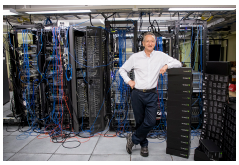
Part V

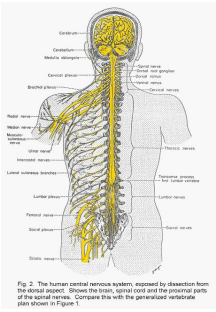


Neural Networks

Structure

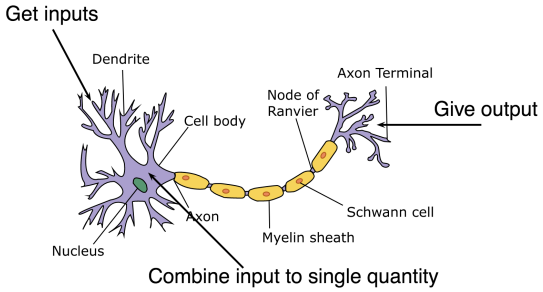
- Biological inspiration
- History of Artificial Neural Networks (ANNs)
- Two basic models of neural networks:
  - The perceptron
  - The feed-forward network
- Learning in ANNs:
  - The backpropagation algorithm

Brain and nervous system

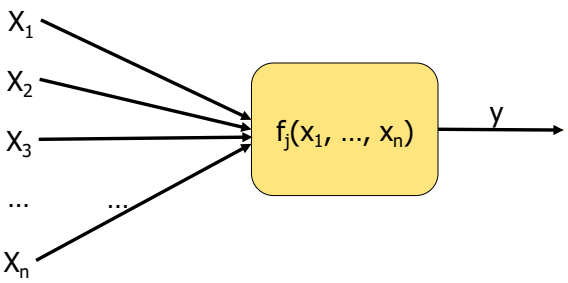


- 86bn neurons in human brain; 30bn in gorillas; 14bn in baboons
- Each is connected to around 10,000 neurons
- Regions of the brain are (somewhat) specialised.
- Some neurons connect to senses (input) and muscles (action).

Detail of a Neuron



General model for neuron j



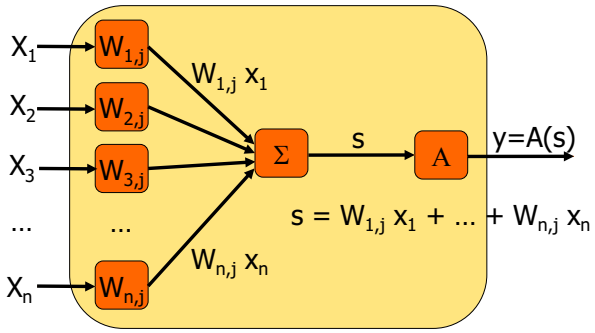
What are the inputs/outputs?

- Inputs (x_1, …, x_n): sensors on a robot
- Output (y): motor speed

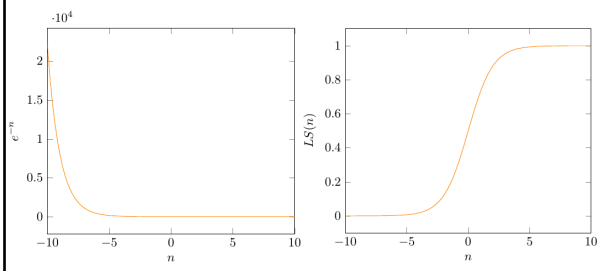
- Inputs: pixel values
- Output: a decimal number

- Inputs: vital signs of patient
- Output: likelihood of a heart attack

Example model for neuron j



Log-Sigmoid function (switch)
$LS(n) = 1/(1 + e^{-n})$



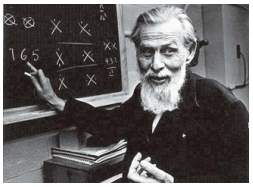
Exemplar of Bio-inspired computing

- Looking at how natural world deals with problem-solving:
  - ants
  - bees
  - immune systems
- Abstract out underlying mechanisms
- Simulate mechanisms with programs
- Using implementation to solve applications in novel areas

Short History of Artificial Neural Networks

1943: McCullough and Pitts

Described first electronical model of how neurons *might* work as a sum and threshold

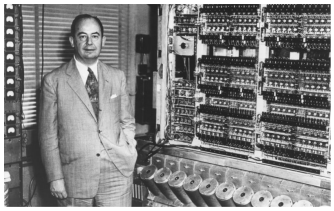


1949: Hebb

- Published book "The Organisation of Behaviour"
- Pointed out that neural pathways are strengthened each time that they are used
- "Cells that fire together, wire together"

1955: von Neumann

Suggested imitating simple neuron functions by using telegraph relays (switch) or vacuum tubes (valve)



1960: Rosenblatt

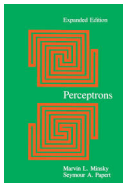
- Found that processing that tells a fly where to flee is performed locally in its eye
- Build a one-layer network (perceptron) in hardware
- Developing training
- "the embryo of an electronic computer that … will be able to walk, talk, see, write, reproduce itself …"

1962: Widrow and Hoff

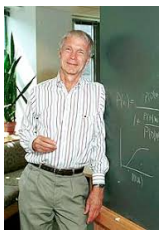
- Error minimisation for linear neural (output was weighted sum of inputs)
- Multiple ADAptive LINear Elements (MADALINE) eliminated echoes on a phone line
- First real application

1969: Minsky and Papert

Showed perceptrons cannot calculate exclusive-or, hence the "disillusioned years"



1982: Hopfield



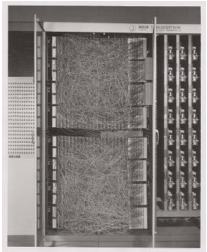
- National Academy of Sciences paper proposed a useful device rather than a brain modem
- Hopfield network is a recurrent neural network that uses internal state to process sequences of inputs
- Renewed interest

1986 and 2012: Hinton

- Nature paper showed how generalised back-propagation could train multi-layer networks
- Convolution Neural Network won the Olympics of vision
- Trained with GPUs for days

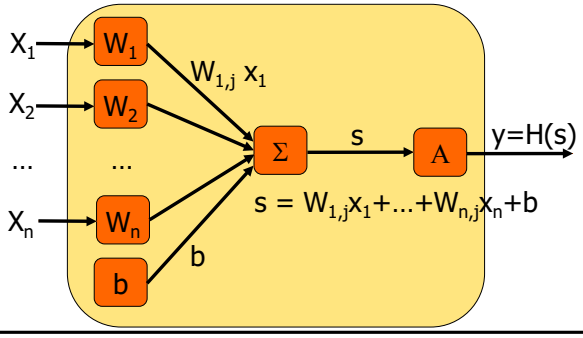
Applications of ANNs

- Pattern cognition:
  - early detection of breast cancer
  - identifying lame cattle
- Computer vision:
  - separating signals of different origin in sky maps
  - recognizing letters and reading signs
- Anomaly detection:
  - network management
  - malware analysis and classification



Perceptron (single-layer as opposed to multi-layer)

Perceptron (single neuron)



Sigma (Σ) and dot (.) notation

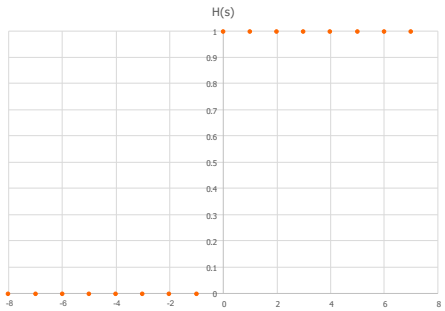
- Clear but cumbersome:
  - $s = w_{1,j}x_1 + \ldots + w_{n,j}x_n + b$
- Sigma notation:
  - $s = (\Sigma_{k=1}{}^{k=n}\ w_{k,j}x_k) + b$

really written as:

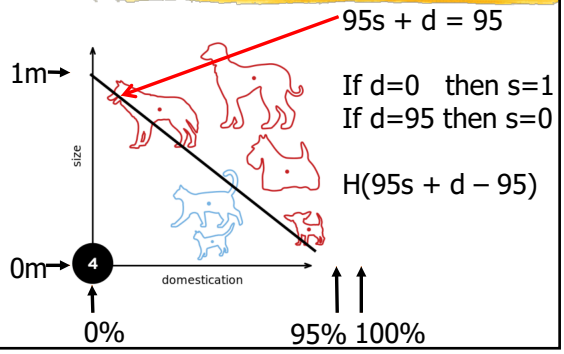
$$S = \left(\sum_{k=1}^{k=n} w_{kj} x_k \right) + w_{n+1,j}$$

- Dot notation:
  - $s = (w_{1,j}, \ldots, w_{n,j}) \cdot (x_1, \ldots, x_n) + b$ or
  - $s = \mathbf{w} \cdot (x_1, \ldots, x_n) + b$ for $\mathbf{w} = (w_{1,j}, \ldots, w_{n,j})$
  - **bold** is used to vector type

Activation function (switch) Heaviside step function



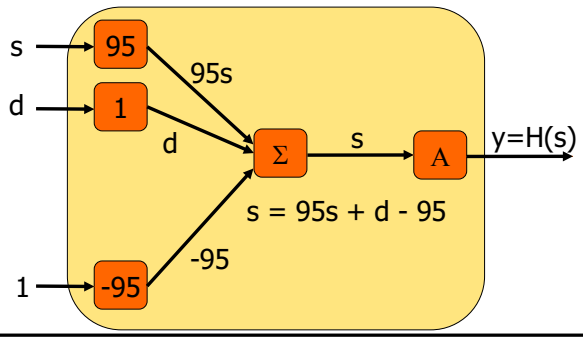
Perceptron to detect dogs using s and d



If d=0 then s=1
If d=95 then s=0

H(95s + d – 95)

Perceptron to detect dogs



Rosenblatt's learning rule

- Train on $\mathbf{p}_1=(1,1)$, $\mathbf{p}_2=(-1,1)$, $\mathbf{p}_3=(0,-1)$ using labels $t_1 = 1$, $t_2 = 0$, $t_3 = 0$
- Fix $0<\alpha<1$ which is the learning rate
- Assign a random $\mathbf{w}_0$ and b_0 and then compute new $\mathbf{w}_1$, b_1, $\mathbf{w}_2$, b_2 ... using:
- If $t_k = 1$ and $H(\mathbf{w}_j.\mathbf{p}_k+b_j) = 0$ then assign $\mathbf{w}_{j+1} = \mathbf{w}_j + \alpha\mathbf{p}_k$ and $b_{j+1} = b_j + \alpha$
- If $t_k = 0$ and $H(\mathbf{w}_j.\mathbf{p}_k+b_j) = 1$ then assign $\mathbf{w}_{j+1} = \mathbf{w}_j - \alpha\mathbf{p}_k$ and $b_{j+1} = b_j - \alpha$
- Otherwise stop

Intuition behind learning rule (to see $\mathbf{w}_{j+1}.\mathbf{p}_k \geq \mathbf{w}_j.\mathbf{p}_k$)

- Suppose $t_k = 1$ and $H(\mathbf{w}_j . \mathbf{p}_k+b_j) = 0$
- Then $\mathbf{w}_j.\mathbf{p}_k+b_j < 0$ but want $\mathbf{w}_{j+1}.\mathbf{p}_k+b_{j+1} \geq 0$

- Consider $\mathbf{w}_{j+1}.\mathbf{p}_k = (\mathbf{w}_j + \alpha\mathbf{p}_k).\mathbf{p}_k =$
- $(w_j^1 + \alpha p_k^1, \ldots, w_j^n + \alpha p_k^n) . (p_k^1, \ldots, p_k^n) =$
- $(w_j^1 + \alpha p_k^1)p_k^1 + \ldots + (w_j^n + \alpha p_k^n)p_k^n =$
- $(w_j^1 p_k^1 + \ldots + w_j^n p_k^n) + \alpha(p_k^1 p_k^1 + \ldots + p_k^n p_k^n) =$
- $\mathbf{w}_j.\mathbf{p}_k + \alpha(p_k^1 p_k^1 + \ldots + p_k^n p_k^n) \geq \mathbf{w}_j.\mathbf{p}_k$

Intuition behind learning rule (to see $\mathbf{w}_{j+1}.\mathbf{p}_k+b_{j+1} > \mathbf{w}_j.\mathbf{p}_k+b_j$)

- Suppose $t_k = 1$ and $H(\mathbf{w}_j.\mathbf{p}_k+b_j) = 0$
- Then $\mathbf{w}_j.\mathbf{p}_k+b_j < 0$ but want $\mathbf{w}_{j+1}.\mathbf{p}_k+b_{j+1} \geq 0$

- Now consider $\mathbf{w}_{j+1}.\mathbf{p}_k + b_{j+1} \geq \mathbf{w}_j.\mathbf{p}_k + (b_j + \alpha)$ $= (\mathbf{w}_j.\mathbf{p}_k + b_j) + \alpha$

- Although $\mathbf{w}_{j+1}.\mathbf{p}_k+b_{j+1} > \mathbf{w}_j.\mathbf{p}_k+b_j$ does not always follow $\mathbf{w}_{j+1}.\mathbf{p}_k+b_{j+1} \geq 0$ so then repeat

Intuition behind learning rule

- Suppose $t_k = 0$ and $H(w_j . p_k) = 1$
- Likewise $\mathbf{w}_{j+1}.\mathbf{p}_k+b_{j+1} \leq (\mathbf{w}_j.\mathbf{p}_k+b_j) - \alpha$
- And likewise although $\mathbf{w}_{j+1}.\mathbf{p}_k+b_{j+1} < \mathbf{w}_j.\mathbf{p}_k+b_j$ does not necessarily follow $\mathbf{w}_{j+1}.\mathbf{p}_k+b_{j+1} < 0$

- Keep going until reach w_m with:
  - $H(w_m . p_1) = t_1$
  - $H(w_m . p_2) = t_2$
  - $H(w_m . p_3) = t_3$

Learning with $\mathbf{p}_1=(1,1)$, $\mathbf{p}_2=(-1,1)$, $\mathbf{p}_3=(0,-1)$

- Recall $t_1 = 1$, $t_2 = 0$, $t_3 = 0$

- Step 0: assign $\mathbf{w}_0 = (-3, 0)$ and $b_0 = 0$

- Step 1: $H(\mathbf{w}_0.\mathbf{p}_1+b_0) = H(-3) = 0$ but $t_1 = 1$
- Put $\mathbf{w}_1 = \mathbf{w}_0+\alpha\mathbf{p}_1$
- $= (-3, 0) + 0.1(1, 1) = (-2.9, 0.1)$
- Put $b_1 = b_0 + \alpha = 0 + 0.1 = 0.1$

Greedy error reduction

w		b	w.p1+b	t1		new w		new b	w.p2+b	t2		new w		new b	w.p3+b	t3		new w		new b
-3	0	0	-3	1	3	-2.9	0.1	0.1	3	0	3	-2.9	-0.1	-0.1	0	0	0	-3	0.1	-0.1
-2.9	0.1	0.1	-2.7	1	2.7	-2.8	0.2	0.2	3.1	0	3.1	-2.8	0	0	0	0	0	-2.9	0.2	0
-2.8	0	0	-2.8	1	2.8	-2.7	0.1	0.1	2.8	0	2.8	-2.7	-0.1	-0.1	0	0	0	-2.8	0.1	-0.1
-2.7	0.1	0.1	-2.5	1	2.5	-2.6	0.2	0.2	2.9	0	2.9	-2.6	0	0	0	0	0	-2.7	0.2	0
-2.6	0	0	-2.6	1	2.6	-2.5	0.1	0.1	2.6	0	2.6	-2.5	-0.1	-0.1	0	0	0	-2.6	0.1	-0.1
-2.5	0.1	0.1	-2.3	1	2.3	-2.4	0.2	0.2	2.7	0	2.7	-2.4	0	0	0	0	0	-2.5	0.2	0
-2.4	0	0	-2.4	1	2.4	-2.3	0.1	0.1	2.4	0	2.4	-2.3	-0.1	-0.1	0	0	0	-2.4	0.1	-0.1
-2.3	0.1	0.1	-2.1	1	2.1	-2.2	0.2	0.2	2.5	0	2.5	-2.2	0	0	0	0	0	-2.3	0.2	0
-2.2	0	0	-2.2	1	2.2	-2.1	0.1	0.1	2.2	0	2.2	-2.1	-0.1	-0.1	0	0	0	-2.2	0.1	-0.1
-2.1	0.1	0.1	-1.9	1	1.9	-2	0.2	0.2	2.3	0	2.3	-2	0	0	0	0	0	-2.1	0.2	0
-2	0	0	-2	1	2	-1.9	0.1	0.1	2	0	2	-1.9	-0.1	-0.1	0	0	0	-2	0.1	-0.1

Learning with $p_1=(1,1)$, $p_2=(-1,1)$, $p_3=(0,-1)$

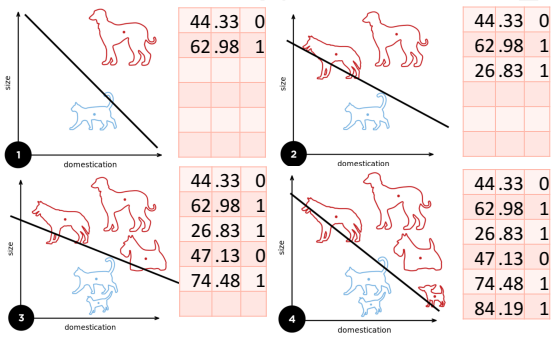
- Step 2: $H(\mathbf{w}_1.\mathbf{p}_2+b_1) = H(2.7) = 1$ but $t_2 = 0$
- Put $\mathbf{w}_2 = \mathbf{w}_1 - \alpha\mathbf{p}_2$
- $= (-2.9, 0.1) - 0.1(-1, 1) = (-2.8, 0)$
- Put $b_2 = b_1 - \alpha = 0.1 - 0.1 = 0$

- On step 38: $\mathbf{w}_{38} = (0, 0.1)$ and $b_{38} = -0.1$
  - $H(\mathbf{w}_{38}.\mathbf{p}_1+b_{38}) = H(10^{-15}) = 1 = t_1$
  - $H(\mathbf{w}_{38}.\mathbf{p}_2+b_{38}) = H(0) = 0 = t_2$
  - $H(\mathbf{w}_{38}.\mathbf{p}_3+b_{38}) = H(-0.2) = 0 = t_3$

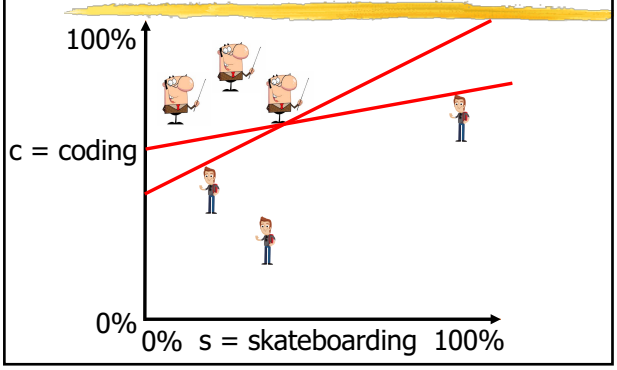
First 10 of the 38 (greedy) learning steps

w		b	w.p1+b	H(w.p1+b)	err	new w		new b	w.p2+b	H(w.p2+b)	err	new w		new b	w.p3+b	H(w.p3+b)	err	new w		new b
-3	0	0	-3	0	3	-2.9	0.2	0.1	3	1	3	-2.9	-0.2	-0.1	0	1	0	-3	0.1	-0.1
-2.9	0.2	0.1	-2.4	0	2.4	-2.8	0.4	0.2	3.4	1	3.4	-2.8	0	0	-0.1	0	0	-2.9	0.2	0.1
-2.8	0	0	-2.8	0	2.8	-2.7	0.2	0.1	2.8	1	2.8	-2.7	-0.2	-0.1	0	1	0	-2.8	0.1	-0.1
-2.7	0.2	0.1	-2.2	0	2.2	-2.6	0.4	0.2	3.2	1	3.2	-2.6	0	0	-0.1	0	0	-2.7	0.2	0.1
-2.6	0	0	-2.6	0	2.6	-2.5	0.2	0.1	2.6	1	2.6	-2.5	-0.2	-0.1	0	1	0	-2.6	0.1	-0.1
-2.5	0.2	0.1	-2	0	2	-2.4	0.4	0.2	3	1	3	-2.4	0	0	-0.1	0	0	-2.5	0.2	0.1
-2.4	0	0	-2.4	0	2.4	-2.3	0.2	0.1	2.4	1	2.4	-2.3	-0.2	-0.1	0	1	0	-2.4	0.1	-0.1
-2.3	0.2	0.1	-1.8	0	1.8	-2.2	0.4	0.2	2.8	1	2.8	-2.2	0	0	-0.1	0	0	-2.3	0.2	0.1
-2.2	0	0	-2.2	0	2.2	-2.1	0.2	0.1	2.2	1	2.2	-2.1	-0.2	-0.1	0	1	0	-2.2	0.1	-0.1
-2.1	0.2	0.1	-1.6	0	1.6	-2	0.4	0.2	2.6	1	2.6	-2	0	0	-0.1	0	0	-2.1	0.2	0.1

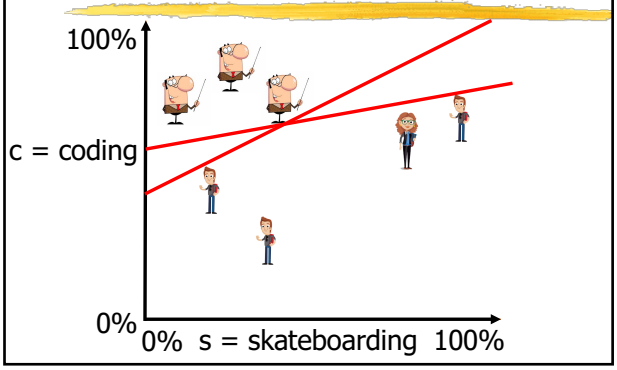
Learning with cats and dogs (in epochs)



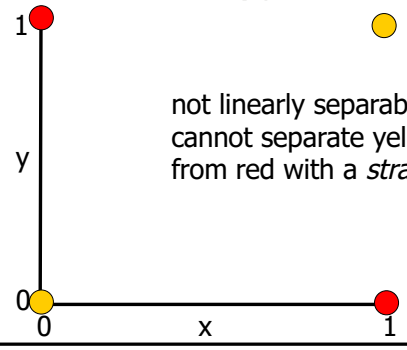
Perceptron to detect academics using c and s



Perceptron fails to detect academics using c and s



Minsky and Papert xor example



not linearly separable: cannot separate yellow from red with a *straight* line

Rosenblatt's learning rule on xor

- Learning will never terminate for xor, whatever the initial weights and bias
- But Rosenblatt's learning rule will always terminate (whatever the weights and bias) if the problem is linearly separable
- Never oscillate between $\mathbf{p}_i$ and $\mathbf{p}_j$ forever

Summary

- History of Artificial Neural Networks (ANNs)
- Introduced the Perceptron
- Understood Rosenblatt's learning rule
- Applied Rosenblatt's learning rule
- Seen the limitations of the Perceptron