Final Exam Review

What Have We Covered?

- Agents
- States and State Spaces
- Search
  - Problem solving as search
  - Uninformed search
  - Informed search
  - Local search, genetic algorithms
- Constraint Satisfaction
- Game playing
- Probabilistic reasoning
- Bayesian networks
- Decision making under uncertainty
- Multi-Agent Systems
- Knowledge
  - Knowledge-Based Agents
  - First-Order Logic & Inference
- Planning
  - Classical
  - PO Planning
- Machine Learning
  - Decision Trees
  - Classification

What does that mean?

- Exam will mostly cover stuff since the midterm, but will draw from the first half of the class.
  - For example, how to search a plan space.
- You should expect a problem on state spaces.
- There will likely be a problem about CSPs.

Logistics

- No food
- Calculators okay (simple calculations only) *but not phones*
- Bring pencils
- Exam held in classroom

Kinds of Questions

- Definitions (a few)
- Word problems ("You are trying to find a…", "What kind of planner would you use to…")
- Problem-solving (e.g., variable elimination)
  - Esp. from homework or class examples
- Design: represent knowledge, draw a graph, assign probabilities to, FOL descriptions…

Can you *understand* and *apply* concepts and math

The Exam Itself

- Will it be as long as the midterm?
  - No.
- Will it be a similar level of difficulty?
  - Probably.
- What's the best way to study?
  - Homeworks
  - Sample problems from class
  - Slides!
- Study groups are the best thing you can do!

Unsolicited Advice

- Get some sleep beforehand.
- 4h. study + 4h. sleep < 2h. study +6h. sleep
- Really!
- You don't think as clearly when you're muzzy, and thinking clearly is more critical than cramming a few more concepts.

State Spaces

- **What information is necessary to describe all relevant aspects to solving the goal?**
- The size of a problem is usually described in terms of the possible number of states
  - Tic-Tac-Toe has about 39 states.
  - Checkers has about 1040 states.
  - Rubik's Cube has about 1019 states.
  - Chess has about 10120 states in a typical game.
  - Theorem provers may deal with an infinite space
- State space size ≈ solution difficulty

State Spaces

- What information is necessary to describe all relevant aspects to solving the goal?
- The size of a problem is usually described in terms of the possible number of states
- **Please be able to specify a state space; see Russell & Norvig pg. 70, 71, and 72 for examples.**

Search

- Solving problems by traversing states
- Generally, a state is an node, an edge is an action
  - state: location (7,6)
  - possible actions: N, S, E, W
  - Leading to new states
- What's the best thing to try first?
- Do we care about the path (sequences of moves) or just the solution found?

Types of Search

- Blind/uninformed:
  - Don't know anything about the problem domain
  - Just that there's an answer somewhere (probably)
- Informed:
  - We know something about the problem that guides search
  - E.g.: solutions for Tic-Tac-Toe are deep in tree
  - This leads to heuristics
- Local:
  - We only consider nearby states when applying heuristics

Heuristics

- Rule of thumb; a general idea what to "try first"
  - Given as a predicted cost to solution from a state
- Domain-specific
- Admissible: predicts ≤ actual cost from a state
  - That is, it's optimistic.
  - Can you come up with a state where your heuristic gives too high a number?
- Beware "holy grail" answers

α-β Pruning and Chance

- α-β Pruning for chance trees:
  - Bound the possible values a chance node can take, given current average
  - Consider whether *n* more values averaged into the first value can change that bound
- This requires known bounds on the utility function

Agents

- An **agent** is a physical or virtual entity, capable of…
  - Perceiving environment (at least partially)
  - Acting in (and on) an environment
  - Taking actions
- And which has…
  - Choices of action
  - Goals (or sometimes "tendencies")
  - Some form of **planning**, **problem-solving**, or **reacting**
- **Types:**
  - Reflex agents
  - Model-based agents
  - Goal-based agents

Know what these mean

Multi-Agent Systems

- So, a MAS is a systems with multiple agents that…
  - Communicate with one another (sometimes)
  - Affect one another (Directly or through environment)
- Possible kinds of interactions
  - Cooperate (**share** goals), or
  - Compete (have **non-mutually-satisfiable** goals), or
  - Are self-interested (have possibly **interacting** goals)

Kinds of Interaction

- Cooperative MAS
  - How can they solve problems working together?
    - Distribute the **planning** (what needs doing?)
    - Distribute the **doing** (who can do what piece?)
- Competitive or self-interested MAS:
  - Distributed rationality: Voting, auctions
  - Negotiation: Contract nets
  - Strictly adversarial interactions
- Takeaways: **types of interactions**
  - Nash equilibria, Pareto optimality, voting systems, auctions
  - Explain these terms; choose or explain a voting system

Decision-Making

- **Value function**: In decision theory, gives a ranking of the "goodness" (desirability) of states
  - E.g.: Eating Italian > pizza > burgers > sandwiches
- **Utility function** gives a **number**, not just a ranking
  - E.g.: Pizza = 19, burgers = 9, sandwiches = 5
  - Lottery outputs $5000, $100, $5

CSPs

- Defining a CSP:
  - What are the variables we are trying to assign values to?
  - What are the values they could take?
  - How do the assignments for some of them constrain assignments for others?
  - There are often several possible representations. Think carefully.
- Make sure you understand: values, variables, domains, instantiations, constraints and how to represent them!

Knowledge-Based Agents

- A knowledge-based agent needs (at least):
  - A **knowledge base** (representations of facts)
  - An **inference system** (can conclude things from facts)

Living people give tests.
Dr. M is a living person.
∴ Dr. M gives tests.

Represented / Inferred — Is this right?

- Represent **sentences** or **assertions**
- The KB: how the agent represents the environment
  - What it "perceives" and "knows"
- Inference: how the agent solves problems/reaches goals
  - Actions change the KB, and have effects derived from inference

19

Knowledge-Based Agents

- Takeaways
  - What the agent can **represent**, it **knows**
    - What is not represented or representable, it **doesn't**
  - Actions are based on knowledge of change
  - Action choices can be found through **inference**
- Be careful not to assume un-represented knowledge!

Entailment and Derivation

- **Entailment: KB ⊨ Q**
  - Q is entailed by KB if and only if there is no logically possible world in which Q is false while all the premises in KB are true.
  - Or, stated positively, Q is entailed by KB if and only if the conclusion is true in every logically possible world in which all the premises in KB are true.
- **Derivation: KB ⊢ Q**
  - We can derive Q from KB if there i[s a proof] consisting of a sequence of valid inference steps starting from the premises in KB and resulting in Q

x ⊨ y: x semantically entails y

x ⊢ y: y is provable from x

Please be able to answer questions using these words.

21

Knowledge Representations

- Propositional Logic
- First-Order Logic
- Higher-Order Logic
  - Know what it is and why you might use it
- States and Situations

Propositional Logic

- Components
  - **Logical constants**: true, false
  - **Propositional symbols**: P, Q, S, ... (**sentences**)
  - Sentences are combined by **connectives**: ∧ , ∨ , ⇒, ⇔, ¬
- Terminology
  - Worlds; assignments; truth values; validity; entailment; derivation; tautologies; inconsistent
- Rules
  - **Logical inference** is used to create new sentences that logically follow from existing sentences
  - IF/THEN and definitions
  - "If A then B" == A → B

First-Order Logic Adds…

- **Variable symbols**
  - E.g., x, y, foo
- **Constants**
  - E.g., John, MyUstairsNeighbor
- **Connectives**
  - Same as in PL: not (¬), and (∧), or (∨), implies (→), if and only if (biconditional ↔)
- **Quantifiers**
  - Universal **∀x** or **(Ax)**: for all, for each, for every
  - Existential **∃x** or **(Ex)**: there exists, there is some

First-Order Logic

- The world in FOL:
  - **Constants,** which are things with individual identities
  - **Properties** of objects that distinguish them from other objects
  - **Relations** that hold among sets of objects
  - **Functions,** which are a subset of relations where there is only one "value" for any given "input"
- Examples:
  - Objects: Students, lectures, companies, cars ...
  - Relations: Brother-of, bigger-than, outside, part-of, has-color, occurs-after, owns, visits, precedes, ...
  - Properties: blue, oval, even, large, ...
  - Functions: father-of, best-friend, second-half, one-more-than ...

A Common Error

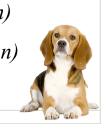
- A **complex sentence** is formed from atomic sentences connected by the logical connectives:
  $\neg P$, $P \vee Q$, $P \wedge Q$, $P \rightarrow Q$, $P \leftrightarrow Q$ where P and Q are sentences

has-a(x, Bachelors) ∧ *is-a(x, human)*

does NOT SAY everyone with a bachelors' is human

has-a(John, Bachelors) ∧ *is-a(John, human)*

has-a(Mary, Bachelors) ∧ *is-a(Mary, human)*



PL/FOL Takeaways

- Representations
  - Represent something in FOL
  - Understand and change representations
- Derive (simple) conclusions from a KB
  - Not full proofs; might need Modus Ponens
- Understand KB-agents
  - Understand how a KB changes
  - Understand how KB, agents, inference, and actions interrelate
- Use existential and universal quantification properly

Inference

- Drawing conclusions from the knowledge you have.
- Types
  - Rule Applications
  - Forward- and Backward-chaining
  - Model Checking
    - Given KB, does sentence S hold?
    - Basically **generate and test**:
      - Generate all the possible models
      - Consider the models M in which KB is TRUE
      - If ∀M S , then S is **provably true**
      - If ∀M ¬S, then S is **provably false**
      - Otherwise (∃M1 S ∧ ∃M2 ¬S): S is **satisfiable** but neither provably true or provably false

Make sure you understand these thoroughly (look at examples)

Model Checking

- Given KB, does sentence S hold?

  Quick review: What's a KB? What's a sentence?
- Basically **generate and test**:
  - Generate all the possible models
  - Consider the models M in which KB is TRUE
  - If ∀M S , then S is **provably true**
  - If ∀M ¬S, then S is **provably false**
  - Otherwise (∃M1 S ∧ ∃M2 ¬S): S is **satisfiable** but neither provably true or provably false

What does model mean?

Reasoning and Inference

- Given a formally represented world
  - Agents and their behaviors
  - Goals
  - State spaces
- What is **inference**?
- What kinds of inference can you do?
  - Forward Chaining
  - Backward Chaining

Planning

1. Classical Planning
   - Produce a fully ordered set of actions that accomplish a goal according to some test
2. Partial-order planning
   - Produce a set of sub-sequences of actions that must be accomplished in some order, with some constraints
3. Probabilistic planning
   - Same as 1 or 2, but with non-deterministic actions

Planning Problem

- Find a **sequence of actions** [operations] that achieves a **goal** when executed from the **initial world state**.
- That is, given:
  - A set of operator descriptions (possible primitive actions by the agent)
  - An initial state description
  - A goal state (description or predicate)
- Compute a **plan**, which is
  - A sequence of operator instances [**operations**]
  - Executing them in initial state → state satisfying description of goal-state

With "Situations"

- **Initial state** and **Goal state** with explicit situations

  At(Home, S_0) ∧ ¬Have(Milk, S_0) ∧ ¬Have(Bananas, S_0) ∧ ¬Have(Drill, S_0)

  (∃s) At(Home,s) ∧ Have(Milk,s) ∧ Have(Bananas,s) ∧ Have(Drill,s)
- **Operators:**

  ∀(a,s) Have(Milk,Result(a,s)) ⇔
  ((a=Buy(Milk) ∧ At(Grocery,s)) ∨
  (Have(Milk, s) ∧ a ≠ Drop(Milk)))

  ∀(a,s) Have(Drill,Result(a,s)) ⇔
  ((a=Buy(Drill) ∧ At(HardwareStore,s)) ∨
  (Have(Drill, s) ∧ a ≠ Drop(Drill)))

With Implicit Situations

- **Initial state**

  At(Home) ∧ ¬Have(Milk) ∧ ¬Have(Bananas) ∧ ¬Have(Drill)
- **Goal state**

  At(Home) ∧ Have(Milk) ∧ Have(Bananas) ∧ Have(Drill)
- **Operators:**

  Have(Milk) ⇔
  ((a=Buy(Milk) ∧ At(Grocery)) ∨ (Have(Milk) ∧ a ≠ Drop(Milk)))

  Have(Drill) ⇔
  ((a=Buy(Drill) ∧ At(HardwareStore)) ∨ (Have(Drill) ∧ a ≠ Drop(Drill)))

Planning as Inference

At(Home) ∧ ¬Have(Milk) ∧ ¬Have(Drill)

At(Home) ∧ Have(Milk) ∧ Have(Drill)

- Knowledge Base for MilkWorld
  - What do we have? Not have?
  - How does one "have" things? (2 rules recommended)
  - Where are drills sold?
  - Where is milk sold?
  - What actions do we have available?

Planning as Inference

At(Home) ∧ ¬Have(Milk) ∧ ¬Have(

At(Home) ∧ Have(Milk) ∧ Have(Dril

- Knowledge Base for MilkW
  - What do we have? Not have?
  - How does one "have" things? (
  - Where are drills sold?
  - Where is milk sold?
  - What actions do we have avail

Knowledge Base

1. We're currently home.
2. We don't have anything.
3. One has things when they are bought at *appropriate* places.
4. One has things one already has and hasn't dropped.
5. Hardware stores sell drills.
6. Groceries sell milk.
7. Our actions are:

Inference

Knowledge Base
1. We're currently home.
At(Home)
2. We don't have anything.
¬Have(Drill)
¬Have(Milk)
3. One has things when they are bought at *appropriate* places.
Have(X) ⇔
(At(Y) ∧ (Sells(X,Y) ∧ (a=Buy(X))
4. You have things you already have and haven't dropped.
(Have(X) ∧ a ≠ Drop(X)))
5. Hardware stores sell drills.
(Sells(Drill,HWS)
6. Groceries sell milk.
(Sells(Milk,GS)
7. Our actions are:
At(X) ∧ Go(Y) => At(Y) ∧ ¬At(X)
Drop(X) => ¬Have(X)
Buy(X) [defined above]

- What two things do we combine first (by number)?
  - How about 1 and 7(a)?
  - action 1 = Go(GS)
  - action 2 = Buy(Drill)
- What then changes in the knowledge base?
  - ¬At(X)
  - At(GS)

And so on…

Partial-Order Planning

- **Linear planner**
  - Plan is a **totally ordered sequence** of plan steps
- **Non-linear planner (aka partial-order planner)**
  - Plan is a set of steps with some interlocking constraints
  - E.g., S1<S2 (step S1 must come before S2)
- Partially ordered plan (POP) **refined** by either:
  - adding a new **plan step**, or
  - adding a new **constraint** to the steps already in the plan.
- A POP can be **linearized** (converted to a totally ordered plan)
  - In more than one way, typically!

Non-Linear Plan: Steps

- A non-linear plan consists of
  (1) A set of **steps** $\{S_1, S_2, S_3, S_4 \ldots\}$
  Each step has an **operator description**, **preconditions** and **post-conditions**
  (2) A set of **causal links** $\{ \ldots (S_i,C,S_j) \ldots\}$
  (One) goal of step S_i is to achieve precondition C of step S_j
  (3) A set of **ordering constraints** $\{ \ldots S_i<S_j \ldots \}$
  if step S_i must come before step S_j
- Be able to: **generate plans**, order **sequences of actions**, and know how to **resolve threats**.

Back to Milk World…

Knowledge Base
1. We're currently home.
At(Home) ← this was not true throughout!
2. We have milk and a drill.
Have(Drill)
Have(Milk)
None of these has changed.
3. One has things when they are bought at *appropriate* places.
Have(X) ⇔
(At(Y) ∧ (Sells(X,Y) ∧ (a=Buy(X))
4. You have things you already have and haven't dropped.
(Have(X) ∧ a ≠ Drop(X)))
5. Hardware stores sell drills.
(Sells(Drill,HWS)
6. Groceries sell milk.
(Sells(Milk,GS)
7. Our actions are:
At(X) ∧ Go(Y) => At(Y) ∧ ¬At(X)
Drop(X) => ¬Have(X)
Buy(X) [defined above]

- Actions:
  1. Go(GS)
  2. Buy(Milk)
  3. Go(HWS)
  4. Buy(Drill)
  5. Go(Home)

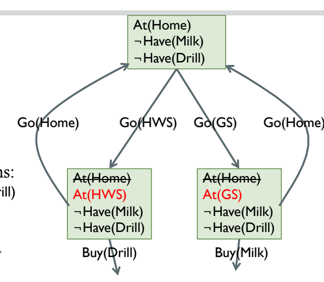
Specifying Steps and Constraints

- Go(X)
  - Preconditions: ¬At(X)
  - Postconditions: At(X)
- Buy(T)
  - Preconditions: At(Z) ^ Sells(T, Z)
  - Postconditions: Have(T)
- Causal Links: Go(X) → At(X)
- Ordering Constraints: Go(X) < At(X)

Eventually…

1. Go(GS)
2. Buy(Milk)
3. Go(HWS)
4. Buy(Drill)
5. Go(Home)

- Ordering is not strict.
- Go(HWS) preconditions:
  - ¬At(HWS) ^ ¬Have(Drill)
- So, 1<2, 3<4
- How many non-loopy paths – i.e., plans?



Machine Learning

- Decision Trees, others
- Supervised vs. Unsupervised
  - What is **classification**?
  - What is **clustering**?
  - Exploitation v. Exploration
  - K-Means, EM, and failure modes

Why Learn?

- Discover previously-unknown new things or structure
- Fill in skeletal or incomplete domain knowledge
- Build agents that can adapt to users or other agents
- Understand and improve efficiency of human learning
- Stop doing things by hand and per-domain
- When is ML appropriate? When not?

What are…

- Classification?
- Regression?
- Hypothesis?
- Hypothesis space?
- Training set and test set?
- Ockham's razor?
- Supervised/unsupervised learning?
- Rote learning?
- Induction?
- Clustering?
- Analogy?
- Discovery?
- Genetic algorithms?
- Reinforcement Learning?
- GIGO?

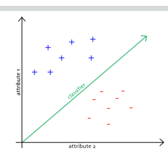
A General Model of Learning Agents

- A learning agent is composed of:
  1. Representation: how do we describe the problem space?
  2. Actor: the part of the system that actually does things.
  3. Critic: Provides the experience we learn from.
  4. Learner: the actual learning algorithm.
  5. (sometimes): Environment.
- Please make sure you can define a learning agent in these terms.

The Classification Problem

- Extrapolate from **examples** (training data) to make accurate predictions about future data
- Supervised vs. unsupervised learning
  - Learn some unknown function f(X) = Y, where
  - X is an input example
  - Y is the desired output.
  - **Supervised learning** implies we are given a **training set** of (X, Y) pairs by a "teacher"
  - **Unsupervised learning** means we are only given the Xs and some (ultimate) feedback function on our performance

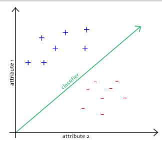
The Classification Problem (1)



- Extrapolate from **examples** (training data) to make accurate predictions about future data
- Supervised vs. unsupervised learning
  - Learn an unknown function f(X) = Y, where
  - X is an input example
  - Y is the desired output. (*f* is the..?)
  - **Supervised learning** implies we are given a **training set** of (X, Y) pairs by a "teacher"
  - **Unsupervised learning** means we are only given the Xs and some (ultimate) feedback function on our performance

The Classification Problem (2)

- Concept learning or classification (aka "induction")
  - Given a set of examples of some concept/class/category:
  - Determine if a given example is an instance of the concept (class member) or not
  - If it **is**, we call it a positive example
  - If it **is not**, it is called a negative example
  - Or we can make a probabilistic prediction (e.g., 90% sure it's a member)

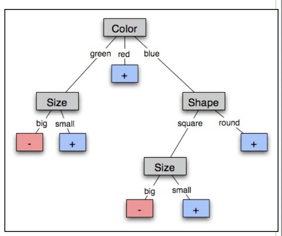


Learning Decision Trees

- Goal: Classify examples as positive or negative instances using supervised learning from a training set
- A decision tree is a tree where:
  - Each **non-leaf** node is associated with an attribute (feature)
  - Each **leaf** node has associated with it a classification (+ or -)
    - Positive and negative data points
    - That is: does it, or does it not, belong to a class?
  - Each **arc** is associated with one possible value of the attribute at the node from which the arc is directed

For Example

- Each **non-leaf** node is associated with an attribute (feature)
- Each **leaf** node is associated with a classification (+ or -)
- Each **arc** is associated with one possible value of the attribute at the node from which the arc is directed



Learning a Decision Tree

1. Select attribute to split on
2. Generate child nodes
3. Partition examples
4. Assign examples to child
5. Repeat until all training examples at node are +ve or -ve

Choosing the Best Attribute

- **Key problem:** which attribute to split on
  - Some possibilities are:
    - **Random:** Select any attribute at random
    - **Least-Values:** attribute with the smallest number of possible values
    - **Most-Values:** attribute with the largest number of possible values
    - **Max-Gain:** attribute that has the largest expected information gain– i.e., the attribute that will result in the smallest expected size of the subtrees rooted at its children
  - ID3 uses Max-Gain to select the best attribute
- Know **what** the choices are and **when** to use them

Measuring Model Quality

- How good is a model?
  - Precision/Recall
  - Training Error
  - Cross-Validation
- Overfitting: coming up with a model that is TOO specific to your training data

Naïve Bayes

- Use Bayesian modeling
- Make the simplest possible independence assumption:
  - Each attribute is independent of the values of the other attributes, given the class variable
  - In our restaurant domain: Cuisine is independent of Patrons, *given* a decision to stay (or not)

Bayesian Formulation

- The probability of class C given $F_1, ..., F_n$
  $p(C \mid F_1, ..., F_n) = p(C)\, p(F_1, ..., F_n \mid C) / P(F_1, ..., F_n)$
  $= \alpha\, p(C)\, p(F_1, ..., F_n \mid C)$
- Assume that each feature F_i is conditionally independent of the other features given the class C. Then:
  $p(C \mid F_1, ..., F_n) = \alpha\, p(C)\, \Pi_i\, p(F_i \mid C)$
- We can estimate each of these conditional probabilities from the observed counts in the training data:
  $p(F_i \mid C) = N(F_i \wedge C) / N(C)$
  - One subtlety of using the algorithm in practice: When your estimated probabilities are zero, ugly things happen
  - The fix: Add one to every count (aka "Laplacian smoothing")

Naive Bayes: Example

- p(Wait | Cuisine, Patrons, Rainy?)
  = α p(Cuisine $\wedge$ Patrons $\wedge$ Rainy? | Wait)
  = α p(Wait) p(Cuisine | Wait) p(Patrons | Wait) p(Rainy? | Wait)

naive Bayes assumption: is it reasonable?

Bayesian Learning: Bayes' Rule

- Given some **model space** (set of hypotheses h_i) and **evidence** (data D):
  - $P(h_i \mid D) = \alpha\, P(D \mid h_i)\, P(h_i)$
- We assume observations are independent of each other, given a model (hypothesis), so:
  - $P(h_i \mid D) = \alpha \prod_j P(d_j \mid h_i)\, P(h_i)$
- To predict the value of some unknown quantity X (e.g., the class label for a future observation):
  - $P(X \mid D) = \sum_i P(X \mid D, h_i)\, P(h_i \mid D) = \sum_i P(X \mid h_i)\, P(h_i \mid D)$

These are equal by our independence assumption