The Perceptron

CMSC 422
SOHEIL FEIZI
sfeizi@cs.umd.edu

Slides adapted from MARINE CARPUAT
This week

• A new model/algorithm
  – the perceptron
  – and its variants: voted, averaged

• Fundamental Machine Learning Concepts
  – Online vs. batch learning
  – Error-driven learning
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 $x = [x_1, \ldots, x_D]$
• Unknown target function $f: X \to Y$
  – $Y$ is binary valued $\{-1; +1\}$
• Set of function hypotheses $H = \{h \mid h: X \to Y\}$
  – 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** \texttt{PerceptronTest}(w_0, w_1, \ldots, w_D, b, \hat{x})

1. \( a \leftarrow \sum_{d=1}^{D} w_d \hat{x}_d + b \)  
   // compute activation for the test example
2. \texttt{return} \ \texttt{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 \( d = 1 \ldots D \)  // initialize weights
2: \( b \leftarrow 0 \)  // initialize bias
3: for iter = 1 \ldots MaxIter do
4:     for all \( (x,y) \in D \) do
5:         \( a \leftarrow \sum_{d=1}^{D} w_d x_d + b \)  // 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
7:     end if
6:     end for
5: end for
4: 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

$W_{old}$

misclassified

$x$

$W_{new}$
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** \textsc{PerceptronTrain}(D, MaxIter)

1: \( w_d \leftarrow 0 \), for all \( d = 1 \ldots D \) \hspace{1cm} // initialize weights
2: \( b \leftarrow 0 \) \hspace{1cm} // initialize bias
3: \textbf{for} iter = 1 \ldots MaxIter \textbf{do} \hspace{1cm} // training loop
4: \hspace{1cm} \textbf{for} all \((x,y) \in D\) \textbf{do} \hspace{1cm} // iterate through examples
5: \hspace{2cm} a \leftarrow \sum_{d=1}^{D} w_d x_d + b \hspace{1cm} // compute activation for this example
6: \hspace{2cm} \textbf{if} ya \leq 0 \textbf{ then} \hspace{1cm} // classify incorrectly
7: \hspace{3cm} w_d \leftarrow w_d + yx_d, \text{ for all } d = 1 \ldots D \hspace{1cm} // update weights
8: \hspace{3cm} b \leftarrow b + y \hspace{1cm} // update bias
9: \hspace{2cm} \textbf{end if} \hspace{1cm} // classify correctly
10: \hspace{1cm} \textbf{end for} \hspace{1cm} // iterate through examples
11: \textbf{end for} \hspace{1cm} // train model
12: \textbf{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?
Convergence of Perceptron

• 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 \( D = \{(x_1, y_1), \ldots, (x_N, y_N)\} \) is linearly separable with margin \( \gamma \) by a unit norm hyperplane \( w_\ast (\|w_\ast\| = 1) \) with \( b = 0 \),

Then perceptron training converges after \( \frac{R^2}{\gamma^2} \)

errors during training (assuming \( \|x\| < R \) for all \( x \)).
Margin of a data set $D$

$$margin(D, w, b) = \begin{cases} \min_{(x,y) \in D} y(w \cdot x + b) & \text{if } w \text{ separates } D \\ -\infty & \text{otherwise} \end{cases}$$  \hspace{1cm} (4.8)$$

Distance between the hyperplane $(w,b)$ and the nearest point in $D$

$$margin(D) = \sup_{w,b} margin(D, w, b)$$  \hspace{1cm} (4.9)$$

Largest attainable margin on $D$
Theorem (Block & Novikoff, 1962)
If the training data $D = \{(x_1, y_1), \ldots, (x_N, y_N)\}$ is linearly separable with margin $\gamma$ by a unit norm hyperplane $w_*$ ($||w_*||=1$) with $b = 0$, then perceptron training converges after $\frac{R^2}{\gamma^2}$ errors during training (assuming $(||x||< R)$ for all $x$).

Proof:
- Margin of $w_*$ on any arbitrary example $(x_n, y_n)$: $\frac{y_n w^T_* x_n}{||w_*||} = y_n w^T_* x_n \geq \gamma$
- Consider the $(k+1)^{th}$ mistake: $y_n w^T_k x_n \leq 0$, and update $w_{k+1} = w_k + y_n x_n$
- $w^T_{k+1} w_* = w^T_k w_* + y_n w^T_* x_n \geq w^T_k w_* + \gamma$ (why is this nice?)
- Repeating iteratively $k$ times, we get $w^T_{k+1} w_* \geq k \gamma$ \hspace{1cm} (1)
- $||w_{k+1}||^2 = ||w_k||^2 + 2 y_n w^T_k x_n + ||x||^2 \leq ||w_k||^2 + R^2$ (since $y_n w^T_k x_n \leq 0$)
- Repeating iteratively $k$ times, we get $||w_{k+1}||^2 \leq k R^2$ \hspace{1cm} (2)
What does this 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
Practical Implications

- Sensitivity to noise
  - If the data is not linearly separable due to noise, no guarantee of convergence or accuracy

- Linear separability in practice
  - Data may be linearly separable in practice
  - Especially when number of features $>>$ number of examples

- Risk of overfitting mitigated by
  - Early stopping
  - Averaging
What you should know

• 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
This week

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