3041 Neural computation: Models of brain function

- **Course organisers - contact details**
  Dr Caswell Barry: e-mail: caswell.barry@ucl.ac.uk
  Prof Neil Burgess: e-mail: n.burgess@ucl.ac.uk

[www.ucl.ac.uk/icn/people/space-and-memory/space-leader/neil-teaching](http://www.ucl.ac.uk/icn/people/space-and-memory/space-leader/neil-teaching)

**C41 – Half unit. Aims**

- To introduce the consideration of neurons and synapses in terms of their computational properties, and interpretation of their action in terms of information processing.
- To introduce the analysis of an animal’s ability to learn, remember or act in terms of the action of neurons and synapses within the animal’s nervous system.
- To understand several examples of how the action of individual neurons and synapses in various parts of the central nervous system contribute to the learning, memory or behaviour of an organism.
Levels of neurophysiological description

1) Brain  
   Network  
   Neuron

2) Artificial neural networks vs models of brain function
## Schedule 2017

<table>
<thead>
<tr>
<th>Day</th>
<th>Time</th>
<th>Subject</th>
<th>Lecturer</th>
<th>Venue</th>
</tr>
</thead>
<tbody>
<tr>
<td>13 Oct</td>
<td>10:00 – 11:00</td>
<td>Introduction to artificial neural networks &amp; unsupervised learning</td>
<td>Prof. Neil Burgess</td>
<td>Chadwick Building G07</td>
</tr>
<tr>
<td>18 Oct</td>
<td>11:00 – 12:00</td>
<td>Intro to artificial neural networks &amp; unsupervised learning, cont.</td>
<td>Prof. Neil Burgess</td>
<td>Medawar Building G02 Watson LT</td>
</tr>
<tr>
<td></td>
<td>12:00 – 13:00</td>
<td>Intro to artificial neural networks &amp; unsupervised learning, cont.</td>
<td>Prof. Neil Burgess</td>
<td>Medawar Building G02 Watson LT</td>
</tr>
<tr>
<td>20 Oct</td>
<td>10:00 – 11:00</td>
<td>Computational properties of individual neurons</td>
<td>Prof. David Attwell</td>
<td>Chadwick Building G07</td>
</tr>
<tr>
<td>25 Oct</td>
<td>11:00 – 12:00</td>
<td>Artificial neural networks, feedback &amp; simple supervised learning.</td>
<td>Prof. Neil Burgess</td>
<td>Medawar Building G02 Watson LT</td>
</tr>
<tr>
<td></td>
<td>12:00 – 13:00</td>
<td>More advanced learning algorithms in artificial neural networks.</td>
<td>Prof. Neil Burgess</td>
<td>Medawar Building G02 Watson LT</td>
</tr>
<tr>
<td>27 Oct</td>
<td>10:00 – 11:00</td>
<td>More advanced learning algorithms in artificial neural networks, cont.</td>
<td>Prof. Neil Burgess</td>
<td>Chadwick Building G07</td>
</tr>
<tr>
<td>1 Nov</td>
<td>11:00 – 12:00</td>
<td>The hippocampus and spatial representation.</td>
<td>Dr Andrej Bicanski</td>
<td>Medawar Building G02 Watson LT</td>
</tr>
<tr>
<td></td>
<td>12:00 – 13:00</td>
<td>Path integration, continuous attractors and grid cells.</td>
<td>Dr Daniel Bush</td>
<td>Medawar Building G02 Watson LT</td>
</tr>
<tr>
<td>3 Nov</td>
<td>10:00 – 11:00</td>
<td>Hippocampal and striatal navigation.</td>
<td>Dr Caswell Barry</td>
<td>Chadwick Building G07</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reading Week</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15 Nov</td>
<td>11:00 – 12:00</td>
<td>Hippocampus and associative memory</td>
<td>Dr Andrej Bicanski</td>
<td>Medawar Building G02 Watson LT</td>
</tr>
<tr>
<td></td>
<td>12:00 – 13:00</td>
<td>Hippocampus and associative memory</td>
<td>Dr Andrej Bicanski</td>
<td>Medawar Building G02 Watson LT</td>
</tr>
<tr>
<td>17 Nov</td>
<td>10:00 – 11:00</td>
<td>Spatial processing in the spine and motor cortex.</td>
<td>Dr Caswell Barry</td>
<td>Chadwick Building G07</td>
</tr>
<tr>
<td>22 Nov</td>
<td>11:00 – 12:00</td>
<td>Reinforcement learning.</td>
<td>Prof. Neil Burgess</td>
<td>Medawar Building G02 Watson LT</td>
</tr>
<tr>
<td></td>
<td>12:00 – 13:00</td>
<td>Reinforcement learning, cont.</td>
<td>Prof. Neil Burgess</td>
<td>Medawar Building G02 Watson LT</td>
</tr>
<tr>
<td>24 Nov</td>
<td>10:00 – 11:00</td>
<td>Models of prefrontal cortex.</td>
<td>Dr Sam Gilbert</td>
<td>Chadwick Building G07</td>
</tr>
<tr>
<td>29 Nov</td>
<td>11:00 – 12:00</td>
<td>Learning, performing and remembering serially ordered actions.</td>
<td>Dr Caswell Barry</td>
<td>Medawar Building G02 Watson LT</td>
</tr>
<tr>
<td></td>
<td>12:00 – 13:00</td>
<td>Neural bases of sensory decision making.</td>
<td>Prof Peter Latham</td>
<td></td>
</tr>
<tr>
<td>1 Dec</td>
<td>10:00 – 11:00</td>
<td>NO LECTURE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6 Dec</td>
<td>11:00 – 12:00</td>
<td>Filtering and normalization in sensory systems.</td>
<td>Prof. Matteo Carandini</td>
<td>Medawar Building G02 Watson LT</td>
</tr>
<tr>
<td></td>
<td>12:00 – 13:00</td>
<td>Theories of the cerebellum.</td>
<td>Dr Peter Gilbert</td>
<td></td>
</tr>
<tr>
<td>8 Dec</td>
<td>10:00 – 11:00</td>
<td>Temporal processing in audition and olfaction.</td>
<td>Dr Caswell Barry</td>
<td>Chadwick Building G07</td>
</tr>
<tr>
<td>13 Dec</td>
<td>11:00 – 12:00</td>
<td>Computing with spike timing and delays; course review.</td>
<td>Prof. Neil Burgess</td>
<td>TBA</td>
</tr>
</tbody>
</table>
General reading list

General: Fundamentals of Computational Neuroscience by Thomas Trappenberg (OUP, 2002)

Artificial Neural Networks:
1. An Introduction to Neural Networks, James A. Anderson (MIT Press, 1995);
2. An Introduction to Neural Networks, Kevin Gurney (UCL Press, 1997);

Biological neural networks:

Models of brain systems/systems neuroscience:

Computational Neuroscience (v. mathematical)
Introduction to artificial neural networks & unsupervised learning

AIMS

1. Understand simple mathematical models of how a neuron’s firing rate depends on the firing rates of the neurons with synaptic connections to it.
2. Describe how Hebbian learning rules relate change in synaptic weights to the firing rates of the pre- and post-synaptic neurons.
3. Describe how application of these rules can lead to self-organisation in artificial neural networks.
4. Relate self-organisation in artificial neural networks to organisation of the brain, such as in topographic maps.

READING

1. Books 1, 2, 8.
Modelling the function of a neuron: levels of description

- full compartmental models
- leaky integrate-and-fire models
- integrate-and-fire models
- standard artificial neuron
- threshold logic unit

Very detailed

Keep it simple: there’s going to be lots of them..
Compartmental models. Each compartment is simple:

\( V_m \): membrane potential inside the compartment (relative to "ground" outside cell).

\( C_m \): membrane capacitance - charged or discharged as ions flow in or out of the compartment (changing \( V_m \)) from adjacent compartments (\( V_m' \) and \( V_m'' \), axial resistances \( R_a \) and \( R_a' \) ) or through the membrane.

\( R_m \): leakage resistance & equilibrium potential \( E_m \) represent passive channels (rate and direction of current flow depends on size \( V_m \) versus \( E_m \)).

\( G_k \): variable conductances (1/resistance) specific to particular ions. Each has its own equilibrium potential \( E_k \) and may vary with \( V_m \) (active channels).
Compartmental models, cont.

Computer simulation of a Purkinje cell, color represents membrane potential.
Integrate-and-fire models

inputs current pulses: output: spikes

No physical structure:

temporal integration

membrane potential ($V$)

firing threshold ($T$)

resting potential ($V_r$)

Input current ($I$)

‘Leaky’ integrate-&-fire

time constant ($\tau$), due to passive current leakage ($\tau = CR$)

Leakage $\Rightarrow$ takes longer to reach firing threshold & inputs must arrive closer together in time to summate.

\[ C = \frac{q}{V}, \quad CV = q, \quad C \frac{dV}{dt} = I; \quad C \frac{dV}{dt} = (V_r - V) / R + I, \]

\[ V = IR, \quad I = V/R; \]

with spike if $V = T$
Standard artificial neuron

Inputs ($x_i$): firing rates
Connection weights ($w_i$): net synaptic ‘efficacy’

No physical structure, no simulation of spikes, connections can be +ve or -ve

The ‘net input’ to the neuron is:
\[ h = \sum_i w_i x_i = w_1 x_1 + w_2 x_2 + w_3 x_3, \]
aka. the ‘weighted sum’ of input activation.

The ‘transfer function’ \( f(h) \) relates the output \( o \) to the net input: \( o = f(h) = f(\sum_i w_i x_i) \)
and includes the firing threshold \( T \).

Common types of transfer function

- **Linear**, e.g.
  \[ f(h) = h - T \]

- **Threshold-linear**, e.g.
  \[ f(h) =\begin{cases} h - T & \text{if } h > T \\ 0 & \text{if } h < T \end{cases} \]

- **Threshold logic function**
  \[ f(h) =\begin{cases} 1 & \text{if } h > T \\ 0 & \text{if } h < T \end{cases} \]

- **Sigmoidal**
  \[ f(h) = \frac{1}{1 + \exp(-(h - T))} \]
Implications of using the ‘weighted sum’ of input activations as the ‘net input’ to the artificial neuron

If the total amounts of input activation & connection weight are limited*, the maximum net input $h$ (and thus output firing rate) occurs when the patterns of input activations and of connection weights match.

* e.g. $|\mathbf{w}| = 1$, $|\mathbf{x}| = 1$. 

\[ o = f(h), \quad h = w_1 x_1 + w_2 x_2 \]

The input activations and the connection weights can be thought of as vectors $\mathbf{x}$ and $\mathbf{w}$. The weighted sum $w_1 x_1 + w_2 x_2$ is also known as the ‘dot product’ ($\mathbf{w} \cdot \mathbf{x}$) of $\mathbf{x}$ and $\mathbf{w}$ and depends on the angle $\theta$ between them: $\mathbf{w} \cdot \mathbf{x} = |\mathbf{w}| \cdot |\mathbf{x}| \cdot \cos(\theta)$
The vector ‘dot product’

**Definition:** \( \mathbf{A} \cdot \mathbf{B} = |\mathbf{A}| \cdot |\mathbf{B}| \cdot \cos(\theta) \)

**So:**

\[
\mathbf{A} \cdot \mathbf{B} = |\mathbf{A}| \cdot |\mathbf{B}| \cdot \cos(\theta) \cdot \cos(\theta_B) + |\mathbf{A}| \cdot |\mathbf{B}| \cdot \sin(\theta) \cdot \sin(\theta_B)
\]

\[
\mathbf{A} \cdot \mathbf{B} = A_x B_x + A_y B_y
\]

More generally:

\[
\mathbf{A} \cdot \mathbf{B} = \sum_i A_i B_i
\]
Types of artificial neural networks

feed-forward

feed-back

recurrent

single-layer

multi-layer
Learning

The problem: find connection weights such that the network does something useful.

Solution:

Experience-dependent learning rules to modify connection weights, i.e. learn from examples.

1. ‘Unsupervised’ (no ‘teacher’ or feedback about right and wrong outputs)

2. ‘Supervised’:
   A. Evolution/genetic algorithms
   B. Occasional reward or punishment (‘reinforcement learning’)
   C. Fully-supervised: each example includes correct output.
Unsupervised learning

- The ‘Hebb rule’, often interpreted as: strengthen connections between neurons that tend to be active at the same time. (cf Hebb, 1949)

\[ w_{ij} \rightarrow w_{ij} + \Delta w_{ij} \]

\[ x_j \rightarrow x_j + \varepsilon x_j x_i \]

Cf. Long-term potentiation, long-term depression.

N.B. ANNs just model firing rate, so cannot implement more complex ‘spike-time dependent’ synaptic plasticity (Bi & Poo, J Neurosci., 1998)
Unsupervised learning, example 1: 

**Competitive learning**

‘Lateral inhibition’ => one ‘winner’ if strongly activated enough 

- Random initial connection weights
- Present $n^{th}$ input pattern $x^n$
- winner: output $o_k$ (i.e. $h_k > h_i$ for all $i \neq k$)
  set $o_k=1$, $o_i=0$ for all $i \neq k$
- Hebbian learning: $w_{ij} \rightarrow w_{ij} + \varepsilon o_i x_j^n$
  i.e. $w_{kj} \rightarrow w_{kj} + \varepsilon x_j^n$, other weights don’t change.
- Normalisation: reduce total size of connection weights
  to each output (so $|w_i| = 1$) by dividing each by $|w_i|
  $ or using alternative combined learning rule:
  $w_{kj} \rightarrow w_{kj} + \varepsilon (x_j^n - w_{kj})$
- Present next input pattern..

The output whose weights are most similar to $x^n$ wins and its weights then become more similar. Different outputs find their own clusters in input data.
The output whose weights $w$ are most similar to $x^n$ wins, and its weights then become more similar. Different outputs find their own clusters in input data.
Sharp's (1991) model of place cell firing

Inputs: Cue distances, directions

Competitive learning

35 SIMULTANEously recorded place cells
Competitive learning cont.

**Competitive learning** is built upon 3 ideas:

- **Hebbian learning principle:** when pre-synaptic and post-synaptic units are co-active, the connection between them should increase.
- **Competition between different units for activation**, through lateral inhibition / winner-take-all activation rule
- **Competition between incoming weights of a unit**, to prevent all weights from saturating, by normalizing the weights to have fixed net size: if some incoming weights to a unit grow, the others will shrink.

• Competitive learning performs **clustering** on the input patterns:
  - Each time a unit wins, it moves its weights closer to the current input pattern
  - A given unit will therefore be more likely to win the competition for similar inputs
  - Each unit’s weights thereby move toward the centre of a cluster of input patterns


• Example of competitive learning: Sharp’s model of place cell firing.
Topographic organisation of orientation selectivity in V1
Unsupervised learning, example 2:

**Feature maps & self-organisation**

Arrange output units in a sheet:

\[ \text{Willshaw and Von der Marlsburg's (1976) 'retinotopic map'} \]

Lateral connections (between outputs) vary with neurons' separation - excitatory nearby, inhibitory far apart ('mexican hat' function):

- Works like competitive learning, but not only 1 winner active: nearby units also active and so also learn to respond to similar input patterns.
- Produces a 2D map of the similarities present in a large set of input patterns.
Kohonen’s feature map (1982)

1 winner-takes-all as in competitive learning, but learning rule modified so that weights to outputs neighbouring the winner ($o_k$) are also modified using a ‘neighbourhood function’ $F$.

Present many input patterns, for each change weights according to:

$$w_{ij} \rightarrow w_{ij} + \varepsilon F(i,k)(x_j^n - w_{ij})$$

Causes nearby outputs to learn to represent (be active for) similar stimuli: producing a 2D map of complex (many D) data.

Cf. Learning in a volume, e.g. caused by the physical spread of chemical neurotransmitters or messengers, does not need (implausible?) lateral connections used by Willshaw & Von der Marlsburg.
Kohonen’s feature map (1982), cont.

The structure of a map of 2-D data, and how it changes with learning, can be seen by showing each output unit in the part of input space that it represents:
Kohonen’s feature map (1982), cont.

The structure of a map of high-dimensional data can be seen by labelling what each output unit represents:

The input for each animal is a long binary vector of its attributes (e.g. 2-feet, 4-feet, can swim, can fly, has feathers, eats meat etc etc).

Figure 2. Visualization of a 10 × 10 feature map for a set of pattern vectors describing binary features of 16 animal species. The spatial arrangement of the labeled map regions reflects the similarity relationships between the animals.
SUMMARY: Introduction to Artificial Neural Networks, Unsupervised learning

1. An artificial neuron (v. simple model of a real neuron, McCulloch and Pitts, Rosenblatt..).
   Input values: $x_i$, connection weights: $w_i$, ‘weighted sum’ of inputs $\sum_i w_i x_i$, threshold $T$, output $o$; ‘transfer function’ $f(input)$.

2. Learning: How to find a useful set of connections $w_{ij}$:
   The Hebb rule and LTP: connection weight $w_{ij}$ between neurons with activation $x_i$ and $x_j$ changes as $w_{ij} \rightarrow w_{ij} + \varepsilon x_i x_j$.

3. Unsupervised learning/ self-organisation in ‘feed-forward’ neural networks (NNs). Training set of input activations $x^k$; each causes output activations $o^k$, and connection weights between active units are strengthened.
   (a) Competitive Learning (Rumelhart and Zipser, 1986). Lateral inhibition/ winner-take-all dynamics. Weight normalisation. Feature extraction: data clustering; Sharp’s (1991) model of place field formation.
   (b) Feature Maps. ‘Mexican hat’ lateral connections, Willshaw and Von der Marlsburg’s retinotopic map. Kohnonen’s ‘feature map’: learning in a local volume (cf chemical diffusion?).
Unsupervised learning, example 3:
Hopfield’s (1982) associative memory network

- Fully connected recurrent network (no input)
- Symmetric connection weights ($w_{ij} = w_{ji}$)
- Units are active ($S_i = 1$) or inactive ($S_i = -1$)

**Learning:** impose pattern of activation, use ‘Hebbian’ rule to change weights

$$W_{ij} \rightarrow W_{ij} + \varepsilon S_i S_j$$

**Recall:** start from similar pattern of activation, change activation according to sign of input to recover original pattern

$$s_i = \text{sign}(\sum_j w_{ij} s_j)$$

**Activation:**

$$s_i = f(h_i), \text{ where } h_i = \sum_j w_{ij} s_j$$

**Learning:**

$$\Delta W_{ij}$$

<table>
<thead>
<tr>
<th>$S_i$</th>
<th>-1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>↑</td>
<td>↓</td>
</tr>
<tr>
<td>1</td>
<td>↓</td>
<td>↑</td>
</tr>
</tbody>
</table>

$h = T = 0$
Hopfield networks, cont.

Patterns of activation are learned as ‘stable states’ under the rule for updating activations, e.g.

\[ \Delta w_{ij} = s_i \cdot s_j \]

Several different patterns can be learned in the same network, but the memory capacity is limited to about 0.14N.

Memory is ‘content addressable’: performing ‘pattern completion’ of partial cue. Spurious memories (combinations of real ones) are also formed.

More plausible learning rules show similar behaviour.
Hopfield networks: attractors & stable states

To support a pattern of activation connections should be positive between units in the same state (i.e. 11 or -1-1) and negative between units in different states (1-1 or -11), i.e. $s_i s_j w_{ij} > 0$

The ‘frustration’ or ‘energy’ of the system is how much this is not true, i.e. $E = -\sum_{ij} s_i s_j w_{ij}$

The update rule changes each units activation to reduce the overall frustration, until the network ends up in a stable state from which it cannot be reduced further.

The learning rule sets the weights so that to-be-remembered patterns of activity are stable states (aka ‘attractor states’).
Examples of Hopfield networks

A 5x9 network storing 8 patterns

Retrieval in a 130x180 network

Figure 6.3 The training set for the Hopfield network.
• **Activation rule:**
  – If net input is greater than zero, unit gets an activation of 1; otherwise activation is -1.
  – Random, asynchronous update of activations

• **Architecture:**
  Symmetrically connected recurrent network.

• **Hebbian learning:**
  For each training pattern,
  – Set states of units to corresponding elements of pattern.
  – Increment each weight in proportion to product of pre- and post-synaptic states.

• **Desirable features:**
  – Attractor dynamics: guaranteed convergence to an attractor state.
  – Pattern completion

• **Undesirable features:**
  – Spurious attractors
  – Limited storage capacity