NEURAL NETWORKS

Chapter 20, Section 5

Outline

- Brains
- Neural networks
- Perceptrons
- Multilayer perceptrons
- Applications of neural networks

Brains

10^{11} neurons of > 20 types, 10^{14} synapses, 1ms–10ms cycle time
Signals are noisy “spike trains” of electrical potential

McCulloch–Pitts “unit”

Output is a “squashed” linear function of the inputs:

\[ a_i \leftarrow g(\text{in}_i) = g(\sum W_{ij} a_j) \]

A gross oversimplification of real neurons, but its purpose is to develop understanding of what networks of simple units can do

Activation functions

(a) is a step function or threshold function
(b) is a sigmoid function \( \frac{1}{1 + e^{-x}} \)

Changing the bias weight \( W_0 \) moves the threshold location

Implementing logical functions

McCulloch and Pitts: every Boolean function can be implemented

\( W_0 = 1.5 \)
\( W_1 = 1 \)
\( W_2 = 1 \)

AND

\( W_0 = 0.5 \)
\( W_1 = 1 \)
\( W_2 = 1 \)

OR

\( W_0 = -0.5 \)
\( W_1 = 1 \)
\( W_2 = 1 \)

NOT

10^{11} neurons of > 20 types, 10^{14} synapses, 1ms–10ms cycle time
Signals are noisy “spike trains” of electrical potential
Network structures

Feed-forward networks:
- single-layer perceptrons
- multi-layer perceptrons

Feed-forward networks implement functions, have no internal state

Recurrent networks:
- Hopfield networks have symmetric weights ($W_{i;j} = W_{j;i}$)
- Boltzmann machines use stochastic activation functions, $\approx$ MCMC in Bayes nets
- recurrent neural nets have directed cycles with delays
  $\Rightarrow$ have internal state (like flip-flops), can oscillate etc.

Expressiveness of perceptrons

Consider a perceptron with $g = \text{step function}$ (Rosenblatt, 1957, 1960)
Can represent AND, OR, NOT, majority, etc., but not XOR

Represents a linear separator in input space:

\[
\sum_j W_j x_j > 0 \quad \text{or} \quad W \cdot x > 0
\]

Minsky & Papert (1969) pricked the neural network balloon

Perceptron learning

Learn by adjusting weights to reduce error on training set

The squared error for an example with input $x$ and true output $y$ is

\[
E = \frac{1}{2} \text{Err}^2 \equiv \frac{1}{2} (y - h_W(x))^2
\]

Perform optimization search by gradient descent:

\[
\frac{\partial E}{\partial W_j} = \text{Err} \cdot g'(m) \cdot x_j
\]

Simple weight update rule:

\[
W_j \leftarrow W_j + \alpha \cdot \text{Err} \times g'(m) \cdot x_j
\]

E.g., +ve error $\Rightarrow$ increase network output
  $\Rightarrow$ increase weights on +ve inputs, decrease on -ve inputs

Perceptron learning contd.

Perceptron learning rule converges to a consistent function for any linearly separable data set

Output units all operate separately—no shared weights

Adjusting weights moves the location, orientation, and steepness of cliff

Perceptron learns majority function easily, DTL is hopeless

DTL learns restaurant function easily, perceptron cannot represent it
Multilayer perceptrons

Layers are usually fully connected; numbers of hidden units typically chosen by hand.

Output units \( a_i \)

Hidden units \( a_j \)

Input units \( a_k \)

\[
\begin{align*}
\text{Output layer: same as for single-layer perceptron,} \\
W_{ij} &\leftarrow W_{ij} + \alpha \times a_j \times \Delta_i \\
\text{Hidden layer: back-propagate the error from the output layer:} \\
\Delta_j &\leftarrow g'(m_j) \sum_i W_{ij} \Delta_i \\
\text{Update rule for weights in hidden layer:} \\
W_{kj} &\leftarrow W_{kj} + \alpha \times a_k \times \Delta_j
\end{align*}
\]

(Most neuroscientists deny that back-propagation occurs in the brain.)
Back-propagation learning contd.

Learning curve for MLP with 4 hidden units:

MLPs are quite good for complex pattern recognition tasks, but resulting hypotheses cannot be understood easily.

Handwritten digit recognition

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
</tbody>
</table>

3-nearest-neighbor = 2.4\% error
400–300–10 unit MLP = 1.6\% error
LeNet: 768–192–30–10 unit MLP = 0.9\% error
Current best (kernel machines, vision algorithms) = 0.6\% error

Summary

Most brains have lots of neurons; each neuron \approx\ linear–threshold unit (?)
Perceptrons (one-layer networks) insufficiently expressive
Multi-layer networks are sufficiently expressive; can be trained by gradient descent, i.e., error back-propagation
Many applications: speech, driving, handwriting, fraud detection, etc.
Engineering, cognitive modelling, and neural system modelling subfields have largely diverged