OMICS GROUP

OMICS Group International through its Open Access Initiative is committed to make genuine and reliable contributions to the scientific community. OMICS Group hosts over 400 leading-edge peer reviewed Open Access Journals and organizes over 300 International Conferences annually all over the world. OMICS Publishing Group journals have over 3 million readers and the fame and success of the same can be attributed to the strong editorial board which contains over 30000 eminent personalities that ensure a rapid, quality and quick review process. OMICS Group signed an agreement with more than 1000 International Societies to make healthcare information Open Access.

OMICS Journals are welcoming Submissions

OMICS Group welcomes submissions that are original and technically so as to serve both the developing world and developed countries in the best possible way.

OMICS Journals are poised in excellence by publishing high quality research. OMICS Group follows an Editorial Manager® System peer review process and boasts of a strong and active editorial board.

Editors and reviewers are experts in their field and provide anonymous, unbiased and detailed reviews of all submissions.

The journal gives the options of multiple language translations for all the articles and all archived articles are available in HTML, XML, PDF and audio formats. Also, all the published articles are archived in repositories and indexing services like DOAJ, CAS, Google Scholar, Scientific Commons, Index Copernicus, EBSCO, HINARI and GALE.

For more details please visit our website:
http://omicsonline.org/Submitmanuscript.php

Neural Networks

Mario Pavone

Professsor
University of Catania
Italy

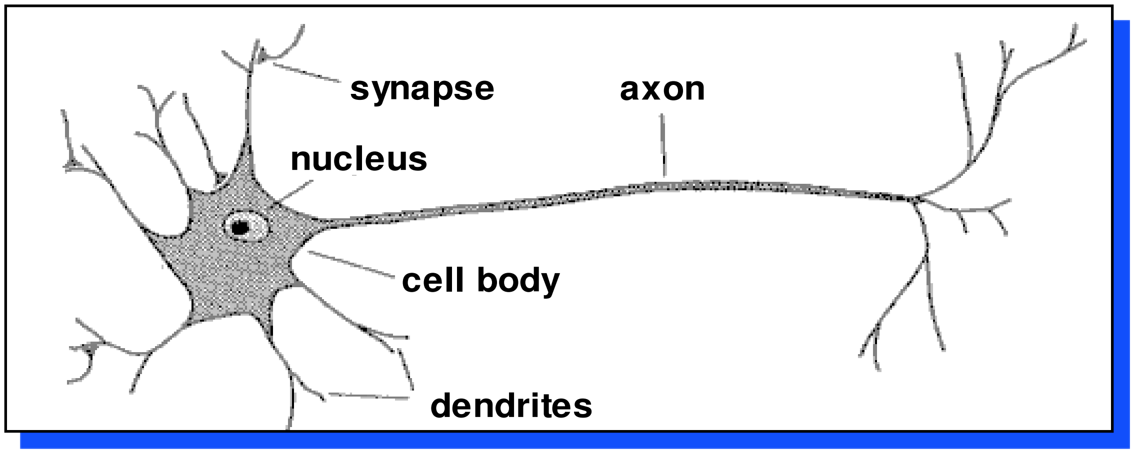
Recently published articles

- Clonal Selection - An Immunological Algorithm for Global Optimization over Continuous Spaces
- Swarm Intelligence Heuristics for Graph Coloring Problem
- O-BEE-COL: Optimal BEEs for COLoring Graphs
- Escaping Local Optima via Parallelization and MigrationProtein Multiple Sequence Alignment by Hybrid Bio-Inspired Algorithms
- Effective Calibration of Artificial Gene Regulatory Networks
- Large scale agent-based modeling of the humoral and cellular immune response
- A Memetic Immunological Algorithm for Resource Allocation Problem.

Biological inspirations

- Some numbers…
  - The human brain contains about 10 billion nerve cells (neurons)
  - Each neuron is connected to the others through 10000 synapses
- Properties of the brain
  - It can learn, reorganize itself from experience
  - It adapts to the environment
  - It is robust and fault tolerant

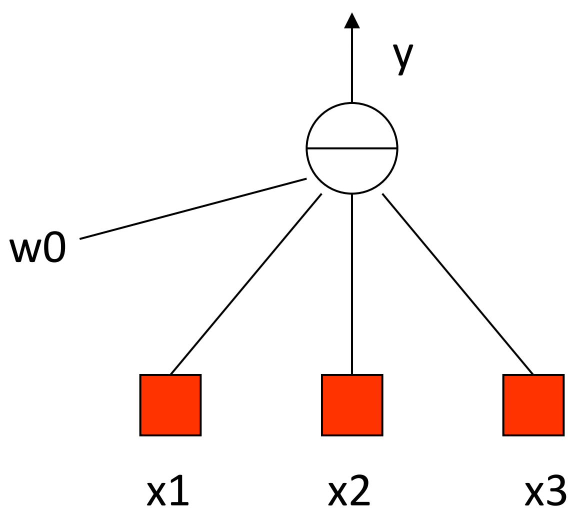
Biological neuron



- A neuron has
  - A branching input (dendrites)
  - A branching output (the axon)
- The information circulates from the dendrites to the axon via the cell body
- Axon connects to dendrites via synapses
  - Synapses vary in strength
  - Synapses may be excitatory or inhibitory

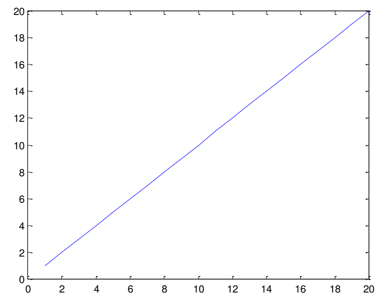
What is an artificial neuron ?

- Definition : Non linear, parameterized function with restricted output range



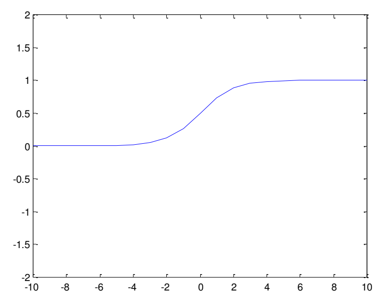
$$y = f\left(w_0 + \sum_{i=1}^{n-1} w_i x_i \right)$$

Activation functions



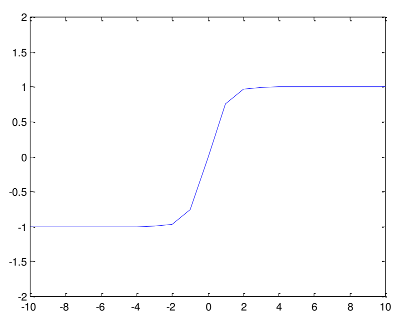
Linear

$$y = x$$



Logistic

$$y = \frac{1}{1 + \exp(-x)}$$



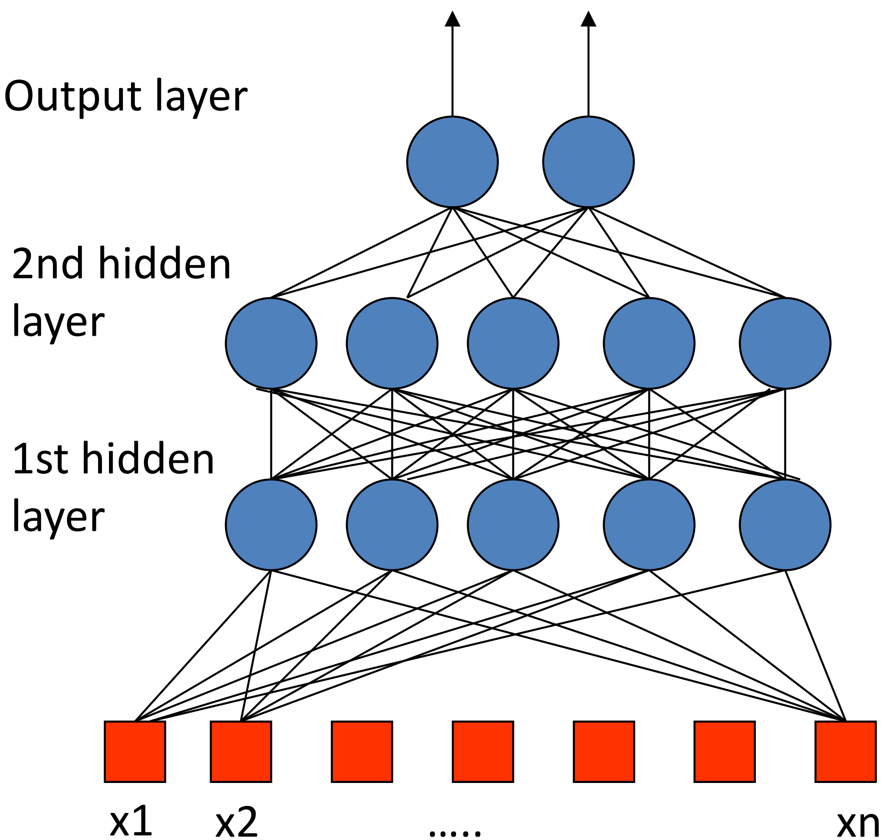
Hyperbolic tangent

$$y = \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)}$$

Neural Networks

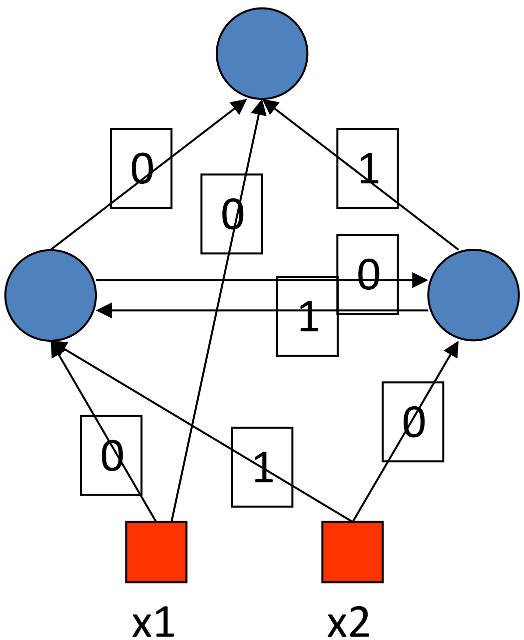
- A mathematical model to solve engineering problems
  - Group of highly connected neurons to realize compositions of non linear functions
- Tasks
  - Classification
  - Discrimination
  - Estimation
- 2 types of networks
  - Feed forward Neural Networks
  - Recurrent Neural Networks

Feed Forward Neural Networks



- The information is propagated from the inputs to the outputs
- Computations of No non linear functions from n input variables by compositions of Nc algebraic functions
- Time has no role (NO cycle between outputs and inputs)

Recurrent Neural Networks



- Can have arbitrary topologies
- Can model systems with internal states (dynamic ones)
- Delays are associated to a specific weight
- Training is more difficult
- Performance may be problematic
  - Stable Outputs may be more difficult to evaluate
  - Unexpected behavior (oscillation, chaos, …)

Learning

- The procedure that consists in estimating the parameters of neurons so that the whole network can perform a specific task

- 2 types of learning
  - The supervised learning
  - The unsupervised learning

- The Learning process (supervised)
  - Present the network a number of inputs and their corresponding outputs
  - See how closely the actual outputs match the desired ones
  - Modify the parameters to better approximate the desired outputs

Supervised learning

- The desired response of the neural network in function of particular inputs is well known.
- A “Professor” may provide examples and teach the neural network how to fulfill a certain task

Unsupervised learning

- Idea : group typical input data in function of resemblance criteria un-known a priori
- Data clustering
- No need of a professor
  - The network finds itself the correlations between the data
  - Examples of such networks :
    - Kohonen feature maps

Properties of Neural Networks

- Supervised networks are universal approximators (Non recurrent networks)
- Theorem : Any limited function can be approximated by a neural network with a finite number of hidden neurons to an arbitrary precision
- Type of Approximators
  - Linear approximators : for a given precision, the number of parameters grows exponentially with the number of variables (polynomials)
  - Non-linear approximators (NN), the number of parameters grows linearly with the number of variables

Other properties

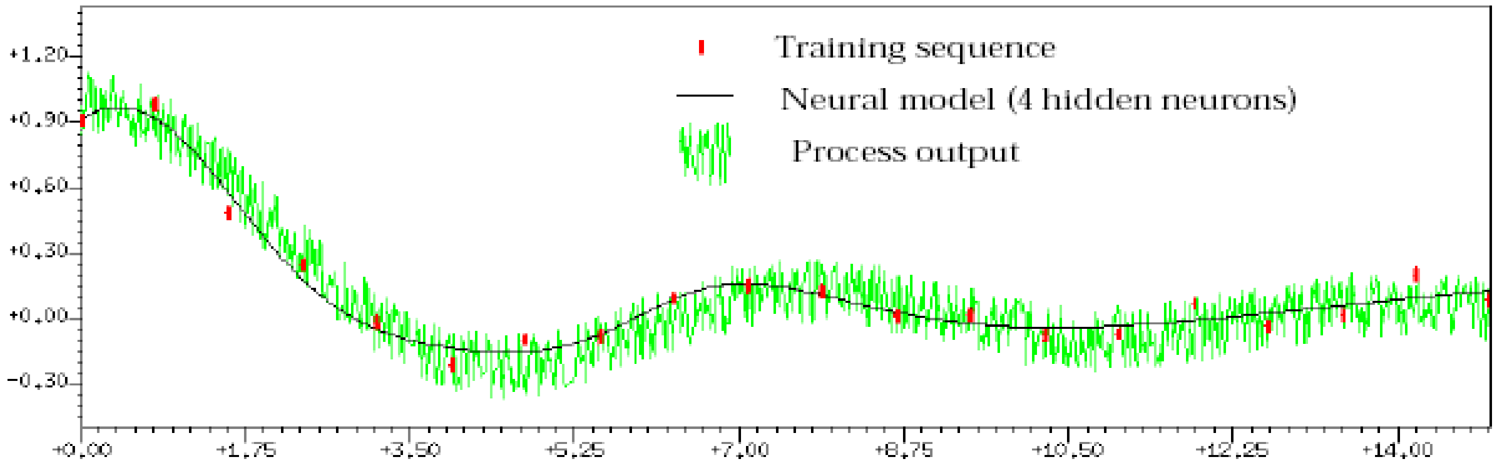
- Adaptivity
  - Adapt weights to environment and retrained easily
- Generalization ability
  - May provide against lack of data
- Fault tolerance
  - Graceful degradation of performances if damaged => The information is distributed within the entire net.

Static modeling

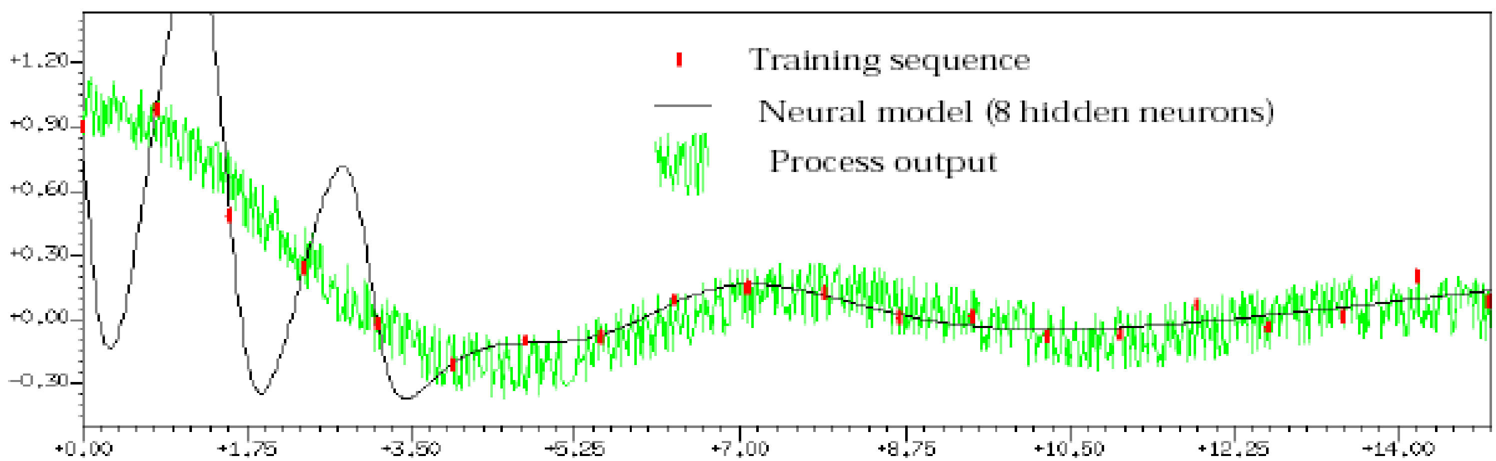
- In practice, it is rare to approximate a known function by a uniform function
- “black box” modeling : model of a process
- The y output variable depends on the input variable x with k=1 to N $\left\{x^k, y_p^k\right\}$
- Goal : Express this dependency by a function, for example a neural network

- If the learning ensemble results from measures, the noise intervenes
- Not an approximation but a fitting problem
- Regression function
- Approximation of the regression function : Estimate the more probable value of yp for a given input x
- Cost function: $$J(w) = \frac{1}{2}\sum_{k=1}^{N}\left[y_p(x^k) - g(x^k, w)\right]^2$$
- Goal: Minimize the cost function by determining the right function g

Example



Training sequence
Neural model (4 hidden neurons)
Process output
+1.20
+0.90
+0.60
+0.30
+0.00
-0.30
+0.00
+1.75
+3.50
+5.25
+7.00
+8.75
+10.50
+12.25
+14.00



Training sequence
Neural model (8 hidden neurons)
Process output
+1.20
+0.90
+0.60
+0.30
+0.00
-0.30
+0.00
+1.75
+3.50
+5.25
+7.00
+8.75
+10.50
+12.25
+14.00

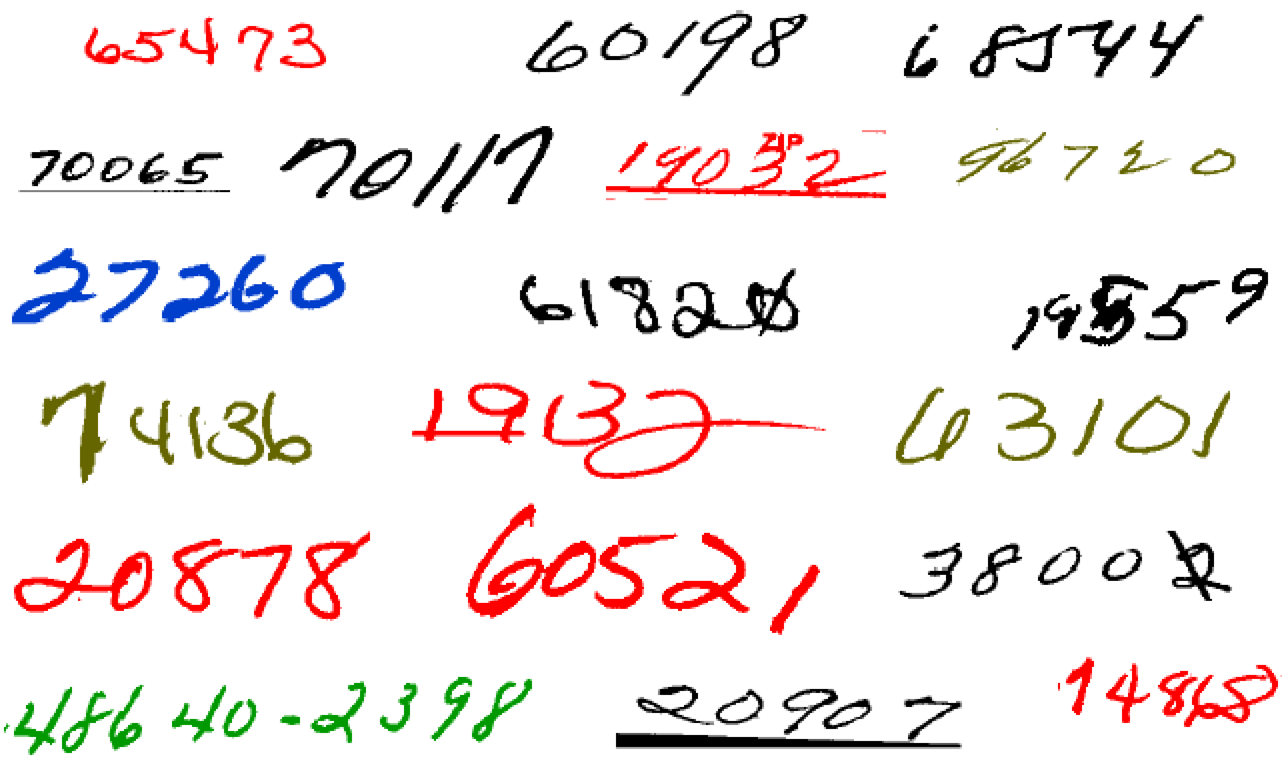
Classification (Discrimination)

- Class objects in defined categories
- Rough decision OR
- Estimation of the probability for a certain object to belong to a specific class

Example : Data mining

- Applications : Economy, speech and patterns recognition, sociology, etc.

Example



Examples of handwritten postal codes
drawn from a database available from the US Postal service

What do we need to use NN ?

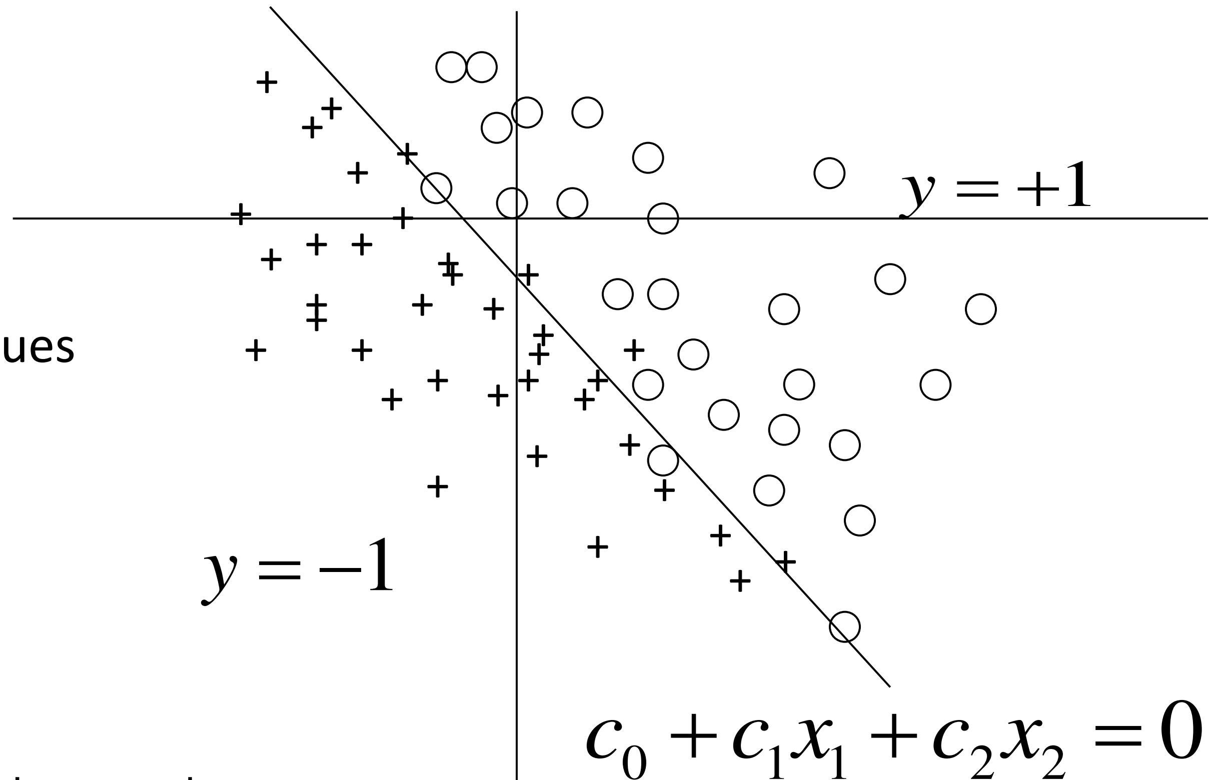
- Determination of pertinent inputs
- Collection of data for the learning and testing phase of the neural network
- Finding the optimum number of hidden nodes
- Estimate the parameters (Learning)
- Evaluate the performances of the network
- IF performances are not satisfactory then review all the precedent points

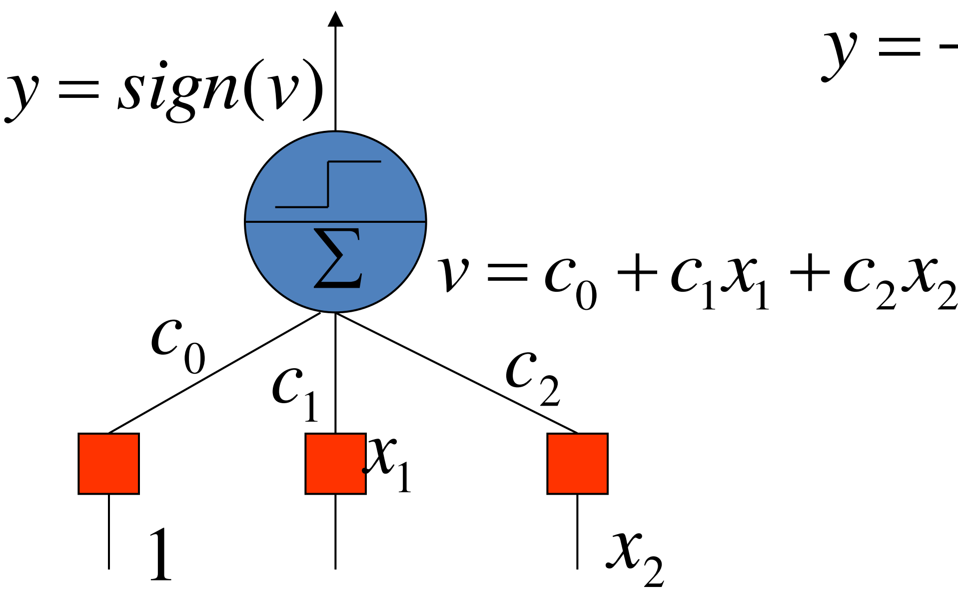
Classical neural architectures

- Perceptron
- Multi-Layer Perceptron
- Radial Basis Function (RBF)
- Kohonen Features maps
- Other architectures
  - An example : Shared weights neural networks

Perceptron

- Rosenblatt (1962)
- Linear separation
- Inputs :Vector of real values
- Outputs :1 or -1





Learning (The perceptron rule)

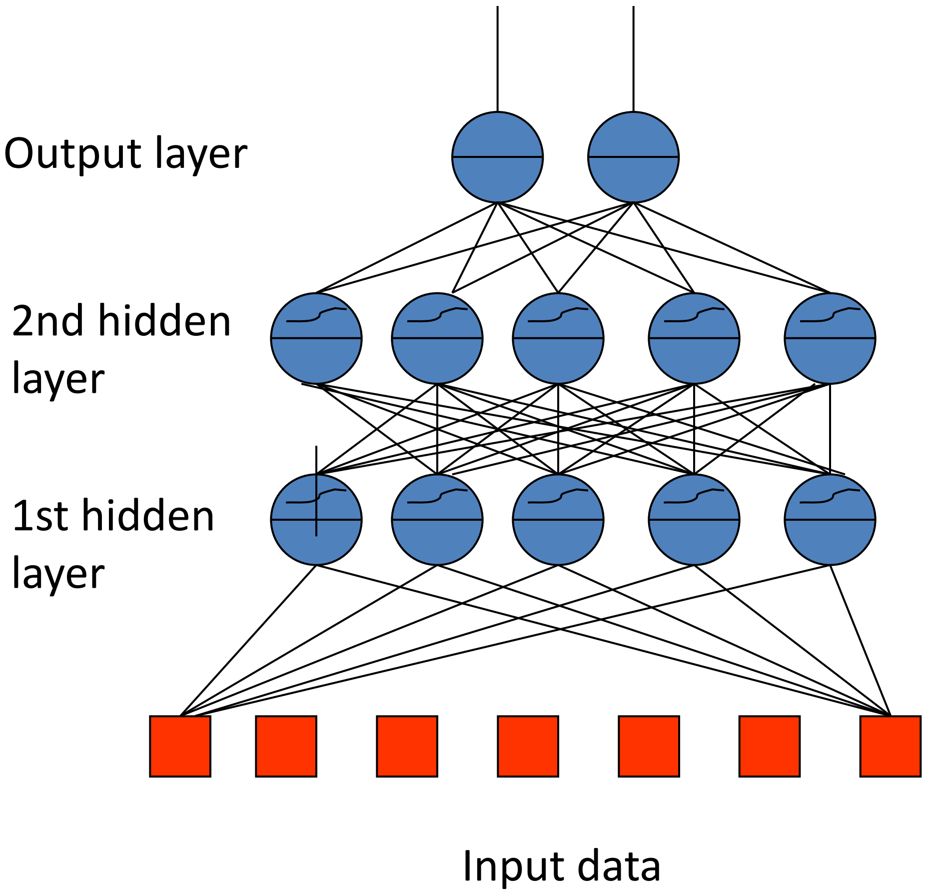
- Minimization of the cost function : $J(c) = \sum_{k \in M} -y_p^k v^k$
- J(c) is always >= 0 (M is the ensemble of bad classified examples)
- y_p^k is the target value
- Partial cost
  - If x^k is not well classified : $J^k(c) = -y_p^k v^k$
  - If x^k is well classified $J^k(c) = 0$
- Partial cost gradient $\longrightarrow$ $\boxed{\dfrac{\partial J^k(c)}{\partial c} = -y_p^k x^k}$
- Perceptron algorithm

$$\text{if } y_p^k v^k > 0 \ \ (\mathrm{x^k \text{ is well classified}}) : \mathrm{c(k) = c(k-1)}$$

$$\text{if } y_p^k v^k < 0 \ \ (\mathrm{x^k \text{ is not well classified}}) : \mathrm{c(k) = c(k-1)} + y_p^k x^k$$

- The perceptron algorithm converges if examples are linearly separable

Multi-Layer Perceptron



- One or more hidden layers
- Sigmoid activations functions

Learning

- Back-propagation algorithm

Credit assignment

$$\delta_j = -\frac{\partial E}{\partial net_j}$$

$$net_j = w_{j0} + \sum_i^n w_{ji} o_i$$

$$o_j = f_j\left(net_j\right)$$

$$\Delta w_{ji} = -\alpha \frac{\partial E}{\partial w_{ji}} = -\alpha \frac{\partial E}{\partial net_j} \frac{\partial net_j}{\partial w_{ji}} = \alpha \delta_j o_i$$

$$\delta_j = -\frac{\partial E}{\partial o_j} \frac{\partial o_j}{\partial net_j} = -\frac{\partial E}{\partial o_j} f'(net_j)$$

$$E = \frac{1}{2}(t_j - o_j)^2 => \frac{\partial E}{\partial o_j} = -(t_j - o_j)$$

If the jth node is an output unit

$$\delta_j = (t_j - o_j) f'(net_j)$$

$$\frac{\partial E}{\partial o_j} = \sum_k^{\kappa} \frac{\partial E}{\partial net_\kappa} \frac{\partial net_\kappa}{\partial o_j} = -\sum_k^{\kappa} \delta_k w_{kj}$$

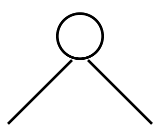
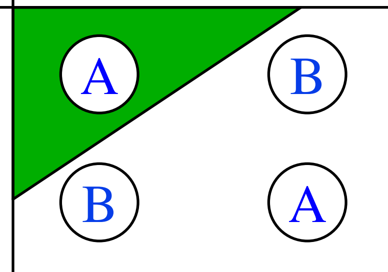
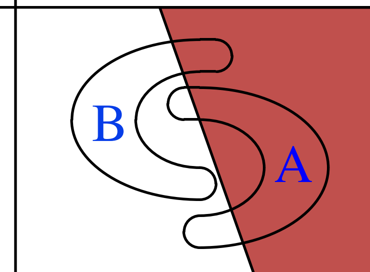
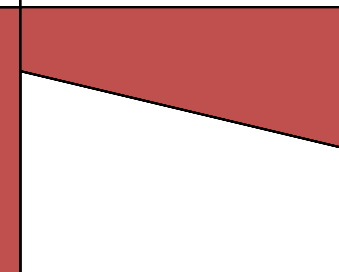
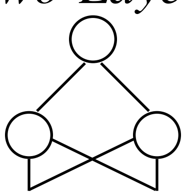
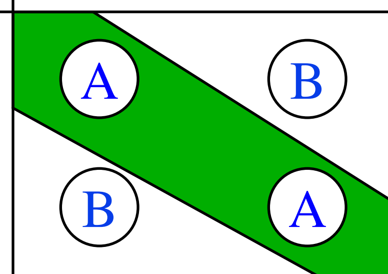
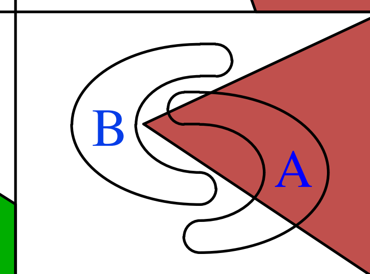
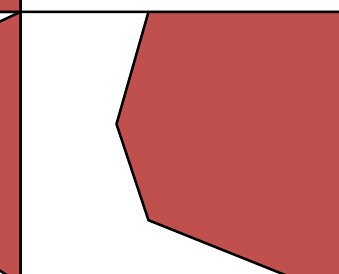
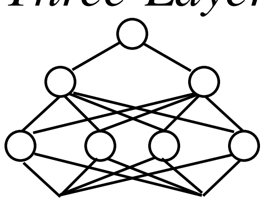
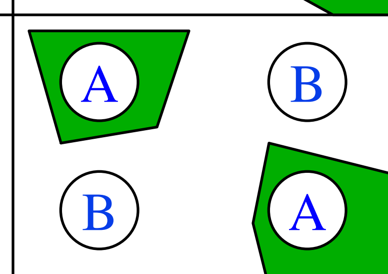
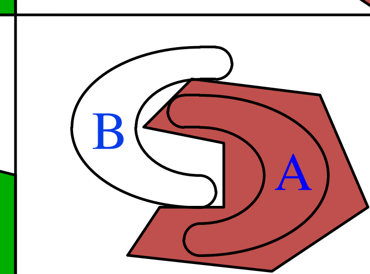
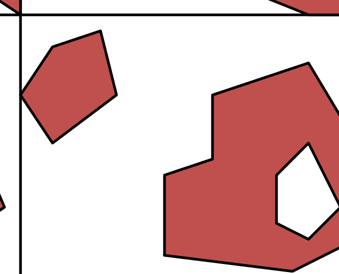
$$\delta_j = f'_j(net_j) \sum_k^{\kappa} \delta_k w_{kj}$$

Momentum term to smooth
The weight changes over time

$$\Delta w_{ji}(t) = \alpha \delta_j(t) o_i(t) + \gamma \Delta w_{ji}(t-1)$$

$$w_{ji}(t) = w_{ji}(t-1) + \Delta w_{ji}(t)$$

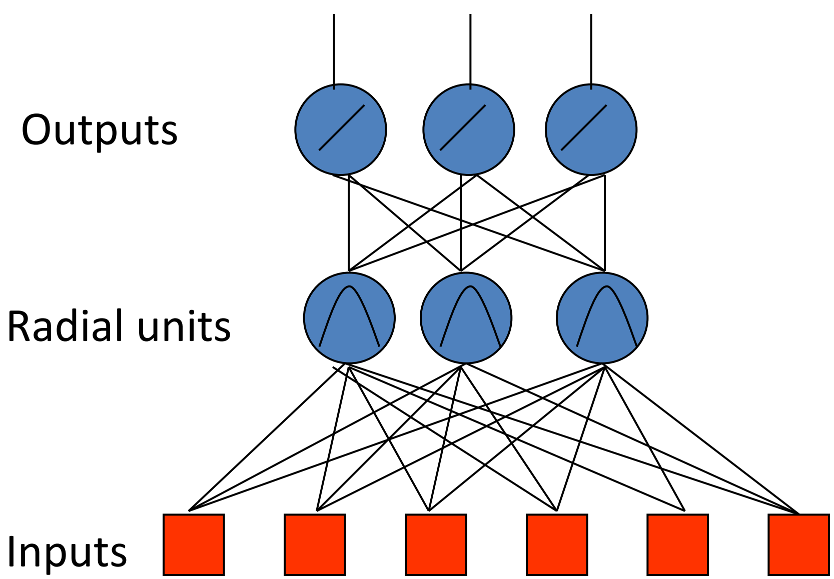
Different non linearly separable problems

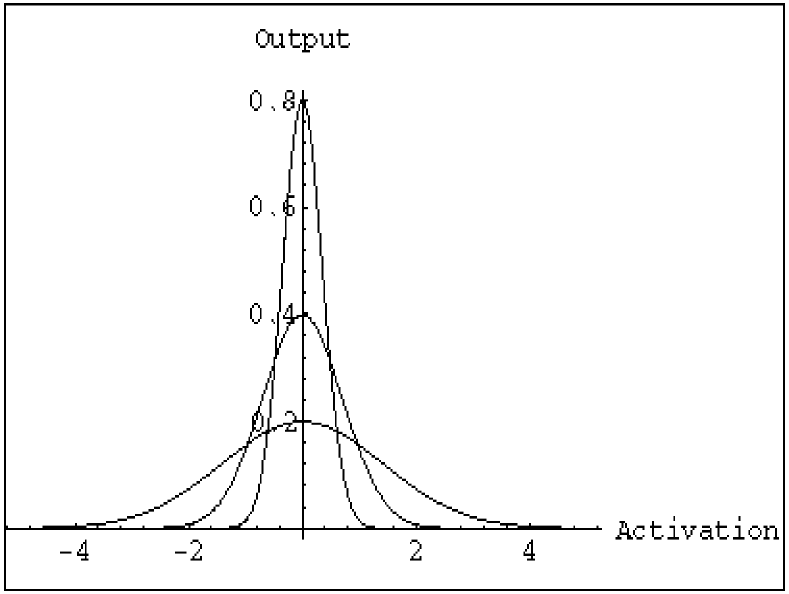
Structure	*Types of Decision Regions*	*Exclusive-OR Problem*	*Classes with Meshed regions*	*Most General Region Shapes*
Single-Layer 	*Half Plane Bounded By Hyperplane*			
Two-Layer 	*Convex Open Or Closed Regions*			
Three-Layer 	Abitrary (Complexity Limited by No. of Nodes)			

Neural Networks – An Introduction Dr. Andrew Hunter

Radial Basis Functions (RBFs)

- Features
  - One hidden layer
  - The activation of a hidden unit is determined by the distance between the input vector and a prototype vector





- RBF hidden layer units have a receptive field which has a centre
- Generally, the hidden unit function is Gaussian
- The output Layer is linear
- Realized function

$$s(x) = \sum_{j=1}^{K} W_j \Phi\left(\left\| x - c_j \right\|\right)$$

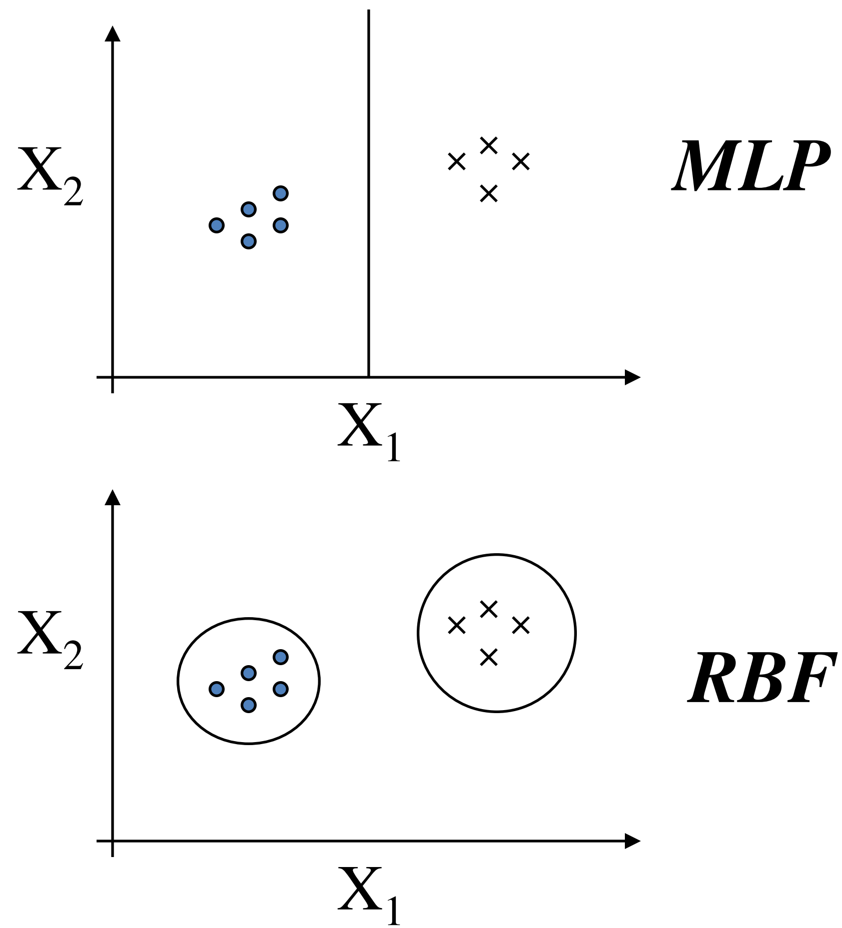
$$\Phi\left(\left\| x - c_j \right\|\right) = \exp - \left(\frac{\left\| x - c_j \right\|}{\sigma_j} \right)^2$$

Learning

- The training is performed by deciding on
  - How many hidden nodes there should be
  - The centers and the sharpness of the Gaussians
- 2 steps
  - In the 1st stage, the input data set is used to determine the parameters of the basis functions
  - In the 2nd stage, functions are kept fixed while the second layer weights are estimated (Simple BP algorithm like for MLPs)

MLPs versus RBFs

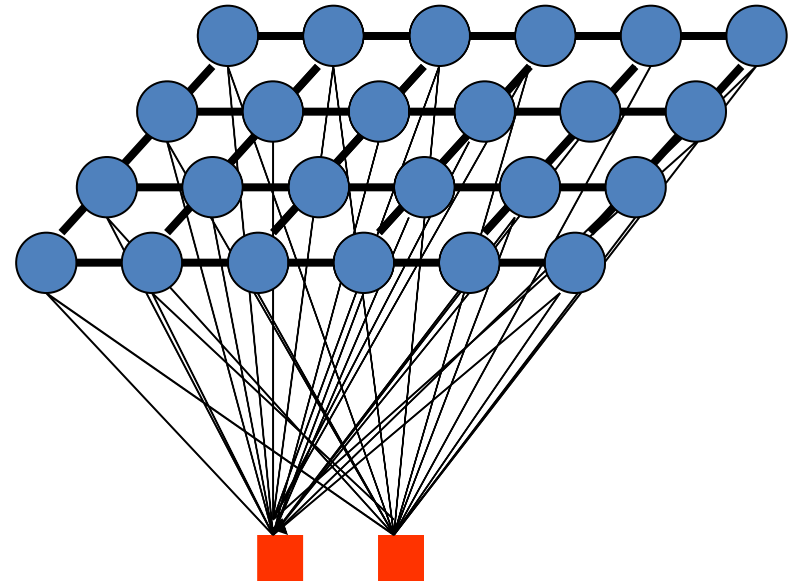
- **Classification**
  - MLPs separate classes via hyperplanes
  - RBFs separate classes via hyperspheres
- **Learning**
  - MLPs use distributed learning
  - RBFs use localized learning
  - RBFs train faster
- **Structure**
  - MLPs have one or more hidden layers
  - RBFs have only one layer
  - RBFs require more hidden neurons => curse of dimensionality

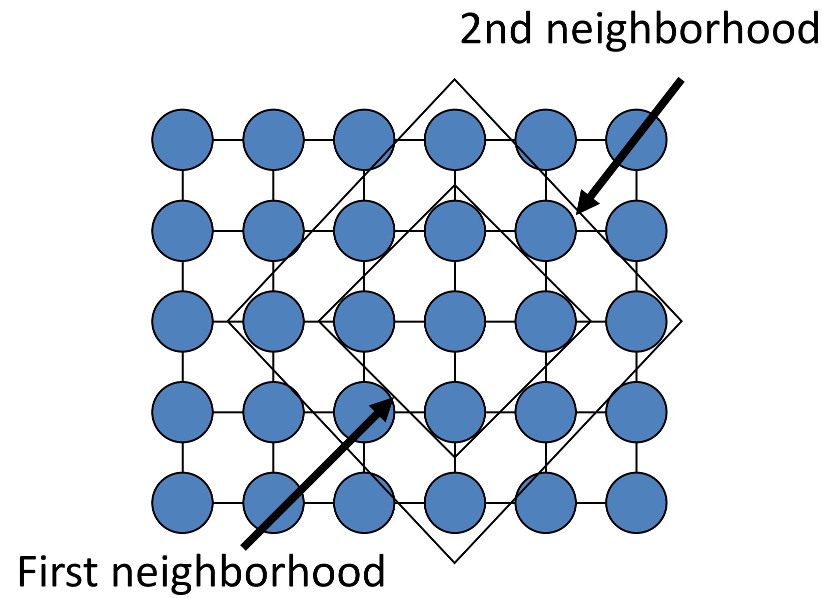


Self organizing maps

- The purpose of SOM is to map a multidimensional input space onto a topology preserving map of neurons
  - Preserve a topological so that neighboring neurons respond to « similar »input patterns
  - The topological structure is often a 2 or 3 dimensional space
- Each neuron is assigned a weight vector with the same dimensionality of the input space
- Input patterns are compared to each weight vector and the closest wins (Euclidean Distance)

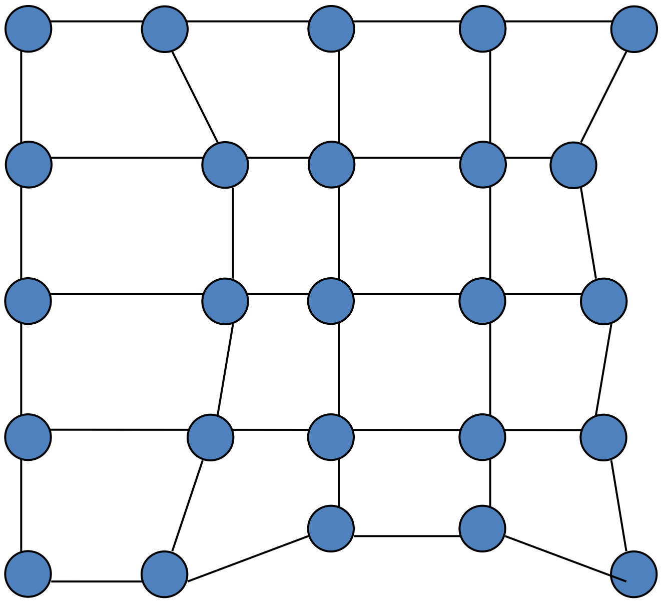
- The activation of the neuron is spread in its direct neighborhood =>neighbors become sensitive to the same input patterns
- Block distance
- The size of the neighborhood is initially large but reduce over time => Specialization of the network





Adaptation

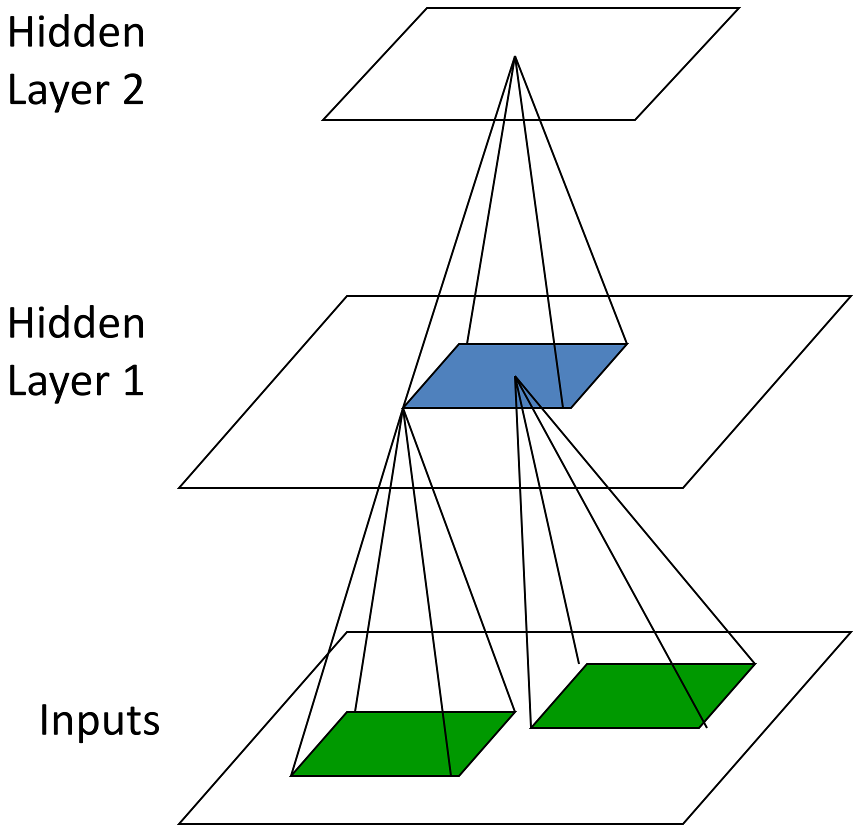
- During training, the “winner” neuron and its neighborhood adapts to make their weight vector more similar to the input pattern that caused the activation
- The neurons are moved closer to the input pattern
- The magnitude of the adaptation is controlled via a learning parameter which decays over time



Shared weights neural networks: Time Delay Neural Networks (TDNNs)

- Introduced by Waibel in 1989
- Properties
  - Local, shift invariant feature extraction
  - Notion of receptive fields combining local information into more abstract patterns at a higher level
  - Weight sharing concept (All neurons in a feature share the same weights)
    - All neurons detect the same feature but in different position
- Principal Applications
  - Speech recognition
  - Image analysis

TDNNs (cont'd)



- Objects recognition in an image
- Each hidden unit receive inputs only from a small region of the input space : receptive field
- Shared weights for all receptive fields => translation invariance in the response of the network

- Advantages
  - Reduced number of weights
    - Require fewer examples in the training set
    - Faster learning
  - Invariance under time or space translation
  - Faster execution of the net (in comparison of full connected MLP)

Neural Networks (Applications)

- Face recognition
- Time series prediction
- Process identification
- Process control
- Optical character recognition
- Adaptative filtering
- Etc…

Conclusion on Neural Networks

- Neural networks are utilized as statistical tools
  - Adjust non linear functions to fulfill a task
  - Need of multiple and representative examples but fewer than in other methods
- Neural networks enable to model complex static phenomena (FF) as well as dynamic ones (RNN)
- NN are good classifiers BUT
  - Good representations of data have to be formulated
  - Training vectors must be statistically representative of the entire input space
  - Unsupervised techniques can help
- The use of NN needs a good comprehension of the problem

International Journal of Swarm Intelligence and Evolutionary Computation

International Journal of Swarm Intelligence and Evolutionary Computation

International Journal of Swarm Intelligence and Evolutionary Computation

- A Global Colloquium on Artificial Intelligence

OMICS Group Open Access Membership

OMICS publishing Group Open Access Membership enables academic and research institutions, funders and corporations to actively encourage open access in scholarly communication and the dissemination of research published by their authors.

For more details and benefits, click on the link below:

http://omicsonline.org/membership.php