The Perceptron

CMSC 422
MARINE CARPUAT
marine@cs.umd.edu

Credit: figures by Piyush Rai and Hal Daume III
This week

• The perception: a new model/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
• Project 1 published today
Recap: Perceptron for binary classification

- Classifier = hyperplane that separates positive from negative examples

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

- Perceptron training
  - Finds such a hyperplane
  - Online & error-driven
Recap: Perceptron updates

Update for a misclassified positive example:

\[ \mathbf{w}_{\text{new}} = \mathbf{w}_{\text{old}} + \mathbf{x} \]
Recap: Perceptron updates

Update for a misclassified negative example:

\[ \mathbf{w}_{\text{new}} = \mathbf{w}_{\text{old}} - \mathbf{x} \]
Standard Perceptron: predict based on final parameters

Algorithm 5 \textsc{PerceptronTrain}(\textbf{D}, \textit{MaxIter})

\begin{enumerate}
\item $w_d \leftarrow 0$, for all $d = 1 \ldots D$  \hfill \text{\# initialize weights}
\item $b \leftarrow 0$ \hfill \text{\# initialize bias}
\item \textbf{for} iter = 1 \ldots \textit{MaxIter} \textbf{do}
\item \quad \textbf{for} all $(x,y) \in \textbf{D}$ \textbf{do}
\item \quad \quad $a \leftarrow \sum_{d=1}^{D} w_d x_d + b$ \hfill \text{\# compute activation for this example}
\item \quad \quad \textbf{if} $ya \leq 0$ \textbf{then}
\item \quad \quad \quad $w_d \leftarrow w_d + yx_d$, for all $d = 1 \ldots D$ \hfill \text{\# update weights}
\item \quad \quad \quad $b \leftarrow b + y$ \hfill \text{\# update bias}
\item \quad \quad \textbf{end if}
\item \quad \textbf{end for}
\item \textbf{end for}
\item \textbf{return} $w_0, w_1, \ldots, w_D, b$
\end{enumerate}
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 \( c^{(1)}, \ldots, c^{(K)} \)
How would you modify this algorithm for voted perceptron?

Algorithm 5 \textsc{PerceptronTrain}(D, \text{MaxIter})

\begin{algorithmic}[1]
\STATE $w_d \leftarrow 0$, for all $d = 1 \ldots D$ \quad // initialize weights
\STATE $b \leftarrow 0$ \quad // initialize bias
\FOR {\text{iter} = 1 \ldots \text{MaxIter}} 
\FOR {all $(x,y) \in D$} 
\STATE $a \leftarrow \sum_{d=1}^{D} w_d x_d + b$ \quad // compute activation for this example
\IF {$ya \leq o$} 
\STATE $w_d \leftarrow w_d + yx_d$, for all $d = 1 \ldots D$ \quad // update weights
\STATE $b \leftarrow b + y$ \quad // update bias
\ENDIF 
\ENDFOR 
\ENDFOR 
\RETURN $w_0, w_1, \ldots, w_D, b$
\end{algorithmic}
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 \( (x, y) \in D \) do
5:     \( a \leftarrow \sum_{d=1}^{D} w_d x_d + b \)  // compute activation for this example
6:     if \( ya \leq o \) 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 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

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

Algorithm 7 \text{AveragedPerceptronTrain}(D, \text{MaxIter})

1: \( w \leftarrow \langle 0,0,\ldots,0 \rangle \), \( b \leftarrow 0 \) // initialize weights and bias
2: \( u \leftarrow \langle 0,0,\ldots,0 \rangle \), \( \beta \leftarrow 0 \) // initialize cached weights and bias
3: \( c \leftarrow 1 \) // initialize example counter to one

4: for \( \text{iter} = 1 \ldots \text{MaxIter} \) do
5: \hspace{1em} for all \( (x,y) \in D \) do
6: \hspace{2em} if \( y(w \cdot x + b) \leq 0 \) then
7: \hspace{3em} \( w \leftarrow w + yx \) // update weights
8: \hspace{3em} \( b \leftarrow b + y \) // update bias
9: \hspace{2em} \( u \leftarrow u + yc x \) // update cached weights
10: \hspace{2em} \( \beta \leftarrow \beta + yc \) // update cached bias
11: \hspace{1em} end if

12: \( c \leftarrow c + 1 \) // increment counter regardless of update
13: end for
14: end for
15: return \( w - \frac{1}{c} u, b - \frac{1}{c} \beta \) // return averaged weights and bias
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 $w_∗ (\|w_∗\| = 1)$ 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 $D$

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

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

$$\text{margin}(D) = \sup_{w,b} \text{margin}(D, w, b) \quad (4.9)$$

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: $y_n w_k^T x_n \leq 0$, and update $w_{k+1} = w_k + y_n x_n$
- $w_{k+1}^T w_* = w_k^T w_* + y_n w_*^T x_n \geq w_k^T w_* + \gamma$
- Repeating iteratively $k$ times, we get $w_{k+1}^T w_* > k\gamma$ (1)
- $||w_{k+1}||^2 = ||w_k||^2 + 2y_n w_k^T x_n + ||x||^2 \leq ||w_k||^2 + R^2$ (since $y_n w_k^T x_n \leq 0$)
- Repeating iteratively $k$ times, we get $||w_{k+1}||^2 \leq kR^2$ (2)
Theorem (Block & Novikoff, 1962)
If the training data $D = \{(x_1, y_1), \ldots, (x_N, y_N)\}$ is \textbf{linearly separable} with margin $\gamma$ by a unit norm hyperplane $w_*(||w_*|| = 1)$ with $b = 0$, then \textbf{perceptron training converges} after $\frac{R^2}{\gamma^2}$ errors during training (assuming $(||x|| < R)$ for all $x$).

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
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