Introduction to Machine Learning

Perceptron

Barnabás Póczos

Contents

- History of Artificial Neural Networks
- Definitions: Perceptron, Multi-Layer Perceptron
- Perceptron algorithm

Short History of Artificial Neural Networks

Short History

- **Progression (1943-1960)**
  - First mathematical model of neurons
    - Pitts & McCulloch (1943)
  - Beginning of artificial neural networks
  - Perceptron, Rosenblatt (1958)
    - A single neuron for classification
    - Perceptron learning rule
    - Perceptron convergence theorem

- **Degression (1960-1980)**
  - Perceptron can't even learn the XOR function
  - We don't know how to train MLP
  - 1963 Backpropagation... but not much attention...

    Bryson, A.E.; W.F. Denham; S.E. Dreyfus. Optimal programming problems with inequality constraints. I: Necessary conditions for extremal solutions. AIAA J. 1, 11 (1963) 2544-2550

Short History

- **Progression (1980-)**
  - 1986 Backpropagation reinvented:
    - Rumelhart, Hinton, Williams:
      Learning representations by back-propagating errors.
      Nature, 323, 533—536, 1986
  - Successful applications:
    - Character recognition, autonomous cars,…
  - **Open questions**: Overfitting? Network structure? Neuron number? Layer number? Bad local minimum points? When to stop training?
  - Hopfield nets (1982), Boltzmann machines,…

Short History

- **Degression (1993-)**

  - SVM: Vapnik and his co-workers developed the Support Vector Machine (1993). It is a shallow architecture.
  - SVM and Graphical models almost kill the ANN research.
  - Training deeper networks consistently yields poor results.
  - Exception: deep convolutional neural networks, Yann LeCun 1998. (discriminative model)

Short History

Progression (2006-)

Deep Belief Networks (DBN)

- Hinton, G. E, Osindero, S., and Teh, Y. W. (2006). A fast learning algorithm for deep belief nets. Neural Computation, 18:1527-1554.
- Generative graphical model
- Based on restrictive Boltzmann machines
- Can be trained efficiently

Deep Autoencoder based networks

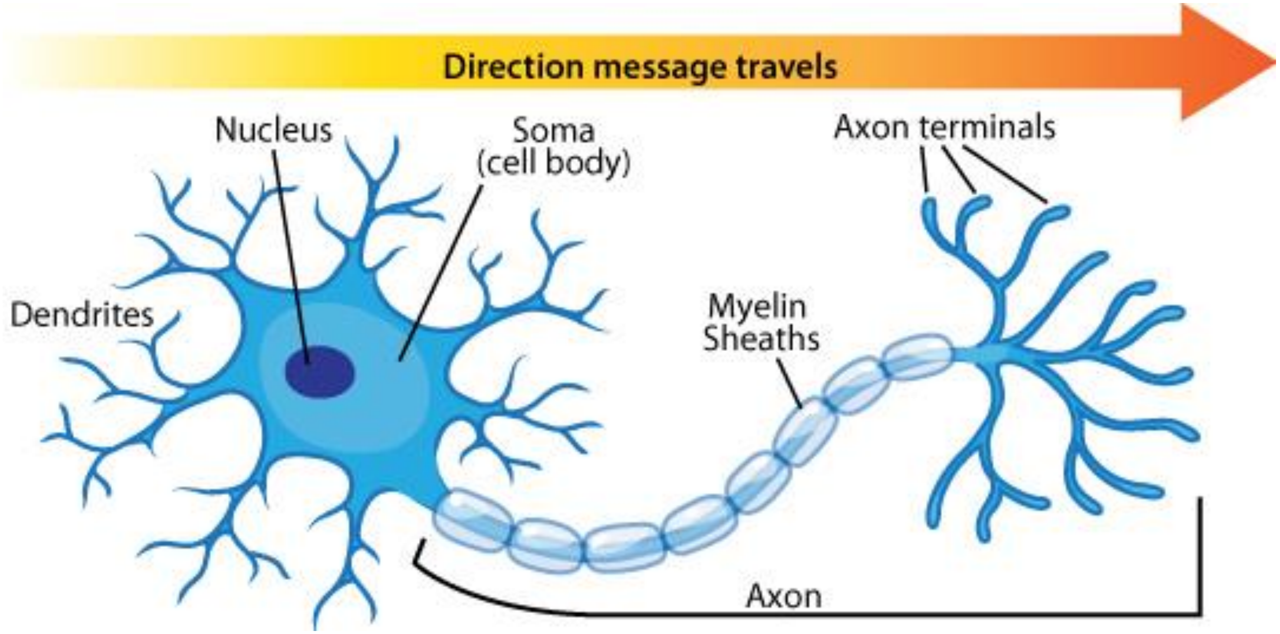
Bengio, Y., Lamblin, P., Popovici, P., Larochelle, H. (2007).
Greedy Layer-Wise Training of Deep Networks,
Advances in Neural Information Processing Systems 19

Convolutional neural networks running on GPUs

Alex Krizhevsky, Ilya Sutskever, Geoffrey Hinton, Advances in Neural Information Processing Systems 2012

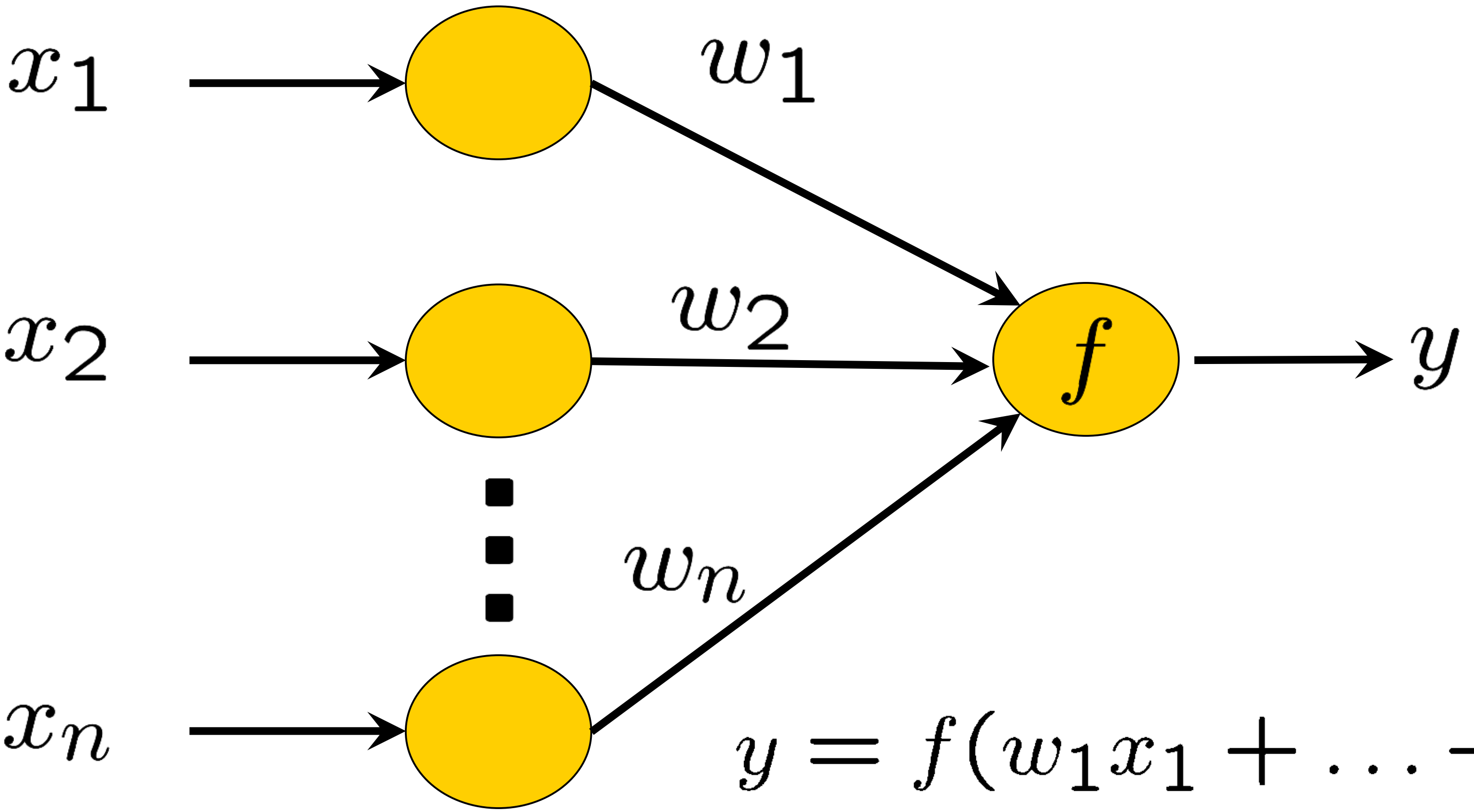
The Neuron

The Neuron



- Each neuron has a body, axon, and many dendrites
- A neuron can fire or rest
- If the sum of weighted inputs larger than a threshold, then the neuron fires.
- Synapses: The gap between the axon and other neuron’s dendrites. It determines the weights in the sum.

The Mathematical Model of a Neuron

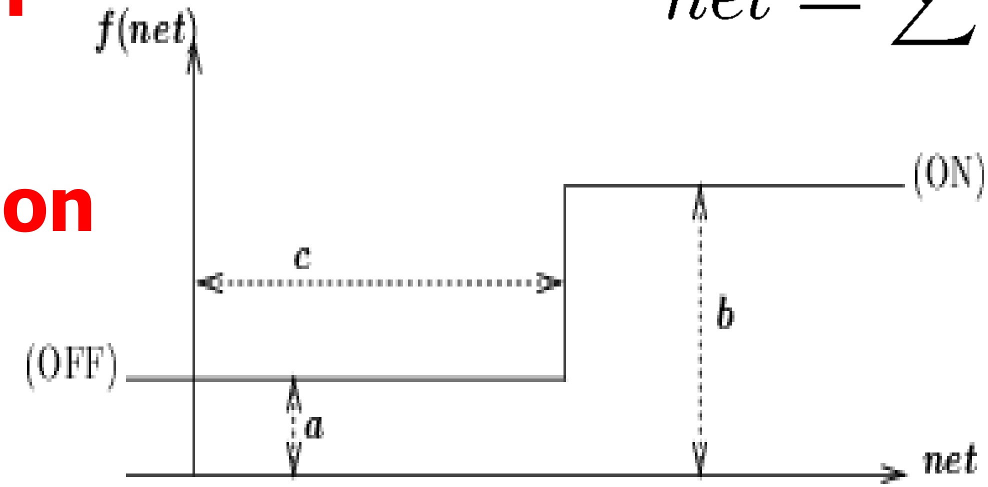


$$y = f(w_1 x_1 + \ldots + w_n x_n)$$

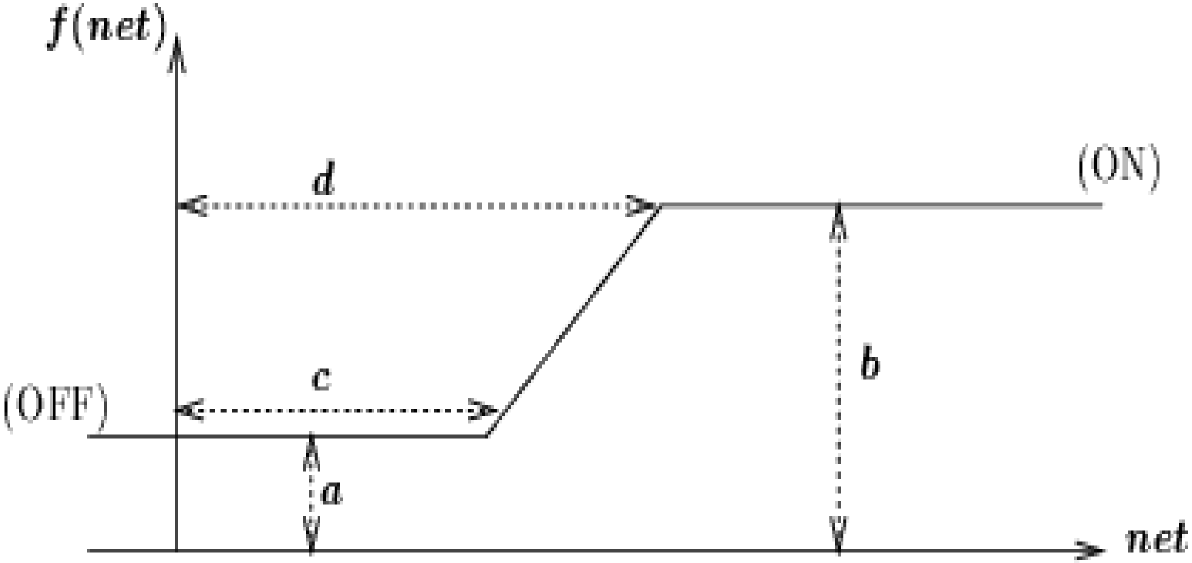
Typical activation functions

$$net = \sum w_i x_i$$

- **Identity function**
- **Threshold function** (perceptron)



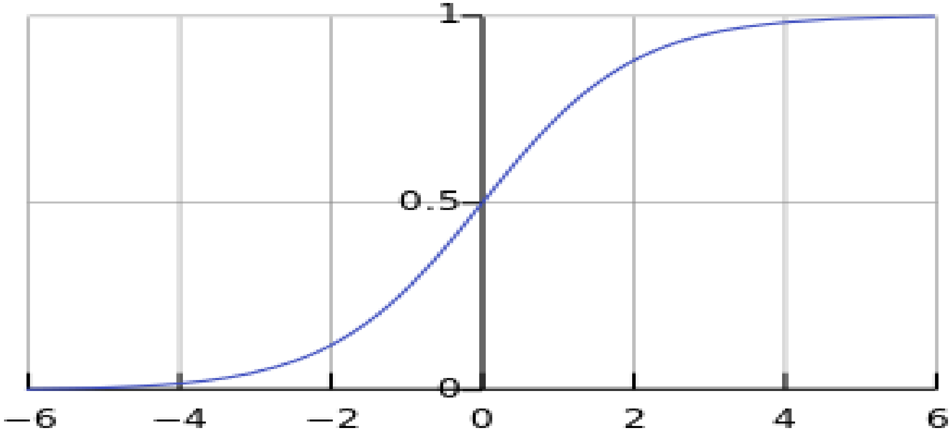
- **Ramp function**



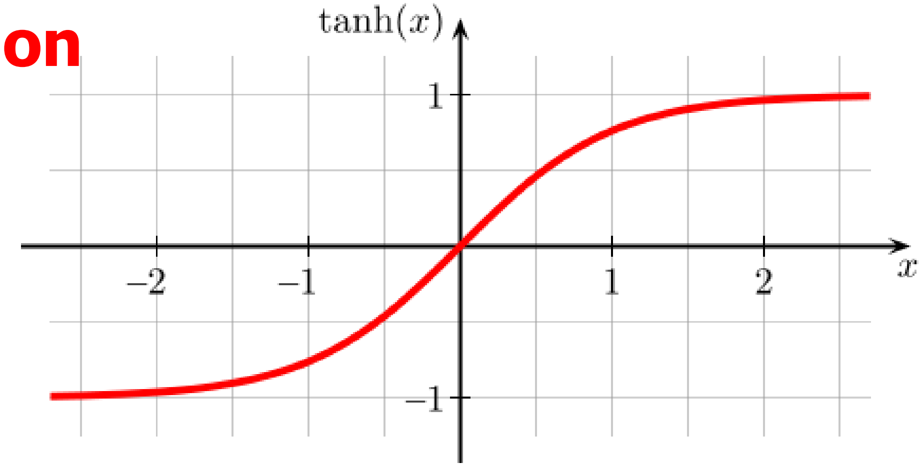
Typical activation functions

- **Logistic function**

$$f(x) = (1 + e^{-x})^{-1}$$



- **Hyperbolic tangent function**



$$f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} = \frac{e^{2x} - 1}{e^{2x} + 1} = \frac{1 - e^{-2x}}{1 + e^{-2x}}$$

Typical activation functions

- **Rectified Linear Unit (ReLU)**

$$f(x) = x^+ = \max(0, x)$$

- **Softplus function**
(This is a smooth approximation of ReLU)

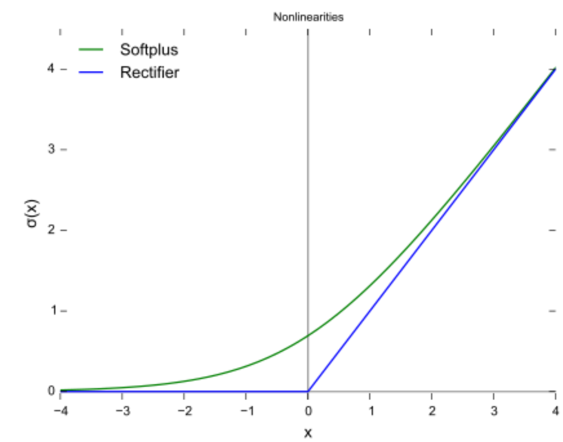
$$f(x) = \ln[1 + \exp(x)]$$

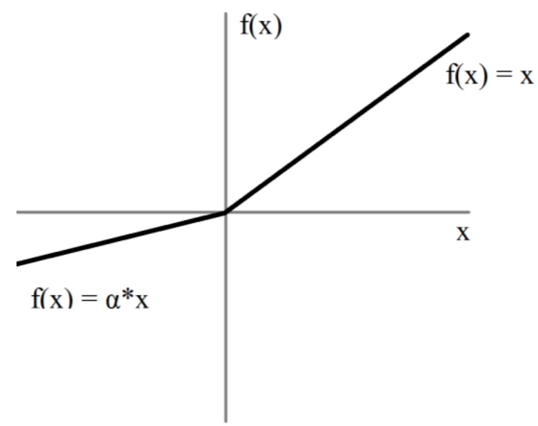
- **Leaky ReLU**

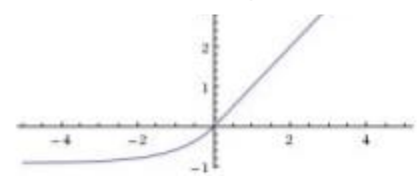
$$f(x) = \begin{cases} x & \text{if } x > 0 \\ ax & \text{otherwise} \end{cases}$$

- **Exponential Linear Unit**

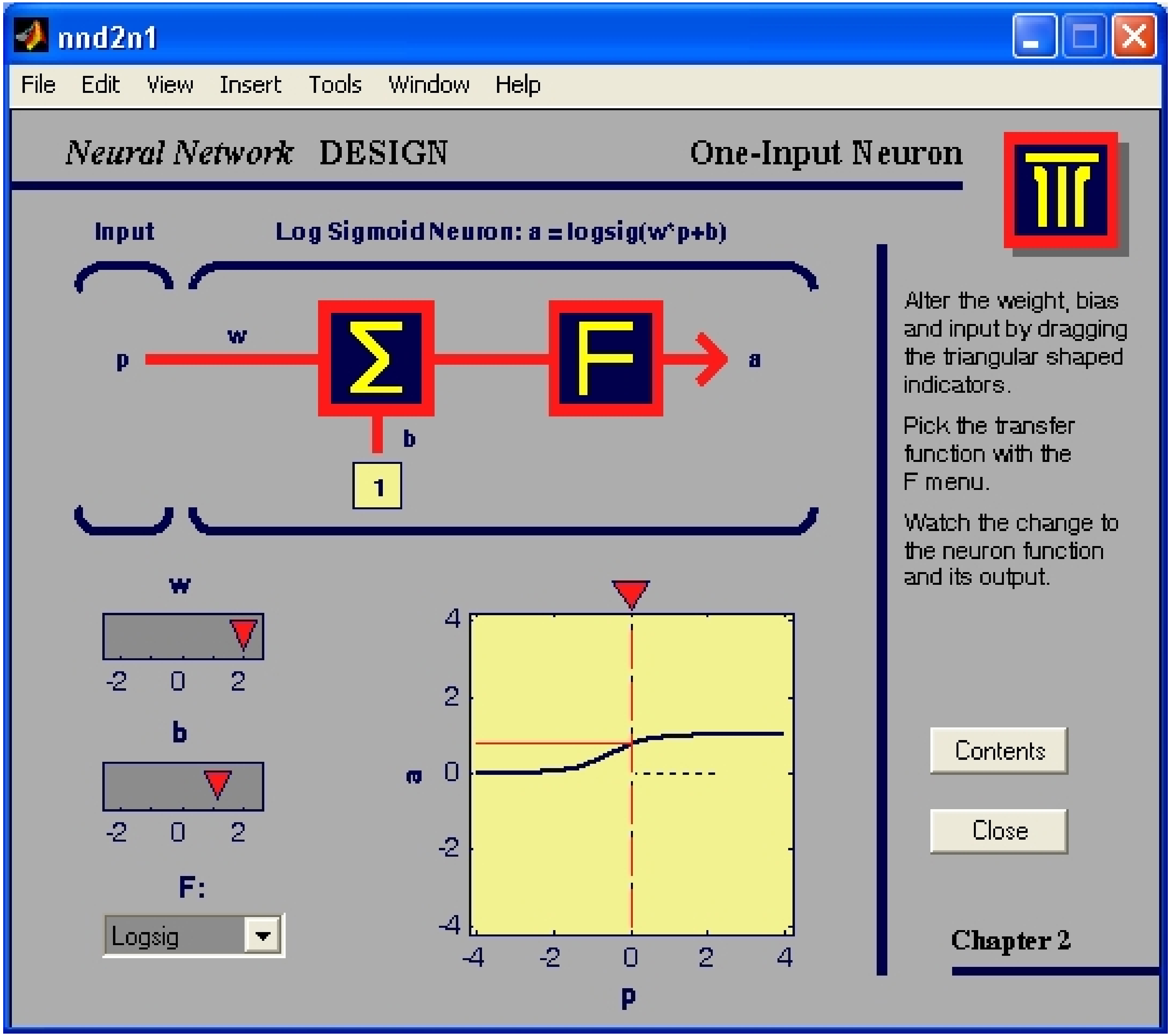
$$f(x) = \begin{cases} x & \text{if } x >= 0 \\ a[\exp(x) - 1] & \text{otherwise} \end{cases}$$



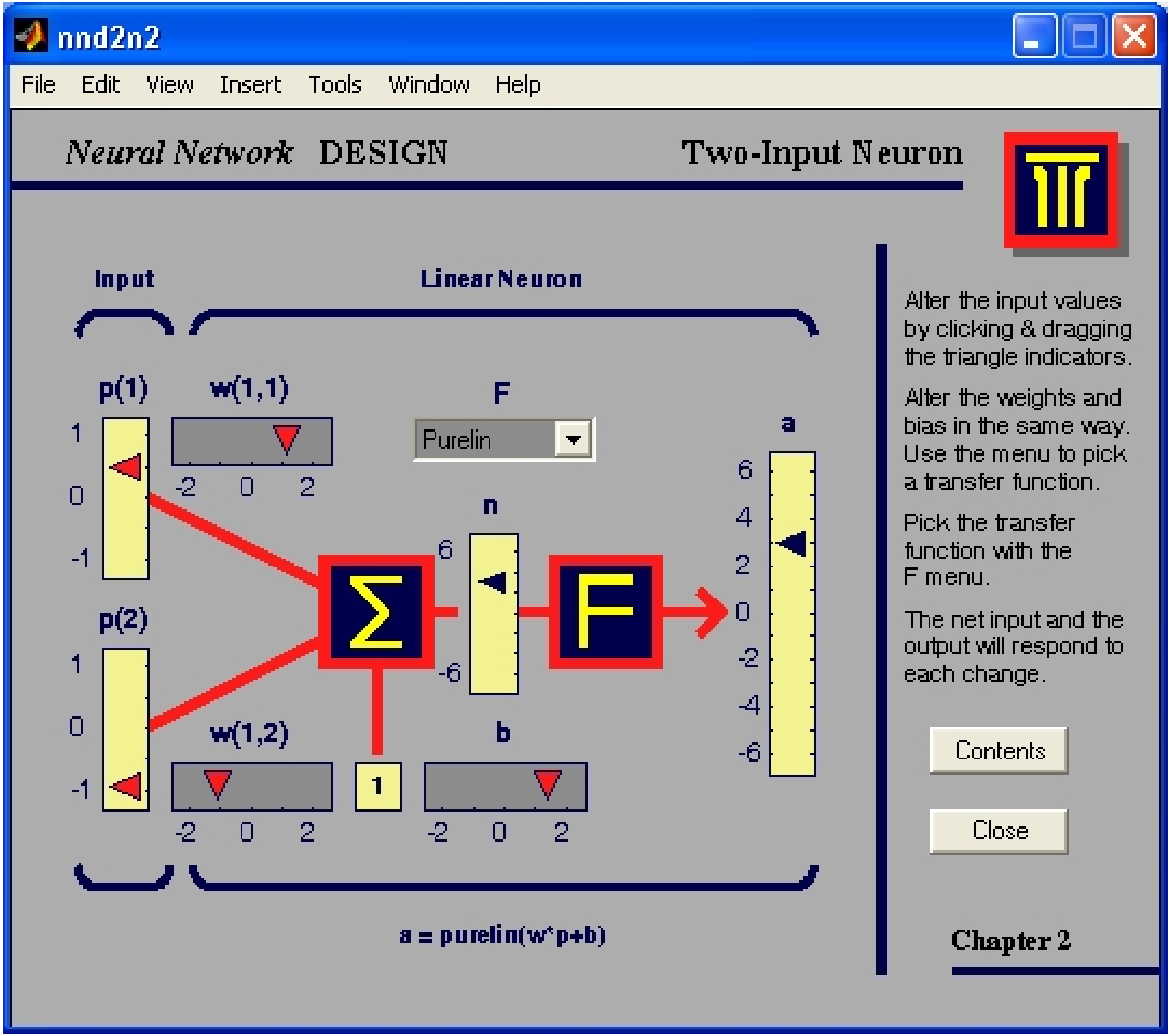




$$f(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha\,(\exp(x) - 1) & \text{if } x \le 0 \end{cases}$$

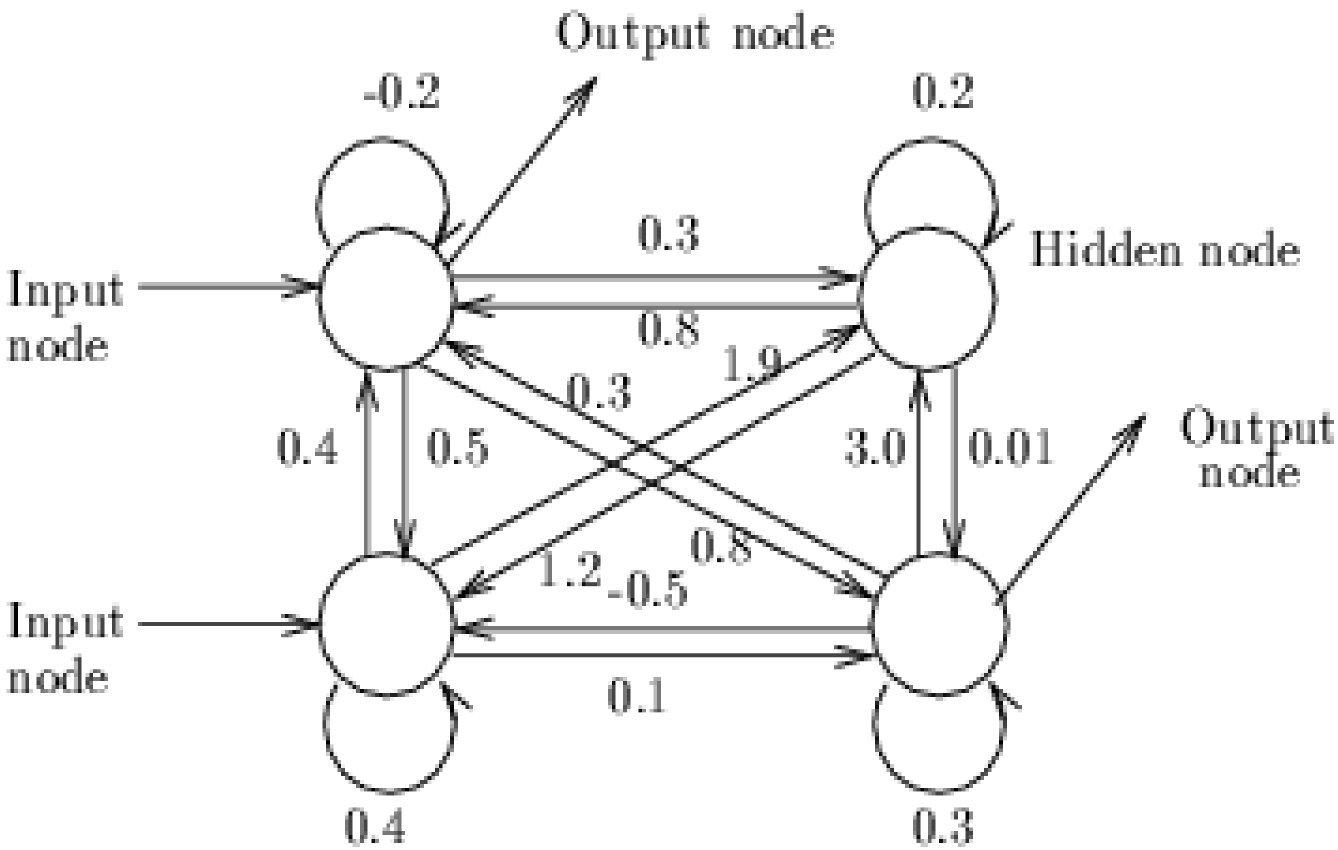


nnd2n1
File Edit View Insert Tools Window Help
Neural Network DESIGN
One-Input Neuron
Input
Log Sigmoid Neuron: a = logsig(w*p+b)
p
w
b
1
a
Alter the weight, bias and input by dragging the triangular shaped indicators.
Pick the transfer function with the F menu.
Watch the change to the neuron function and its output.
w
-2 0 2
b
-2 0 2
F:
Logsig
a
4
2
0
-2
-4
-4 -2 0 2 4
p
Contents
Close
Chapter 2


nnd2n2
File Edit View Insert Tools Window Help
Neural Network DESIGN
Two-Input Neuron
Input
Linear Neuron
p(1)
w(1,1)
F
Purelin
a
n
p(2)
w(1,2)
b
Alter the input values by clicking & dragging the triangle indicators.
Alter the weights and bias in the same way. Use the menu to pick a transfer function.
Pick the transfer function with the F menu.
The net input and the output will respond to each change.
Contents
Close
a = purelin(w*p+b)
Chapter 2

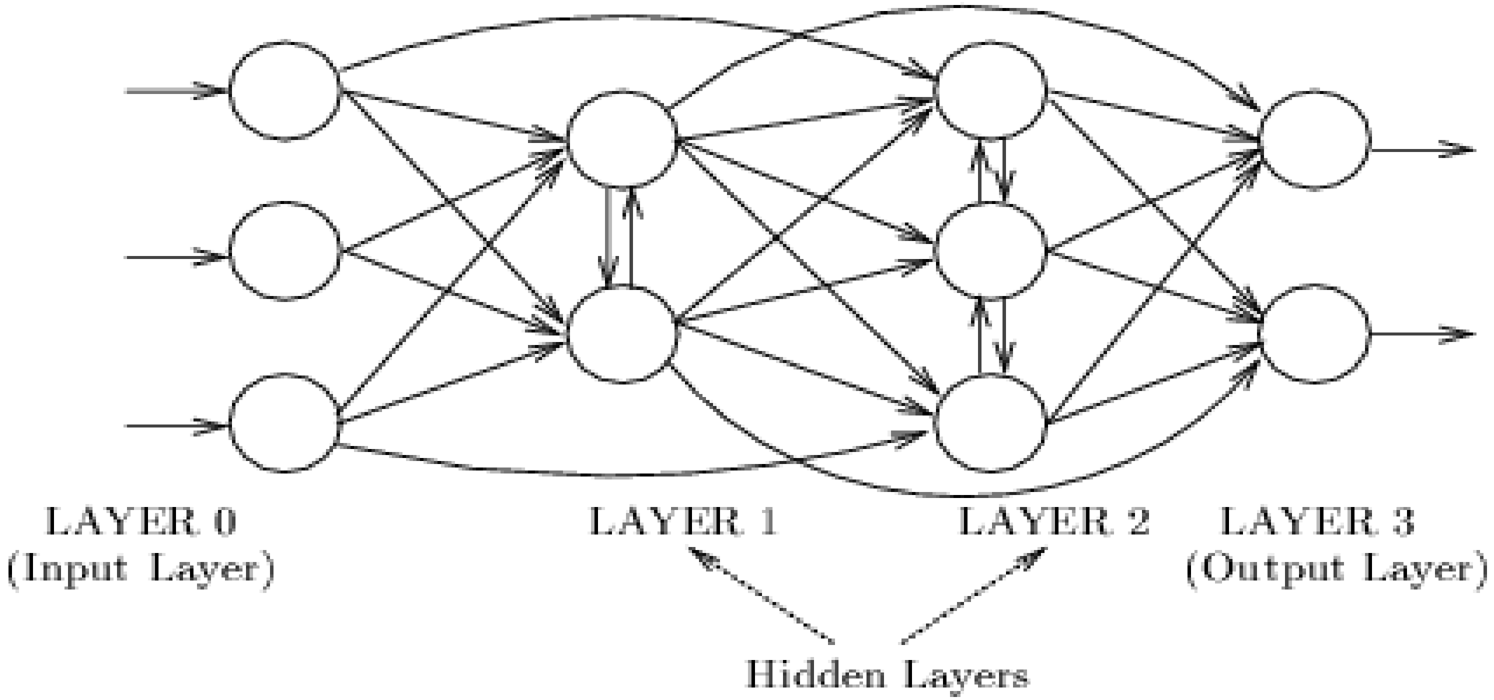
Structure of Neural Networks

Fully Connected Neural Network



Input neurons, Hidden neurons, Output neurons

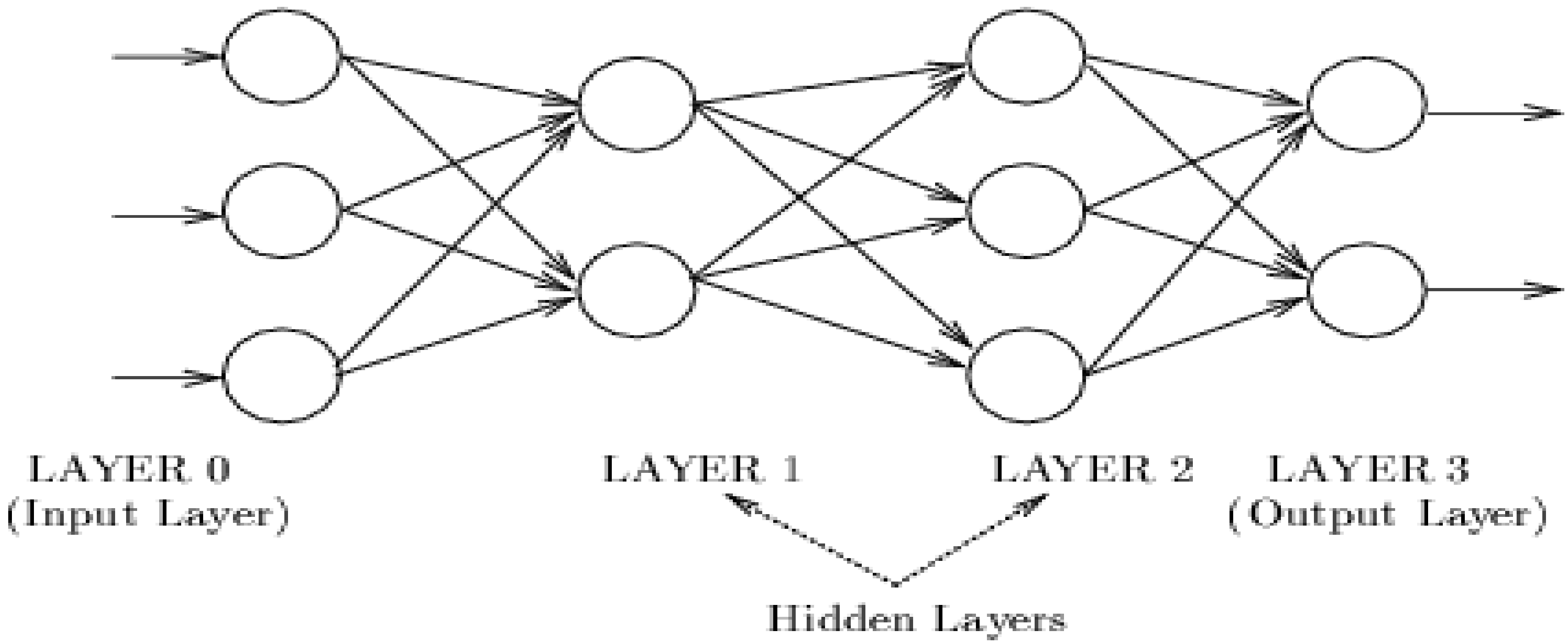
Layers, Feedforward neural networks

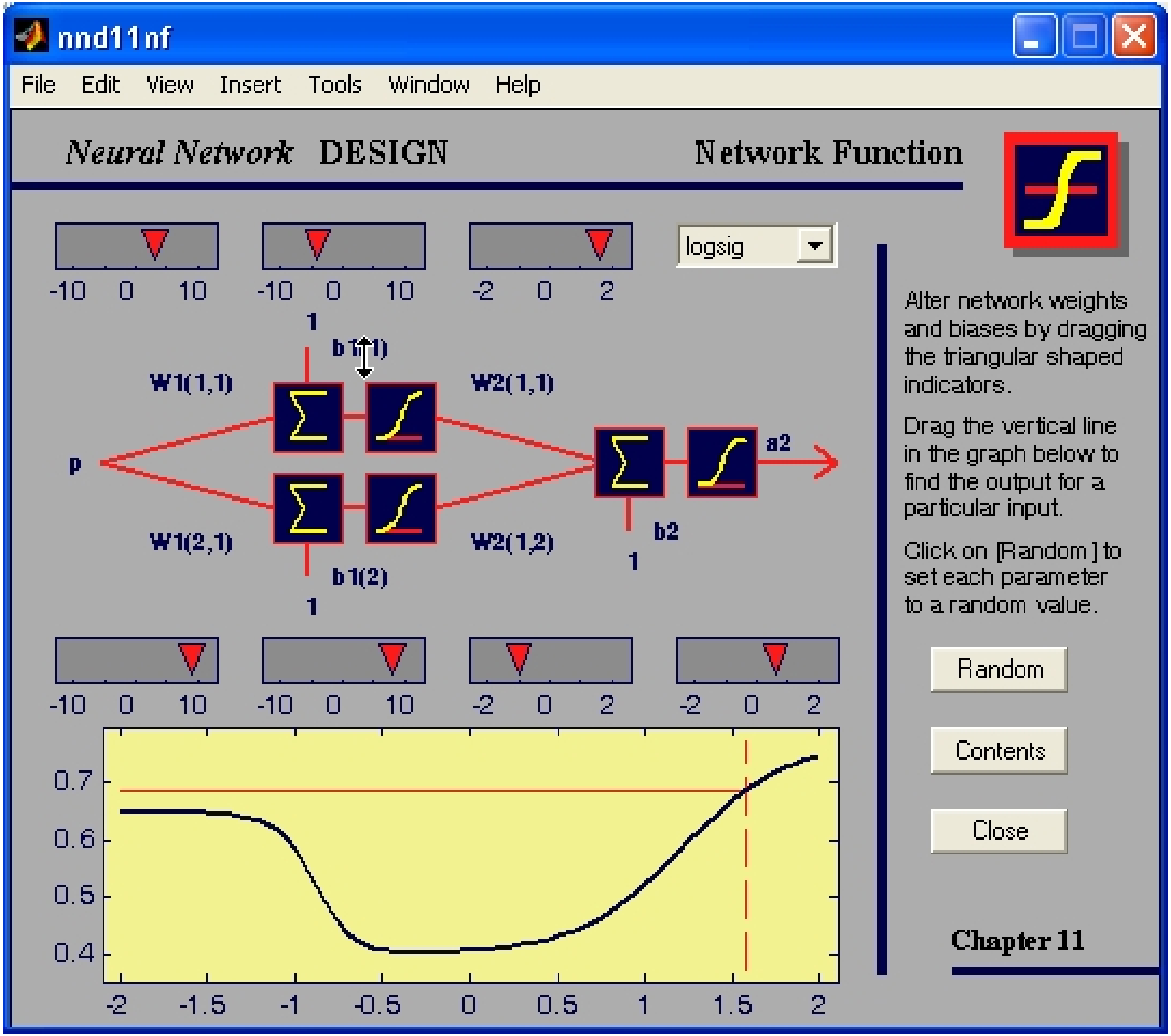


Convention: The input layer is Layer 0.

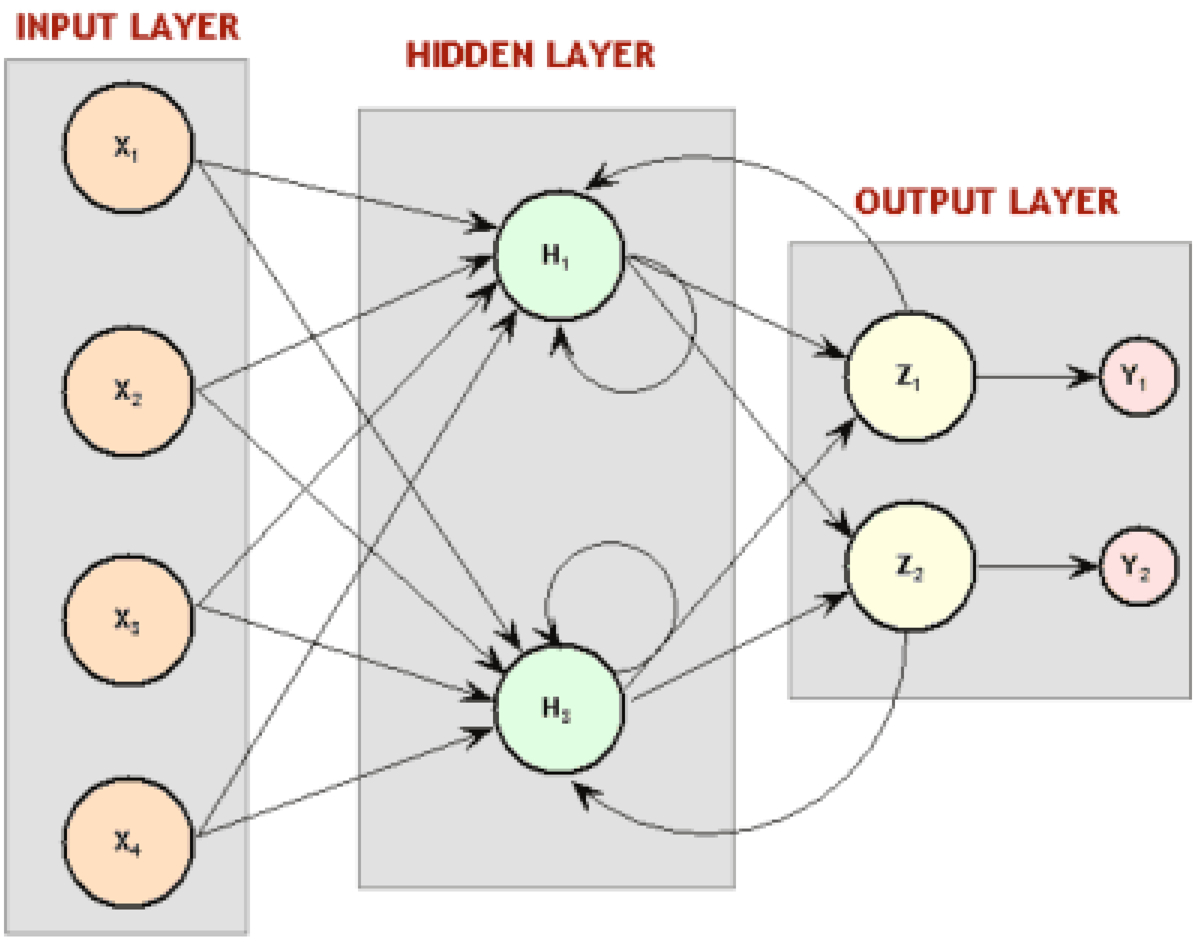
Multilayer Perceptron

- **Multilayer perceptron:** Connections only between Layer i and Layer i+1
- The most popular architecture.




nnd11nf
File Edit View Insert Tools Window Help
Neural Network DESIGN
Network Function
logsig
-10 0 10
-10 0 10
-2 0 2
1
b1(1)
W1(1,1)
W2(1,1)
p
a2
W1(2,1)
W2(1,2)
b2
1
b1(2)
1
-10 0 10
-10 0 10
-2 0 2
-2 0 2
0.7
0.6
0.5
0.4
-2 -1.5 -1 -0.5 0 0.5 1 1.5 2
Alter network weights and biases by dragging the triangular shaped indicators.
Drag the vertical line in the graph below to find the output for a particular input.
Click on [Random] to set each parameter to a random value.
Random
Contents
Close
Chapter 11

Recurrent Neural Networks



Recurrent NN: there are connections backwards too.

The Perceptron

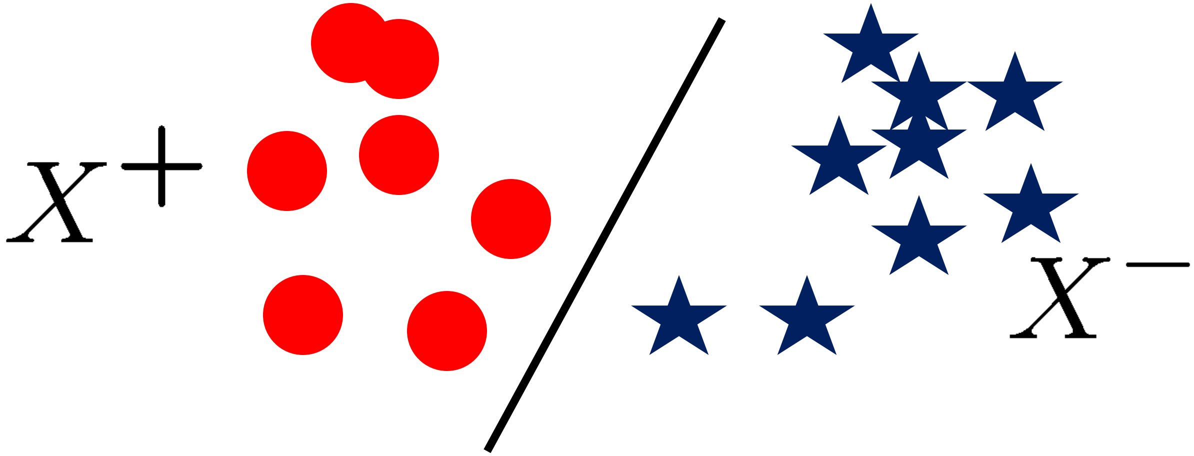
The Training Set

Let

$$X^+ = \{\mathbf{x}_k | \mathbf{x}_k \in \text{Class A}\}$$

$$X^- = \{\mathbf{x}_k | \mathbf{x}_k \in \text{Class B}\}$$

be the training set. Assume that they are linearly separable.

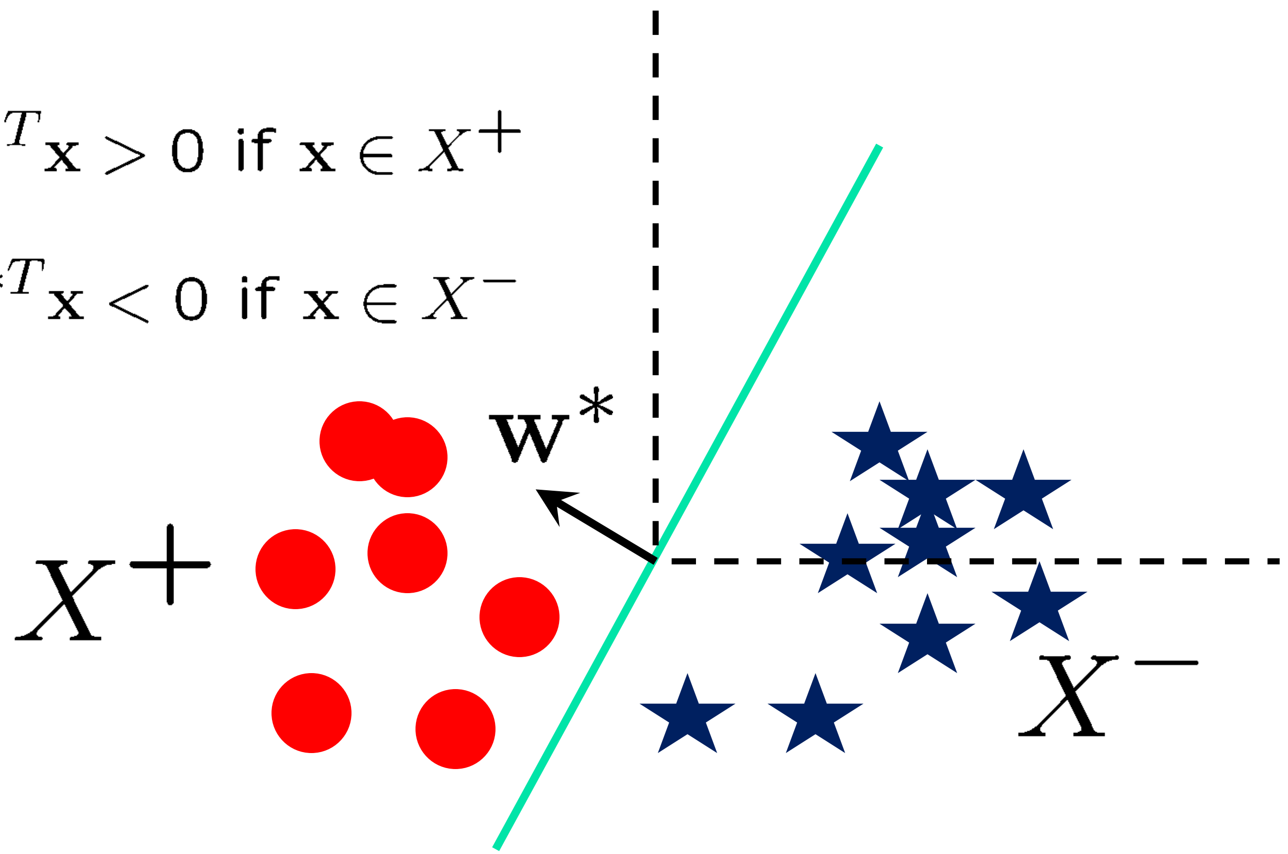


The Perceptron

Let $\mathbf{w}^*$ be the normal vector of the separating hyperplane through the origin:

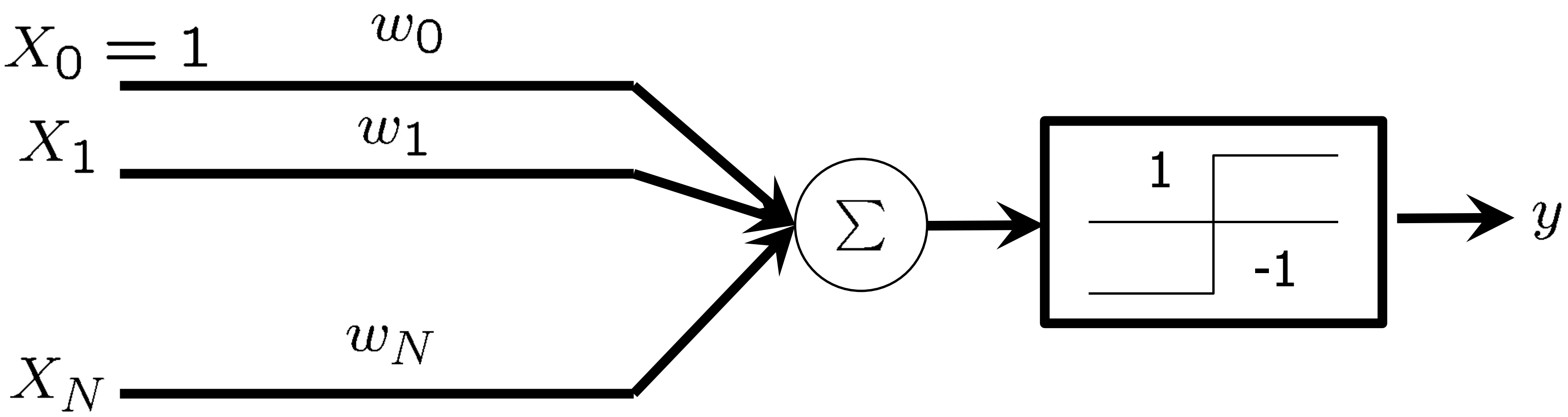
$\mathbf{w}^{*T}\mathbf{x} > 0$ if $\mathbf{x} \in X^+$

$\mathbf{w}^{*T}\mathbf{x} < 0$ if $\mathbf{x} \in X^-$



Goal: find such $\mathbf{w}^*$

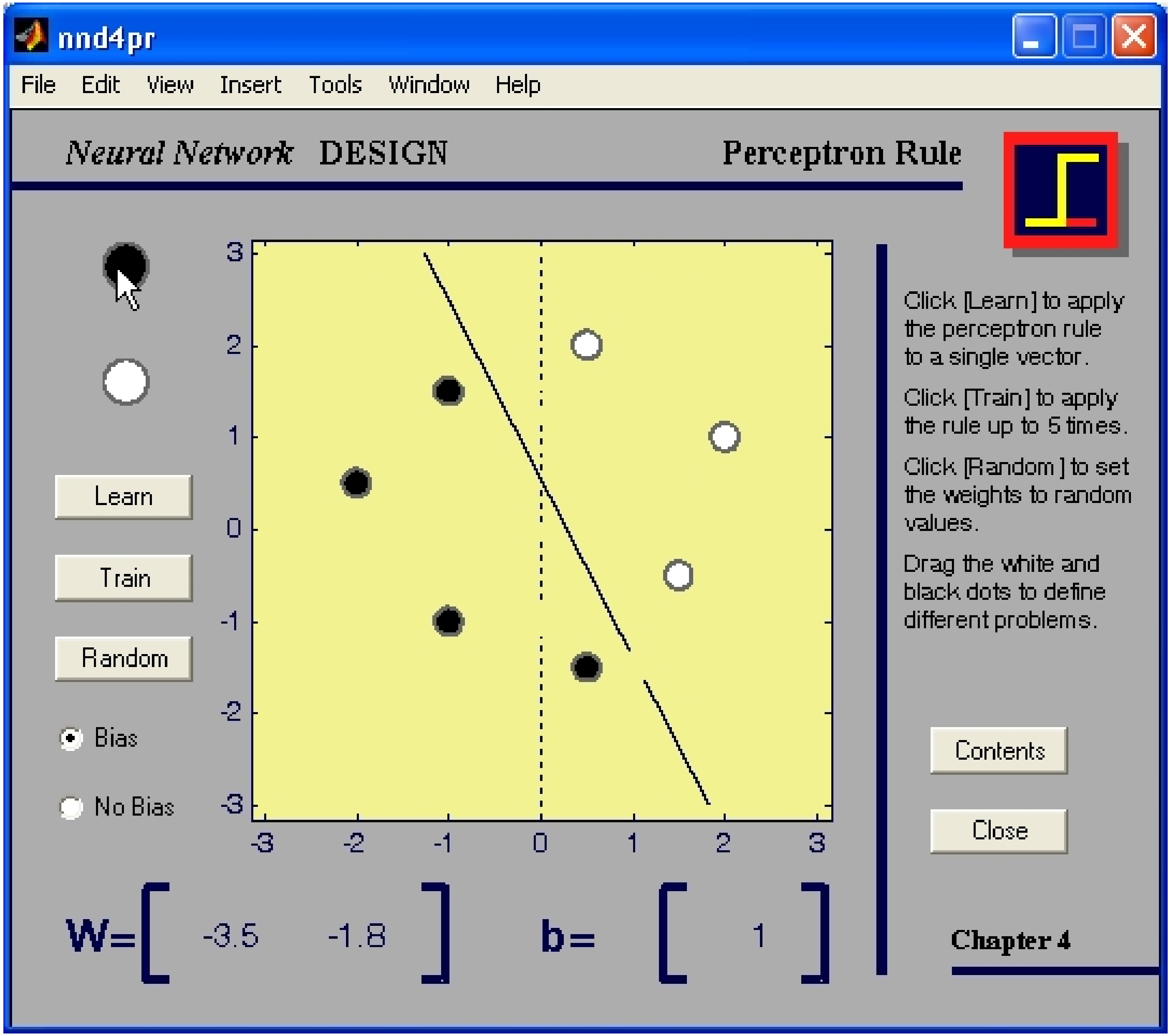
The Perceptron



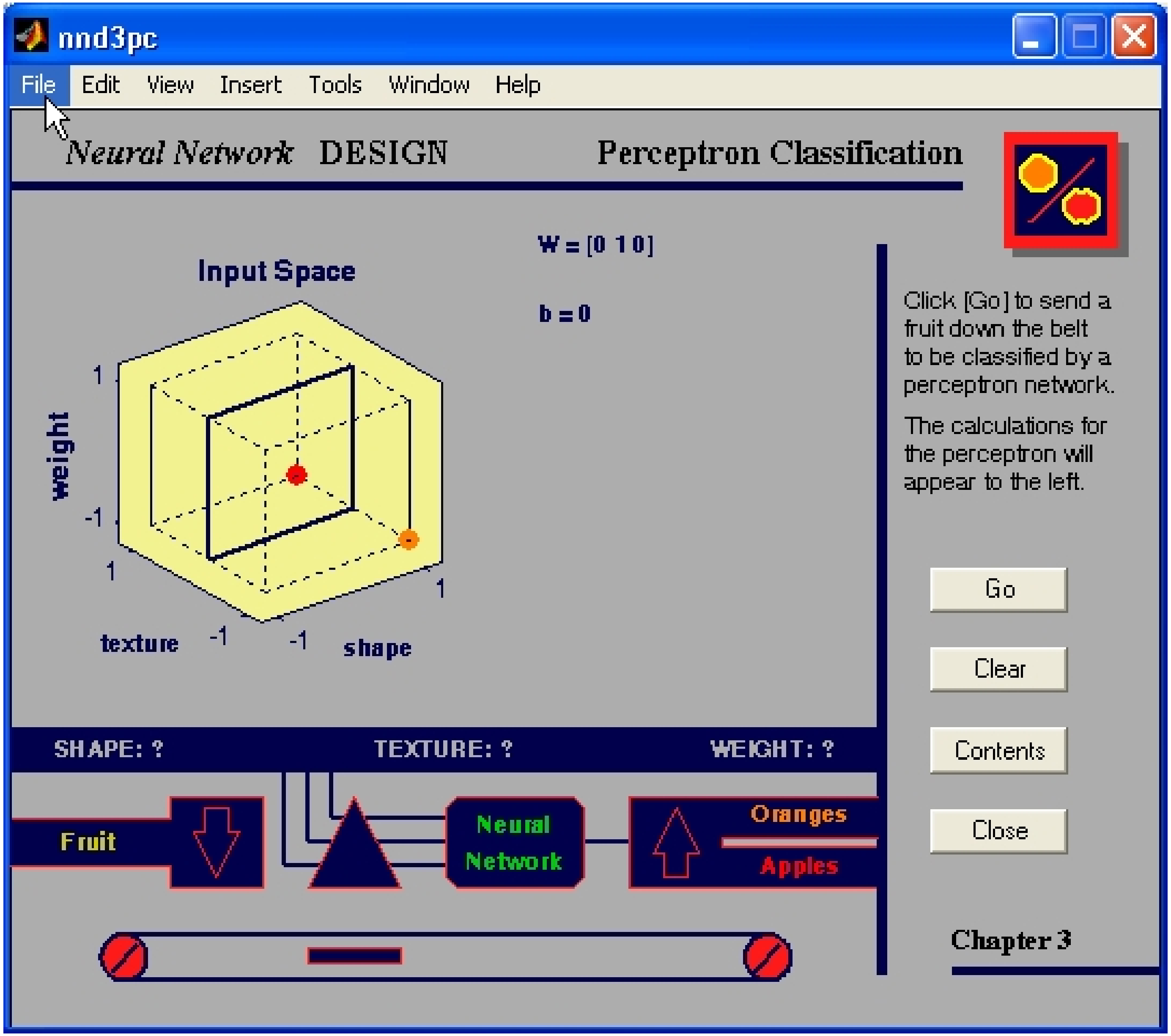
$$y = sgn(\sum_{i=0}^{N} w_i x_i)$$

$$y = sgn(\mathbf{w}^T \mathbf{x}) \in \{-1, 1\}$$

Matlab: opengl hardwarebasic, nnd4pr



Matlab demos: nnd3pc



The Perceptron Algorithm

The Perceptron algorithm

The perceptron learning algorithm

$$\widehat{y}(k) = sgn(\mathbf{w}(k-1)^T\mathbf{x}(k))$$

$$\mathbf{w}(k) = \mathbf{w}(k-1) + \mu(y(k) - \widehat{y}(k))\mathbf{x}(k)$$

$$\boxed{\mathbf{w}(k) = \mathbf{w}(k-1) + \mu\varepsilon(k)\mathbf{x}(k)}$$

- $\mu > 0$ learning rate
- if $y(k), \widehat{y}(k) \in \{-1, 1\} \Rightarrow \varepsilon(k) \in \{0, 2, -2\}$

The perceptron algorithm

1., If $k = 1$, let $\mathbf{w}(0)$ be arbitrary.

2., Let $\mathbf{x}(k) \in X^{+} \bigcup X^{-}$ be a training point misclassified by $\mathbf{w}(k-1)$

3., If there is no such vector $\Rightarrow$ 5.

4., If $\exists$ a misclassified vector $\Rightarrow \begin{cases} \hat{y}(k) = sgn(\mathbf{w}(k-1)^T \mathbf{x}(k)) \\ \alpha(k) = \mu\epsilon(k) = \mu(y(k) - \hat{y}(k)) \\ \mathbf{w}(k) = \mathbf{w}(k-1) + \alpha(k)\mathbf{x}(k) \\ k = k+1 \\ \text{Back to 2} \end{cases}$

5., END

Observation

- If $y(k) = 1$ and $\mathbf{w}(k-1)^T \mathbf{x}(k) < 0 \Rightarrow \alpha(k) > 0$.

- If $y(k) = -1$ and $\mathbf{w}(k-1)^T \mathbf{x}(k) > 0 \Rightarrow \alpha(k) < 0$.

The Perceptron Algorithm

$$\mathbf{w}(k) = \mathbf{w}(k-1) + \mu(y(k) - \widehat{y}(k))\mathbf{x}(k)$$

How can we remember this rule?

Gradient descent on $\frac{1}{2}(y(k) - \widehat{y}(k))^2$ with learning rate μ:

$$\mathbf{w}(k) = \mathbf{w}(k-1) - \mu\frac{\partial\frac{1}{2}(y(k)-\widehat{y}(k))^2}{\partial\mathbf{w}(k-1)} \text{ where } \widehat{y}(k) = \mathbf{w}(k-1)^T\mathbf{x}(k)$$

An interesting property:

we do not require the learning rate to go to zero!

The Perceptron Algorithm

- Each input $\mathbf{x}_i$ determines a hyperplane orthogonal to $\mathbf{x}_i$
- On the $+$ side of the hyperplane for each $\mathbf{w} \in \mathbb{R}^n$: $\mathbf{w}^T\mathbf{x}_i > 0$, $sgn(\mathbf{w}^T\mathbf{x}_i) = 1$
- On the $-$ side of the hyperplane for each $\mathbf{w} \in \mathbb{R}^n$: $\mathbf{w}^T\mathbf{x}_i < 0$, $sgn(\mathbf{w}^T\mathbf{x}_i) = -1$
- We need to update the weights, if $\exists \mathbf{x}_i$ in the training set, such that $sign(\mathbf{w}^T\mathbf{x}_i) \neq y_i$, where $y_i = class(\mathbf{x}_i) \in \{-1, 1\}$
- Then update $\mathbf{w}$ such that $\hat{y}_i = \mathrm{sgn}((\mathbf{w} \pm |\alpha_i|\mathbf{x}_i)^T\mathbf{x}_i)$ gets closer to $y_i \in \{-1, 1\}$

Perceptron Convergence

Theorem

If the samples are linearly separable, then the perceptron algorithm finds a separating hyperplane in finite steps.

The running time does not depend on the sample size n.

Perceptron Convergence

Proof of the Theorem

Lemma Let

$$\bar{X} = X^{+} \bigcup \{-X^{-}\}$$

Then $\exists b > 0$ such that $\forall \bar{\mathbf{x}} \in \bar{X}$ we have $\mathbf{w}^{*T}\bar{\mathbf{x}} \geq b > 0$

Proof of the Lemma:

Since

$$\mathbf{w}^{*T}\mathbf{x} > 0 \text{ if } \mathbf{x} \in X^{+}$$

$$\mathbf{w}^{*T}\mathbf{x} < 0 \text{ if } \mathbf{x} \in X^{-}$$

by the definition of X^{+} and X^{-},

therefore $\exists b > 0$ such that $\forall \bar{\mathbf{x}} \in \bar{X}$ we have $\mathbf{w}^{*T}\bar{\mathbf{x}} \geq b > 0$.

Perceptron Convergence

We need an update step at iteration $k-1$, if $\exists \bar{\mathbf{x}} \in \bar{X}$ such that $\mathbf{w}(k-1)^T\bar{\mathbf{x}} \leq 0$. Let this $\bar{\mathbf{x}}$ be denoted by $\bar{\mathbf{x}}(k)$.

If $\bar{\mathbf{x}}(k) \in X^+ \Rightarrow \mathbf{x}(k) = \bar{\mathbf{x}}(k) \in X^+$.

If $\bar{\mathbf{x}}(k) \in -X^- \Rightarrow \mathbf{x}(k) = -\bar{\mathbf{x}}(k) \in X^-$.

Lemma Using this notation, the update rule can be written as

$$\mathbf{w}(k) = \mathbf{w}(k-1) + \alpha(k)\mathbf{x}(k) = \mathbf{w}(k-1) + \bar{\alpha}\bar{\mathbf{x}}(k)$$

where $\bar{\alpha} > 0$ is an arbitrary constant.

Proof

- If $\mathbf{x}(k) \in X^+$, $\mathbf{w}(k-1)^T\mathbf{x}(k) < 0 \Rightarrow \alpha(k) > 0$, $\bar{\alpha} = \alpha(k) > 0, \bar{\mathbf{x}}(k) = \mathbf{x}(k)$

- If $\mathbf{x}(k) \in X^-$, $\mathbf{w}(k-1)^T\mathbf{x}(k) > 0 \Rightarrow \alpha(k) < 0$, $\bar{\alpha} = -\alpha(k) > 0, \bar{\mathbf{x}}(k) = -\mathbf{x}(k)$

In both cases $\alpha(k)\mathbf{x}(k) = \bar{\alpha}\bar{\mathbf{x}}(k)$.

Perceptron Convergence

Lemma

Let

$$\mathbf{w}(k) = \mathbf{w}(k-1) + \bar{\alpha}\bar{\mathbf{x}}(k)$$

where $\bar{\alpha} > 0$ is an arbitrary constant. Then,

$$\mathbf{w}(k)^T\bar{\mathbf{x}}(k) = \underbrace{\mathbf{w}(k-1)^T\bar{\mathbf{x}}(k)}_{\leq 0} + \underbrace{\bar{\alpha}\bar{\mathbf{x}}(k)^T\bar{\mathbf{x}}(k)}_{>0}$$

Therefore,

$$\mathbf{w}(k)^T\bar{\mathbf{x}}(k) > \mathbf{w}(k-1)^T\bar{\mathbf{x}}(k)$$

Perceptron Convergence

Let us see how the weights change on set $\bar{X}$.

$$\begin{aligned}
\mathbf{w}(0) &= 0 \\
\mathbf{w}(1) &= \bar{\alpha}\bar{\mathbf{x}}(1) \\
\mathbf{w}(2) &= \mathbf{w}(1) + \bar{\alpha}\bar{\mathbf{x}}(2) = \bar{\alpha}(\bar{\mathbf{x}}(1) + \bar{\mathbf{x}}(2)) \\
&\vdots \\
\mathbf{w}(k) &= \mathbf{w}(k-1) + \bar{\alpha}\bar{\mathbf{x}}(k) = \bar{\alpha}\sum_{i=1}^{k}\bar{\mathbf{x}}(i)
\end{aligned}$$

Therefore,

$$\mathbf{w}^T(k)\mathbf{w}^* = \bar{\alpha}\sum_{i=1}^{k}\bar{\mathbf{x}}(i)^T\mathbf{w}^* \geq \bar{\alpha}kb$$

Lower bound

We have proved:

$$\mathbf{w}^T(k)\mathbf{w}^* \;=\; \bar{\alpha}\sum_{i=1}^{k}\bar{\mathbf{x}}(i)^T\mathbf{w}^* \geq \alpha k b$$

From Cauchy-Schwarz

$$\| \mathbf{w}(k) \|^2 \| \mathbf{w}^* \|^2 \;\geq\; (\mathbf{w}^T(k)\mathbf{w}^*)^2 \geq \alpha^2 k^2 b^2$$

Therefore,

$$\| \mathbf{w}(k) \|^2 \geq \frac{\alpha^2 k^2 b^2}{\| \mathbf{w}^* \|^2}$$

and thus $\| \mathbf{w}(k) \|^2$ is at least quadratic in k.

Upper bound

Let us find an upperbound on $\mathbf{w}(k)$.

Let $\mathbf{w}(0) = 0$, and let $M > \max_{\bar{\mathbf{x}}(i) \in \bar{X}} \| \bar{\mathbf{x}}(i) \|^2$

$$
\begin{aligned}
\mathbf{w}(k) &= \mathbf{w}(k-1) + \bar{\alpha}\bar{\mathbf{x}}(k) \\
\| \mathbf{w}(k) \|^2 &= \| \mathbf{w}(k-1) \|^2 + 2\bar{\alpha} \underbrace{\mathbf{w}^T(k-1)\bar{\mathbf{x}}(k)}_{\leq 0} + \bar{\alpha}^2 \| \bar{\mathbf{x}}(k) \|^2
\end{aligned}
$$

$\mathbf{w}^T(k-1)\bar{\mathbf{x}}(k) \leq 0$ since we had to make an update step.

Therefore,

$$
\| \mathbf{w}(k) \|^2 - \| \mathbf{w}(k-1) \|^2 \leq \bar{\alpha}^2 \| \bar{\mathbf{x}}(k) \|^2
$$

Upper bound

Therefore,

$$\begin{aligned}
\| \mathbf{w}(k) \|^2 - \| \mathbf{w}(k-1) \|^2 &\leq \bar{\alpha}^2 \| \bar{\mathbf{x}}(k) \|^2 \\
\| \mathbf{w}(k-1) \|^2 - \| \mathbf{w}(k-2) \|^2 &\leq \bar{\alpha}^2 \| \bar{\mathbf{x}}(k-1) \|^2 \\
&\vdots \\
\| \mathbf{w}(1) \|^2 - \| \mathbf{w}(0) \|^2 &\leq \bar{\alpha}^2 \| \bar{\mathbf{x}}(1) \|^2
\end{aligned}$$

$$\begin{aligned}
\Rightarrow \| \mathbf{w}(k) \|^2 &\leq \bar{\alpha}^2 \sum_{i=1}^{k} \| \bar{\mathbf{x}}(i) \|^2 \\
\Rightarrow \| \mathbf{w}(k) \|^2 &\leq \bar{\alpha}^2 k M
\end{aligned}$$

$\Rightarrow \| \mathbf{w}(k) \|^2$ does not grow faster than a linear function in k.

The Perceptron Algorithm

We have proved:

$$\| \mathbf{w}(k) \|^2 \geq \frac{\bar{\alpha}^2 k^2 b^2}{\| \mathbf{w}^* \|^2}$$

$$\| \mathbf{w}(k) \|^2 \leq \bar{\alpha}^2 k M$$

- Therefore k is finite, and there exists k_{max}
- k_{max} does not depend on the size of the training set.
- $\alpha > 0$ arbitrary fixed.

Take me home!

- History of Neural Networks
- Mathematical model of the neuron
- Activation Functions
- Perceptron definition
- Perceptron algorithm
- Perceptron Convergence Theorem