Introduction to Artificial Neural Networks

Lecture 2:

Models of Neuron and Basic Structures and Properties of Artificial Neural Networks

By: Ali Motie Nasrabadi

Outline

- A simplistic model of a biological neuron
- Models of artificial neurons
- Types of activation functions
- Neural network as directed Graph
- Network Architecture
- How many kinds of NNs exist?
- Introduction to learning
- Knowledge Representation

A simplistic model of a biological neuron

- Neuron is information processing unit
- A set of synapses or connecting links
  - characterized by weight or strength
- An adder
  - summing the input signals weighted by synapses
  - a linear combiner
- An activation function
  - also called squashing function
    - squash (limits) the output to some finite values

Lecture 2-3

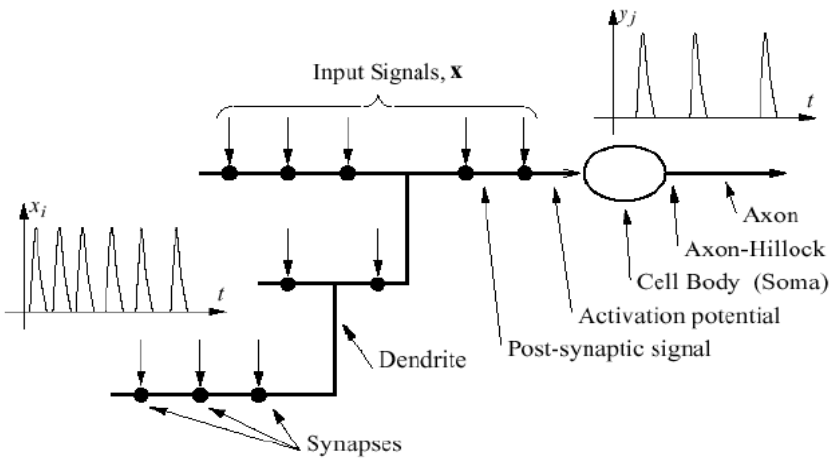
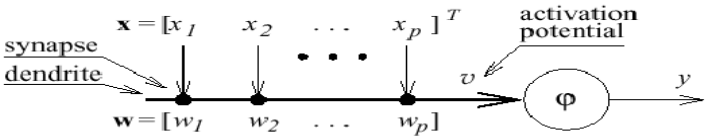


Figure 2–1: Conceptual structure of a biological neuron

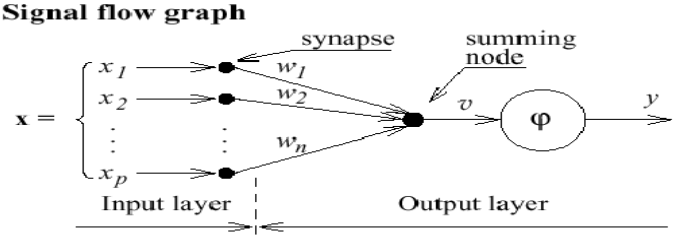
Lecture 2-4

Models of artificial neurons

a. **Dendritic representation**



b. **Signal flow graph**



c. **Block-diagram representation**

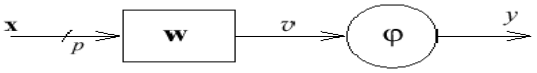
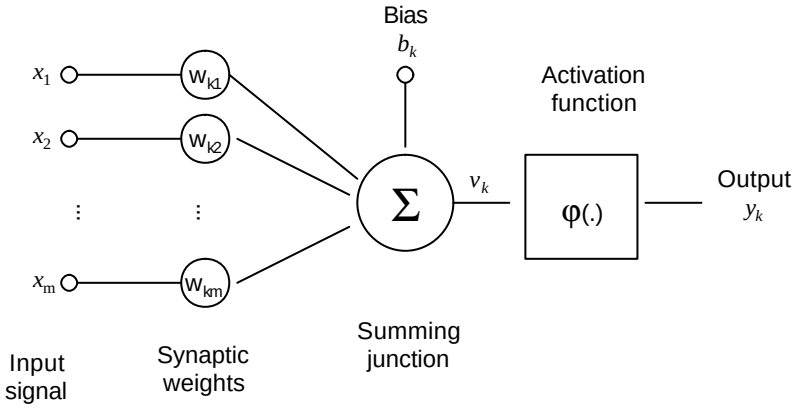


Figure 2.2. Three basic graphical representations of a single p- input (p- synapse) neuron

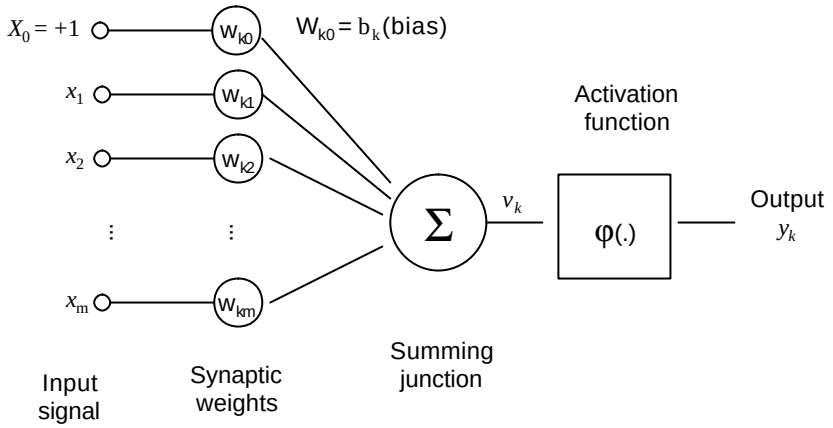
Lecture 2-5

Nonlinear model of a neuron (1)
McCulloch-Pitts Neuron



Lecture 2-6

Nonlinear model of a neuron (2)



Lecture 2-7

The activation potential is formed as a linear combination of input signals and synaptic strength parameters, that is, as an inner product of the weight and input vectors:

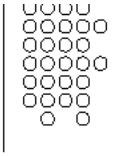
$$v = \sum_{i=1}^{p} w_i x_i = \mathbf{w} \cdot \mathbf{x} = \begin{bmatrix} w_1 & w_2 & \cdots & w_p \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_p \end{bmatrix} \qquad (2.1)$$

Lecture 2-8

Types of activation functions

- **Typically, the activation function generates either** unipolar **or** bipolar **signals.**
- **Different types of activation functions:**
  - Linear function
  - Step function
  - Step function with bias
  - Piecewise- linear function
  - Sigmoidal functions

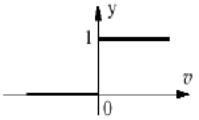
Types of activation functions *(cont.)*



- Linear function
  - $y = v$.
  - Such linear processing elements, sometimes called ADALINEs, are studied in the theory of linear systems, for example, in the "traditional" signal processing and statistical regression analysis.
- Step function
  - Such a processing element is traditionally called **Perceptron**, and it works as a threshold element with a binary output.

unipolar:

$$y = \varphi(v) = \begin{cases} 1 & \text{if } v \geq 0 \\ 0 & \text{if } v < 0 \end{cases}$$

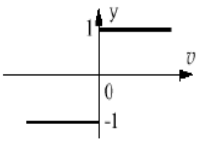


Types of activation functions *(cont.)*

- Step function (*cont.):* bipolar or *signum* function

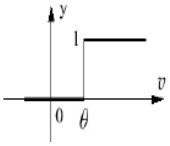
bipolar:

$$y = \varphi(v) = \begin{cases} +1 & \text{if } v \geq 0 \\ -1 & \text{if } v < 0 \end{cases}$$



- A step function with bias
  - The bias (threshold) can be added to both, unipolar and bipolar step function. We then say that a neuron is "fired", when the synaptic activity exceeds the threshold level, θ.
  - For a unipolar case, we have: *(McCulloch-Pits perceptron-1943)*

$$y = \varphi(v) = \begin{cases} 1 & \text{if } \mathbf{w} \cdot \mathbf{x} \geq \theta \\ 0 & \text{if } \mathbf{w} \cdot \mathbf{x} < \theta \end{cases}$$

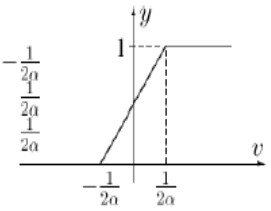


Lecture 2-11

Types of activation functions *(cont.)*

- Piecewise- linear function

$$y = \varphi(v) = \begin{cases} 0 & \text{if } v \leq -\frac{1}{2\alpha} \\ \alpha v + \frac{1}{2} & \text{if } |v| < \frac{1}{2\alpha} \\ 1 & \text{if } v \geq \frac{1}{2\alpha} \end{cases}$$



- For small activation potential v, the neuron works as a linear combiner (an ADALINE) with the gain (slope) α.
- For large activation potential, $|v|$, the neuron saturates and generates the output signal either 0 or 1.
- For large gains $\alpha \to \infty$, the piecewise- linear function is reduced to a step function.

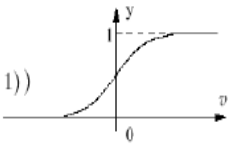
Lecture 2-12

Types of activation functions *(cont.)*

- Sigmoidal function

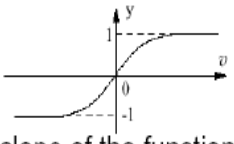
unipolar:

$$\varphi(v) = \frac{1}{1+e^{-\alpha v}} = \frac{1}{2}(\tanh(\alpha v/2)+1))$$



bipolar:

$$\varphi(v) = \tanh(\alpha v)$$



- The parameter α controls the slope of the function.
- The hyperbolic tangent (bipolar sigmoidal) function is perhaps the most popular choice of the activation function specifically in problems related to function mapping and approximation.

Lecture 2-13

Types of activation functions(1): concluding remarks

- **The smooth activation functions, like sigmoidal, or Gaussian, for which a continuous derivative exists, are typically used in networks performing a function approximation task.**
- **The step functions are used as parts of pattern classification networks.**
- **Many learning algorithms, like Back-propagation, require calculation of the derivative of the activation function.**

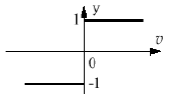
Lecture 2-14

Types of activation functions *(2)*

- **Step function (*cont.):* bipolar or *signum* function**

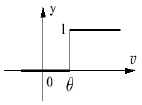
bipolar:

$$y = \varphi(v) = \begin{cases} +1 & \text{if } v \geq 0 \\ -1 & \text{if } v < 0 \end{cases}$$



- **A step function with bias**
  - **The bias (threshold) can be added to both, unipolar and bipolar step function. We then say that a neuron is "fired", when the synaptic activity exceeds the threshold level, ?.**
  - **For a unipolar case, we have: *(McCulloch-Pits perceptron-1943)***

$$y = \varphi(v) = \begin{cases} 1 & \text{if } \mathbf{w} \cdot \mathbf{x} \geq \theta \\ 0 & \text{if } \mathbf{w} \cdot \mathbf{x} < \theta \end{cases}$$



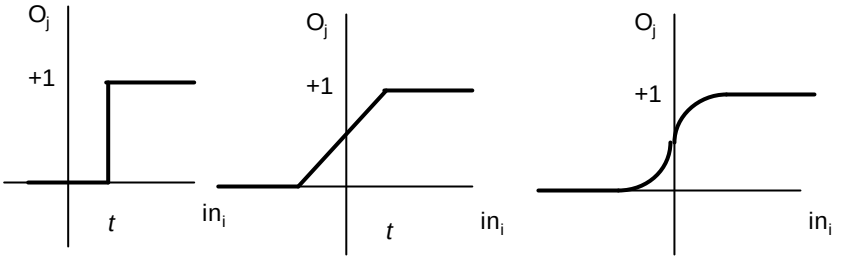
Lecture 2-15

Types of activation functions *(3)*

- **Piecewise- linear function**
- **For small activation potential, , the neuron works as a**
- **linear combiner (an ADALINE) with the gain (slope) .**
- **For large activation potential, the neuron saturates and generates the output signal either 0 or 1.**
- **For large gains , the piecewise- linear function is reduced to a step function.**

Lecture 2-16

Types of Activation Function(4)



Threshold Function

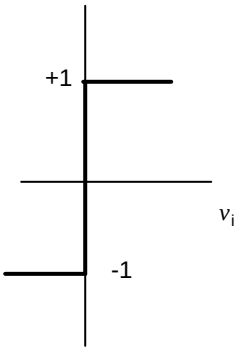
Piecewise-linear Function

Sigmoid Function (differentiable)

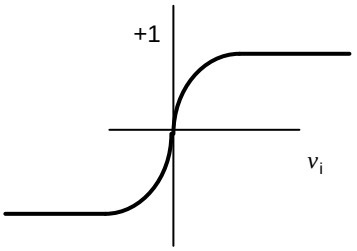
a is slope parameter

Lecture 2-17

Activation Function value range



Signum Function



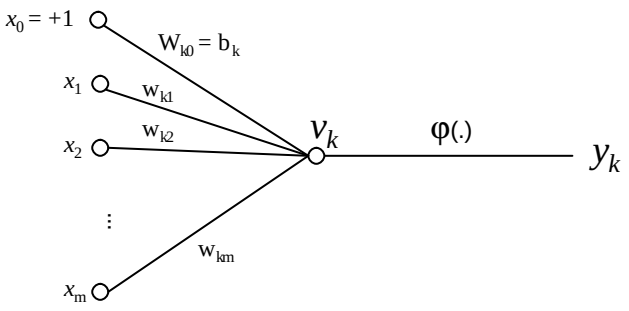
Hyperbolic tangent Function

Lecture 2-18

Neural network as directed Graph

- Block diagram can be simplify by the idea of signal flow graph
- node is associated with signal
- directed link is associated with transfer function
  - synaptic links
    - governed by linear input-output relation
    - signal x_j is multiplied by synaptic weight w_{kj}
  - activation links
    - governed by nonlinear input-output relation
    - nonlinear activation function

Signal Flow Graph of a Neuron



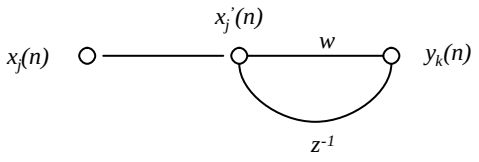
Architectural graph of a Neuron

- Partially complete directed graph describing layout
- Computation node : shaded
- source node : small square

- Three graphical representations
  - Block diagram - providing functional description of a NN
  - Signal flow graph - complete description of signal flow
  - architectural graph - network layout

Feedback

- Output determines in part own output via feedback

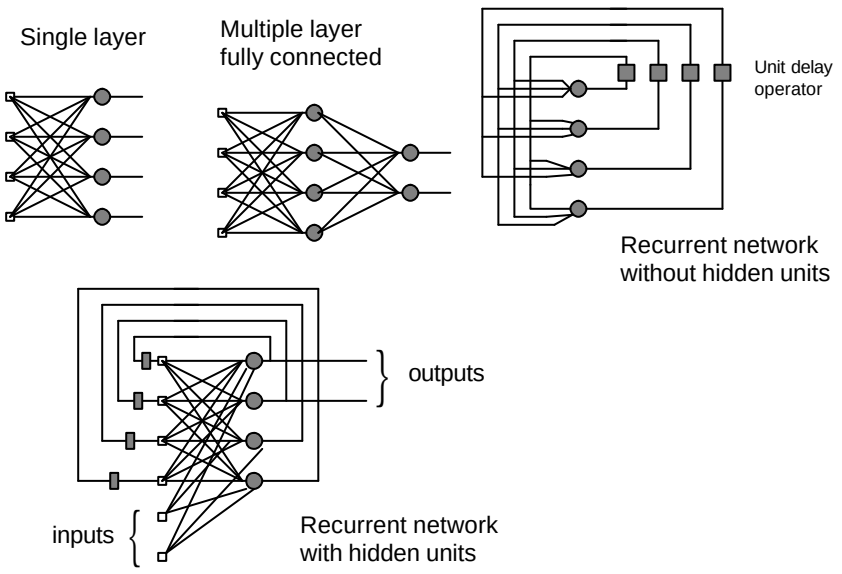


- depending on w
  - stable, linear divergence, exponential divergence
  - we are interested in the case of |w| <1 ; infinite memory
    - output depends on inputs of infinite past
- NN with feedback loop : recurrent network

Network Architecture

- **Single-layer Feedforward Networks**
  - **input layer and output layer**
    - **single (computation) layer**
  - **feedforward, acyclic**
- **Multilayer Feedforward Networks**
  - **hidden layers - hidden neurons and hidden units**
  - **enables to extract high order statistics**
  - **10-4-2 network, 100-30-10-3 network**
  - **fully connected layered network**
- **Recurrent Networks**
  - **at least one feedback loop**
  - **with or without hidden neuron**

Network Architecture



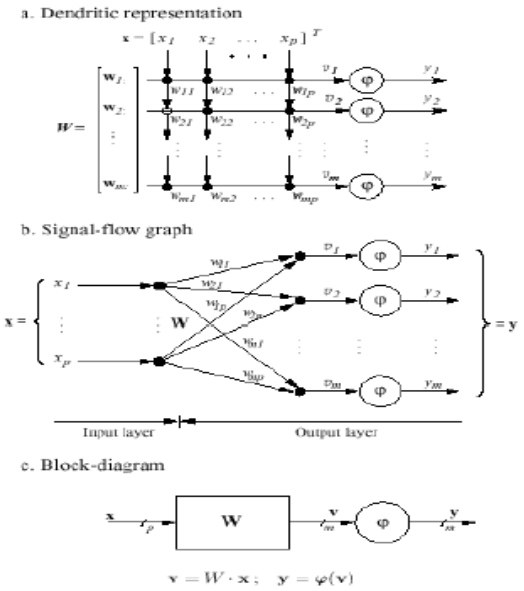


Figure 2–5: Three basic graphical representations of a p-input m-neuron single layer neural network

Lecture 2-25

Taxonomy of neural networks

- **Two phases of ANNs:**
  - Learning **or** encoding **phase;**
  - Active **or** decoding **phase.**
- **From the point of view of their** learning **or** encoding **phase, artificial neural networks can be classified into:**
  - Supervised Learning **(learning with a teacher)**
  - Unsupervised Learning **(learning with no help)**
  - **Reinforcement** Learning **(learning with limited feedback)**
- **From the point of view of their** active **or** decoding **phase, artificial neural networks can be classified into:**
  - feedforward **(static)**
  - feedback **(dynamic, recurrent)**

Lecture 2-26

How many kinds of NNs exist?

- Nobody knows exactly how many
- There is a collection of some of the most well known methods, not claiming to be complete.
- The two main kinds of learning algorithms are supervised and unsupervised:
  - In supervised learning, the correct results (target values, desired outputs) are known and are given to the NN during training so that the NN can adjust its weights to try match its outputs to the target values. After training, the NN is tested by giving it only input values, not target values, and seeing how close it comes to outputting the correct target values.
  - In unsupervised learning, the NN is not provided with the correct results during training. Unsupervised NNs usually perform some kind of data compression, such as dimensionality reduction or clustering. See "What does unsupervised learning learn?"

- Two major kinds of network topology are feedforward and feedback.
  - In a feedforward NN, the connections between units do not form cycles. Feedforward NNs usually produce a response to an input quickly. Most feedforward NNs can be trained using a wide variety of efficient conventional numerical methods (e.g. see "What are conjugate gradients, Levenberg-Marquardt, etc.?") in addition to algorithms invented by NN researchers.
  - In a feedback or recurrent NN, there are cycles in the connections. In some feedback NNs, each time an input is presented, the NN must iterate for a potentially long time before it produces a response. Feedback NNs are usually more difficult to train than feedforward NNs.

- **NNs also differ in the kinds of data they accept. Two major kinds of data are categorical and quantitative.**
  - **Categorical variables take only a finite (technically, countable) number of possible values, and there are usually several or more cases falling into each category. Categorical variables may have symbolic values (e.g., "male" and "female", or "red", "green" and "blue") that must be encoded into numbers before being given to the network. Both supervised learning with categorical target values and unsupervised learning with categorical outputs are called "classification."**
  - **Quantitative variables are numerical measurements of some attribute, such as length in meters. The measurements must be made in such a way that at least some arithmetic relations among the measurements reflect analogous relations among the attributes of the objects that are measured. Supervised learning with quantitative target values is called "regression."**

- **Supervised**
  - **1. Feedforward**
    - **Linear**
      - **Hebbian - Hebb (1949), Fausett (1994)**
      - **Perceptron - Rosenblatt (1958), Minsky and Papert (1969/1988), Fausett (1994)**
      - **Adaline - Widrow and Hoff (1960), Fausett (1994) o**
      - **Higher Order - Bishop (1995)**
      - **Functional Link - Pao (1989)**
    - **MLP: Multilayer perceptron - Bishop (1995), Reed and Marks (1999), Fausett (1994)**
      - **Backprop - Rumelhart, Hinton, and Williams (1986)**
      - **Cascade Correlation - Fahlman and Lebiere (1990), Fausett (1994)**
      - **Quickprop - Fahlman (1989)**
      - **RPROP - Riedmiller and Braun (1993)**
    - **RBF networks - Bishop (1995), Moody and Darken (1989), Orr (1996)**
      - **OLS: Orthogonal Least Squares - Chen, Cowan and Grant (1991)**
    - **CMAC: Cerebellar Model Articulation Controller - Albus (1975), Brown and Harris (1994)**
    - **Classification only**
      - **LVQ: Learning Vector Quantization - Kohonen (1988), Fausett (1994) o**
      - **PNN: Probabilistic Neural Network - Specht (1990), Masters (1993), Hand (1982), Fausett (1994) o Regression only**
      - **GNN: General Regression Neural Network - Specht (1991), Nadaraya (1964), Watson (1964)**

- **Unsupervised - Hertz, Krogh, and Palmer (1991)**
  - **Competitive**
    - **Vector Quantization**
      - **Grossberg - Grossberg (1976)**
      - **Kohonen - Kohonen (1984)**
      - **Conscience - Desieno (1988)**
    - **Self-Organizing Map**
      - **Kohonen - Kohonen (1995), Fausett (1994)**
      - **GTM: - Bishop, Svens and Williams (1997)**
      - **Local Linear - Mulier and Cherkassky (1995)**
    - **Adaptive resonance theory**
      - **ART 1 - Carpenter and Grossberg (1987a), Moore (1988), Fausett (1994)**
      - **ART 2 - Carpenter and Grossberg (1987b), Fausett (1994)**
      - **ART 2-A - Carpenter, Grossberg and Rosen (1991a)**
      - **ART 3 - Carpenter and Grossberg (1990)**
      - **Fuzzy ART - Carpenter, Grossberg and Rosen (1991b)**
    - **DCL: Differential Competitive Learning - Kosko (1992)**
  - **2. Dimension Reduction - Diamantaras and Kung (1996)**
    - **Hebbian - Hebb (1949), Fausett (1994)**
    - **Oja - Oja (1989) o Sanger - Sanger (1989)**
    - **Differential Hebbian - Kosko (1992)**
  - **3. Autoassociation**
    - **Linear autoassociator - Anderson et al. (1977), Fausett (1994)**
    - **BSB: Brain State in a Box - Anderson et al. (1977), Fausett (1994)**
    - **Hopfield - Hopfield (1982), Fausett (1994**

Lecture 2-31

Introduction to learning

- **Learning**
  - **In Biological networks is through to occur when modifications are made to the effective coupling between one cell and another , at the synaptic junction**
  - **In ANN is a dynamic process which modifies the weights of the network in some desirable way**
- **In the** decoding **part of a neural network, one assumes that the** weight matrix**, *W*, is given. If the weight matrix is satisfactory, during the decoding process the network performs some useful task it has been design to do.**
- **In simple or specialized cases the weight matrix can be precomputed, but more commonly it is obtained through the** learning **process.**
- **Learning is a dynamic process which modifies the weights of the network in some desirable way. As any dynamic process learning can be described either in the continuous- time or in the discrete-time framework.**

Lecture 2-32

Introduction to learning *(cont.)*

- **Consider a neural network as in Figure 2–7:**

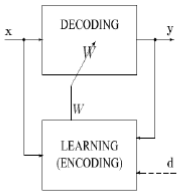


Figure 2– 7: A neural network with a learning part

- **The learning can be described either by differential equations (continuous- time) or by the difference equations (discrete- time).**

Introduction to learning *(cont.)*

- **Continuous- time case:**

$$\dot{W}(t) = L(\, W(t), \mathbf{x}(t), \mathbf{y}(t), \mathbf{d}(t)\,)$$

- **Discrete- time case:**

$$W(n+1) = L(\, W(n), \mathbf{x}(n), \mathbf{y}(n), \mathbf{d}(n)\,)$$

- **where** d **is an external teaching or supervising signal used in** supervised learning**. This signal in not present in networks employing** unsupervised learning**.**
- **The discrete- time learning law is often used in a form of a** weight update **equation:**

$$\begin{aligned} W(n+1) &= W(n) + \Delta W(n) \\ \Delta W(n) &= L(\, W(n), \mathbf{x}(n), \mathbf{y}(n), \mathbf{d}(n)\,) \end{aligned}$$

Knowledge Representation

- Knowledge refers to stored information or models used by a person or machine to interpret, predict and appropriately respond to the outside world
  - What information is actually made explicit;
  - How the information is physically encoded for the subsequent use
- Good solution depends on good representation of knowledge
- In NN, knowledge is represented by internal network parameters
  - real challenge
- Knowledge of the world
  - world state represented by known facts - prior knowledge
  - observations - obtained by (noisy) sensors; training examples

Knowledge Acquisition by NN Training

- Training examples : either labeled or unlabeled
  - labeled : input signal and desired response
  - unlabeled : different realizations of input signal
  - Examples represent the knowledge of environment
- Handwritten digit recognition
  1. Appropriate architecture is selected for NN
     - source node = number of pixels of input image
     - 10 output node for each digit
     - subset of examples for training NN
     - by suitable learning algorithm
  2. Recognition performance is tested by the rest of the examples
- Positive and negative examples

Rules of Knowledge representation in NN

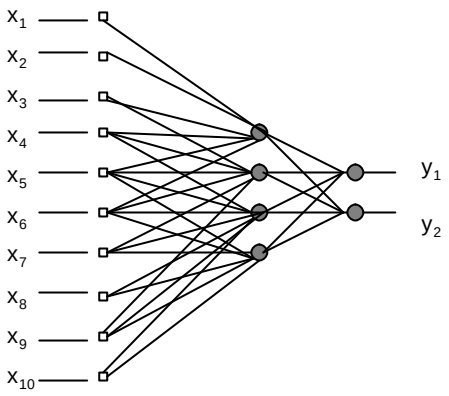
1. Similar input from similar classes produce similar representations
   - similarity measures
     - Euclidian distance, dot (inner) product, cos
     - random variable : Mahalanobis distance
     - ...
2. Separate classes produce widely different representations
3. More neurons should be involved in representation of more important feature
   - probability of detection / false alarm
4. Prior information and invariances should be built into the design of the network
   - general purpose vs specialized

Lecture 2-37

Building Prior to NN design

- Specialized structure
  - learns fast because of small free parameters
  - runs fast because of simple structure
- No well-defined rules for building specialized NN
  - ad hoc approach
  - restricting the network architecture thru use of local connection
    - receptive field
  - Constraining the choice of synaptic weights
    - weight sharing, parameter tying

Lecture 2-38



Combining receptive field and weight-sharing (convolution network)

Lecture 2-39

Building invariance to NN design

- **Want to be capable to cope with transformations**
  - **Invariance by structure**
    - **synaptic connections are arranged not to affected by the transform**
    - **rotation invariant forcing $w_{ji} = w_{jk}$ for all k in the same distance from the center of image**
  - **Invariance by training**
    - **train by data of many different transformations**
    - **computationally infeasible**
  - **invariant feature space**
    - **use features invariant to the transformations**
- **No well-developed theory of optimizing architecture of NN**
- **NN lacks explanation capability**

Lecture 2-40