COS470 Final Study Guide

Introduction & Agents

- What is AI?
  - Getting computers to do intelligent things
  - Medical diagnostic reasoning, planning, designing… doing the right thing and being rational
- What is AI technique?
  - The method of exploiting knowledge through search, structure of knowledge and abstraction
- **Tractability** – a problem is called intractable if the time required to solve instances of the problem grows exponentially with the size of the instances.
- Ideal agent: responsive, proactive, rational, maximizes utility, autonomous, social
- Properties of Agents: sensors, actuators, knowledge of goal, knowledge of utility, can it change via learning
- Good Agent: rational, does the right thing with the information it has
  - Should be in relation to human goals
  - Some agents know their performance measure
  - Best performance measure: relate to outcome wanted, not how agent behaves
- Rational Agent: For each possible sequence, a rational agent should select an action that is expected to maximize its performance measure, given the evidence provided by the percept sequence and whatever built in knowledge the agent has
- Task Environments
  - Properties of world
  - Circumscribed to problem and things affecting solution
  - Domain impacts task environment properties
  - Complexity of environment -> effectively nondeterministic
  - Sensor/actuator noise -> effectively nondeterministic
    - Deterministic
    - Static
    - Episodic
    - Sequential
    - Discreet
    - Continuous
- Types of problems:
  - Classification/Analysis
    - Image recognition
    - Diagnosis
    - Data mining
    - NLP
    - Sentiment analysis
  - Synthesis Construction
    - Automated planning
    - Scheduling
    - Designing buildings, car parts, new drugs

- Divide and conquer method for decomposable problems into subproblems"
  - P is O(n^2) -> O(m*2^(n/m))
- Types of agents
  - Reflex Agent – simplest has condition action rules that matches what the world is like now, atemporal, no history (condition action pairs)
  - Model Based Agent – Consider what we've done in the past, How the world evolves over time. How actions affect the world
  - Goal based agent – what would happen if I did this, compare alternatives, plan out ahead, search agent
  - Utility based agent – what do I want to happen
  - Learning Agent – learning, how to do things better by how well its doing something.

Search Space

- State-space search
    - Modeled with graph, usually digraph
    - Problem = state space + initial state + description of final state
  - State: the configuration of the world
  - State space: collection of all possible states
  - Transitions or links: events or processes in the world, Agent's, or other Agent's actions
  - Solution: path from start to goal states.
- Uninformed search
  - Complete? optimal?
  - BFS – complete, optimal as long as cost of edges are equal
  - DFS – complete, not optimal
  - IDDFS – complete, optimal
- Heuristic search & heuristics
  - Use knowledge to prioritize nodes
  - Heuristic rule: search space topology, problem domain property
  - Heuristic function maps state (local information how good, how good are the next states) (Global information, how close is this state/next state compared to goal)
  - A* - complete and optimal
- Local & online search
  - Hillclimb – local best first search
    - Not optimal, not complete
  - Backtracking HC – continuing if there exists child s' and h(s') > h(s)
    - Otherwise backtrack to previous choice point
    - Can check for repeated states
    - If can recognize goal and it's a local minimum, backtrack.
  - Simulated annealing
    - **Not optimal, complete**
    - Instead of random jumps, try suboptimal states
    - Start – probability of random moves is high (decreases over time/jumps)
    - At node take a random move
      - if better take it
      - if worse – take it some percentage of the time
  - Beam search
    - BFS branching factor is high

        - If we can reduce b that's great!
        - Search only a certain beam width (maybe **not optimal, might not be complete**)
    - Stochastic Beam Search
        - Like beam search with random elements
        - Choose I nodes at random

- Constraint-satisfaction
    - Constraint Graph
    - Nodes = variables
    - Arcs = constraints
    - Can treat CSP as search problem
        - Which variable to set
            - Pick variable with smallest remaining domain
            - Reduces branching factor = fewest alternatives to backtrack to
- Adversarial search
    - Minimax and alpha beta pruning
    - Making sure your opponent doesn't get you to make a bad move
    - Alpha(max)-beta(beta) cutoffs O(b^d) worst case
    - Best case O(b^d/2) if nodes in worst case order

Neural networks: Searching to create agents

- Search and ANNs
    - Search comes in during training – ANNs learn by adjusting weights or parameters
    - Goal: find the best value for each weight
- Neurons – a thing that holds data (activation)
    - Output number = confidence
    - Activations from one layer determine activations passed to the next layer (hidden/output)
- Training
    - Activation sequence is known through experimentation
    - Weights are set and modified with training
    - Why layers?
        - Components are processed individually
    - Activation weight = weighted sum
        - We want 0 -> 1
        - Use activation function to force weight to 0->1
            - sigmoid function
            - rectified linear unit (easier to compute no exponentiation)
    - bias
        - force neuron to activate only if the weighted sum > the bias
        - so w' = w – bias (alpha)
- Learning
    - Just find the right weights and biases
    - Gradient decent (ball rolling down a hill) gradient of the loss function (how far off we are) update weights to minimize loss.
    - Backpropagation is the algorithm that computes the gradient.

- Weights are modified along the steepest descent of the gradient to minimize error efficiently

Types of Neural Networks

- Types of neural nets
  - Convolutional
    - Great for image recognition
  - LSTM
    - Natural language processing, time series prediction
  - Multilayer perceptron
    - Basic NN

Knowledge Representation

- Five types of knowledge
  - Declarative knowledge – concepts, facts objects
  - Structural knowledge – problem solving knowledge, relationship between concepts and objects
  - Procedural knowledge – know how to do something, rules, strategies, procedures
  - Meta-knowledge – knowledge about knowledge
  - Heuristic Knowledge – expert knowledge in field
- Cycle of knowledge representation
- Knowledge versus intelligence
  - Knowledge fuels intelligence
  - Knowledge is the base of intelligent behavior
- Techniques
  - Logical Representation
    - Language with definite rules dealing with propositions
    - No ambiguity
    - Concludes on conditions and cements communication rules
    - Syntax (how we construct in logic define symbols) and semantics (rules of governing interpretation of sentences, prescribes meaning)
    - Logical reasoning, basis of programming languages
  - Semantic Network Representation
    - Alternative of predicate logic
    - Graphical networks
      - Nodes are objects, edges are relationships
      - Isa, kindof, etc
    - Natural representation… simple, more compute time, not intelligent
  - Frame Representation
    - Collection of attributes describing an entity in the world
    - Slots, facets, groups related data, easily understandable, generalization
  - Production Rules
    - Check if condition exist
      - If yes, production rule fires and action is executed
    - 3 parts:

      - Set of rules
      - Working memory
      - Recognize act cycle
    - Natural language with no learning capabilities & inefficient
  -
  -

Planning and acting

Backward Chaining

    Work backward from goal, what rules can conclude something about hypothesis
    Needs to account for uncertainty (probability), incomplete (Bayesian Nets)

Forward Chaining

– Automated planning
– Handling uncertainty
– Decision making with utilities

Machine learning

– Symbolic ML
– Going beyond MLPs (Assignment 6: Convolutional Neural Networks)
– Reinforcement learning

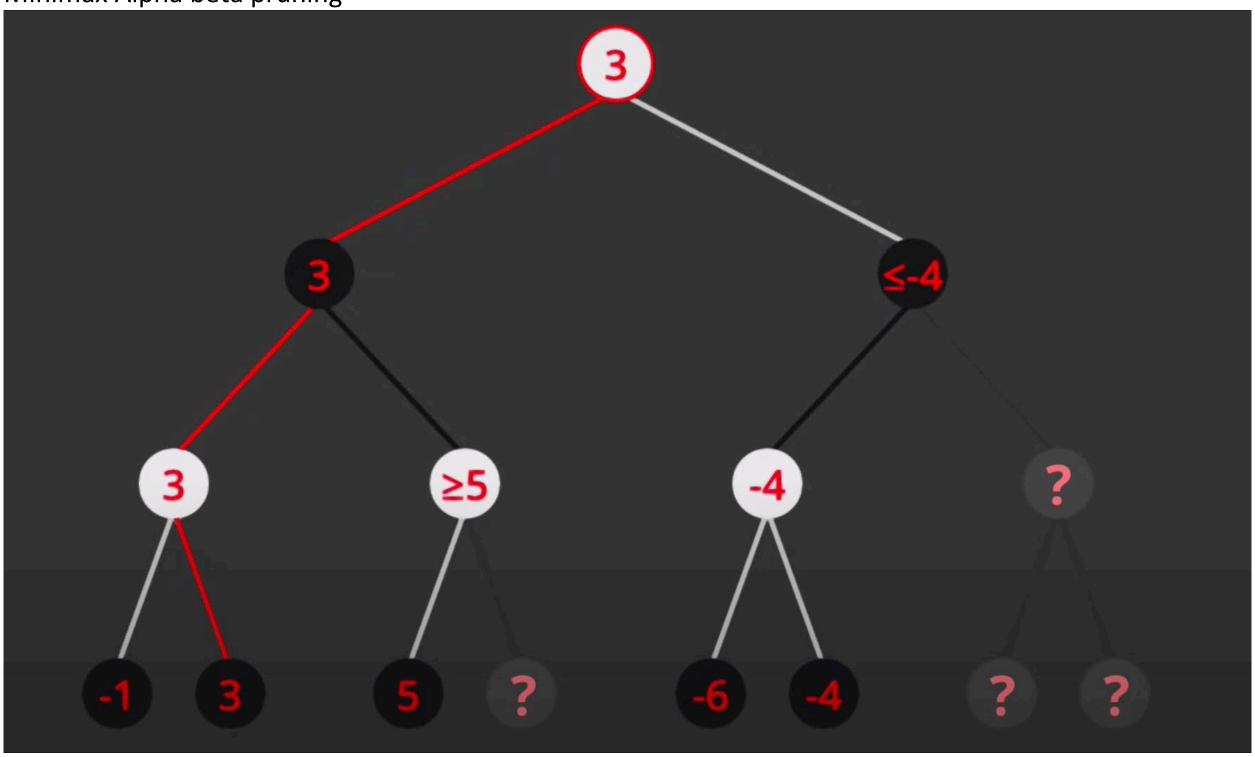
Interacting with others

– Natural language processing
– Multiagent systems

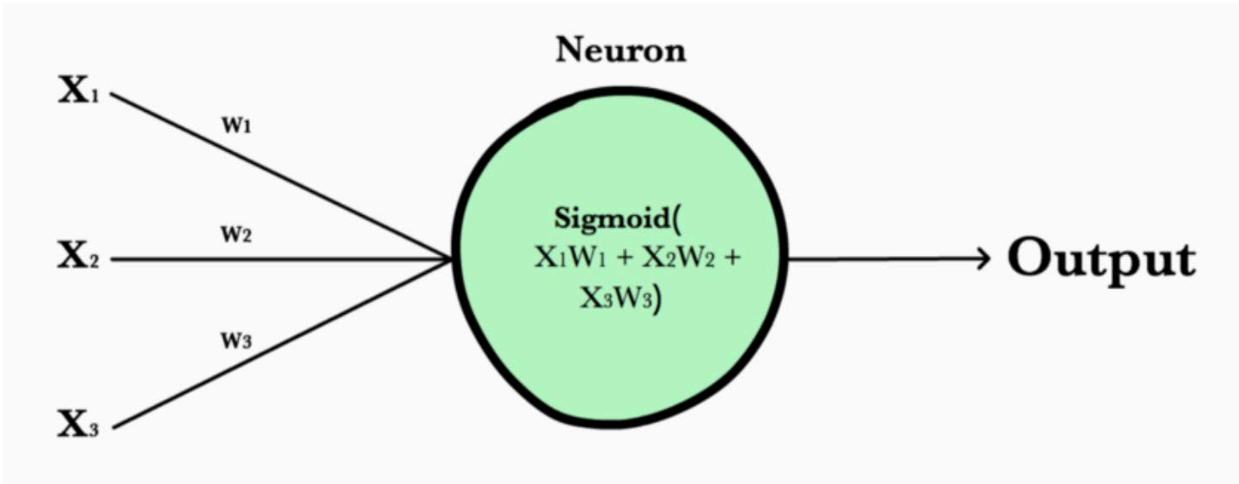
A comparison table between DFS, BFS and IDDFS

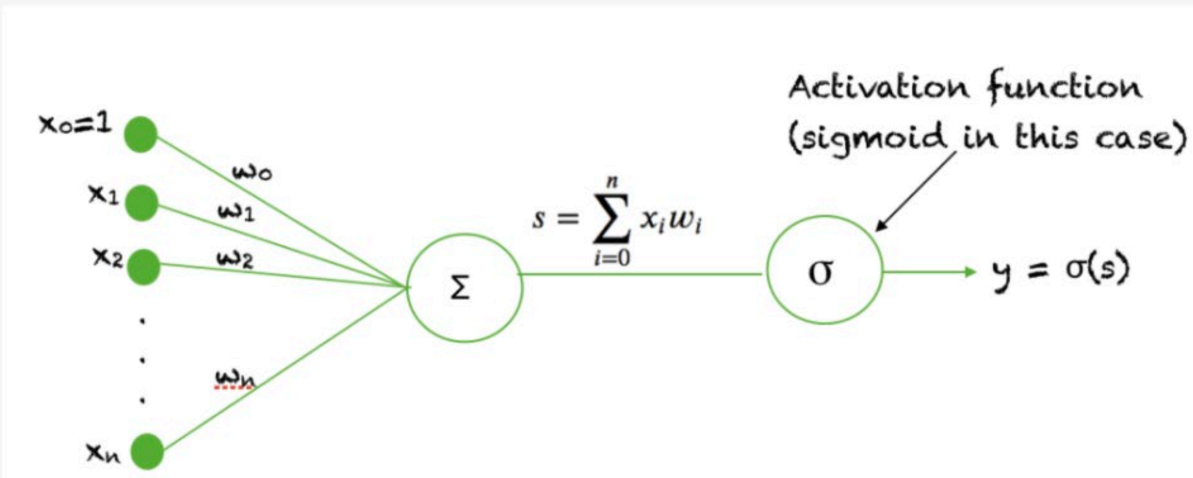
	Time Complexity	Space Complexity	When to Use ?
DFS	$O(b^d)$	$O(d)$	=> Don't care if the answer is closest to the starting vertex/root. => When graph/tree is not very big/infinite.
BFS	$O(b^d)$	$O(b^d)$	=> When space is not an issue => When we do care/want the closest answer to the root.
IDDFS	$O(b^d)$	$O(bd)$	=> You want a BFS, you don't have enough memory, and somewhat slower performance is accepted. In short, you want a BFS + DFS.

Minimax Alpha beta pruning

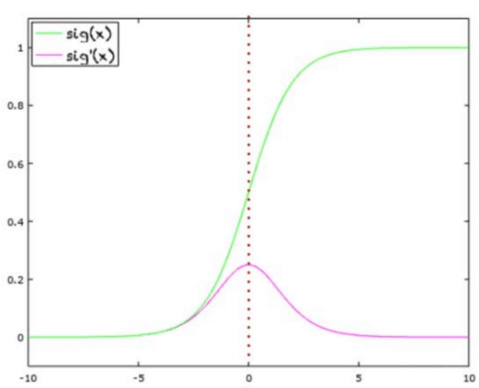


Simple Neural network





A sigmoid unit in a neural network



Plot of σ(x) and its derivate σ'(x)

Domain: (−∞, +∞)
Range: (0, +1)
σ(0) = 0.5

Other properties

$$\sigma(x) = 1 - \sigma(-x)$$

$$\sigma(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{e^x+1}$$

$$\sigma'(x) = \sigma(x)(1-\sigma(x))$$

Inference by enumeration

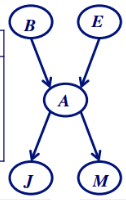
$$P(b,j,m) = \sum_e \sum_a P(b)P(e)P(a \mid b,e)P(j \mid a)P(m \mid a)$$

$$= P(b)\sum_e \sum_a P(e)P(a \mid b,e)P(j \mid a)P(m \mid a)$$

P(B)
0.001

P(E)
0.001

B	E	P(A)
t	t	0.95
t	f	0.94
f	t	0.29
f	f	0.001



A	P(J)
t	0.9
f	0.05

A	P(M)
t	0.7
f	0.01

$$\begin{array}{llllll}
B & E & A & J & M & \\
= 0.001 \times & (0.001 \times & 0.95 \times & 0.9 \times & 0.7 + & e, a \\
 & 0.001 \times & 0.05 \times & 0.05 \times & 0.01 + & e, \neg a \\
 & 0.999 \times & 0.94 \times & 0.9 \times & 0.7 + & \neg e, a \\
 & 0.999 \times & 0.06 \times & 0.05 \times & 0.01) & \neg e, \neg a
\end{array}$$