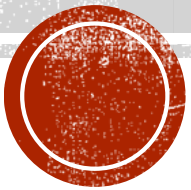
THE PERCEPTRON



CS 412 Introduction to Machine Learning

Prof. Zheleva

February 1, 2018

Reading Assignment: CIML: 4

LAST LECTURE: NEAREST NEIGHBORS

- From real-world problem to data to model to solution
- Using geometry for classification
  - Curse of dimensionality
- kNN: Very simple non-linear classification technique
  - Works well with large amounts of data
  - Related to very sophisticated machine learning technique: kernel methods!
- Decision boundaries

FROM DATA TO ML SOLUTION

Why one of Africa's worst conflicts is getting worse

By Zack Beauchamp | @zackbeauchamp | zack@vox.com | Apr 12, 2014, 11:00am EDT

SHARE MORE

CAR civilians besieged by the anti-balaka militias take shelter under French military protection. | Miguel Medina/AFP/Getty Images

The conflict in the Central African Republic (CAR) is a huge international issue — thousands of people have been killed since it began in December 2012, and its worsening rapidly enough that the UN has just greenlit a

Input: x

Output: y

dsjVoxArticles.tsv

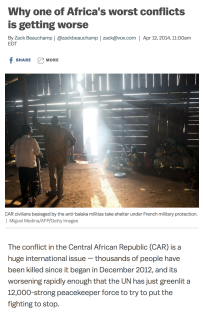
Download

	title	author	category
19	This slide is the best thing to come out of the Apple v. Samsung trial	Nilay Patel	Apple
20	40 states relaxed their drug laws in the past 5 years	German Lopez	Criminal Justice
21	The Dow topped 18,000 today. Learn why that matters.	Danielle Kurtzleben	Business & Finance
22	Statistical controls tell us how the gender pay gap works, not that it isn'	Matthew Yglesias	Politics & Policy
23	Don't panic yet, but early Obamacare enrollees might cost more	German Lopez	Health Care
24	The best evidence we have that Obamacare is working	Ezra Klein	Health Care
25	How does one doctor earn $21 million from Medicare?	Sarah Kliff	Health Care
26	Teens are shockingly great at using birth control	Sarah Kliff	Health Care
27	Some of you are lying about going to church	Brandon Ambrosino	Culture
28	How does one pill cost $1,000?	German Lopez	Health Care
29	HHS Secretary Sebelius to resign	Sarah Kliff	Health Care
30	Kathleen Sebelius is resigning because Obamacare has won	Ezra Klein	Health Care
31	Why one of Africa's worst conflicts is getting worse	Zack Beauchamp	Human Rights
32	Obama wants to fight gerrymandering once he leaves office. He should look t	Andrew Prokop	The Latest
33	5 disease outbreaks happening right now that vaccines could have prevented	German Lopez	Health Care
34	Do e-cigarettes kill you? Researchers donâ€™t know	German Lopez	Health Care
35	The $2.8 trillion question: Are health costs growing fast again?	Sarah Kliff	Health Care

Source: https://data.world/elenadata/vox-articles

GEOMETRIC VIEW OF DATA

- Each feature is a dimension
- The feature values of each data instance correspond to a point in n-dimensional space where n is the number of features



Features: bag of words

Feature values: presence of a word

aardvark	africa	beach	conflict	…	worse	zebra
0	1	0	1	…	1	0

Feature values: count of a word

aardvark	africa	beach	conflict	…	worse	zebra
0	5	0	2	…	1	0

COMPONENTS OF A K-NN CLASSIFIER

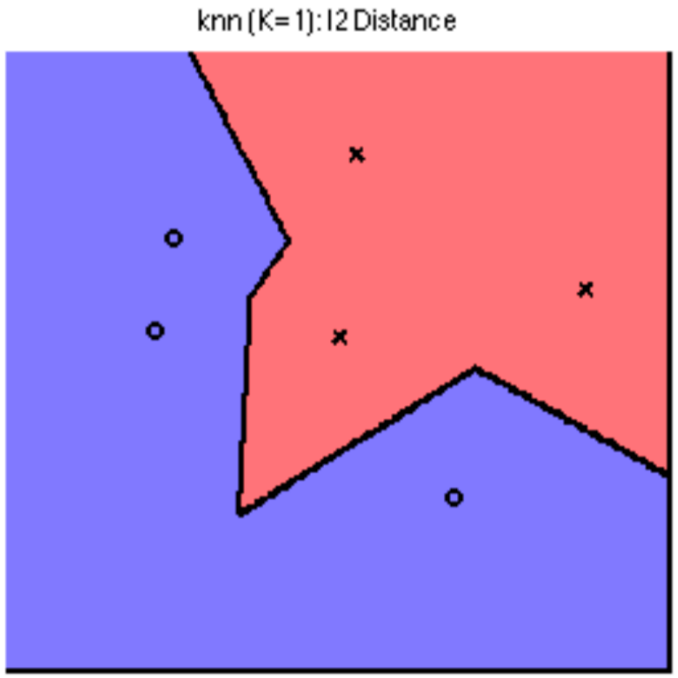
- Distance metric
  - How do we measure distance between instances
  - Determines the layout of the example space
- The k hyperparameter
  - How large a neighborhood should we consider?
  - Determines the complexity of the hypothesis space

DECISION BOUNDARY OF A CLASSIFIER

- The line that separates positive and negative regions in the feature space

- Why is it useful?
  - It helps us visualize how examples will be classified for the entire feature space
  - It helps us visualize the complexity of the learned model

- Decision boundary for 1NN →



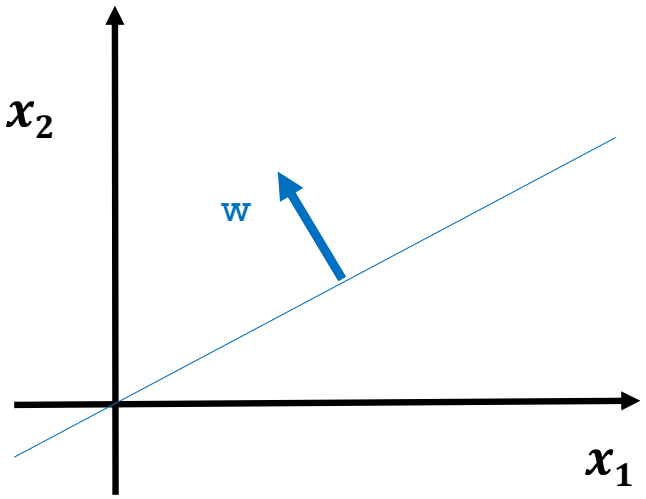
TODAY: THE PERCEPTRON

- The perceptron: a new model and algorithm
  - Its variants: voted, averaged
  - Convergence proof

- Fundamental machine learning concepts
  - Online vs. batch learning
  - Error-driven learning
  - Linear separability and margin of a dataset

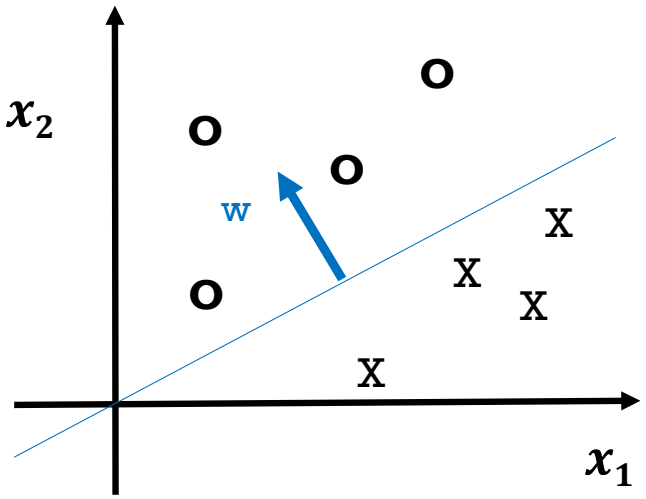
GEOMETRY CONCEPT: HYPERPLANE

- 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

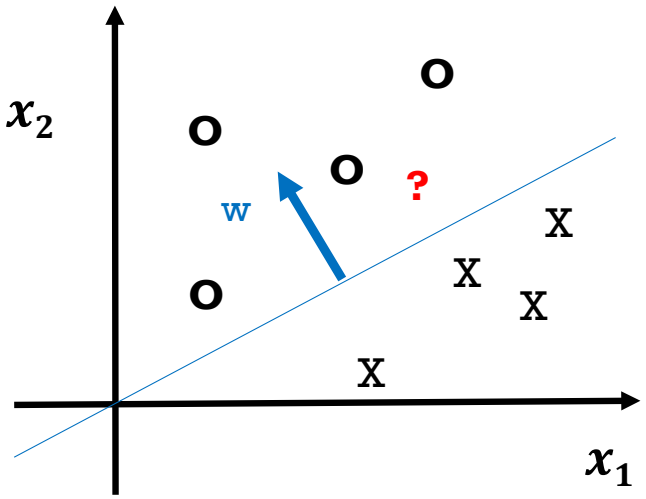
- Let's assume that the decision boundary is a hyperplane
- Then, training consists in finding a hyperplane *w* that separates positive from negative examples



BINARY CLASSIFICATION VIA HYPERPLANES

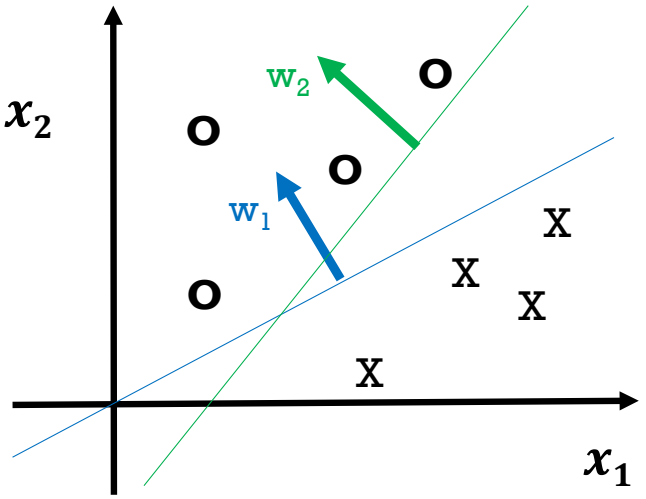
- At test time, we check on what side of the hyperplane examples fall
- Classifier = hyperplane that separates positive from negative examples

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



FUNCTION APPROXIMATION WITH PERCEPTRON

- Problem setting
  - **X** – set of possible instances
    - Each instance $x \in X$ is a feature vector $x = [x_1, \ldots, x_D]$
  - Unknown target function f: **X** $\rightarrow$ Y
    - Classification: Y is binary valued
  - Set of function hypotheses H={h | h: **X** $\rightarrow$ Y}
    - Each hypothesis h is a hyperplane in D-dimensional space



- Input
  - Training examples $\{(x^{(1)}, y^{(1)}), \ldots (x^{(N)}, y^{(N)})\}$ of unknown distribution

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

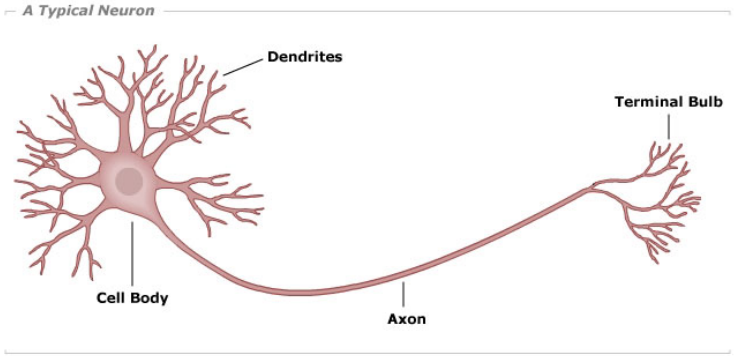
PERCEPTRON: PREDICTION ALGORITHM

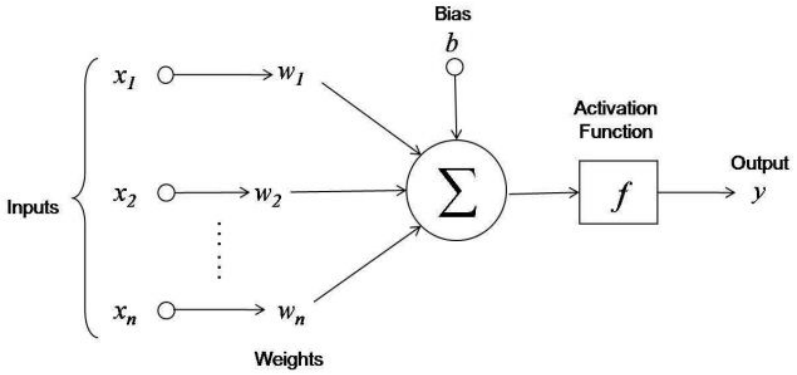
Algorithm 6 PERCEPTRONTEST($w_0, w_1, \ldots, w_D, b, \hat{\boldsymbol{x}}$)

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

- The perceptron as a neuron
  - Can receive both stimulatory and inhibitory signals
  - If adequate stimulus, it produces a nerve impulse





$$a \leftarrow \sum_{d=1}^{D} w_d \, x_d + b$$

PERCEPTRON: TRAINING ALGORITHM

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

```
1:  w_d ← 0, for all d = 1 ... D                    // initialize weights
2:  b ← 0                                           // initialize bias
3:  for iter = 1 ... MaxIter do
4:      for all (x,y) ∈ D do
5:          a ← Σ_{d=1}^{D} w_d x_d + b             // compute activation for this example
6:          if ya ≤ 0 then
7:              w_d ← w_d + y x_d, for all d = 1 ... D   // update weights
8:              b ← b + y                           // update bias
9:          end if
10:     end for
11: end for
12: return w_0, w_1, ..., w_D, b
```

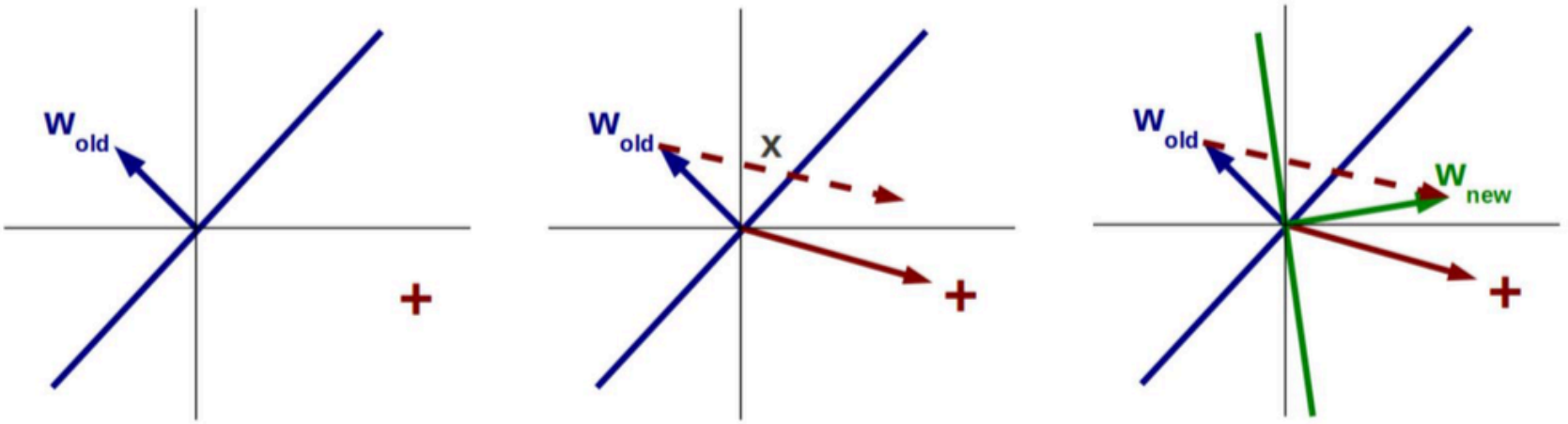
epoch

Non-obvious why this update works!

PERCEPTRON UPDATE: GEOMETRIC INTERPRETATION

Update for a misclassified positive example (assume b=0):

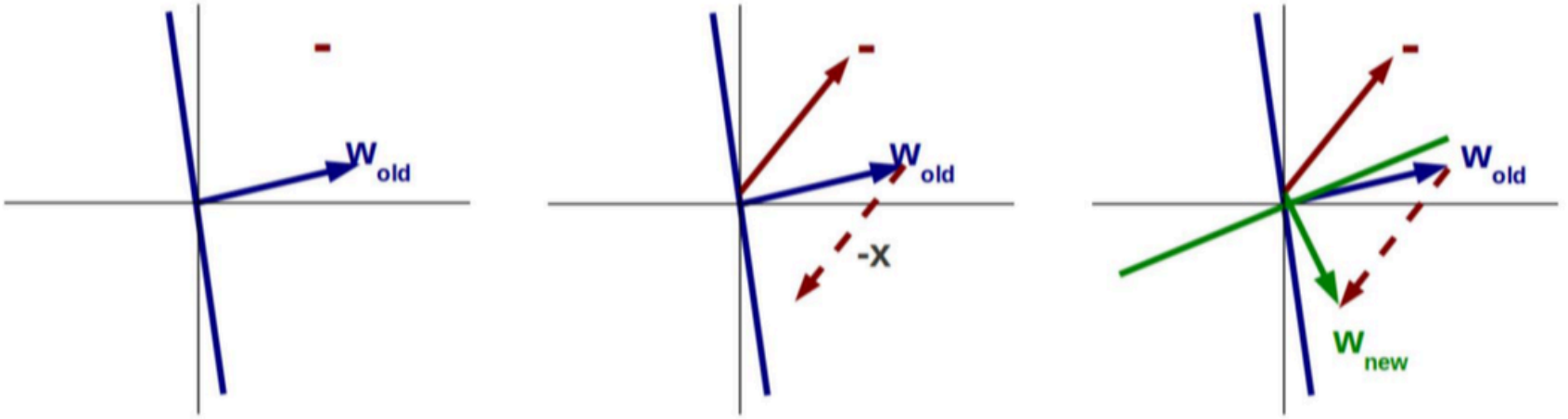
$$\mathbf{w}_{new} = \mathbf{w}_{old} + \mathbf{x}$$



PERCEPTRON UPDATES

Update for a misclassified negative example:

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



PREDICTION VARIANTS

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

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

- The voted perceptron
  - Better than vanilla, can be slow

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

- The averaged perceptron

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

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

HOW WOULD YOU MODIFY THIS ALGORITHM FOR VOTED PERCEPTRON?

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

1: $w_d \leftarrow 0$, for all $d = 1 \ldots D$ // initialize weights
2: $b \leftarrow 0$ // initialize bias
3: **for** $iter = 1 \ldots MaxIter$ **do**
4: **for all** $(x,y) \in \mathbf{D}$ **do**
5: $a \leftarrow \sum_{d=1}^{D} w_d x_d + b$ // compute activation for this example
6: **if** $ya \leq 0$ **then**
7: $w_d \leftarrow w_d + yx_d$, for all $d = 1 \ldots D$ // update weights
8: $b \leftarrow b + y$ // update bias
9: **end if**
10: **end for**
11: **end for**
12: **return** $w_0, w_1, \ldots, w_D, b$

HOW WOULD YOU MODIFY THIS ALGORITHM FOR AVERAGED PERCEPTRON?

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

1: $w_d \leftarrow 0$, for all $d = 1 \ldots D$ // initialize weights
2: $b \leftarrow 0$ // initialize bias
3: **for** $iter = 1 \ldots MaxIter$ **do**
4: **for all** $(x,y) \in \mathbf{D}$ **do**
5: $a \leftarrow \sum_{d=1}^{D} w_d x_d + b$ // compute activation for this example
6: **if** $ya \leq 0$ **then**
7: $w_d \leftarrow w_d + yx_d$, for all $d = 1 \ldots D$ // update weights
8: $b \leftarrow b + y$ // update bias
9: **end if**
10: **end for**
11: **end for**
12: **return** $w_0, w_1, \ldots, w_D, b$

AVERAGED PERCEPTRON TRAINING

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

can be re-written 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(**D**, *MaxIter*)

```
1:  w ← ⟨0, 0, ... 0⟩ , b ← 0                          // initialize weights and bias
2:  u ← ⟨0, 0, ... 0⟩ , β ← 0                          // initialize cached weights and bias
3:  c ← 1                                              // initialize example counter to one
4:  for iter = 1 ... MaxIter do
5:      for all (x,y) ∈ D do
6:          if y(w · x + b) ≤ 0 then
7:              w ← w + y x                            // update weights
8:              b ← b + y                              // update bias
9:              u ← u + y c x                          // update cached weights
10:             β ← β + y c                            // update cached bias
11:         end if
12:         c ← c + 1                                  // increment counter regardless of update
13:     end for
14: end for
15: return w - (1/c) u, b - (1/c) β                    // return averaged weights and bias
```

PERCEPTRON TRAINING ALGORITHM

- Online algorithm
  - We look at one example at a time, and update the model when 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!
  - Randomized is better
  - Permuting the order in every iteration makes learning much faster
- Early stopping
  - Good strategy to avoid overfitting
- Simple modifications dramatically improve performance
  - Voting or averaging

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

CONVERGENCE OF TRAINING

- The perceptron has converged if it can classify every training example correctly
  - i.e. if it has found a hyperplane that correctly separates positive and negative examples
- Under which conditions does the perceptron converge and how long does it take?

CONVERGENCE OF PERCEPTRON

Theorem (Block & Novikoff, 1962):

If the training data $\mathrm{D} = \{(\boldsymbol{x}^{(1)}, y^{(1)}), \ldots (\boldsymbol{x}^{(N)}, y^{(N)})\}$ is linearly separable with margin γ by a unit norm hyperplane w_*, assuming $\left|\left|\boldsymbol{x}^{(i)}\right|\right| \leq R$ for all i and b=0, then perceptron converges after $\frac{R^2}{\gamma^2}$ errors during training.

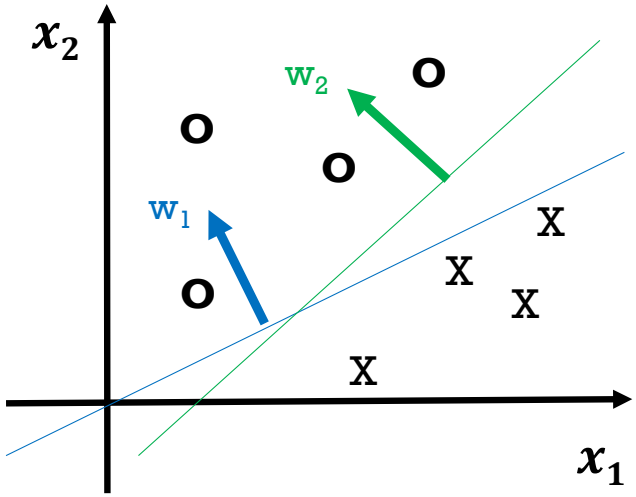
DATASET MARGIN

- Hyperplane margin on D: Distance between the hyperplane (w,b) and the nearest point in D

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

- Dataset margin: Largest attainable margin on D

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



CONVERGENCE OF PERCEPTRON

Theorem (Block & Novikoff, 1962):

If the training data $\mathrm{D} = \{(\boldsymbol{x}^{(1)}, y^{(1)}), \ldots (\boldsymbol{x}^{(N)}, y^{(N)})\}$ is linearly separable with margin γ by a unit norm hyperplane w_*, assuming $\left|\left|\boldsymbol{x}^{(i)}\right|\right| \leq R$ for all i and b=0, then perceptron converges after $\frac{R^2}{y^2}$ errors during training.

CONVERGENCE OF PERCEPTRON

Theorem (Block & Novikoff, 1962):

If the training data $D = \{(\boldsymbol{x}^{(1)}, y^{(1)}), \ldots (\boldsymbol{x}^{(N)}, y^{(N)})\}$ is linearly separable with margin γ by a unit norm hyperplane w_*, assuming $\left|\left|\boldsymbol{x}^{(i)}\right|\right| \leq R$ for all i and b=0, then perceptron converges after $\frac{R^2}{\gamma^2}$ errors during training.

What does it mean?

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

DECISION TREES VS. KNN VS. PERCEPTRON

Properties of classification problem	Decision trees	k-NN	Perceptron
Binary features			
Numeric features			
Categorical features			
Robust to noisy training examples			
Fast classification is crucial			
Many irrelevant features			
Relevant features have very different scale			

SUMMARY

- Perceptron concepts
  - training/prediction algorithms (standard, voting, averaged)
  - convergence theorem and what practical guarantees it gives us
  - how to draw/describe the decision boundary of a perceptron classifier
- Fundamental ML concepts
  - Determine whether a data set is linearly separable and define its margin
  - Error driven algorithms, online vs. batch algorithms

ANNOUNCEMENTS

- HW 2 is out
  - due February 20, 11:59pm

ACKNOWLEDGEMENTS

- These slides use materials by Marine Carpuat, Piyush Rai and Hal Daume III