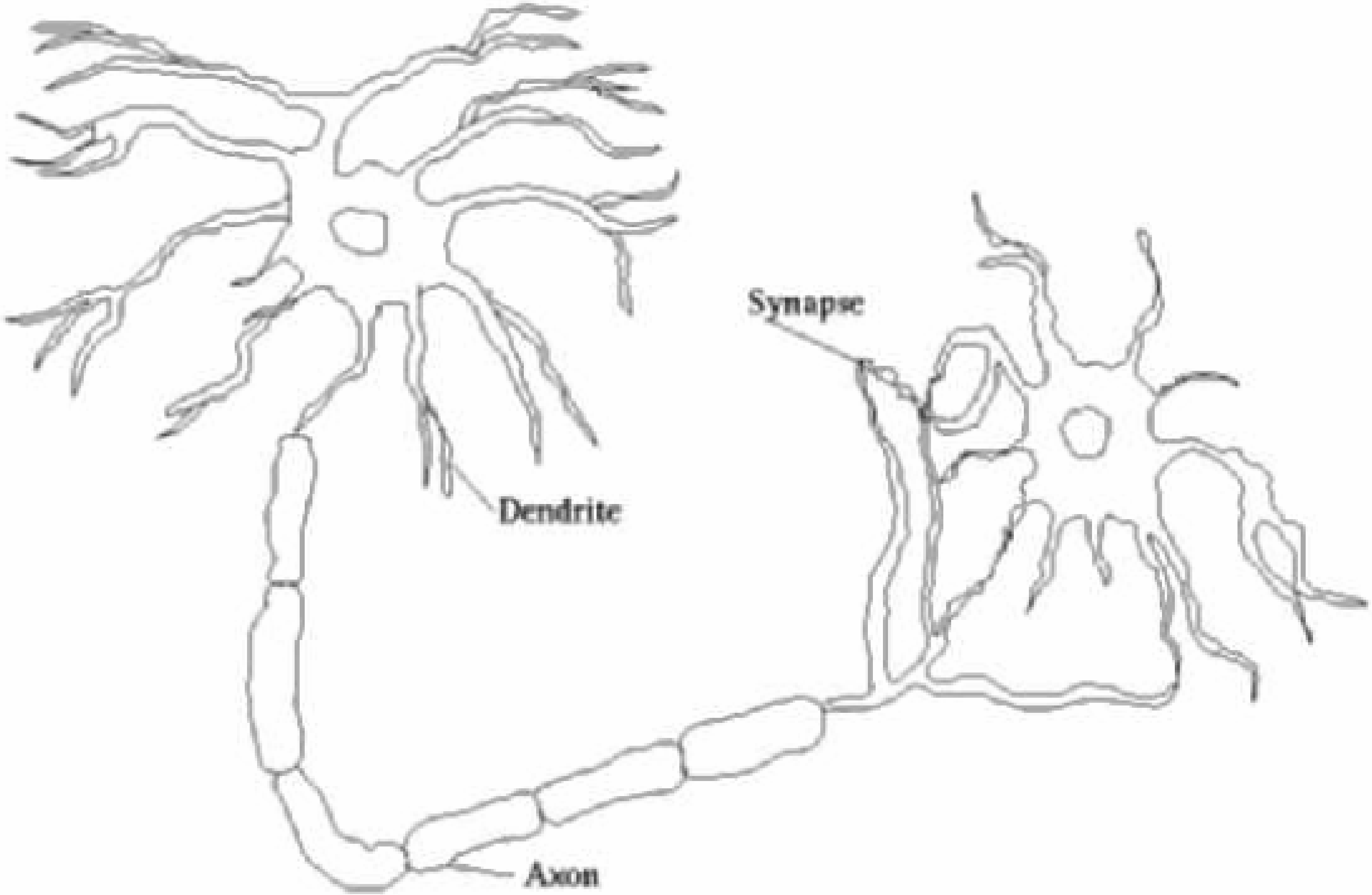
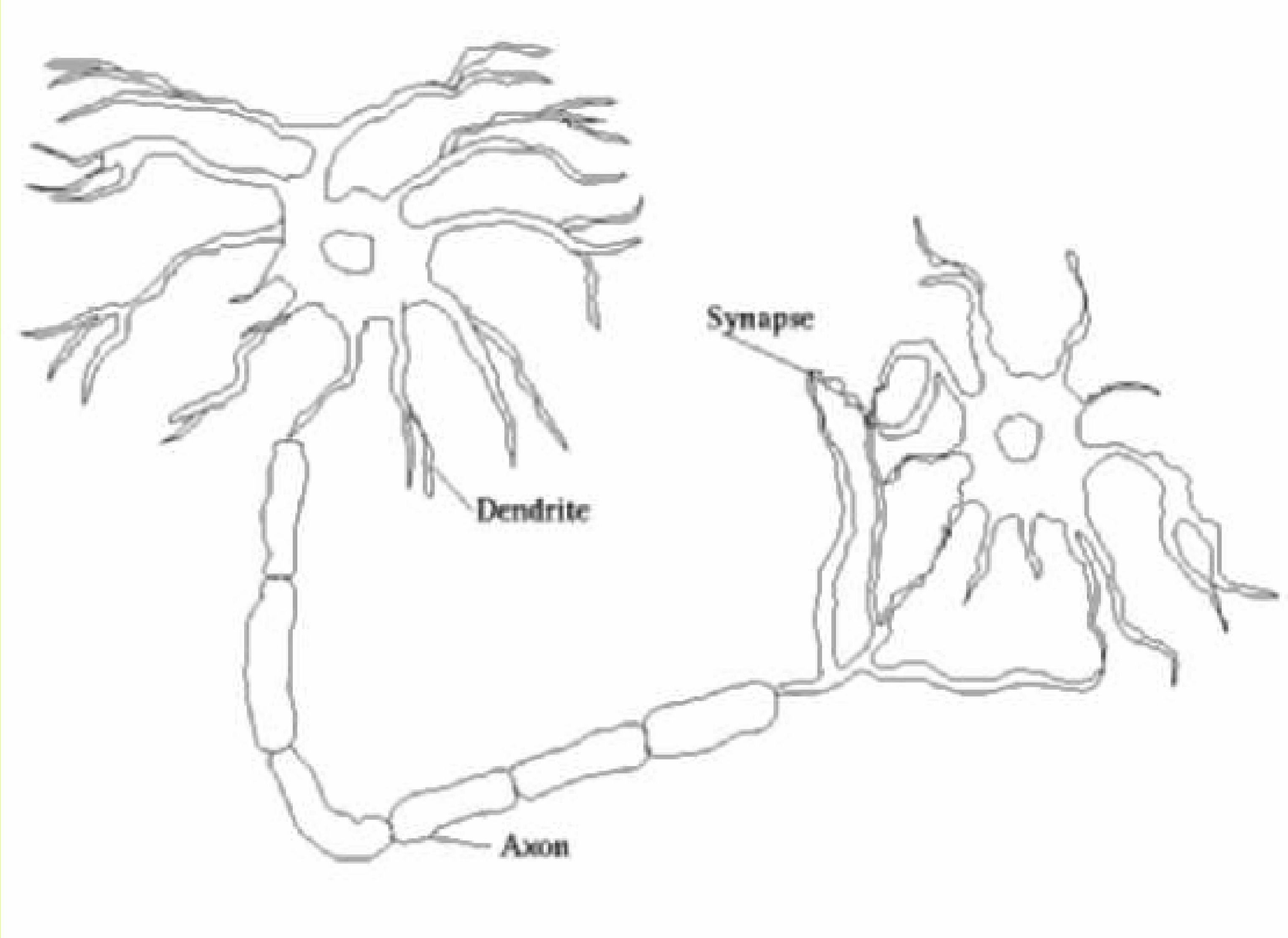
CS623: Introduction to Computing with Neural Nets
(lecture-2)

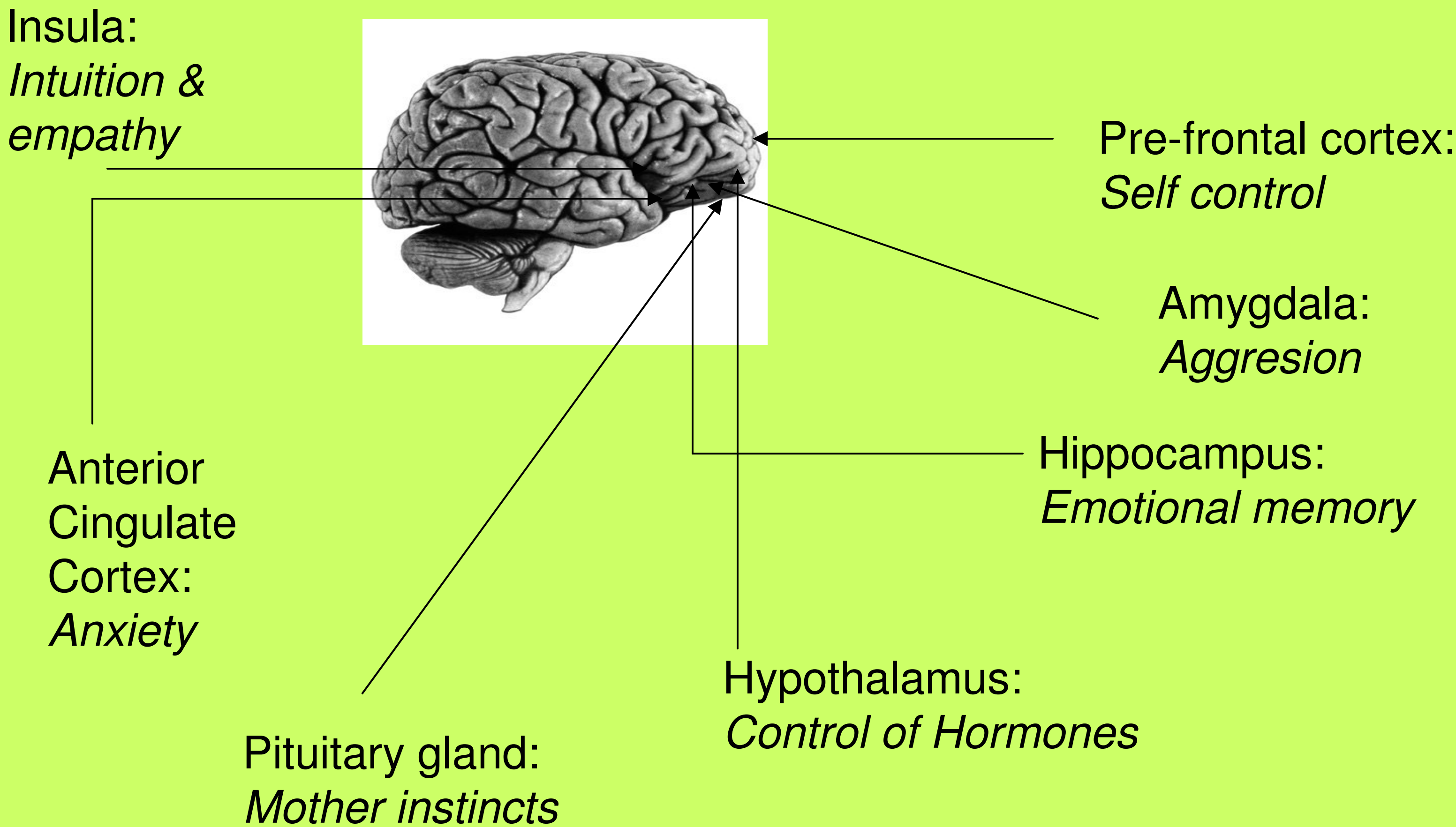
Pushpak Bhattacharyya
Computer Science and Engineering Department
IIT Bombay

Recapitulation

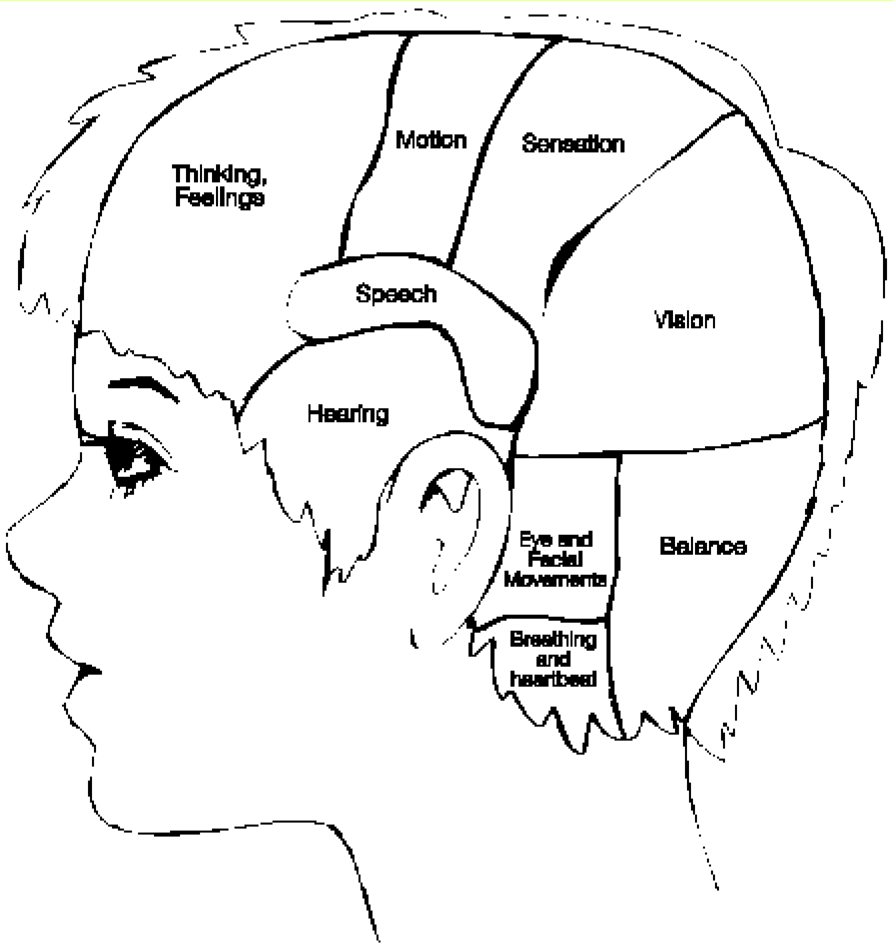

Synapse
Dendrite
Axon


Synapse
Dendrite
Axon

The human brain

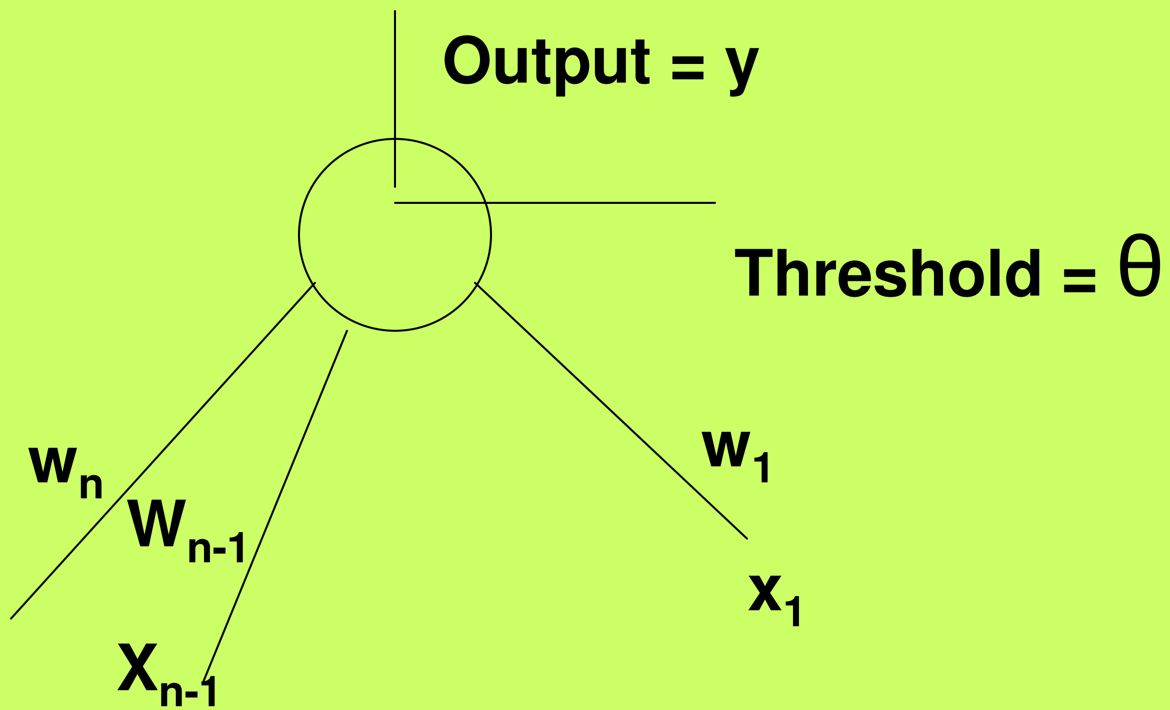


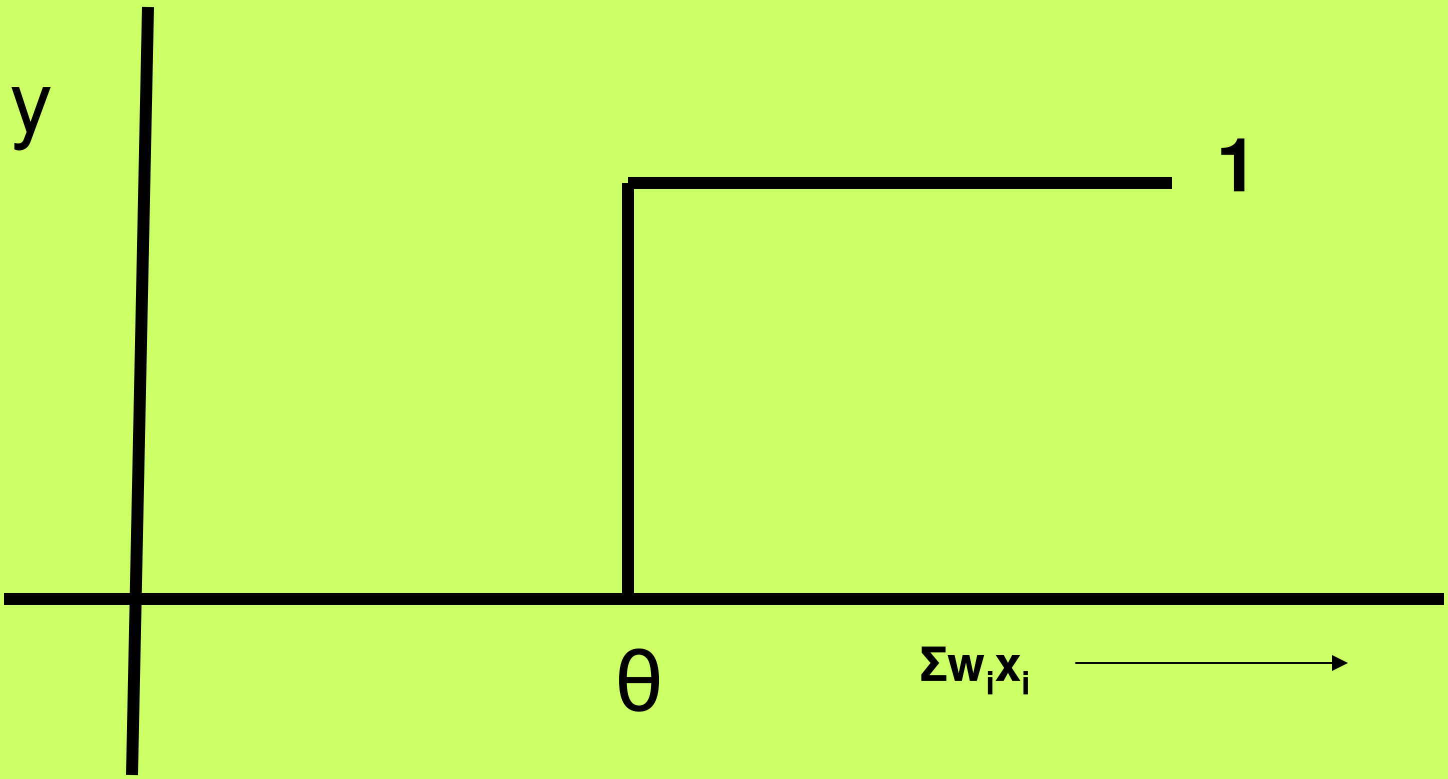
Functional Map of the Brain



The Perceptron Model

A perceptron is a computing element with input lines having associated weights and the cell having a threshold value. The perceptron model is motivated by the biological neuron.





Step function / Threshold function

y = 1 for $\Sigma w_i x_i >= \theta$

=0 otherwise

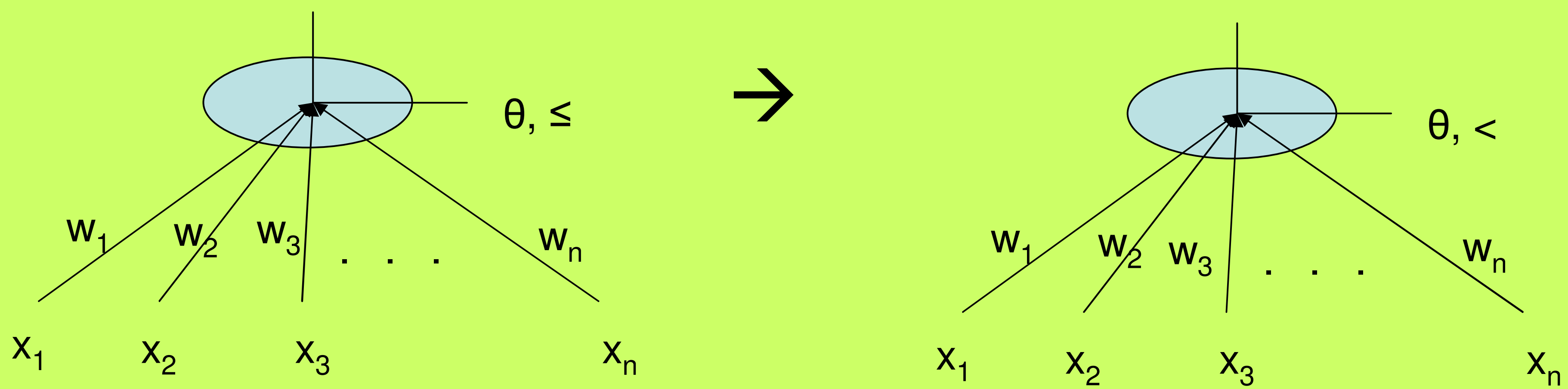
Perceptron Training Algorithm (PTA)

Preprocessing:

1. The computation law is modified to

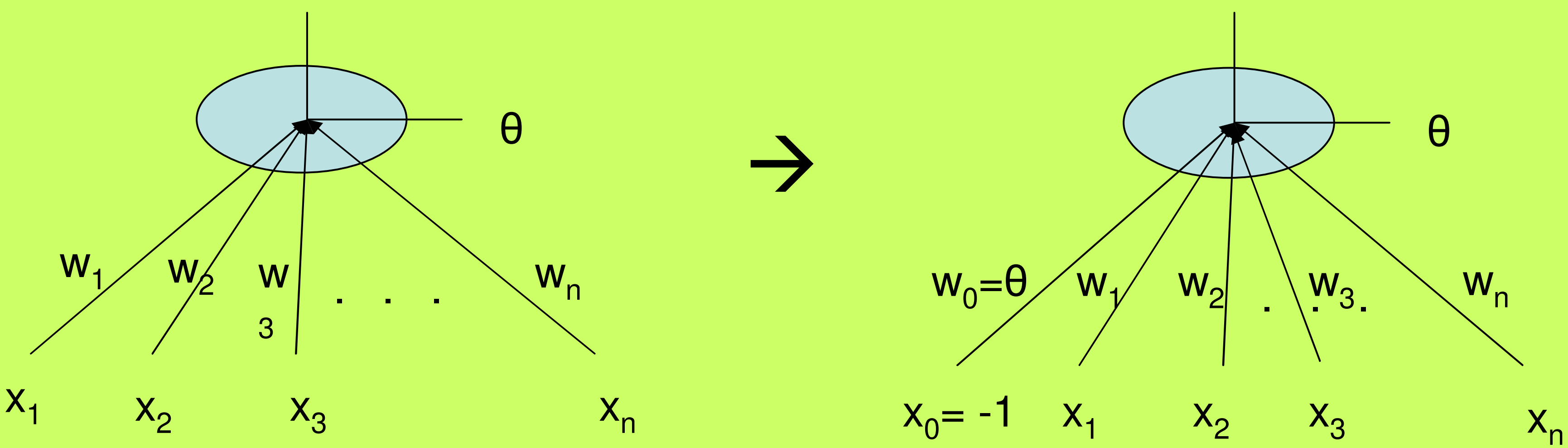
$$y = 1 \text{ if } \sum w_i x_i > \theta$$

$$y = o \text{ if } \sum w_i x_i < \theta$$



PTA – preprocessing cont…

2. Absorb θ as a weight



3. Negate all the zero-class examples

Example to demonstrate preprocessing

- **OR perceptron**

1-class <1,1> , <1,0> , <0,1>

0-class <0,0>

Augmented x vectors:-

1-class <-1,1,1> , <-1,1,0> , <-1,0,1>

0-class <-1,0,0>

Negate 0-class:- <1,0,0>

Example to demonstrate preprocessing cont..

Now the vectors are

	x_0	x_1	x_2
X_1	-1	0	1
X_2	-1	1	0
X_3	-1	1	1
X_4	1	0	0

Perceptron Training Algorithm

1. Start with a random value of w

   ex: <0,0,0…>
2. Test for $wx_i > 0$

   If the test succeeds for i=1,2,…n

   then return w
3. Modify w, $w_{next} = w_{prev} + x_{fail}$

Tracing PTA on OR-example

w=<0,0,0>	wx_1 fails
w=<-1,0,1>	wx_4 fails
w=<0,0,1>	wx_2 fails
w=<-1,1,1>	wx_1 fails
w=<0,1,2>	wx_4 fails
w=<1,1,2>	wx_2 fails
w=<0,2,2>	wx_4 fails
w=<1,2,2>	success

Theorems on PTA

1. The process will terminate
2. The order of selection of x_i for testing and w_{next} does not matter.

End of recap

Proof of Convergence of PTA

- Perceptron Training Algorithm (PTA)

- Statement:

  Whatever be the initial choice of weights and whatever be the vector chosen for testing, PTA converges if the vectors are from a linearly separable function.

Proof of Convergence of PTA

- Suppose w_n is the weight vector at the n^{th} step of the algorithm.
- At the beginning, the weight vector is w_0
- Go from w_i to w_{i+1} when a vector X_j fails the test $w_i X_j > 0$ and update w_i as

  $$w_{i+1} = w_i + X_j$$

- Since Xjs form a linearly separable function,

  $$\exists\ w^* \text{ s.t. } w^* X_j > 0\ \forall j$$

Proof of Convergence of PTA

- Consider the expression

  $$G(w_n) = \frac{w_n \,.\, w^*}{|\, w_n|}$$

  where w_n = weight at nth iteration
- $$G(w_n) = \frac{|w_n| \,.\, |w^*| \,.\, \cos\theta}{|w_n|}$$

  where θ = angle between w_n and w^*
- $G(w_n) = |w^*| \,.\, \cos\theta$
- $G(w_n) \leq |w^*|$ (as $-1 \leq \cos\theta \leq 1$)

Behavior of Numerator of G

$w_n \,.\, w^* = (w_{n-1} + X^{n-1}_{fail}) \,.\, w^*$

$= w_{n-1} \,.\, w^* + X^{n-1}_{fail} \,.\, w^*$

$= (w_{n-2} + X^{n-2}_{fail}) \,.\, w^* + X^{n-1}_{fail} \,.\, w^* \;\ldots\ldots$

$= w_0 \,.\, w^* + (X^0_{fail} + X^1_{fail} + \ldots + X^{n-1}_{fail}) \,.\, w^*$

$w^*.X^i_{fail}$ is always positive: note carefully

- Suppose $|X_j| \geq \delta$, where δ is the minimum magnitude.
- Num of G $\geq |w_0 \,.\, w^*| + n\, \delta \,.\, |w^*|$
- So, numerator of G grows with n.

Behavior of Denominator of G

- $|w_n| = \sqrt{w_n \, . \, w_n}$

$= \sqrt{(w_{n-1} + X^{n-1}_{fail})^2}$

$= \sqrt{(w_{n-1})^2 + 2. \, w_{n-1.} X^{n-1}_{fail} + (X^{n-1}_{fail})^2}$

$\leq \sqrt{(w_{n-1})^2 + (X^{n-1}_{fail})^2}$ (as $w_{n-1.} X^{n-1}_{fail} \leq 0$)

$\leq \sqrt{(w_0)^2 + (X^0_{fail})^2 + (X^1_{fail})^2 + \ldots. + (X^{n-1}_{fail})^2}$

- $|X_j| \leq \rho$ (max magnitude)
- So, Denom $\leq \sqrt{(w_0)^2 + n\rho^2}$

Some Observations

- Numerator of G grows as n
- Denominator of G grows as $\sqrt{n}$

  => Numerator grows faster than denominator
- If PTA does not terminate, $G(w_n)$ values will become unbounded.

Some Observations contd.

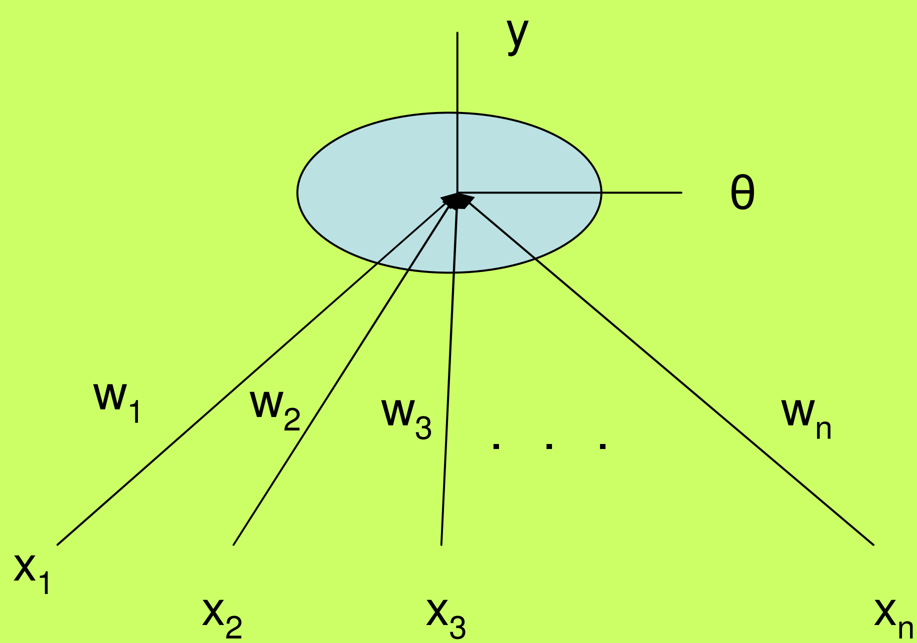
- But, as $|G(w_n)| \leq |w^*|$ which is finite, this is impossible!
- Hence, PTA has to converge.
- Proof is due to Marvin Minsky.

Convergence of PTA

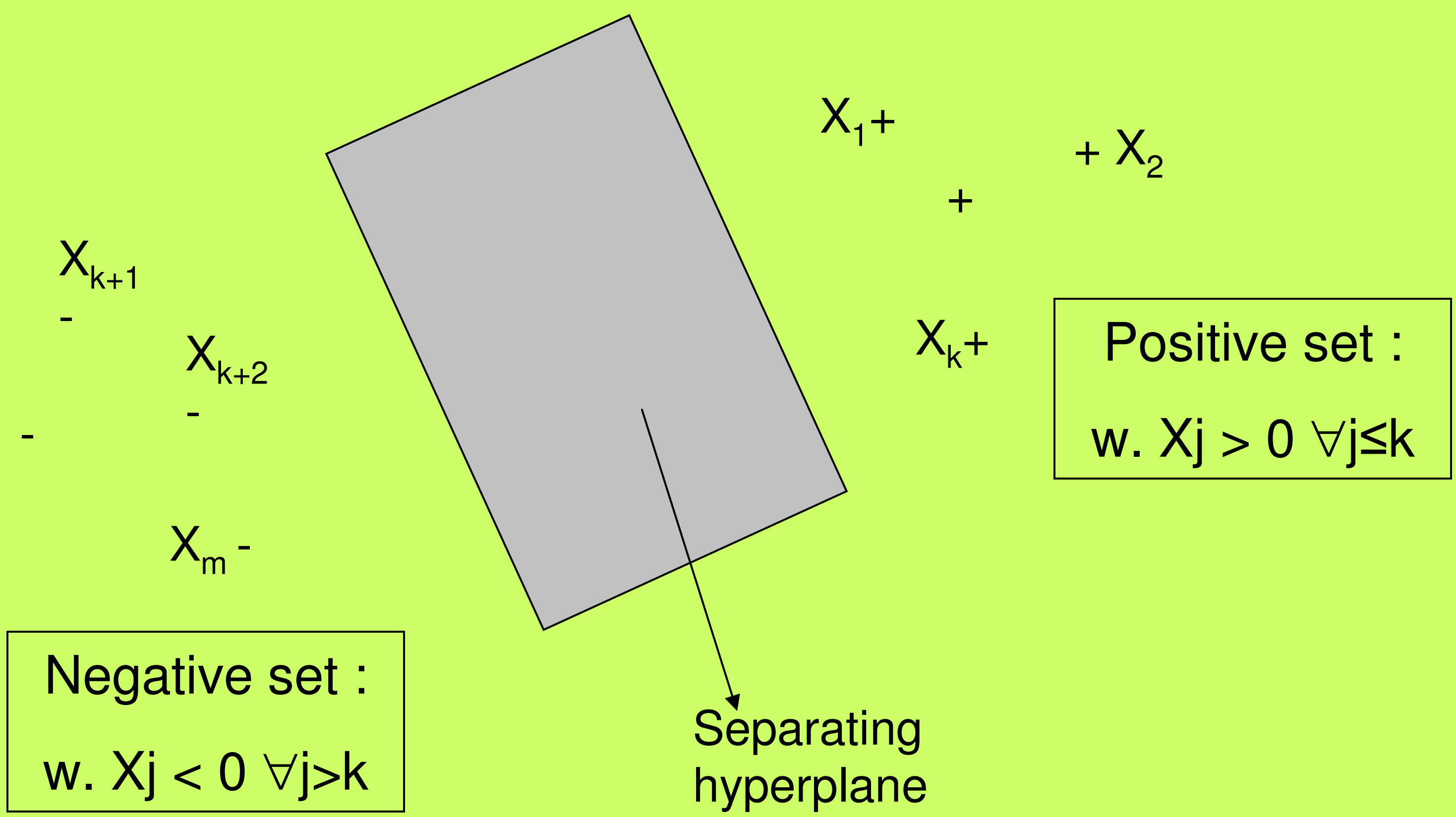
- *Whatever be the initial choice of weights and whatever be the vector chosen for testing, PTA converges if the vectors are from a linearly separable function.*

Study of Linear Separability

- $W. X_j = 0$ defines a hyperplane in the (n+1) dimension.
  - => W vector and X_j vectors are perpendicular to each other.



Linear Separability



Test for Linear Separability (LS)

- Theorem:

  A function is linearly separable iff the vectors corresponding to the function do not have a Positive Linear Combination (PLC)

- PLC – Both a necessary and sufficient condition.

- $X_1, X_2, \ldots, X_m$ - Vectors of the function
- $Y_1, Y_2, \ldots, Y_m$ - Augmented negated set

- Prepending -1 to the 0-class vector X_i and negating it, gives Y_i

Example (1) - XNOR

- The set $\{Y_i\}$ has a PLC if $\sum P_i Y_i = 0$,

  $1 \leq i \leq m$
  - where each P_i is a non-negative scalar and
  - atleast one $P_i > 0$
- Example : 2 bit even-parity (X-NOR function)

X_1	<0,0> +	Y_1	<-1,0,0>
X_2	<0,1> -	Y_2	<1,0,-1>
X_3	<1,0> -	Y_3	<1,-1,0>
X_4	<1,1> +	Y_4	<-1,1,1>

Example (1) - XNOR

- $P_1 [\,-1\ 0\ 0\,]^T + P_2 [\,1\ 0\ -1\,]^T$
  $+ P_3 [\,1\ -1\ 0\,]^T + P_4 [\,-1\ 1\ 1\,]^T$
  $= [\,0\ 0\ 0\,]^T$
- All $P_i = 1$ gives the result.
- For Parity function,
  PLC exists => Not linearly separable.

Example (2) – Majority function

- 3-bit majority function

X_i	Y_i
<0 0 0> -	<1 0 0 0>
<0 0 1> -	<1 0 0 -1>
<0 1 0> -	<1 0 -1 0>
<0 1 1> +	<-1 0 1 1>
<1 0 0> -	<1 -1 0 0>
<1 0 1> +	<-1 1 0 1>
<1 1 0> +	<-1 1 1 0>
<1 1 1> +	<-1 1 1 1>

- Suppose PLC exists. Equations obtained are:
- $P_1 + P_2 + P_3 - P_4 + P_5 - P_6 - P_7 - P_8 = 0$
- $-P_5 + P_6 + P_7 + P_8 = 0$
- $-P_3 + P_4 + P_7 + P_8 = 0$
- $-P_2 + P_4 + P_6 + P_8 = 0$
- On solving, all P_i will be forced to 0
- 3 bit majority function

  => No PLC => LS

Limitations of perceptron

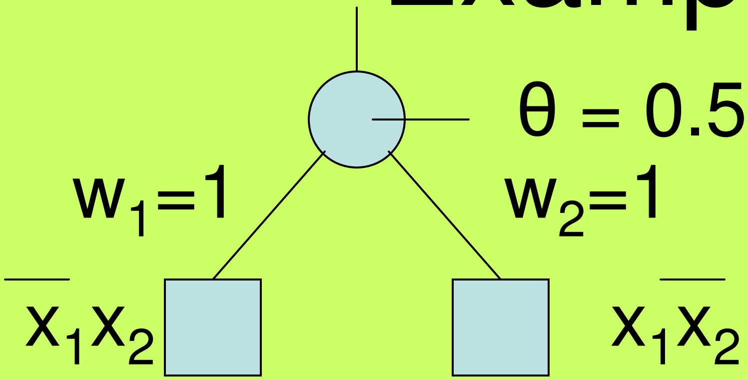
- Non-linear separability is all pervading
- Single perceptron does not have enough computing power
- Eg: XOR cannot be computed by perceptron

Solutions

- Tolerate error (Ex: *pocket algorithm* used by connectionist expert systems).
  - Try to get the best possible hyperplane using only perceptrons
- Use higher dimension surfaces

  Ex: Degree - 2 surfaces like parabola
- Use layered network

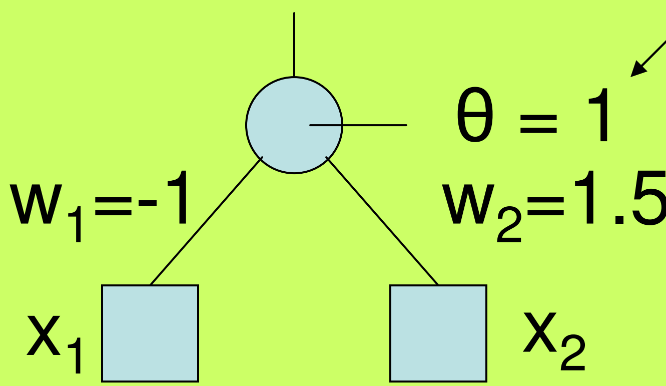
Example - XOR



← Calculation of XOR

x_1	x_2	$x_1\overline{x_2}$
0	0	0
0	1	1
1	0	0
1	1	0

Calculation of $x_1\overline{x_2}$



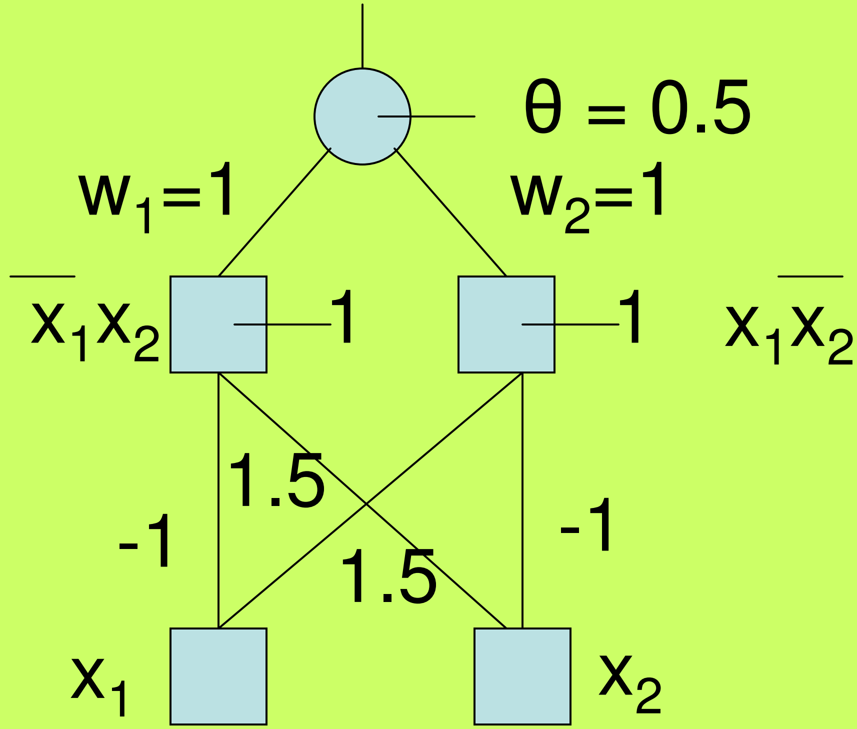
$0 < \Theta$

$w2 \geq \Theta$

$w1 < \Theta$

$w1 + w2 < \Theta$

Example - XOR



Multi Layer Perceptron (MLP)

- Question:- How to find weights for the hidden layers when no target output is available?

- Credit assignment problem – to be solved by "*Gradient Descent*"

Assignment

- Fine by solving linear inequalities the perceptron to solve the majority Boolean Function.
- Then feed it to your implementation of the perceptron training algorithm and study its behaviour.
- Download SNNS and WEKA packages and try your hand at the perceptron algorithm built in the packages.