# The Perceptron

CMSC 422

MARINE CARPUAT
marine@cs.umd.edu

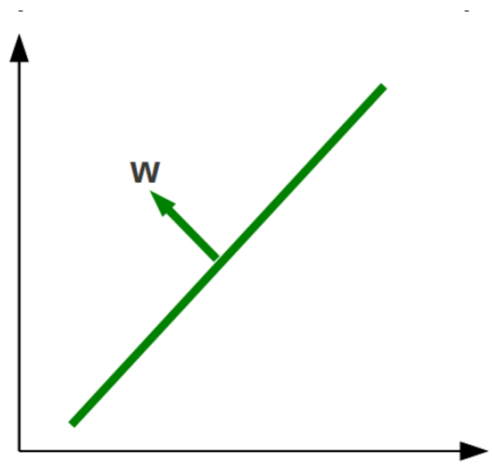
Credit: figures by Piyush Rai and Hal Daume III

# This week

- A new model/algorithm
  - the perceptron
  - and its variants: voted, averaged
- Fundamental Machine Learning Concepts
  - Online vs. batch learning
  - Error-driven learning

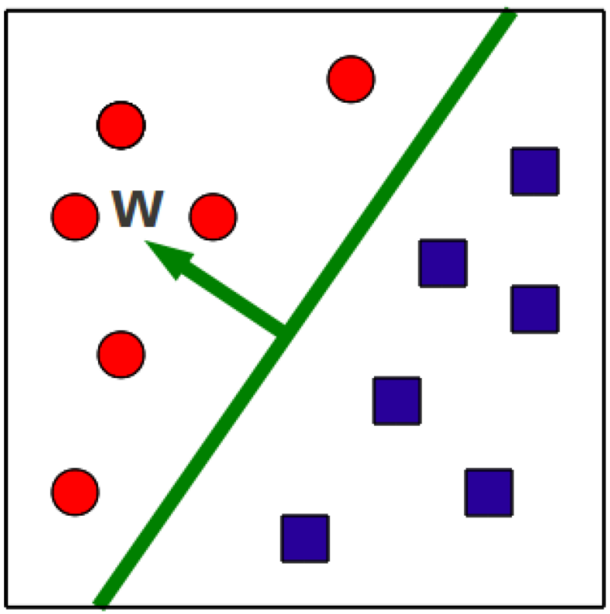
- Project 1 coming soon!

# Geometry concept: **Hyperplane**



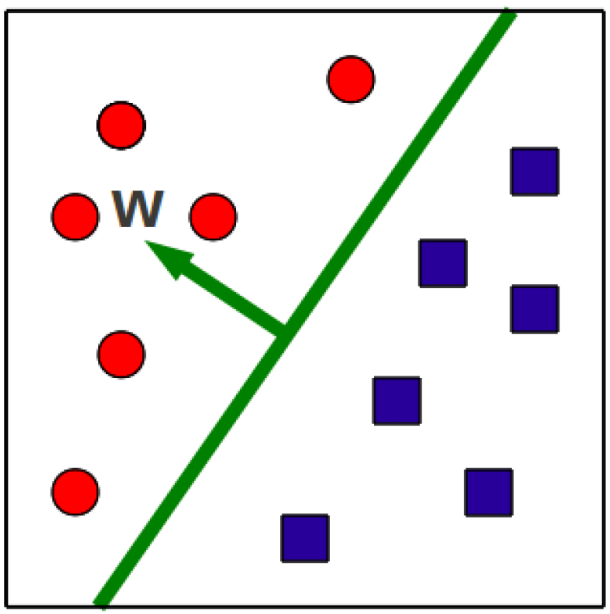

- Separates a D-dimensional space into two half-spaces
- Defined by an outward pointing normal vector $w \in \mathbb{R}^D$
  - $w$ is **orthogonal** to any vector lying on the hyperplane
- Hyperplane passes through the origin, unless we also define a **bias** term b

# Binary classification via hyperplanes




- Let's assume that the decision boundary is a hyperplane
- Then, training consists in finding a hyperplane $w$ that separates positive from negative examples

# Binary classification via hyperplanes




- At test time, we check on what side of the hyperplane examples fall

$$\hat{y} = sign(w^T x + b)$$

# Function Approximation with Perceptron

Problem setting

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

Input

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

Output

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

# Perception: Prediction Algorithm

---

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

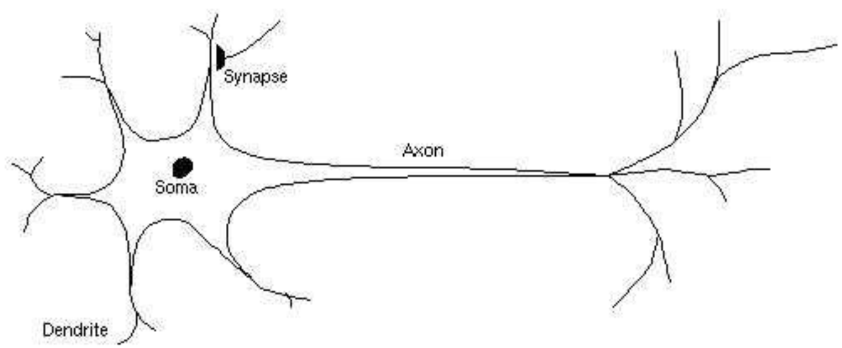
---

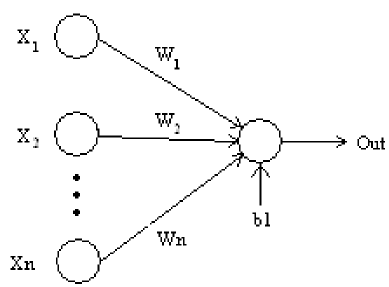
1: $a \leftarrow \sum_{d=1}^{D} w_d \, \hat{x}_d + b$ // compute activation for the test example

2: **return** sign($a$)

---

# Aside: biological inspiration



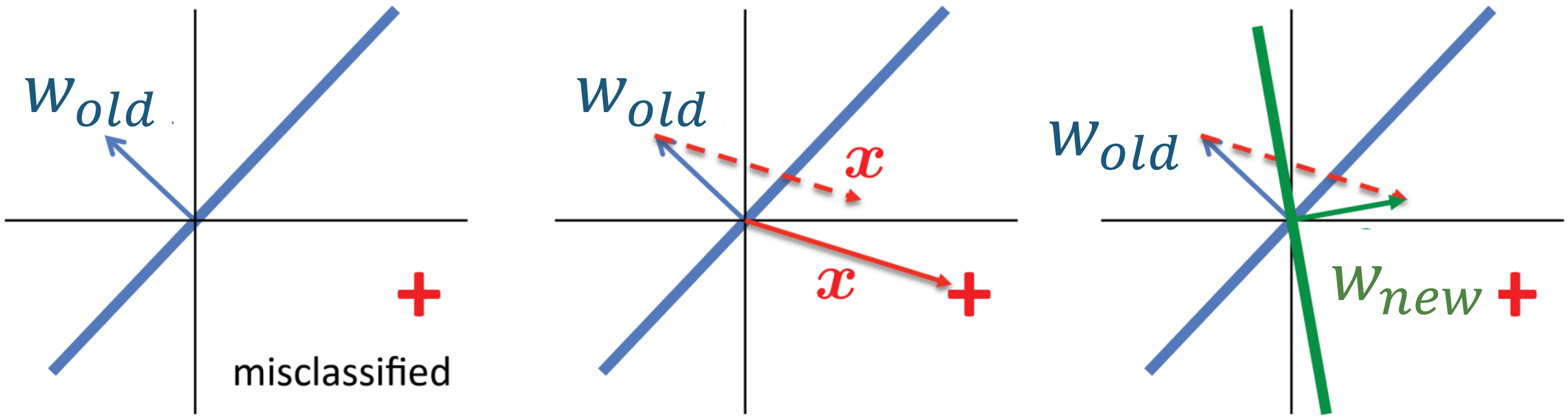

Analogy: the perceptron as a neuron

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

# Perceptron update: geometric interpretation



# Properties of the Perceptron training algorithm

- Online
  - We 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

# Practical considerations

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