Neural networks intro

Minds & Machines
SymSys 1/Phil 99/Psych 35/Ling 35
11/7/16
Names

- neural networks
- deep learning
- parallel distributed processing
- connectionism
‘it is likely that connectionist models will offer the most significant progress of the past several millennia on the mind/body problem’

- Smolenksy (1988)
'The theory that the brain is a symbol-manipulator rose to popularity among cognitive scientists not so much because there is hard evidence in its favor but because it seemed to be the ‘only straw afloat’. There seemed no plausible alternative conception of how the brain might function. PDP has changed all this …

-Copeland (1993)
Cognitive attraction of neural nets

- computational architecture mimics brain
- parallel, not serial
- **distributed** representations
- no strict distinction between memory and processing – like the brain (?)
- ‘graceful degradation’, context-sensitivity, similarity-based generalization, …

more on this in Jay’s lecture
Engineering side

• huge advances in recent years
  – vision
  – speech recognition, language processing
  – autonomous vehicles
  – others where we can find or generate lots of data
  – many tasks that are very difficult to do top-down

• some driving forces
  – availability of (much) more data
  – new hardware (GPUs), software
  – new architectures

• more on this in Andrej’s lecture
McCulloch-Pitts neurons

a.k.a ‘perceptrons’

fires if sum of inputs x weights ≥ threshold
McCulloch-Pitts neurons
McCulloch-Pitts neurons
Multiple inputs

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>(A \text{ OR } B)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

\(A\) \(B\) \(A \text{ AND } B\)

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
learning procedure (simplified!!)

1. Start with whatever weights, thresholds
2. Calculate current output for inputs
3. Modify weights in direction that would produce desired output, by error x learning rate
4. Repeat

\[ \eta = 0.1 \]

<table>
<thead>
<tr>
<th>input</th>
<th>current output</th>
<th>desired output</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1,0)</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

direction/amount of changes?
Learning AND, starting with OR

\[ \eta = .1 \]

<table>
<thead>
<tr>
<th>input</th>
<th>current output</th>
<th>desired output</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1,0)</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Learning AND, starting with OR

\[ \eta = .1 \]

<table>
<thead>
<tr>
<th>input</th>
<th>current output</th>
<th>desired output</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1,0)</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Learning AND, starting with OR

\[ \eta = 0.1 \]

<table>
<thead>
<tr>
<th>input</th>
<th>current output</th>
<th>desired output</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1,0)</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Learning AND, starting with OR

\[ \eta = .1 \]

<table>
<thead>
<tr>
<th>input</th>
<th>current output</th>
<th>desired output</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1,0)</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Learning AND, starting with OR

\[ \eta = .1 \]

<table>
<thead>
<tr>
<th>input</th>
<th>current output</th>
<th>desired output</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1,0)</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Learning AND, starting with OR

\[ \eta = .1 \]

<table>
<thead>
<tr>
<th>input</th>
<th>current output</th>
<th>desired output</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1,0)</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Learning AND, starting with OR

\[ \eta = 0.1 \]

<table>
<thead>
<tr>
<th>input</th>
<th>current output</th>
<th>desired output</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1,0)</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Another way to visualize

A linearly separable function

OR
Another way to visualize

A linearly separable function

OR
Another way to visualize

A linearly separable function

OR
Another way to visualize

A linearly separable function

OR
Another way to visualize

A linearly separable function

OR
Another way to visualize

A linearly separable function

AND
The ‘bombshell’

Minsky & Papert 1979, *Perceptrons*:

A single-layer NN can’t compute – and so, can’t learn – XOR.

Why not?

– AND, OR are linearly separable
– XOR is not
A non-linearly separable function
A non-linearly separable function
The ‘bombshell’


XOR can’t be learned by a single-layer NN.

- Unlearnable? No – you know how to build it!
- Real problem: Rosenblatt’s learning method didn’t generalize to multilayer networks

This finding (unfairly) killed most NN research for a time
We need **two** classifiers working together

![A non-linearly separable function](chart.png)
Let’s train a multilayer network!

http://playground.tensorflow.org/
Multilayer networks

Figure 2.1
General multilayer-perceptron architecture: Input nodes, hidden nodes, and output nodes attached to each other by weighted connections.
Multilayer networks

- can have many layers
- ‘distort’ input space so it’s linearly separable
- typically don’t use thresholds – fancier non-linear ‘activation functions’ instead

```
<table>
<thead>
<tr>
<th>Total input to unit</th>
<th>Output activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Threshold</td>
<td>Sigmoid</td>
</tr>
<tr>
<td></td>
<td>ReLU</td>
</tr>
</tbody>
</table>
```
Backpropagation

Intuition:

- forward pass: each layer provides input to next
- backward pass: each layer provides error signal to previous
Learning to classify digits

MINIST database
A recent objection

**Human-level concept learning through probabilistic program induction**

Brenden M. Lake,^1^* Ruslan Salakhutdinov,^2^ Joshua B. Tenenbaum^3^

People learning new concepts can often generalize successfully from just a single example, yet machine learning algorithms typically require tens or hundreds of examples to perform with similar accuracy. People can also use learned concepts in richer ways than conventional algorithms—for action, imagination, and explanation. We present a computational model that captures these human learning abilities for a large class of simple visual concepts: handwritten characters from the world’s alphabets. The model represents concepts as simple programs that best explain observed examples under a Bayesian criterion. On a challenging one-shot classification task, the model achieves human-level performance while outperforming recent deep learning approaches. We also present several “visual Turing tests” probing the model’s creative generalization abilities, which in many cases are indistinguishable from human behavior.

*Science*, December 2015
A recent objection

cf. ‘probabilistic language of thought’
flashback to memory

Gallistel & King (2009): neural networks are essentially big finite-state machines

Arguably true of multilayer perceptrons

But …

– **recurrent** nets have a kind of read/write memory

– ever-more-complex architectures in recent work
Recurrent neural nets can cope with sequenced inputs, outputs memory in addition to state-dependence – write passable sonnets, sci-fi, JavaScript. [Sunspring]
Hybrid computing using a neural network with dynamic external memory

Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, Adrià Puigdomènech Badia, Karl Moritz Hermann, Yori Zwols, Georg Ostrovski, Adam Cain, Helen King, Christopher Summerfield, Phil Blunsom, Koray Kavukcuoglu & Demis Hassabis

‘a neural network that can read from and write to an external memory matrix … it can use its memory to represent and manipulate complex data structures …’
This just in … (Oct 12)

Illustration of the DNC architecture. The neural network controller receives external inputs and, based on these, interacts with the memory using read and write operations known as 'heads'. To help the controller navigate the memory, DNC stores 'temporal links' to keep track of the

https://deepmind.com/blog/differentiable-neural-computers/
Some famous objections

• symbolic computation needed to explain **productivity** of language, thought, etc
  – Fodor & Pylyshyn: productive use of concepts
  – Pinker: rule-based generalization
  – long-distance dependencies: continuing debate about whether RNNs solve this

• need read/write memory (Gallistel)

• not as really biologically realistic as sold
  – backprop requires varying between excitatory & inhibitory connections
  – no neurotransmitters, etc
Next 3 classes

‘Deep learning’
Andrej Karpathy

‘Parallel distributed processing’
Jay McClelland

‘Vision and the brain’
Justin Gardner