Perceptrons

Learning Linear Classifiers: The Perceptron

Goal of classification learning:

Given: \((1, x_{11}, x_{12}, \ldots, x_{1n}), y_1\), \((1, x_{21}, x_{22}, \ldots, x_{2n}), y_2\), \ldots, \((1, x_{N1}, x_{N2}, \ldots, x_{Nn}), y_N\)

Find: \((w_0, \ldots, w_n)\)
Linear Classification

General form of a linear classifier

\[ h_{\bar{w}}(x) = \begin{cases} +1 & \text{if } \sum_{i=0}^{n} w_i x_i = \bar{w} \cdot \bar{x} \geq 0 \\ 0 & \text{otherwise} \end{cases} \]

Finding a classifier given a set of data means finding a \( \bar{w} \) for which \( h_{\bar{w}}(\bar{x}_i) \) comes close to \( y_i = f_{\bar{w}}(\bar{x}_i) \) for the training data

How?
Search
Learning with Perceptrons

• Start with an initial $\overline{w}$.
• Take an example $<\overline{x}_i, y_i>$
  • If $h_{\overline{w}}(\overline{x}_i) = y_i$ do nothing
  • Otherwise move $\overline{w}$ to make $\overline{w} \cdot \overline{x}$
    bigger (if $y_i=1$)
    smaller (if $y_i=0$)
• Repeat until you get all the labels right
Learning with Perceptrons

• Start with an initial $\overrightarrow{w}$.
• Take an example $<\overrightarrow{x}_i, y_i>$
  • If $h_{\overrightarrow{w}}(\overrightarrow{x}_i) = y_i$ do nothing
  • Otherwise move $\overrightarrow{w}$ to make $\overrightarrow{w} \cdot \overrightarrow{x}$
    bigger (if $y_i=1$)
    smaller (if $y_i=0$)
• Repeat until you get all the labels right

move $\overrightarrow{w}$?
Learning with Perceptrons

• Start with an initial \( \vec{w} \).
• Take an example \( <\vec{x}_i, y_i> \)
  • If \( h_{\vec{w}}(\vec{x}_i) = y_i \) do nothing
  • Otherwise move \( \vec{w} \) to make \( \vec{w} \cdot \vec{x} \)
    bigger (if \( y_i = 1 \))
    smaller (if \( y_i = 0 \))
• Repeat until you get all the labels right

move \( \vec{w} \)?

\[
\begin{align*}
    w_j &\leftarrow w_j + \alpha x_{ij} (y_i - h_{\vec{w}}(\vec{x}_i)) \\
\end{align*}
\]

Perceptron Learning Rule
Perceptron Learning Rule

\[ w_j \leftarrow w_j + \alpha x_{ij}(y_i - h_w(\bar{x}_i)) \]

- If \( h_w(\bar{x}_i) \) is correct, all \( w_j \) are unchanged
  - \( y_i = h_w(\bar{x}_i) \), so \( (y_i - h_w(\bar{x}_i)) = 0 \)

- If \( h_w(\bar{x}_i) \) is too big, \( w_j \) decreases

- If \( h_w(\bar{x}_i) \) is too small, \( w_j \) increases

- \( \alpha \) is the **learning rate** (sometimes called \( \eta \))
Perceptron Learning Rule

Current hypothesis: $h_{w}(\bar{x})$

$w_0 = w_1 = w_2 = \ldots = w_n = 0$  [alternatively: set to random values]

Repeat

For $i = 1$ to $N$  [for each example]

$h \leftarrow h_{w}(\bar{x}_i)$

For $j = 0$ to $n$  [for each feature]

$w_j \leftarrow w_j + \alpha x_{ij}(y_i - h)$  [Perceptron learning rule]

Until $h_{w}(\bar{x})$ gets all data correct
Perceptron Learning Rule: Example

\[ w_j \leftarrow w_j + \alpha x_{ij}(y_i - h) \]
Perceptron Learning Rule: AND Example

\[ w_j \leftarrow w_j + \alpha x_{ij}(y_i - h) \]
Perceptron Learning Rule: AND Example

$$w_j \leftarrow w_j + \alpha x_{ij}(y_i - h)$$

And gate

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$f(x_1, x_2)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Perceptron Learning Rule: AND Example

\[ w_j \leftarrow w_j + \alpha x_{ij}(y_i - h) \]
\[ \alpha = 0.3, \; w_0 = w_1 = w_2 = 0 \]

Training Data

<table>
<thead>
<tr>
<th>( x_1 )</th>
<th>( x_2 )</th>
<th>( f(x_1, x_2) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Perceptron Learning Rule: 2\textsuperscript{nd} Example

\[ w_j \leftarrow w_j + \alpha x_{ij} (y_i - h) \]
\[ \alpha = 0.3, \quad w_0 = w_1 = w_2 = 0 \]

Training Data

<table>
<thead>
<tr>
<th>( x_1 )</th>
<th>( x_2 )</th>
<th>( f(x_1, x_2) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>0.5</td>
<td>1</td>
</tr>
<tr>
<td>-1</td>
<td>-1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0.5</td>
<td>0</td>
</tr>
<tr>
<td>0.5</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Perceptron Learning Rule: 3rd Example

\[ w_j \leftarrow w_j + \alpha x_{ij}(y_i - h) \]
\[ \alpha = 0.3, \ w_0 = w_1 = w_2 = 0 \]

Training Data

<table>
<thead>
<tr>
<th>( x_1 )</th>
<th>( x_2 )</th>
<th>( f(x_1, x_2) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Perceptron Convergence Theorem

If the training data are linearly separable (there exists a hyperplane that separates the 0’s and 1’s) then the perceptron learning rule (0 < α ≤ 1) will find a separating hyperplane in a finite number of steps.
Perceptrons: The Problem with XOR

(https://medium.com/@claude.coulombe/the-revenge-of-perceptron-learning-xor-with-tensorflow-eb52cbdf6c60)
Perceptron: The Solution for XOR?
Perceptron: The Solution for XOR?
Perceptron: The Solution for XOR?

Multi-layer Neural Network
Multi-Layer Perceptron

Multi-layer Neural Network

1

2

3

4

5

\[ a_5 \]

\[ w_{05} \]

\[ w_{04} \]

\[ w_{13} \]

\[ w_{14} \]

\[ w_{23} \]

\[ w_{24} \]

\[ w_{35} \]

\[ w_{45} \]
Recap

• We know how to train a single perceptron using the perceptron learning rule

• A perceptron can’t represent XOR (and many other functions)

• Use multiple layers

• The perceptron learning rule could be used to train the top-most level of the network, but it’s unclear what to do with lower levels
Training Multi-Layer Perceptrons?

Multi-layer Neural Network
Solution

Replace the hard threshold with a continuous function
(and use calculus)
\[
\frac{1}{1 + e^{-z}}
\]
\( \frac{1}{1 + e^{-\bar{w} \cdot \bar{x}}} \)
Learning

• Consider Error($\bar{w}$,D):

$$\sum_{i=1}^{N} (y_i - h_{\bar{w}}(\bar{x}_i))^2$$
Learning

• Consider Error($\overline{w}, D$):

\[
\sum_{i=1}^{N} (y_i - h_{\overline{w}}(\overline{x}_i))^2
\]

• For a given set of data D, the $\overline{x}_i$’s are fixed
• Different $\overline{w}$ give different values for Error($\overline{w}, D$)
• We want to minimize Error($\overline{w}, D$) by changing the $\overline{w}_i$’s
• We’ll do this incrementally by following the derivative of Error($\overline{w}, D$)
Abstract Representation of a Perceptron

\[ a(\bar{x}) = g(\bar{w} \cdot \bar{x}) \]

where \( g(z) = 1 \) if \( z \geq 0 \)

0 otherwise

Separates weighted sum from threshold
Abstract Representation of a Logistic Neuron

\[ a(\bar{x}) = g(\bar{w} \cdot \bar{x}) \]

where \( g(z) = \frac{1}{1+e^{-z}} \)
Error\(_{D}(\overline{w})\) and Its Gradient

\[
\text{Error}_{D}(\overline{w}) = \sum_{\overline{x}_i \in D} (y_i - h_{\overline{w}}(\overline{x}_i))^2 = \sum_{\overline{x}_i \in D} (y_i - g(\overline{w} \cdot \overline{x}_i))^2
\]

\[
\frac{\partial}{\partial w_j} \text{Error}_{D}(\overline{w}) = \frac{\partial}{\partial w_j} \sum_{\overline{x}_i \in D} (y_i - g(\overline{w} \cdot \overline{x}_i))^2
\]

\[
= \sum_{\overline{x}_i \in D} \frac{\partial}{\partial w_j} (y_i - g(\overline{w} \cdot \overline{x}_i))^2 = -2 \sum_{\overline{x}_i \in D} g'(\overline{w} \cdot \overline{x}_i)(y_i - g(\overline{w} \cdot \overline{x}_i))\overline{x}_{ij}
\]
Logistic Regression Error $\text{Error}_D(\mathbf{w})$ and Its Gradient

$$\frac{\partial}{\partial w_j} \text{Error}_D(\mathbf{w}) = -2 \sum_{\mathbf{x}_i \in D} g'(\mathbf{w} \cdot \mathbf{x}_i)(y_i - g(\mathbf{w} \cdot \mathbf{x}_i)) \mathbf{x}_{ij}$$

$$g'(z) = g(z)(1 - z)$$

for $g(z) = \text{logistic function}$

$$\frac{\partial}{\partial w_j} \text{Error}_D(\mathbf{w}) = -2 \sum_{\mathbf{x}_i \in D} g(\mathbf{w} \cdot \mathbf{x}_i)(1 - g(\mathbf{w} \cdot \mathbf{x}_i))(y_i - g(\mathbf{w} \cdot \mathbf{x}_i)) \mathbf{x}_{ij}$$
Perceptron Learning Algorithm

Current hypothesis: \( h_{\overline{w}}(\overline{x}) \)

Initialize \( w_0, w_1, w_2, \ldots, w_n \)

Repeat

   For \( i = 1 \) to \( N \) [for each example]
   
   \( h \leftarrow h_{\overline{w}}(\overline{x}_i) \)

   For \( j = 0 \) to \( n \) [for each feature]
   
   \( w_j \leftarrow w_j + \alpha x_{ij} (y_i - h) \)

Until \( h_{\overline{w}}(\overline{x}) \) gets all data correct
Gradient Descent Algorithm

Current hypothesis: \( h_{\bar{w}}(\bar{x}) = g(\bar{w} \cdot \bar{x}) \)

Initialize \( w_0, w_1, w_2, \ldots, w_n \)

Repeat

  For \( j = 0 \) to \( n \)  
    \[ \Delta w \leftarrow 0 \]
    For \( i = 1 \) to \( N \)  
      \[ \Delta w \leftarrow \Delta w + g'(\bar{w} \cdot \bar{x}_i)(y_i - g(\bar{w} \cdot \bar{x}_i))\bar{x}_{ij} \]
    \[ w_j \leftarrow w_j + \alpha \Delta w \] \( \text{The -2 is captured in } \alpha \)  

[Randomly reorder the data]

Until stopping condition