10-301/601: Introduction to Machine Learning
Lecture 6 – Perceptron

Henry Chai & Matt Gormley
9/19/22
Suppose we do model selection using a validation dataset. For our final model, should we train using both training and validation datasets?

- Yes, absolutely! So really the sketch from last lecture should look something like:

1. Split $\mathcal{D}$ into $\mathcal{D}_{train} \cup \mathcal{D}_{val} \cup \mathcal{D}_{test}$
2. Learn classifiers using $\mathcal{D}_{train}$
3. Evaluate models using $\mathcal{D}_{val}$ and choose the one with lowest validation error:
   - 4. **Learn a new classifier from the best model using** $\mathcal{D}_{train} \cup \mathcal{D}_{val}$
5. Optionally, use $\mathcal{D}_{test}$ to estimate the true error
Q & A:

Can we use KNNs with categorical features?

- Again, yes! We can either convert categorical features into binary ones or use a distance metric that works over categorical features e.g., the Hamming distance:

\[
d(x, x') = \sum_{d=1}^{D} \mathbb{1}(x_d = x'_d)
\]
• Announcements:
  • HW2 released 9/7, due 9/19 (today!) at 11:59 PM
  • HW3 released 9/21, due 9/28 at 11:59 PM
    • Only two grace days allowed on HW3
Recall: Fisher Iris Dataset
• Notation: in this class vectors will be assumed to be column vectors by default, i.e.,

\[
\mathbf{a} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_D \end{bmatrix} \quad \text{and} \quad \mathbf{a}^T = [a_1 \ a_2 \ \cdots \ a_D]
\]

• The dot product between two \( D \)-dimensional vectors is

\[
\mathbf{a}^T \mathbf{b} = [a_1 \ a_2 \ \cdots \ a_D] \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_D \end{bmatrix} = \sum_{d=1}^{D} a_d b_d
\]

• The \( L2 \)-norm of \( \mathbf{a} = \|\mathbf{a}\|_2 = \sqrt{\mathbf{a}^T \mathbf{a}} \)

• Two vectors are orthogonal iff \( \mathbf{a}^T \mathbf{b} = 0 \)
1. On the axes below, draw the region corresponding to
   \[ w_1 x_1 + w_2 x_2 + b > 0 \]
   where \( w_1 = 1 \), \( w_2 = 2 \) and \( b = -4 \).

2. Then draw the vector \( w = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \)
1. On the axes below, draw the region corresponding to 
\( w_1 x_1 + w_2 x_2 + b > 0 \)
where \( w_1 = 1, w_2 = 2 \) and \( b = -4 \).

2. Then draw the vector \( w = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \)
Linear Decision Boundaries

- In 2 dimensions, \( w_1 x_1 + w_2 x_2 + b = 0 \) defines a line
- In 3 dimensions, \( w_1 x_1 + w_2 x_2 + w_3 x_3 + b = 0 \) defines a plane
- In 4+ dimensions, \( w^T x + b = 0 \) defines a hyperplane
  - The vector \( w \) is always orthogonal to this hyperplane and always points in the direction where \( w^T x + b > 0 \)!
- A hyperplane creates two halfspaces:
  - \( S_+ = \{ x : w^T x + b > 0 \} \) or all \( x \) s.t. \( w^T x + b \) is positive
  - \( S_- = \{ x : w^T x + b < 0 \} \) or all \( x \) s.t. \( w^T x + b \) is negative
Goal: learn classifiers of the form $h(x) = \text{sign}(w^T x + b)$ (assuming $y \in \{-1, +1\}$)

Key question: how do we learn the parameters, $w$ and $b$?
So far, we’ve been learning in the batch setting, where we have access to the entire training dataset at once.

A common alternative is the online setting, where examples arrive gradually and we learn continuously.

Examples of online learning:

- online shopping: customer purchases as new examples
- predicting success in classes
- recommender systems
- siri (any voice agent)
Online Learning: Setup

• For $t = 1, 2, 3, ...$
  
  • Receive an unlabeled example, $x^{(t)}$
  
  • Predict its label, $\hat{y} = h_{w,b}(x^{(t)})$
  
  • Observe its true label, $y^{(t)}$
  
  • Pay a penalty if we made a mistake, $\hat{y} \neq y^{(t)}$
  
  • Update the parameters, $w$ and $b$

• Goal: minimize the number of mistakes made
(Online) Perceptron Learning Algorithm

- Initialize the weight vector and intercept to all zeros:
  \[ \mathbf{w} = [0 \ 0 \ \cdots \ 0] \text{ and } b = 0 \]

- For \( t = 1, 2, 3, \ldots \)
  - Receive an unlabeled example, \( \mathbf{x}^{(t)} \)
  - Predict its label, \( \hat{y} = \text{sign}(\mathbf{w}^T \mathbf{x} + b) = \begin{cases} +1 & \text{if } \mathbf{w}^T \mathbf{x} + b \geq 0 \\ -1 & \text{otherwise} \end{cases} \)
  - Observe its true label, \( y^{(t)} \)
  - If we misclassified a positive example (\( y^{(t)} = +1, \hat{y} = -1 \)):
    - \( \mathbf{w} \leftarrow \mathbf{w} + \mathbf{x}^{(t)} \)
    - \( b \leftarrow b + 1 \)
  - If we misclassified a negative example (\( y^{(t)} = -1, \hat{y} = +1 \)):
    - \( \mathbf{w} \leftarrow \mathbf{w} - \mathbf{x}^{(t)} \)
    - \( b \leftarrow b - 1 \)
(Online) Perceptron Learning Algorithm

- Initialize the weight vector and intercept to all zeros:
  \[
  \mathbf{w} = [0 \ 0 \ \cdots \ 0] \quad \text{and} \quad b = 0
  \]

- For \( t = 1, 2, 3, \ldots \)
  - Receive an unlabeled example, \( \mathbf{x}^{(t)} \)
  - Predict its label, \( \hat{y} = \text{sign}(\mathbf{w}^T \mathbf{x} + b) = \begin{cases} +1 & \text{if } \mathbf{w}^T \mathbf{x} + b \geq 0 \\ -1 & \text{otherwise} \end{cases} \)
  - Observe its true label, \( y^{(t)} \)
  - If we misclassified an example \( (y^{(t)} \neq \hat{y}) \):
    \[
    \begin{cases}
    \mathbf{w} \leftarrow \mathbf{w} + y^{(t)} \mathbf{x}^{(t)} \\
    b \leftarrow b + y^{(t)}
    \end{cases}
    \]
(Online) Perceptron Learning Algorithm: Example (no Intercept)

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$\hat{y}$</th>
<th>$y$</th>
<th>Mistake?</th>
</tr>
</thead>
<tbody>
<tr>
<td>$-1$</td>
<td>$2$</td>
<td>$+$</td>
<td>$-$</td>
<td>Yes</td>
</tr>
</tbody>
</table>

$w = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$
(Online) Perceptron Learning Algorithm: Example (no Intercept)

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$\hat{y}$</th>
<th>$y$</th>
<th>Mistake?</th>
</tr>
</thead>
<tbody>
<tr>
<td>$-1$</td>
<td>$2$</td>
<td>$+$</td>
<td>$-$</td>
<td>Yes</td>
</tr>
</tbody>
</table>

$$w = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$w \leftarrow w + y^{(1)} x^{(1)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} - \begin{bmatrix} -1 \\ 2 \end{bmatrix} = \begin{bmatrix} 1 \\ -2 \end{bmatrix}$$

Example courtesy of Nina Balcan
(Online) Perceptron Learning Algorithm: Example (no Intercept)

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$\hat{y}$</th>
<th>$y$</th>
<th>Mistake?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>+</td>
<td>+</td>
<td>No</td>
</tr>
<tr>
<td>−1</td>
<td>2</td>
<td>+</td>
<td>−</td>
<td>Yes</td>
</tr>
</tbody>
</table>

$w = \begin{bmatrix} 1 \\ -2 \end{bmatrix}$
Perceptron Learning Algorithm: Example (no Intercept)

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$\hat{y}$</th>
<th>$y$</th>
<th>Mistake?</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>2</td>
<td>+</td>
<td>-</td>
<td>Yes</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>+</td>
<td>+</td>
<td>No</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>-</td>
<td>+</td>
<td>Yes</td>
</tr>
</tbody>
</table>

$w = \begin{bmatrix} 1 \\ -2 \end{bmatrix}$

$w \leftarrow w + y^{(3)}x^{(3)} = \begin{bmatrix} 1 \\ -2 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ -1 \end{bmatrix}$

(Online) Perceptron Learning Algorithm: Example (no Intercept)

Example courtesy of Nina Balcan
(Online) Perceptron Learning Algorithm: Example (no Intercept)

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$\hat{y}$</th>
<th>$y$</th>
<th>Mistake?</th>
</tr>
</thead>
<tbody>
<tr>
<td>$-1$</td>
<td>$2$</td>
<td>$+$</td>
<td>$-$</td>
<td>Yes</td>
</tr>
<tr>
<td>$1$</td>
<td>$0$</td>
<td>$+$</td>
<td>$+$</td>
<td>No</td>
</tr>
<tr>
<td>$1$</td>
<td>$1$</td>
<td>$-$</td>
<td>$+$</td>
<td>Yes</td>
</tr>
</tbody>
</table>

$w = \begin{bmatrix} 1 \\ -2 \end{bmatrix}$

$w \leftarrow w + y^{(3)} x^{(3)} = \begin{bmatrix} 1 \\ -2 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ -1 \end{bmatrix}$

Example courtesy of Nina Balcan
(Online) Perceptron Learning Algorithm: Example (no Intercept)

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$\hat{y}$</th>
<th>$y$</th>
<th>Mistake?</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>2</td>
<td>+</td>
<td>-</td>
<td>Yes</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>+</td>
<td>+</td>
<td>No</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>-</td>
<td>+</td>
<td>Yes</td>
</tr>
<tr>
<td>-1</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>No</td>
</tr>
</tbody>
</table>

$w = \begin{bmatrix} 2 \\ -1 \end{bmatrix}$
(Online) Perceptron Learning Algorithm: Example (no Intercept)

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$\hat{y}$</th>
<th>$y$</th>
<th>Mistake?</th>
</tr>
</thead>
<tbody>
<tr>
<td>−1</td>
<td>2</td>
<td>+</td>
<td>−</td>
<td>Yes</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>+</td>
<td>+</td>
<td>No</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>−</td>
<td>+</td>
<td>Yes</td>
</tr>
<tr>
<td>−1</td>
<td>0</td>
<td>−</td>
<td>−</td>
<td>No</td>
</tr>
<tr>
<td>−1</td>
<td>−2</td>
<td>+</td>
<td>−</td>
<td>Yes</td>
</tr>
</tbody>
</table>

\[
\mathbf{w} = \begin{bmatrix} 2 \\ -1 \end{bmatrix}
\]

\[
\mathbf{w} \leftarrow \mathbf{w} + y^{(5)} \mathbf{x}^{(5)} = \begin{bmatrix} 2 \\ -1 \end{bmatrix} - \begin{bmatrix} -1 \\ -2 \end{bmatrix} = \begin{bmatrix} 3 \\ 1 \end{bmatrix}
\]

Decision Boundary
(Online) Perceptron Learning Algorithm: Example (no Intercept)

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$\hat{y}$</th>
<th>$y$</th>
<th>Mistake?</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>2</td>
<td>+</td>
<td>-</td>
<td>Yes</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>+</td>
<td>+</td>
<td>No</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>-</td>
<td>+</td>
<td>Yes</td>
</tr>
<tr>
<td>-1</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>No</td>
</tr>
<tr>
<td>-1</td>
<td>-2</td>
<td>+</td>
<td>-</td>
<td>Yes</td>
</tr>
</tbody>
</table>

$w = \begin{bmatrix} 2 \\ -1 \end{bmatrix}$

$w \leftarrow w + y^{(5)} x^{(5)} = \begin{bmatrix} 2 \\ -1 \end{bmatrix} - \begin{bmatrix} -1 \\ -2 \end{bmatrix} = \begin{bmatrix} 3 \\ 1 \end{bmatrix}$
Perceptron Learning Algorithm: Example (no Intercept)

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$\hat{y}$</th>
<th>$y$</th>
<th>Mistake?</th>
</tr>
</thead>
<tbody>
<tr>
<td>$-1$</td>
<td>2</td>
<td>+</td>
<td>$-$</td>
<td>Yes</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>+</td>
<td>+</td>
<td>No</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>$-$</td>
<td>+</td>
<td>Yes</td>
</tr>
<tr>
<td>$-1$</td>
<td>0</td>
<td>$-$</td>
<td>$-$</td>
<td>No</td>
</tr>
<tr>
<td>$-1$</td>
<td>$-2$</td>
<td>+</td>
<td>$-$</td>
<td>Yes</td>
</tr>
<tr>
<td>1</td>
<td>$-1$</td>
<td>+</td>
<td>+</td>
<td>No</td>
</tr>
</tbody>
</table>

$w = \begin{bmatrix} 3 \\ 1 \end{bmatrix}$

Example courtesy of Nina Balcan
Poll Question 1:

• **True or False:** Unlike Decision Trees and $k$-Nearest Neighbors, the Perceptron algorithm does not suffer from overfitting because it does not have any hyperparameters that could be over-tuned on the training data.
  
  A. True
  
  B. True and False *(TOXIC)*
  
  C. False
• If we add a 1 to the beginning of every example e.g.,

\[ x' = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ \vdots \\ x_D \end{bmatrix} \ldots \]

• ... we can just fold the intercept into the weight vector!

\[ \theta = \begin{bmatrix} b \\ w_1 \\ w_2 \\ \vdots \\ w_D \end{bmatrix} \rightarrow \theta^T x' = w^T x + b \]
• Initialize the weight vector and intercept to all zeros:

\[
\mathbf{w} = [0 \ 0 \ \cdots \ 0] \text{ and } b = 0
\]

• For \( t = 1, 2, 3, \ldots \)
  • Receive an unlabeled example, \( \mathbf{x}^{(t)} \)
  • Predict its label, \( \hat{y} = \text{sign}(\mathbf{w}^T \mathbf{x} + b) = \begin{cases} 
+1 \text{ if } \mathbf{w}^T \mathbf{x} + b \geq 0 \\
-1 \text{ otherwise}
\end{cases} \)
  • Observe its true label, \( y^{(t)} \)
  • If we misclassified an example \( (y^{(t)} \neq \hat{y}) \):
    \[
    \begin{cases} 
    \mathbf{w} \leftarrow \mathbf{w} + y^{(t)} \mathbf{x}^{(t)} \\
    b \leftarrow b + y^{(t)}
    \end{cases}
    \]
- Initialize the parameters to all zeros:
  \[ \theta = [0 \ 0 \ \cdots \ 0] \]

- For \( t = 1, 2, 3, \ldots \)
  - Receive an unlabeled example, \( x^{(t)} \)
  - Predict its label, \( \hat{y} = \text{sign} (\theta^T x'^{(t)}) = \begin{cases} +1 & \text{if } \theta^T x'^{(t)} \geq 0 \\ -1 & \text{otherwise} \end{cases} \)
  - Observe its true label, \( y^{(t)} \)
  - If we misclassified an example (\( y^{(t)} \neq \hat{y} \)):
    - \( \theta \leftarrow \theta + y^{(t)} x'^{(t)} \)

(Online) Perceptron Learning Algorithm

Automatically handles updating the intercept
The intercept shifts the decision boundary off the origin

- Increasing $b$ shifts the decision boundary towards the negative side (in the opposite direction of $w$)
- Decreasing $b$ shifts the decision boundary towards the positive side (in the direction of $w$)
(Online)
Perceptron Learning
Algorithm: Inductive Bias

1. More recent mistakes are more important than older ones (and should be corrected immediately)
2. The decision boundary should be linear
3. The features should behave similarly
• Initialize the parameters to all zeros:

\[ \boldsymbol{\theta} = [0 \ 0 \ \cdots \ 0] \]

• For \( t = 1, 2, 3, \ldots \)
  
  • Receive an unlabeled example, \( x^{(t)} \)
  
  • Predict its label, \( \hat{y} = \text{sign} \left( \theta^T x^{(t)} \right) \)
  
  • Observe its true label, \( y^{(t)} \)
  
  • If we misclassified an example (\( y^{(t)} \neq \hat{y} \)):
    
    \[ \boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + y^{(t)} x^{(t)} \]
(Batch) Perceptron Learning Algorithm

- Input: \( \mathcal{D} = \{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(N)}, y^{(N)})\} \)
- Initialize the parameters to all zeros:
  \[
  \theta = [0 \ 0 \ \ldots \ 0]
  \]
- While NOT CONVERGED
  - For \( t \in \{1, \ldots, N\} \)
  - Predict the label of \( x'^{(t)}, \hat{y} = \text{sign}(\theta^T x'^{(t)}) \)
  - Observe its true label, \( y^{(t)} \)
  - If we misclassified \( x'^{(t)} \) (\( y^{(t)} \neq \hat{y} \)):
    - \( \theta \leftarrow \theta + y^{(t)} x'^{(t)} \)
Poll Question 2:

- **True or False**: The parameter vector $\mathbf{w}$ learned by the batch Perceptron Learning Algorithm can be written as a **linear combination** of the examples, i.e.,

$$\mathbf{w} = c_1 \mathbf{x}^{(1)} + c_2 \mathbf{x}^{(2)} + \cdots + c_N \mathbf{x}^{(N)}$$

A. True if you replace “linear” with “polynomial”
B. True and False (TOXIC)
C. True
D. False
Definitions:

- A dataset $\mathcal{D}$ is *linearly separable* if $\exists$ a linear decision boundary that perfectly classifies the examples in $\mathcal{D}$.
- The margin, $\gamma$, of a dataset $\mathcal{D}$ is the greatest possible distance between a linear separator and the closest example in $\mathcal{D}$ to that linear separator.
• Theorem: if the examples seen by the Perceptron Learning Algorithm (online and batch)
  1. lie in a ball of radius $R$ (centered around the origin)
  2. have a margin of $\gamma$

then the algorithm makes at most $(R/\gamma)^2$ mistakes.

• Key Takeaway: if the training dataset is linearly separable, the batch Perceptron Learning Algorithm will converge (i.e., stop making mistakes on the training dataset or achieve 0 training error) in a finite number of steps!
Computing the Margin

Let \( x' \) be an arbitrary point on the hyperplane \( w^T x + b = 0 \) and let \( x'' \) be an arbitrary point on the hyperplane.

The distance between \( x'' \) and \( w^T x + b = 0 \) is equal to the magnitude of the projection of \( x'' - x' \) onto \( \frac{w}{\|w\|_2} \), the unit vector orthogonal to the hyperplane.

\[ w^T x + b = 0 \]
Computing the Margin

- Let $x'$ be an arbitrary point on the hyperplane $w^T x + b = 0$ and let $x''$ be an arbitrary point.
- The distance between $x''$ and $w^T x + b = 0$ is equal to the magnitude of the projection of $x'' - x'$ onto $\frac{w}{\|w\|_2}$, the unit vector orthogonal to the hyperplane.
Computing the Margin

• Let \( x' \) be an arbitrary point on the hyperplane \( w^T x + b = 0 \) and let \( x'' \) be an arbitrary point.

• The distance between \( x'' \) and \( w^T x + b = 0 \) is equal to the magnitude of the projection of \( x'' - x' \) onto \( \frac{w}{\|w\|_2} \), the unit vector orthogonal to the hyperplane.
Computing the Margin

• Let \( x' \) be an arbitrary point on the hyperplane and let \( x'' \) be an arbitrary point.

• The distance between \( x'' \) and \( w^T x + b = 0 \) is equal to the magnitude of the projection of \( x'' - x' \) onto \( \frac{w}{\|w\|_2} \), the unit vector orthogonal to the hyperplane.

\[
\frac{|w^T (x'' - x')|}{\|w\|_2} = \frac{|w^T x'' - w^T x'|}{\|w\|_2} = \frac{|w^T x'' + b|}{\|w\|_2}
\]
You should be able to...

- Explain the difference between online learning and batch learning
- Implement the perceptron algorithm for binary classification [CIML]
- Determine whether the perceptron algorithm will converge based on properties of the dataset, and the limitations of the convergence guarantees
- Describe the inductive bias of perceptron and the limitations of linear models
- Draw the decision boundary of a linear model
- Identify whether a dataset is linearly separable or not
- Defend the use of a bias term in perceptron (shifting points after projection onto weight vector)