Symbolic versus Subsymbolic AI

• **Symbolic AI**
  Everything is represented using symbols.

  \[ A \text{ is a block} \quad \text{Block}(A) \]

  Representation of a state

  Expert Systems, Frames, Scripts, Semantic Nets, Knowledge Graphs etc.

• **Subsymbolic AI**
  There are NO SYMBOLS.

  Approaches that employ Neural Networks and other statistical mechanisms
Neural Networks – “Inspired by the Brain”

A Neural Network

From: https://today.ucsd.edu/story/why_are_neuron_axons_long_and_spindly

From: https://sites.dartmouth.edu/dujs/2015/01/20/neural-networks-real-the-primate-brain-at-object-recognition/
McCulloch-Pitts Neuron, 1943

- Binary Threshold Units
- Captures the inhibitory and excitatory connections between biological neurons.
- Limited in what such a model can actually do.
- Missing the learning capability: how to model changes in inhibitory and excitatory connections. This was later included in Hebb's model (1949). Repeated firings can modify the nature of the connections.
- Frank Rosenblatt, 1958 combined these ideas into the model of a Perceptron.

\[
g(x_1, x_2, x_3, \ldots, x_n) = g(x) = \sum_{i=1}^{n} x_i
\]

\[
y = f(g(x)) =\begin{cases} 1 & \text{if } g(x) \geq \theta \\ 0 & \text{if } g(x) < \theta \end{cases}
\]

From: https://towardsdatascience.com/mcculloch-pitts-model-5f9e65ac5dd1

The Perceptron – A Gross Approximation (Rosenblatt, 1958)

- A single “neuron” (unit) aka Threshold Logic Unit (TLU)
- **Transfer Function**
  T is the Threshold value (assume \( T = 0 \))

\[
I = \sum_{i=1}^{i=n} w_i x_i
\]

\[
y = \begin{cases} +1, & \text{if } I \geq T \\ -1, & \text{if } I < T \end{cases}
\]
The Perceptron

- A single unit
- Transfer Function (assume $T = 0$)

$$I = \sum_{i=1}^{i=n} w_i x_i$$

$$y = \begin{cases} 
+1, & \text{if } I \geq T \\
-1, & \text{if } I < T 
\end{cases}$$
### Training Data

<table>
<thead>
<tr>
<th></th>
<th>$x_1$</th>
<th>$x_2$</th>
<th>Desired Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>0.3</td>
<td>0.7</td>
<td>+1</td>
</tr>
<tr>
<td>B1</td>
<td>-0.6</td>
<td>0.3</td>
<td>-1</td>
</tr>
<tr>
<td>A2</td>
<td>0.7</td>
<td>0.3</td>
<td>+1</td>
</tr>
<tr>
<td>B2</td>
<td>-0.2</td>
<td>-0.8</td>
<td>-1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>$w_1x_1$</th>
<th>$w_2x_2$</th>
<th>Actual Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>-0.18</td>
<td>0.56</td>
<td>+1</td>
</tr>
<tr>
<td>B1</td>
<td>0.36</td>
<td>0.24</td>
<td>+1</td>
</tr>
<tr>
<td>A2</td>
<td>-0.42</td>
<td>0.24</td>
<td>-1</td>
</tr>
<tr>
<td>B2</td>
<td>0.12</td>
<td>-0.56</td>
<td>-1</td>
</tr>
</tbody>
</table>

Incorrect Output!
Perceptron Learning Rule

- Changes the weights

\[ \vec{w} = [w_1, w_2] \] weight vector

\[ \vec{x} = [x_1, x_2] \] input vector

\[ \vec{w}_{new} = \vec{w}_{old} - y^* \vec{x} \] Training Rule

Training Data

<table>
<thead>
<tr>
<th></th>
<th>( x_1 )</th>
<th>( x_2 )</th>
<th>Desired Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>0.3</td>
<td>0.7</td>
<td>+1</td>
</tr>
<tr>
<td>B1</td>
<td>-0.6</td>
<td>0.3</td>
<td>-1</td>
</tr>
<tr>
<td>A2</td>
<td>0.7</td>
<td>0.3</td>
<td>+1</td>
</tr>
<tr>
<td>B2</td>
<td>-0.2</td>
<td>-0.8</td>
<td>-1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>( w_1 x_1 )</th>
<th>( w_2 x_2 )</th>
<th>Actual Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>-0.18</td>
<td>0.56</td>
<td>+1</td>
</tr>
<tr>
<td>B1</td>
<td>0.36</td>
<td>0.24</td>
<td>+1</td>
</tr>
<tr>
<td>A2</td>
<td>-0.42</td>
<td>0.24</td>
<td>-1</td>
</tr>
<tr>
<td>B2</td>
<td>0.12</td>
<td>-0.56</td>
<td>-1</td>
</tr>
</tbody>
</table>
Vocabulary

- **Labelled Training Dataset**
  N samples/patterns/input vector with desired outputs (targets/labels)

- **Output Error (Loss)**
  Error = Desired Output - Actual Output

- **Learning Rule**
  Specifies change in the weights using the Error

- **Prediction/Forward Pass**
  Application of a pattern to produce output

- **Epoch**
  1 pass through the training dataset

Vocabulary - Perceptron

- **Labelled training Dataset**
  N samples/patterns/input vector with desired outputs (targets/labels)

- **Output Error (Loss)**
  $y$ is the perceptron’s answer/output

  $\beta = \begin{cases} 
  +1, & \text{if perceptron’s answer is correct} \\
  -1, & \text{if perceptron’s answer is wrong}
  \end{cases}$

- **Learning Rule**
  Specifies change in the weights using the Error

  Perceptron Learning Rule:
  
  $$w_{\text{new}} = w_{\text{old}} + \beta y^* x$$

- **Prediction/Forward Pass**
  Application of a pattern to produce output

- **Epoch**
  1 pass through the training dataset
Perceptron Training Algorithm

Initialize all weights to random values
   #In what range? Typically [-1.0..1.0]
Set #Epochs to some N
   // How to decide what N should be?
Do N times or until all outputs are correct
   Do for each pattern in the training set
      apply the pattern to the perceptron
      change the weight vector as defined

Example – Iris Dataset

• 150 Samples, 50 of each variety

https://peaceadegbite1.medium.com/iris-flower-classification-60790e9718a1

inputs
\[ w_1 \]
\[ w_2 \]

outputs
\[ y = \begin{cases} 
0, & \text{if Setosa} \\
1, & \text{if Versicolor} 
\end{cases} \]

Sepal Length & Width
\[ [5.1, 3.5, 1.4, 0.2], \]
\[ [4.9, 3.0, 1.4, 0.2], \]
\[ [4.7, 3.2, 1.3, 0.2], \]
\[ [4.6, 3.1, 1.5, 0.2], \]
\[ \ldots \]

Petal Length & Width

Petal Length & Width

Sepal Length & Width


Iris setosa  Iris versicolor  Iris virginica

https://peaceadegbite1.medium.com/iris-flower-classification-60790e9718a1
Example – The Iris Dataset

• 150 Samples, 50 of each variety
  Labelled: 0 (Setosa), 1 (Versicolor), 2 (Virginica)

[5.1, 3.5, 1.4, 0.2],
[4.9, 3. , 1.4, 0.2],
[4.7, 5.2, 1.3, 0.2],
[4.6, 3.1, 1.5, 0.2],
...


https://peaceadegbite1.medium.com/iris-flower-classification-60790e9718a1

Introducing Google Colab

• Live demo...

• Writing a Perceptron from scratch...
References


