CMPS 460 – Spring 2022

MACHINE LEARNING

Tamer Elsayed

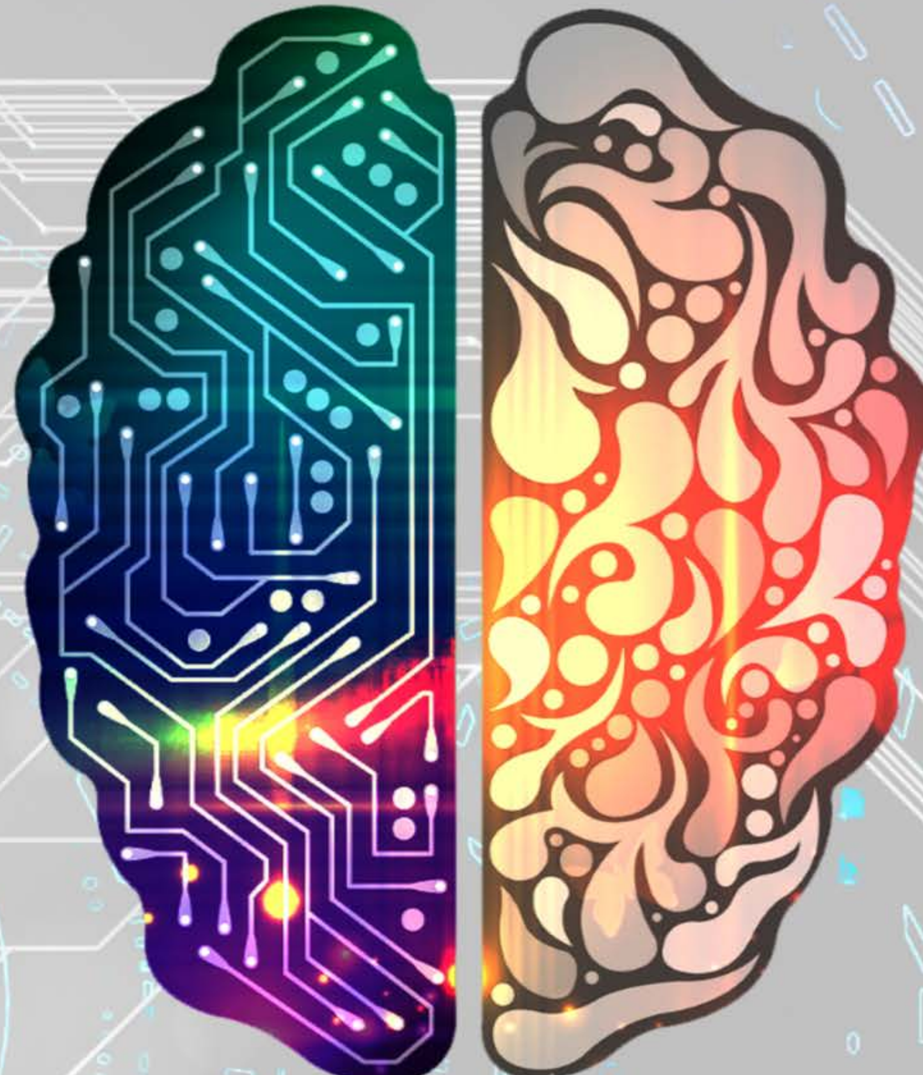


Image hosted by: WittySparks.com | Image source: Pixabay.com

4

Perceptron

Chapter 4

Roadmap …

- A new model/algorithm
  - the perceptron
    - and its variants: voted, averaged
  - convergence

- Fundamental Machine Learning Concepts
  - Online vs. batch learning
  - Error-driven learning
  - Linear separability and margin of a dataset

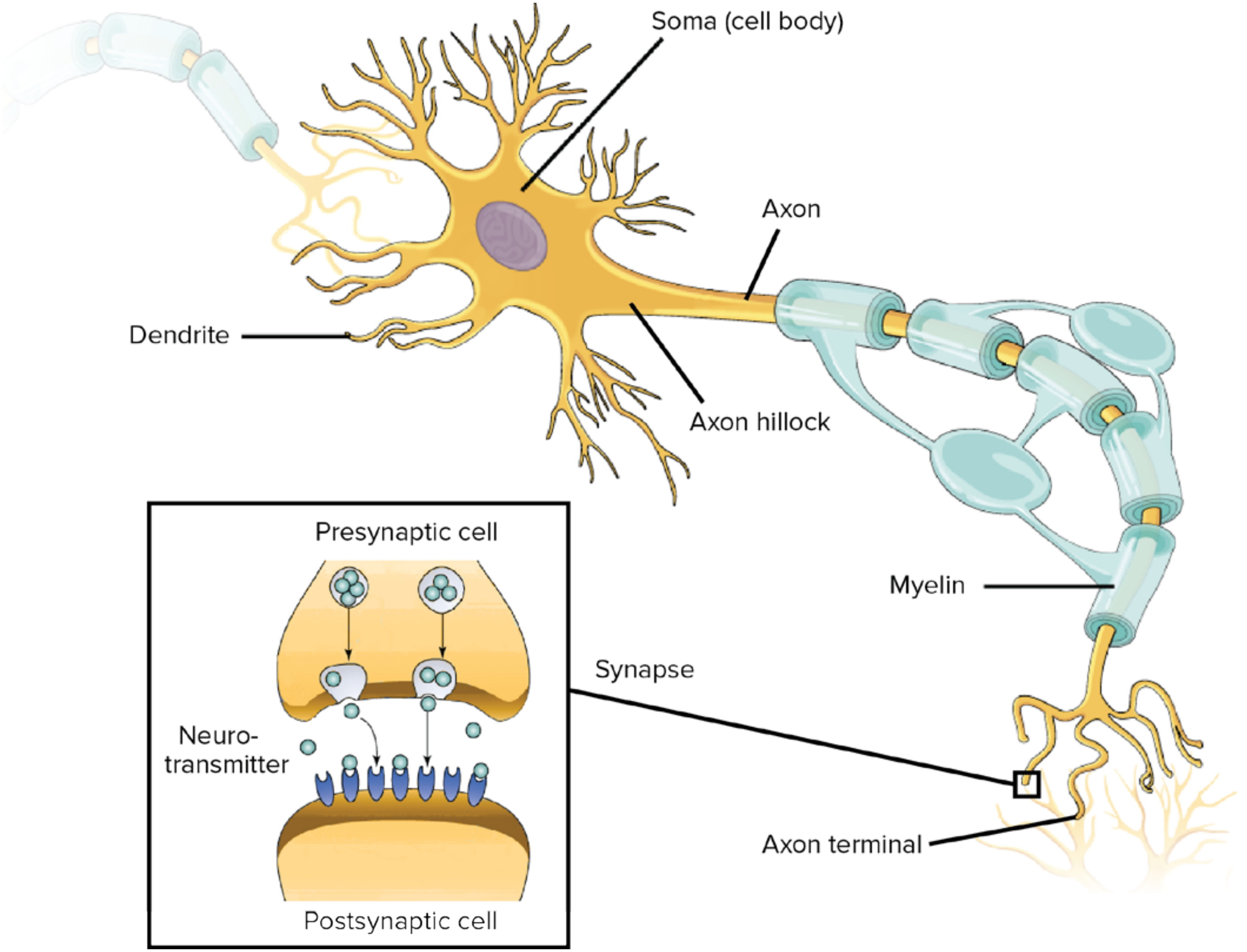
Motivation

Why?

- In DTs: only a small number of features are used.
- In kNN: all features are used equally.
- What if we want to use most of the features, but use some more than others.

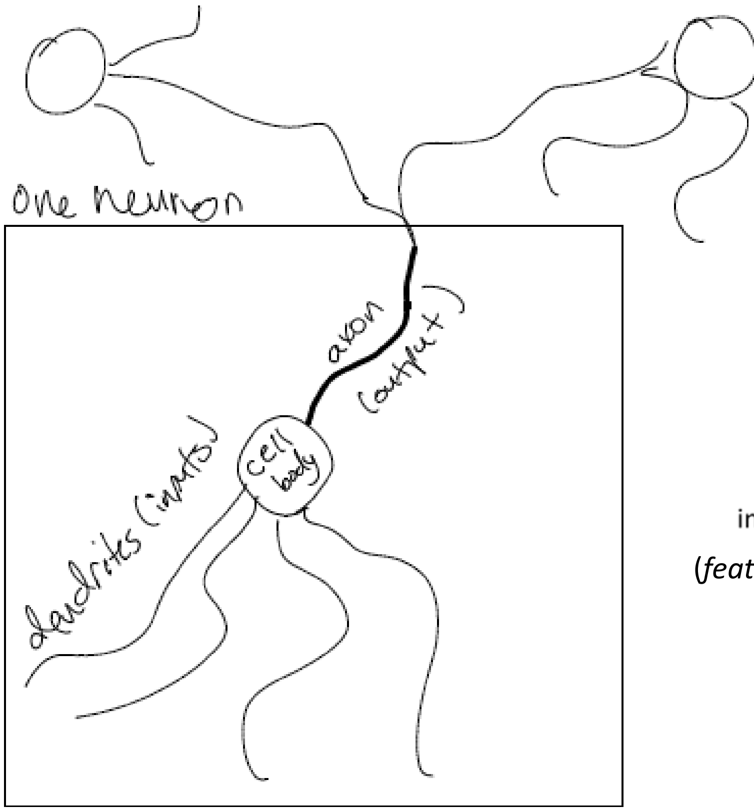
- Perceptron algorithm: learning weights for features

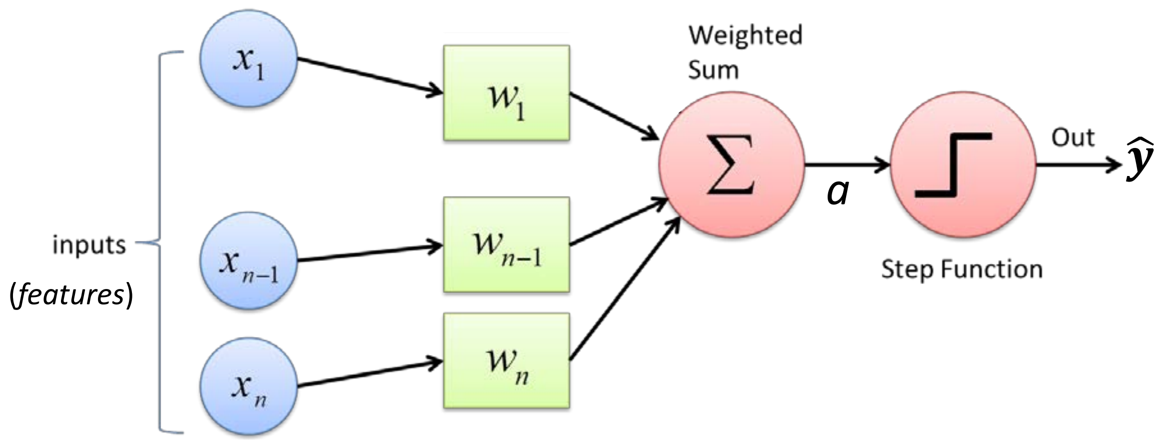
Biological Inspiration: A Neuron …



https://www.khanacademy.org/science/biology/human-biology/neuron-nervous-system/a/overview-of-neuron-structure-and-function

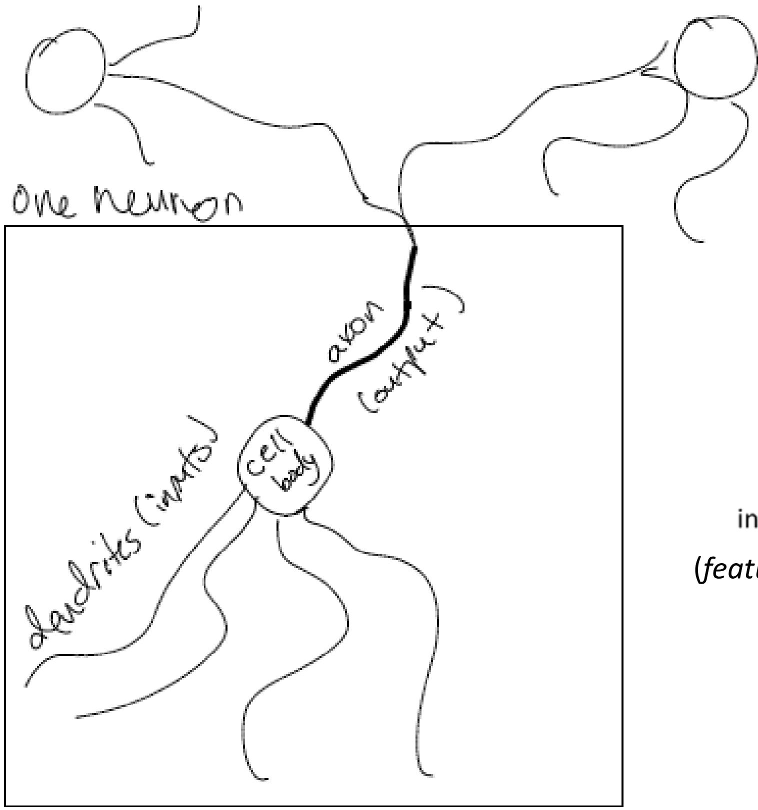
Perceptron: One Neuron …

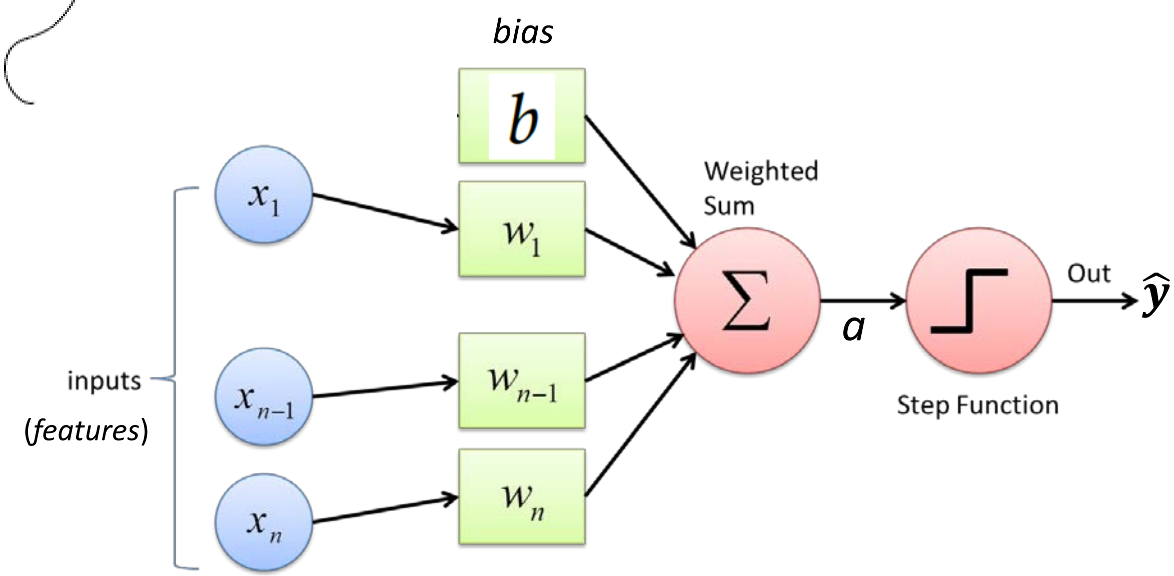




$$a = \left[\sum_{i=1}^{n} w_i x_i \right]$$

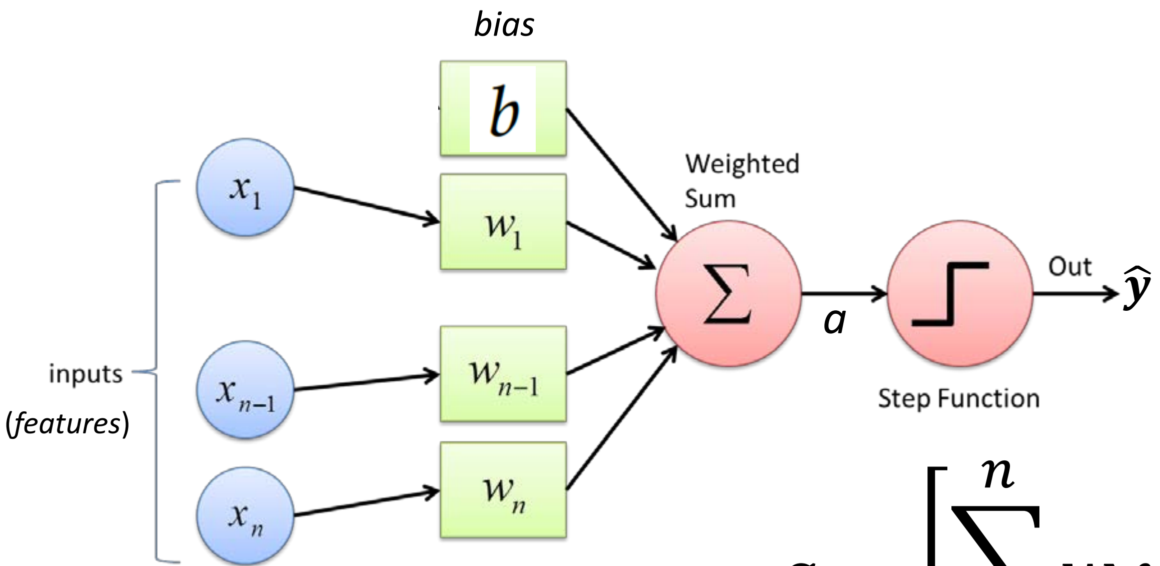
Perceptron: Neural Model of Learning





$$a = \left[\sum_{i=1}^{n} w_i x_i\right] + b$$

Perceptron: Example



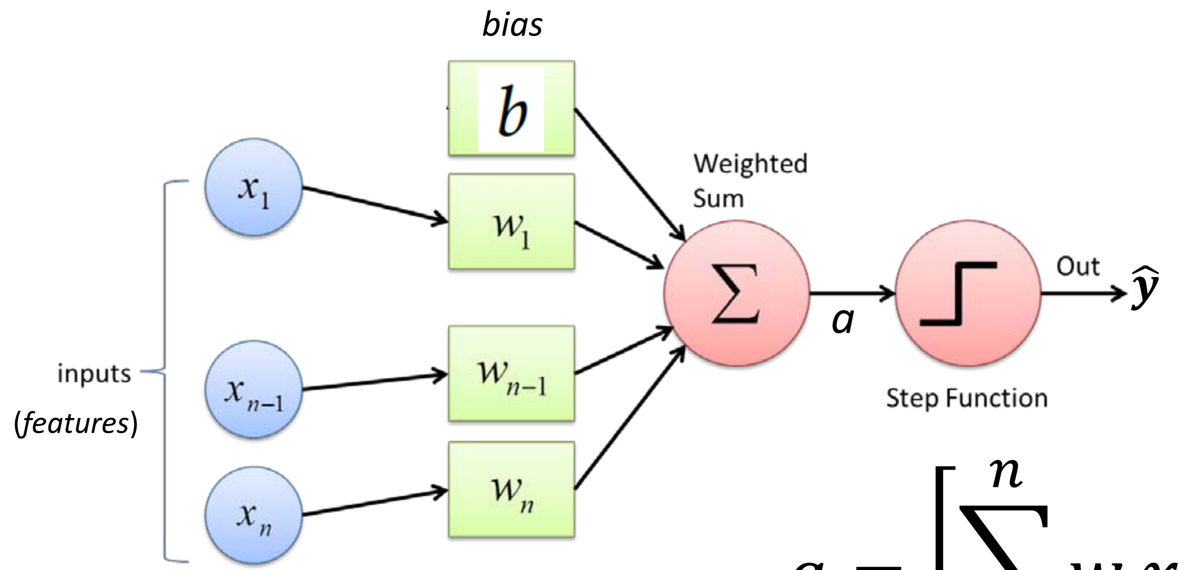
$$a = \left[\sum_{i=1}^{n} w_i x_i\right] + b$$

$n = 3$
$x_1 = 1, x_2 = -1, x_3 = 3$
$w_1 = -1, w_2 = 2, w_3 = 3, b = 0$
$a = ?\ \hat{y} = ?$

$a = -1 * 1 + 2 * -1 + 3 * 3 + 0 = 6$
$\hat{y} = 1$

if $b = -7$?

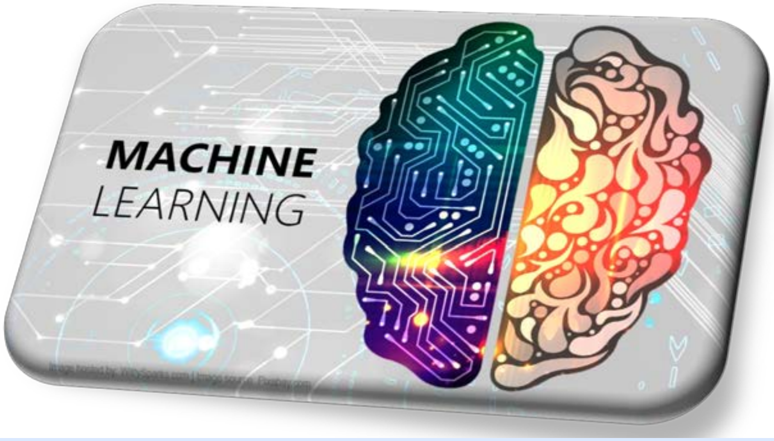
Prediction in Perceptron



$$a = \left[\sum_{i=1}^{n} w_i x_i\right] + b$$

Parameters?

if $a > 0 \Rightarrow$ positive



Perceptron Algorithm

Properties of Training Algorithm

- **Online**
  - look at one example at a time (and update the model as soon as we make an error).
  - As opposed to *batch* algorithms that update parameters after seeing the entire training set.

- **Error-driven**
  - We only update parameters/model if we make an error.

Perceptron Algorithm

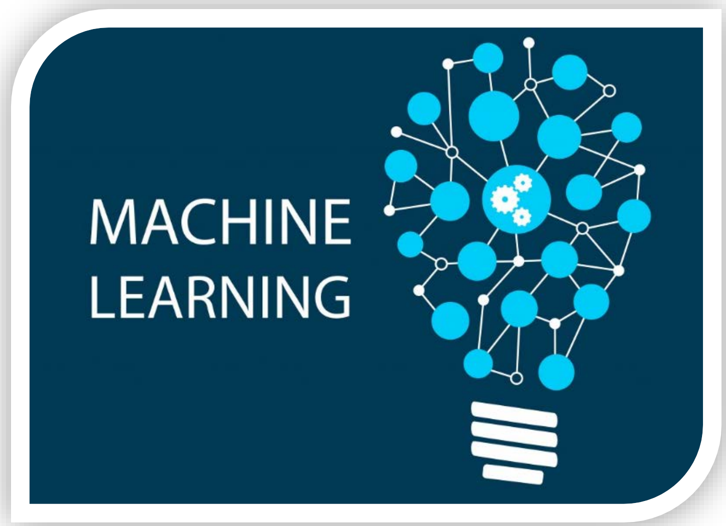
Algorithm 5 PERCEPTRONTRAIN($\mathbf{D}$, *MaxIter*)

$w_d \leftarrow 0$, for all $d = 1 \dots D$ — // initialize weights
$b \leftarrow 0$ — // initialize bias
for $iter = 1 \dots MaxIter$ **do**
  for all $(x,y) \in \mathbf{D}$ **do**
    $a \leftarrow \sum_{d=1}^{D} w_d\, x_d + b$ — // compute activation for this example
    if $ya \leq 0$ **then**
      $w_d \leftarrow w_d + yx_d$, for all $d = 1 \dots D$ — // update weights
      $b \leftarrow b + y$ — // update bias
    end if
  end for
end for
return $w_0, w_1, \dots, w_D, b$

Hyper-parameters?

Algorithm 6 PERCEPTRONTEST($w_0, w_1, \dots, w_D, b, \hat{x}$)

$a \leftarrow \sum_{d=1}^{D} w_d\, \hat{x}_d + b$ — // compute activation for the test example
return SIGN(a)



Geometric Interpretation

Decision Boundary?

- Where the sign of the activation changes from -1 to +1.
- The set of points x that achieve zero activation.

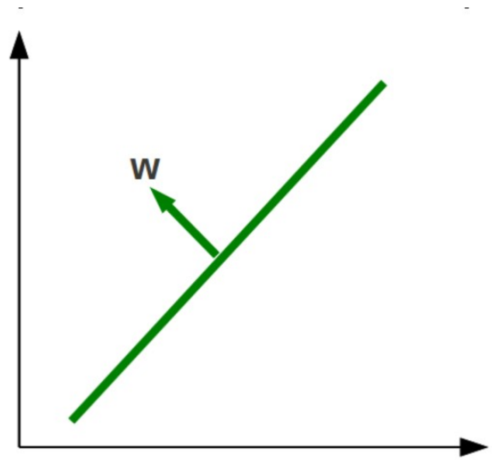
$$\sum_d w_d x_d = 0$$

$$\vec{w}\vec{x} = 0$$

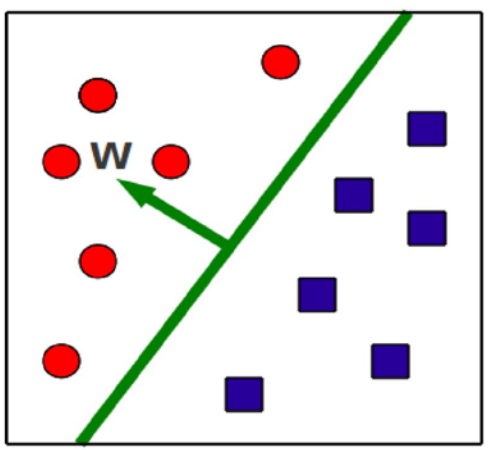
- Two vectors have a zero dot product if and only if they are perpendicular.

Decision Boundary

- The decision boundary is simply the *hyperplane perpendicular to w*.

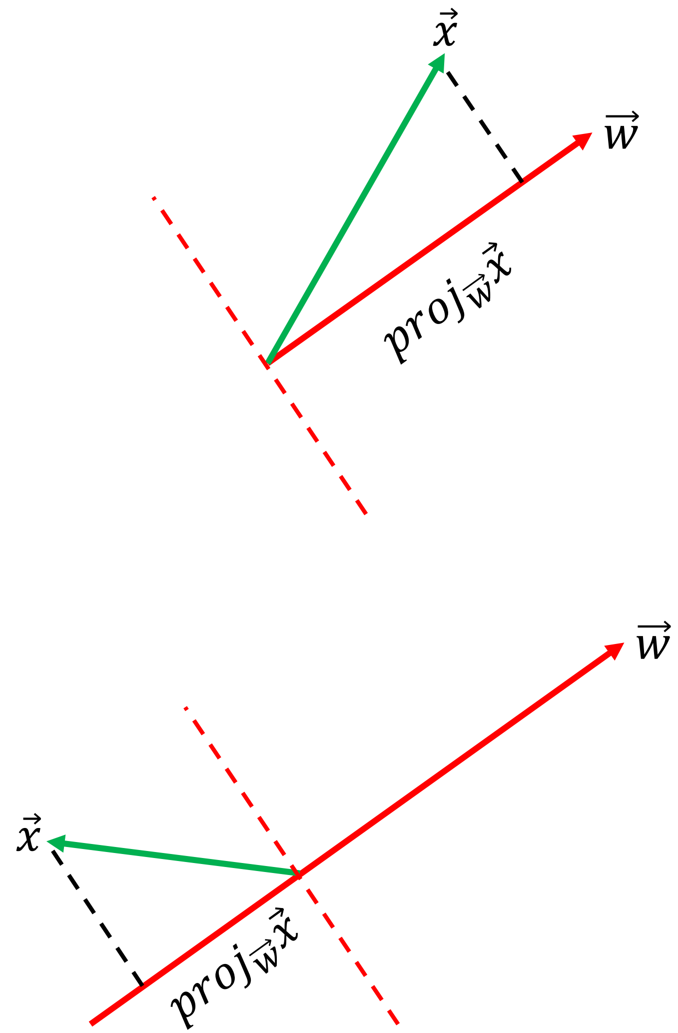


Training consists of finding a hyperplane *w* that separates +ve from -ve examples.



At **test** time, check what side of the hyperplane examples fall

Dot Product as a Projection

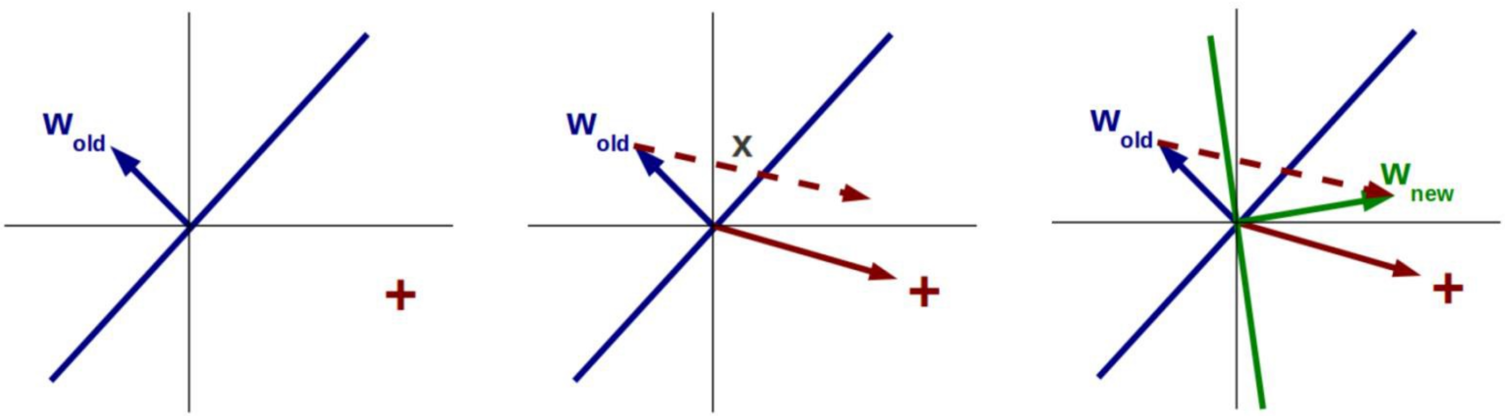


What happens with the bias b?

Perceptron Update

Update for a misclassified positive example:

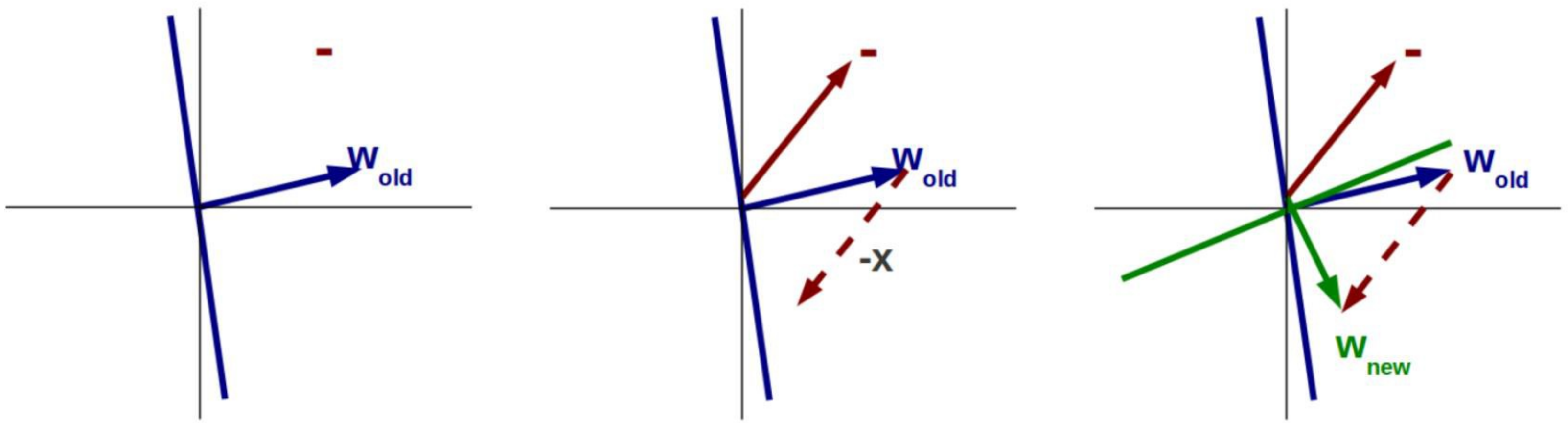
$$\mathbf{w}_{new} = \mathbf{w}_{old} + \mathbf{x}$$



Perceptron Update

Update for a misclassified negative example:

$$\mathbf{w}_{new} = \mathbf{w}_{old} - \mathbf{x}$$



Function Approx. with Perceptron

Problem setting

- Set of possible instances X
  - Each instance $x \in X$ is a feature vector $x = [x_1, x_2, \dots, xD]$
- Unknown target function f^*:$X \rightarrow Y$
  - Y is binary valued {-1; +1}
- Set of function hypotheses H={h | h: $X \rightarrow Y$}
  - Each hypothesis h is a *hyperplane* in D-dimensional space

Input

- Training examples {($x^{(1)}$,$y^{(1)}$),…($x^{(N)}$,$y^{(N)}$)} of unknown target function f^*

Output

- Hypothesis $h \in H$ that best approximates target function f^*

Practical Considerations

Practical Considerations

- The order of training examples matters!
  - Random is better
- Early stopping
  - Good strategy to avoid overfitting
- Simple modifications dramatically change performance
  - voting or averaging

Voted Perceptron

- Predict based on final + intermediate parameters

$$\hat{y} = \text{sign}\left(\sum_{k=1}^{K} c^{(\text{k})}\text{sign}\left(\boldsymbol{w}^{(\text{k})} \cdot \hat{\boldsymbol{x}} + b^{(\text{k})}\right)\right)$$

- Requires keeping track of previous weight vectors and their “survival times” $c^{(1)}, \ldots, c^{(\text{K})}$

Why is that a problem?

Averaged Perceptron

$$\hat{y} = \text{sign}\left(\sum_{k=1}^{K} c^{(k)}\left(\boldsymbol{w}^{(k)} \cdot \hat{\boldsymbol{x}} + b^{(k)}\right)\right)$$

- can be rewritten as

$$\hat{y} = \text{sign}\left(\left(\sum_{k=1}^{K} c^{(k)}\boldsymbol{w}^{(k)}\right) \cdot \hat{\boldsymbol{x}} + \sum_{k=1}^{K} c^{(k)} b^{(k)}\right)$$

Does that solve the problem?

Convergence of Perceptron

- Does the perceptron converge?
- If so, what does it converge to?
- How long does it take?

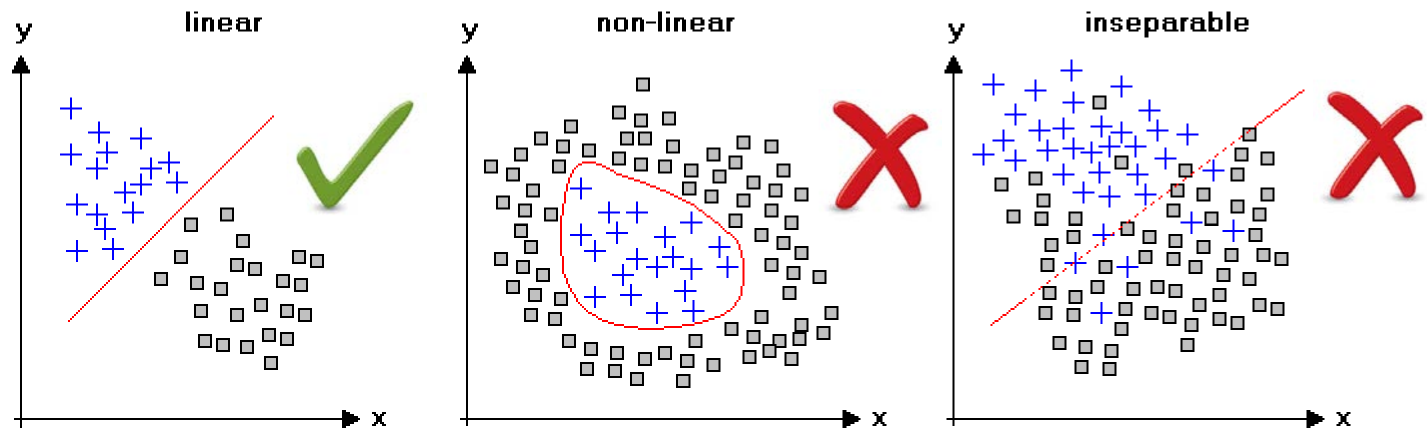
Convergence?

- Can make an entire pass through the training data without making any more updates, i.e., correctly classified every training example.
- Geometrically: found a hyperplane correctly separating data into positive and negative examples

Can the perceptron always find a hyperplane to separate positive from negative examples?

Convergence Condition

- For convergence, data has to be **linearly separable**!



- if data is linearly-separable, then it will converge to a weight vector that separates the data.
- If data is **linearly inseparable**, then the perceptron will **never** converge!
  - It could never possibly classify each point correctly.

How long does it take to converge?

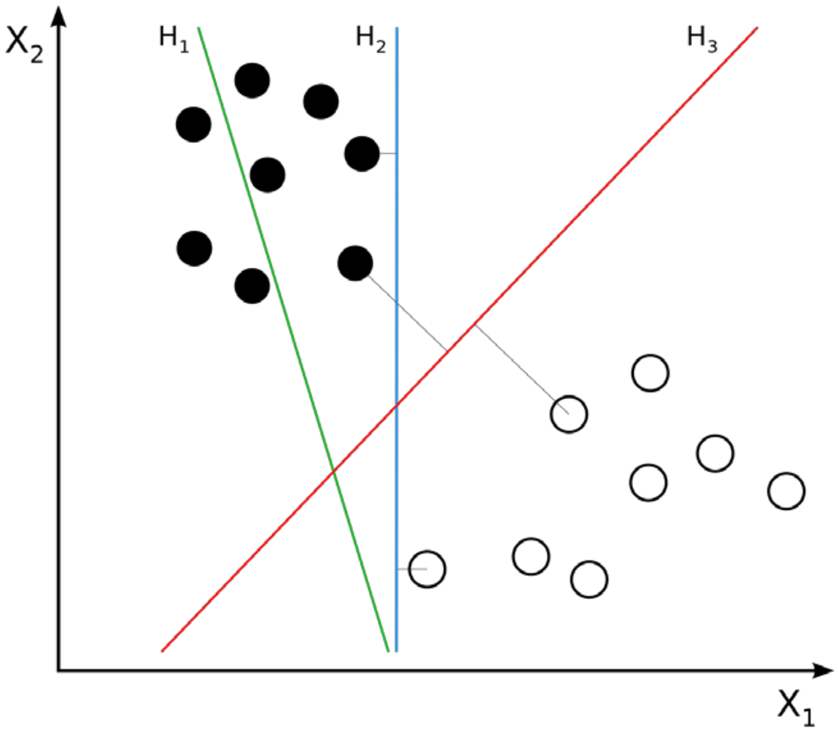
Margin of a dataset D

$$margin(\mathbf{D}, w, b) = \begin{cases} \min_{(x,y)\in\mathbf{D}} y(w \cdot x + b) & \text{if } w \text{ separates } \mathbf{D} \\ -\infty & \text{otherwise} \end{cases}$$

Distance between the hyperplane (w, b) and the nearest point in **D**

$$margin(\mathbf{D}) = \sup_{w,b} margin(\mathbf{D}, w, b)$$

Largest attainable margin on **D**



https://en.wikipedia.org/wiki/Margin_(machine_learning)

Perceptron Convergence Theorem (Rosenblatt, 1958)

Theorem 2 (Perceptron Convergence Theorem). *Suppose the perceptron algorithm is run on a linearly separable data set* $\mathbf{D}$ *with margin* $\gamma > 0$. *Assume that* $||\boldsymbol{x}|| \leq 1$ *for all* $\boldsymbol{x} \in \mathbf{D}$. *Then the algorithm will converge after at most* $\frac{1}{\gamma^2}$ *updates.*

i.e., number of errors that the perceptron algorithm makes in this case is bounded by $1/\gamma^2$.

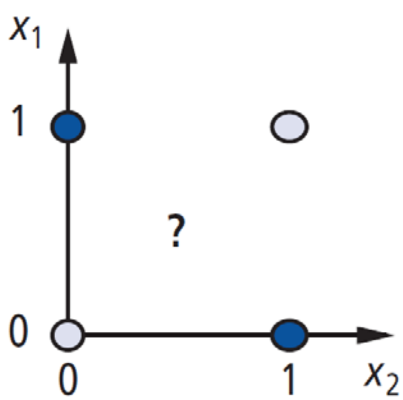
What does this mean?

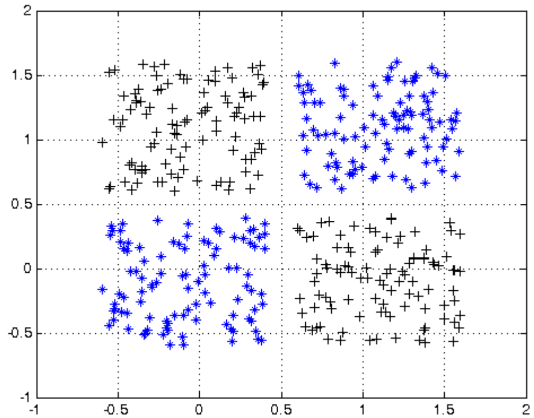
- Perceptron converges quickly when margin is large, slowly when it is small.
- Bound does not depend on number of training examples nor on number of features.
- Proof guarantees that perceptron converges, but not necessarily to the max margin separator!

What if the data is not linearly separable due to noise?

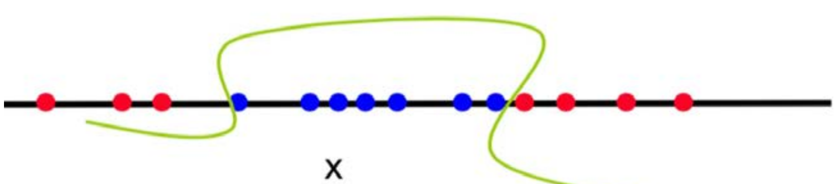
Limitations?

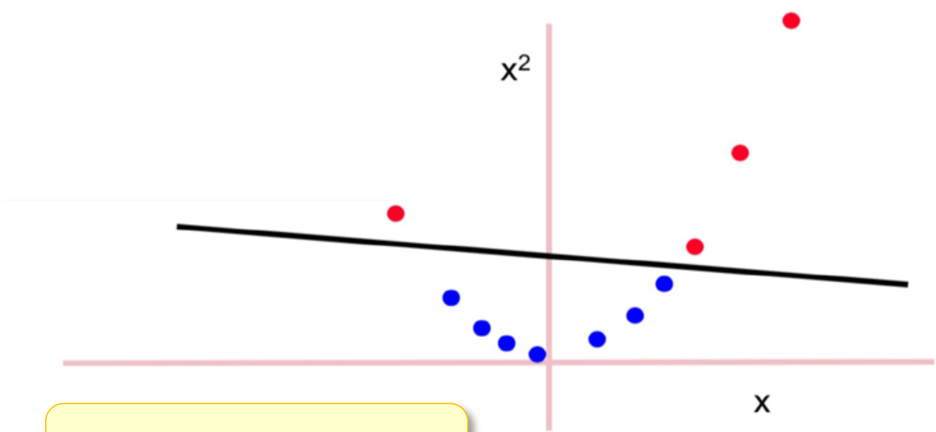
- Decision boundaries can only be linear!





- Add feature combinations (**feature mapping**)?





Problem?

Limitations?

- Two other approaches:
  1. combine multiple perceptrons (**neural networks)**
  2. find computationally efficient ways of feature mapping (**kernels)**