CSC 550: Introduction to Artificial Intelligence

Spring 2004

More neural nets

- neural net (backpropagation) examples
- associative memory
  Hopfield networks
  parallel relaxation, relaxation as search

Neural net example

consider the following political poll, taken by six potential voters

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

	Budget	**Defense**	**Crime**	**Environment**	**Social Security**
voter 1	9	6	8	3	1
voter 2	8	8	4	6	4
voter 3	7	2	4	6	3
voter 4	5	5	8	4	8
voter 5	3	1	6	8	8
voter 6	6	3	4	3	6

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

Neural net example (cont.)

utilize the neural net (backpropagation) simulator at:

http://www.cs.ubc.ca/labs/lci/CIspace/Version4/neural/

note: inputs to network can be real values between –1.0 and 1.0

- in this example, can use fractions to indicate the range of survey responses
  e.g., response of 8 → input value of 0.8

make sure you recognize the training set accurately.

- how many training cycles are needed?
- how many hidden nodes?

Neural net example (cont.)

using the neural net, try to classify the following new respondents

	Budget	Defense	Crime	Environment	Social Security
voter 1	9	6	8	3	1
voter 2	8	8	4	6	4
voter 3	7	2	4	6	3
voter 4	5	5	8	4	8
voter 5	3	1	6	8	8
voter 6	6	3	4	3	6
voter 7	10	10	10	1	1
voter 8	5	2	2	7	7
voter 9	8	3	3	3	8

More examples

neural nets for pattern recognition

- train a network to recognize picture with crosses and those without

- how big does the training set need to be?
- how many hidden nodes?

for HW 6, you will design a neural net for student advising

- ask (at least 5) questions about interests, aptitudes, lifestyle goals
- differentiate between (at least 3) different majors

- train on a set of peers (at least 2 people per major – the more the better)

- test on at least 1 person per major

Interesting variation: Hopfield nets

in addition to uses as acceptor/classifier, neural nets can be used as associative memory – Hopfield (1982)

- can store multiple patterns in the network, retrieve

interesting features

- distributed representation
  info is stored as a pattern of activations/weights
  multiple info is imprinted on the same network

- content-addressable memory
  store patterns in a network by adjusting weights
  to retrieve a pattern, specify a portion (will find a near match)

- distributed, asynchronous control
  individual processing elements behave independently

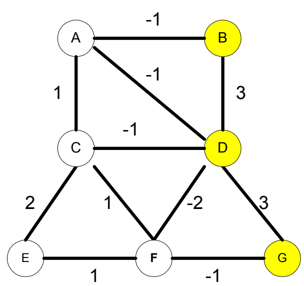
- fault tolerance
  a few processors can fail, and the network will still work

Hopfield net examples

processing units are in one of two states: *active* or *inactive*

units are connected with weighted, symmetric connections

positive weight → excitatory relation
negative weight → inhibitory relation



to imprint a pattern

- adjust the weights appropriately (no general algorithm is known, basically ad. hoc)

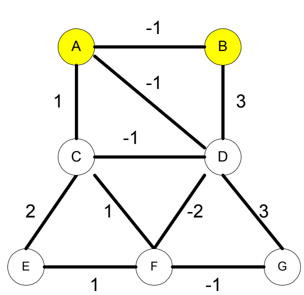
to retrieve a pattern:

- specify a partial pattern in the net
- perform *parallel relaxation* to achieve a steady state representing a near match

Parallel relaxation

parallel relaxation algorithm:

1. pick a random unit
2. sum the weights on connections to active neighbors
3. if the sum is positive → make the unit active
   if the sum is negative → make the unit inactive
4. repeat until a stable state is achieved



this Hopfield net has 4 stable states

- what are they?

- parallel relaxation will start with an initial state and converge to one of these stable states

Why does it converge?

parallel relaxation is guaranteed to converge on a stable state in a finite number of steps (i.e., node state flips)

WHY?

Define H(net) = Σ (weights connecting active nodes)

Theorem: Every step in parallel relaxation increases H(net).

If step involves making a node active, this is because the sum of weights to active neighbors > 0. Therefore, making this node active increases H(net).

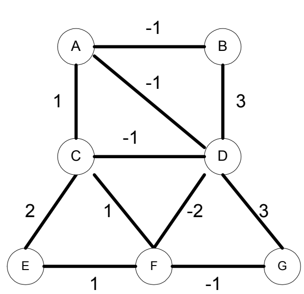
If step involves making a node inactive, this is because the sum of the weights to active neighbors < 0. Therefore, making this node active increases H(net).

Since H(net) is bounded, relaxation must eventually stop → stable state

Hopfield nets in Scheme

need to store the Hopfield network in a Scheme structure

- could be unstructured, graph = collection of edges
- could structure to make access easier




```
(define HOPFIELD-NET
  '((A (B -1) (C 1) (D -1))
    (B (A -1) (D 3))
    (C (A 1) (D -1) (E 2) (F 1))
    (D (A -1) (B 3) (C -1) (F -2) (G 3))
    (E (C 2) (F 1))
    (F (C 1) (D -2) (E 1) (G -1))
    (G (D 3) (F -1))))
```

Parallel relaxation in Scheme

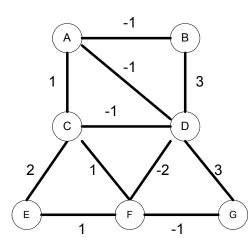
```
(define (relax active)

 (define (neighbor-sum neighbors active)
    (cond ((null? neighbors) 0)
          ((member (caar neighbors) active)
           (+ (cadar neighbors) (neighbor-sum (cdr neighbors) active)))
          (else (neighbor-sum (cdr neighbors) active))))

 (define (get-unstables net active)
    (cond ((null? net) '())
          ((and (member (caar net) active) (< (neighbor-sum (cdar net) active) 0))
           (cons (caar net) (get-unstables (cdr net) active)))
          ((and (not (member (caar net) active))
                (> (neighbor-sum (cdar net) active) 0))
           (cons (caar net) (get-unstables (cdr net) active)))
          (else (get-unstables (cdr net) active))))

 (let ((unstables (get-unstables HOPFIELD-NET active)))
    (if (null? unstables)
        active
        (let ((selected (list-ref unstables (random (length unstables)))))
          (if (member selected active)
              (relax (remove selected active))
              (relax (cons selected active)))))))
```


Relaxation examples




```
> (relax '())
()

> (relax '(b d g))
(b d g)

> (relax '(a c e f))
(a c e f)

> (relax '(b c d e g))
(b c d e g)
```

```
> (relax '(a b))
(g d b)

> (relax '(a b c e f))
(a c e f)

> (relax '(a b c d e f g))
(b c d e g)

> (relax '(a b c d))
(e g b c d)

> (relax '(d c b a))
(g d b)
```

parallel relaxation will identify stored patterns (since stable)

if you input a partial pattern, parallel relaxation will converge on a stored pattern

- what can you say about the stored pattern that is reached?
- is it in some sense the "closest" match?

Associative memory

a Hopfield net is associative memory

- patterns are stored in the network via weights

- if presented with a stored pattern, relaxation will verify its presence in the net
- if presented with a new pattern, relaxation will find a match in the net

  if unstable nodes are selected at random, can't make any claims of closeness

- ideally, we would like to find the "closest" or "best" match

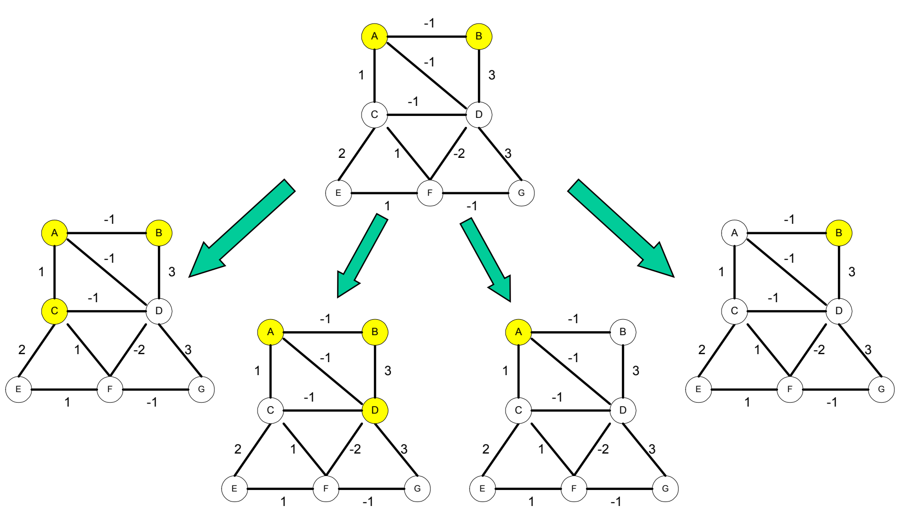
  fewest differences in active nodes?

  fewest flips between states?

Parallel relaxation as search

can view the parallel relaxation algorithm as search

- state is a list of active nodes
- moves are obtained by flipping an unstable neighbor state



Parallel relaxation using BFS

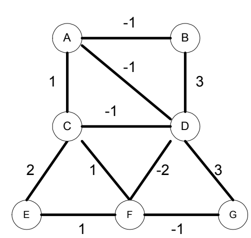
could use breadth first search (BFS) to find the pattern that is the fewest number of flips away from input pattern

```
(define (relax active)
  (car (bfs-nocycles active)))


(define (GET-MOVES active)
  (define (get-moves-help unstables)
    (cond ((null? unstables) '())
          ((member (car unstables) active)
           (cons (remove (car unstables) active)
                 (get-moves-help (cdr unstables))))
          (else (cons (cons (car unstables) active)
                      (get-moves-help (cdr unstables))))))
  (get-moves-help (get-unstables HOPFIELD-NET active)))


(define (GOAL? active)
  (null? (get-unstables HOPFIELD-NET active)))
```


Relaxation examples




```
> (relax '())
()

> (relax '(b d g))
(b d g)

> (relax '(a c e f))
(a c e f)

> (relax '(b c d e g))
(b c d e g)
```

```
> (relax '(a b))
(g d b)

> (relax '(a b c e f))
(a c e f)

> (relax '(a b c d e f g))
(b c d e g)

> (relax '(a b c d))
(g b d)

> (relax '(d c b a))
(g d b)
```

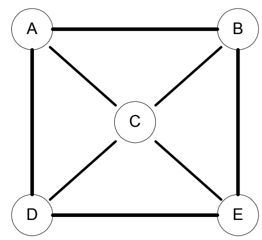
parallel relaxation will identify stored patterns (since stable)

if you input a partial pattern, parallel relaxation will converge on "closest" pattern

Another example

consider the following Hopfield network

- specify weights that would store the following patterns: AD, BE, ACE



Next week…

Emergent models of machine learning

- genetic algorithms
- cellular automata
- artificial life

Read Chapter 11

Be prepared for a quiz on

- this week's lecture (moderately thorough)
- the reading (superficial)