the ANN
  - ANN changes due to environment
  - ANN responds differently to the environment due to the change

Notation

- v_j and v_k are activations of neurons *j* and *k*
- x_j and x_k are outputs of neurons *j* and *k*
- Let $w_{kj}(n)$ be synaptic weights at time *n*



Notation

- If in step n synaptic weight $w_{kj}(n)$ is changed by $\Delta w_{kj}(n)$ we get the new weight:

  $$w_{kj}(n+1) = w_{kj}(n) + \Delta w_{kj}(n)$$

  where $w_{kj}(n)$ and $w_{kj}(n+1)$ are old and new weights between neurons k and j
- A set of rules that are solution to the learning problem is called a learning algorithm
- There is no unique learning algorithm, but many different learning algorithms, each with its advantages and drawbacks

Algorithms and learning paradigms

- Learning algorithms determine how weight correction $\Delta w_{kj}(n)$ is computed
- Learning paradigms determine the relation of the ANN to the environment
- Three basic learning paradigms are:
  - Supervised learning
  - Reinforcement learning
  - Unsupervised learning

Basic learning approaches

- According to learning algorithm:
  - Error-correction learning
  - Hebb learning
  - Competitive learning
  - Boltzmann learning
  - Thorndike learning
- According to learning paradigm:
  - Supervised learning
  - Reinforcement learning
  - Unsupervised learning

Error-correction learning

- Belongs to the supervised learning paradigm
- Let $d_k(n)$ be desired output of neuron k at moment n
- Let $y_k(n)$ be obtained output of neuron k at moment n
- Output $y_k(n)$ is obtained using input vector $\boldsymbol{x}(n)$
- Input vector $\boldsymbol{x}(n)$ and desired output $d_k(n)$ represent an example that is presented to ANN at moment n
- Error is the difference between desired and obtained output of neuron k at moment n:

$$e_k(n) = d_k(n) - y_k(n)$$

Error-correction learning

- The goal of error-correction learning is to minimize an error function derived from errors $e_k(n)$ so that the obtained output of all neurons approximates the desired output in some statistical sense
- A frequently used error function is mean square error:

$$J = E\left[\frac{1}{2}\sum_k e_k^2(n)\right]$$

where $E[.]$ is the statistical expectation operator, and summation is for all neurons in the output layer

Error function

- The problem with minimization of error function *J* is that it is necessary to know statistical properties of random processes $e_k(n)$
- For this reason an estimate of the error function in step *n* is used as the optimization function:

$$\mathcal{E}(n) = \frac{1}{2}\sum_k e_k^2(n)$$

- This approach yields an approximate solution

Delta learning rule

- Minimization of error function *J* with respect to weights $w_{kj}(n)$ gives Delta learning rule:

  $$\Delta w_{kj}(n) = \eta \, e_k(n) \, x_j(n)$$

  where η is a positive constant determining the learning rate
- Weight change is proportional to error and to the value at respective input
- Learning rate η must be carefully chosen
  - Small η gives stability but learning is slow
  - Large η speeds up learning but brings instability risk

Error surface

- If we draw error value *J* with respect to synaptic weights we obtain a multidimensional error surface
- The learning problem consists of finding a point on the error surface that has the smallest error (i.e. to minimize the error)

Error surface

- Depending on the type of neurons there are two possibilities:
  - ANN consists of linear neurons – in this case the error surface is a quadratic function with one global minimum
  - ANN consists of nonlinear neurons – in this case the error surface has one or more global minima and multiple local minima
- Learning starts from an arbitrary point on the error surface and through minimization process:
  - In the first case it converges to the global minimum
  - In the second case it can also converge to a local minimum

Hebb learning

- Hebbov principle of learning says (Hebb, The Organization of Behavior, 1942):
  - When axon of neuron A is close enough to activate neuron B and it repeats this many times there will be metabolical changes so that efficiency of neuron A in activating neuron B is increased
- Extension of this principle (Stent, 1973):
  - If one neuron does not influence (stimulate) another neuron then the synapse between them becomes weaker or is completely eliminated

Activity product rule

- According to Hebb principle weights are changed as follows:

  $$\Delta w_{kj}(n) = F(y_k(n), x_j(n))$$

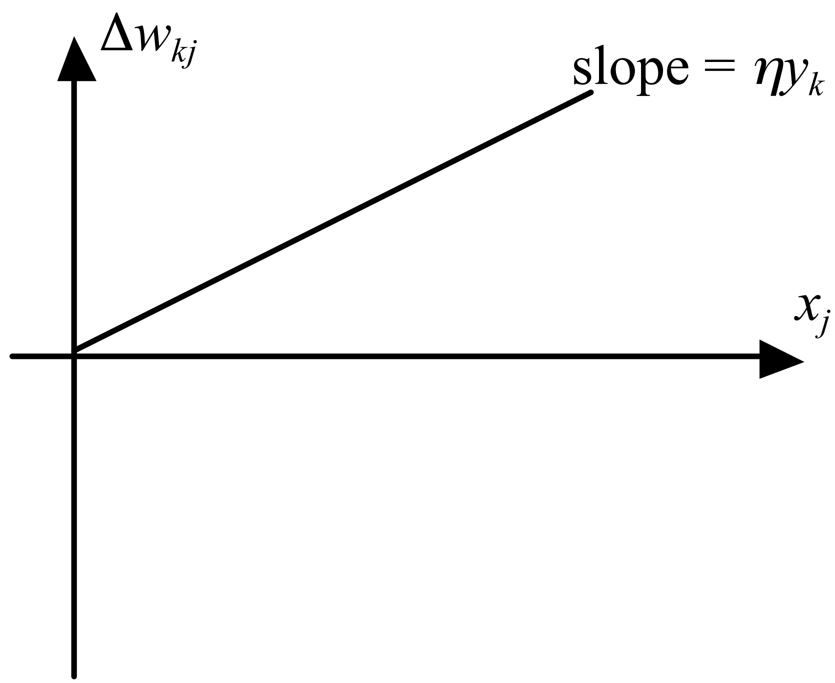
  where $y_k(n)$ *and* $x_j(n)$ are the output and *j*-th input of *k*-th neuron
- A special case of this prinicple is:

  $$\Delta w_{kj}(n) = \eta\, y_k(n)\, x_j(n)$$

  where constant η determines the learning rate
- This rule is called activity product rule

Activity product rule

- Weight update is proportional to input value:
  $\Delta w_{kj}(n) = \eta\, y_k(n)\, x_j(n)$
- Problem: Iterative update with the same input and output causes continuous increase of weight w_{kj}



Generalized activity product rule

- To overcome the problem of weight saturation modifications are porposed that are aimed at limiting the increase of weight w_{kj}
- Non-linear limiting factor (Kohonen, 1988):

$$\Delta w_{kj}(n) = \eta \, y_k(n) \, x_j(n) - \alpha \, y_k(n) \, w_{kj}(n)$$

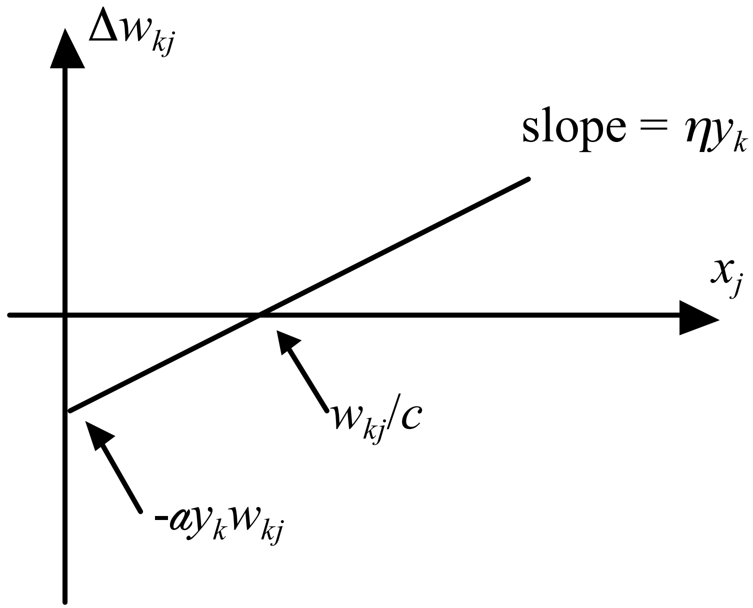
  where α is a positive constant
- This expression can be written as:

$$\Delta w_{kj}(n) = \alpha \, y_k(n)[c x_j(n) - w_{kj}(n)]$$

  where $c = \eta/\alpha$

Generalized activity product rule

- In generalized Hebb rule all inputs such that $x_j(n)<w_{kj}(n)/c$ result in reduction of weight w_{kj}
- Inputs for which $x_j(n)>w_{kj}(n)/c$ increase weight w_{kj}

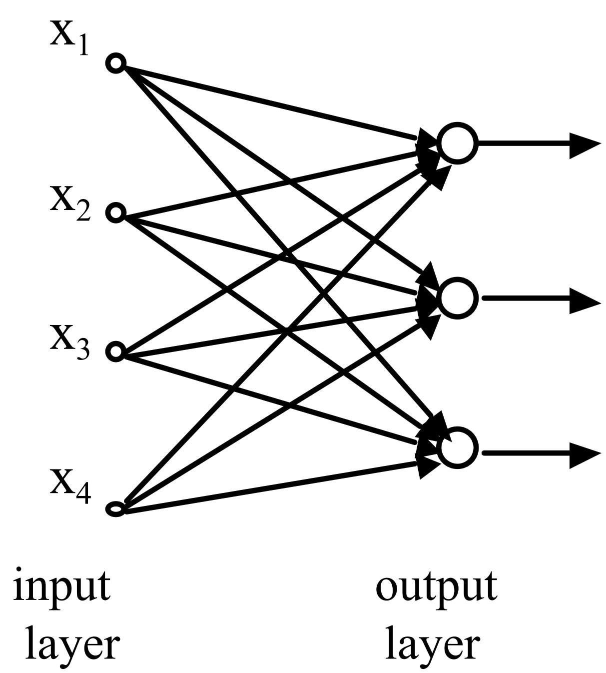


Competitive learning

- Unsupervised learning
- Neurons compete to get opportunity to become active
- Only one neuron can be active at any time
- Useful for classification problems
- Three elements of competitive learning:
  - A set of neurons having randomly selected weights, so they have different response for a given input
  - Limited weight of each neuron
  - Mechanism for competition of neurons so that only one neuron is given at any single time (winner-takes-all neuron)

Competitive learning

- An example network with a single neuron layer



Competitive learning

- In order to win, activity v_j of neuron x must be the largest of all neurons
- Output y_j of the winning neuron j is equal to 1; for all other neurons the output is 0
- The learning rule is defined as:

$$\Delta w_{ji} = \begin{cases} \eta\left(x_i - w_{ji}\right) & \text{if neuron j won} \\ 0 & \text{if neuron j lost} \end{cases}$$

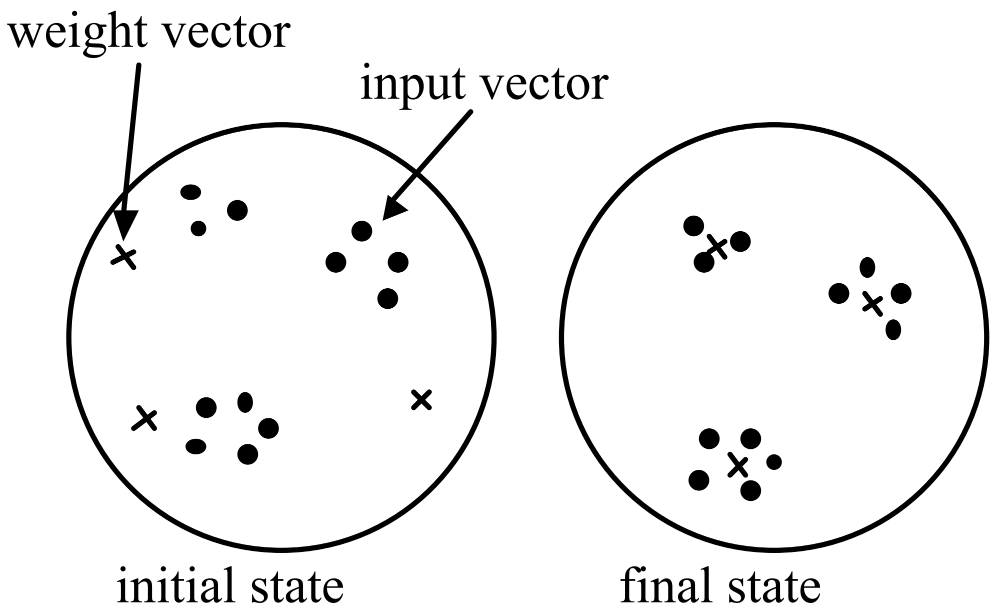
- The learning rule has effect of shifting the weight vector w_j towards the vector x

An example of competitive learning

- Let us assume that each input vector has norm equal to one – so that the vector can be represented as a point on N-dimensional unit sphere
- Let us assume that weight vectors have norm equal to one – so they can also be represented as points on unit N-dimensional sphere
- During training, input vectors are input to the network and the winning neuron weight is updated

An example of competitive learning

- The learning process can be represented as movement of weight vectors along unit sphere

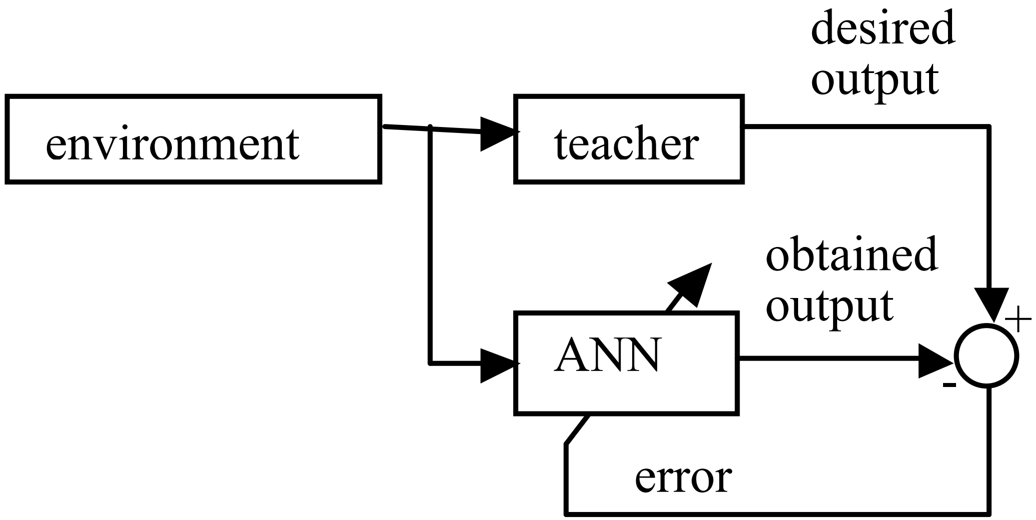


Credit-assignment problem

- Credit-assignment problem is an important issue in learning algorithms
- Credit-assignment problem is in assignment of credit/blame for the overall learning outcome that depends on a large number of internal decisions of the learning system

Supervised learning

- Supervised learning is characterized by the presence of a teacher



Supervised learning

- Teacher has knowledge in the form of input-output pairs used for training
- Error is a difference between desired and obtained output for a given input vector
- ANN parameters change under the influence of input vectors and error values
- The learning process is repeated until ANN learns to imitate the teacher
- After learning is completed, the teacher is no longer required and ANN can work without supervision

Supervised learning

- Error function can be mean square error and it depends on the free parameters (weights)
- Error function can be represented as a multidimensional error surface
- Any ANN configuration is defined by weights and corresponds to a point on the error surface
- Learning process can be viewed as movement of the point down the error surface towards the global minimum of the error surface

Supervised learning

- A point on error surface moves towards the minimum based on gradient
- The gradient at any point on the surface is a vector showing the direction of the steepest ascent

Supervised learning

- Examples of supervised learning algorithms are:
  - LMS (least-mean-square) algorithm
  - BP (back-propagation) algorithm
- A disadvantage of supervised learning is that learning is not possible without a teacher
  - ANN can only learn based on provided examples

Supervised learning

- Suppervised learning can be implemented to work offline or online
- In offline learning:
  - ANN learns first
  - When learning is completed ANN does not change any more
- In online learning:
  - ANN learns during exploitation phase
  - Learning is perfomed in real-time – ANN is dynamic

Reinforcement learning

- Reinforcement learning is of an online character
- Input-output learning mapping is learned through the iterative process where a measure of learning quality is maximized
- Reinforcement learning overcomes the problem of supervised learning where training examples are required

Reinforcement learning

- In reinforcement learning the teacher does not present input-output training examples, but only gives a grade representing a measure of learning quality
- The grade is a scalar value (a number)
- Error function is unknown in reinforcement learning
- Learning algorithm must determine direction of motion in the learning space through a trial-and-error approach

Thorndike law of effect

- Reinforcement learning principle:
  - If learning system actions result in positive grade then there is higher likelihood that the system will take similar actions in the future
  - Otherwise the likelihood of taking such actions is reduced

Unsupervised learning

- In unsupervised learning there is no teacher assisting the learning process
- Competitive learning is an example of unsupervised learning

Unsupervised learning

- A layer of neurons compete for a chance to learn (to modify their weights based on the input vector)
- In the simplest approach the winner-takes-all strategy is used

Comparison of supervised and unsupervised learning

- The most popular algorithm for supervised learning is error-backpropagation algorithm
- A disadvantage of this algorithm is bad scaling – learning complexity grows exponentially with the number of layers

Problems

- Problem 2.1.
  - Delta rule and Hebb rule are two different learning algorithms. Describe differences between these rules.
- Problem 2.5.
  - Input of value 1 is connected to the input of synaptic weight with initial value equal to 1. Calculate weight update using:
    - Basic Hebb rule with learning rate parametrom h=0.1
    - modified Hebb rule with h=0.1 and *c*=0.1