Backup plans:

If this Zoom meeting ends prematurely, five-minute break, check Piazza
Perceptrons

Abstract Representation of a Perceptron

\[ a(\bar{x}) = +1 \text{ if } \sum_{i=0}^{n} w_i x_i = \bar{w} \cdot \bar{x} \geq 0 \]

0 otherwise

"bias"

\[ x_0, x_1, x_2 \]

"activation"

\[ w_0, w_1, w_2 \]
Abstract Representation of a Perceptron

\[ a(x) = +1 \text{ if } \sum_{i=0}^{n} w_i x_i = w \cdot \vec{x} \geq 0 \]

0 otherwise

"activation"
Abstract Representation of a Perceptron

\[ a(\bar{x}) = 1 \text{ if } \sum_{i=0}^{n} w_i x_i = \bar{w} \cdot \bar{x} \geq 0 \]

\[ 0 \text{ otherwise} \]
Abstract Representation of a Perceptron

\[ a(x) = +1 \text{ if } \sum_{i=0}^{n} w_i x_i = \bar{w} \cdot \bar{x} \geq 0 \]

0 otherwise
Linear Classification

General form of a linear classifier

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

Finding a classifier given a set of data means finding a \( \bar{w} \) for which \( h_w(\bar{x}_i) \) comes close to \( y_i = f_w(\bar{x}_i) \) for the training data.
Linear Classification

General form of a linear classifier

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

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

General form of a linear classifier

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

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

TBA
Linear Classification

General form of a linear classifier

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

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

General form of a linear classifier

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

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 $\vec{w}$.

• 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 $y_i=1$ and $h_{\vec{w}}(\vec{x}_i)=0$)
    smaller (if $y_i=0$ and $h_{\vec{w}}(\vec{x}_i)=1$)

• Repeat until you get all the labels right
Learning with Perceptrons

• Start with an initial \( \vec{w} \). [Often either all \( w_i = 0 \) or \( w_i = \text{small number} \)]

• Take an example \( \langle \vec{x}_i, y_i \rangle \)
  • If \( h_{\vec{w}}(\vec{x}_i) = y_i \) do nothing
  • Otherwise move \( \vec{w} \) to make \( \vec{w} \cdot \vec{x} \)
    bigger (if \( y_i = 1 \) and \( h_{\vec{w}}(\vec{x}_i) = 0 \))
    smaller (if \( y_i = 0 \) and \( h_{\vec{w}}(\vec{x}_i) = 1 \))

• Repeat until you get all the labels right
Learning with Perceptrons

• Start with an initial $\mathbf{w}$.

• Take an example $<\mathbf{x}_i, y_i>$
  • If $h_{\mathbf{w}}(\mathbf{x}_i)=y_i$ do nothing
  • Otherwise move $\mathbf{w}$ to make $\mathbf{w} \cdot \mathbf{x}$
    bigger (if $y_i=1$ and $h_{\mathbf{w}}(\mathbf{x}_i)=0$)
    smaller (if $y_i=0$ and $h_{\mathbf{w}}(\mathbf{x}_i)=1$)

• Repeat until you get all the labels right
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 $y_i = 1$ and $h_{\overline{w}}(\overline{x}_i) = 0$)
    smaller (if $y_i = 0$ and $h_{\overline{w}}(\overline{x}_i) = 1$)

• Repeat until you get all the labels right

“comes close to”
Learning with Perceptrons

• Start with an initial $\vec{w}$.

• Take an example $\langle \vec{x}_i, y_i \rangle$
  • If $h_{\vec{w}}(\vec{x}_i) = y_i$ do nothing
  • Otherwise move $\vec{w}$ to make $\vec{w} \cdot \vec{x}$
    bigger (if $y_i = 1$ and $h_{\vec{w}}(\vec{x}_i) = 0$)
    smaller (if $y_i = 0$ and $h_{\vec{w}}(\vec{x}_i) = 1$)

• Repeat until you get all the labels right
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 \( \text{move } \overline{w} \) to make \( \overline{w} \cdot \overline{x} \)
    
    bigger (if \( y_i = 1 \) and \( h_{\overline{w}}(\overline{x}_i) = 0 \))
    
    smaller (if \( y_i = 0 \) and \( h_{\overline{w}}(\overline{x}_i) = 1 \))

• Repeat until you get all the labels right
Perceptron Learning Rule

Move $\vec{w}$: Perceptron Learning Rule

\[ w_j \leftarrow w_j + \alpha x_{ij} (y_i - h_{\vec{w}}(\vec{x}_i)) \]
Perceptron Learning Rule

Move $\bar{w}$: Perceptron Learning Rule

$$w_j \leftarrow w_j + \alpha x_{ij} (y_i - h_{\bar{w}}(\bar{x}_i))$$

New value       Old value
Perceptron Learning Rule

Move $\vec{w}$: Perceptron Learning Rule

$$w_j^{t+1} \leftarrow w_j^t + \alpha x_{ij} (y_i - h_\vec{w}^t(\bar{x}_i))$$

New value  Old value
Move $\bar{w}$: Perceptron Learning Rule

\[
w_j \leftarrow w_j + \alpha x_{ij} (y_i - h_{\bar{w}}(\bar{x}_i))
\]
Perceptron Learning Rule

Move $\vec{w}$: Perceptron Learning Rule

$$w_j \leftarrow w_j + \Delta w_j$$

$\Delta w_j = \alpha x_{ij} (y_i - h_{\vec{w}}(\vec{x}_i))$

New value  Old value
Perceptron Learning Rule

Move $\bar{w}$: Perceptron Learning Rule

$$\Delta w_j$$

$$w_j \leftarrow w_j + \alpha x_{ij}(y_i - h_{\bar{w}}(\bar{x}_i))$$
Perceptron Learning Rule

Move $\vec{w}$: Perceptron Learning Rule

$$\Delta w_j$$

$$w_j \leftarrow w_j + \alpha x_{ij}(y_i - h_{\vec{w}}(\vec{x}_i))$$

- If $h_{\vec{w}}(\vec{x}_i)$ is correct ($y_i = h_{\vec{w}}(\vec{x}_i)$), all $w_j$ are unchanged
- If $h_{\vec{w}}(\vec{x}_i)$ is too big ($y_i = 0$ and $h_{\vec{w}}(\vec{x}_i) = 1$), $w_j$ decreases
- If $h_{\vec{w}}(\vec{x}_i)$ is too small ($y_i = 1$ and $h_{\vec{w}}(\vec{x}_i) = 0$), $w_j$ increases
Perceptron Learning Rule

Move $\bar{w}$: Perceptron Learning Rule

$$
\Delta w_j \\
w_j \leftarrow w_j + \alpha x_{ij}(y_i - h_{\bar{w}}(\bar{x}_i))
$$

• $x_{ij}$ gets the sign right and scales the update by the input value
Perceptron Learning Rule

Move $\overline{w}$: Perceptron Learning Rule

$$\Delta w_j$$

$$w_j \leftarrow w_j + \alpha x_{ij}(y_i - h_{\overline{w}}(\vec{x}_i))$$

- $\alpha$ is the learning rate (sometimes called $\eta$)
  - Smaller means smaller increments, more conservative and slower
  - Larger means bigger jumps, can be faster, or can miss solution*
Perceptron Learning Rule

Perceptron\(\{(\bar{x}_i,y_i)_{i\in\{1:N\}}\}\)

\[w_0 = w_1 = w_2 = \ldots = w_n = 0\]

Repeat

For \(i = 1\) to \(N\)  \[\text{[for each example]}\]

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

For \(j = 0\) to \(n\)  \[\text{[for each feature]}\]

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

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

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

\[ w_j \leftarrow w_j + \alpha x_{ij}(y_i - h) \]
Abstract Representation of a Perceptron

\[ a(\bar{x}) = +1 \text{ if } \sum_{i=0}^{n} w_i x_i = \bar{w} \cdot \bar{x} \geq 0 \]
\[ 0 \text{ otherwise} \]
Perceptron Learning Rule: AND 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>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

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

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: 3\textsuperscript{rd} 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 Learning Rule: 3rd Example

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

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>
Perceptrons: The Problem with XOR

(https://medium.com/@claude.coulombe/the-revenge-of-perceptron-learning-xor-with-tensorflow-eb52cbdf6c60)
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 \((0 < \alpha \leq 1)\) will find a separating hyperplane in a finite number of steps.
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 ($0<\alpha\leq 1$) will find a separating hyperplane in a finite number of steps.

Many other theorems. Examples:

- If not linearly separable weights will eventually cycle. ($\alpha$ constant)
- Yields lowest error on data ($\alpha$ decreases as $O(1/\text{iteration-number})$)
Perceptron: The Solution for XOR?

```
\[ a_2 = -0.5 \]
\[ a_1 = 1 \]
\[ x_1 \]
\[ x_2 \]
```
Perceptron: The Solution for XOR?

Multi-layer Neural Network
Multi-Layer Perceptron

Multi-layer Neural Network