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 \( \mathbf{w} \in \mathbb{R}^D \)
  - \( \mathbf{w} \) 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

\[ \hat{y} = \text{sign}(w^T x + b) \]
Function Approximation with Perceptron

Problem setting
• Set of possible instances $X$
  – Each instance $x \in X$ is a feature vector $x = [x_1, ..., x_D]$
• 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 $\{(x^{(1)}, y^{(1)}), ..., (x^{(N)}, y^{(N)})\}$ of unknown target function $f$

Output
• Hypothesis $h \in H$ that best approximates target function $f$
Perception: Prediction Algorithm

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

1. \( a \leftarrow \sum_{d=1}^{D} w_d \hat{x}_d + b \)  
   \hspace{1em} // compute activation for the test example
2. \textbf{return} \texttt{SIGN}(a)
Aside: biological inspiration

Analogy: the perceptron as a neuron
Perceptron Training Algorithm

**Algorithm 5** \texttt{PerceptronTrain}(D, MaxIter)

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

$w_{\text{old}}$

misclassified

$w_{\text{old}}$

$x$

$w_{\text{new}}$
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