Machine Learning (CSE 446): Perceptron

Sham M Kakade

© 2018

University of Washington
skakade@cs.washington.edu
Happy Medium?

Decision trees (that aren’t too deep): use relatively few features to classify.

$K$-nearest neighbors: all features weighted equally.

Today: use all features, but weight them.
Happy Medium?

Decision trees (that aren’t too deep): use relatively few features to classify.

\( K \)-nearest neighbors: all features weighted equally.

Today: use all features, but weight them.

For today’s lecture, assume that \( y \in \{-1, +1\} \) instead of \( \{0, 1\} \), and that \( x \in \mathbb{R}^d \).
Happy Medium?

Decision trees (that aren’t too deep): use relatively few features to classify.

$K$-nearest neighbors: all features weighted equally.

Today: use all features, but weight them.

For today’s lecture, assume that $y \in \{-1, +1\}$ instead of $\{0, 1\}$, and that $x \in \mathbb{R}^d$.

Remember that $x[j] = \phi_j(x)$. (Features have already been applied to the data.)
Inspiration from Neurons

Image from Wikimedia Commons.

Input signals come in through dendrites, output signal passes out through the axon.
Neuron-Inspired Classifier

\[ \sum \left( w_1 x_1 + w_2 x_2 + \ldots + w_d x_d + b \right) \]

- Input: \( x_1, x_2, \ldots, x_d \)
- Weight parameters: \( w_1, w_2, \ldots, w_d \)
- Bias parameter: \( b \)
- “Activation” function
- Output: \( \hat{y} \)
Neuron-Inspired Classifier

\[ f(x) = \text{sign}\left( \mathbf{w} \cdot \mathbf{x} + b \right) \]

remembering that: \( \mathbf{w} \cdot \mathbf{x} = \sum_{j=1}^{d} w[j] \cdot x[j] \)
Neuron-Inspired Classifier

\[ f(x) = \text{sign}(w \cdot x + b) \]

remembering that: \( w \cdot x = \sum_{j=1}^{d} w[j] \cdot x[j] \)

Learning requires us to set the weights \( w \) and the bias \( b \).
Perceptron Learning Algorithm

Data: $D = \langle (x_n, y_n) \rangle_{n=1}^N$, number of epochs $E$

Result: weights $w$ and bias $b$

initialize: $w = 0$ and $b = 0$;

for $e \in \{1, \ldots, E\}$ do
  for $n \in \{1, \ldots, N\}$, in random order do
    # predict
    $\hat{y} = \text{sign}(w \cdot x_n + b)$;
    if $\hat{y} \neq y_n$ then
      # update
      $w \leftarrow w + y_n \cdot x_n$;
      $b \leftarrow b + y_n$;
    end
  end
end

return $w, b$

Algorithm 1: PerceptronTrain
Linear Decision Boundary

\[ w \cdot x + b < 0 \]

\[ w \cdot x + b > 0 \]

\[ w \cdot x + b = 0 \]
Linear Decision Boundary

\[ \mathbf{w} \cdot \mathbf{x} + b = 0 \]

activation = \( \mathbf{w} \cdot \mathbf{x} + b \)
Linear Decision Boundary

\[ \mathbf{w} \cdot \mathbf{x} + b = 0 \]
Interpretation of Weight Values

What does it mean when . . .
Interpretation of Weight Values

What does it mean when . . .

▶ \( w_{12} = 100 \)?
Interpretation of Weight Values

What does it mean when . . .

- $w_{12} = 100$?
- $w_{12} = -1$?
Interpretation of Weight Values

What does it mean when . . .

- $w_{12} = 100$?
- $w_{12} = -1$?
- $w_{12} = 0$?
Interpretation of Weight Values

What does it mean when . . .

- $w_{12} = 100$?
- $w_{12} = -1$?
- $w_{12} = 0$?

What if we multiply $w$ by 2?
Linear Separability

A dataset $D = \langle (x_n, y_n) \rangle_{n=1}^{N}$ is **linearly separable** if there exists some linear classifier (defined by $w, b$) such that, for all $n$, $y_n = \text{sign}(w \cdot x_n + b)$.

$$\text{margin}(D, w, b) = \begin{cases} \min_n y_n \cdot (w \cdot x_n + b) & \text{if } w \text{ and } b \text{ separate } D \\ -\infty & \text{otherwise} \end{cases}$$

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

Due to ?.

If $D$ is linearly separable with margin $\gamma > 0$ and for all $n \in \{1, \ldots, N\}$, $\|x_n\|_2 \leq 1$, then the perceptron algorithm will converge in at most $\frac{1}{\gamma^2}$ updates.
If $D$ is linearly separable with margin $\gamma > 0$ and for all $n \in \{1, \ldots, N\}$, $\|x_n\|_2 \leq 1$, then the perceptron algorithm will converge in at most $\frac{1}{\gamma^2}$ updates.

- Proof can be found in ?, pp. 50–51.
- The theorem does not guarantee that the perceptron’s classifier will achieve margin $\gamma$. 
What if $D$ isn't linearly separable?
What if $D$ isn't linearly separable?

No guarantees.
Voted Perceptron

Let $\mathbf{w}^{(e,n)}$ and $b^{(e,n)}$ be the parameters after updating based on the $n$th example on epoch $e$.

$$\hat{y} = \text{sign} \left( \sum_{e=1}^{E} \sum_{n=1}^{N} \text{sign}(\mathbf{w}^{(e,n)} \cdot \mathbf{x} + b^{(e,n)}) \right)$$
Voted Perceptron

Let $w^{(e,n)}$ and $b^{(e,n)}$ be the parameters after updating based on the $n$th example on epoch $e$.

$$
\hat{y} = \text{sign} \left( \sum_{e=1}^{E} \sum_{n=1}^{N} \text{sign}(w^{(e,n)} \cdot x + b^{(e,n)}) \right)
$$

Do we need to store $E \cdot N$ parameter settings?
Let $w^{(e,n)}$ and $b^{(e,n)}$ be the parameters after updating based on the $n$th example on epoch $e$.

$$\hat{y} = \text{sign} \left( \sum_{e=1}^{E} \sum_{n=1}^{N} \text{sign} \left( w^{(e,n)} \cdot x + b^{(e,n)} \right) \right)$$

Can do a little better by storing “survival” time for each vector (number of times an update didn’t need to be made because $\hat{y} = y$).
Voted Perceptron
Voted Perceptron
Voted Perceptron
Voted Perceptron
Voted Perceptron
Voted Perceptron
Averaged Perceptron

\[ \hat{y} = \text{sign} \left( \sum_{e=1}^{E} \sum_{n=1}^{N} \mathbf{w}^{(e,n)} \cdot \mathbf{x} + b^{(e,n)} \right) \]
Averaged Perceptron

\[ \hat{y} = \text{sign}\left( \sum_{e=1}^{E} \sum_{n=1}^{N} w^{(e,n)} \cdot x + b^{(e,n)} \right) \]

Do we need to store $E \cdot N$ parameter settings?
Averaged Perceptron

\[ \hat{y} = \text{sign} \left( \sum_{e=1}^{E} \sum_{n=1}^{N} w^{(e,n)} \cdot x + b^{(e,n)} \right) \]

Do we need to store \( E \cdot N \) parameter settings?

What kind of decision boundary does the averaged perceptron have?
Averaged Perceptron

\[ \hat{y} = \text{sign} \left( \sum_{e=1}^{E} \sum_{n=1}^{N} w^{(e,n)} \cdot x + b^{(e,n)} \right) \]

= \text{sign} \left( \left( \sum_{e=1}^{E} \sum_{n=1}^{N} w^{(e,n)} \right) \cdot x + \left( \sum_{e=1}^{E} \sum_{n=1}^{N} b^{(e,n)} \right) \right)

Do we need to store \( E \cdot N \) parameter settings?

What kind of decision boundary does the averaged perceptron have?