

CSC 121
Computers and Scientific
Thinking

Fall 2005

Applications in
Artificial Intelligence

1

Artificial Intelligence



Artificial Intelligence (AI) is a subfield of computer science closely tied with biology and cognitive science

- AI is concerned with computing techniques and models that simulate/investigate intelligent behavior
- AI research builds upon our understanding of the brain and evolutionary development
- in return, AI research provides insights into the way the brain works, as well as the larger process of biological evolution

two hot research areas in AI are:

1. *neural networks*: building a model of the brain and "training" that model to recognize certain types of patterns
2. *genetic algorithms*: "evolving" solutions to complex problems (especially problems that are intractable using other methods)

2

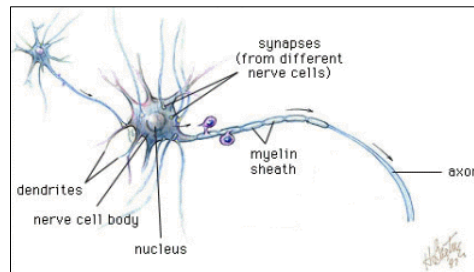
Neural Networks

the idea of neural networks predates modern computers

- in 1943, McCulloch and Pitts described a simple computational model of a neuron

neural networks were a focus of CS research in the 1950's

- humans lack the speed & memory of computers, yet are capable of complex reasoning/action → maybe our brain architecture is well-suited for certain tasks



general brain architecture:

- many (relatively) slow neurons, interconnected
- dendrites serve as input devices (receive electrical impulses from other neurons)
- cell body "sums" inputs from the dendrites (possibly inhibiting or exciting)
- if sum exceeds some threshold, the neuron fires an output impulse along axon

3

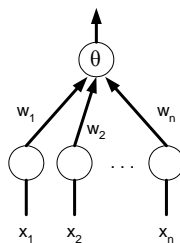
Artificial Neurons

neural networks are based on the brain metaphor

- large number of simple, neuron-like processing elements
- large number of weighted connections between neurons
note: the weights encode information, not symbols!
- parallel, distributed control
- emphasis on learning

McCulloch & Pitts (1943) described an artificial neuron

- inputs are either electrical impulse (1) or not (0)
- each input has a weight associated with it
- the activation function multiplies each input value by its weight
- if the sum of the weighted inputs $\geq \theta$, then the neuron fires (returns 1), else doesn't fire (returns 0)



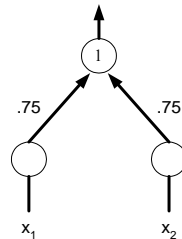
if $\sum w_i x_i \geq \theta$, output = 1
if $\sum w_i x_i < \theta$, output = 0

4

Computation via Neurons

can view an artificial neuron as a computational element

- accepts or classifies an input if the output fires



INPUT: $x_1 = 1, x_2 = 1$
 $.75 * 1 + .75 * 1 = 1.5 \geq 1 \rightarrow$ OUTPUT: 1

INPUT: $x_1 = 1, x_2 = 0$
 $.75 * 1 + .75 * 0 = .75 < 1 \rightarrow$ OUTPUT: 0

INPUT: $x_1 = 0, x_2 = 1$
 $.75 * 0 + .75 * 1 = .75 < 1 \rightarrow$ OUTPUT: 0

INPUT: $x_1 = 0, x_2 = 0$
 $.75 * 0 + .75 * 0 = 0 < 1 \rightarrow$ OUTPUT: 0

this neuron *computes* the AND function

5

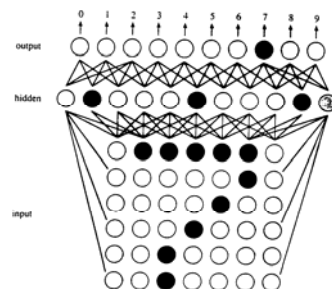
Learning Algorithm

Rosenblatt (1958) devised a learning algorithm for artificial neurons

- start with a training set (example inputs & corresponding desired outputs)
- train the network to recognize the examples in the training set (by adjusting the weights on the connections)
- once trained, the network can be applied to new examples

while this algorithm is simple and easy to execute, it doesn't always work

- there are some patterns that cannot be recognized by a single neuron
- however, by adding additional layers of neurons, the network can develop complex feature detectors (i.e., internal representations)



e.g., Optical Character Recognition (OCR)

- perhaps one hidden unit "looks for" a horizontal bar
- another hidden unit "looks for" a diagonal
- another looks for the vertical base
- the combination of specific hidden units indicates a 7

6

Neural Net Example

consider the following political poll, taken by six potential voters

- each ranked topics as to their importance, scale of 0 to 10
- voters 1-3 identified themselves as Republicans, voters 4-6 as Democrats

	Budget	Defense	Environment
voter 1	9	6	3
voter 2	8	8	6
voter 3	7	5	2
voter 4	7	4	6
voter 5	3	1	8
voter 6	6	3	7

based on survey responses, can we train a neural net to recognize Republicans and Democrats?

7

Neural Net Example

the most commonly used training algorithm for multi-layer neural networks is called *backpropagation*

- training the network can take many iterations
- the algorithm is not guaranteed to converge on a solution in all cases, but works well in practice

backpropagation simulator: <http://www.cs.ubc.ca/labs/lci/CISpace/Versi on4/neural/>

- note: inputs to network can be real values between -1.0 and 1.0
- for this example, response of 8 → input value of 0.8

generalization problem

- you can train a network to recognize a collection of patterns, but you can't be sure of what features it is using to decide
- how do you know if the trained network will behave "reasonably" on new inputs?

classic example: A military neural net was trained to identify tanks in photos. After extensive training on both positive and negative examples, it proved very effective at classification. But when tested on new photos, it failed miserably. WHY?

- various techniques are used to select training examples to help guard against these types of bad generalizations, but can't know for sure!

8

Neural Net Applications



pattern classification

- 9 of top 10 US credit card companies use Falcon
 - uses neural nets to model customer behavior, identify fraud
 - claims improvement in fraud detection of 30-70%
- scanners, tablet PCs, PDAs -- Optical Character Recognition (OCR)

prediction & financial analysis

- Merrill Lynch, Citibank, ... -- financial forecasting, investing
- Spiegel – marketing analysis, targeted catalog sales

control & optimization

- Texaco – process control of an oil refinery
- Intel – computer chip manufacturing quality control
- AT&T – echo & noise control in phone lines (filters and compensates)
- Ford engines utilize neural net chip to diagnose misfirings, reduce emissions

- ALVINN project at CMU: trained a neural net to drive a van
backpropagation network: video input, 9 hidden units, 45 outputs

9

Evol uti onary Model s



neural networks are patterned after the processes underlying brain activity

- artificial neurons are interconnected into networks
- information is sub-symbolic, stored in the strengths of the connections

genetic algorithms represent an approach to problem-solving that is patterned after the processes underlying evolution

- potential solutions to problems form a population
- better (more fit) solutions evolve through natural selection

Darwin saw "... no limit to the power of slowly and beautifully adapting each form to the most complex relations of life ..."

- through the process of introducing variations into successive generations and selectively eliminating less fit individuals, adaptations of increasing capability and diversity emerge in a population
- evolution and emergence occur in populations of embodied individuals, whose actions affect others and that, in turn, are affected by others
- selective pressures come not only from the outside, but also from the interactions between members of the population

10

Evolution & Problem-Solving



evolution slowly but surely produces populations in which individuals are suited to their environment

- the characteristics/capabilities of individuals are defined by their *chromosomes*
- those individuals that are most fit (have the best characteristics/capabilities for their environment) are more likely to survive and reproduce
- since the chromosomes of the parents are combined in the offspring, combinations of fit characteristics/capabilities are passed on
- with a small probability, mutations can also occur resulting in offspring with new characteristics/capabilities

in 1975, psychologist/computer scientist John Holland applied these principles to problem-solving → genetic algorithms

- solve a problem by starting with a population of candidate solutions
- using reproduction, mutation, and survival-of-the-fittest, evolve even better solutions

11

Genetic Algorithm (GA)



for a given problem, must define:

<i>chromosome:</i>	bit string that represents a potential solution
<i>fitness function:</i>	a measure of how good/fit a particular chromosome is
<i>reproduction scheme:</i>	combining two parent chromosomes to yield offspring
<i>mutation rate:</i>	likelihood of a random mutation in the chromosome
<i>replacement scheme:</i>	replacing old (unfit) members with new offspring
<i>termination condition:</i>	when is a solution good enough?

in general, the genetic algorithm:

```
start with an initial (usually random) population of chromosomes
while the termination condition is not met
1. evaluate the fitness of each member of the population
2. select members of the population that are most fit
3. produce the offspring of these members via reproduction & mutation
4. replace the least fit member of the population with these offspring
```

12

GA example

A thief has a bag in which to carry away the 'loot' from a robbery. The bag can hold up to 50 pounds. There are 8 items he could steal, each with a monetary value and a weight. What items should he steal to maximize his \$\$ haul?

tiara	\$5000	3 lbs
coin collection	\$2200	5 lbs
HDTV	\$2100	40 lbs
laptop	\$2000	8 lbs
silverware	\$1200	10 lbs
stereo	\$800	25 lbs
PDA	\$600	1 lb
clock	\$300	4 lbs

could try a greedy approach (take next highest value item that fits)

- based on value: tiara + coins + HDTV + PDA = 49 lbs, \$9,900

note that this collection is not optimal

- tiara + coins + laptop + silverware + PDA + clock = 31 lbs, \$11,300

13

GA example (cont.)

tiara	\$5000	3 lbs
coin collection	\$2200	5 lbs
HDTV	\$2100	40 lbs
laptop	\$2000	8 lbs
silverware	\$1200	10 lbs
stereo	\$800	25 lbs
PDA	\$600	1 lb
clock	\$300	4 lbs

chromosome: a string of 8 bits with each bit corresponding to an item

- 1 implies that the corresponding item is included; 0 implies not included
e.g., 11100000 represents (tiara + coins + HDTV)
01101000 represents (coins + HDTV + silverware)

fitness function: favor collections with higher values

- $\text{fit}(\text{chromosome}) = \text{sum of dollar amounts of items, or 0 if weight} > 50$
e.g., $\text{fit}(11100000) = 9300$
 $\text{fit}(01101000) = 0$

reproduction scheme: utilize crossover (a common technique in GA's)

- pick a random index, and swap left & right sides from parents
e.g., parents 11100000 and 01101000, pick index 4
1110|0000 and 0110|1000 yield offspring 11101000 and 01100000

14

GA example (cont.)

tiara	\$5000	3 lbs
coin collection	\$2200	5 lbs
HDTV	\$2100	40 lbs
laptop	\$2000	8 lbs
silverware	\$1200	10 lbs
stereo	\$800	25 lbs
PDA	\$600	1 lb
clock	\$300	4 lbs

visual example:
www.rennard.org/alife/english/gavgb.html

Generation 0 (randomly selected):

11100000 (fit = 9300)	01101000 (fit = 0)	11001011 (fit = 9300)
11010000 (fit = 9200)	00010100 (fit = 2800)	01001011 (fit = 4300)
11110111 (fit = 0)	10011000 (fit = 8200)	

choose fittest 4, perform crossover with possibility of mutation

111000 00 + 110010 11	→	11100011	11001001
110 10000 + 100 11000	→	11011000	10010000

Generation 1 (replacing least fit from Generation 0):

11100000 (fit = 9300)	11100011 (fit = 0)	11001011 (fit = 9300)
11010000 (fit = 9200)	11001001 (fit = 8700)	11011000 (fit = 10400)
10010000 (fit = 7000)	10011000 (fit = 8200)	

choose fittest 4, perform crossover with possibility of mutation

1101 0000 + 1100 1011	→	11011011	11001000
1110000 0 + 1101000 0	→	11100000	11010000

Generation 2 (replacing least fit from Generation 1):

11100000 (fit = 9300)	11001000 (fit = 8400)	11001011 (fit = 9300)
11010000 (fit = 9200)	11100000 (fit = 9300)	11011000 (fit = 10400)
11011011 (fit = 11300)	11010000 (fit = 9200)	

15

GA Applications

genetic algorithms for scheduling complex resources

e.g., Smart Airport Operations Center by Ascent Technology

- uses GA for logistics: assign gates, direct baggage, direct service crews, ...
- considers diverse factors such as plane maintenance schedules, crew qualifications, shift changes, locality, security sweeps, ...
- too many variables to juggle using a traditional algorithm (NP-hard)
- GA is able to evolve sub-optimal schedules, improve performance
- Ascent claims 30% increase in productivity (including SFO, Logan, Heathrow, ...)

genetic algorithms for data mining

using GA's, it is possible to build statistical predictors over large, complex sets of data

- e.g., stock market predictions, consumer trends, ...

GA's do not require a deep understanding of correlations, causality, ...

- start with a random population of predictors
- fitness is defined as the rate of correct predictions on validation data
- "evolution" favors those predictors that correctly predict the most examples

e.g., Prediction Company was founded in 1991 by astrophysicists (Farmer & Packard)

- developed software using GA's to predict the stock market – very successful

16