Slides adapted from Prof. Carpuat

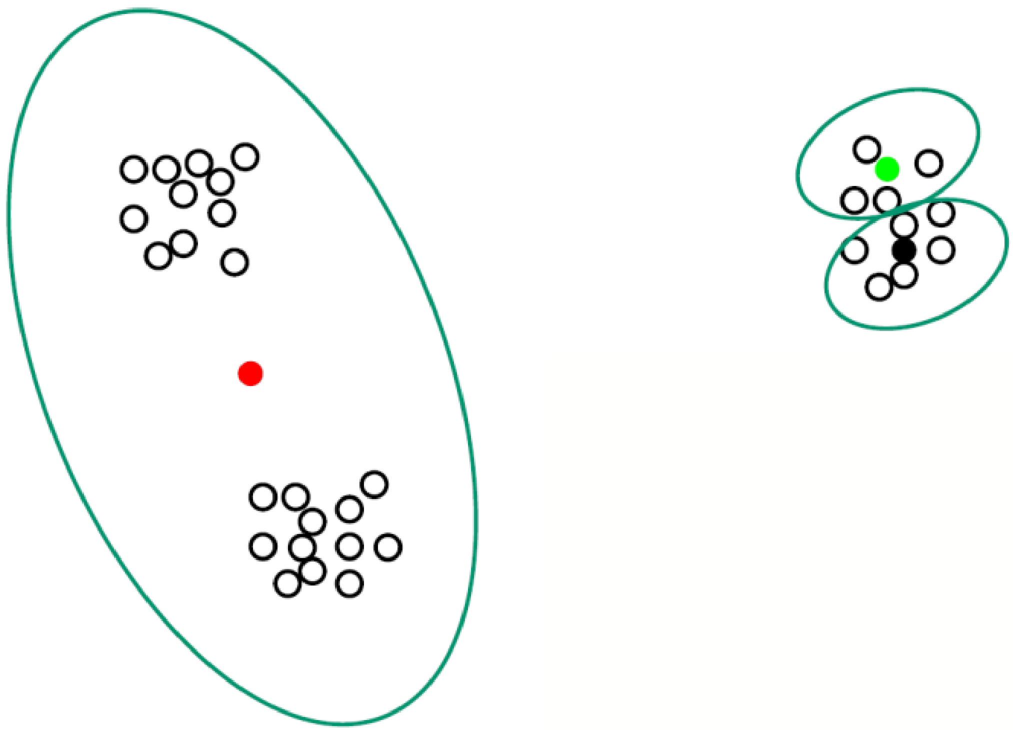
CMSC 422 Introduction to Machine Learning

Lecture 6 The Perceptron

Furong Huang / furongh@cs.umd.edu

UNIVERSITY OF MARYLAND

Impact of initialization



Optimization View of K-means

Given a set of observations $(x_1, x_2, \ldots, x_n)$ where each observation is a d-dimensional real vector

k-means clustering aims to partition the n observations into $\mathrm{k}(\leq x)$ sets $S = \{S_1, S_2, \ldots, S_\mathrm{k}\}$ so as to minimize the within-cluster sum of squares.

Formally, the objective is to find:

$$\arg\min_S \sum_{i=1}^{k} \sum_{x \in S_i} \|x - \mu_i\|^2 = \arg\min_S \sum_{i}^{k} |S_i| \operatorname{Var}(S_i)$$

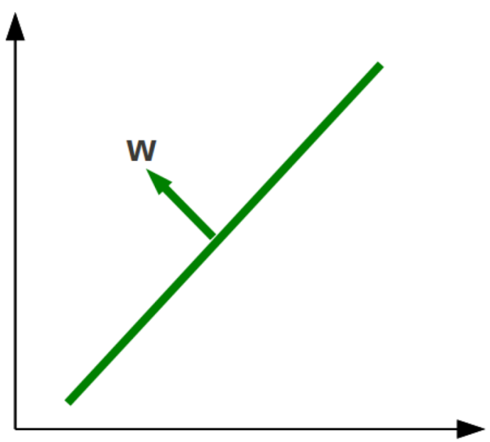
where μ_i is the mean of points in S_i.

This week

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

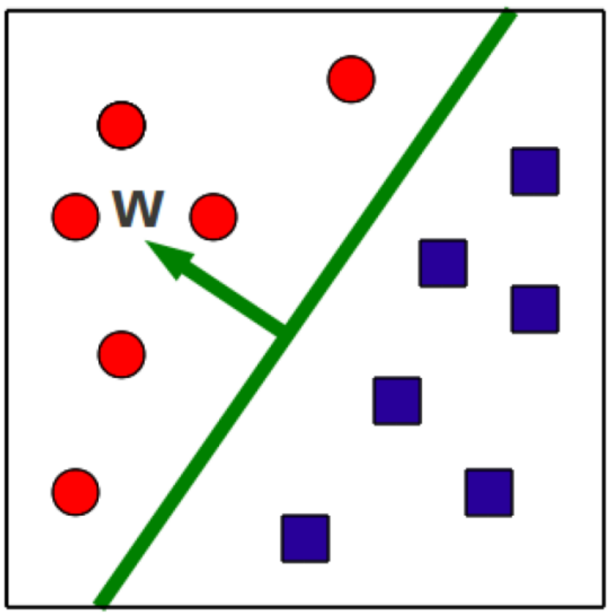
- Project 1 coming soon!

Geometry concept: Hyperplane



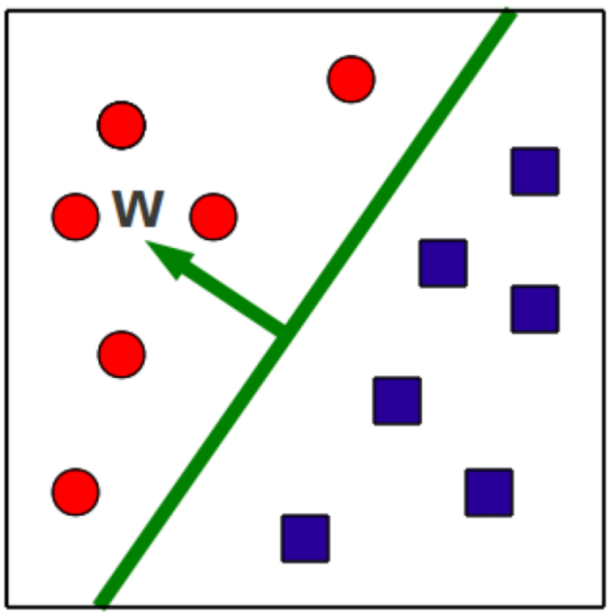
- Separates a D-dimensional space into two half-spaces
- Defined by an outward pointing normal vector $w \in \mathbb{R}^D$
  - w is **orthogonal** to any vector lying on the hyperplane
- Hyperplane passes through the origin, unless we also define a **bias** term b

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 $x = [x_1, \ldots, x_D]$
- Unknown target function $f: X \rightarrow Y$
  - Y is binary valued {-1; +1}
- Set of function hypotheses $H = \{h \mid h: X \rightarrow Y\}$
  - **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

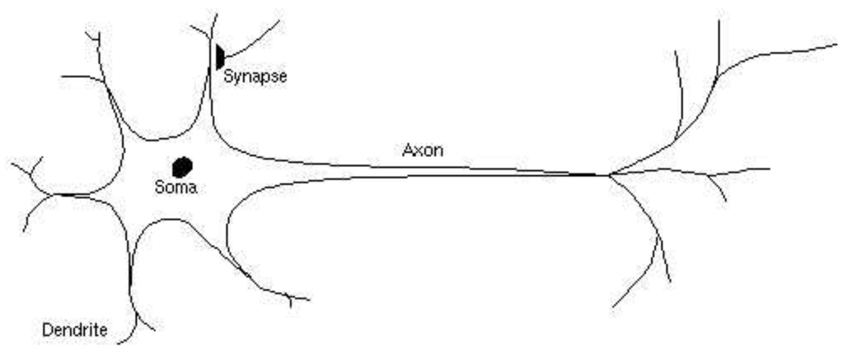
Perception: Prediction Algorithm

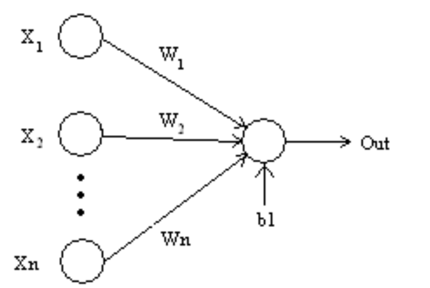
Algorithm 6 $\textsc{PerceptronTest}(w_0, w_1, \dots, w_D, b, \hat{\boldsymbol{x}})$

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

2: **return** $\textsc{sign}(a)$

Aside: biological inspiration





Analogy: the perceptron as a neuron

Perceptron Training Algorithm

Algorithm 5 $\textsc{PerceptronTrain}(\mathbf{D}, \mathit{MaxIter})$

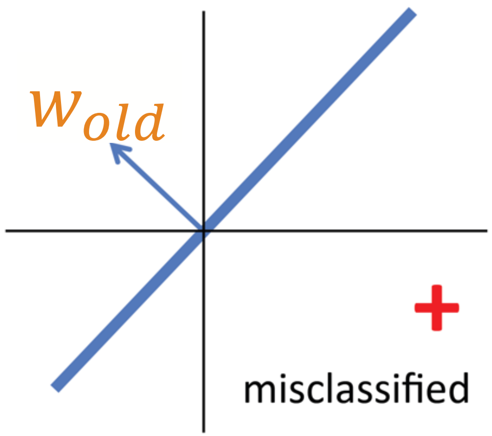
$w_d \leftarrow 0$, for all $d = 1 \dots D$ // initialize weights
$b \leftarrow 0$ // initialize bias
for $\mathit{iter} = 1 \dots \mathit{MaxIter}$ **do**
  for all $(x,y) \in \mathbf{D}$ **do**
    $a \leftarrow \sum_{d=1}^{D} w_d\, x_d + b$ // compute activation for this example
    if $ya \leq 0$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

Perceptron update: geometric interpretation

A training example $(\boldsymbol{x}, y)$ is misclassified, i.e.,

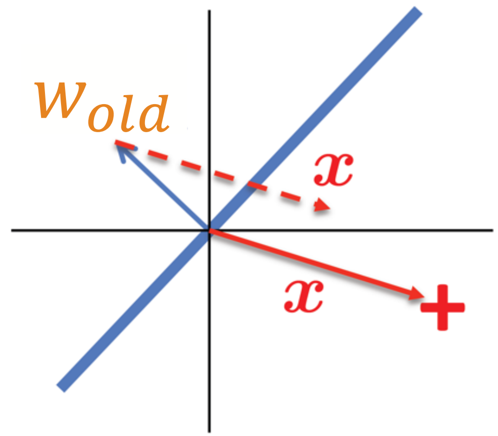
$$sign\left(\boldsymbol{w}_{old}^{\top}\boldsymbol{x} + b\right) \neq y$$

Let's say $y = +1$



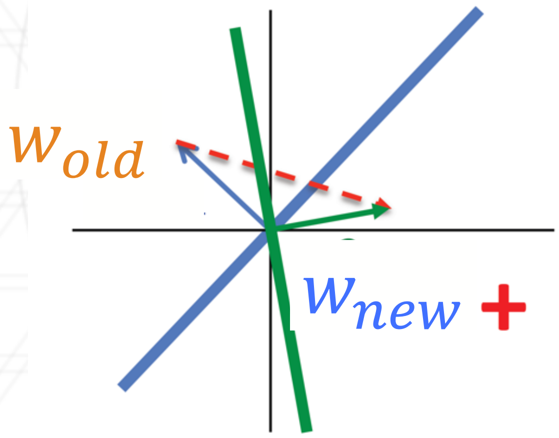
Perceptron update: geometric interpretation

Update: $\boldsymbol{w}_{new} \leftarrow \boldsymbol{w}_{old} + y\boldsymbol{x}$, i.e., $\boldsymbol{w}_{new} \leftarrow \boldsymbol{w}_{old} + \boldsymbol{x}$



Perceptron update: geometric interpretation

Update: $\boldsymbol{w}_{new} \leftarrow \boldsymbol{w}_{old} + y\boldsymbol{x}$, i.e., $\boldsymbol{w}_{new} \leftarrow \boldsymbol{w}_{old} + \boldsymbol{x}$



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

Predicting with

- 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

$$c^{(1)}, \ldots, c^{(K)}$$

How would you modify this algorithm for voted perceptron?

Algorithm 5 $\textsc{PerceptronTrain}(\mathbf{D}, \mathit{MaxIter})$

$w_d \leftarrow 0$, 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 $w_0, w_1, \dots, w_D, b$

How would you modify this algorithm for averaged perceptron?

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

```
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 ← Σ_{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
```

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

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

Averaged Perceptron Training

Algorithm 7 AveragedPerceptronTrain($\mathbf{D}$, *MaxIter*)

$\boldsymbol{w} \leftarrow \langle 0, 0, \dots 0 \rangle \quad , \quad b \leftarrow 0$ — // initialize weights and bias

$\boldsymbol{u} \leftarrow \langle 0, 0, \dots 0 \rangle \quad , \quad \beta \leftarrow 0$ — // initialize cached weights and bias

$c \leftarrow 1$ — // initialize example counter to one

for $iter = 1 \dots MaxIter$ **do**

    for all $(\boldsymbol{x}, y) \in \mathbf{D}$ **do**

        if $y(\boldsymbol{w} \cdot \boldsymbol{x} + b) \leq 0$ **then**

            $\boldsymbol{w} \leftarrow \boldsymbol{w} + y\,\boldsymbol{x}$ — // update weights

            $b \leftarrow b + y$ — // update bias

            $\boldsymbol{u} \leftarrow \boldsymbol{u} + y\,c\,\boldsymbol{x}$ — // update cached weights

            $\beta \leftarrow \beta + y\,c$ — // update cached bias

        end if

        $c \leftarrow c + 1$ — // increment counter regardless of update

    end for

end for

return $\boldsymbol{w} - \frac{1}{c}\,\boldsymbol{u},\ b - \frac{1}{c}\,\beta$ — // return averaged weights and bias

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

Project 1 coming soon!

Furong Huang
3251 A.V. Williams, College Park, MD 20740
301.405.8010 / furongh@cs.umd.edu