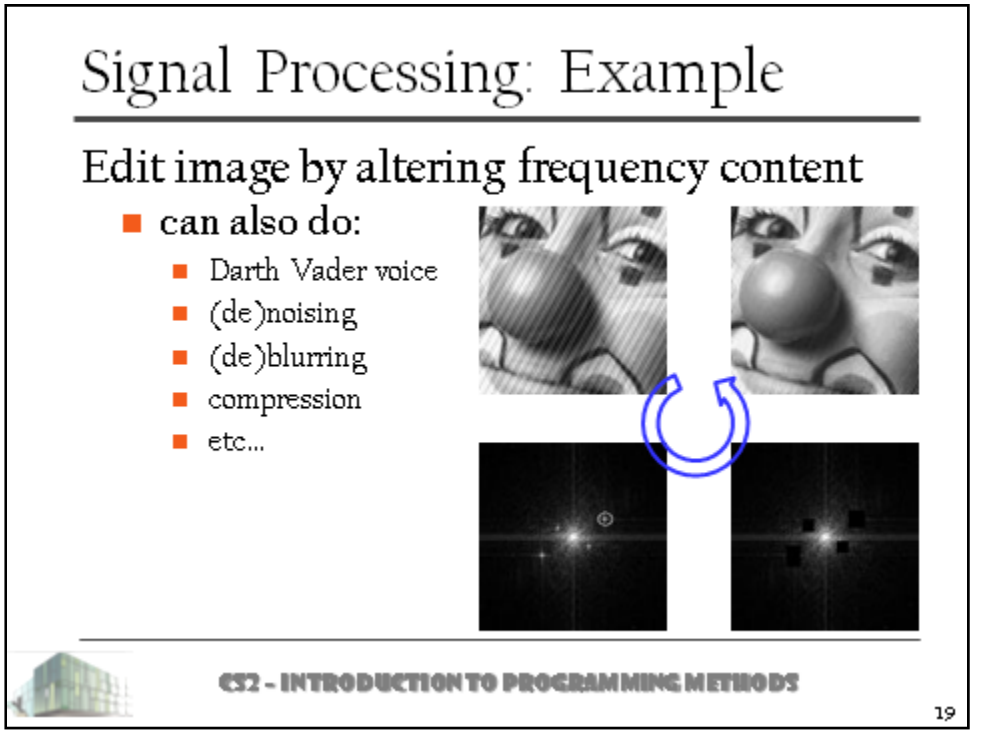
CS 2

Introduction to Programming Methods

Last Time

Numerics & FFT

- Fourier transform in O(n log n)



Artificial Intelligence

Overused, overloaded, & misused term

- intelligence is the computational part of the ability to achieve goals
- AI: science and engineering of making intelligent machines
  - especially, intelligent computer programs

Artificial Intelligence

Quick history

- coined in 1956 by McCarthy (techer!) & co
  - ideas dating back to Alan Turing in the 50's
  - logic programming used towards electronic brain
- a bit too optimistic initially...
- but interesting developments since then
  - '97: deep blue beats Kasparov, a chess grandmaster
  - '05-'07: DARPA Grand/Urban challenge
    - 100+/50+ miles of autonomous driving in desert/city
  - '11: Watson beats masters of Jeopardy!

Wide Range of AI Research

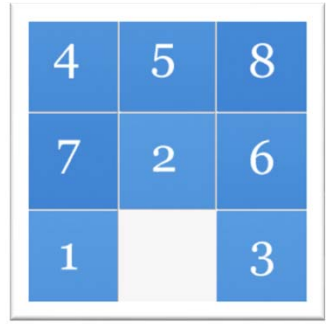
Typical subfields of AI

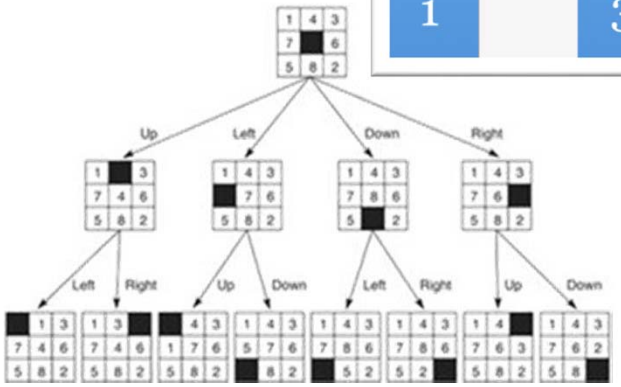
- problem solving & deduction/reasoning
  - puzzles, logic
- planning & coordination
  - autonomous agents to achieve global goal(s)
- machine learning
  - infer relationships, find patterns from data
- natural language processing
  - reading and understanding human language(s)
- knowledge representation
  - how do you encode the obvious?
    - » "the flesh is weak" does not mean that the meat is on sale

AI for Games I

For basic puzzles, easy

- search graph of possibilities
  - BFS or DFS
- small size graphs for:
  - 8-puzzle, sudoku, ...
- often impossible: too big...
  - needs heuristic—e.g., prune off branches of the search
  - example: A* algorithm
    - "best"-first search
      - » "prioritized" Dijkstra's
      - » mix btw BFS & DFS





source: Wikipedia

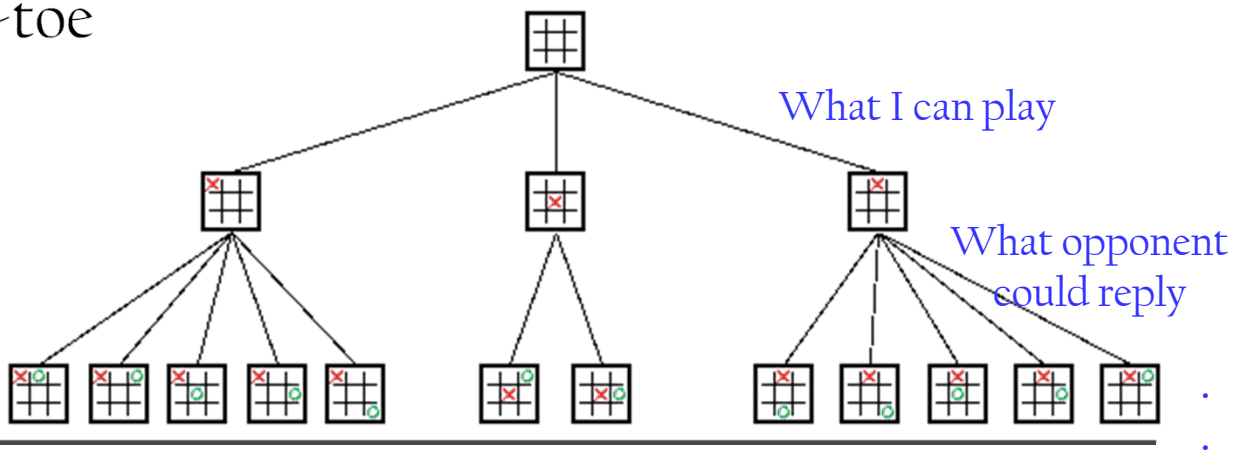
AI for Games II

2-Player Games?

- Min-max (minimax) algorithm predominant
  - always assume the adversary will play best move
  - search for best move *within this assumption* (and for n moves ahead)
    - dual recursion [*minimizing the max loss; or maximize the min gain*]
    - again, possibly with heuristics (e.g., AlphaBeta)
  - ex: tic-tac-toe

Computer's turn
- maximize outcome

Adversary's turn
- minimize outcome



AI for Games II

2-Player

- Min
  - al
  - se
  - ex

```
function minimax(node, depth, maximizingPlayer)
        if depth = 0 or node is a terminal node
        return the heuristic value of node

        if maximizingPlayer
          bestValue := -∞
          for each child of node
              v := minimax(child, depth-1, false)
              bestValue := max(bestValue, v)
          return bestValue
        else   //minimizing player
          bestValue := +∞
          for each child of node
              v := minimax(child, depth-1, true)
              bestValue := min(bestValue, v)
          return bestValue
```

nant

ve

n moves ahead)

min gain]

an play

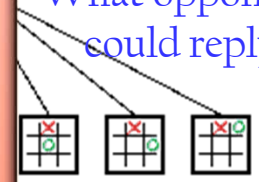
Computer's t

- maximize

Adversary's t

- minimize

What opponent could reply



AI for Games II

2-Player Games?

-]

```
function minimax(node, depth, ply)
    if node is terminal or if depth <= 0:
        return heuristic evaluation of node * ply
    α=-∞
    for child in node          // max(a,b)=-min(-a,-b)
            α=max(α,-minimax(child, depth-1,-ply))
            // find max if I am the player, min otherwise
    return α
```

ant

e

moves ahead)

n gain]

OR

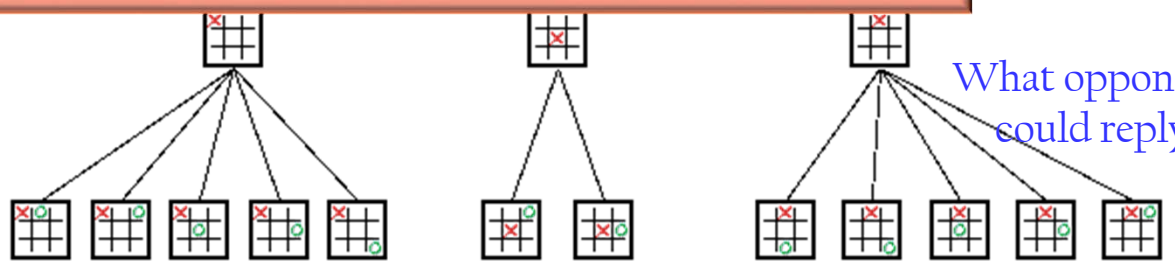
play

Compu

- maximize outcome

Adversary's turn

- minimize outcome



What opponent could reply

AI for Games III

State of the art

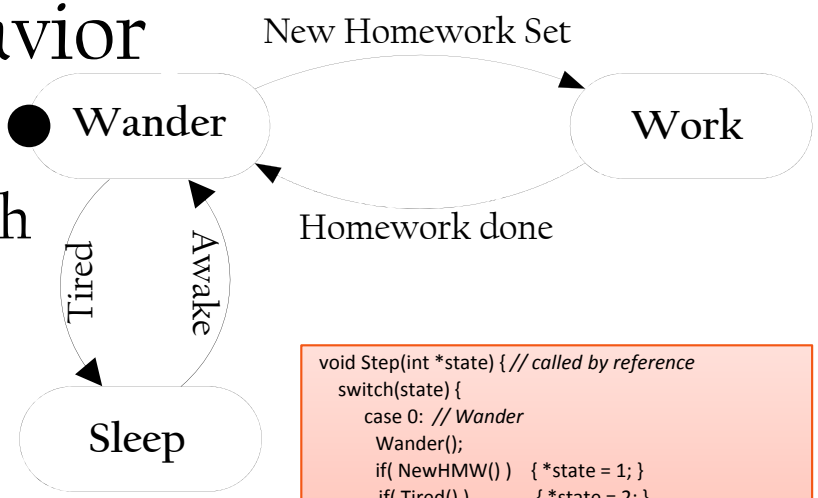
- easy: 8-puzzle, Tic-tac-toe, Connect Four.
- thought hard, now easy: Checkers, Chess, Backgammon.
  - "easy" meaning easily beating humans
- still hard: Go, Othello.

AI for Behavior

Finite state machine

- abstract model of behavior
  - starting state
  - event-based directed graph
- autonomous agents
  - e.g., in games
- automata theory
  - various types
    - (non)deterministic, pushdown, …
  - can recognize languages too!

You also need to:
- ✓ eat & have fun
- ✓ attend CS2




```
void Step(int *state) { // called by reference
   switch(state) {
      case 0:  // Wander
        Wander();
        if( NewHMW() )    { *state = 1; }
        if( Tired() )     { *state = 2; }
        break;
      case 1:  // Work
        Work();
        f( HMWDone() ) { *state = 0; }
        break;
      case 2:  // Sleep
        Sleep();
        if( Awake() )   { *state = 0; }
        break;
   }
}
```

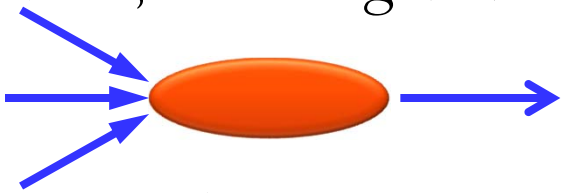
Neural Networks

Motivation: analogy to the brain

- massively interconnected neurons

Artificial neuron?

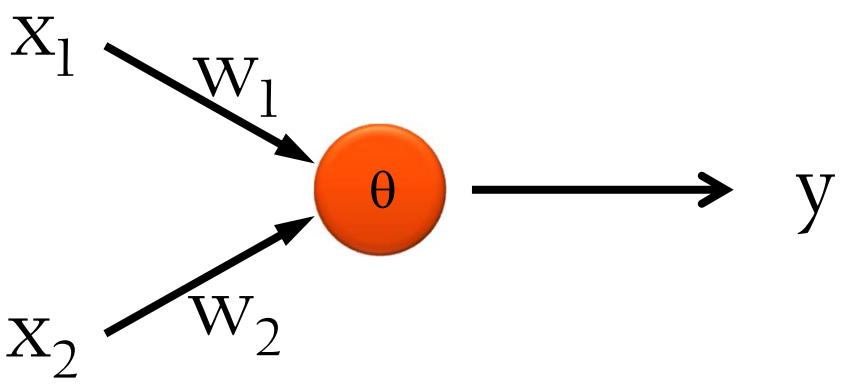
- much simpler, abstract notion
  - "processing unit", receiving and sending information



- connect them to do complex computations
  - connected through a **weighted graph**
  - parallel processing

Simple Neuron (Perceptron)

Two inputs, one output



- $y = f(w_1x_1+w_2x_2)$
- activation function f is, e.g., step function:
  - $f(x) = 1$ for $x \geq \theta$, 0 otherwise

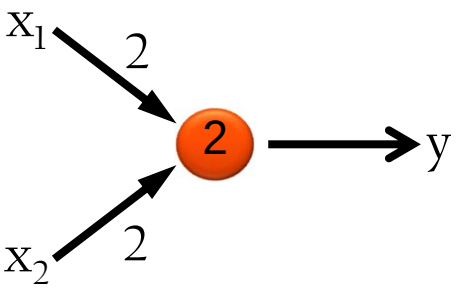
Examples of Simple Perceptrons

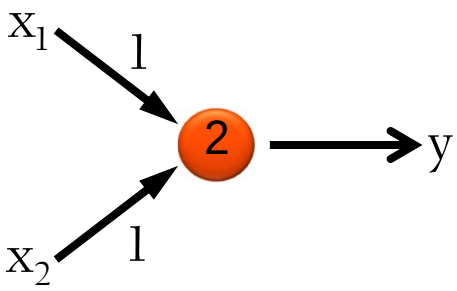
Binary gates

- {in/out}put 0 or 1

OR

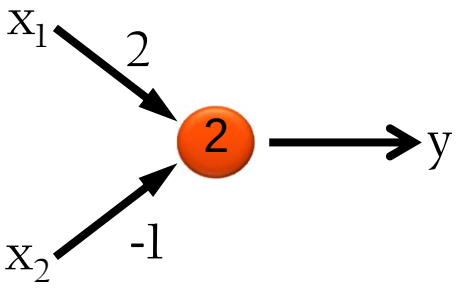
X_1	X_2	Y
0	0	**0**
1	0	**1**
0	1	**1**
1	1	**1**





AND

X_1	X_2	Y
0	0	**0**
1	0	**0**
0	1	**0**
1	1	**1**



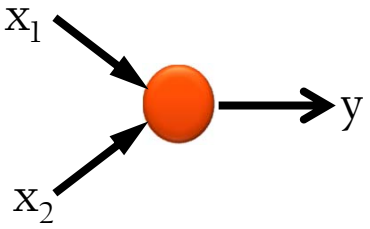
AND NOT

X_1	X_2	Y
0	0	**0**
1	0	**1**
0	1	**0**
1	1	**0**

What About XOR?

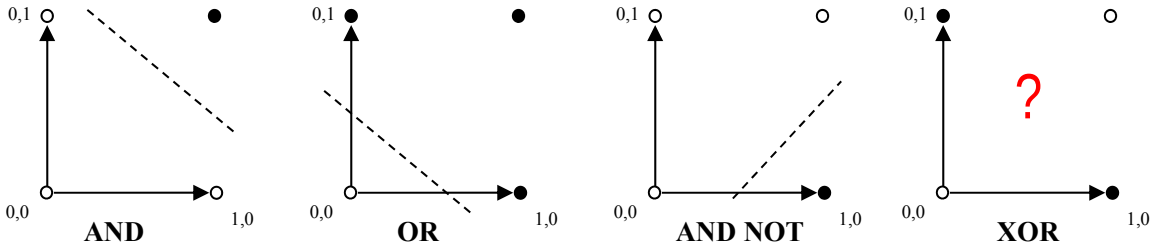
Apparent glitch...

- no weights will do
  - why? not linearly separable

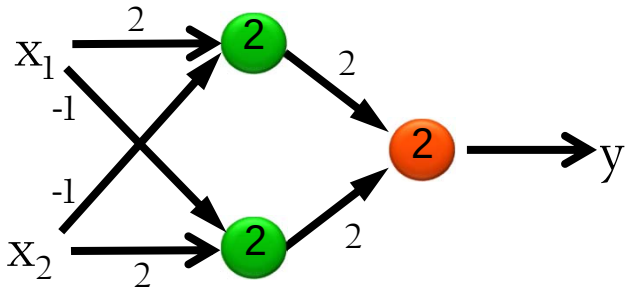


XOR

X_1	X_2	Y
0	0	**0**
1	0	**1**
0	1	**1**
1	1	**0**



- but by adding a "hidden layer", easy

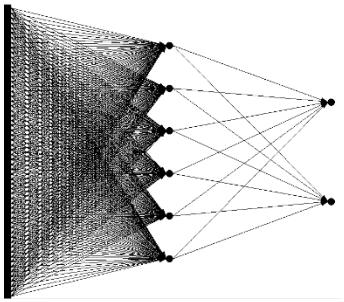


Simply implementing:
x_1 XOR x_2 =
(x_1 AND NOT x_2) OR (x_2 AND NOT x_1)

Training Multilayer Networks

Perceptrons can be trained

- tweak weights and thresholds
  - or more complex parameterized activation functions
- to "classify" inputs properly
  - feed inputs and look at outputs
  - not right? change weights and thresholds
  - and try again



- use backpropagation, reinforcement learning, evolutionary algorithms...
  - recognizing cats in videos w/ 9 layers, 1B connections...

More Neural Networks

Talked only about feed-forward

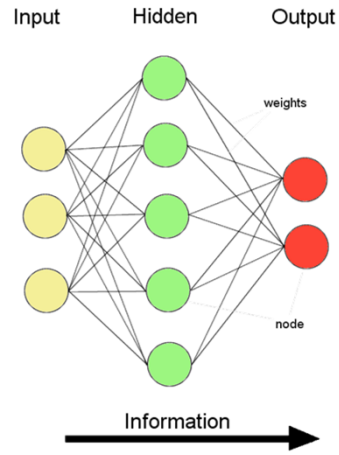
- unidirectional flow

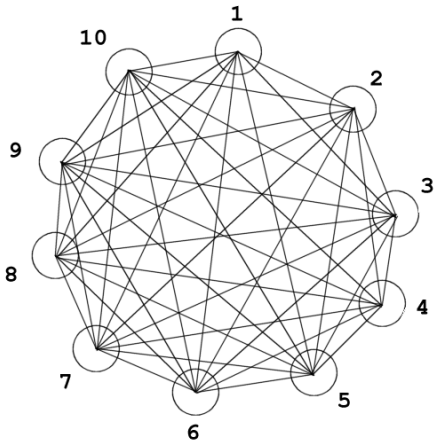
Recurrency in networks too

- "feedback" loop
- Hopfield, Elman, …
- can now handle sequences
  - "states", or memory

See also Bayesian networks

- probabilistic graphical model





FYI

John Hopfield
at Caltech '80-'97