Introduction

Admin

1. Course web page: `https://github.com/sje30/dl2019`
2. Office hour: Friday 12-1pm (3-4pm on Friday 1 Mar).
3. One practical assignment to be set at end of term.
4. Key reference placed in paperpile: `https://paperpile.com/shared/pb4w0p`.
5. Key texts: Chollet (with Allaire), Goodfellow et al ("bible"), Stone (coming soon).

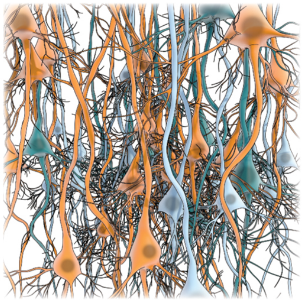
Goals of course

1. Introduce key theory for supervised learning
2. Practical issues (computing)
3. Applications (AstraZeneca)

What is a neural network? (Ullman, 2019)

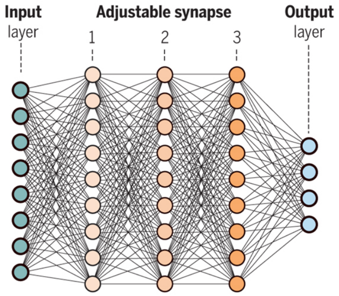
Brain circuitry and learning

A major open question is whether the highly simplified structures of current network models compared with cortical circuits are sufficient to capture the full range of human-like learning and cognition.



Complex neural network

Connectivity in cortical networks includes rich sets of connections, including local and long-range lateral connectivity, and top-down connections from high to low levels of the hierarchy.



Informed AI network

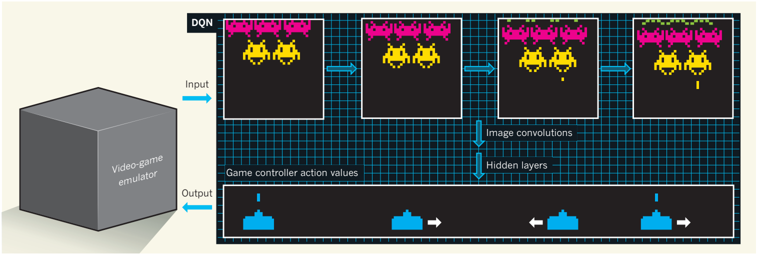
Biological innate connectivity patterns provide mechanisms that guide human cognitive learning. Discovering similar mechanisms, by machine learning or by mimicking the human brain, may prove crucial for future artificial systems with human-like cognitive abilities.

Why are they useful?

1. Speech recognition (Android) since 2012 (LeCun et al 2015).
2. Image recognition since 2012.
3. Atari video games (2015)
4. Go and Chess (Silver et al 2016; 2018).

Deep reinforcement learning: breakout

Mnih et al (2015) Human-level control through deep reinforcement learning. Nature 518:529–533.



System played better than professional human on 49 Atari 2600 games.
`https://www.youtube.com/watch?v=V1eYniJ0Rnk`

A brief history of Neural networks

1. McCulloch and Pitts (1943): all-or-nothing model of neurons.
2. Rosenblatt (1957): perceptron – mechanism for learning based on Hebb (1949).
3. Limitations of perceptrons: Minksy and Papert (1969)
4. 1973: Lighthill report led to first AI winter.
5. Backpropagation (Werbos; 1974). Popularised by Hinton et al in 1980s.
6. Limitations in hardware led to 2nd AI winter late 1990s.
7. 2012: resurgence due to hardware and some new "tricks".
8. Computational biology now a big user (Angermueller et al 2016), e.g. protein folding (ALphaFold) and biomedicine (Ching et al 2018).

See Schmidhuber (2015) for further history.

Why now?

1. Advances in computational hardware (GPU, CPU, TPU)
2. Some algorithmic developments, help in training
3. Advent of big data: many more samples now than ever before

Practical matters

- Like computational biology, 80% of the work is mundane but critical (data collection, cleaning, hyper-parameter selction).
- Good news: many frameworks. We will use Keras (with Tensor Flow backend).
- GPU vs CPU
- Desktop vs Cloud

Types of learning

1. Learning with a teacher: supervised learning.
2. Learning with a critic: reinforcement learning.
3. Learning on your own: unsupervised learning.

- How we study learning in neural networks:

  Fixed architecture, alter synapses/connection strengths

Classification

Formal notion of classification: input vectors $\mathbf{x}$ associated with output vectors $\mathbf{y}$. Learn mapping. Generalise to data not seen during learning. ("Training set" vs "test set" and also "validation set").
Approaches to classification:

1. Logistic regression (binary outputs). Applied Statistics.
2. Naive Bayes. Machine Learning / probablistic modelling.
3. Multi-layer perceptron. Neural networks part I.
4. Support vector machines. Kernel methods.
5. Decision Trees and Forests.
6. Neural networks part II.

The perceptron (Rosenblatt 1957)

Single layer perceptron, with activation function

Activity now summarised as firing rate (spikes/sec). Notation:

- x_i: activity of input unit i (binary or [0,1]).
- y: activity of output unit.
- t: desired output (of use later). (μ superscript denotes training sample.)
- $f(\cdot)$: transfer function:
    1. linear: f(z) = z
    2. threshold: $f(z, \theta) = \begin{cases} 1 & \text{if } z \geq \theta \\ 0 & \text{otherwise} \end{cases}$
    3. sigmoidal: $f(z) = \frac{1}{1+\exp(-z)}$

Transfer function

Given total weighted input to neuron, what is its output?

1. identity: $f(z) = z$.
2. threshold: $f(z, \theta) = \begin{cases} 1 & \text{if } z \geq \theta \\ 0 & \text{otherwise} \end{cases}$
3. sigmoidal: $f(z) = \frac{1}{1+\exp(-kz)}$
4. tanh: $f(z) = tanh(z)$
5. rectified linear unit (ReLU): $f(z) = max(0, z)$

How to choose? One key property: differentiable.

Perceptron decisions

Whether a perceptrons output is 0 or 1 depends on whether $\sum_{i=1}^{N} w_i x_i$ is less or greater than θ.

The equation

$$\sum_{i=1}^{N} w_i x_i = \theta$$

defines a hyperplane in N-dimensional space.

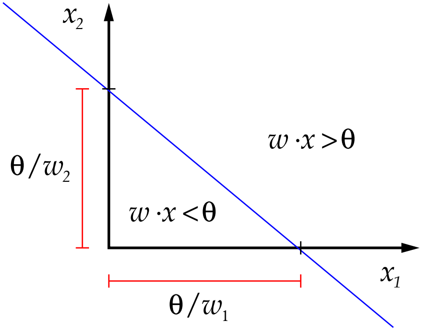
This hyperplane cuts the space in two.

N=2

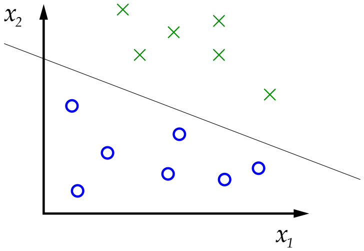
$$w_1 x_1 + w_2 x_2 = \theta$$

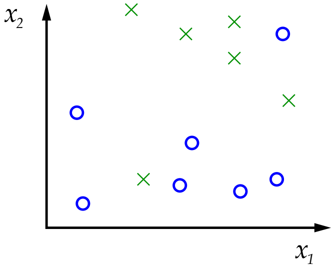
$$x_2 = \left(\frac{-w_1}{w_2}\right) x_1 + \frac{\theta}{w_2}$$

This is an equation for a straight line.



Linearly separable problems and the perceptron



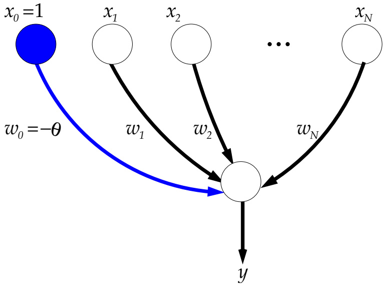


Learning involves adjusting the values of w and θ so that the decision plane can correctly divide the two classes.

Threshold & Bias

The threshold, θ, can be treated as just another weight from a new input unit which always has a value of 1.

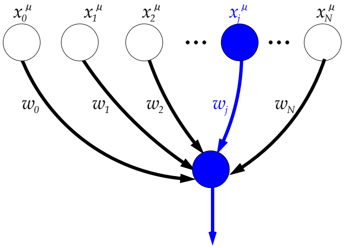
This new input unit is called the **bias** unit.



Instead of comparing $\sum_{i=1}^{N} w_i x_i$ with θ, we compare $\sum_{i=0}^{N} w_i x_i$ with 0, or

$$y = \begin{cases} 1 & \text{if } \sum_{i=0}^{N} w_i x_i > 0, \\ 0 & \text{if } \sum_{i=0}^{N} w_i x_i \leqslant 0. \end{cases}$$

Intuitively... (positive valued inputs)



$$y^\mu = \text{step}\left(\sum_{i=0}^{N} w_i x_i^\mu\right)$$

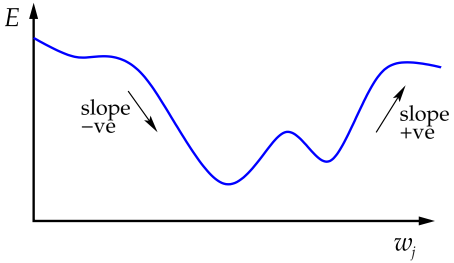
Consider w_j's contribution to $\sum_i w_i x_i^\mu$ in different cases:

1. $y^\mu = t^\mu$ Perceptron has classified input μ correctly – change nothing
2. $x_j^\mu = 0$ Changing w_j will not affect the $\sum_i w_i x_i^\mu$ – change nothing
3. $x_j^\mu \neq 0, y^\mu < t^\mu$ The sum $\sum_i w_i x_i^\mu$ is too low – so increase it
4. $x_j^\mu \neq 0, y^\mu > t^\mu$ The sum $\sum_i w_i x_i^\mu$ is too high – so decrease it

The local rule:

$$\boxed{\Delta w_j \propto (t^\mu - y^\mu)\, x_j^\mu}$$

Perceptron learning rule



$$y = f(\mathbf{w} \cdot \mathbf{x}) \qquad \text{e.g.} \quad f(z) = 1/(1+\exp(-z)), \quad f(z) = z$$

$$E = \frac{1}{2}(t-y)^2 \qquad t \text{ is target output}$$

$$\Delta w_j = -\epsilon \frac{\partial E}{\partial w_j} = -\epsilon \frac{\partial E}{\partial y} \frac{\partial y}{\partial w_j} =$$

This is the method of gradient descent with learning-rate parameter ϵ.

Perceptron convergence theorem (PCT)

Starting from initial weights $\mathbf{w} = \mathbf{0}$, the perceptron convergence theorem guarantees that *if* there are ideal weights $\mathbf{w}^*$ separating two classes, the perceptron rule will stop training after finite number of updates.
The network weights $\mathbf{w}$ at the end of training will then perfectly solve the classification task.
(Note that $\mathbf{w}$ will be similar, but not necessarily equal, to $\mathbf{w}^*$.)

See (Dayan and Abbott, page 327) for proof of convergence.

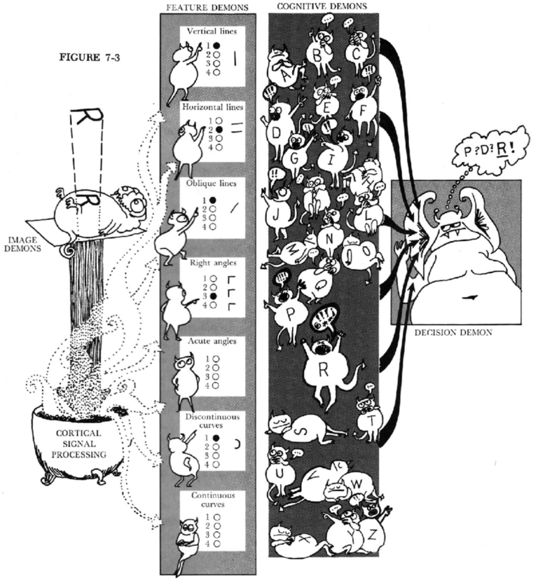
Perceptron pros and cons

- The *perceptron convergence theorem* guarantees that solution will be found iff there is a linearly separable solution.
- Many complex problems are not linearly separable, e.g. XOR problem.

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

Multi-layer perceptrons (MLPs)

The importance of features / 1



(Lindsay and Norman's view of Selfridge's Pandemonium model, 1959).

The importance of features / 2

- Find the right features to make the task solvable.
- Classification of points in an annulus.
- Engineering features by hand is hard.
- Neural networks learn features that they find important.

How many layers of features do you need?

In theory, one hidden layer is all you need **in theory** to make a “universal approximator” (Cybenko 1989; Hornik 1991).
With linear transfer functions how many layers do we need?

Neural networks and linear algebra

- Activation of a layer of neurons stored in a **vector**.
- Synapses from one layer to another stored in a **weight matrix**: $\mathbf{W}_{ji}$ is strength of connection from unit *i* in one layer to unit *j* in the next layer.
- "The single key fact about vectors and matrices is that each vector represents a point located in space, and a matrix moves that point to a different location. Everything else is just details." (Stone 2019, appendix C).

drawing of weight matrices