Connectionistic Networks and learning

ME60033 Intelligent Machines and Systems

C.S.Kumar

2014

What are Neural Networks?

- Models of the brain and nervous system
- Highly parallel
  - Process information much more like the brain than a serial computer
- Learning

- Very simple principles
- Very complex behaviours

- Applications
  - As powerful problem solvers
  - As biological models

Biological inspiration

Animals are able to react adaptively to changes in their external and internal environment, and they use their nervous system to perform these behaviours.

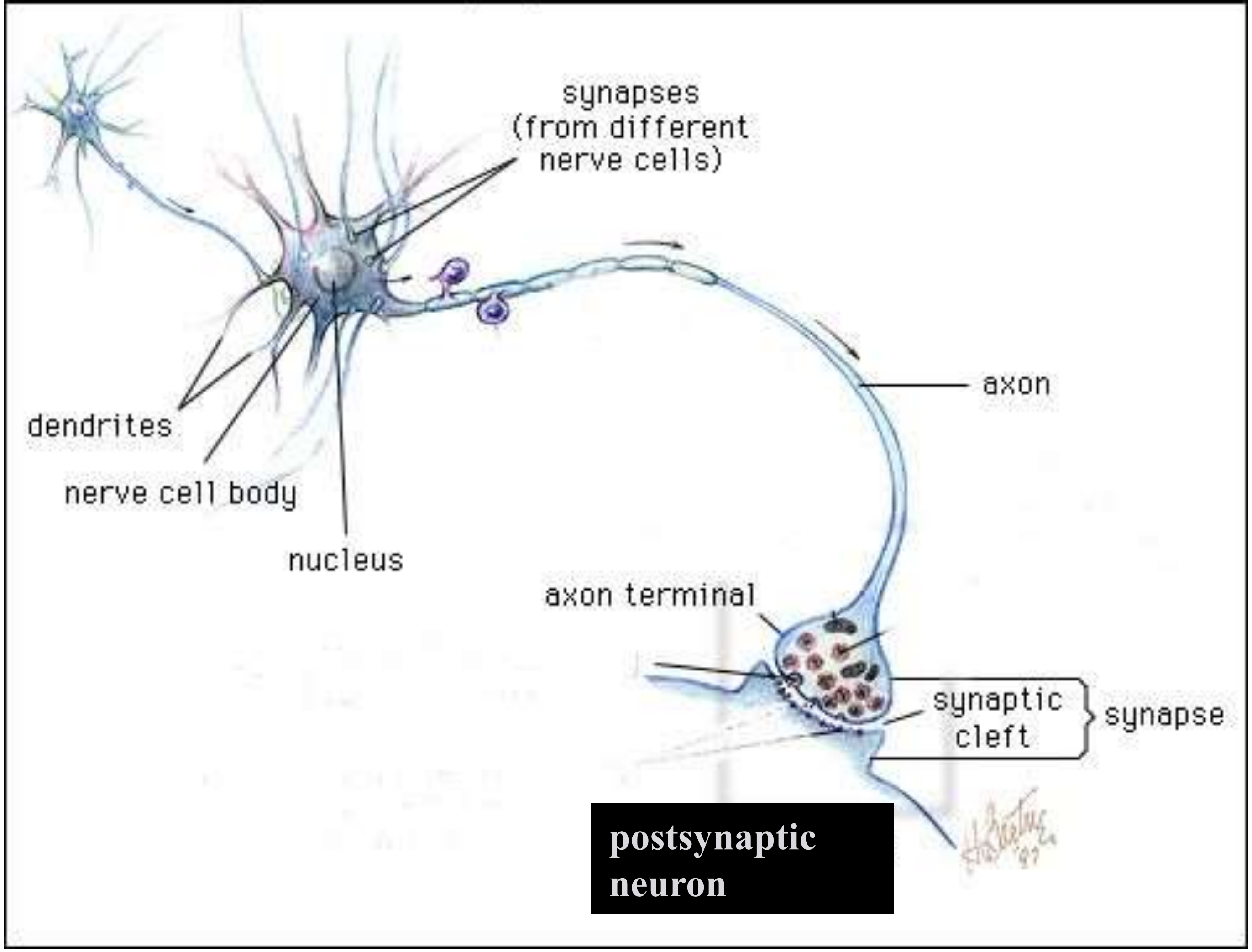
An appropriate model/simulation of the nervous system should be able to produce similar responses and behaviours in artificial systems.

The nervous system is build by relatively simple units, the neurons, so copying their behavior and functionality should be the solution.

Neurons

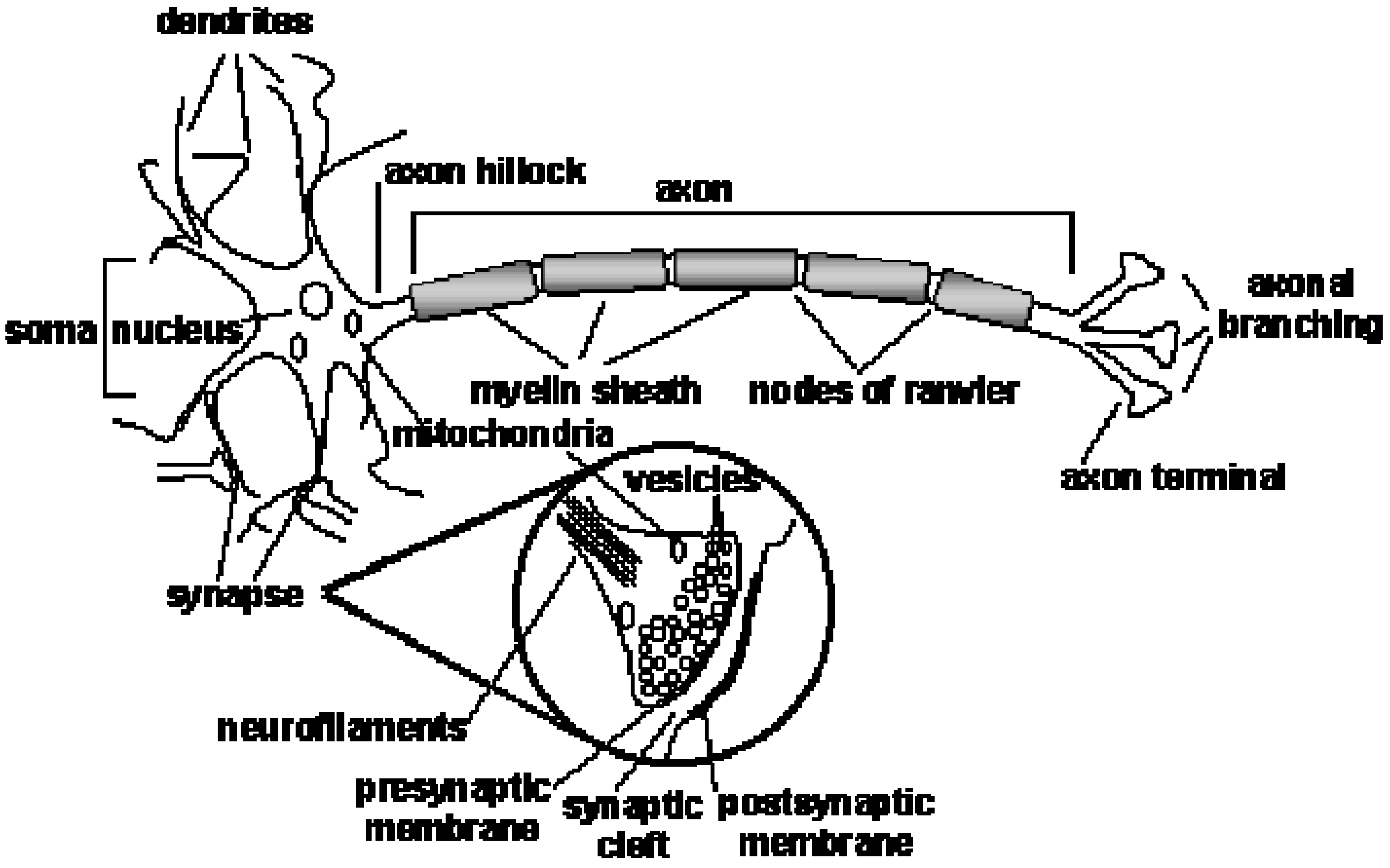
- ***cell body***
- ***dendrites (input structure)***
  - *receive inputs from other neurons*
  - *perform spatio-temporal integration of inputs*
  - *relay them to the cell body*
- ***axon (output structure)***
  - *a fiber that carries messages (spikes) from the cell to dendrites of other neurons*

Neuron Forming a Chemical Synapse



science-education.nih.gov

Schematic Diagram of a Biological Neuron

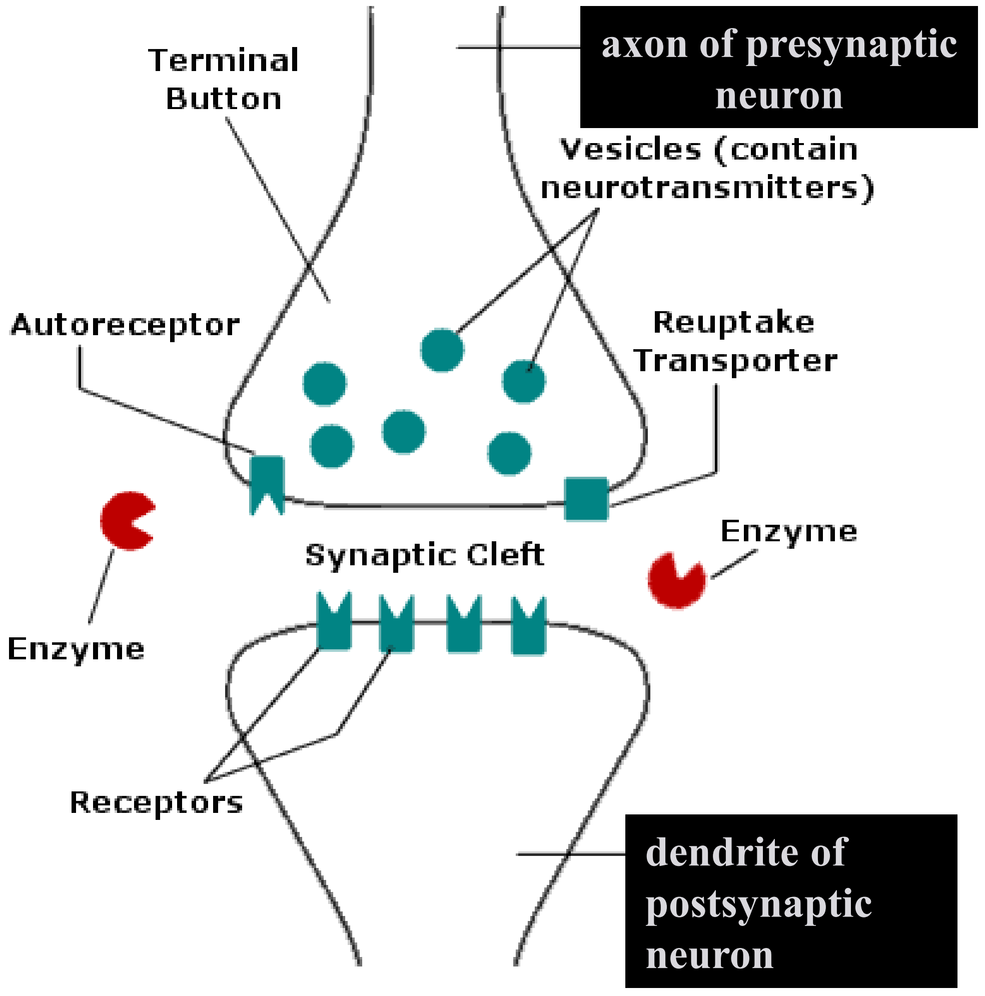


Synapse

- ***site of communication between two cells***

- ***formed when an axon of a*** <u>presynaptic</u> ***cell "connects" with the dendrites of a*** <u>postsynaptic</u> ***cell***

Synapse



Terminal Button
axon of presynaptic neuron
Vesicles (contain neurotransmitters)
Autoreceptor
Reuptake Transporter
Enzyme
Synaptic Cleft
Enzyme
Receptors
dendrite of postsynaptic neuron

bipolar.about.com/library

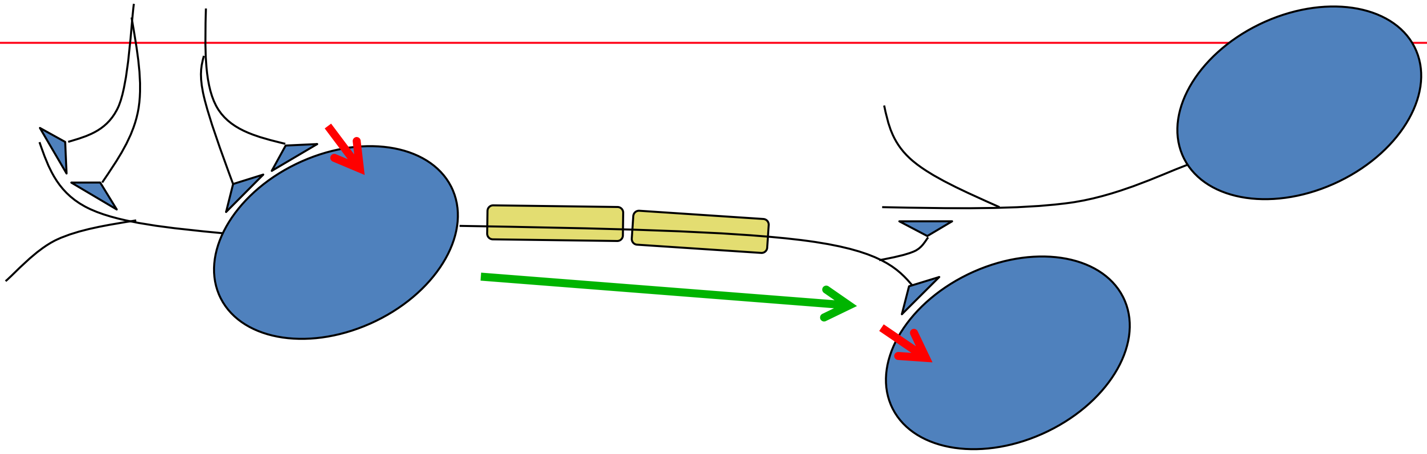
Synapse

- *a synapse can be <u>excitatory</u> or <u>inhibitory</u>*
- *arrival of activity at an excitatory synapse <u>depolarizes</u> the local membrane potential of the postsynaptic cell and makes the cell <u>more</u> prone to firing*
- *arrival of activity at an inhibitory synapse <u>hyperpolarizes</u> the local membrane potential of the postsynaptic cell and makes it <u>less</u> prone to firing*
- *the greater the <u>synaptic strength</u>, the greater the depolarization or hyperpolarization*

Transmission of information

Information must be transmitted

- within each neuron
- and between neurons



Biological inspiration

The spikes travelling along the axon of the pre-synaptic neuron trigger the release of neurotransmitter substances at the synapse.

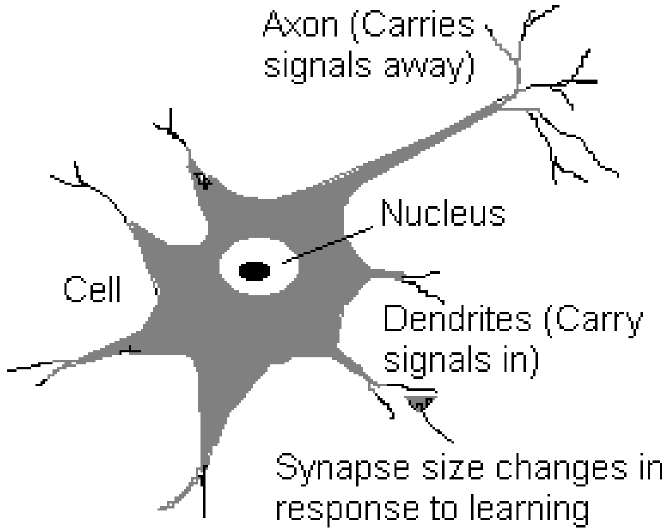
The neurotransmitters cause excitation or inhibition in the dendrite of the post-synaptic neuron.

The integration of the excitatory and inhibitory signals may produce spikes in the post-synaptic neuron.

The contribution of the signals depends on the strength of the synaptic connection.

Inspiration from Neurobiology

- A neuron: many-inputs / one-output unit
- output can be *excited* or *not excited*
- incoming signals from other neurons determine if the neuron shall excite ("fire")
- Output subject to attenuation in the *synapses,* which are junction parts of the neuron



Axon (Carries signals away)
Nucleus
Cell
Dendrites (Carry signals in)
Synapse size changes in response to learning

Synapse concept

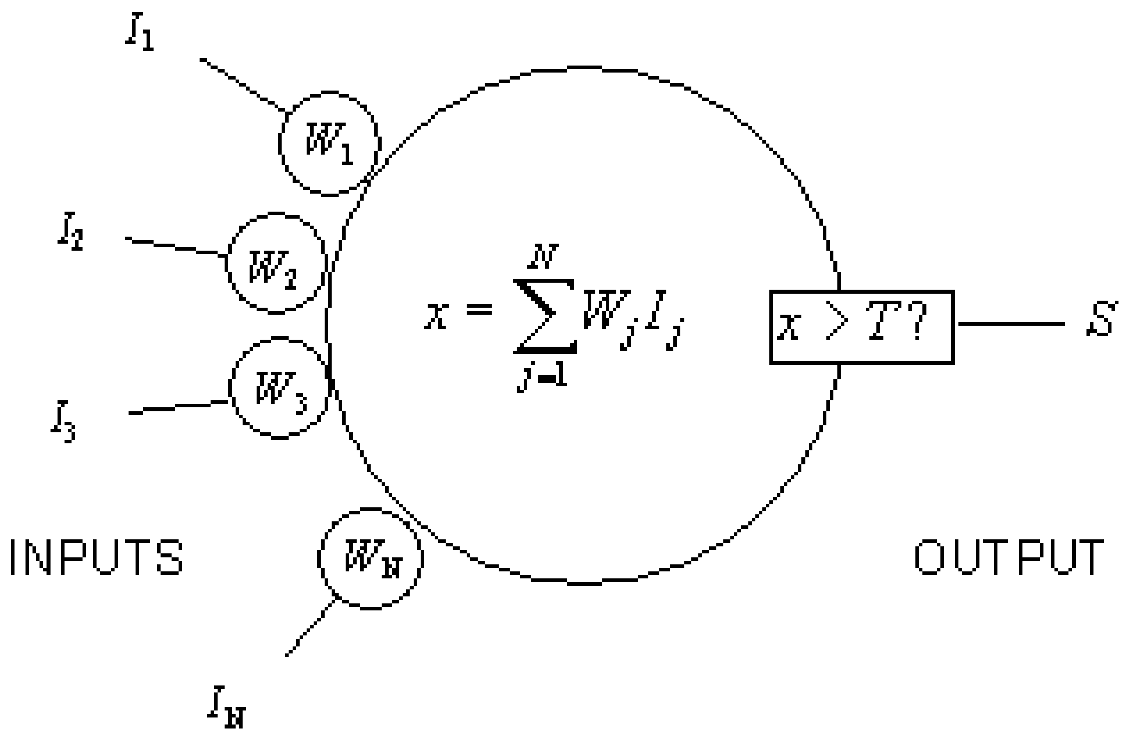
- The synapse resistance to the incoming signal can be changed during a "learning" process [1949]

Hebb’s Rule:

If an input of a neuron is repeatedly and persistently causing the neuron to fire, a metabolic change happens in the synapse of that particular input to reduce its resistance

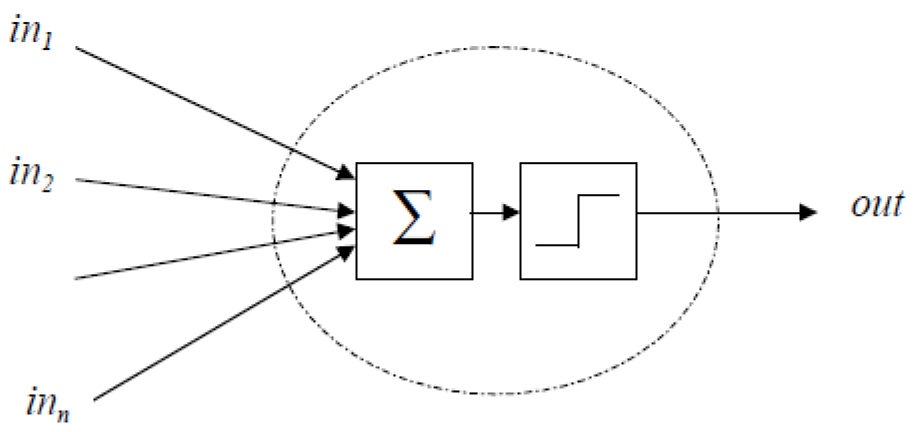
Mathematical representation

The neuron calculates a weighted sum of inputs and compares it to a threshold. If the sum is higher than the threshold, the output is set to 1, otherwise to -1.



The McCulloch-Pitts Neuron

This vastly simplified model of real neurons is also known as a ***Threshold Logic Unit*** :



1. A set of synapses (i.e. connections) brings in activations from other neurons.
2. A processing unit sums the inputs, and then applies a non-linear activation function (i.e. squashing/transfer/threshold function).
3. An output line transmits the result to other neurons.

Artificial neurons

The McCullogh-Pitts model:

- spikes are interpreted as spike rates;
- synaptic strength are translated as synaptic weights;
- excitation means positive product between the incoming spike rate and the corresponding synaptic weight;
- inhibition means negative product between the incoming spike rate and the corresponding synaptic weight;

Artificial neurons

Nonlinear generalization of the McCullogh-Pitts neuron:

$$y = f(x, w)$$

y is the neuron’s output, x is the vector of inputs, and w is the vector of synaptic weights.

Examples:

$$y = \frac{1}{1 + e^{-w^T x - a}}$$ sigmoidal neuron

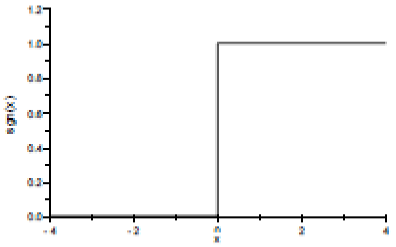
$$y = e^{-\frac{||x-w||^2}{2a^2}}$$ Gaussian neuron

Some Useful Functions

A function $y = f(x)$ describes a relationship (input-output mapping) from x to y.

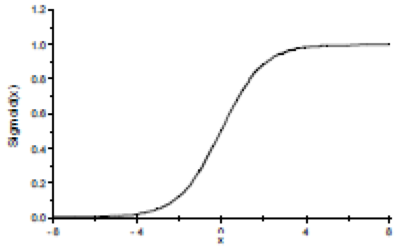
Example 1 The threshold or sign function $\operatorname{sgn}(x)$ is defined as

$$\operatorname{sgn}(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}$$



Example 2 The logistic or sigmoid function $\text{Sigmoid}(x)$ is defined as

$$\text{Sigmoid}(x) = \frac{1}{1 + e^{-x}}$$



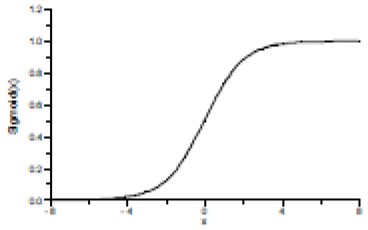
This is a smoothed (differentiable) form of the threshold function.

Other Types of Activation/Transfer Function

Sigmoid Functions These are smooth (differentiable) and monotonically increasing.

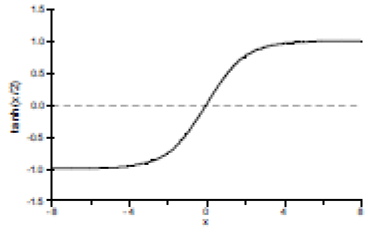
The logistic function

$$\text{Sigmoid}(x) = \frac{1}{1+e^{-x}}$$



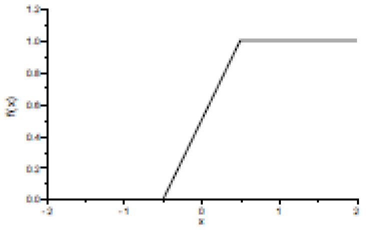
Hyperbolic tangent

$$\tanh\left(\tfrac{x}{2}\right) = \frac{1-e^{-x}}{1+e^{-x}}$$



Piecewise-Linear Functions Approximations of a sigmoid functions.

$$f(x) = \begin{cases} 1 & \text{if } x \geq 0.5 \\ x+0.5 & \text{if } -0.5 \leq x \leq 0.5 \\ 0 & \text{if } x \leq 0.5 \end{cases}$$



The McCulloch-Pitts Neuron Equation

Using the above notation, we can now write down a simple equation for the ***output*** *out* of a McCulloch-Pitts neuron as a function of its n ***inputs*** in_i :

$$out = \mathrm{sgn}(\sum_{i=1}^{n} in_i - \theta)$$

where θ is the neuron's activation ***threshold***. We can easily see that:

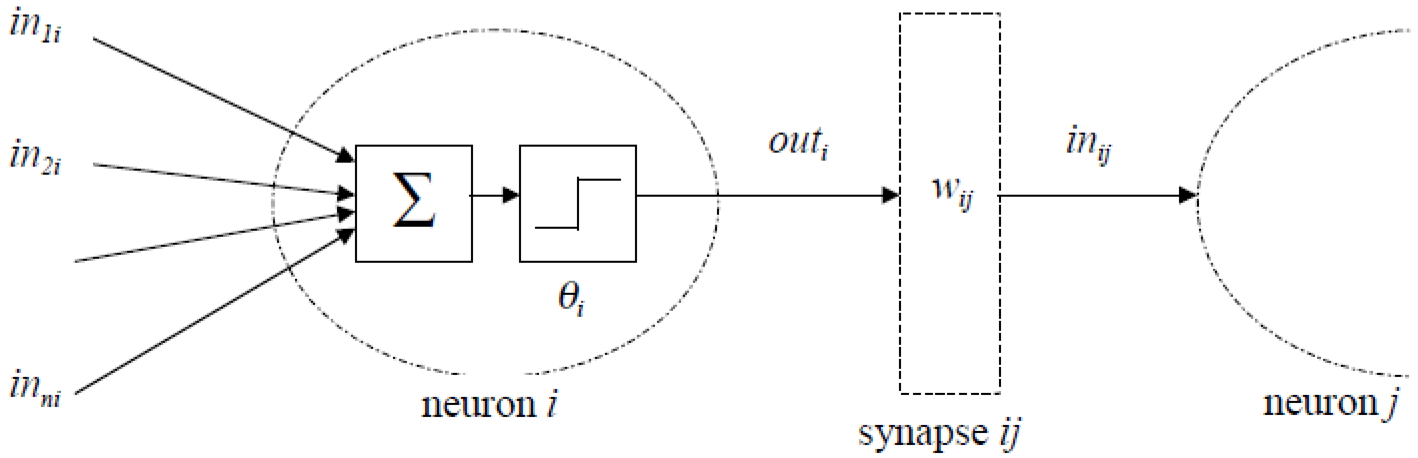
$$out = 1 \quad \text{if} \quad \sum_{k=1}^{n} in_k \geq \theta \qquad\qquad out = 0 \quad \text{if} \quad \sum_{k=1}^{n} in_k < \theta$$

Note that the McCulloch-Pitts neuron is an extremely simplified model of real biological neurons. Some of its missing features include: non-binary inputs and outputs, non-linear summation, smooth thresholding, stochasticity, and temporal information processing.

Nevertheless, McCulloch-Pitts neurons are computationally very powerful. One can show that assemblies of such neurons are capable of universal computation.

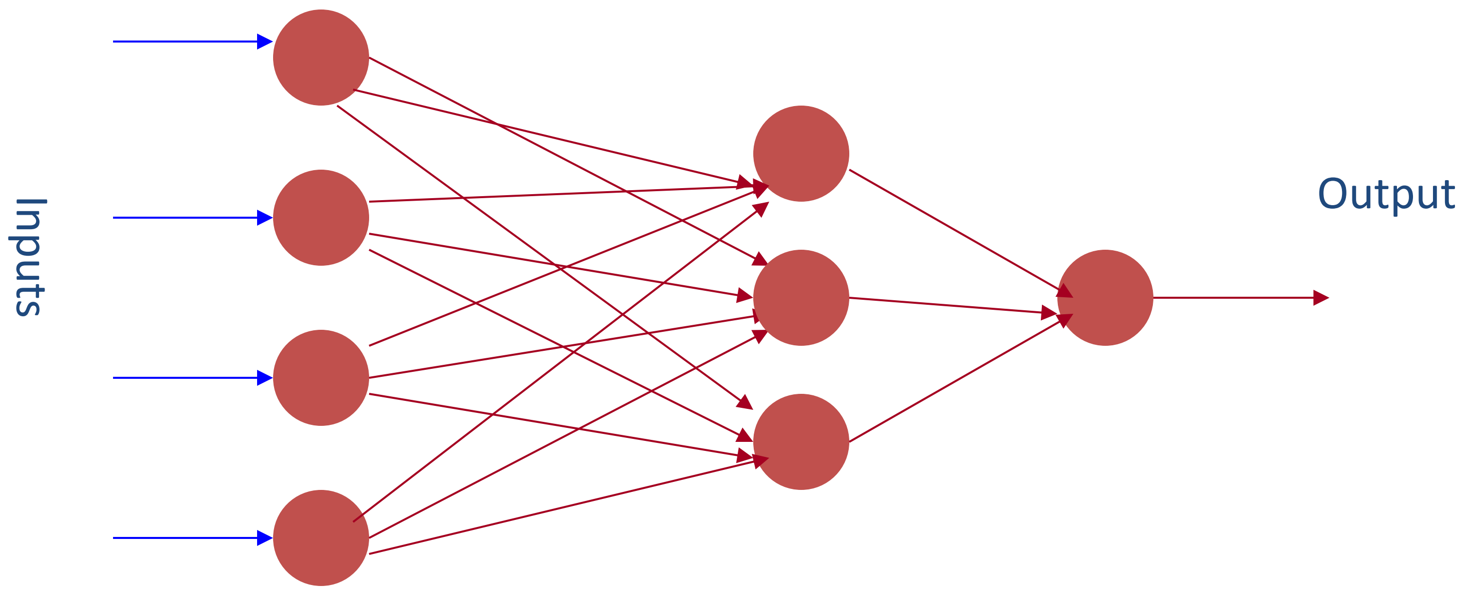
Networks of McCulloch-Pitts Neurons

One neuron can't do much on its own. Usually we will have many neurons labelled by indices *k*, *i*, *j* and activation flows between them via synapses with strengths w_{ki}, w_{ij}:



$in_{ki} = out_k w_{ki}$ $\qquad$ $out_i = \mathrm{sgn}(\sum_{k=1}^{n} in_{ki} - \theta_i)$ $\qquad$ $in_{ij} = out_i w_{ij}$

Artificial neural networks



An artificial neural network is composed of many artificial neurons that are linked together according to a specific network architecture. The objective of the neural network is to transform the inputs into meaningful outputs.

Artificial neural networks

Tasks to be solved by artificial neural networks:

- controlling the movements of a robot based on self-perception and other information (e.g., visual information);
- deciding the category of potential food items (e.g., edible or non-edible) in an artificial world;
- recognizing a visual object (e.g., a familiar face);
- predicting where a moving object goes, when a robot wants to catch it.

Evolving networks

- *Continuous process of:*
  - Evaluate output
  - Adapt weights
  - Take new inputs

"Learning"

- ANN evolving causes stable state of the weights, but neurons **continue working**: network has 'learned' dealing with the problem

Learning performance

- Network architecture
- Learning method:
  - Unsupervised
  - Reinforcement learning
  - Backpropagation

Unsupervised learning

- No help from the outside
- No training data, no information available on the desired output
- *Learning by doing*
- Used to pick out structure in the input:
    - Clustering
    - Reduction of dimensionality → compression
- *Example: Kohonen's Learning Law*

Competitive learning: example

- Example: Kohonen network

Winner takes all

only update weights of winning neuron

- Network topology
- Training patterns
- Activation rule
- Neighbourhood
- Learning

Reinforcement learning

- Teacher: training data
- The teacher scores the performance of the training examples
- Use performance score to shuffle weights ‘randomly’
- Relatively slow learning due to ‘randomness’

Back propagation

- Desired output of the training examples
- Error = difference between actual & desired output
- Change weight relative to error size
- Calculate output layer error , then propagate back to previous layer
- Improved performance, very common!

Generalization vs. specialization

- Optimal number of hidden neurons
  - Too many hidden neurons : you get an over fit, training set is memorized, thus making the network useless on new data sets
  - Not enough hidden neurons:
    network is unable to learn problem concept

    ~ conceptually: the network's language isn't able to express the problem solution

Generalization vs. specialization

- Overtraining:
  - Too much examples, the ANN memorizes the examples instead of the general idea
- Generalization vs. specialization trade-off:

# hidden nodes & training samples

Learning in biological systems

Learning = learning by adaptation

The young animal learns that the green fruits are sour, while the yellowish/reddish ones are sweet. The learning happens by adapting the fruit picking behavior.

At the neural level the learning happens by changing of the synaptic strengths, eliminating some synapses, and building new ones.

Learning in biological neural networks

The learning rules of Hebb:

- synchronous activation increases the synaptic strength;
- asynchronous activation decreases the synaptic strength.

These rules fit with energy minimization principles.

Maintaining synaptic strength needs energy, it should be maintained at those places where it is needed, and it shouldn’t be maintained at places where it’s not needed.

Learning principle for artificial neural networks

ENERGY MINIMIZATION

We need an appropriate definition of energy for artificial neural networks, and having that we can use mathematical optimisation techniques to find how to change the weights of the synaptic connections between neurons.

ENERGY = measure of task performance error

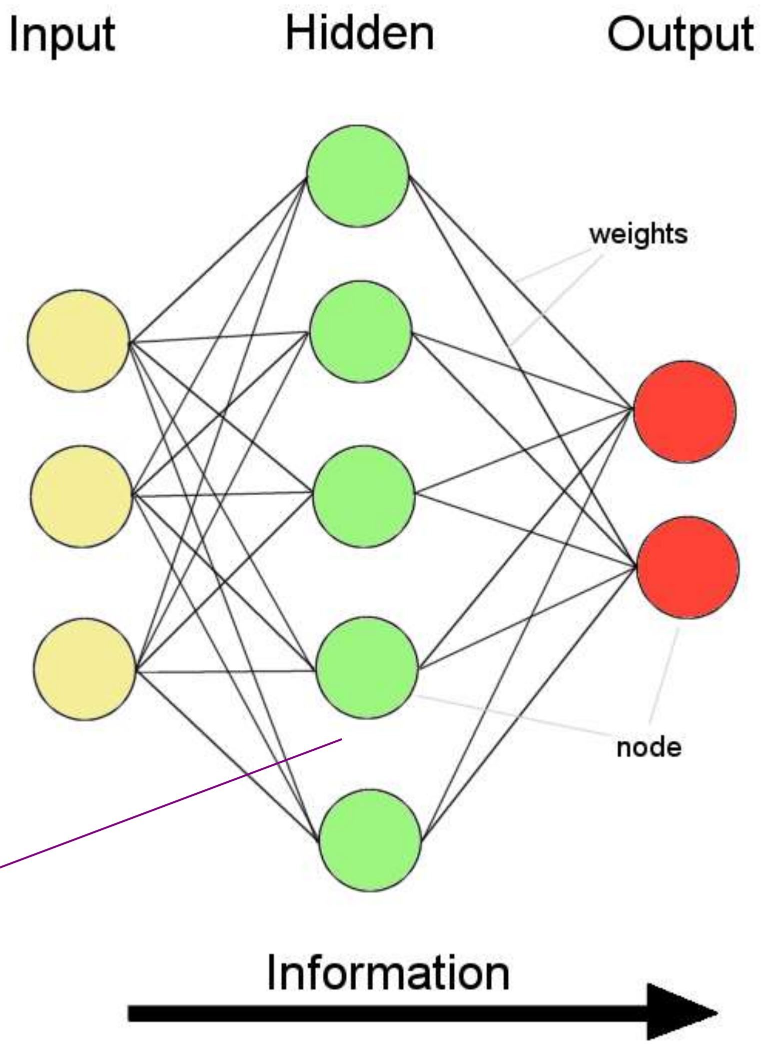
Feed-forward nets

Information flow is unidirectional

- Data is presented to *Input layer*
- Passed on to *Hidden Layer*
- Passed on to *Output layer*

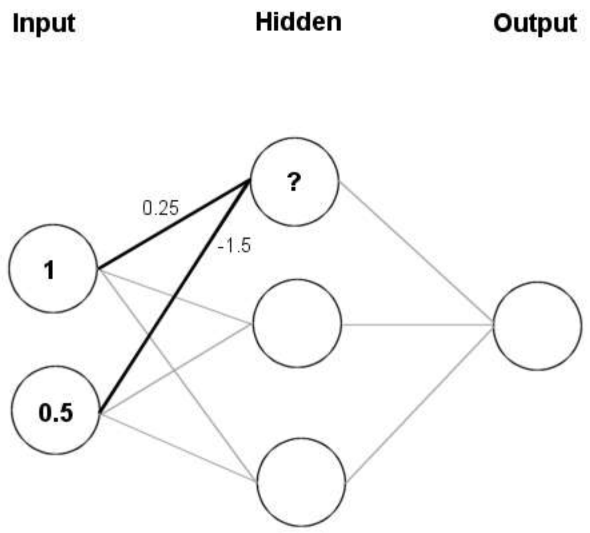
Information is distributed

Information processing is parallel



Internal representation (interpretation) of data

- Feeding data through the net:



(1 × 0.25) + (0.5 × (-1.5)) = 0.25 + (-0.75) = -
0.5

Squashing: $$\frac{1}{1+e^{0.5}} = 0.3775$$

- Data is presented to the network in the form of activations in the input layer

- Examples
  - Pixel intensity (for pictures)
  - Molecule concentrations (for artificial nose)
  - Share prices (for stock market prediction)

- Data usually requires preprocessing
  - Analogous to senses in biology

- How to represent more abstract data, e.g. a name?
  - Choose a pattern, e.g.
    - 0-0-1 for “Chris”
    - 0-1-0 for “Becky”

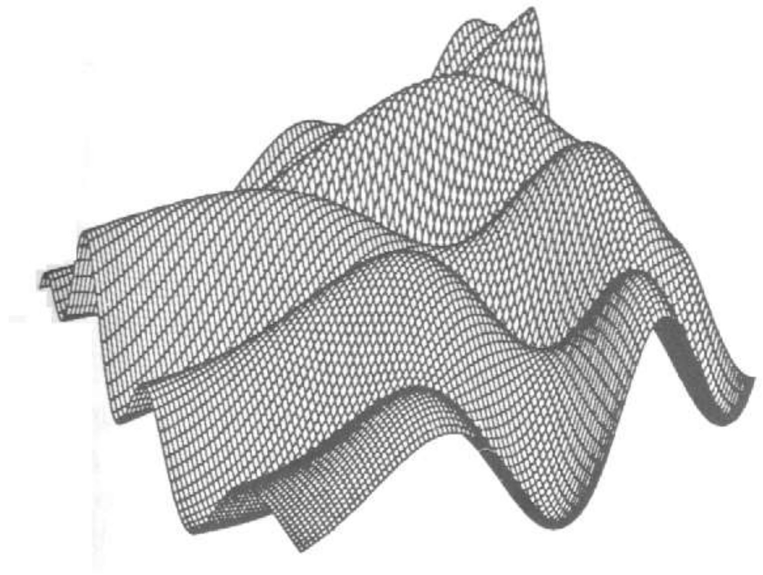
- Weight settings determine the behaviour of a network

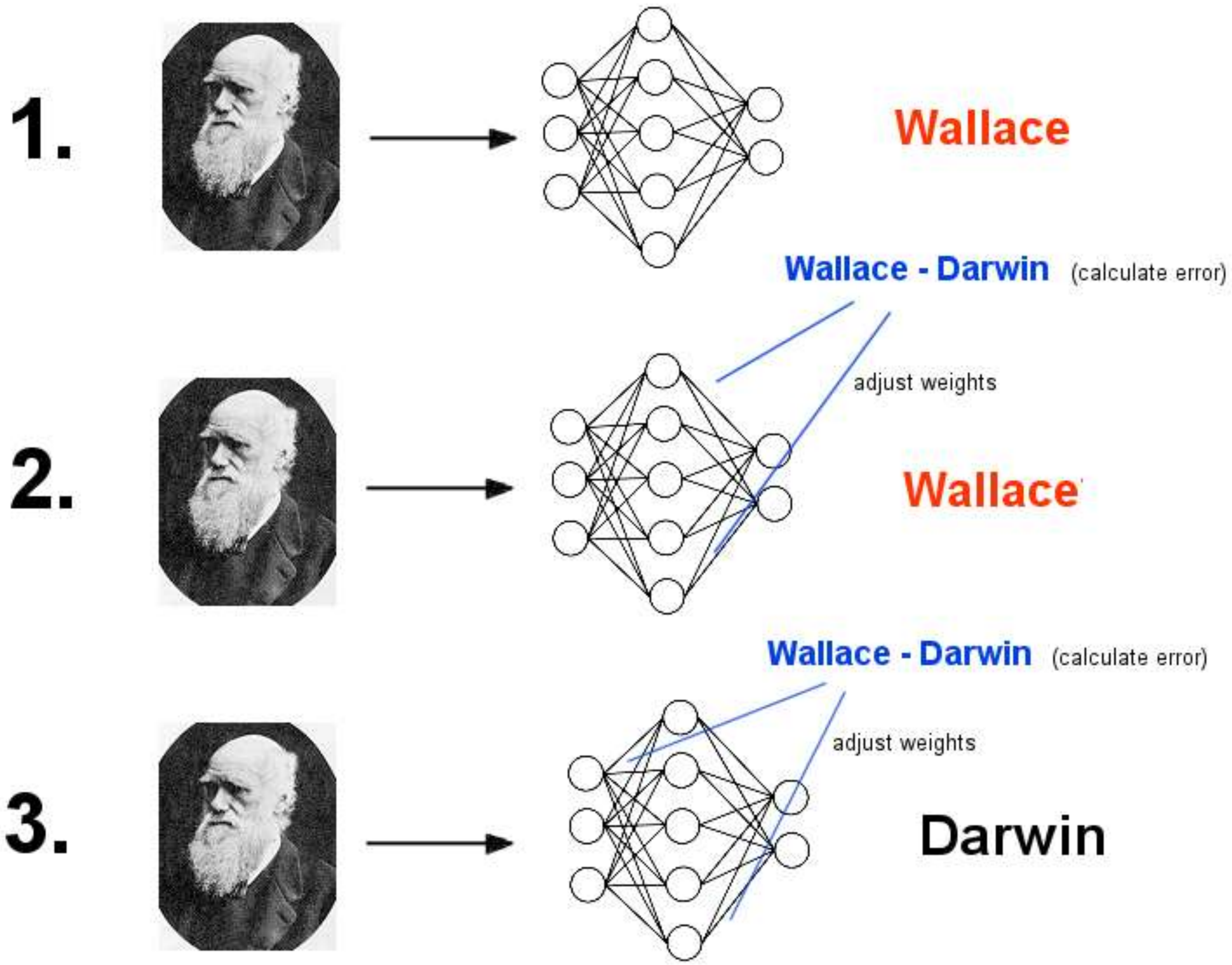
  → How can we find the right weights?

Training the Network - Learning

- Backpropagation
  - Requires training set (input / output pairs)
  - Starts with small random weights
  - Error is used to adjust weights (supervised learning)

  → Gradient descent on error landscape



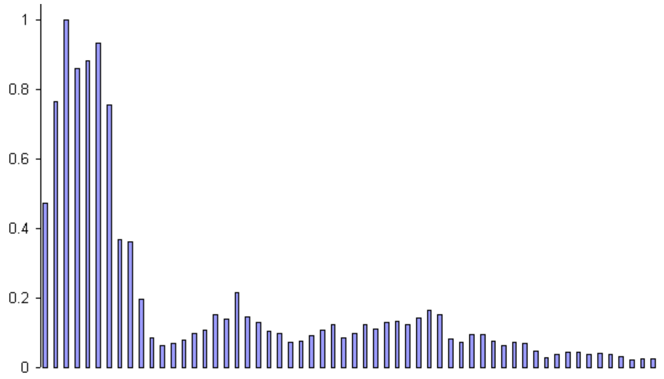

1.
Wallace
Wallace - Darwin (calculate error)
adjust weights
2.
Wallace
Wallace - Darwin (calculate error)
adjust weights
3.
Darwin

- **Advantages**
  - It works!
  - Relatively fast
- **Downsides**
  - Requires a training set
  - Can be slow
  - Probably not biologically realistic
- **Alternatives to Backpropagation**
  - Hebbian learning
    - Not successful in feed-forward nets
  - Reinforcement learning
    - Only limited success
  - Artificial evolution
    - More general, but can be even slower than backprop

Example: Voice Recognition

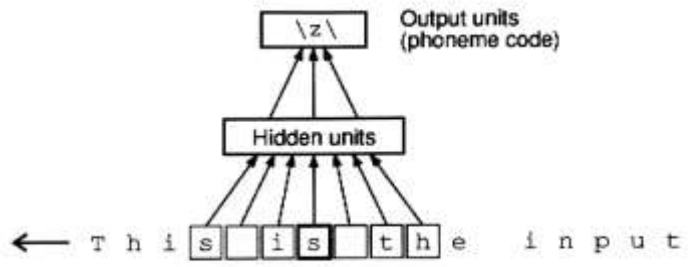
- Task: Learn to discriminate between two different voices saying “Hello”

- Data
  - Sources
    - Steve Simpson
    - David Raubenheimer
  - Format
    - Frequency distribution (60 bins)
    - Analogy: cochlea



Applications of Feed-forward nets

- Pattern recognition
  - Character recognition
  - Face Recognition
- Sonar mine/rock recognition (Gorman & Sejnowksi, 1988)
- Navigation of a car (Pomerleau, 1989)
- Stock-market prediction
- Pronunciation (NETtalk)
  (Sejnowksi & Rosenberg, 1987)



Various Architectures

- Multi Layer Perceptrons (MLP)
- Radial Basis Functions (RBF)

Neural network tasks

- control
- classification
- prediction
- approximation

These can be reformulated in general as

FUNCTION APPROXIMATION

tasks.

Approximation: given a set of values of a function g(x) build a neural network that approximates the g(x) values for any input x.

Neural network approximation

Task specification:

Data: set of value pairs: (x^t, y_t), $y_t = g(x^t) + z_t$; z_t is random measurement noise.

Objective: find a neural network that represents the input / output transformation (a function) $F(x,W)$ such that

$F(x,W)$ approximates $g(x)$ for every x

Learning to approximate

Error measure:

$$E = \frac{1}{N}\sum_{t=1}^{N}(F(x_t;W) - y_t)^2$$

Rule for changing the synaptic weights:

$$\Delta w_i^j = -c \cdot \frac{\partial E}{\partial w_i^j}(W)$$

$$w_i^{j,new} = w_i^j + \Delta w_i^j$$

c is the learning parameter (usually a constant)

Learning with a perceptron

Perceptron: $y_{out} = w^T x$

Data: $(x^1, y_1), (x^2, y_2), \ldots, (x^N, y_N)$

Error: $E(t) = (y(t)_{out} - y_t)^2 = (w(t)^T x^t - y_t)^2$

Learning:

$$w_i(t+1) = w_i(t) - c \cdot \frac{\partial E(t)}{\partial w_i} = w_i(t) - c \cdot \frac{\partial (w(t)^T x^t - y_t)^2}{\partial w_i}$$

$$w_i(t+1) = w_i(t) - c \cdot (w(t)^T x^t - y_t) \cdot x_i^t$$

$$w(t)^T x = \sum_{j=1}^{m} w_j(t) \cdot x_j^t$$

A perceptron is able to learn a linear function.

Learning with RBF neural networks

RBF neural network: $$y_{out} = F(x,W) = \sum_{k=1}^{M} w_k^2 \cdot e^{-\frac{\|x-w^{1,k}\|^2}{2(a_k)^2}}$$

Data: $$(x^1, y_1),(x^2, y_2),\ldots,(x^N, y_N)$$

Error: $$E(t) = (y(t)_{out} - y_t)^2 = (\sum_{k=1}^{M} w_k^2(t) \cdot e^{-\frac{\|x^t - w^{1,k}\|^2}{2(a_k)^2}} - y_t)^2$$

Learning:
$$w_i^2(t+1) = w_i^2(t) - c \cdot \frac{\partial E(t)}{\partial w_i^2}$$

$$\frac{\partial E(t)}{\partial w_i^2} = 2 \cdot (F(x^t, W(t)) - y_t) \cdot e^{-\frac{\|x^t - w^{1,i}\|^2}{2(a_i)^2}}$$

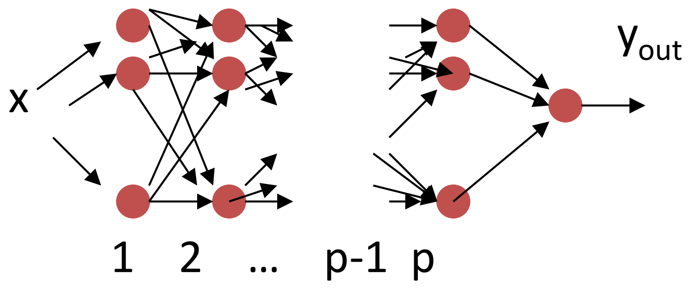
Only the synaptic weights of the output neuron are modified.

An RBF neural network learns a nonlinear function.

Learning with MLP neural networks

MLP neural network:

with p layers



$$y_k^1 = \frac{1}{1+e^{-w^{1kT}x-a_k^1}},\ k=1,\dots,M_1$$

$$y^1 = (y_1^1,\dots,y_{M_1}^1)^T$$

$$y_k^2 = \frac{1}{1+e^{-w^{2kT}y^1-a_k^2}},\ k=1,\dots,M_2$$

$$y^2 = (y_1^2,\dots,y_{M_2}^2)^T$$

$$\dots$$

$$y_{out} = F(x;W) = w^{pT}y^{p-1}$$

Data: $(x^1,y_1),(x^2,y_2),\dots,(x^N,y_N)$

Error: $E(t) = (y(t)_{out} - y_t)^2 = (F(x^t;W) - y_t)^2$

It is very complicated to calculate the weight changes.

Learning with backpropagation

Solution of the complicated learning:

- calculate first the changes for the synaptic weights of the output neuron;
- calculate the changes backward starting from layer p-1, and propagate backward the local error terms.

The method is still relatively complicated but it is much simpler than the original optimisation problem.

Learning as optimisation

The objective of adapting the responses on the basis of the information received from the environment is to achieve a better state. E.g., the animal likes to eat many energy rich, juicy fruits that make its stomach full, and makes it feel happy.

In other words, the objective of learning in biological organisms is to optimise the amount of available resources, happiness, or in general to achieve a closer to optimal state.

Learning with general optimisation

In general it is enough to have a single layer of nonlinear neurons in a neural network in order to learn to approximate a nonlinear function.

In such case general optimisation may be applied without too much difficulty.

Example: an MLP neural network with a single hidden layer:

$$y_{out} = F(x;W) = \sum_{k=1}^{M} w_k^2 \cdot \frac{1}{1+e^{-w^{1,kT}x-a_k}}$$

Learning with general optimisation

Synaptic weight change rules for the output neuron:

$$w_i^2(t+1) = w_i^2(t) - c \cdot \frac{\partial E(t)}{\partial w_i^2}$$

$$\frac{\partial E(t)}{\partial w_i^2} = 2 \cdot (F(x^t, W(t)) - y_t) \cdot \frac{1}{1 + e^{-w^{1,iT} x^t - a_i}}$$

Synaptic weight change rules for the neurons of the hidden layer:

$$w_j^{1,i}(t+1) = w_j^{1,i}(t) - c \cdot \frac{\partial E(t)}{\partial w_j^{1,i}}$$

$$\frac{\partial E(t)}{\partial w_j^{1,i}} = 2 \cdot (F(x^t, W(t)) - y_t) \cdot \frac{\partial}{\partial w_j^{1,i}} \left(\frac{1}{1 + e^{-w^{1,iT} x^t - a_i}} \right)$$

$$\frac{\partial}{\partial w_j^{1,i}} \left(\frac{1}{1 + e^{-w^{1,iT} x^t - a_i}} \right) = \frac{e^{-w^{1,iT} x^t - a_i}}{\left(1 + e^{-w^{1,iT} x^t - a_i}\right)^2} \cdot \frac{\partial}{\partial w_j^{1,i}} \left(- w^{1,iT} x^t - a_i \right)$$

$$\frac{\partial}{\partial w_j^{1,i}} \left(- w^{1,iT} x^t - a_i \right) = -x_j^t$$

$$w_j^{1,i}(t+1) = w_j^{1,i}(t) - c \cdot 2 \cdot (F(x^t, W(t)) - y_t) \cdot \frac{e^{-w^{1,iT} x^t - a_i}}{\left(1 + e^{-w^{1,iT} x^t - a_i}\right)^2} \cdot (-x_j^t)$$

New methods for learning with neural networks

Bayesian learning:

the distribution of the neural network parameters is learnt

Support vector learning:

the minimal representative subset of the available data is used to calculate the synaptic weights of the neurons

A new sort of computer

- What are (everyday) computer systems good at... and not so good at?

Good at	***Not so good at***
Rule-based systems: doing what the programmer wants them to do	Dealing with noisy data
	Dealing with unknown environment data
	Massive parallelism
	Fault tolerance
	Adapting to circumstances

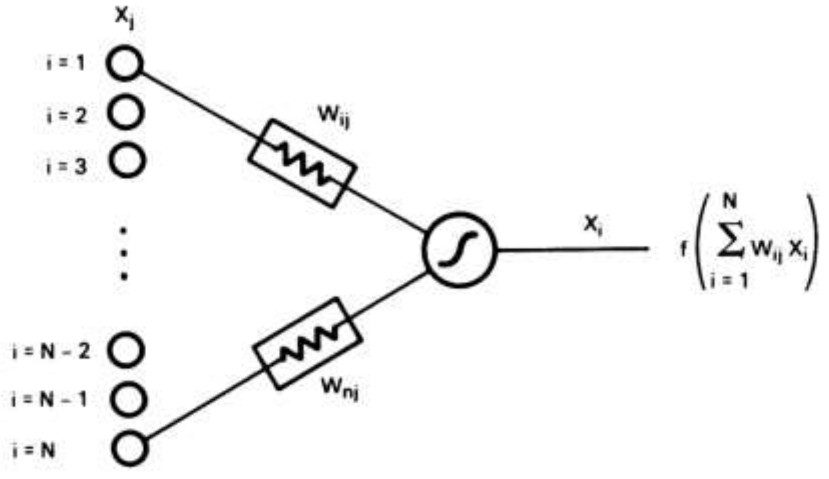
Neural networks as computers

- **Neural network:** *information processing paradigm inspired by biological nervous systems, such as our brain*
- Structure: large number of highly interconnected processing elements (*neurons*) working together
- Like people, they learn *from experience* (by example)

Brains vs Computers : Some Numbers

- There are approximately 10 billion neurons in the human cortex, compared with 10 of thousands of processors in the most powerful parallel computers.
- Each biological neuron is connected to several thousands of other neurons, similar to the connectivity in powerful parallel computers.
- Lack of processing units can be compensated by speed. The typical operating speeds of biological neurons is measured in milliseconds (10-3 s), while a silicon chip can operate in nanoseconds (10-9 s).
- The human brain is extremely energy efficient, using approximately 10-16 joules per operation per second, whereas the best computers today use around 10-6 joules per operation per second. Brains have been evolving for tens of millions of years, computers have been evolving for tens of decades.

DARPA 1988 Report on comparision



- MANY MORE INTERCONNECTS THAN NEURONS
- EACH INTERCONNECT REQUIRES 1 MULTIPLY/ACCUMULATE OPERATION

	STORAGE (memory)	SPEED (operations/s)
DIGITAL COMPUTER	WORDS	IPS, FLOPS
NEURAL NETWORK	INTERCONNECTS	INTERCONNECTS/s

Figure 2-12. Neural Network Computation Units.

Comparison with Biological Networks (DARPA 1988)

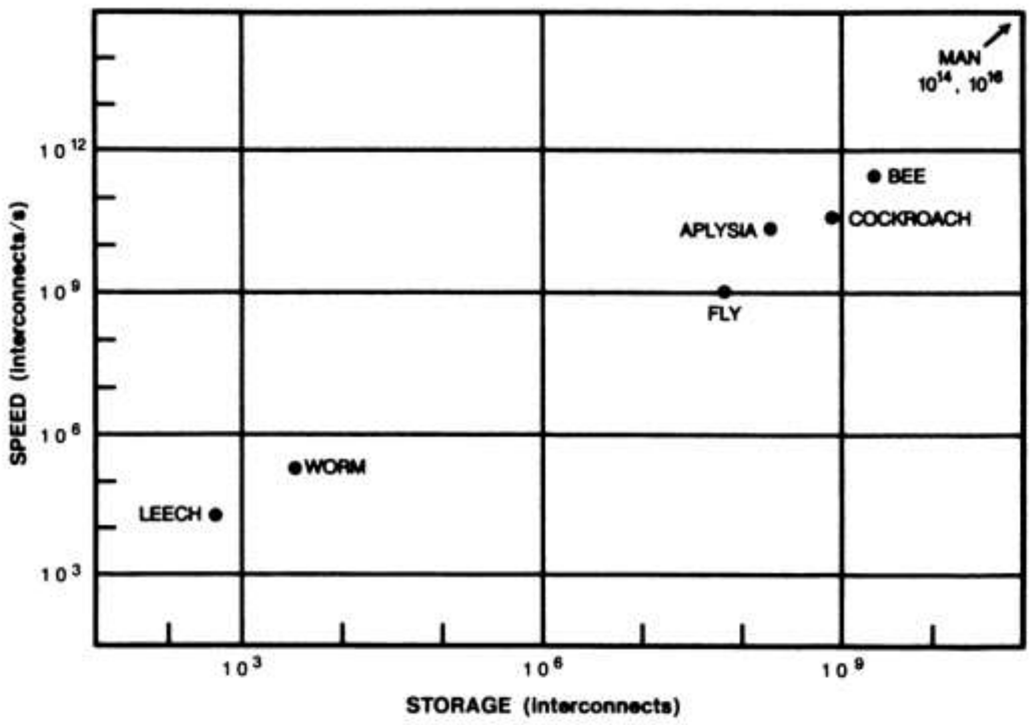


Figure 2-13. Computational Capabilities of Biological Networks.

Extracted from DARPA Report of 1988

Current Projects: DARPA SyNAPSE Project and others (www.artificial brian.com)
SpiNNaker (Manchester); NeuroGRID (Stanford) and several more

Neural networks and learning

- Neural networks are configured for a specific application, such as pattern recognition or data classification, through a learning process
- In a biological system, learning involves adjustments to the synaptic connections between neurons

➔ same for artificial neural networks (ANNs)

Where can neural network systems help

- when we can't formulate an algorithmic solution.
- when we ***can*** get lots of examples of the behavior we require.

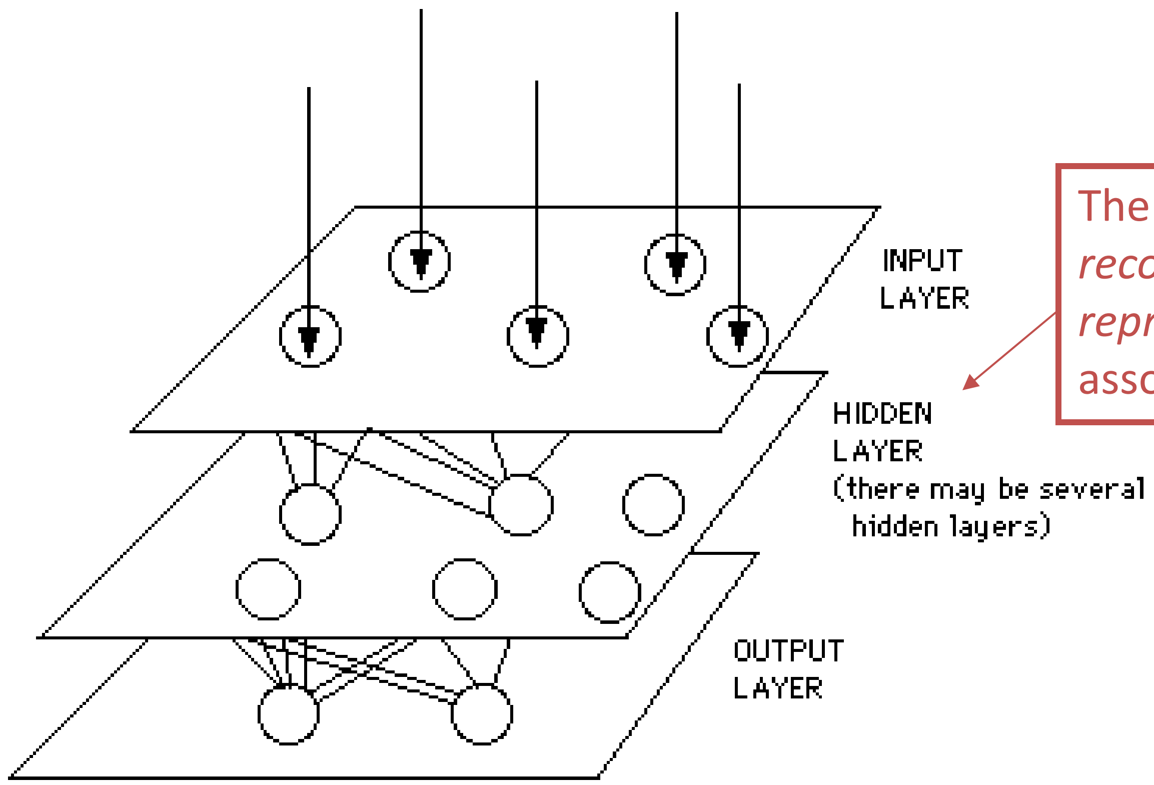
‘learning from experience’

- when we need to pick out the structure from existing data.

Artificial Neural Networks

Adaptive interaction between individual neurons

Power: collective behavior of interconnected neurons



The hidden layer learns to *recode* (or to *provide a representation* of) the inputs: associative mapping

Where are NN used?

- Recognizing and matching complicated, vague, or incomplete patterns
- Data is unreliable
- Problems with noisy data
    - Prediction
    - Classification
    - Data association
    - Data conceptualization
    - Filtering
    - Planning

Applications

- **Prediction: learning from past experience**
  - *pick the best stocks in the market*
  - *predict weather*
  - *identify people with cancer risk*
- **Classification**
  - *Image processing*
  - *Predict bankruptcy for credit card companies*
  - *Risk assessment*

Applications

- **Recognition**
  - *Pattern recognition: SNOOPE* (bomb detector in U.S. airports)
  - *Character recognition*
  - *Handwriting: processing checks*
- **Data association**
  - *Not only identify the characters that were scanned but identify when the scanner is not working properly*

Applications

- **Data Conceptualization**
  - *infer grouping relationships*
    e.g. extract from a database the names of those most likely to buy a particular product.
- **Data Filtering**

  e.g. take the noise out of a telephone signal, signal smoothing
- **Planning**
  - *Unknown environments*
  - *Sensor data is noisy*
  - *Fairly new approach to planning*

Strengths of a Neural Network

- **Power**: Model complex functions, nonlinearity built into the network
- **Ease of use**:
  - Learn by example
  - Very little user domain-specific expertise needed
- **Intuitively appealing**: based on model of biology, will it lead to genuinely intelligent computers/robots?

Neural networks cannot do anything that cannot be done using traditional computing techniques, **BUT** they can do some things which would otherwise be very difficult.

General Advantages

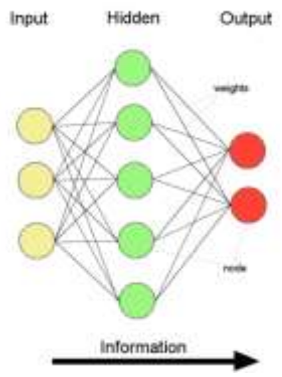
- Advantages
  - Adapt to unknown situations
  - Robustness: fault tolerance due to network redundancy
  - Autonomous learning and generalization
- Disadvantages
  - Not exact
  - Large complexity of the network structure
- For motion planning?

Status of Neural Networks

- Most of the reported applications are still in research stage
- No formal proofs, but they seem to have useful applications that work

Recurrent Networks

- Feed forward networks:
  - Information only flows one way
  - One input pattern produces one output
  - No sense of time (or memory of previous stat

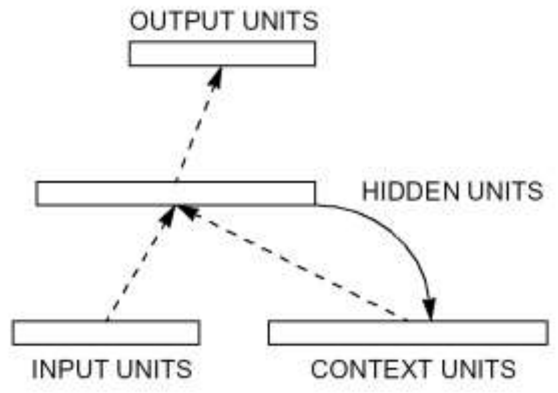


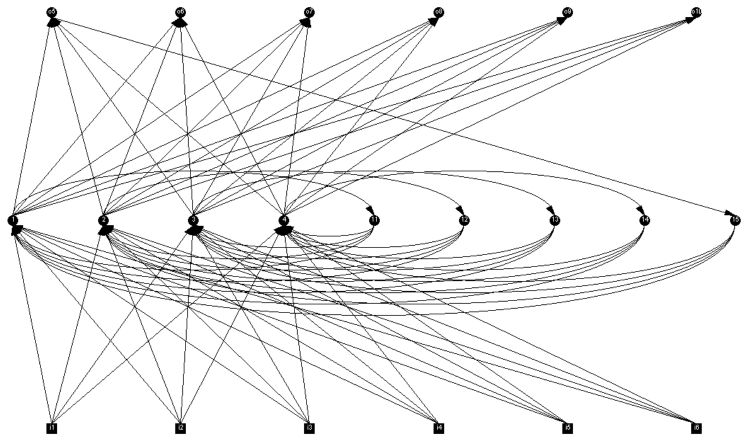
- Recurrency
  - Nodes connect back to other nodes or themselves
  - Information flow is multidirectional
  - Sense of time and memory of previous state(s)

- Biological nervous systems show high levels of recurrency (but feed-forward structures exists too)

Elman Nets

- *Elman nets* are feed forward networks with partial recurrency



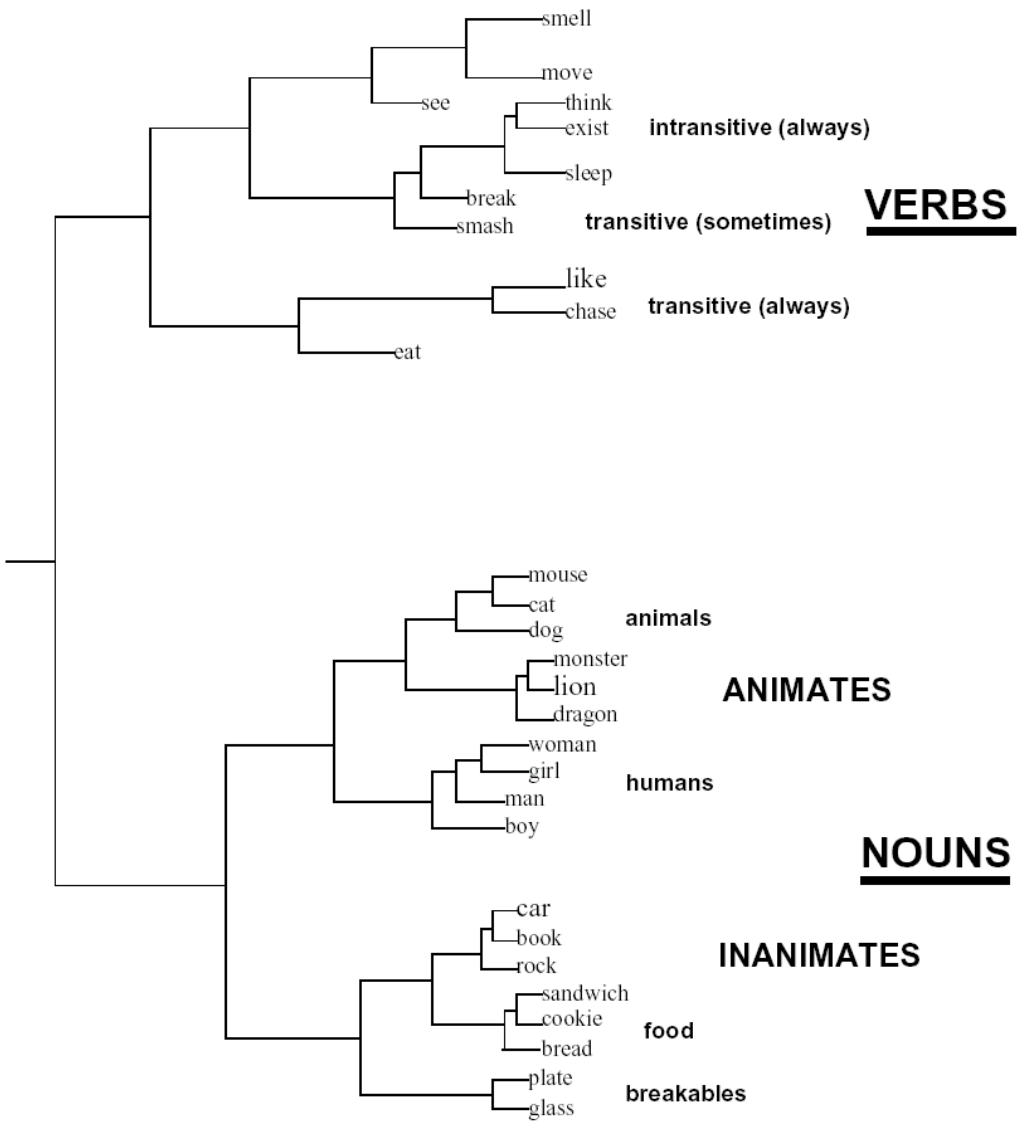


- Unlike feed forward nets, Elman nets have a *memory* or *sense of time*

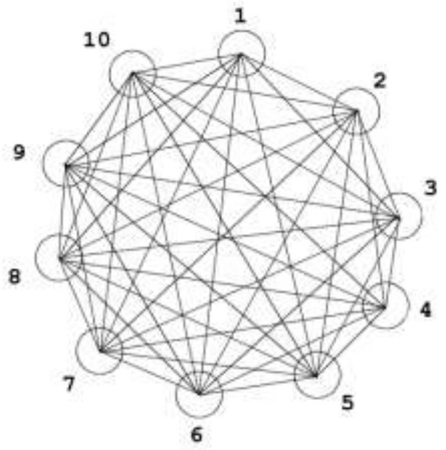
Classic experiment on language acquisition and processing (Elman, 1990)

- **Task**
  - Elman net to predict successive words in sentences.
- **Data**
  - Suite of sentences, e.g.
    - "The boy catches the ball."
    - "The girl eats an apple."
  - Words are input one at a time
- **Representation**
  - Binary representation for each word, e.g.
    - 0-1-0-0-0 for "girl"
- **Training method**
  - Backpropagation

- Internal representation of words

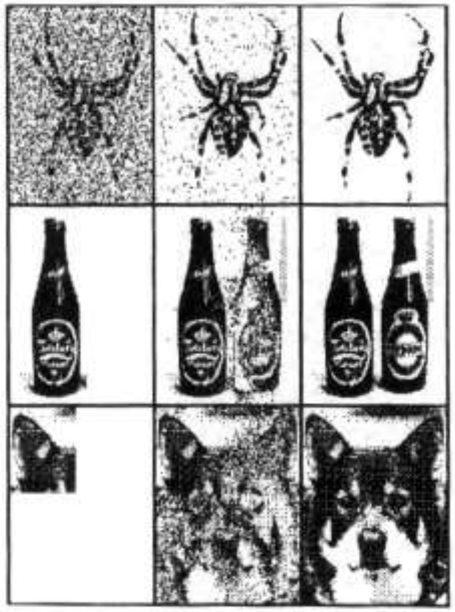


Hopfield Networks



- Sub-type of recurrent neural nets
  - Fully recurrent
  - Weights are symmetric
  - Nodes can only be *on* or *off*
  - Random updating
- Learning: **Hebb rule** (cells that fire together wire together)
  - Biological equivalent to LTP and LTD
- Can recall a memory, if presented with a corrupt or incomplete version

→ **auto-associative** or **content-addressable memory**



Task: store images with resolution of 20x20 pixels

→ Hopfield net with 400 nodes

Memorise:

1. Present image
2. Apply Hebb rule (*cells that fire together, wire together*)
   - Increase weight between two nodes if both have same activity, otherwise decrease
3. Go to 1

Recall:

1. Present incomplete pattern
2. Pick random node, update
3. Go to 2 until settled

Catastrophic forgetting

- *Problem*: memorising new patterns corrupts the memory of older ones
  - → Old memories cannot be recalled, or spurious memories arise

- *Solution*: allow Hopfield net to **sleep**

- Two approaches (both using randomness):

  - **Unlearning** (Hopfield, 1986)
    - Recall old memories by random stimulation, but use an *inverse* Hebb rule

    → 'Makes room' for new memories (basins of attraction shrink)

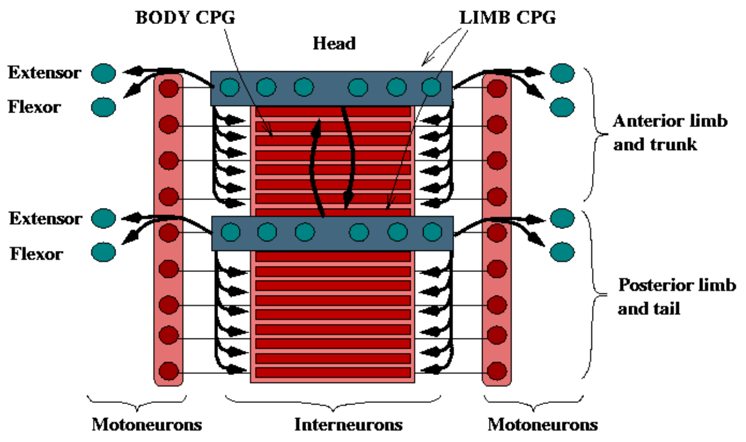
  - **Pseudorehearsal** (Robins, 1995)
    - While learning new memories, recall old memories by random stimulation
    - Use *standard* Hebb rule on new and old memories

    → Restructure memory
    - Needs short-term + long term memory
    - Mammals: hippocampus plays back new memories to neo-cortex, which is randomly stimulated at the same time

RNNs as Central Pattern Generators

- **CPGs:** group of neurones creating rhythmic muscle activity for locomotion, heart-beat etc.
- Identified in several invertebrates and vertebrates
- Hard to study

- → Computer modelling
  - E.g. lamprey swimming (Ijspeert *et al.*, 1998)


BODY CPG
LIMB CPG
Head
Extensor
Flexor
Extensor
Flexor
Anterior limb and trunk
Posterior limb and tail
Motoneurons
Interneurons
Motoneurons