The Perceptron

CMSC 422

SOHEIL FEIZI

sfeizi@cs.umd.edu

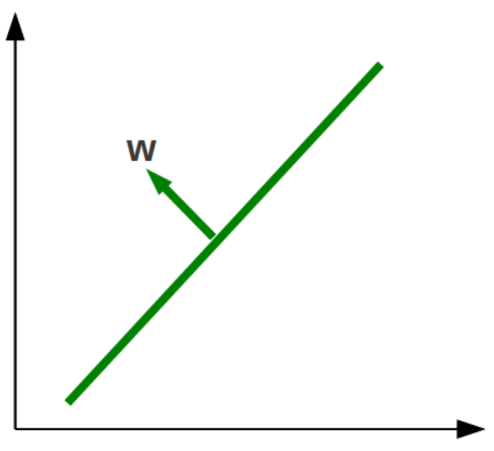
Slides adapted from MARINE CARPUAT

This week

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

- HW3 will be posted this week.

Geometry concept: **Hyperplane**

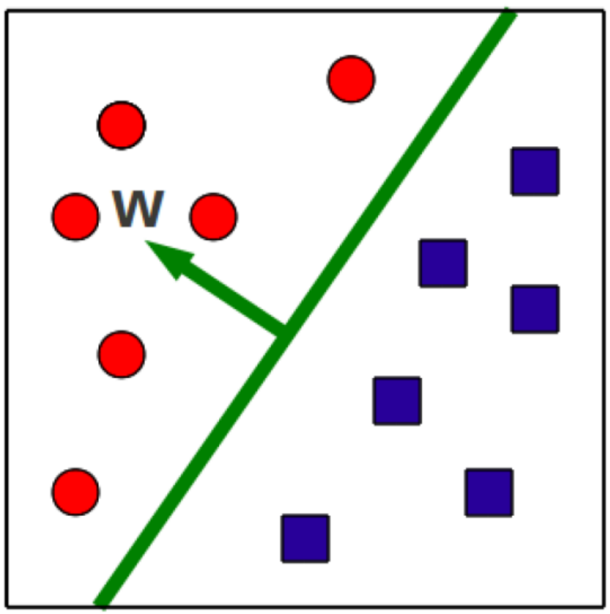


- Separates a D-dimensional space into two half-spaces

- Defined by an outward pointing normal vector $w \in \mathbb{R}^D$
  - w is **orthogonal** to any vector lying on the hyperplane

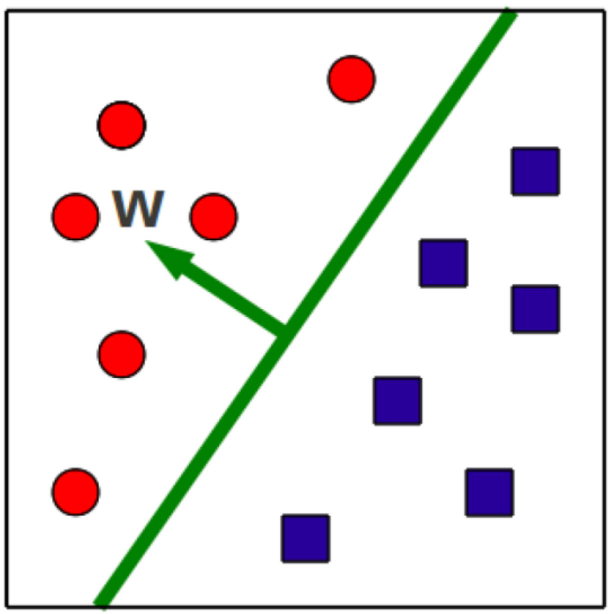
- Hyperplane passes through the origin, unless we also define a **bias** term b

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 $x = [x_1, \ldots, x_D]$
- 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 $\{(x^{(1)}, y^{(1)}), \ldots (x^{(N)}, y^{(N)})\}$ of unknown target function f

Output

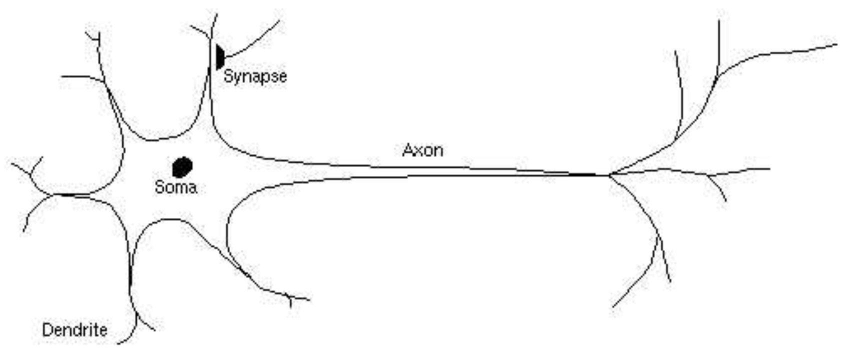
- Hypothesis $h \in H$ that best approximates target function f

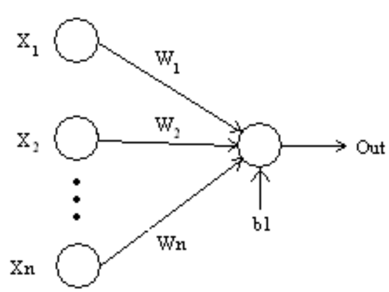
Perception: Prediction Algorithm

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

```
a ← ∑_{d=1}^{D} w_d x̂_d + b        // compute activation for the test example
return sign(a)
```

Aside: biological inspiration





Analogy: the perceptron as a neuron

Perceptron Training Algorithm

Algorithm 5 PerceptronTrain(**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

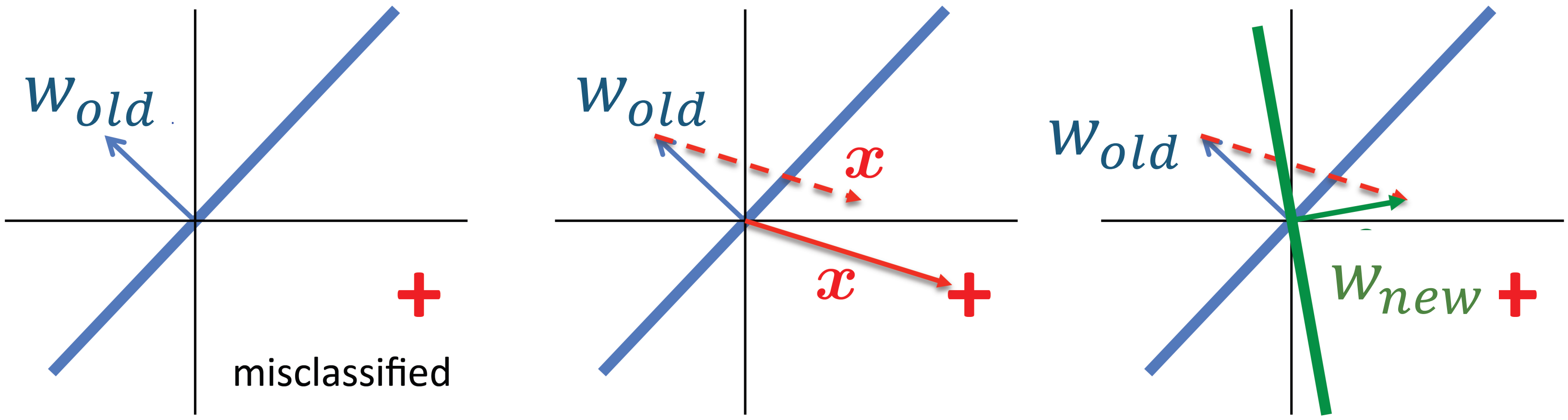
return $w_0, w_1, \dots, w_D, b$

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

Perceptron update: geometric interpretation



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

Standard Perceptron: predict based on final parameters

Algorithm 5 PerceptronTrain(**D**, *MaxIter*)

```
w_d ← 0, for all d = 1 ... D                         // initialize weights
b ← 0                                                // initialize bias
for iter = 1 ... MaxIter do
    for all (x,y) ∈ D do
        a ← ∑_{d=1}^{D} w_d x_d + b                  // compute activation for this example
        if ya ≤ 0 then
            w_d ← w_d + y x_d, for all d = 1 ... D   // update weights
            b ← b + y                                // update bias
        end if
    end for
end for
return w_0, w_1, ..., w_D, b
```

Predict based on final + intermediate parameters

- The voted perceptron

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

- The averaged perceptron

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

- Require keeping track of "survival time" of weight vectors $c^{(1)}, \ldots, c^{(K)}$

Averaged perceptron decision rule

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

can be rewritten as

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

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

This week

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

- HW3 coming soon!