Artificial Neurons

Danna Gurari
University of Colorado Boulder
Fall 2022

https://home.cs.colorado.edu/~DrG/Courses/NeuralNetworksAndDeepLearning/AboutCourse.html
Review

• Last lecture:
  • Deep learning applications
  • History of neural networks and deep learning
  • How does a machine learn?
  • Course logistics

• Assignments (Canvas):
  • Problem set 1 due next week

• Questions?
Today’s Topics

• Binary classification applications

• Evaluating classification models

• Biological neurons: inspiration

• Artificial neuron: Perceptron
Today’s Topics

• Binary classification applications

• Evaluating classification models

• Biological neurons: inspiration

• Artificial neuron: Perceptron
Today’s Scope: Binary Classification

Distinguish 2 classes
## Binary Classification: Spam Detection

A screenshot of a Gmail inbox showing various spam messages. Notable messages include:

- **Congrats!!**  
  - (12) Your request has been granted.  
  - 12:27 PM

- **Final Reminder- Hello:**  
  - Last Hour Hire a Book Ghost Writer at 85% Off for Book Writing  
  - 12:13 PM

- **Unsubscribe:**  
  - Dannag, We need your confirmation please.  
  - 11:09 AM

- **WikiPedia:**  
  - Month End Offer! Get your Wikipedia page at 85% off  
  - 10:57 AM

- **Private-MESSAGE:**  
  - Hi I sent some private Image & Video you will be surprised!!  
  - 9:53 AM

- **Paralegal Studies w:**  
  - Study Online, Paralegal Studies  
  - 9:40 AM

- **iM Horny:**  
  - 😂🔥 RokuRebecca has unlocked her private video for you 😂🔥  
  - 9:03 AM

- **utsafetyalert:**  
  - CAMPUS ALERT: All clear issued after threat to main building  
  - 8:57 AM
Binary Classification: Resume Pre-Screening

- Whaii Market
- Prime Talent Chain
- Skillate

- AI and Blockchain Staffing & Recruitment
  - Decentralizing and simplifying the staffing industry
  - Eliminate the intermediaries between the job seeker, through an open ecosystem, and human resource managers by using Blockchain, AI, and other technologies, that ultimately makes the entire process faster and more cost-effective

Automatic screening of resumes
Trained with over 20 million diverse profiles, Skillate’s AI algorithm helps to screen and shortlist resume with the click of a button. Seamlessly integrate with all external channels and ATS to source resume directly

Matching algorithm and candidate recommendation
Skillate’s matching engine maps all the relevant profiles with the job requirements - be it skills, education or experience and recommends the best candidate
Binary Classification: Cancer Diagnosis

Pathology Evolved.
Advanced learning toward faster, more accurate diagnosis of disease.
Binary Classification: Cognitive Impairment Recognition by Apple App Usage

Image Credit: https://www.techradar.com/news/the-10-best-phones-for-seniors
Binary Classification: Sentiment Analysis

Its cast, its attitude, its overall eagerness to please -- all benefits, one would think -- don't add up to a good movie. They add up to a blueprint of the movie this ought to be.

October 23, 2020 | Rating: 2.5/5 | Full Review...

While glorious to look at, the movie still feels slightly hollow. All the right pieces are there, but an emotional connection to the characters is lacking.

September 10, 2020 | Rating: 6.8/10 | Full Review...

K. Austin Collins
Rolling Stone
⭐ TOP CRITIC

Amy Amatangelo
Paste Magazine
⭐ TOP CRITIC
Binary Classification: Food Quality Control

Demo: https://www.youtube.com/watch?v=Bl3XzBWpZbY
Can you think of other binary classification applications?
Today’s Topics

• Binary classification applications

• Evaluating classification models

• Biological neurons: inspiration

• Artificial neuron: Perceptron
Goal: Design Models that **Generalize** Well to New, Previously Unseen Examples

**Example:**

<table>
<thead>
<tr>
<th>Label</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hairy</td>
<td>🐱</td>
</tr>
<tr>
<td>Hairy</td>
<td>🐶</td>
</tr>
<tr>
<td>Not Hairy</td>
<td>🥂</td>
</tr>
<tr>
<td>● ● ●</td>
<td>🕷</td>
</tr>
<tr>
<td>Hairy</td>
<td>● ● ●</td>
</tr>
</tbody>
</table>
Goal: Design Models that **Generalize** Well to New, Previously Unseen Examples

1. Split data into a “**training set**” and “**test set**”

Example:

**Label:**

- Hairy
- Hairy
- Not Hairy

**Example:**

- Cat
- Puppy
- Glass
- Tarantula
Goal: Design Models that **Generalize** Well to New, Previously Unseen Examples

2. Train model on “**training set**” to try to minimize prediction error on it.
Goal: Design Models that **Generalize** Well to New, Previously Unseen Examples

3. Apply trained model on “**test set**” to measure generalization error
Goal: Design Models that **Generalize** Well to New, Previously Unseen Examples

3. Apply trained model on “**test set**” to measure generalization error

![Diagram showing test data process](image-url)
Goal: Design Models that **Generalize** Well to New, Previously Unseen Examples

3. Apply trained model on “**test set**” to measure generalization error

![Diagram showing prediction model and test data example](image)
**Evaluation Methods: Confusion Matrix**

<table>
<thead>
<tr>
<th>Predicted</th>
<th>Actual</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spam</td>
<td>TP</td>
<td>FP</td>
</tr>
<tr>
<td>Not spam</td>
<td>FN</td>
<td>TN</td>
</tr>
</tbody>
</table>

- **TP** = true positive
- **TN** = true negative
- **FP** = false positive
- **FN** = false negative
Evaluation Methods: Descriptive Statistics

Commonly-used statistical descriptions:

- How many *actual spam* results are there? - 65
- How many *actual trusted* results are there? - 110
- How many *correctly classified instances*? - 150/175 ~ 86%
- How many *incorrectly classified instances*? - 25/175 ~ 14%

- What is the precision?
  - $\frac{TP}{TP + FP} ~ 83\%$

- What is the recall?
  - $\frac{TP}{TP + FN} ~ 77\%$
Group Discussion

- Which of these evaluation metrics would you use versus not use and why?
  - Accuracy (percentage of correctly classified examples)
  - Precision
  - Recall

- Scenario 1: Medical test for a rare disease affecting one in every million people.

- Scenario 2: Deciding which emails to flag as spam.
Today’s Topics

• Binary classification applications

• Evaluating classification models

• Biological neurons: inspiration

• Artificial neuron: Perceptron
Inspiration: Animal’s Computing Machinery

Neuron
- basic unit in the nervous system for receiving, processing, and transmitting information; e.g., messages such as...

“hot”

“loud”

“spicy”
https://www.babycenter.com/404_when-can-my-baby-eat-spicy-foods_1368539.bc
Inspiration: Animal’s Computing Machinery

Nematode worm: 302 neurons

Human: ~100,000,000,000 neurons

https://en.wikipedia.org/wiki/Nematode#/media/File:CelegansGoldsteinLabUNC.jpg

https://www.britannica.com/science/human-nervous-system
Inspiration: Animal’s Computing Machinery

Demo (0-1:14): https://www.youtube.com/watch?v=oa6rvUJlg7o
When the input signals exceed a certain threshold within a short period of time, a neuron “fires.” Neuron “firing” is an “all-or-none” process, where either a signal is sent or nothing happens.

Image Source: https://becominghuman.ai/introduction-to-neural-networks-bd042ebf2653
Sidenote: It Remains An Open Research Problem to Understand How Individual Neurons Work
Today’s Topics

• Binary classification applications

• Evaluating classification models

• Biological neurons: inspiration

• Artificial neuron: Perceptron
Historical Context: Artificial Neurons

**First mathematical model of neuron**

- **1943**: First programmable machine
- **1945**: Turing test
- **1950**: Artificialneurons
- **1956**: Al
- **1957**: Machine learning
- **1959**: Perceptron

Emerges from interdisciplinary collaboration

**Warren McCulloch**
- Neurophysiologist
- [Web CSULB](http://web.csulb.edu/~cwallis/artificialn/warren_mcculloch.html)

**Walter Pitts**
- Mathematician
- [Wikipedia](https://en.wikipedia.org/wiki/Walter_Pitts)
Artificial Neuron: McCulloch-Pitts Neuron

Artificial Neuron:

Biological Neuron:

Python Machine Learning; Raschka & Mirjalili

Image Source: https://becominghuman.ai/introduction-to-neural-networks-bd042ebf2653
Artificial Neuron: McCulloch-Pitts Neuron

- inputs \((x)\) and weights \((w)\) can be 0 or 1
- weights \((w)\) and threshold values are fixed
- outputs 1 or 0 (mimics neurons by “firing” only when aggregate value exceeds threshold)

\[
\sum w_1 x_1 + \ldots + w_m x_m = w^T x
\]
Artificial Neuron: McCulloch-Pitts Neuron

- inputs (x) and weights (w) can be 0 or 1
- weights (w) and threshold values are fixed
- outputs 1 or 0 (mimics neurons by “firing” only when aggregate value exceeds threshold)

This neuron representation supports propositional logic; e.g., if weights equal 1 and there are 3 inputs, how is the AND function achieved?

Figure source: Python Machine Learning; Raschka & Mirjalili

Warren McCulloch and Walter Pitts, A Logical Calculus of Ideas Immanent in Nervous Activity, 1943
Artificial Neuron: McCulloch-Pitts Neuron

- inputs (x) and weights (w) can be 0 or 1
- weights (w) and threshold values are fixed

- outputs 1 or 0 (mimics neurons by “firing” only when aggregate value exceeds threshold)

This neuron representation supports propositional logic; e.g., if weights equal 1 and there are 3 inputs, how is the OR function achieved?

Figure source: Python Machine Learning; Raschka & Mirjalili
Warren McCulloch and Walter Pitts, A Logical Calculus of Ideas Immanent in Nervous Activity, 1943
Artificial Neuron: McCulloch-Pitts Neuron

- inputs (x) and weights (w) can be 0 or 1
- weights (w) and threshold values are fixed
- outputs 1 or 0 (mimics neurons by “firing” only when aggregate value exceeds threshold)

This neuron representation supports propositional logic; more examples found at https://home.csulb.edu/~cwallis/artificialn/History.htm

Figure source: Python Machine Learning; Raschka & Mirjalili
Warren McCulloch and Walter Pitts, A Logical Calculus of Ideas Immanent in Nervous Activity, 1943
Artificial Neuron: McCulloch-Pitts Neuron

- inputs (x) and weights (w) can be 0 or 1
- weights (w) and threshold values are fixed
- outputs 1 or 0 (mimics neurons by “firing” only when aggregate value exceeds threshold)

Proposed for computation on a “Turing machine”

Figure source: Python Machine Learning; Raschka & Mirjalili
Warren McCulloch and Walter Pitts, A Logical Calculus of Ideas Immanent in Nervous Activity, 1943
Historical Context: Artificial Neurons

First mathematical model of neuron

1943 1945 1950

First programmable machine

Turing test

AI

Perceptron

Machine learning

1956 1957 1959
“[The perceptron is] the embryo of an electronic computer that [the Navy] expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence.... [It] is expected to be finished in about a year at a cost of $100,000.”


https://en.wikipedia.org/wiki/Frank_Rosenblatt
Perceptron: Architecture (Linear Threshold Unit)

Extends McCulloch-Pitts neuron as follows:
- inputs and weights can be any value
- weights (W) are learned

Python Machine Learning; Raschka & Mirjalili
Frank Rosenblatt, The perceptron, a perceiving and recognizing automaton Project Para. Cornell Aeronautical Laboratory, 1957
Perceptron: Architecture (Linear Threshold Unit)

• Function deciding output value ("fire" or not):

\[
\phi(z) = \begin{cases} 
1 & \text{if } z \geq \theta \\
-1 & \text{otherwise}
\end{cases}
\]

• Rewriting function:

\[
\phi(z) = \begin{cases} 
1 & \text{if } z \geq 0 \\
-1 & \text{otherwise}
\end{cases}
\]

• Where:

\[ z = w_0 x_0 + w_1 x_1 + \ldots + w_m x_m = w^T x \]

* Note: Kamath textbook offers two common conventions for Perceptrons of using two possible output values of \{-1, 1\} and \{0, 1\}, in Chapters 2.5 and 4. The output choice dictates whether the threshold should be set to 0.5 or 0.

Frank Rosenblatt, The perceptron, a perceiving and recognizing automaton Project Para. Cornell Aeronautical Laboratory, 1957.
Perceptron: Architecture (Linear Threshold Unit)

Graphical representation:

Python Machine Learning; Raschkka & Mirjalili
What is the motivation for weights? e.g., for predicting if you will like a movie?
Perceptron: Architecture (Linear Threshold Unit)

Motivation for **bias**: without it, the model must go through the origin.

\[ z = w_0 x_0 + w_1 x_1 + \ldots + w_m x_m = \mathbf{w}^T \mathbf{x} \]

Motivation for bias: with it, model does not have to go through origin

\[ z = w_0 x_0 + w_1 x_1 + \ldots + w_m x_m = w^T x \]
Perceptron: Learning Algorithm

Learns weights and bias values

Python Machine Learning; Raschka & Mirjalili
Perceptron: Learning Algorithm

Process: iteratively update boundary with observation of each additional example:
Perceptron: Learning Algorithm

Process: iteratively update boundary with observation of each additional example:

https://en.wikipedia.org/wiki/Perceptron
Perceptron: Learning Algorithm

1. Initialize weights/bias to 0 or small random numbers
2. For each training sample (i.e., $i$):
   1. Compute predicted value (i.e., {-1, 1}): $\sum_{j=0}^{m} x_j w_j = w^T x$
   2. Update parameters based on prediction success:
      $$\Delta w_j = \eta \left( \text{target}^{(i)} - \text{output}^{(i)} \right) x_j^{(i)}$$

   [Link](https://sebastianraschka.com/faq/docs/diff-perceptron-adaline-neuralnet.html)
Perceptron: Learning Algorithm

1. Initialize weights/bias to 0 or small random numbers
2. For each training sample (i.e., $i$):
   1. Compute predicted value (i.e., $\{-1, 1\}$):
      \[ \sum_{j=0}^{m} x_j w_j = w^T x \]
   2. Update parameters based on prediction success:
      \[ w_j := w_j + \Delta w_j \]

What happens to the weights when the model predicts the **correct** class label?
- no weight update since result is 0

https://sebastianraschka.com/faq/docs/diff-perceptron-adaline-neuralnet.html
Perceptron: Learning Algorithm

1. Initialize weights/bias to 0 or small random numbers
2. For each training sample (i.e., $i$):
   1. Compute predicted value (i.e., {-1, 1}): $\sum_{j=0}^{m} x_j w_j = w^T x$
   2. Update parameters based on prediction success:
      $$\Delta w_j = \eta (\text{target}^{(i)} - \text{output}^{(i)}) x_j^{(i)}$$

What happens to the weights when the model predicts the wrong class label?
- weights change since result is “2” or “-2”

https://sebastianraschka.com/faq/docs/diff-perceptron-adaline-neuralnet.html
Perceptron: Example

- True Model: $Y$ is 1 if at least two of the three inputs are equal to 1.

![Table and Diagram](https://www-users.cs.umn.edu/~kumar001/dmbook/slides/chap4_ann.pdf)
Perceptron: Example

- True Model: $Y$ is 1 if at least two of the three inputs are equal to 1.

![Table and diagram showing input and output connections for perceptron example.](https://www-users.cs.umn.edu/~kumar001/dmbook/slides/chap4_ann.pdf)
Perceptron: Example

• True Model: $Y$ is 1 if at least two of the three inputs are equal to 1.

![Table and Diagram](https://www-users.cs.umn.edu/~kumar001/dmbook/slides/chap4_ann.pdf)
Perceptron: Example

- True Model: Y is 1 if at least two of the three inputs are equal to 1.

<table>
<thead>
<tr>
<th>X₁</th>
<th>X₂</th>
<th>X₃</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>-1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>-1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>-1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-1</td>
</tr>
</tbody>
</table>

Perceptron: Example (Training with 1rst Sample)

- Compute predicted value: \( \sum_{j=0}^{m} x_j w_j = w^T x \); \( \phi(w^T x) = \begin{cases} 1 & \text{if } \phi(w^T x) \geq 0 \\ -1 & \text{otherwise} \end{cases} \)

<table>
<thead>
<tr>
<th>( X_1 )</th>
<th>( X_2 )</th>
<th>( X_3 )</th>
<th>( Y )</th>
<th>Predicted</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>?</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( w_0 )</th>
<th>( w_1 )</th>
<th>( w_2 )</th>
<th>( w_3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Perceptron: Example (Training with 1rst Sample)

• Update params: \( w_j = w_j + \eta (\text{target}^{(i)} - \text{output}^{(i)}) x_j^{(i)} \); learning rate = 0.1

<table>
<thead>
<tr>
<th>( X_1 )</th>
<th>( X_2 )</th>
<th>( X_3 )</th>
<th>( Y )</th>
<th>Predicted</th>
<th>( w_0 )</th>
<th>( w_1 )</th>
<th>( w_2 )</th>
<th>( w_3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

\[
\Delta w_0 = \eta (\text{target}^{(i)} - \text{output}^{(i)}) \\
\Delta w_1 = \eta (\text{target}^{(i)} - \text{output}^{(i)}) x_1^{(i)} \\
\Delta w_2 = \eta (\text{target}^{(i)} - \text{output}^{(i)}) x_2^{(i)} \\
\Delta w_3 = \eta (\text{target}^{(i)} - \text{output}^{(i)}) x_3^{(i)}
\]

Perceptron: Example (Training with 1rst Sample)

• Update params: $w_j = w_j + \eta (\text{target}^{(i)} - \text{output}^{(i)}) x_j^{(i)}$; learning rate = 0.1

Updates make weights more negative so that the model is more likely to classify the sample as -1 next time.

<table>
<thead>
<tr>
<th>$X_1$</th>
<th>$X_2$</th>
<th>$X_3$</th>
<th>$Y$</th>
<th>Predicted</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>$w_0$</th>
<th>$w_1$</th>
<th>$w_2$</th>
<th>$w_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td></td>
</tr>
</tbody>
</table>

$\Delta w_0 = 0.1(-1-1)*1 = -0.2$

$\Delta w_1 = 0.1(-1-1)*1 = -0.2$

$\Delta w_2 = 0.1(-1-1)*0 = 0$

$\Delta w_3 = 0.1(-1-1)*0 = 0$

Perceptron: Example (Training with 2nd Sample)

- Compute output value: \( \sum_{j=0}^{m} x_j w_j = w^T x \); 
  \[ \phi(w^T x) = \begin{cases} 
  1 & \text{if } \phi(w^T x) \geq 0 \\
  -1 & \text{otherwise} 
\end{cases} \]

<table>
<thead>
<tr>
<th>( X_1 )</th>
<th>( X_2 )</th>
<th>( X_3 )</th>
<th>( Y )</th>
<th>Predicted</th>
<th>( w_0 )</th>
<th>( w_1 )</th>
<th>( w_2 )</th>
<th>( w_3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>?</td>
<td>-0.2</td>
<td>-0.2</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Perceptron: Example (Training with 2nd Sample)

- Update params: \( w_j = w_j + \eta \left( \text{target}^{(i)} - \text{output}^{(i)} \right) x_j^{(i)} \); learning rate = 0.1

<table>
<thead>
<tr>
<th>( X_1 )</th>
<th>( X_2 )</th>
<th>( X_3 )</th>
<th>( Y )</th>
<th>Predicted</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>-1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( \Delta w_0 )</th>
<th>( \Delta w_1 )</th>
<th>( \Delta w_2 )</th>
<th>( \Delta w_3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( = \eta \left( \text{target}^{(i)} - \text{output}^{(i)} \right) )</td>
<td>( = \eta \left( \text{target}^{(i)} - \text{output}^{(i)} \right) x_1^{(i)} )</td>
<td>( = \eta \left( \text{target}^{(i)} - \text{output}^{(i)} \right) x_2^{(i)} )</td>
<td>( = \eta \left( \text{target}^{(i)} - \text{output}^{(i)} \right) x_3^{(i)} )</td>
</tr>
</tbody>
</table>

Perceptron: Example (Training with 2nd Sample)

• Update params: \( w_j = w_j + \eta (\text{target}^i - \text{output}^i) x_j^i \); learning rate = 0.1

<table>
<thead>
<tr>
<th>( X_1 )</th>
<th>( X_2 )</th>
<th>( X_3 )</th>
<th>( Y )</th>
<th>Predicted</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>-1</td>
</tr>
</tbody>
</table>

\[
\begin{align*}
\Delta w_0 &= 0.1(1 - 1) * 1 = 0.2 \\
\Delta w_1 &= 0.1(1 - 1) * 1 = 0.2 \\
\Delta w_2 &= 0.1(1 - 1) * 0 = 0 \\
\Delta w_3 &= 0.1(1 - 1) * 1 = 0.2
\end{align*}
\]

Updates make weights more positive so that the model is more likely to classify the sample as 1 next time

Perceptron: Example (Training with 2nd Sample)

- Update params: \( w_j = w_j + \eta (\text{target}^{(i)} - \text{output}^{(i)}) x_j^{(i)} \); learning rate = 0.1

<table>
<thead>
<tr>
<th>( X_1 )</th>
<th>( X_2 )</th>
<th>( X_3 )</th>
<th>( Y )</th>
<th>Predicted</th>
<th>( w_0 )</th>
<th>( w_1 )</th>
<th>( w_2 )</th>
<th>( w_3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>-0.2</td>
<td>-0.2</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

\[
\Delta w_0 = 0.1(1-1) \times 1 = 0.2 \\
\Delta w_1 = 0.1(1-1) \times 1 = 0.2 \\
\Delta w_2 = 0.1(1-1) \times 0 = 0 \\
\Delta w_3 = 0.1(1-1) \times 1 = 0.2
\]

What is the influence of the learning rate? i.e., what would happen if the value was larger/smaller?
Perceptron: Example – One Epoch (Training with All Samples)

• \( w_j = w_j + \eta (\text{target}^{(i)} - \text{output}^{(i)}) x_j^{(i)} \); learning rate = 0.1

<table>
<thead>
<tr>
<th>( X_1 )</th>
<th>( X_2 )</th>
<th>( X_3 )</th>
<th>( Y )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>-1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>-1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>-1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( X_1 )</th>
<th>( X_2 )</th>
<th>( X_3 )</th>
<th>( Y )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( w_0 )</th>
<th>( w_1 )</th>
<th>( w_2 )</th>
<th>( w_3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>-0.2</td>
<td>-0.2</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0.2</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0.2</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0.2</td>
</tr>
<tr>
<td>5</td>
<td>-0.2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>-0.2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0.2</td>
</tr>
<tr>
<td>8</td>
<td>-0.2</td>
<td>0</td>
<td>0.2</td>
</tr>
</tbody>
</table>

Perceptron: Example – Six Epochs

- \( w_j = w_j + \eta (\text{target}^{(i)} - \text{output}^{(i)}) x_j^{(i)} \); learning rate = 0.1

<table>
<thead>
<tr>
<th>X_1</th>
<th>X_2</th>
<th>X_3</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>-1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>-1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>-1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Epoch</th>
<th>w_0</th>
<th>w_1</th>
<th>w_2</th>
<th>w_3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Perceptron: Example – Six Epochs

- \[ w_j = w_j + \eta (\text{target}^{(i)} - \text{output}^{(i)}) x_j^{(i)} \]; learning rate = 0.1

<table>
<thead>
<tr>
<th>X_1</th>
<th>X_2</th>
<th>X_3</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>-1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>-1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>-1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Epoch</th>
<th>w_0</th>
<th>w_1</th>
<th>w_2</th>
<th>w_3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>-0.2</td>
<td>-0.2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.2</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.2</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.2</td>
</tr>
<tr>
<td>5</td>
<td>-0.2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>-0.2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0.2</td>
<td>0.2</td>
</tr>
<tr>
<td>8</td>
<td>-0.2</td>
<td>0</td>
<td>0.2</td>
<td>0.2</td>
</tr>
</tbody>
</table>

Perceptron: Learning Algorithm Choices

- Learning rate
- Number of epochs (passes over the dataset)
Today’s Topics

• Binary classification applications

• Evaluating classification models

• Biological neurons: inspiration

• Artificial neuron: Perceptron
The End
Credits

• Image of Boulder: http://boulderrunning.com/where2run/five-trails-for-hill-running-and-mountain-training/  
• Stick person figure: https://drawception.com/game/AsPNcppPND/draw-yourself-blindfolded-pio/  
• Figure: https://www.quora.com/What-is-meant-by-gradient-descent-in-laymen-terms  
• Figure and great reference: https://beamandrew.github.io/deeplearning/2017/02/23/deep_learning_101_part1.html