



DS 310 Machine Learning Linear Classifiers: Perceptrons

Vasant G. Honavar

Dorothy Foehr Huck and J. Lloyd Huck Chair in Biomedical Data Sciences and Artificial Intelligence
Professor of Data Sciences, Informatics, Computer Science and Engineering, Bioinformatics & Genomics,
Public Health Sciences and Neuroscience
Director, Center for Artificial Intelligence Foundations and Scientific Applications
Associate Director, Institute for Computational and Data Sciences
Pennsylvania State University

vhonavar@psu.edu
<http://faculty.ist.psu.edu/vhonavar>
<http://ailab.ist.psu.edu>

Linear Classifiers: Simple Neural Networks

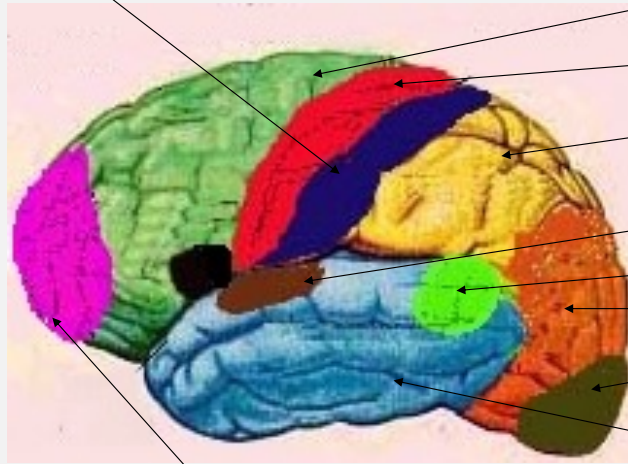
- Background
- Threshold logic functions
- Connection to logic
- Connection to geometry
- Learning threshold functions – perceptron algorithm
- Perceptron convergence theorem
- Multi-category extensions
- Alternative loss functions and algorithms

Background – Brains and Computers

- Brain consists of 10^{11} neurons, each connected to $\sim 10^4$ other neurons
- Each neuron is slow
 - 1 millisecond to respond to a stimulus
- Brain is astonishingly fast at perceptual tasks e.g. face recognition
- Brain processes and learns from multiple sources of sensory information (visual, tactile, auditory...)
- Brain is massively parallel, shallowly serial, modular and roughly hierarchical with recurrent and lateral connectivity within and between modules
- Turing: Thinking can be modeled by computation
- If thinking can be modeled by – computation
 - it is natural to ask how and what are the algorithms that underly thinking or
 - what do brains compute

Brain and information processing

Primary somato-sensory cortex



Motor association
cortex

Primary motor
cortex

Sensory association area

Auditory cortex

Speech comprehension

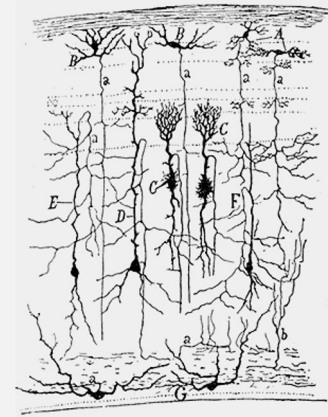
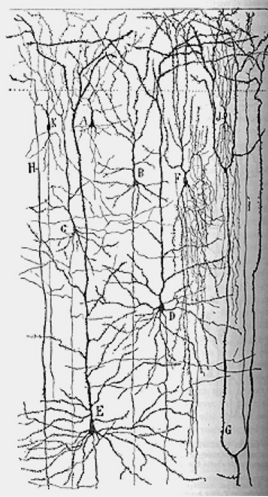
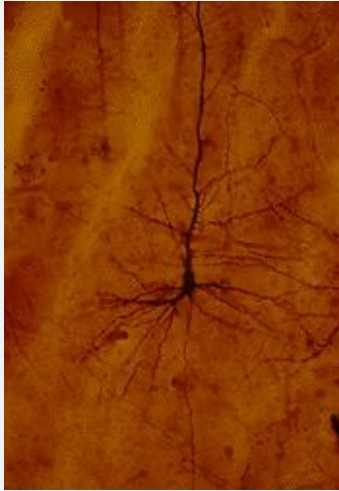
Visual association area

Primary visual cortex

Auditory association
area

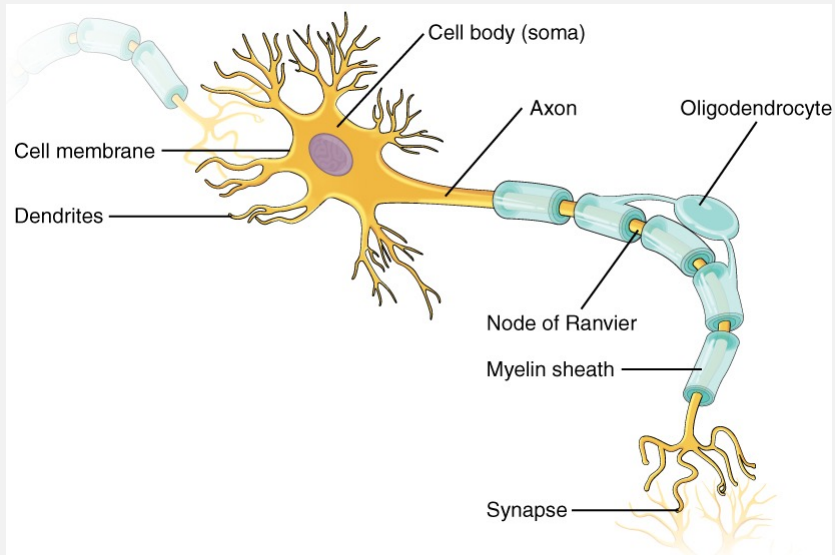
Prefrontal cortex


Neural Networks



Ramon Cajal, 1900


Neurons and Computation





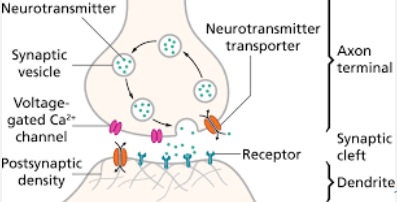
PennState
Institute for Computational
and Data Sciences

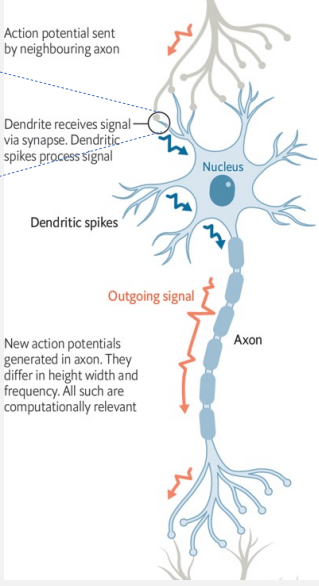
Center for Artificial Intelligence Foundations & Scientific Applications
Artificial Intelligence Research Laboratory

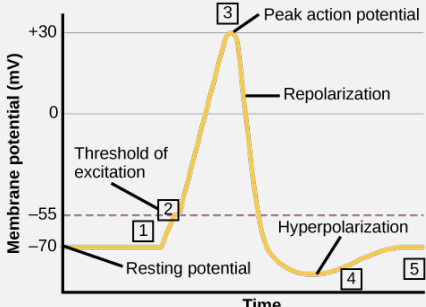



PennState
Clinical and Translational
Science Institute

Neural information processing







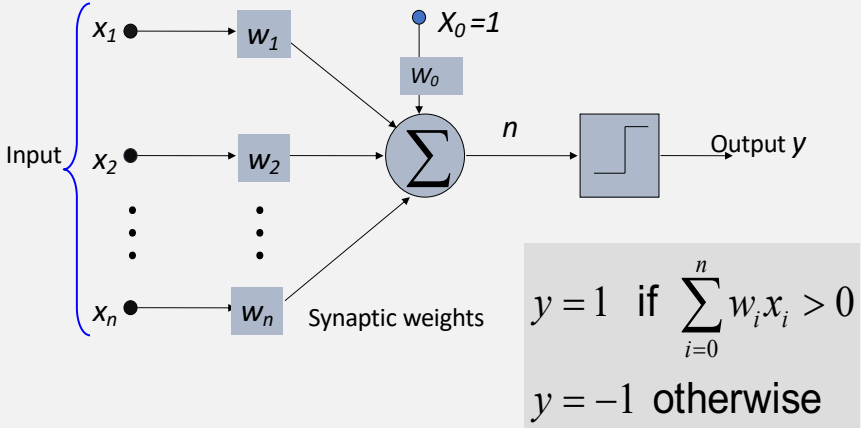


PennState
College of Earth and Atmospheric
Sciences and Technology

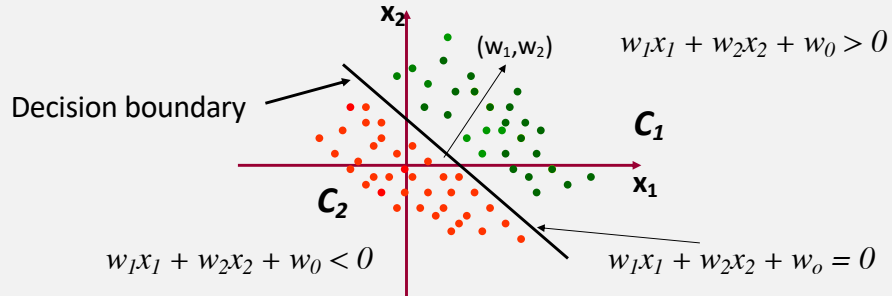
Fall 2022

Vasant G Honavar

McCulloch-Pitts computational model of a neuron



Threshold neuron – Connection with Geometry



$\sum_{i=1}^n w_i x_i + w_0 = 0$ describes a hyperplane which divides the instance space \mathcal{R}^n into two half-spaces

$$\mathcal{X}_+ = \{ \mathbf{X}_p \in \mathcal{R}^n \mid \mathbf{W} \cdot \mathbf{X}_p + w_0 > 0 \} \quad \text{and} \quad \mathcal{X}_- = \{ \mathbf{X}_p \in \mathcal{R}^n \mid \mathbf{W} \cdot \mathbf{X}_p + w_0 < 0 \}$$

McCulloch-Pitts Neuron or Threshold Neuron

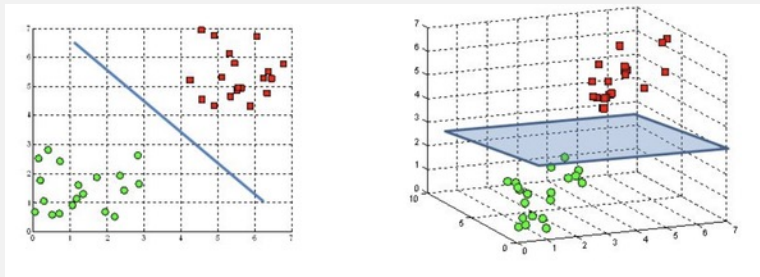
$$\begin{aligned} y &= \text{sign}(\mathbf{W} \bullet \mathbf{X} + w_0) \\ &= \text{sign}\left(\sum_{i=0}^n w_i x_i\right) \\ &= \text{sign}(\mathbf{W}^T \mathbf{X} + w_0) \end{aligned}$$

$$\mathbf{X} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

$$\mathbf{W} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix}$$

$$\begin{aligned} \text{sign}(v) &= 1 \text{ if } v > 0 \\ &= 0 \text{ otherwise} \end{aligned}$$

McCulloch-Pitts Neuron - Connection to geometry



- A perceptron with 3 weights $[w_0, w_1, w_2]$ implements a line in 2-D
- A perceptron with 4 weights $[w_0, w_1, w_2, w_4]$ implements a plane in 3-D
- A perceptron with $n + 1$ weights $[w_0, \dots, w_n]$ implements an $(n - 1)$ dimensional hyperplane in n -D
 - Dividing the n -D space into two half spaces

Perceptron – Connection with Geometry

- Data live in \mathfrak{R}^n or \mathfrak{R}^{n+1} if we add a dummy input of $x_0 = 1$
- Weights that define hyperplanes live in \mathfrak{R}^{n+1}
- A particular choice of weights defines a hyperplane given by

$$\sum_{i=1}^n w_i x_i + w_0 = 0$$

or

$$\sum_{i=0}^n w_i x_i = 0$$

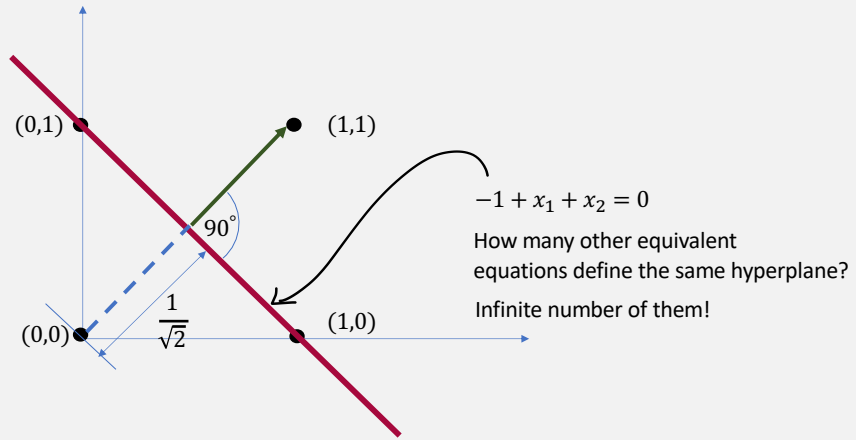
or

$$\mathbf{w} \cdot \mathbf{x} = 0$$

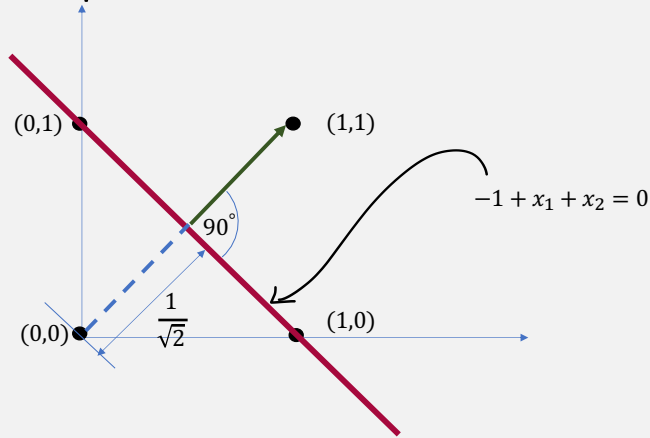
- The orientation of the hyperplane is specified by its normal vector $[w_1 \cdots w_n]^T$
- The distance of the hyperplane from a given data point \mathbf{x}_p is given by

$$\frac{|\mathbf{w} \cdot \mathbf{x}_p|}{\sqrt{w_1^2 + \cdots + w_n^2}} = \frac{|w_0 + w_1 x_{1p} + \cdots + w_n x_{np}|}{\sqrt{w_1^2 + \cdots + w_n^2}}$$

Example



Example



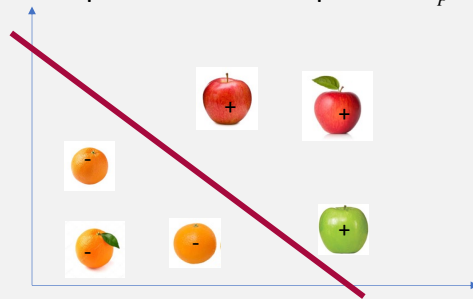
Which side of the hyperplane does the point (2,1) lie?

Check the sign of $w_0 + \sum_{i=1}^2 w_i x_i = -1 + 2 + 1 = 2$

The sign is positive, so on the “positive side” pointed by the direction of the positive normal

Perceptron as a pattern classifier

- The threshold neuron, that implements the “right” hyperplane, can be used to classify a set of data samples into one of two classes C_1 , C_2 e.g., apples and oranges when they are represented as points in a suitable feature space
- If the output of the neuron for input pattern X_p is +1 then X_p is assigned to class C_1
- If the output is -1 then the pattern X_p is assigned to C_2



Threshold neuron – Connection with Logic

- Suppose the input space is $\{0,1\}^n$
- Then threshold neuron computes a Boolean function $f : \{0,1\}^n \rightarrow \{-1,1\}$

Example

Let $w_0 = -1.5; w_1 = w_2 = 1$

- In this case, if we interpret 1 as TRUE and -1 as FALSE, the threshold neuron implements the logical AND function

x_1	x_2	$h(X) = \mathbf{w} \cdot \mathbf{x}$	y
0	0	-1.5	-1
0	1	-0.5	-1
1	0	-0.5	-1
1	1	0.5	1

Threshold neuron – Connection with Logic

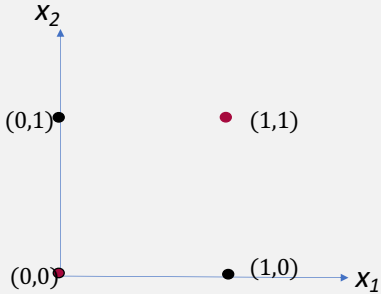
- A threshold neuron with the appropriate choice of weights can implement Boolean AND, OR, and NOT function
- **Theorem:** For any arbitrary Boolean function f , there exists a network of threshold neurons that can implement f .
- **Theorem:** Any arbitrary finite state automaton can be realized using threshold neurons and *delay* units
- Networks of threshold neurons, given access to unbounded memory, can compute any Turing-computable function
- **Corollary:** Brains if given access to enough working memory, can compute any computable function

Threshold neuron: Connection with Logic

Theorem: There exist Boolean functions that cannot be implemented by a single threshold neuron.

Example: Exclusive OR

x_1	x_2	y
0	0	-1
0	1	1
1	0	1
1	1	-1



Why?
A hyperplane separating the red points from the black points does not exist!

Threshold neuron – Connection with Logic

- Definition: A function that can be computed by a single threshold neuron is called a threshold function
- Of the 16 2-input Boolean functions, 14 are Boolean threshold functions
- As n increases, the number of Boolean threshold functions becomes an increasingly small fraction of the total number of n -input Boolean functions

$$N_{Threshold}(n) \leq 2^{n^2}$$

$$N_{Boolean}(n) = 2^{2^n}$$

Terminology and Notation

- **Synonyms:** Threshold function, Linearly separable function, linear discriminant function
- **Synonyms:** Threshold neuron, McCulloch-Pitts neuron, Perceptron, Threshold Logic Unit (TLU)
- We often include w_0 as one of the components of \mathbf{w} and incorporate x_0 as the corresponding component of \mathbf{x} with the understanding that $x_0 = 1$.
- Then $y = 1$ if $\mathbf{w} \cdot \mathbf{x} > 0$ and $y = -1$ otherwise.

Learning Threshold functions

A training example E_k is an ordered pair (X_k, d_k) where

$$\mathbf{X}_k = [x_{0k} \ x_{1k} \ \dots \ x_{nk}]^T$$

is an $(n + 1)$ dimensional input sample, $d_k = f(\mathbf{X}_k) \in \{-1, 1\}$
is the desired output of the classifier and f is an unknown
target function to be learned.

A training set E is simply a multi-set of examples.

Learning Threshold functions

$$S^+ = \{X_k | (X_k, d_k) \in E \text{ and } d_k = 1\}$$

$$S^- = \{X_k | (X_k, d_k) \in E \text{ and } d_k = -1\}$$

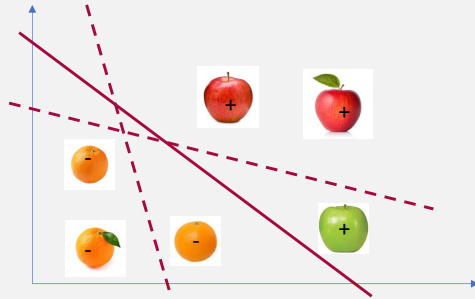
We say that a training set E is linearly separable if and only if

$$\exists W^* \text{ such that } \forall X_p \in S^+, W^* \bullet X_p > 0 \\ \text{and } \forall X_p \in S^-, W^* \bullet X_p < 0$$

Learning Task: Given a linearly separable training set E , find a solution

$$W^* \text{ such that } \forall X_p \in S^+, W^* \bullet X_p > 0 \text{ and } \forall X_p \in S^-, W^* \bullet X_p < 0$$

Learning to classify = finding separating hyperplane



Rosenblatt's Perceptron Learning Algorithm

Initialize $\mathbf{W} = [0 \ 0 \ \dots \ 0]^T$ Set learning rate $\eta > 0$

Repeat until a complete pass through E results in no weight updates

For each training example $E_k \in E$

$$\left\{ \begin{array}{l} y_k \leftarrow \text{sign}(\mathbf{W} \bullet \mathbf{X}_k) \\ \mathbf{W} \leftarrow \mathbf{W} + \eta(d_k - y_k)\mathbf{X}_k \end{array} \right\}$$

$\mathbf{W}^* \leftarrow \mathbf{W};$ Return (\mathbf{W}^*)

Perceptron learning algorithm – Example

$$S^+ = \{(1, 1, 1), (1, 1, -1), (1, 0, -1)\}$$

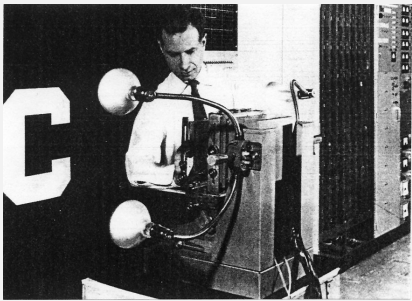
$$S^- = \{(1, -1, -1), (1, -1, 1), (1, 0, 1)\}$$

X_k	d_k	W	$W \cdot X_k$	y_k	Update?	Updated W
(1, 1, 1)	1	(0, 0, 0)	0	-1	Yes	(1, 1, 1)
(1, 1, -1)	1	(1, 1, 1)	1	1	No	(1, 1, 1)
(1, 0, -1)	1	(1, 1, 1)	0	-1	Yes	(2, 1, 0)
(1, -1, -1)	-1	(2, 1, 0)	1	1	Yes	(1, 2, 1)
(1, -1, 1)	-1	(1, 2, 1)	0	-1	No	(1, 2, 1)
(1, 0, 1)	-1	(1, 2, 1)	2	1	Yes	(0, 2, 0)
(1, 1, 1)	1	(0, 2, 0)	2	1	No	(0, 2, 0)

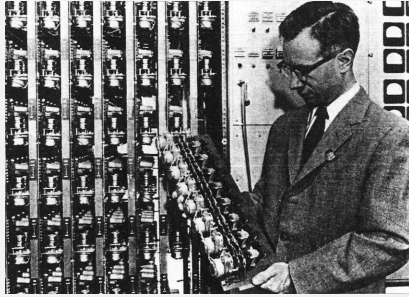
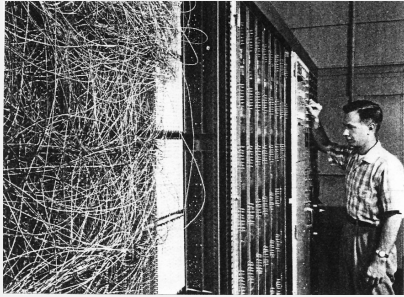
$$W = (0\ 0\ 0)$$

$$\eta = \frac{1}{2}$$

Perceptron (1957)



Perceptron

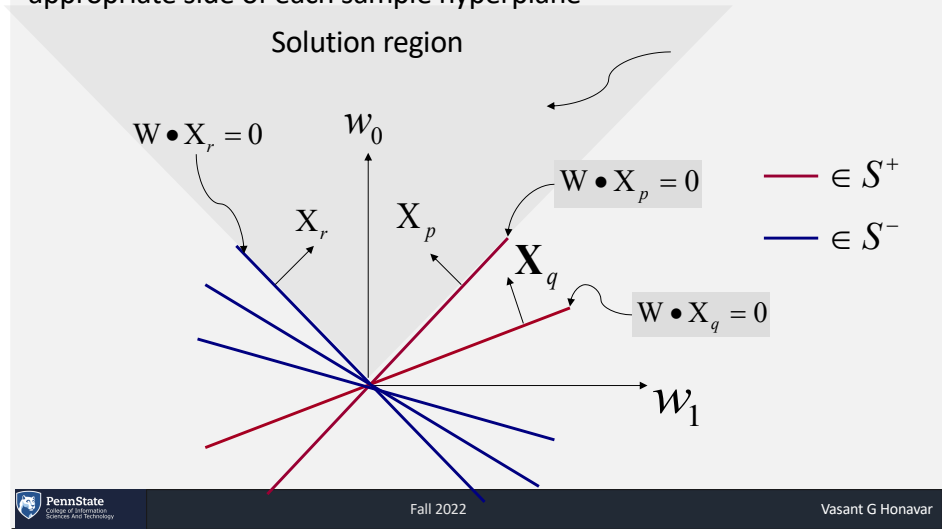


Understanding Weight Updates

- During learning, training data are fixed.
- What is being updated are the weights.
- Consider the weight space defined by the coordinates of the weight vector
- Points in this space correspond to different choices of the weights
- Just as in the data space, weights defined hyperplanes, in the weight space, training data samples define (fixed) hyperplanes.

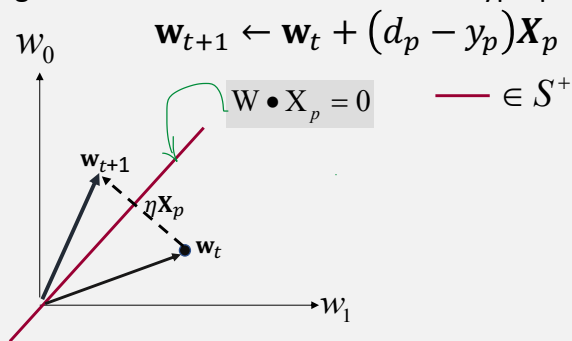
Understanding Weight Updates

Goal: Find a point in the weight space that lies on the appropriate side of each sample hyperplane



Understanding Weight Updates

- **Goal:** Find a point in the weight space that lies on the appropriate side of each sample hyperplane
- If the current weight vector is on the **wrong side** of a pattern hyperplane, the most efficient way to go to the **right side** is to move along the direction of the **normal** to the hyperplane



Perceptron Convergence Theorem (Novikoff)

Theorem Let $E = \{(\mathbf{X}_k, d_k)\}$ be a training set where $\mathbf{X}_k \in \{1\} \times \mathcal{R}^n$
and $d_k \in \{-1, 1\}$

Let $S^+ = \{\mathbf{X}_k | (\mathbf{X}_k, d_k) \in E \ \& \ d_k = 1\}$ and $S^- = \{\mathbf{X}_k | (\mathbf{X}_k, d_k) \in E \ \& \ d_k = -1\}$

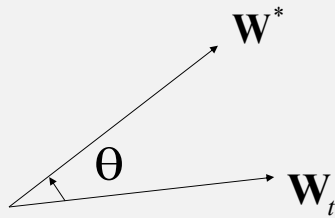
The perceptron algorithm is guaranteed to terminate after a bounded number t of weight updates with a weight vector

\mathbf{W}^* such that $\forall \mathbf{X}_k \in S^+, \mathbf{W}^* \cdot \mathbf{X}_k \geq \delta$ and $\forall \mathbf{X}_k \in S^-, \mathbf{W}^* \cdot \mathbf{X}_k \leq -\delta$
for some $\delta > 0$, whenever such $\mathbf{W}^* \in \mathcal{R}^{n+1}$ and $\delta > 0$ exist
-- that is, E is linearly separable. The bound on the number t of weight updates is given by

$$t \leq \left(\frac{\|\mathbf{W}^*\| L}{\delta} \right)^2 \quad \text{where } L = \max_{\mathbf{X}_k \in S} \|\mathbf{X}_k\| \quad \text{and } S = S^+ \cup S^-$$

Proof of Perceptron Convergence Theorem

Let \mathbf{W}_t be the weight vector after t weight *updates*.



$$\text{Invariant: } \forall \theta \quad |\cos \theta| \leq 1$$

Proof of Perceptron Convergence Theorem

Let W^* be such that

$$\forall X_k \in S^+, W^* \bullet X_k \geq \delta \text{ and } \forall X_k \in S^-, W^* \bullet X_k \leq -\delta$$

WLOG assume that $W^* \bullet X = 0$ passes through the origin.

Let $\forall X_k \in S^+, Z_k = X_k,$

$$\forall X_k \in S^-, Z_k = -X_k,$$

$$Z = \{Z_k\}$$

$$\left(\forall X_k \in S^+, W^* \bullet X_k \geq \delta \ \& \ \forall X_k \in S^-, W^* \bullet X_k \leq -\delta \right)$$

$$\Leftrightarrow \left(\forall Z_k \in Z, W^* \bullet Z_k \geq \delta \right).$$

Let $E' = \{(Z_k, 1)\}$

Proof of Perceptron Convergence Theorem

$$W_{t+1} = W_t + \eta(d_k - y_k)Z_k$$

where $W_0 = [0 \ 0 \ \dots \ 0]^T$ and $\eta > 0$

[Weight update based on example $(Z_k, 1)$]

$$\Leftrightarrow [(d_k = 1) \wedge (y_k = -1)]$$

$$\begin{aligned} \therefore W^* \cdot W_{t+1} &= W^* \cdot (W_t + 2\eta Z_k) \\ &= (W^* \cdot W_t) + 2\eta(W^* \cdot Z_k) \end{aligned}$$

Since $\forall Z_k \in Z, (W^* \cdot Z_k \geq \delta), W^* \cdot W_{t+1} \geq W^* \cdot W_t + 2\eta\delta$

$$\therefore \forall t \quad W^* \cdot W_t \geq 2t\eta\delta \dots\dots\dots(a)$$

Proof of Perceptron Convergence Theorem

$$\begin{aligned} \|W_{t+1}\|^2 &\equiv W_{t+1} \bullet W_{t+1} \\ &= (W_t + 2\eta Z_k) \bullet (W_t + 2\eta Z_k) \\ &= (W_t \bullet W_t) + 4\eta(W_t \bullet Z_k) + 4\eta^2(Z_k \bullet Z_k) \end{aligned}$$

Note weight update based on $Z_k \Leftrightarrow (W_t \bullet Z_k \leq 0)$

$$\therefore \|W_{t+1}\|^2 \leq \|W_t\|^2 + 4\eta^2 \|Z_k\|^2 \leq \|W_t\|^2 + 4\eta^2 L^2$$

Hence $\|W_t\|^2 \leq 4\eta^2 L^2$

$$\therefore \forall t \quad \|W_t\| \leq 2\eta L \sqrt{t} \dots\dots\dots(b)$$

Proof of Perceptron Convergence Theorem

$$\begin{aligned} \text{From (a) we have: } & \forall t \quad (\mathbf{W}^* \bullet \mathbf{W}_t) \geq 2t\eta\delta \\ \Rightarrow & \left\{ \forall t \quad 2t\eta\delta \leq (\mathbf{W}^* \bullet \mathbf{W}_t) \right\} \Rightarrow \left\{ \forall t \quad 2t\eta\delta \leq \|\mathbf{W}^*\| \|\mathbf{W}_t\| \cos\theta \right\} \\ \Rightarrow & \left\{ \forall t \quad 2t\eta\delta \leq \|\mathbf{W}^*\| \|\mathbf{W}_t\| \right\} \because \forall \theta \quad \cos\theta \leq 1, \end{aligned}$$

Substituting for an upper bound on $\|\mathbf{W}_t\|$ from (b),

$$\begin{aligned} \forall t \quad \left\{ 2t\eta\delta \leq \|\mathbf{W}^*\| 2\eta L \sqrt{t} \right\} & \Rightarrow \left\{ \forall t \quad (\delta \sqrt{t} \leq \|\mathbf{W}^*\| L) \right\} \\ \Rightarrow t & \leq \left(\frac{\|\mathbf{W}^*\| L}{\delta} \right)^2 \end{aligned}$$

Notes on the Perceptron Convergence Theorem

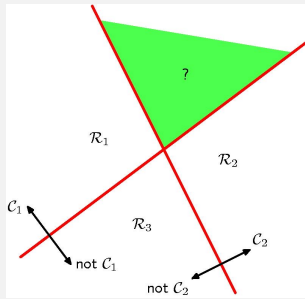
- The bound on the number of weight updates does not depend on the learning rate
- The bound is not useful in determining when to stop the algorithm because it depends on the norm of the unknown weight vector and delta
- The convergence theorem offers no guarantees when the training data set is not linearly separable

Exercise: Prove that the perceptron algorithm is robust with respect to fluctuations in the learning rate

$$0 < \eta_{\min} \leq \eta_t \leq \eta_{\max} < \infty$$

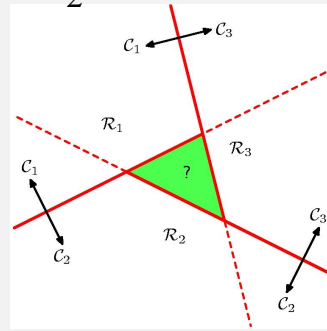
Multi-category classifiers

$K - 1$ binary classifiers



One-versus-rest

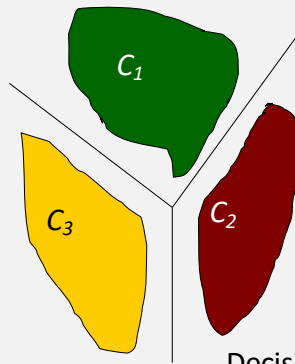
$\frac{K(K-1)}{2}$ binary classifiers



One-versus-one

Problem: Green region has ambiguous class membership

Multi-category classifiers



Define K linear functions of the form:

$$y_k(\mathbf{x}) = \mathbf{w}_k \cdot \mathbf{x}$$

$$h(\mathbf{x}) = \underset{k}{\operatorname{argmax}} y_k(\mathbf{x})$$

$$= \underset{k}{\operatorname{argmax}} \mathbf{w}_k \cdot \mathbf{x}$$

Decision surface between class C_k and C_j is given by

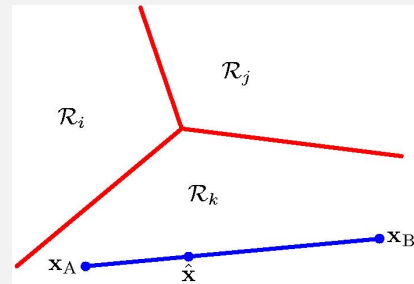
$$(\mathbf{w}_k - \mathbf{w}_j) \cdot \mathbf{x} = 0$$

Linear separator for K classes

- Decision regions defined by

$$(\mathbf{w}_k - \mathbf{w}_j) \cdot \mathbf{x} = 0$$

are singly connected and convex



For any points $\mathbf{x}_A, \mathbf{x}_B \in R_k$,

any $\hat{\mathbf{x}}$ that lies on the line connecting \mathbf{x}_A and \mathbf{x}_B

$$\hat{\mathbf{x}} = \lambda \mathbf{x}_A + (1 - \lambda) \mathbf{x}_B \text{ where } 0 \leq \lambda \leq 1$$

also lies in R_k

Winner-Take-All Networks

$$y_{ip} = 1 \text{ iff } \mathbf{W}_i \cdot \mathbf{X}_p > \mathbf{W}_j \cdot \mathbf{X}_p \quad \forall j \neq i$$

$$y_{ip} = 0 \text{ otherwise} \quad \text{Note: } \mathbf{W}_j \text{ are augmented weight vectors}$$

$$\mathbf{W}_1 = [1 \ -1 \ -1]^T, \mathbf{W}_2 = [1 \ 1 \ 1]^T, \mathbf{W}_3 = [2 \ 0 \ 0]^T$$

			$\mathbf{W}_1 \cdot \mathbf{X}_p$	$\mathbf{W}_2 \cdot \mathbf{X}_p$	$\mathbf{W}_3 \cdot \mathbf{X}_p$	y_1	y_2	y_3
1	-1	-1	3	-1	2	1	0	0
1	-1	+1	1	1	2	0	0	1
1	+1	-1	1	1	2	0	0	1
1	+1	+1	-1	3	2	0	1	0

What does neuron 3 compute?

Linear separability of multiple classes

Let $S_1, S_2, S_3 \dots S_M$ be multisets of instances

Let $C_1, C_2, C_3 \dots C_M$ be disjoint classes

$$\forall i S_i \subseteq C_i$$

$$\forall i \neq j C_i \cap C_j = \emptyset$$

We say that the sets $S_1, S_2, S_3 \dots S_M$ are linearly

separable iff \exists weight vectors $W_1^*, W_2^*, \dots W_M^*$ such that

$$\forall i \left\{ \forall X_p \in S_i, (W_i^* \cdot X_p > W_j^* \cdot X_p) \forall j \neq i \right\}$$

Training WTA Classifiers

$d_{kp} = 1$ iff $\mathbf{X}_p \in C_k$; $d_{kp} = 0$ otherwise

$y_{kp} = 1$ iff $\mathbf{W}_k \cdot \mathbf{X}_p > \mathbf{W}_j \cdot \mathbf{X}_p \quad \forall k \neq j$

Suppose $d_{kp} = 1, y_{jp} = 1$ and $y_{kp} = 0$

$$\mathbf{W}_k \leftarrow \mathbf{W}_k + \eta \mathbf{X}_p; \mathbf{W}_j \leftarrow \mathbf{W}_j - \eta \mathbf{X}_p;$$

All other weights are left unchanged.

Suppose $d_{kp} = 1, y_{jp} = 0$ and $y_{kp} = 1$.

The weights are unchanged.

Suppose $d_{kp} = 1, \forall j y_{jp} = 0$ (there was a tie)

$$\mathbf{W}_k \leftarrow \mathbf{W}_k + \eta \mathbf{X}_p$$

All other weights are left unchanged.

WTA Convergence Theorem

Given a linearly separable training set, the WTA learning algorithm is guaranteed to converge to a solution within a finite number of weight updates.

Proof Sketch:

- **Reduce** the WTA training problem to the problem of training a single perceptron using a **suitably transformed** training set.
- Then the proof of WTA learning algorithm reduces to the proof of perceptron learning algorithm

WTA Convergence Theorem

Let $\mathbf{W}^T = [\mathbf{W}_1 \mathbf{W}_2 \dots \mathbf{W}_M]^T$ be a concatenation of the weight vectors associated with the M neurons in the WTA group. Consider a multi - category training set $E = \{(\mathbf{X}_p, f(\mathbf{X}_p))\}$ where $\forall \mathbf{X}_p, f(\mathbf{X}_p) \in \{C_1, \dots, C_M\}$

Let $\mathbf{X}_p \in C_1$. Generate $(M-1)$ training examples using \mathbf{X}_p for an $M(n+1)$ input perceptron :

$$\mathbf{X}_{p12} = [\mathbf{X}_p \quad -\mathbf{X}_p \quad \phi \quad \phi \dots \phi]$$

$$\mathbf{X}_{p13} = [\mathbf{X}_p \quad \phi \quad -\mathbf{X}_p \quad \phi \dots \phi]$$

...

$$\mathbf{X}_{p1M} = [\mathbf{X}_p \quad \phi \quad \phi \dots \phi \quad -\mathbf{X}_p]$$

where ϕ is an all zero vector with the same dimension as \mathbf{X}_p and set the desired output of the corresponding perceptron to be 1 in each case.

Similarly, from each training example for an $(n+1)$ -input WTA, we can generate $(M-1)$ examples for an $M(n+1)$ input single neuron.

Let the union of the resulting $|E|(M-1)$ examples be E'

WTA Convergence Theorem

By construction, there is a one - to - one correspondence between the weight vector $\mathbf{W}^T = [\mathbf{W}_1 \mathbf{W}_2 \dots \mathbf{W}_M]^T$ that results from training an M - neuron WTA on the multi - category set of examples E and the result of training an $M(n+1)$ input perceptron on the transformed training set E' . Hence the convergence proof of WTA learning algorithm follows from the perceptron convergence theorem.