Artificial Neural Networks

for people in a hurry, in at most four hours!

Joel Suárez Cansino
jsuarez@uaeh.edu.mx

Systems and Information Technologies Research Center
Basic Sciences and Engineering Institute
Autonomous University of the State of Hidalgo

November 6, 2010

Topics

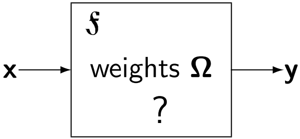
1. Artificial Neural Networks
   - Architectures
   - Learning paradigms
2. Feedforward Artificial Neural Networks
   - Multilayer perceptron
   - Radial Basis Function
3. Associative Artificial Neural Networks
   - Hebb's rule
   - Characteristics of the conditioning process
4. Neurofuzzy Systems
   - Basic extensions of crisp neurons
5. Support Vector Machines
6. Spiking Neurons

Topics

Basic definitions

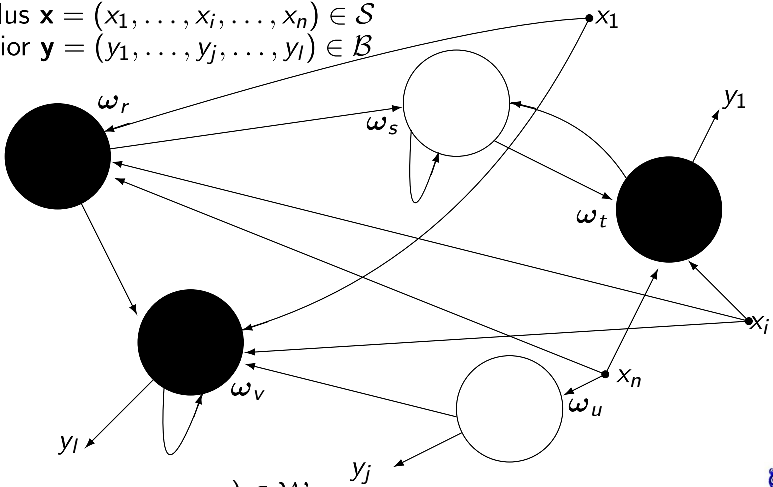
Definition

An artificial neural network is a function $\mathfrak{f}\colon \mathcal{S} \times \mathcal{W} \to \mathcal{B}$, where $\mathcal{S}$ represents the stimuli space, $\mathcal{B}$ the behaviors space and $\mathbf{y} = \mathfrak{f}(\mathbf{x}; \boldsymbol{\Omega})$ is the rule of correspondence, with $\mathbf{x} \in \mathcal{S}$, $\mathbf{y} \in \mathcal{B}$ and $\boldsymbol{\Omega} \in \mathcal{W}$ the weight parameters to be determined through a proper training or learning process inside the weight space $\mathcal{W}$.



General structure

Stimulus $\mathbf{x} = (x_1, \ldots, x_i, \ldots, x_n) \in \mathcal{S}$
Behavior $\mathbf{y} = (y_1, \ldots, y_j, \ldots, y_l) \in \mathcal{B}$



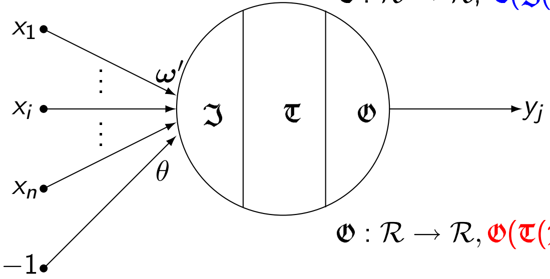
$\mathbf{\Omega} = (\ldots, \omega_r, \ldots, \omega_v, \ldots) \in \mathcal{W}$

General neuron structure

Stimulus $\mathbf{x} = (x_1, \ldots, x_i, \ldots, x_m) \in \mathcal{S}'$

$\mathfrak{I} : \mathcal{S}' \times \mathcal{W}' \to \mathcal{R}$, $\mathfrak{I}(\mathbf{x}; \omega', \theta) \in \mathcal{R}$

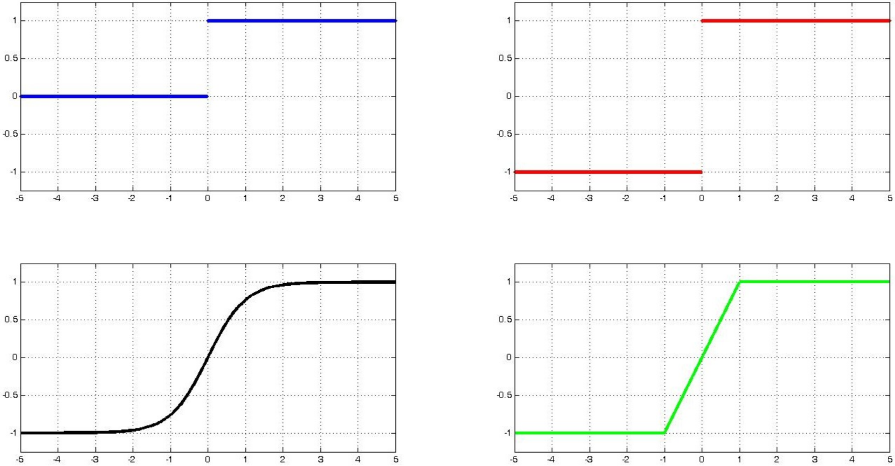
$\mathfrak{T} : \mathcal{R} \to \mathcal{R}$, $\mathfrak{T}(\mathfrak{I}(\mathbf{x}; \omega', \theta)) \in \mathcal{R}$



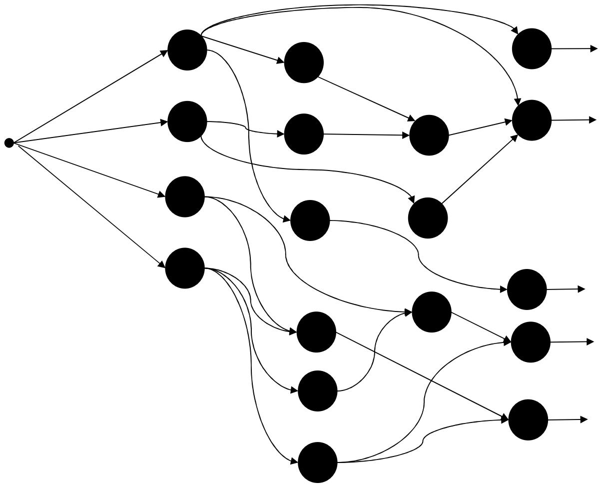
$\mathfrak{O} : \mathcal{R} \to \mathcal{R}, \mathfrak{O}(\mathfrak{T}(\mathfrak{I}(\mathbf{x}; \omega', \theta))) \in \mathcal{R}$

Transfer or activation functions

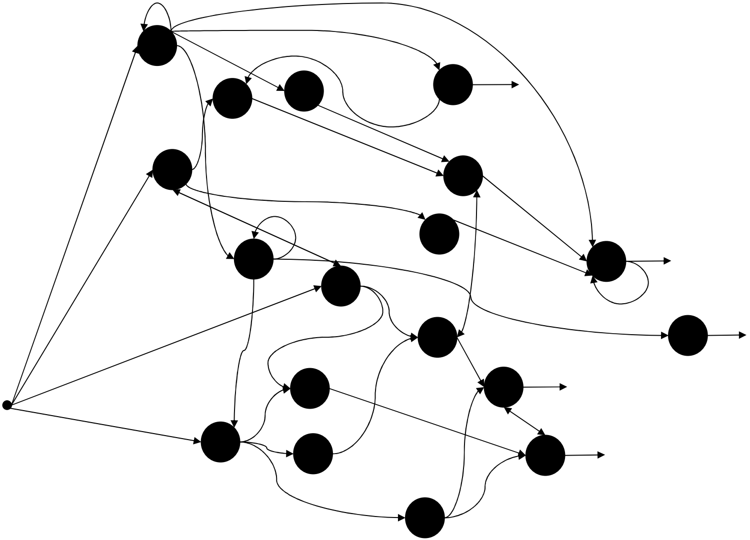
(Chapter 2 and 11 (Network function) MatLab nnd command)



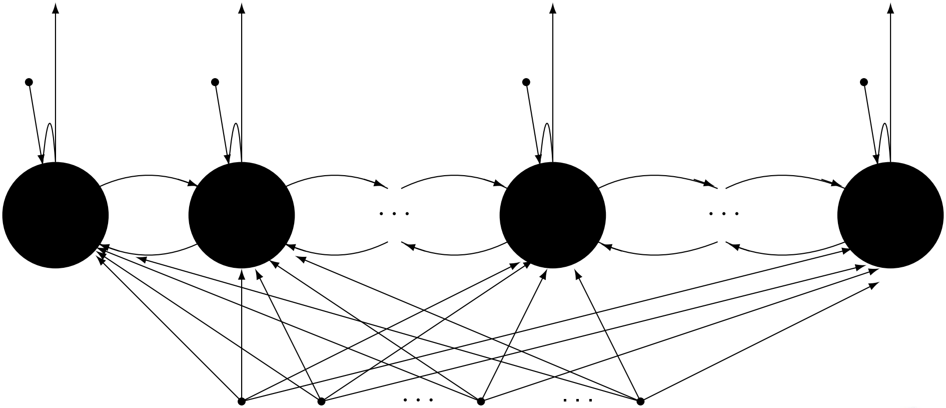
Feedforward artificial neural networks



Feedback artificial neural networks



Competitive artificial neural networks



Supervised learning

Main characteristics

1. The easiest way of learning.

Supervised learning

Main characteristics

1. The easiest way of learning.
2. The training process requires a supervisor or an instructor (not a critic!).

Supervised learning

Main characteristics

1. The easiest way of learning.
2. The training process requires a supervisor or an instructor (not a critic!).
3. Stimuli and their correspondent behaviors need to be previously specified.

Supervised learning

Main characteristics

1. The easiest way of learning.
2. The training process requires a supervisor or an instructor (not a critic!).
3. Stimuli and their correspondent behaviors need to be previously specified.
4. Square error provides theoretical basis for learning procedures.

Unsupervised learning

Main characteristics

1. The most difficult way of learning.

Unsupervised learning

Main characteristics

1. The most difficult way of learning.
2. The training process does not require supervisor or instructor, neither critic.

Unsupervised learning

Main characteristics

1. The most difficult way of learning.
2. The training process does not require supervisor or instructor, neither critic.
3. Stimuli are the only data to be previously specified.

Unsupervised learning

Main characteristics

1. The most difficult way of learning.
2. The training process does not require supervisor or instructor, neither critic.
3. Stimuli are the only data to be previously specified.
4. Looking for correlation between weights and stimuli.

Unsupervised learning

Main characteristics

1. The most difficult way of learning.
2. The training process does not require supervisor or instructor, neither critic.
3. Stimuli are the only data to be previously specified.
4. Looking for correlation between weights and stimuli.
5. Hebb's rule provides theoretical basis for learning procedures.

Topics

Simple perceptron

Main properties (Chapters 3 and 4 MatLab nnd command)

1. The architecture consists of only one layer with a previously specified number of neurons.

Simple perceptron

Main properties (Chapters 3 and 4 MatLab nnd command)

1. The architecture consists of only one layer with a previously specified number of neurons.
2. Only linearly separable problems can be solved.

Simple perceptron

Main properties (Chapters 3 and 4 MatLab nnd command)

1. The architecture consists of only one layer with a previously specified number of neurons.
2. Only linearly separable problems can be solved.
3. The training procedure is based on the application of the recurrence equation (supervised learning)

$$\boldsymbol{\omega}_{new} = \boldsymbol{\omega}_{old} + \eta[\mathbf{d}^T - \mathrm{hardlim}(\boldsymbol{\omega}_{old}\mathbf{x}^T)]\mathbf{x}$$

Simple perceptron

Main properties (Chapters 3 and 4 MatLab nnd command)

1. The architecture consists of only one layer with a previously specified number of neurons.
2. Only linearly separable problems can be solved.
3. The training procedure is based on the application of the recurrence equation (supervised learning)

$$\boldsymbol{\omega}_{new} = \boldsymbol{\omega}_{old} + \eta[\mathbf{d}^T - \mathrm{hardlim}(\boldsymbol{\omega}_{old}\mathbf{x}^T)]\mathbf{x}$$

4. Classification problems with two or more classes can be solved without difficulty only if they are linearly separable.

Simple perceptron

Main properties (Chapters 3 and 4 MatLab nnd command)

1. The architecture consists of only one layer with a previously specified number of neurons.
2. Only linearly separable problems can be solved.
3. The training procedure is based on the application of the recurrence equation (supervised learning)

$$\boldsymbol{\omega}_{new} = \boldsymbol{\omega}_{old} + \eta[\mathbf{d}^T - \text{hardlim}(\boldsymbol{\omega}_{old}\mathbf{x}^T)]\mathbf{x}$$

4. Classification problems with two or more classes can be solved without difficulty only if they are linearly separable.
5. Impossible to solve the XOR problem (non linearly separable problem), without introducing an additional layer of neurons.

Multilayer perceptron

Main properties (Chapters 9 and 11 (Backpropagation, Function approximation and Generalization))

1. The architecture consists of one or more layers with a previously specified number of neurons.

Multilayer perceptron

Main properties (Chapters 9 and 11 (Backpropagation, Function approximation and Generalization))

1. The architecture consists of one or more layers with a previously specified number of neurons.
2. Non linearly separable problems can also be solved.

Multilayer perceptron

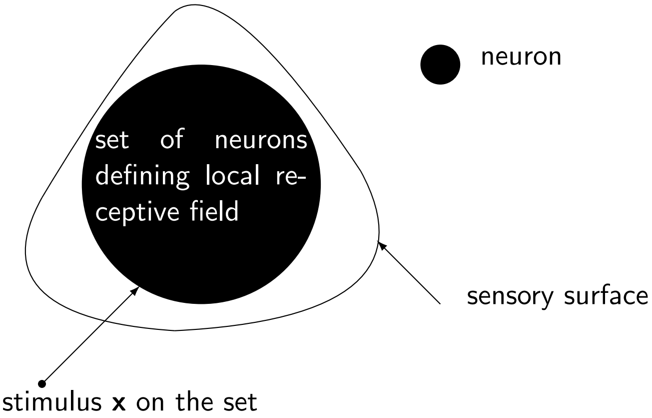
Main properties (Chapters 9 and 11 (Backpropagation, Function approximation and Generalization))

1. The architecture consists of one or more layers with a previously specified number of neurons.
2. Non linearly separable problems can also be solved.
3. The training procedure is based on the application of the recurrence equation (supervised learning)

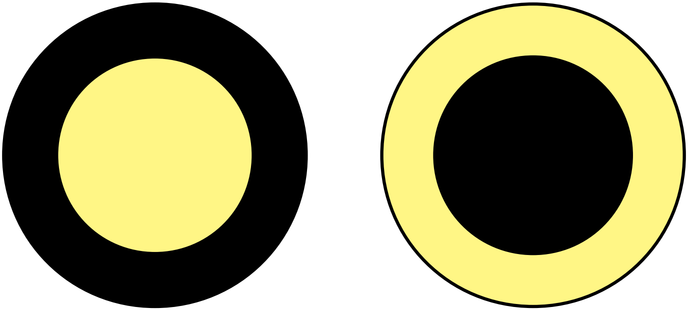
$$\omega_{new} = \omega_{old} - \eta \frac{\partial \mathfrak{E}^2}{\partial \omega},$$

where $\mathfrak{E}^2$ is the square total error as a function of the weights ω and $\frac{\partial}{\partial \omega}$ is the gradient operator.

Local receptive field



Types of local receptive fields



on center - off surround

off center - on surround

Mathematical model

Fourier's transform

$f : \mathcal{R} \rightarrow \mathcal{R}$

$$f(x) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} g(y) e^{ixy} dy,$$

where g is the inverse Fourier's transform

Inverse Fourier's transform

$$g(y) = \int_{-\infty}^{+\infty} f(z) e^{-izy} dz$$

Dirac's delta

Dirac's delta definition

$$f(x) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} f(z) \int_{-\infty}^{+\infty} e^{i(x-z)y} dydz = \int_{-\infty}^{+\infty} f(z)\delta(x-z)dz$$

$$\delta(x-z) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} e^{i(x-z)y} dy$$

Dirac's delta properties

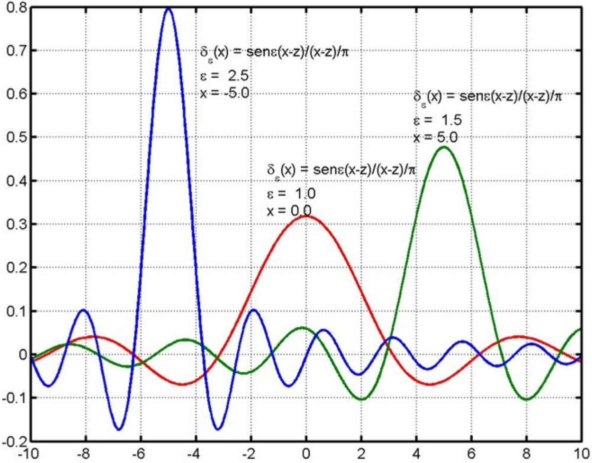
1. $$\int_{-\infty}^{\infty} \delta(x-z)dz = 1$$

2. $$\delta(x-z) = \lim_{\epsilon \to +\infty} \frac{\sin \epsilon(x-z)}{\pi(x-z)}$$

3. Dirac's delta is a sequence of functions
$$\delta_{\epsilon}(x-z) = \frac{\sin \epsilon(x-z)}{\pi(x-z)}, 0 < \epsilon$$

4. Dirac's delta es an even function
$$\delta(x-z) = \delta(z-x)$$

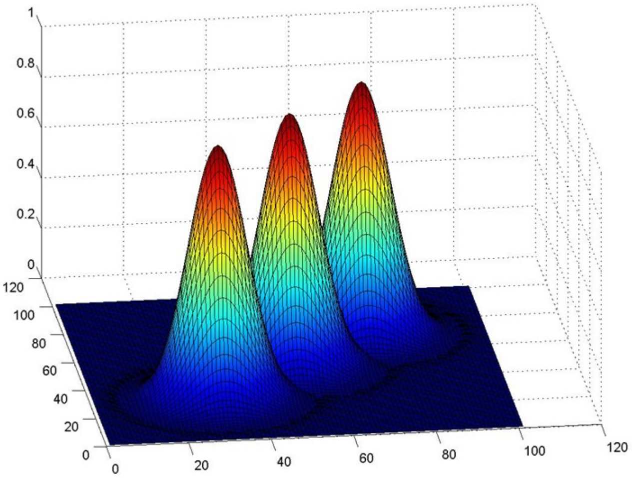
Some sequence examples



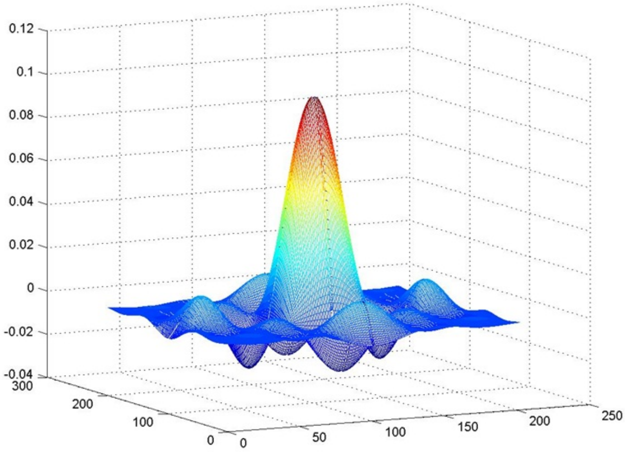
Some approximations to Dirac's delta sequence

1. For z's very close to x, $\delta_\epsilon(x - z) \approx \frac{\epsilon}{\pi}, 0 < \epsilon$
2. $e^{-\lambda(x-z)^2}, 0 < \lambda$
3. $\frac{1}{(x-z)^2+c^2}$
4. Two dimensional Dirac's delta approximation $e^{-\lambda_1(x-z_1)^2-\lambda_2(y-z_2)^2}, 0 < \lambda_1, \lambda_2$
5. Another two dimensional Dirac's approximation $\frac{\sin \epsilon_1(x-z_1) \sin \epsilon_2(y-z_2)}{\pi^2(x-z_1)(y-z_2)}$

Graphics of some two dimensional sequences



Graphics of some two dimensional sequences



Summary (first steps to build rbf artificial neural network)

1. Define a mother function to generate the whole class of generating functions (the basis of the space of functions).

Summary (first steps to build rbf artificial neural network)

1. Define a mother function to generate the whole class of generating functions (the basis of the space of functions).
2. For every class in the problem, take a proper function from the basis.

Summary (first steps to build rbf artificial neural network)

1. Define a mother function to generate the whole class of generating functions (the basis of the space of functions).
2. For every class in the problem, take a proper function from the basis.
3. Assign the selected function to one neuron in the neural network.

Summary (first steps to build rbf artificial neural network)

1. Define a mother function to generate the whole class of generating functions (the basis of the space of functions).
2. For every class in the problem, take a proper function from the basis.
3. Assign the selected function to one neuron in the neural network.
4. The selected set of neurons defines just one layer of neurons.

Summary (first steps to build rbf artificial neural network)

1. Define a mother function to generate the whole class of generating functions (the basis of the space of functions).
2. For every class in the problem, take a proper function from the basis.
3. Assign the selected function to one neuron in the neural network.
4. The selected set of neurons defines just one layer of neurons.
5. The linear combination of the elements of the set provides an approximation to the desired result.

Summary (first steps to build rbf artificial neural network)

1. Define a mother function to generate the whole class of generating functions (the basis of the space of functions).
2. For every class in the problem, take a proper function from the basis.
3. Assign the selected function to one neuron in the neural network.
4. The selected set of neurons defines just one layer of neurons.
5. The linear combination of the elements of the set provides an approximation to the desired result.
6. An output layer with a single neuron defines the approximated output.

Summary (first steps to build rbf artificial neural network)

1. Define a mother function to generate the whole class of generating functions (the basis of the space of functions).
2. For every class in the problem, take a proper function from the basis.
3. Assign the selected function to one neuron in the neural network.
4. The selected set of neurons defines just one layer of neurons.
5. The linear combination of the elements of the set provides an approximation to the desired result.
6. An output layer with a single neuron defines the approximated output.
7. It could be there some intersections of local receptive fields of a subset of some basis functions.

Summary (first steps to build rbf artificial neural network)

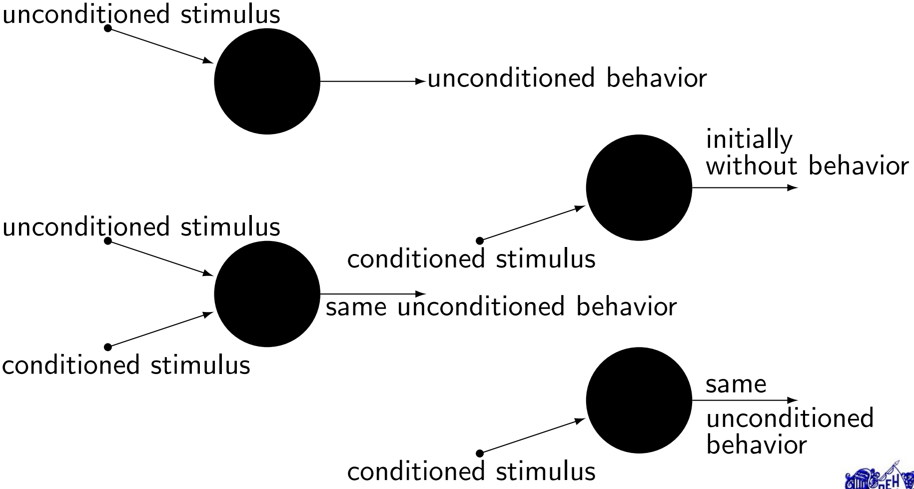
1. Define a mother function to generate the whole class of generating functions (the basis of the space of functions).
2. For every class in the problem, take a proper function from the basis.
3. Assign the selected function to one neuron in the neural network.
4. The selected set of neurons defines just one layer of neurons.
5. The linear combination of the elements of the set provides an approximation to the desired result.
6. An output layer with a single neuron defines the approximated output.
7. It could be there some intersections of local receptive fields of a subset of some basis functions.
8. First layer with parameters of basis functions as weights to be determined through training process (supervised learning).

Summary (first steps to build rbf artificial neural network)

1. Define a mother function to generate the whole class of generating functions (the basis of the space of functions).
2. For every class in the problem, take a proper function from the basis.
3. Assign the selected function to one neuron in the neural network.
4. The selected set of neurons defines just one layer of neurons.
5. The linear combination of the elements of the set provides an approximation to the desired result.
6. An output layer with a single neuron defines the approximated output.
7. It could be there some intersections of local receptive fields of a subset of some basis functions.
8. First layer with parameters of basis functions as weights to be determined through training process (supervised learning).
9. Second layer with single neuron with weights to be determined through training process (supervised learning).

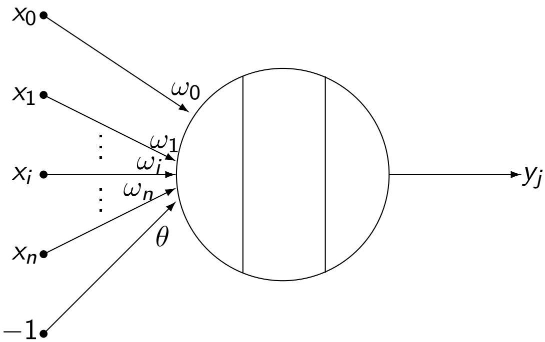
Topics

Pavlov's conditioning



Conditioning neuron structure

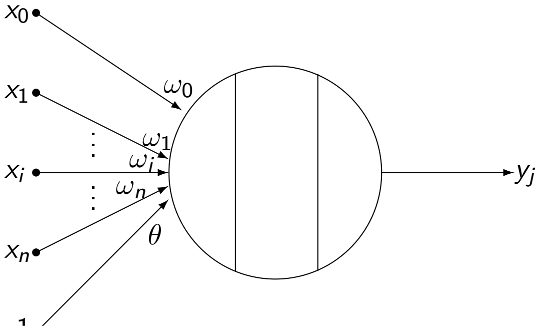
Stimulus $\mathbf{x} = (x_0, x_1, \ldots, x_i, \ldots, x_n, -1)$
Weights $\boldsymbol{\omega} = (\omega_0, \omega_1, \ldots, \omega_i, \ldots, \omega_n, \theta)$



Conditioning neuron structure

Stimulus $\mathbf{x} = (x_0, x_1, \ldots, x_i, \ldots, x_n, -1)$

Weights $\boldsymbol{\omega} = (\omega_0, \omega_1, \ldots, \omega_i, \ldots, \omega_n, \theta)$



Observation!

Since the unconditioned stimulus is previously learnt, the weights ω_0 and θ are not modified along the learning process. The conditioned stimulus needs to be learnt, so the corresponding weights need to be modified.

Summary

Main properties (Chapter 14 (Competitive learning))

1. The only data previously known are the stimuli.

Summary

Main properties (Chapter 14 (Competitive learning))

1. The only data previously known are the stimuli.
2. There are no critic, instructor or supervisor.

Summary

Main properties (Chapter 14 (Competitive learning))

1. The only data previously known are the stimuli.
2. There are no critic, instructor or supervisor.
3. The learning process is completely unsupervised.

Summary

Main properties (Chapter 14 (Competitive learning))

1. The only data previously known are the stimuli.
2. There are no critic, instructor or supervisor.
3. The learning process is completely unsupervised.
4. The Hebb's learning rule provides the mathematical basis for modeling the association.

Summary

Main properties (Chapter 14 (Competitive learning))

1. The only data previously known are the stimuli.
2. There are no critic, instructor or supervisor.
3. The learning process is completely unsupervised.
4. The Hebb's learning rule provides the mathematical basis for modeling the association.
5. The association process looks for correlation between weights and stimulus.

Summary

Main properties (Chapter 14 (Competitive learning))

1. The only data previously known are the stimuli.
2. There are no critic, instructor or supervisor.
3. The learning process is completely unsupervised.
4. The Hebb's learning rule provides the mathematical basis for modeling the association.
5. The association process looks for correlation between weights and stimulus.
6. If forgetting is considered, the learning rule is

$$\boldsymbol{\omega}_{new} = \boldsymbol{\omega}_{old} + \eta y(\mathbf{x} - \boldsymbol{\omega}_{old})$$

Summary

Main properties (Chapter 14 (Competitive learning))

1. The only data previously known are the stimuli.
2. There are no critic, instructor or supervisor.
3. The learning process is completely unsupervised.
4. The Hebb's learning rule provides the mathematical basis for modeling the association.
5. The association process looks for correlation between weights and stimulus.
6. If forgetting is considered, the learning rule is

$$\boldsymbol{\omega}_{new} = \boldsymbol{\omega}_{old} + \eta y(\mathbf{x} - \boldsymbol{\omega}_{old})$$

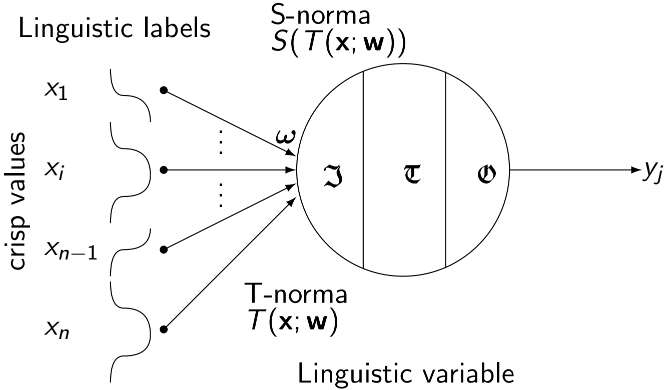
7. The weights are bounded, as well.

Topics

Focus of the main thrust of fuzzy neural nets

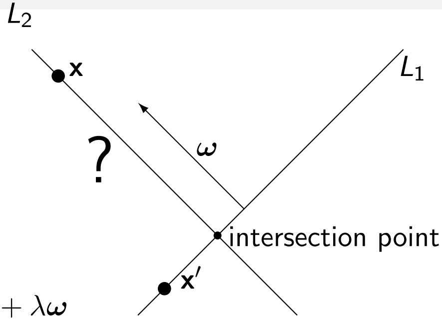
1. The fuzzification of the dendritic inputs.
2. The aggregation operation of a conventional neuron.

Basic structure of a fuzzy neuron



Topics

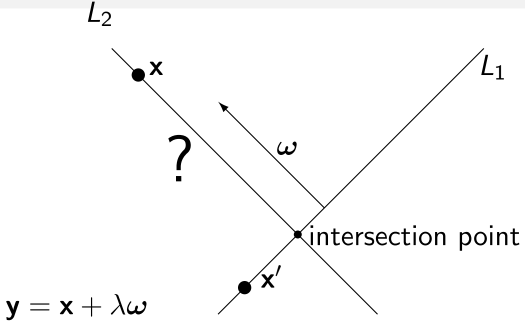
Hard margin support vector machine



$$\mathbf{y} = \mathbf{x} + \lambda\omega$$

$$\mathbf{y} = \mathbf{x}' + \lambda'\omega^{\perp}$$

Hard margin support vector machine



Line equation for L_2 and L_2

line equation L_2

$$\mathbf{y} = \mathbf{x} + \lambda\boldsymbol{\omega}, \lambda \in \mathcal{R}$$

line equation L_1

$$\mathbf{y} = \mathbf{x}' + \lambda'\boldsymbol{\omega}^{\perp}, \lambda' \in \mathcal{R}$$

Shortest distance from $\mathbf{x}$ to line L_1

In the intersection point there do exist scalars λ and λ' such that the following holds

$$\mathbf{x} + \lambda\boldsymbol{\omega} = \mathbf{x}' + \lambda'\boldsymbol{\omega}^{\perp}$$

After multiplying with dot product both sides of the equation by $\boldsymbol{\omega}^{\perp}$

$$\boldsymbol{\omega}^{\perp} \cdot \mathbf{x} = \boldsymbol{\omega}^{\perp} \cdot \mathbf{x}' + \lambda'||\boldsymbol{\omega}||^2$$

So that, the scalar λ' in the intersection point is given by the equation

$$\lambda' = \frac{\boldsymbol{\omega}^{\perp} \cdot (\mathbf{x} - \mathbf{x}')}{||\boldsymbol{\omega}||^2}$$

Therefore, the shortest distance from $\mathbf{x}$ to line L_1 is given by the expression

$$\frac{|\boldsymbol{\omega}^{\perp} \cdot (\mathbf{x} - \mathbf{x}')|}{||\boldsymbol{\omega}||}$$

Shortest distance

This is so, since

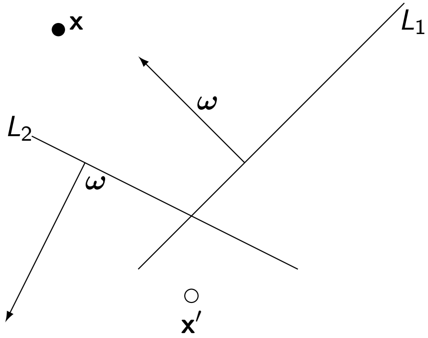
$$\frac{|\omega^{\perp} \cdot (\mathbf{x} - \mathbf{x}')|}{||\omega||} = \frac{\omega^{\perp} \cdot \mathbf{x} - \omega^{\perp} \cdot \mathbf{x}'}{||\omega||}$$

And, after adding a zero value

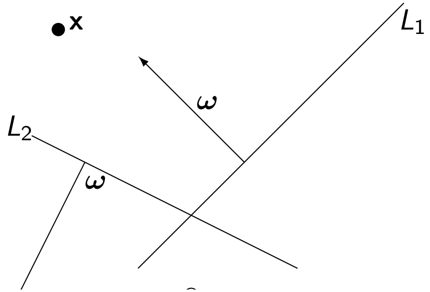
$$\frac{\omega^{\perp} \cdot \mathbf{x} + b - \omega^{\perp} \cdot \mathbf{x}' - b}{||\omega||} = \frac{\omega^{\perp} \cdot \mathbf{x} + b}{||\omega||} = \frac{c}{||\omega||}$$

($\omega^{\perp} \cdot \mathbf{x}' - b = 0$ is the equation of L_1 and $\omega^{\perp} \cdot \mathbf{x} + b = c$ is the equation of the line passing through $\mathbf{x}$ and parallel to L_1)

Two classes case



Two classes case



Important question

What's the minimum weight vector ω maximizing the distance

$$\frac{c}{||\omega||}?$$

Hard margin support vector machine

Problem

Minimize

$$Q(\omega) = \frac{1}{2}||\omega||^2$$

Subject to the constraints

$$y_i(\boldsymbol{\omega}^{\perp}\mathbf{x}_i + b) \geq 1, \forall i = 1, 2, \ldots, M$$

Data x_i's that satisfy the equalities are called support vectors (deleting all the data that satisfy the strict inequalities do not affect the resulting hyperplane)

Topics