Machine Learning

Subfield of AI concerned with learning from data.

Broadly, using:

• Experience
• To Improve Performance
• On Some Task

(Tom Mitchell, 1997)
Recall: Supervised Learning

Formal definition:

Given training data:
\[ X = \{x_1, \ldots, x_n\} \quad \text{inputs} \]
\[ Y = \{y_1, \ldots, y_n\} \quad \text{labels - if discrete: classification} \]

Produce:
Decision function \( f : X \rightarrow Y \)

That minimizes error:
\[
\sum_i \text{err}(f(x_i), y_i)
\]
Decision Trees

```
<table>
<thead>
<tr>
<th>a?</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
</tr>
<tr>
<td>false</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>b?</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
</tr>
<tr>
<td>y=1</td>
</tr>
<tr>
<td>false</td>
</tr>
<tr>
<td>y=2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>c?</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
</tr>
<tr>
<td>false</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>b?</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
</tr>
<tr>
<td>y=2</td>
</tr>
<tr>
<td>false</td>
</tr>
<tr>
<td>y=1</td>
</tr>
</tbody>
</table>
```
The Perceptron

Explicit decision boundary.
The Perceptron

If $x = [x(1), \ldots, x(n)]$:
- Create an $n$-d line
- Slope for each $x(i)$
- Constant offset

$$f(x) = \text{sign}(w \cdot x - c)$$

gradient

offset
The Perceptron

How do you reduce error?

\[ e = (y_i - (w \cdot x_i + c))^2 \]

\[ \frac{\partial e}{\partial w_j} = -2(y_i - (w_i \cdot x_i + c))x_i(j) \]

descend this gradient to reduce error
Perceptrons
Neural Networks

\[ \sigma(w \cdot x + c) \]

logistic regression
Neurons

- Cell body
- Axon
- Axon hillock
- Nucleus
- Endoplasmic reticulum
- Mitochondrion
- Golgi apparatus
- Dendrite
- Dendritic branches
- Telodendria
- Synaptic terminals
Neural Networks

input layer

hidden layer

output layer

$h_1$, $h_2$, $h_3$, $o_1$, $o_2$, $x_1$, $x_2$
Neural Networks

\[ \sigma(w_{h1}^{o1}h_1 + w_{h2}^{o1}h_2 + w_{h3}^{o1}h_3 + w_{h4}^{o1}) \]

value computed

\[ \sigma(w_{h1}^{o2}h_1 + w_{h2}^{o2}h_2 + w_{h3}^{o2}h_3 + w_{h4}^{o2}) \]

\[ h_1 = \sigma(w_{h1}^{h1}x_1 + w_{h2}^{h1}x_2 + w_{h3}^{h1}) \]

value computed

input layer

\[ x_1, x_2 \in [0, 1] \]
Neural Networks

probability of class 1

\( \sigma(w_1 x_1 + w_2 x_2 + w_3 x_3 + w_4) \)

probability of class 2

\( \sigma(w_1 x_1 + w_2 x_2 + w_3 x_3 + w_4) \)

weights (parameters)

input data

\( x_1, x_2 \in [0, 1] \)
A neural network is just a parametrized function: \( y = f(x, w) \)

How to train it?

Write down an error function:

\[
(y_i - f(x_i, w))^2
\]

Minimize it! (w.r.t. \( w \))
Neural Classification

Recall that the *squashing function* is defined as:

\[
\sigma(t) = \frac{1}{1 - e^{-t}}
\]

\[
\frac{\partial \sigma(t)}{\partial t} = \sigma(t)(1 - \sigma(t))
\]
Neural Classification

OK, so we can minimize error using gradient descent.

To do so, we must calculate $\frac{\partial e}{\partial w_i}$ for each $w_i$.

How to do so? Easy for output layers:

$$
\frac{\partial e}{\partial w_i} = \frac{\partial (y_i - o_i)^2}{\partial w_i} = 2(y_i - o_i) \frac{\partial (y_i - o_i)}{\partial w_i} = 2(o_i - y_i) o_i (1 - o_i)
$$

chain rule

Interior weights: repeat chain rule application.
Backpropagation

This algorithm is called *backpropagation*.

Bryson and Ho, 1969
Deep Neural Networks
Nonparametric Methods

Most ML methods are parametric:
- Characterized by setting a few parameters.
- \( y = f(x, w) \)

Alternative approach:
- Let the data speak for itself.
- No finite-sized parameter vector.
- Usually more interesting decision boundaries.
K-Nearest Neighbors

Given training data:
\[ X = \{x_1, \ldots, x_n\} \]
\[ Y = \{y_1, \ldots, y_n\} \]

Distance metric \( D(x_i, x_j) \)

For a new data point \( x_{\text{new}} \):
find \( k \) nearest points in \( X \) (measured via \( D \))
set \( y_{\text{new}} \) is the majority label
K-Nearest Neighbors
K-Nearest Neighbors

Properties:
- No learning phase.
- Must store all the data.
- $\log(n)$ computation per sample - grows with data.

Decision boundary: *any function, given enough data.*

*Classic trade-off:* memory and compute time for flexibility.
Applications

• Fraud detection
• Internet advertising
• Friend or link prediction
• Sentiment analysis
• Face recognition
• Spam filtering
“A breakthrough in machine learning would be worth ten Micrososfts.”
- Bill Gates