Genetic Algorithms and Genetic Programming
Lecture 6

Gillian Hayes

10th October 2006

School of informatics

Evolving Neural Networks

- Reminder of neural networks
- Evolving weights
- Evolving network topology
- Grammars, robotics
- Evolving intelligent behaviours
- Example: evolving communication

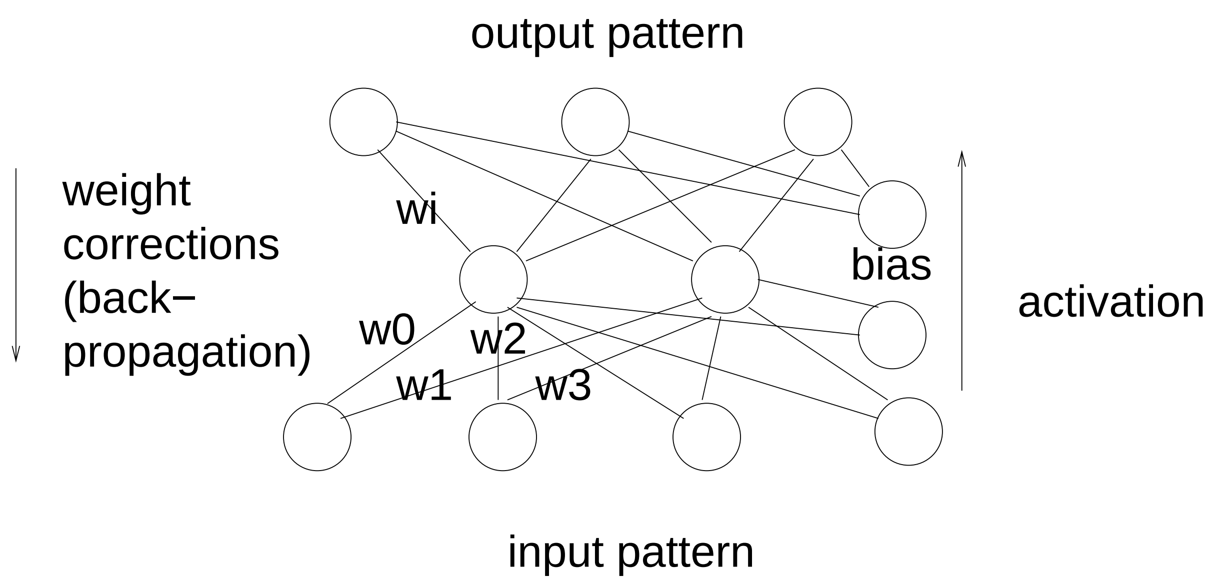
Neural Networks

- Inspired by working of neurons in the brain
- Universal function approximators
- Used widely in machine learning
- Empirical predictive modelling
- Classification
- Robotic controllers

Reminder of Neural Networks

- Nodes and connections
- Weights attached to the connections
- Firing depends on inputs to the node
- Activation threshold function
- Input/hidden/output layers
- Feedforward networks
- Recurrent networks
- Training: back propagation

A Simple Feedforward Neural Network



Learning procedure: use a training set of <input, output> pairs.
Present input, try to adjust weights to reduce the difference between the network's output and the desired output. (Rumelhart et al. 1986)

– supervised learning procedure

Evolving Weights 1

- in a fixed network
- as an alternative to back-propagation

Montana and Davis (IJCAI 1989) looked at:

– underwater sonic recordings (features, preprocessed)

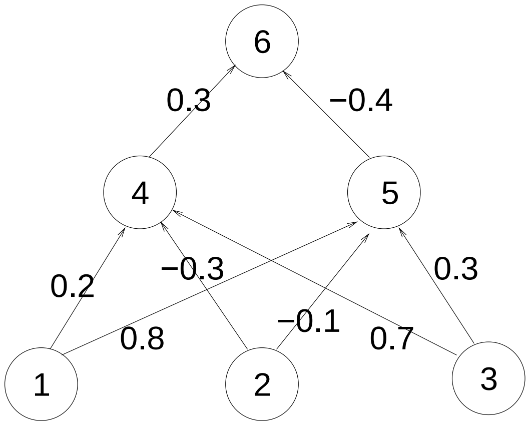
– treated as a classification problem (whales, enemy subs)

– network topology

4 input units	
7 units in hidden layer 1	fully connected
10 units in hidden layer 2	18 extra thresholding connections (biases)
1 output unit	total weights 126

– GA chromosome: a list of 126 real-valued weights

Evolving Weights 2



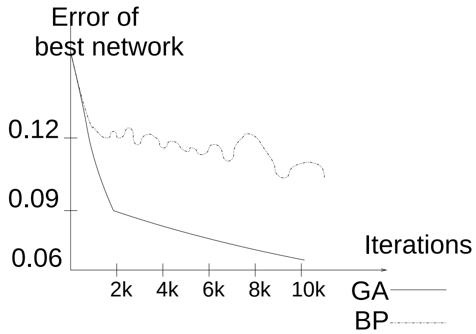
Chromosome: (0.3, -0.4, 0.2, -0.3, 0.7, 0.8, -0.1, -0.3)

Building blocks: all incoming weights to a given unit seems plausible.

Mutation: for **each** link coming in to the chosen unit, add a (different) random value between +1.0 and -1.0

Crossover: for each non-input unit, choose **all** the weights from Parent 1 or **all** the weights from Parent 2.

Evolving Weights 3: Results



Advantages of GA:

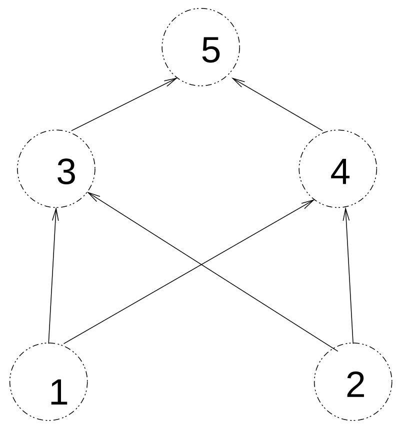
- better than BP for some tasks
- 'unsupervised' learning
- sparse reinforcement available, e.g. robots in unfamiliar environments
- may only need it to work in some parts of the input/output space, i.e. those experienced

Evolving Networks 1

– choosing a network topology is hard

– can it be done automatically?

Miller, Todd and Hegde (1989):



from unit:		1	2	3	4	5
to unit:	1	0	0	0	0	0
	2	0	0	0	0	0
	3	1	?	?		
	4	1	?			
	5	0	?			?

Chromosome: 00000 00000 ... (complete the rest...)

Mutation: bit flipping

Crossover: exchange whole rows

Limit to **feedforward networks**: any links to input units or feedback connections are ignored.

Evolving Networks 2

Tasks tried by Miller et al.:

(a) XOR (exclusive - OR)

(b) four quadrant:

$$< x, y > \rightarrow 0.0 \text{ if } x, y \simeq 0.0 \text{ or } x, y \simeq 1.0$$

$$< x, y > \rightarrow 1.0 \text{ otherwise}$$

(c) pattern copying, with units in the hidden layer $<$ number of input units

Learning: back-propagation

Results: GA can easily find network topologies for these problems.

But are the problems too easy?

See Whitley and Schaffer (1992) for a more sophisticated approach

Grammars and Robotics

- Grammatical encoding of the linkage matrix (here for XOR)

$$
S \rightarrow \begin{array}{cc} A & B \\ C & D \end{array} \rightarrow \begin{array}{cccc} c & p & a & a \\ a & c & a & e \\ a & a & a & a \\ a & a & a & b \end{array} \rightarrow \begin{array}{cccccccc} 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}
$$

(S A B C D | A c p a c | B a a a e . . .)

Generate linkage matrix from the grammar. If at the end of rewriting there are still non-terminal nodes, that node is “dead” – not connected.

• Develop chromosome (genotype) into network (phenotype) and train for fixed no. of training episodes.

• Fitness = error at end of training

Problems with direct encoding

Fixed connections: as size of matrix grows, chromosome size grows

Can't encode repeated patterns, esp. with internal structure

Takes a long time to generate high-performing networks

Advantages of grammatical encoding

Can represent large connectivity matrices in compact form

Shorter encoding, faster search

Variable topologies including recurrent connections

Better on encoder/decoder problem than direct encoding

Evolving Neural Network Behaviours

- Previous examples rely on *training data*
- What if we haven't got any?
- Example: a neural network which controls a mobile agent which is trying to achieve some goals in a dynamic environment.
- No good example of behaviour is available; or we wish to try a range of possible behaviours to see which is best.
- A fitness function is available based on goal achievement.

Evolving Neural Network Behaviours

General approach:

- Decide on how to represent inputs to and outputs from the neural network.
- Decide on a neural network architecture: might need to try a range of possibilities.
- Decide on a simulation which tests the NN's behaviour.
- Decide on a fitness function which tests how well the NN did in the simulation.
- All the usual GA stuff: chromosome representation, crossover, mutation, population size, etc.

Example: Evolving Communication

This is an example from Artificial Life: the study of computer generated "life" forms. (Matthew Quinn, University of Sussex)

- Khepera robots controlled by evolved neural networks
- Group task: robots move together as far as possible – like dancing
- 8 sensor nodes, 4 motor nodes, hidden nodes
- Evolved thresholds, weights, decay parameters, size, connectivity of network
- **Co-evolution**: select two robots from population, rate them for fitness *as a pair*
- Initial result: leaders and followers emerge
- Only get a working pair 50% of the time

Example: Evolving Communication

- After a while a new single species emerges

- This behaviour uses communication based on simple movement:
  - both agents (A and B) rotate anti-clockwise
  - one agent (B) becomes aligned first and moves towards the other agent
  - agent B moves backward and forward while staying close to A
  - when A becomes aligned, it becomes the leader: it reverses its direction and is followed by B

- Very similar to movement communication used in social insects (e.g. dancing in honey bees)

Next: more theory, then another agent/robot example