Linear Classification

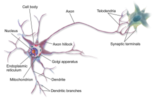
CSC 461: Machine Learning

Fall 2020

Prof. Marco Alvarez
University of Rhode Island

Linear classifiers

- Discriminative
  - ✓ Perceptron
  - ✓ Logistic regression
  - ✓ Support Vector Machines



- Generative
  - ✓ Linear Discriminant Analysis
  - ✓ Naive Bayes

Basics

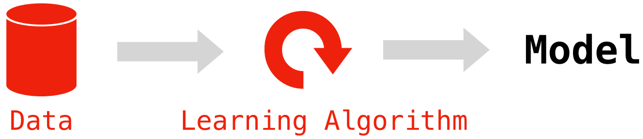
$$\mathscr{D} = \{(\mathbf{x}^{(1)}, y^{(1)}), \ldots, (\mathbf{x}^{(n)}, y^{(n)})\}$$

$$\mathbf{x}^{(i)} \in \mathbb{R}^d, y^{(i)} \in \{-1, +1\}$$

X1	X2	Y
0.5	0.1	+1
0.3	0.9	-1
0.3	0.875	-1
0.45	0.15	+1
...	...	...

Binary Classification

- Goal => learn a decision boundary such that two classes can be separated



$$\vec{x} \qquad f(\vec{x})$$

The sign function

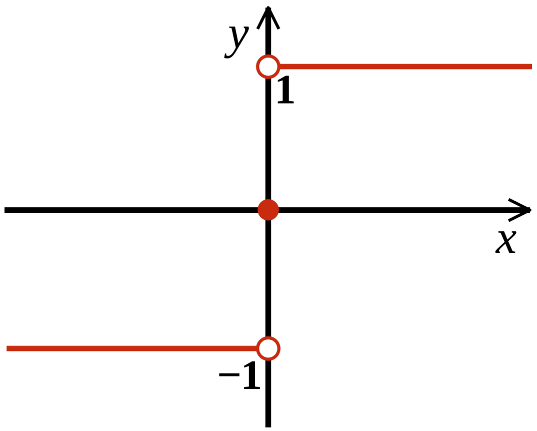
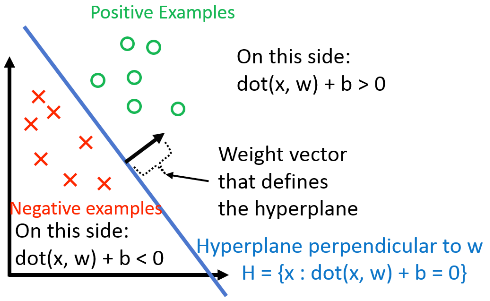


Image credit: Wikipedia

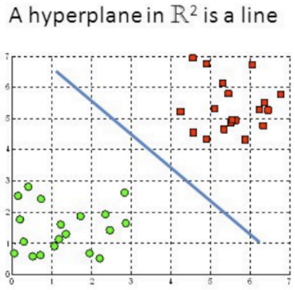
The Linear Classifier

$$f(\overrightarrow{x}) = sign(\overrightarrow{w} \cdot \overrightarrow{x} + b)$$



Credit: http://www.cs.cornell.edu/courses/cs4780/2018sp/lectures/lecturenote03.html

Decision Boundary



$$0 = b + w_1 x_1 + w_2 x_2$$

$$x_2 = -\frac{b}{w_2} - \frac{w_1}{w_2} x_1$$

Image credit: https://mc.ai/why-activation-function-is-used-in-neural-network/

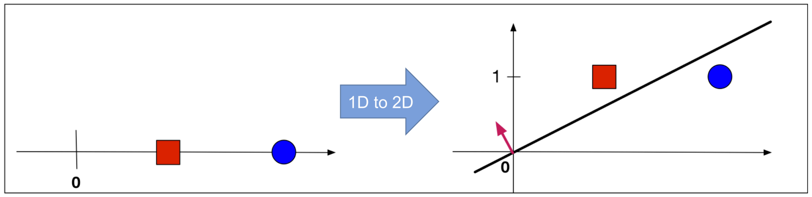
Absorbing the bias

$$\begin{aligned} f(\overrightarrow{x}) &= sign(\overrightarrow{w} \cdot \overrightarrow{x} + b) \\ &= sign\left(\sum_{i=1}^{d} w_i x_i + b\right) \\ &= sign\left(\sum_{i=0}^{d} w_i x_i\right) \\ &= sign(\overrightarrow{w} \cdot \overrightarrow{x}) \end{aligned}$$

X0	X1	X2	Y
1	0.5	0.1	+1
1	0.3	0.9	-1
1	0.3	0.875	-1
1	0.45	0.15	+1
…	…	…	…

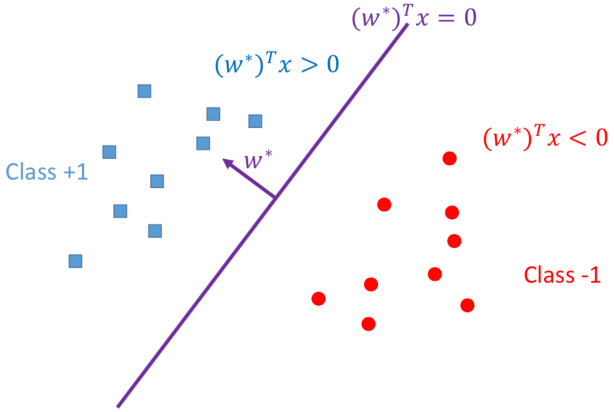
$x_0 = 1$ and $w_0 = -b$

Why using b?



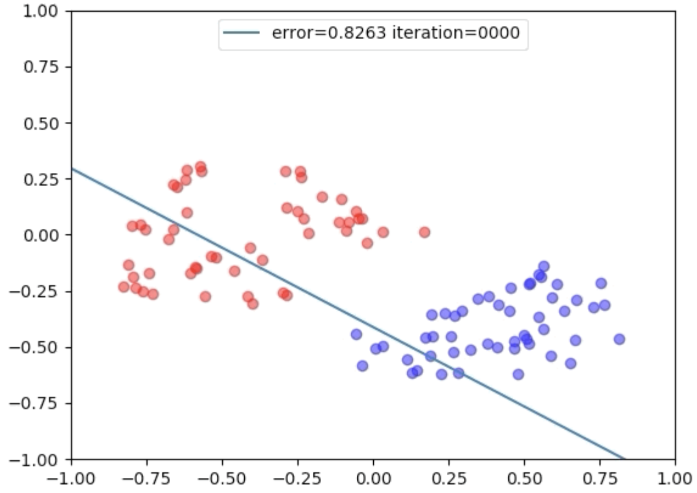
Decision boundary

- Hyperplane defined by h(x)



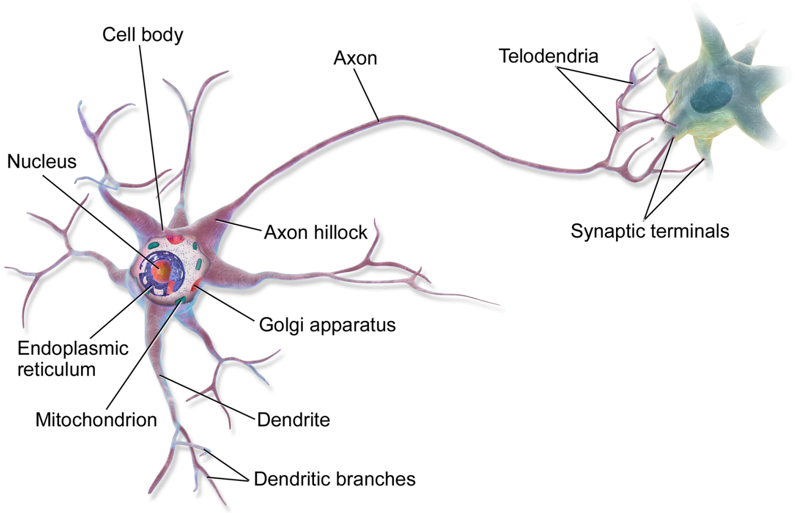
credit: yingyu liang, cos 495, princeton

Learning



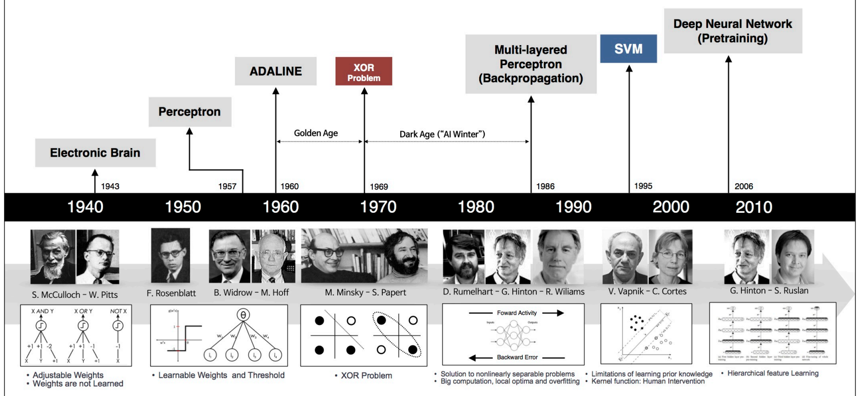
The Perceptron

Neuron



https://en.wikipedia.org/wiki/Neuron

Neural Networks


Deep Neural Network (Pretraining)
Multi-layered Perceptron (Backpropagation)
SVM
ADALINE
XOR Problem
Perceptron
Golden Age
Dark Age ("AI Winter")
Electronic Brain
1943
1957
1960
1969
1986
1995
2006
1940
1950
1960
1970
1980
1990
2000
2010
S. McCulloch - W. Pitts
F. Rosenblatt
B. Widrow - M. Hoff
M. Minsky - S. Papert
D. Rumelhart - G. Hinton - R. Wiliams
V. Vapnik - C. Cortes
G. Hinton - S. Ruslan
X AND Y
X OR Y
NOT X
Foward Activity
Backward Error
• Adjustable Weights
• Weights are not Learned
• Learnable Weights and Threshold
• XOR Problem
• Solution to nonlinearly separable problems
• Big computation, local optima and overfitting
• Limitations of learning prior knowledge
• Kernel function: Human Intervention
• Hierarchical feature Learning

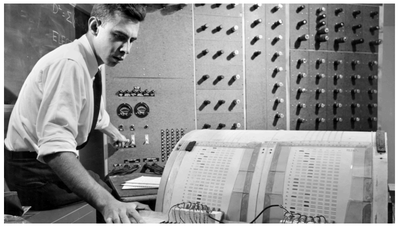
http://beamandrew.github.io/deeplearning/2017/02/23/deep_learning_101_part1.html

Rosenblatt (1958)

- Perceptron introduced by Frank Rosenblatt (psychologist, logician)
  - ✓ based on work from McCulloch-Pitts and Hebb
  - ✓ very powerful **learning** algorithm with high expectations

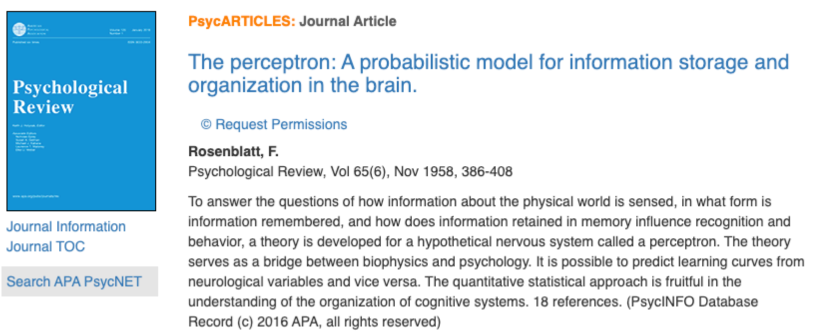
NEW NAVY DEVICE LEARNS BY DOING; Psychologist Shows Embryo of Computer Designed to Read and Grow Wiser

WASHINGTON, July 7, 1958 (UPI) -- The Navy revealed the embryo of an electronic computer today that it expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence.



https://news.cornell.edu/stories/2019/09/professors-perceptron-paved-way-ai-60-years-too-soon

Rosenblatt (1958)



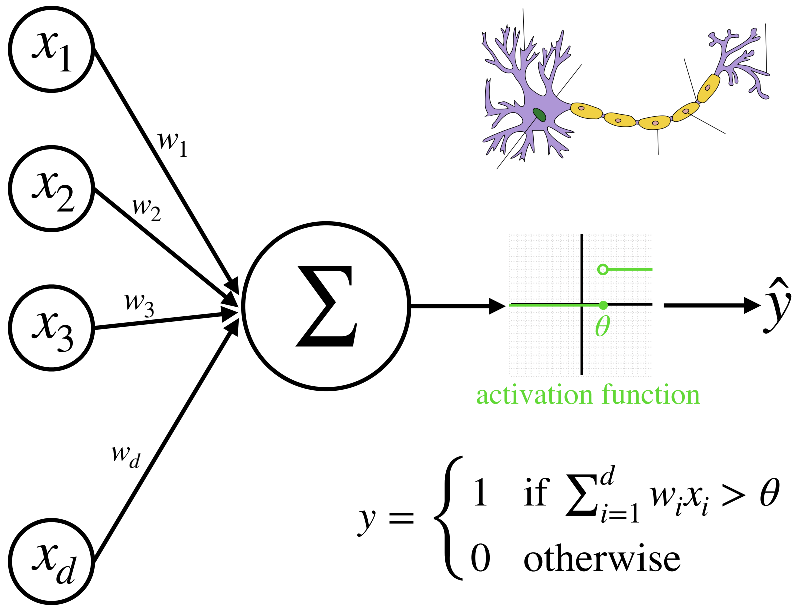
PsycARTICLES: Journal Article

The perceptron: A probabilistic model for information storage and organization in the brain.

© Request Permissions

Rosenblatt, F.
Psychological Review, Vol 65(6), Nov 1958, 386-408

To answer the questions of how information about the physical world is sensed, in what form is information remembered, and how does information retained in memory influence recognition and behavior, a theory is developed for a hypothetical nervous system called a perceptron. The theory serves as a bridge between biophysics and psychology. It is possible to predict learning curves from neurological variables and vice versa. The quantitative statistical approach is fruitful in the understanding of the organization of cognitive systems. 18 references. (PsycINFO Database Record (c) 2016 APA, all rights reserved)



$$y = \begin{cases} 1 & \text{if } \sum_{i=1}^{d} w_i x_i > \theta \\ 0 & \text{otherwise} \end{cases}$$

Absorbing the threshold/bias

$$y = \begin{cases} 1 & \text{if } \sum_{i=1}^{d} w_i x_i > \theta \\ 0 & \text{otherwise} \end{cases}$$

$$y = \begin{cases} 1 & \text{if } \sum_{i=1}^{d} w_i x_i - \theta > 0 \\ 0 & \text{otherwise} \end{cases}$$

$$y = \begin{cases} 1 & \text{if } \sum_{i=0}^{d} w_i x_i > 0 \\ 0 & \text{otherwise} \end{cases}$$

$x_0 = 1$ and $w_0 = -\theta$

Another look

For mathematical convenience we will use +1 and -1 instead of 1 and 0

$$\begin{aligned} h_w(\mathbf{x}) &= \sigma\left(\sum_{i=1}^{d} w_i x_i - \theta\right) \\ &= \sigma\left(\sum_{i=0}^{d} w_i x_i\right) \\ &= \sigma(\mathbf{w}^T \mathbf{x}) \end{aligned}$$

$$\sigma(z) = \begin{cases} 1 & \text{if } z > 0 \\ -1 & \text{if } z \leq 0 \end{cases}$$

$$\mathscr{H} = \{h_w : \mathbf{w} \in \mathbb{R}^{d+1}\}$$

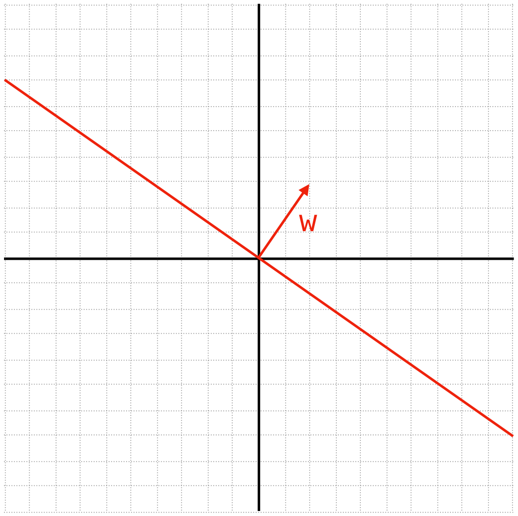
Perceptron Algorithm

- Start with a null vector **w**
- Repeat for T epochs
  - ✓ shuffle the data
  - ✓ for all examples in training data
    - ✓ if misclassified
      - update the weight vector by adding **x** to **w** if the actual label is positive and subtracting **x** from **w** otherwise
- Return **w**

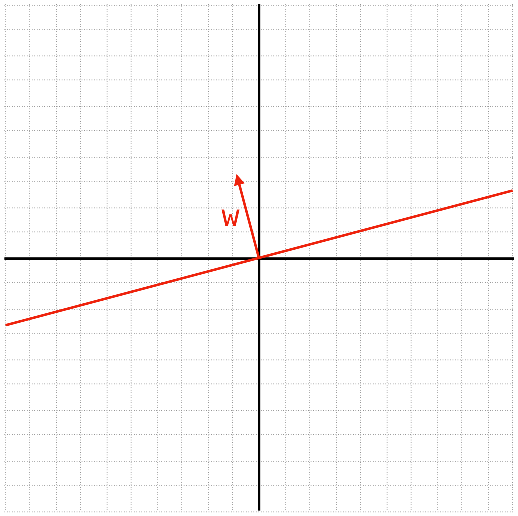
Write the pseudocode

https://colab.research.google.com/drive/1p7LjENLd6whkoVY2yRGg-Nm5v4XNjdDt

Mistake on a positive (update)



Mistake on a negative (update)



Intuition

- Suppose a mistake on the positive side:

$$y = +1 \qquad \mathbf{w}^T x \leq 0$$

- After 1 update the new weight vector will be:

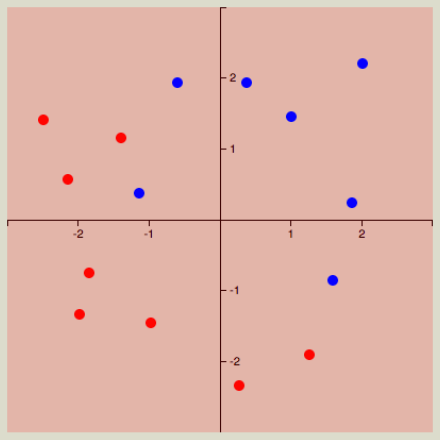
$$\mathbf{w}_{t+1} = \mathbf{w}_t + \mathbf{x}$$

- Classifying the datapoint with the new weight vector:

$$\mathbf{w}_{t+1}{}^T\mathbf{x} = (\mathbf{w}_t + \mathbf{x})^T\mathbf{x} = \mathbf{w}_t^T\mathbf{x} + \mathbf{x}^T\mathbf{x} \geq \mathbf{w}_t^T\mathbf{x}$$

use same idea for mistakes on the negative side

Demo



https://planspace.org/20150907-interactive_perceptron_training_toy/

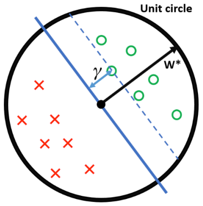
Perceptron convergence theorem

The argument goes as follows: Suppose $\exists \mathbf{w}^*$ such that $y_i(\mathbf{x}^\top \mathbf{w}^*) > 0 \; \forall (\mathbf{x}_i, y_i) \in D$.

Now, suppose that we rescale each data point and the $\mathbf{w}^*$ such that

$$||\mathbf{w}^*|| = 1 \qquad \text{and} \qquad ||\mathbf{x}_i|| \leq 1 \;\; \forall \mathbf{x}_i \in D$$

Let us define the Margin γ of the hyperplane $\mathbf{w}^*$ as $\gamma = \min_{(\mathbf{x}_i, y_i) \in D} |\mathbf{x}_i^\top \mathbf{w}^*|$.



To summarize our setup:

- All inputs $\mathbf{x}_i$ live within the unit sphere
- There exists a separating hyperplane defined by $\mathbf{w}^*$, with $\|\mathbf{w}\|^* = 1$ (i.e. $\mathbf{w}^*$ lies exactly on the unit sphere).
- γ is the distance from this hyperplane (blue) to the closest data point.

Theorem: If all of the above holds, then the Perceptron algorithm makes at most $1/\gamma^2$ mistakes.

http://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote03.html

Perceptron (remarks)

- Assumes data is linearly separable
  - ✓ does not converge if classes are not linearly separable
- Different correct solutions can be found
  - ✓ most are not optimal in terms of generalization

Parameters vs Hyperparameters

- Parameters
  - ✓ weights and bias
- **Hyperparameters**
  - ✓ number of epochs (one epoch is one pass over the training data)