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

\[ \hat{y} = \text{sign}(w^T x + b) \]
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 $H = \{h | h: X \rightarrow Y\}$
  – 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

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

1: \hspace{1em} a \leftarrow \sum_{d=1}^{D} w_d \hat{x}_d + b \hspace{2em} // \textit{compute activation for the test example}
2: \hspace{1em} \textbf{return} \hspace{0.5em} \textsc{SIGN}(a)
Aside: biological inspiration

Analogy: the perceptron as a neuron
Perceptron Training Algorithm

Algorithm 5 PerceptronTrain(D, MaxIter)

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

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

• The voted perceptron

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

• The averaged perceptron

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

• Require keeping track of “survival time” of weight vectors \( c^{(1)}, \ldots, c^{(K)} \)
Averaged perceptron decision rule

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

can be rewritten as

\[ \hat{y} = \text{sign} \left( \left( \sum_{k=1}^{K} c^{(k)} \mathbf{w}^{(k)} \right) \cdot \hat{x} + \sum_{k=1}^{K} c^{(k)} b^{(k)} \right) \]
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!