COMP3308/3608, Lecture 8a

ARTIFICIAL INTELLIGENCE

Introduction to Neural Networks. Perceptrons.

Reference: Russell and Norvig, pp.727-731

Outline

- **Introduction to Neural Networks**
  - **Human nervous system**
  - **Structure and operation of biological neural systems**
  - **What is an artificial neural network?**
  - **Taxonomy of neural networks**

- **Perceptron**
  - **Neuron model**
  - **Investigation of decision boundary**
  - **Learning rule**
  - **Example**
  - **Capabilities and limitations**

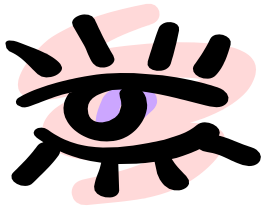
Artificial Neural Networks

- **Field of AI that studies networks of artificial neurons**
- **It was inspired by the desire to:**
  - **produce artificial systems capable of "intelligent" computations, similar to what the human brain does**
  - **enhance our understanding of the human brain**

- **Artificial neurons are very simple abstractions of biological neurons**
- **They are connected to form networks of neurons**
- **These networks:**
  - **are implemented as a computer program or specialized hardware**
  - **do not have a fraction of the power of the human brain but can be trained to perform useful functions, e.g. classification or prediction**

Efficiency of Biological Neural Systems

- **The brain performs tasks such as pattern recognition, perception, motor control *many times faster* than the fastest computers**
- **Efficiency of the human visual system**
  - **Humans – can do perceptual recognition in 100-200 ms (e.g. recognition of a familiar face in a unfamiliar scene)**
  - **Computers – still not able to do this well enough**



- **Efficiency of the sonar system of a bat**
  - **Sonar = SOund NAvigation and Ranging (emitting sounds and listening for echoes; used underwater to navigate and detect other vessels)**
  - **A bat sonar provides information about the distance from a target, its relative velocity, size, azimuth, elevation; the size of various features of the target**
  - **This complex computation (extracting information from the echo) occurs in a brain with the size of a plum!**
  - **The precision of the target location achieved by the bat is still impossible to match by the current radars**
- **How does the human brain or the brain of a bat do this?**

Human Brain

- **We still don't fully understand how it operates**
- **What we know is that we have a huge number of *neurons* (brain cells) and *connections* between them**
  - **100 billion (i.e. 10^{10}) neurons in the brain**
    - **a full Olympic size swimming pool contains 10^{10} raindrops; the number of stars in the Milky Way is of the same magnitude**
  - **10^4 connections per neuron**
  - **=> total number of connections = 10^{10} . 10^4 =100 000 000 000 000**

- **Biological neurons are much slower than computers**
  - **Neurons operate in mili (10^{-3}) seconds, computers in nano (10^{-9}) seconds**

Biological Neuron

- **Structure**

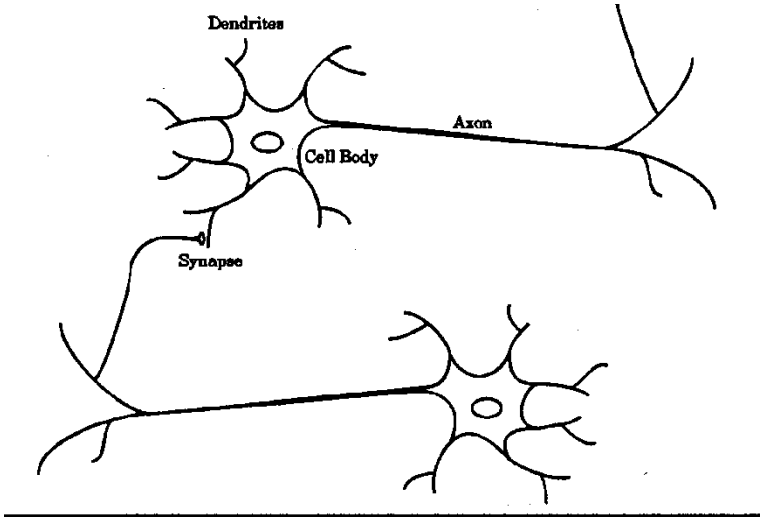


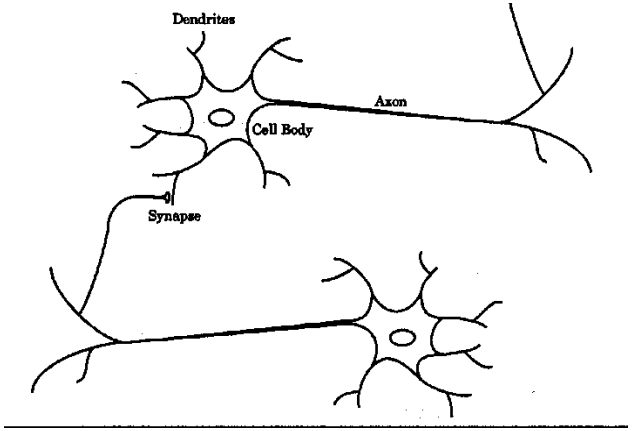
Image ref: R. Beale and T. Jackson, Neural Computing - An Introduction, 1990, CRC Press

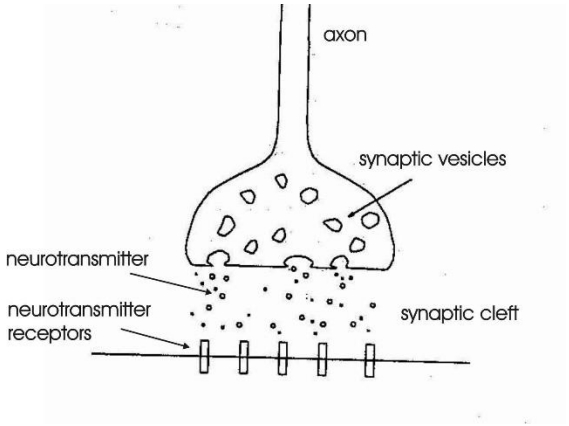
- **body – contains the chromosomes**
- **dendrites – inputs**
- **axon – output**
- **synapse - a narrow *gap* (not a link)**
  - **between the axon of one neuron and a dendrite of another neuron**
  - **can get activated *chemically* allowing information from one neuron to pass to another**

- **Purpose: transmit information in the form of *electrical signals***
  - **A neuron accepts many input signals via its dendrites and adds them together**
  - **If the input signal is strong enough, the neuron gets activated and an electrical signal is generated at its output**
  - **This electrical signal activates the synapse (chemically) and the signal is passed to the input of the connected neuron**

More on the operation of biological neurons

- **Signals are transmitted between neurons by electrical pulses (action potentials, AP) traveling along the axon**
- **When the potential at the synapse is raised sufficiently by the AP, it releases chemicals called neurotransmitters**
  - **it may take the arrival of more than one AP before the synapse is triggered**





- **The neurotransmitters diffuse across the gap and chemically activate the gates on the dendrites, that allows charged ions to flow**

- **The flow of ions alters the potential of the dendrite and provides a voltage pulse on the dendrite (*post-synaptic-potential, PSP)***
  - **some synapses excite the dendrite they affect, while others inhibit it**
  - **the synapses also determine the strength of the new input signal**
- **each PSP travels along its dendrite and spreads over the body**
- **the body sums the effects of thousands PSPs; if the resulting PSP exceeds a threshold, the neuron fires and generates an AP**

Biological Neurons - Examples

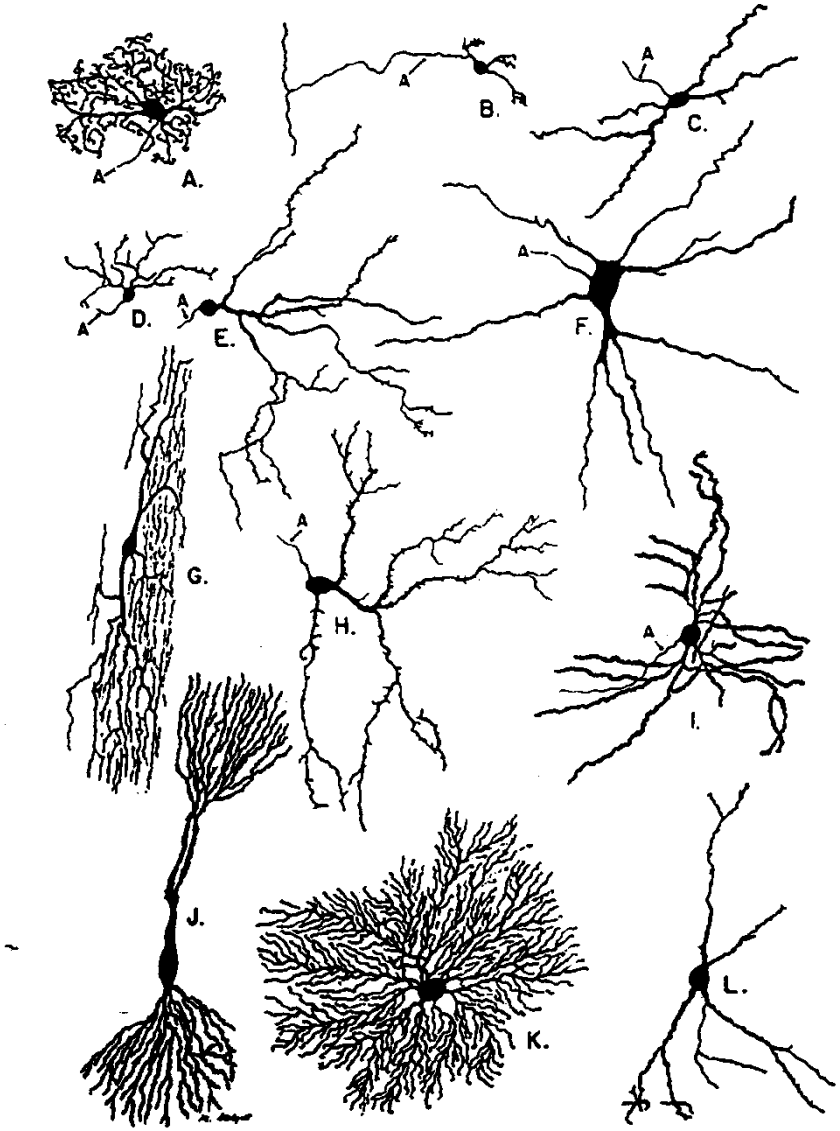


Image ref: J. Anderson, Introduction to Neural Networks, MIT press, 1995

Learning in Humans

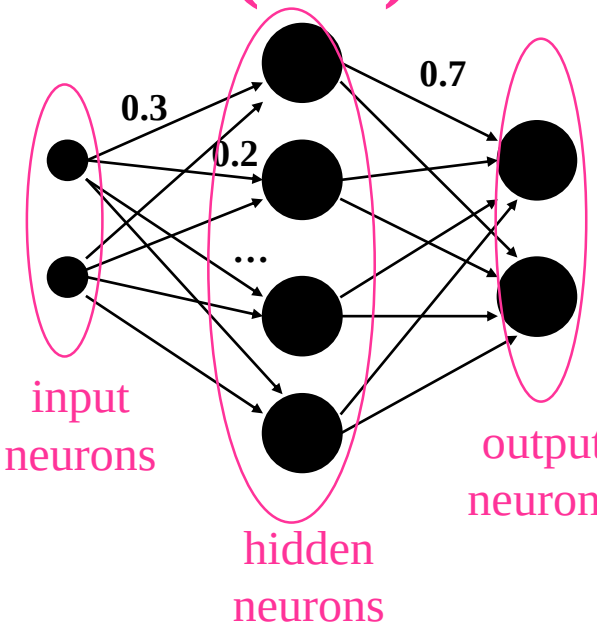
- **We were born with some of our neural structures (e.g. neurons); others (e.g. synapses) are formed and modified via learning and experience**

- **Learning is achieved by:**
  - **creation of new synaptic connections between neurons**
  - **changing the strength of existing synaptic connections**

- **The synapses are thought to be mainly responsible for learning**
  - **Hebb proposed his famous learning rule in 1949:**

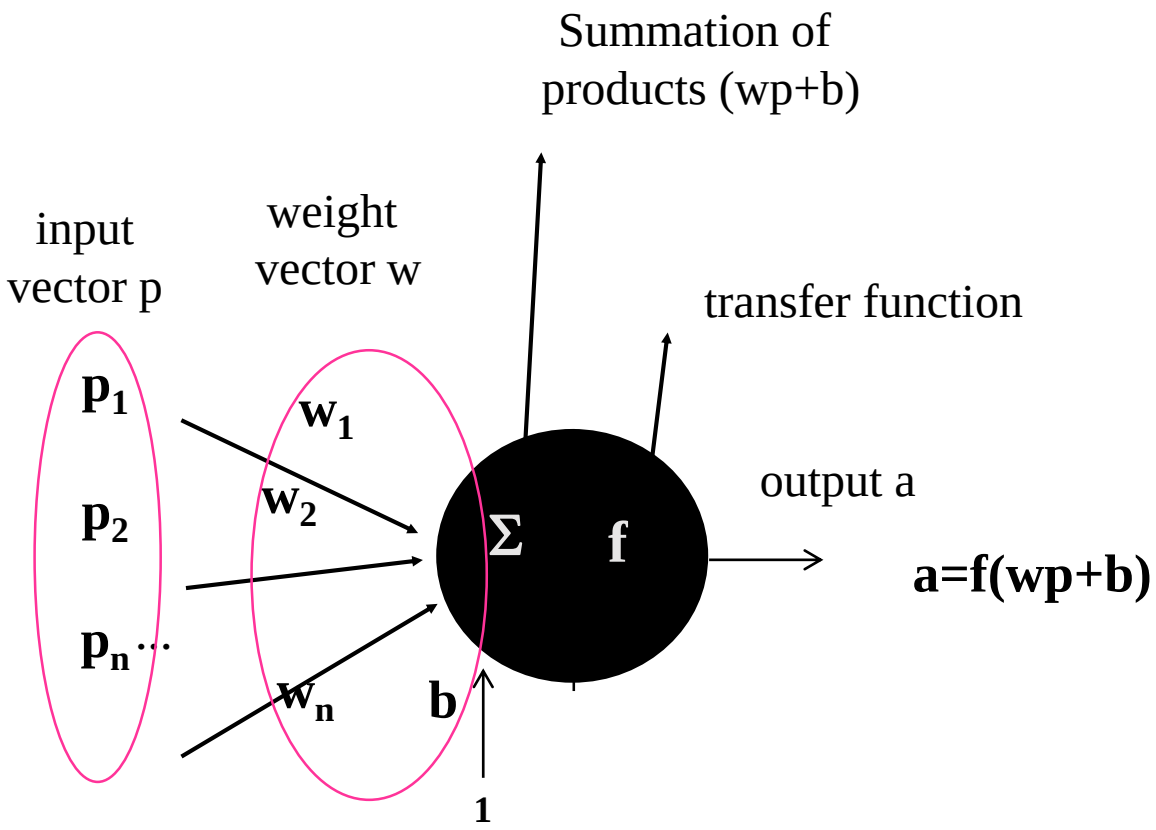
    The strength of a synapse between 2 neurons is increased by the repeated activation of one neuron by the other across the synapse

What is an Artificial Neural Network (NN)?



- **A network of many simple *neurons (units, nodes)***
  - **Neurons are linked by *connections***
  - **Each connection has a numeric *weight***
  - **Neurons:**
    - **receive numeric inputs (from the environment or other neurons) via the connections**
    - **produce numeric output using their weights and the inputs**
    - **are organised into layers: input, hidden and output neurons**
  - **A NN can be represented as a directed graph**
- **NNs *learn from examples***
  - **They can be used for supervised or unsupervised learning**
  - **The knowledge from training examples is stored in the *weights***
  - ***Learning rule* – a procedure for modifying the weights in order to perform a certain task**

A Neuron

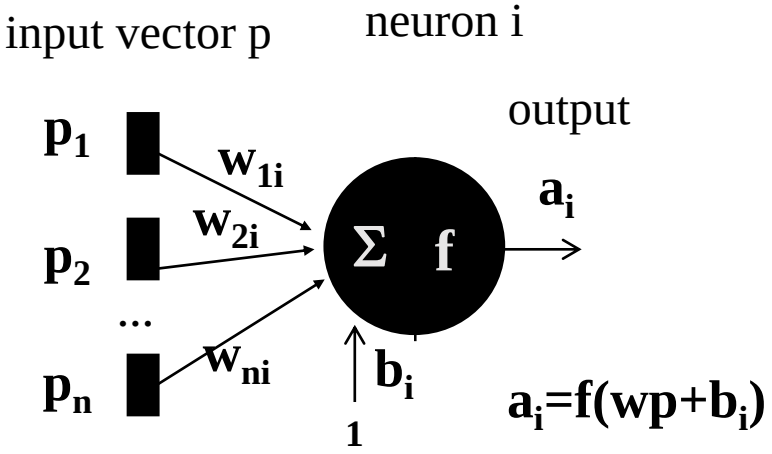


p comes from the data

w & b are the parameters of the neuron and they are learned using a given learning rule for the specific type of NN

A Neuron – More Details

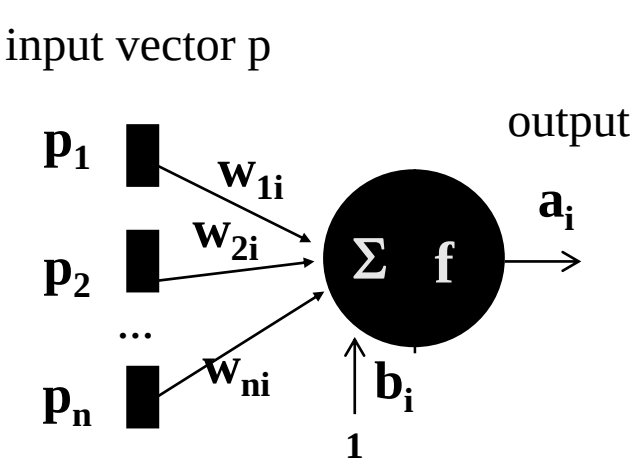
- **A connection from neuron *i* to *j* has a numeric weigh w_{ij} which determines its strength**
- **Given an input vector p, the neuron first computes the weighed sum wp, and then applies a transfer function *f* to determine the output *a***
- **The transfer function *f* has different forms depending on the type of NN**
- **A neuron typically has a special weight called *bias weight b* . It is connected to a fixed input of 1.**
- **A NN represents a function of the inputs p and weights w and b. By adjusting the weights, we change this function. This is done by using a learning rule.**



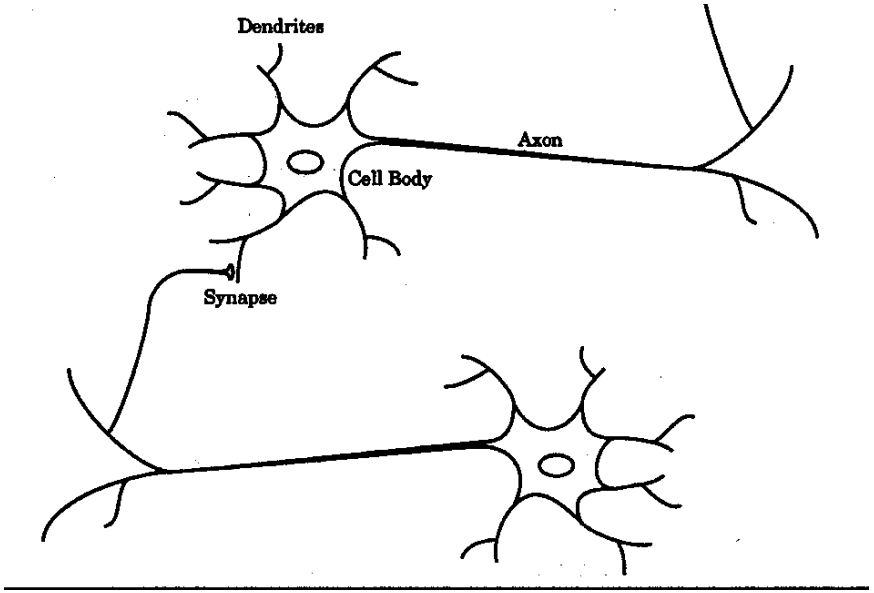
If there are 2 inputs $p_1=2$ and $p_2=3$, and if $w_{11}= 3$, $w_{12}=1$, $b = -1.5$, then
a = f(2*3+3*1 -1.5) = f(7.5)

Correspondence Between Artificial and Biological Neurons

- **How this artificial neuron relates to the biological one?**
  - **input vector p – input signals at the dendrites**
  - **weight *w* (or weight vector w) - strength of the synapse (or synapses)**
  - **summer & transfer function - body**
  - **output *a* - signal at the axon**



$a_i = f(wp + b_i)$



Taxonomy of NNs

- **Feedforward (acyclic) and recurrent (cyclic, feedback)**
- **Supervised and unsupervised**
- **Feedforward supervised networks**
  - **typically used for classification and regression**
  - **perceptrons, ADALINEs, multilayer backpropagation networks, Radial-Basis Function (RBF) networks, Learning Vector Quantization (LVQ) networks, deep neural networks**
- **Feedforward unsupervised networks**
  - **Hebbian networks used for associative learning**
  - **Competitive networks performing clustering and visualization, e.g. Self-Organizing Feature Maps (SOM)**
- **Recurrent networks – temporal data processing**
  - **recurrent backpropagation, associative memories, adaptive resonance networks, LSTM deep neural networks**

Perceptron

(1958)
F. Rosenblatt

The perceptron: a probabilistic model for information storage and organization in the brain
Psychological Review 65:386–408

If we are eventually to understand the capability of higher organisms for perceptual recognition, generalization, recall, and thinking, we must first have answers to three fundamental questions:

1. How is information about the physical world sensed, or detected, by the biological system?
2. In what form is information stored, or remembered?
3. How does information contained in storage, or in memory, influence recognition and behavior?

The first of these questions is in the province of sensory physiology, and is the only one for which appreciable understanding has been achieved. This article will be concerned primarily with the second and third questions, which are still subject to a vast amount of speculation, and where the few relevant facts currently supplied by neurophysiology have not yet been integrated into an acceptable theory.

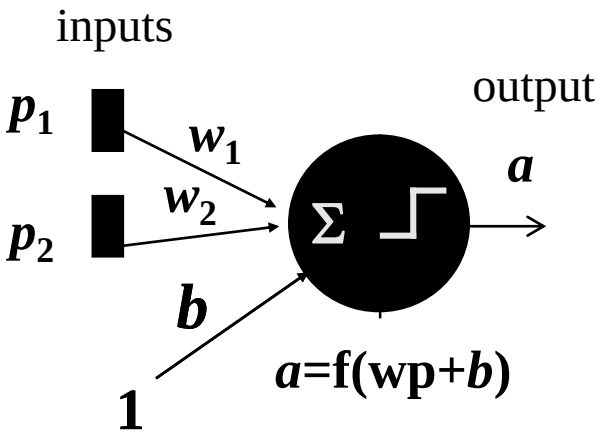
and the stored pattern. According to this hypothesis, if one understood the code or "wiring diagram" of the nervous system, one should, in principle, be able to discover exactly what an organism remembers by reconstructing the original sensory patterns from the "memory traces" which they have left, much as we might develop a photographic negative, or translate the pattern of electrical charges in the "memory" of a digital computer. This hypothesis is appealing in its simplicity and ready intelligibility, and a large family of theoretical brain models has been developed around the idea of a coded, representational memory (2, 3, 9, 14). The alternative approach, which stems from the tradition of British empiricism, hazards the guess that the images of stimuli may never really be recorded at all, and that the central nervous system simply acts as an intricate switching network, where retention takes the form of new connections, or pathways, between centers of activity. In many of the more recent developments of

Image ref: www-cse.ucsd.edu/~dasgupta/250B/lec3.ppt

Perceptron

- **A supervised NN**
- **Uses a step transfer function**
  => its output is binary: either 0 or 1 (or either -1 or 1)
- **Its output is a weighed sum of its inputs, subject to a step transfer function**
- **Forms a linear decision boundary**

Single-Neuron Perceptron - Example

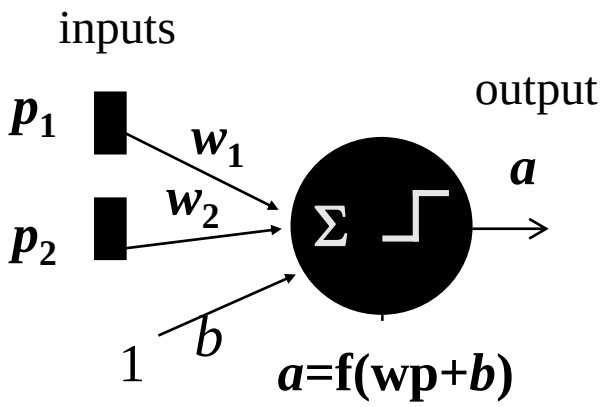


- **Step transfer function:**

$$a = f(n) = \begin{cases} 1 & if \quad n \geq 0 \\ 0 & if \quad n < 0 \end{cases}$$

- **2 inputs p_1 and p_2 ; 1 output a**
- **Each input is weighted (weights w_1 and w_2)**
- **An additional weight b (bias weight) associated with the neuron**
- **The sum of the weighted inputs is sent to a *step transfer function (*denoted as *step;* other names*: threshold* or *hard limit*)**

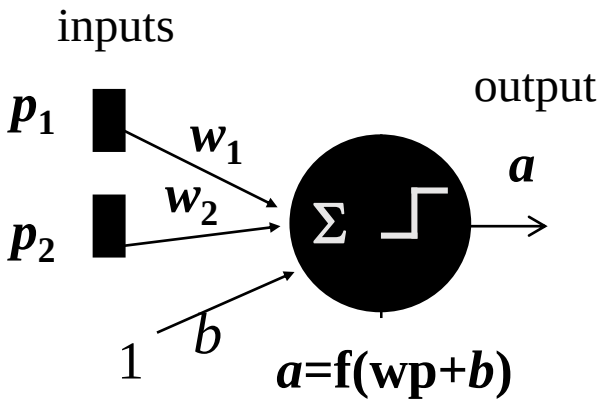
Single-Neuron Perceptron - Investigation of the Decision Boundaries



- **We will use both analytical and graphical representation**

Images in next slides from: Hagan, Demuth, Beale, Neural Network Design, 1996, PWS, Thomson

Decision Boundary (1)



- **1. Output:**

$a = step(\mathbf{wp}+b) =$
$= step(w_1 p_1 + w_2 p_2 + b)$

- **2. Decision boundary:**

$n = \mathbf{wp}+b = w_1 p_1 + w_1 p_2 + b = 0$

- **3. Let's assign values to the parameters of the perceptron (w and b):**

$w_1 = 1;\ w_2 = 1;\ b = -1;$

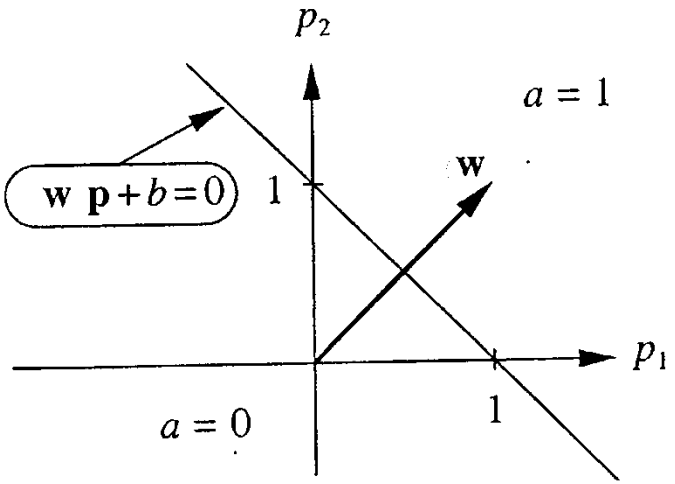
- **4. The decision boundary is:**

$n = \mathbf{wp}+b = w_1 p_1 + w_2 p_2 + b =$
$= p_1 + p_2 - 1 = 0$

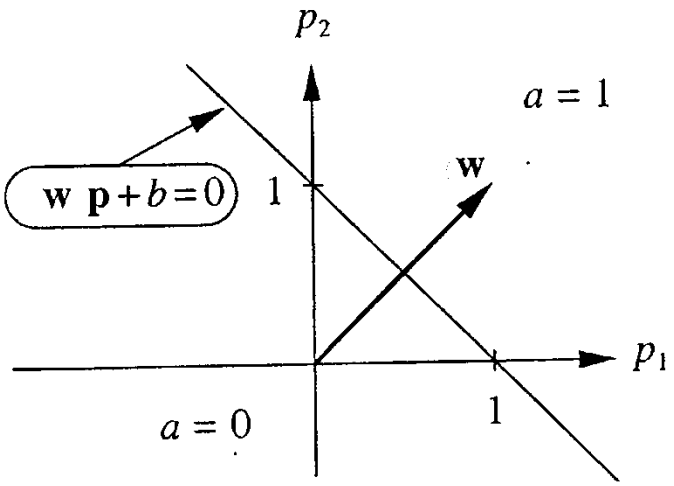
- **It is a line in the input space**
- **It separates the input space into 2 subspaces: output =1 and 0**

- **5. Draw the decision boundary:**

$p_1 = 0 \Rightarrow p_2 = 1;$ (p_2 intersect)
$p_2 = 0 \Rightarrow p_1 = 1;$ (p_1 intersect)



Decision Boundary (2)



- **6. Find the side corresponding to an output of 1:**

$$\mathbf{p} = \begin{bmatrix} 2 \\ 0 \end{bmatrix}$$

$$a = step(\mathbf{wp} + b) =$$

$$= step([11]\begin{bmatrix} 2 \\ 0 \end{bmatrix} - 1) = 1$$

- **Properties of the decision boundary – it is orthogonal to w**

- **=>The decision boundary is defined by the weight vector (if we know the weight vector, we know the decision boundary)**

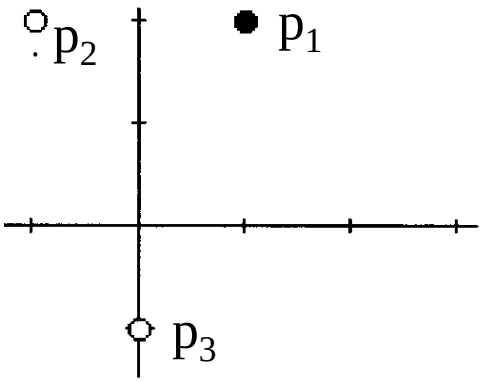
- **Most important conclusion: the decision boundary of a perceptron is a line**

Perceptron Learning Rule – Derivation

$\left\{\mathbf{p}_1 = \begin{bmatrix}1\\2\end{bmatrix}, t_1 = 1\right\}$ $\left\{\mathbf{p}_2 = \begin{bmatrix}-1\\2\end{bmatrix}, t_2 = 0\right\}$ $\left\{\mathbf{p}_3 = \begin{bmatrix}0\\-1\end{bmatrix}, t_3 = 0\right\}$

- **Supervised learning:**
  - **Given: a set of 3 training examples from 2 classes: 1 and 0**
  - **Goal: learn to classify these examples correctly with a perceptron (i.e. learn to associate: input p_i with output t_i)**

- **Idea: Start with a random set of weights (i.e. random initial decision boundary); feed each example, iteratively adjust the weights (i.e. adjust the decision boundary) until the examples are correctly classified, i.e. class 0 and 1 are separated**



• Is it possible to solve this problem? We know that a perceptron forms a linear decision boundary; can we separate the examples with a line?
• How many lines can we draw to separate the examples?
• How many inputs and outputs for our perceptron?

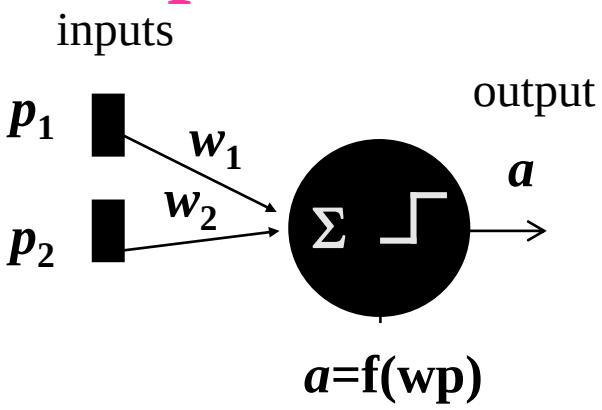
Example - Staring Point and First Input Vector

1) Initialization of the weights

- **1 neuron with 2 inputs; suppose that there is no bias weight**
- **Let our random initial weight vector be:**

$$\mathbf{w} = \begin{bmatrix} 1 & -0.8 \end{bmatrix}$$

- **It defines our initial classification boundary**

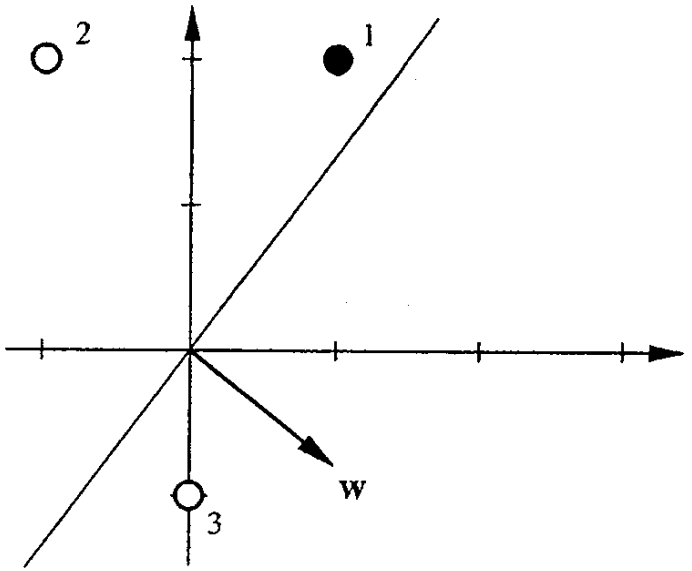


2) Start training: feed the input examples to the perceptron iteratively (1-by-1), calculate the output and adjust w until all 3 examples are correctly classified

- **First input example:**

$$\left\{ \mathbf{p}_1 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, t_1 = 1 \right\}$$

$$a = step(\mathbf{w}\mathbf{p}_1) = step([1 \quad -0.8]\begin{bmatrix} 1 \\ 2 \end{bmatrix}) = step(-0.6) = 0$$



Incorrect classification!

Output should be 1 but it is 0!

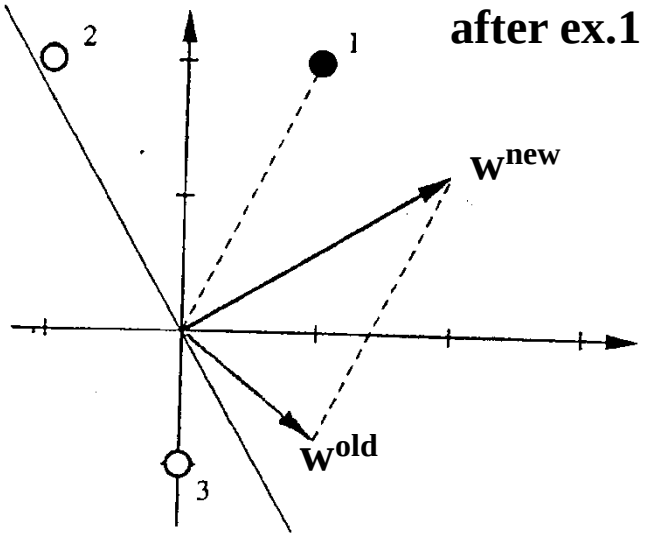
Tentative Learning Rule

- **We need to alter the weight vector so that it points more toward $\mathbf{p}_1$, so that in the future it has a better chance of classifying it correctly**
- **=> Let's add $\mathbf{p}_1$ to $\mathbf{w}$ - repeated presentations of $\mathbf{p}_1$ would cause the direction of $\mathbf{w}$ to approach the direction of $\mathbf{p}_1$**
- **=> Tentative learning rule (rule 1):**

If $t = 1$ and $a = 0$, then $\mathbf{w}^{new} = \mathbf{w}^{old} + \mathbf{p}^{T}$

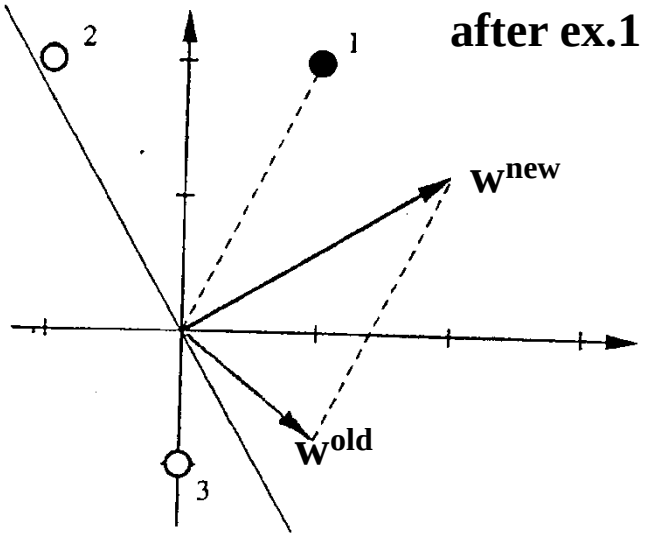
- **Applying the rule:**

$$\mathbf{w}^{new} = \mathbf{w}^{old} + p_1^T = \begin{bmatrix}1 & -0.8\end{bmatrix} + \begin{bmatrix}1 & 2\end{bmatrix} = \begin{bmatrix}2 & 1.2\end{bmatrix}$$



Second Input Vector

$$\left\{\mathbf{p}_2 = \begin{bmatrix}-1\\2\end{bmatrix}, t_2 = 0\right\}$$

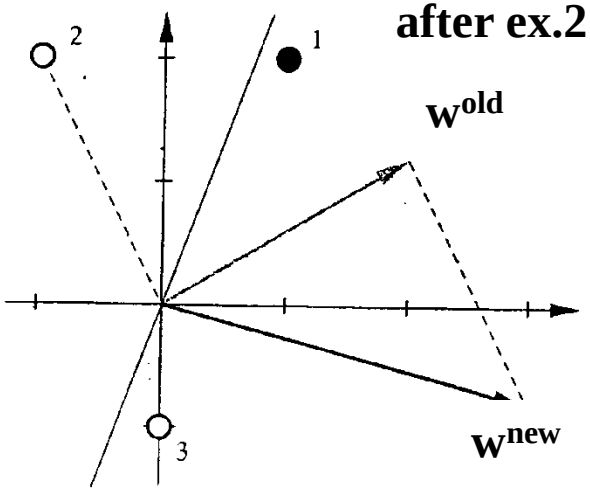


$$a = step(\mathbf{wp}_2) = step([2 \quad 1.2]\begin{bmatrix}-1\\2\end{bmatrix}) = step(0.4) = 1$$

Incorrect classification!
Output should be 0 but it is 1!

- **We can move the weight vector w away from the input (i.e. subtract it) => rule 2:**

If $t = 0$ and $a = 1$, then $\mathbf{w}^{new} = \mathbf{w}^{old} - \mathbf{p}^{T}$



- **Applying the rule:**

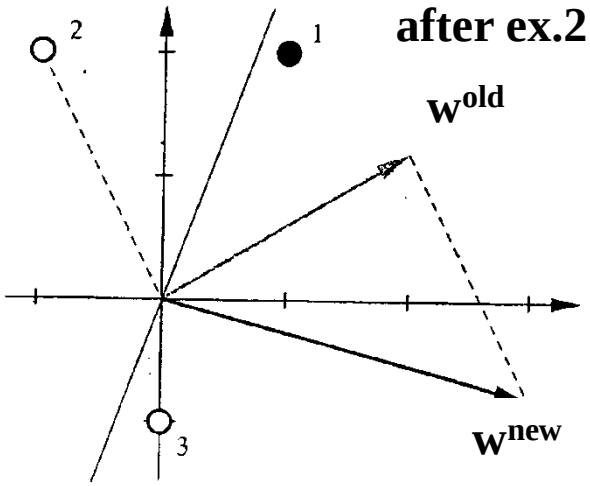
$$\mathbf{w}^{new} = \mathbf{w}^{old} - p_2^T = [2 \quad 1.2] - [-1 \quad 2] = [3 \quad -0.8]$$

Third Input Vector

$$\left\{ \mathbf{p}_3 = \begin{bmatrix} 0 \\ -1 \end{bmatrix}, t_3 = 0 \right\}$$

$$a = step(\mathbf{w}\mathbf{p}_3) = step([3 \quad -0.8]\begin{bmatrix} 0 \\ -1 \end{bmatrix}) = step(0.8) = 1$$

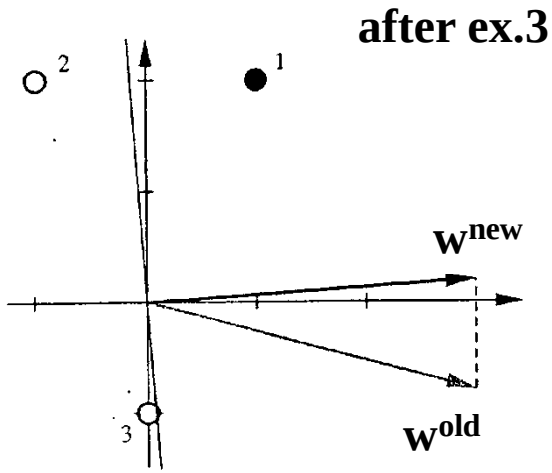
Incorrect classification!
Output should be 0 but it is 1!



- **We already have a rule for this case – apply rule 2:**

If $t = 0$ and $a = 1$, then $\mathbf{w}^{new} = \mathbf{w}^{old} - \mathbf{p}^T$

$$\mathbf{w}^{new} = \mathbf{w}^{old} - p_3^T = [3 \quad -0.8] - [0 \quad -1] = [3 \quad 0.2]$$



- **All examples have been fed once - we say that 1 *epoch* has been completed**
- **Check how each example is classified by the current classifier**
  - **all are correctly classified => stop**
  - **otherwise => repeat**

Unified Learning Rule

- **Covers all combinations of output and target values (0 and 1):**

Rule 1 If $t = 1$ and $a = 0$, then $\mathbf{w}^{new} = \mathbf{w}^{old} + \mathbf{p}^T$

Rule 2 If $t = 0$ and $a = 1$, then $\mathbf{w}^{new} = \mathbf{w}^{old} - \mathbf{p}^T$

Rule 3 If $t = a$, then $\mathbf{w}^{new} = \mathbf{w}^{old}$

- **define:** $e = t - a$

If $e = 1$, then $\mathbf{w}^{new} = \mathbf{w}^{old} + \mathbf{p}^T$

If $e = -1$, then $\mathbf{w}^{new} = \mathbf{w}^{old} - \mathbf{p}^T$

If $e = 0$, then $\mathbf{w}^{new} = \mathbf{w}^{old}$

- **unified rule:**

$$\mathbf{w}^{new} = \mathbf{w}^{old} + e\mathbf{p}^T$$

$$\mathbf{b}^{new} = \mathbf{b}^{old} + e$$

Perceptron Learning Law – Summary

1. **Initialize weights (including biases) to small random values, set *epoch=1*.**
2. **Choose a random example (input-target output pair {p,t}) from the training set**
3. **Calculate the actual output of the network for this example a (also called network activation)**
4. **Compute the output error e=t-a**
5. **Update the weights:**

$$\mathbf{w}^{new} = \mathbf{w}^{old} + e\mathbf{p}^{T}$$
$$\mathbf{b}^{new} = \mathbf{b}^{old} + e$$

6. **Repeat steps 2-5 (by choosing another example from the training data)**
7. **At the end of each *epoch* check if the stopping criterion is satisfied: all examples are correctly classified or a maximum number of epochs is reached; if yes - stop, otherwise *epoch++* and repeat from 2.**

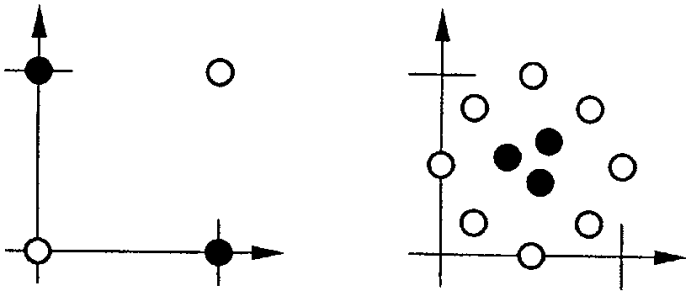
Perceptron – Stopping Criterion

- **Stopping criterion is checked at *the end of each epoch*:**
- **Epoch (= training epoch) - one pass through the training set (i.e. each training example is passed, the perceptron output is computed and the weights are changed, then the next example is passed etc.)**
- **The epoch numbering starts from 1: epoch 1, epoch 2, etc.**
- **To check if all examples are correctly classified at the end of the epoch:**
  - **All training examples are passed again, the actual output is calculated and compared with the target output. There is no weight change.**
  - **Note: this does not count for another epoch as there is no weight change.**

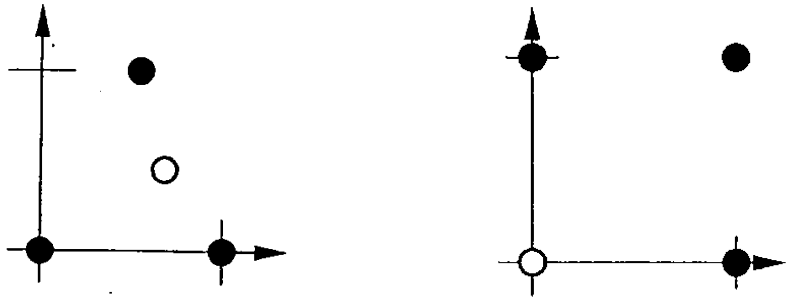
Capability and Limitations

- **The output values of a perceptron can take only 2 values:**
  - **0 and 1 (or –1 and 1)**
- **Theorem: If the training examples are *linearly separable*, the perceptron learning rule is guaranteed to converge to a solution (i.e. a set of weights that correctly classify the training examples) in a finite number of steps**
  - **When the examples are linearly separable, the perceptron will find a line (hyperplane) that separates the two classes**
  - **It doesn't try to find an "optimal" line (e.g. a line that is in the middle of the positive and negative examples), it will simply stop when a solution (a separating line) is found**

Linearly inseparable:



Linearly separable:



Perceptron Learning Rule - Example

- **Given is the following training data:**

Ex#	input	output
1	1 0	1
2	0 1	0
3	1 1	1

- **Train a perceptron without a bias on this data. Stopping criterion: all examples are correctly classified or a maximum number of 3 epochs is reached. Assume that all initial weights are 0.**

Solution

- **Perceptron learning rule:**

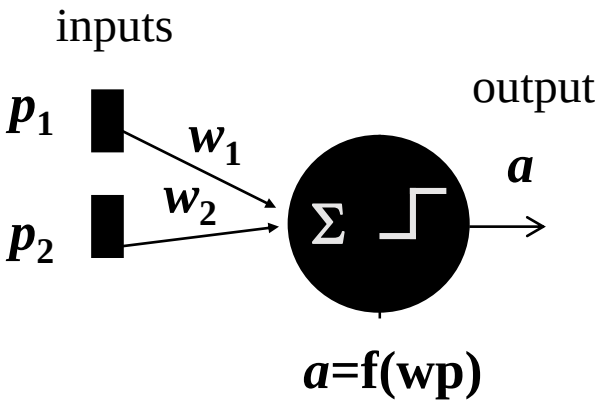
$$\mathbf{w}^{new} = \mathbf{w}^{old} + e\mathbf{p}^T$$
$$e = t - a$$

Ex#	input	output
1	1 0	1
2	0 1	0
3	1 1	1

- **How many inputs?**
- **How many outputs?**

- **Starting point: w=[0 0]**
  Apply ex.1 [1 0], t=1
  a=step([0 0][1 0])=step(0)=1, correct, no weight change

- **Apply ex.2 [0 1], t=0**
  a=step([0 0][0 1])=step(0)=1, incorrect
  w_new=[0 0]+(0-1)[0 1]=[0 0]-[0 1]=[0 -1]



- **Apply ex.3 [1 1], t=1**
  a=step([0-1][1 1])=step(-1)=0, incorrect
  w_new=[0 -1]+(1-0)[1 1]=[0 -1]+[1 1]=[1 0]

End of epoch 1

Solution – cont.

- **Check if the stopping criterion is satisfied**
- **1) All training examples are correctly classified?**

w=[1 0] //current weight vector

Apply ex.1 [1 0], t=1

a=step([1 0][1 0])=step(1)=1, correct

Apply ex.2 [0 1], t=0

a=step([1 0][0 1])=step(0)=1, incorrect

=> condition not satisfied

2) Epoch=3 reached? No

⇒ **Stopping criterion is not satisfied => keep training**

Ex#	input	output
1	1 0	1
2	0 1	0
3	1 1	1

Start epoch 2:

training: …

End epoch 2: check stopping criterion

….

Read at home

Another Example – Apple/Banana Sorter

- **A farmer has a warehouse that stores a variety of fruits. He wants to sort automatically the different types of fruit.**
- **There is a conveyer belt on which the fruit is loaded. It is then passed through a set of sensors, which measure 3 properties of the fruit: shape, texture and weight.**
  - **shape sensor: -1 if the fruit is round, 1 - if it is more elliptical**
  - **texture sensor: -1 if the surface is smooth, 1 - if it is rough**
  - **weight sensor: -1 if the fruit is > 500g , 1 - if < 500g**
- **The sensor outputs are inputs to a NN**
- **NN's purpose:**
  Recognise the fruit type, so that fruits are directed to the correct bin
- **For simplicity, only 2 types of fruits:**
  apples and bananas

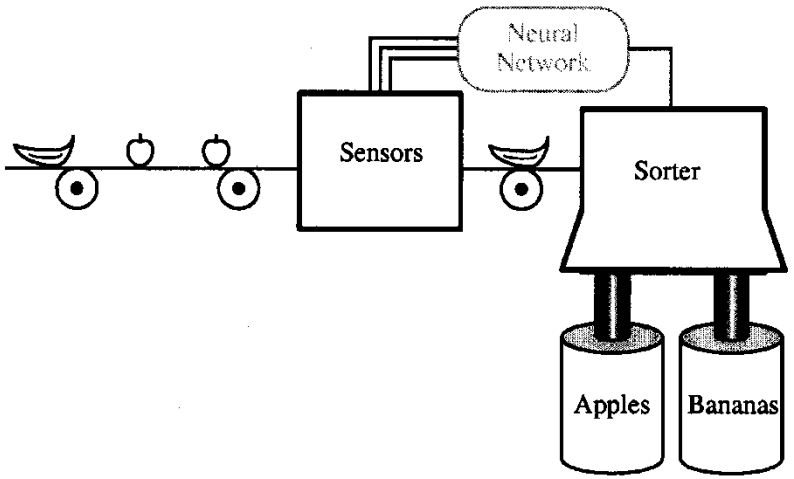


Image ref: Hagan, Demuth, Beale, Neural Network Design, 1996, PWS, Thomson

Read at home

Apple/Banana Example

- **How many inputs?**
- **How many outputs?**
- **How many perceptrons?**
- **What are the input vectors?**

Training set

$$\left\{\mathbf{p}_1 = \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix}, \quad t_1 = 1\right\} - banana \qquad \left\{\mathbf{p}_2 = \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix}, \quad t_2 = 0\right\} - apple$$

Initial weights (random):

$$\mathbf{w} = [0.5 \quad -1 \quad -0.5], \quad b = 0.5$$

Read at home

Apple/Banana Example – Iteration 1 and 2

1

Applying p_1 :

$$a = step(\mathbf{w}\mathbf{p}_1 + b) = step\left(\begin{bmatrix} 0.5 & -1 & -0.5 \end{bmatrix} \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix} + 0.5 \right) = step(-0.5) = 0$$

$$e = t_1 - a = 1 - 0 = 1$$

Updating the weights:

$$\mathbf{w}^{new} = \mathbf{w}^{old} + e\mathbf{p}^T = \begin{bmatrix} 0.5 & -1 & -0.5 \end{bmatrix} + (1)\begin{bmatrix} -1 & 1 & -1 \end{bmatrix} = \begin{bmatrix} -0.5 & 0 & -1.5 \end{bmatrix}$$

$$b^{new} = b^{old} + e = 0.5 + 1 = 1.5$$

2

Applying p_2 :

$$a = step(\mathbf{w}\mathbf{p}_2 + b) = step\left(\begin{bmatrix} -0.5 & 0 & -1.5 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix} + 1.5 \right) = step(2.5) = 1$$

$$e = t_2 - a = 0 - 1 = -1$$

Updating the weights:

$$\mathbf{w}^{new} = \mathbf{w}^{old} + e\mathbf{p}^T = \begin{bmatrix} -0.5 & 0 & -1.5 \end{bmatrix} + (-1)\begin{bmatrix} 1 & 1 & -1 \end{bmatrix} = \begin{bmatrix} -1.5 & -1 & -0.5 \end{bmatrix}$$

$$b^{new} = b^{old} + e = 1.5 + (-1) = 0.5$$

Read at home

Apple/Banana Ex. – Stopping Criterion Check

- **End of epoch; check if the stopping criteria are satisfied:**

p_1 :

$$a = step(\mathbf{w}\mathbf{p}_1 + b) = step\left(\begin{bmatrix} -1.5 & -1 & -0.5 \end{bmatrix} \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix} + 0.5 \right) = step(1.5) = 1 = t_1$$

p2 :

$$a = step(\mathbf{w}\mathbf{p}_2 + b) = step\left(\begin{bmatrix} -1.5 & -1 & -0.5 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix} + 0.5 \right) = step(-1.5) = 0 = t_2$$

- **How many epochs were necessary to train the perceptron?**

Perceptrons - History

- **1943 - Warren McCulloch and Walter Pitts introduced the first artificial neuron**
  - **Weighted sum of input signals is compared to a threshold to determine the output; when the sum is greater or equal to the threshold, the output is 1; else - 0**
  - **Showed that these neurons could compute any Boolean function**
    - **E.g. basic Boolean functions can be represented with appropriate weights and biases: AND: w1=1, w2=1, b=1.5; OR: w1=1, w2=1, t=0.5; NOT: w=-1, b=-0.5**
    - **These neurons can be used to build a NN to compute any Boolean function**
  - **Unlike biological neurons, the parameters of the network had to be designed, as no training method was available**
- **The connection between biology and digital computers generated a lot of interest!**

Perceptrons – History (cont.1)

- **1958 - Frank Rosenblatt developed the perceptrons**
  - **The neurons in them are similar to those of McCulloch and Pitts**
  - **Key contribution: introduction of learning rule for training perceptrons to solve pattern recognition problems**
  - **Proved that the rule will always converge to correct weights, if such weights exist**
  - **Learning: simple and automatic**
  - **Perceptrons showed great success for such a simple model**
  - **Could even learn when initialized with random weights ...**

- **Generated a lot of interest in neural network research!**

Perceptrons – History (cont.2)

- **1969 - Marvin Minsky and Seymour Papert - book "Perceptrons"**
  - **Widely publicized the limitations of the perceptrons**
  - **Demonstrated that the perceptrons were not capable of implementing certain elementary functions, e.g. XOR**
  - **Provided detailed analysis of the capabilities and limitations of perceptrons**
  - **Rosenblatt, Widrow and Hoff were aware of these limitations. To overcome them they proposed a new type of network but they were not able to adapt the percepron's learning rule to train this more complex network.**
- **The majority of scientific community walked away from the filed of neural networks...**

Acknowledgements

- **Hagan, Demuth, Beale, *Neural Network Design*, 1996, PWS, Thomson**
- **Anderson, *Introduction to Neural Networks*, MIT Press, 1995**
- **Beale and Jackson, *Neural Computing: An Introduction*, CRC Press, 1990.**