Cleanup, Review, and Q/A

Bookkeeping

- **Final exam, 12/20 10:30am-12:30pm, this room.**
- Review Session: this Friday, 12/16, 6pm-8pm
  - If you can't make that time, see posted slides.
- Policy on Student Exam Load: (paraphrased)
  - No more than two final exams in one day. Recommended: alternate arrangements for the **second** exam.
  - If you have an 8:00-10:00 exam **and** something after 12, tell me **soon**.
Exam Topics

- Multi-Agent Systems
- Knowledge
  - Knowledge-Based Agents
  - Knowledge Representation
  - First-Order Logic
  - Inference
- Planning
  - State spaces
  - PO Planning
  - Probabilistic Planning
- Machine Learning
  - Decision Trees
  - Classification
  - Reinforcement Learning
  - Clustering
  - Bayes’ Nets
- Applications
  - Robotics
  - Vision and Deep Learning
  - Natural Language

Knowledge Representation

- Ontologies
  - What would an ontology of “living things” look like?
    - Graphically? As a formal representation?
- Semantic Nets
  - Give an eight-node, nine-arc network about food
    - Graphically? As a formal representation?
- Types of relationships
  - Predicates: return true or false (a truth value)
  - Functions: return a value
  - Common types: is-a, part-of, kind-of, member-of
  - Keep individuals (e.g., Einstein) and groups (e.g., scientists) straight
Ontology: Living Things

- Ontologies are...
  - Taxonomic
  - Pyramidal (generally)
  - Interconnected
  - Capture semantic (meaningful) relationships

- What other meaningful relationships are here?

Ontology as Text

- Statements
  - kind-of(Fish, LivingThing)
  - kind-of(Humans, Mammals)
  - is-a(Mary, Human)
  - is-a(Mammals, Phylum)
  - disjoint(Fish, Mammals)

- Rules
  - disjoint(Fish, Mammals)
  - disjoint(Mammals, Birds) …
  - OR...
  - is-a(X,Phylum) ^ is-a(Y,Phylum) ^ (not-equal(X,Y) → disjoint(X,Y)
Semantic Networks

- The ISA (is-a) or AKO (a-kind-of) relation is often used to link instances to classes, classes to superclasses.
- Some links (e.g. hasPart) are inherited along ISA paths.
- The semantics of a semantic net can be informal or very formal.
  - Often defined at the implementation level.

Semantic Net: Food

- Give an eight-node, nine-arc network about food.
  - 8 and 9 are minimum.
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

Forward Chaining

\textbf{Knowledge Base}

1. Allergies lead to sneezing.
   \[ \text{allergies}(X) \rightarrow \text{sneeze}(X) \]
2. Cats cause allergies if allergic to cats.
   \[ \text{cat}(Y) \land \text{allergic-cats}(X) \rightarrow \text{allergies}(X) \]
3. Felix is a cat.
   \[ \text{cat}(\text{Felix}) \]
4. Lise is allergic to cats.
   \[ \text{allergic-cats}(\text{Lise}) \]

\[ \text{sneeze}(\text{Lise}) \leftarrow \text{infer truth of (query)} \]

- Find and apply relevant rules

\[ \text{cat}(Y) \land \text{allergic-cats}(X) \rightarrow \text{allergies}(X) \land \text{cat}(\text{Felix}) \]
\[ \rightarrow \text{cat}(\text{Felix}) \land \text{allergic-cats}(X) \rightarrow \text{allergies}(X) \land \text{allergic-cats}(\text{Lise}) \]
\[ \rightarrow \text{allergies}(\text{Lise}) \land \text{allergies}(X) \rightarrow \text{sneeze}(X) \]
\[ \rightarrow \text{sneeze}(\text{Lise}) \checkmark \]
Last Time: Inference

\[ \text{sneeze}(Lise) \quad \leftarrow \text{query} \]

- Backward Chaining: apply rules that end with the goal

\[ \text{allergies}(X) \rightarrow \text{sneeze}(X) \]

new query: \( \text{allergies}(Lise) \)?

\[ \text{cat}(Y) \land \text{allergic-cats}(X) \rightarrow \text{allergies}(X) \]

new query: \( \text{cat}(Y) \land \text{allergic-cats}(Lise) \)?

\[ \text{cat}(Felix) \land \text{allergic-cats}(Lise) \]

new sentence: \( \text{cat}(Felix) \land \text{allergic-cats}(Lise) \) ✓

Knowledge Base

1. Allergies lead to sneezing.
   \[ \text{allergies}(X) \rightarrow \text{sneeze}(X) \]
2. Cats cause allergies if allergic to cats.
   \[ \text{cat}(Y) \land \text{allergic-cats}(X) \rightarrow \text{allergies}(X) \]
3. Felix is a cat.
   \[ \text{cat}(Felix) \]
4. Lise is allergic to cats.
   \[ \text{allergic-cats}(Lise) \]

Uses of Inference

- Ontologies
  - Conclude new information
  - Sanity check

- Semantic Networks
  - Conclude new information
  - Build out network
  - Maintain probabilities

- Planning
Planning

- Classical Planning
- Partial-order planning
- Probabilistic planning

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
  
  $\text{At(Home, } S_0) \land \neg \text{Have(Milk, } S_0) \land \neg \text{Have(Bananas, } S_0) \land \neg \text{Have(Drill, } S_0)$
  
  $(\exists s) \text{At(Home,} s) \land \text{Have(Milk,} s) \land \text{Have(Bananas,} s) \land \text{Have(Drill,} s)$
  
  • **Operators:**
    
    $\forall (a, s) \text{ Have(Milk, Result(a, s)) } \iff$
    
    $(a= \text{Buy(Milk)} \land \text{At(Grocery,} s)) \lor$
    
    $(\text{Have(Milk,} s) \land a \neq \text{Drop(Milk))})$
    
    $\forall (a, s) \text{ Have(Drill, Result(a, s)) } \iff$
    
    $(a= \text{Buy(Drill)} \land \text{At(HardwareStore,} s)) \lor$
    
    $(\text{Have(Drill,} s) \land a \neq \text{Drop(Drill))})$

With Implicit Situations

• **Initial state**
  
  $\text{At(Home)} \land \neg \text{Have(Milk)} \land \neg \text{Have(Bananas)} \land \neg \text{Have(Drill)}$
  
• **Goal state**
  
  $\text{At(Home)} \land \text{Have(Milk)} \land \text{Have(Bananas)} \land \text{Have(Drill)}$
  
• **Operators:**
  
  $\text{Have(Milk) } \iff$
  
  $(a= \text{Buy(Milk)} \land \text{At(Grocery)}) \lor (\text{Have(Milk)} \land a \neq \text{Drop(Milk))})$
  
  $\text{Have(Drill) } \iff$
  
  $(a= \text{Buy(Drill)} \land \text{At(HardwareStore)}) \lor (\text{Have(Drill)} \land a \neq \text{Drop(Drill))})$
Planning as Inference

\[ \text{At(Home)} \land \lnot \text{Have(Milk)} \land \lnot \text{Have(Drill)} \]
\[ \text{At(Home)} \land \text{Have(Milk)} \land \text{Have(Drill)} \]

- 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

\[ \text{At(Home)} \land \lnot \text{Have(Milk)} \land \lnot \text{Have(Drill)} \]
\[ \text{At(Home)} \land \text{Have(Milk)} \land \text{Have(Drill)} \]

- 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?

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

• 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)

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=Bu(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]

And so on…

Partial-Order Planning
Partial-Order Planning

- A linear planner builds a plan as a **totally ordered sequence** of plan steps

- A non-linear planner (aka partial-order planner) builds up a plan as a set of steps with some temporal 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) by topological sorting*

* from search - R&N 223

Non-Linear Plan: Steps

- A non-linear plan consists of
  1. A set of **steps** \{S₁, S₂, S₃, S₄…\}
     - Each step has an **operator description**, **preconditions** and **post-conditions**
  2. A set of **causal links** \{ … (Sᵢ, C, Sⱼ) … \}
     - (One) goal of step Sᵢ is to achieve precondition C of step Sⱼ
  3. A set of **ordering constraints** \{ … Sᵢ < Sⱼ … \}
     - if step Sᵢ must come before step Sⱼ
Back to Milk World…

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

• **Does ordering matter?**

Knowledge Base
  1. We’re currently home.  
     \( \text{At(Home)} \leftarrow \text{this was not true throughout!} \)
  2. We have milk and a drill.  
     \( \text{Have(Drill)} \)
     \( \text{Have(Milk)} \)
  3. None of these has changed.
  4. One has things when they are bought at appropriate places.
     \( \text{Have(X)} \leftrightarrow (\text{At(Y)} \land (\text{Sells(X,Y)} \land (a=\text{Buy(X)})) \)
  5. Hardware stores sell drills.
     \( (\text{Sells(Drill,HWS)}) \)
  6. Groceries sell milk.
     \( (\text{Sells(Milk,GS)}) \)
  7. Our actions are:
     \( \text{At(X)} \land \text{Go(Y)} \Rightarrow \text{At(Y)} \land \lnot \text{At(X)} \)
     \( \text{Drop(X)} \Rightarrow \lnot \text{Have(X)} \)
     \( \text{Buy(X)} \) [defined above]

Specifying Steps and Constraints

• **Go(X)**
  • Preconditions: \( \lnot \text{At(X)} \)
  • Postconditions: \( \text{At(X)} \)

• **Buy(T)**
  • Preconditions: \( \text{At(Z)} \land \text{Sells(T, Z)} \)
  • Postconditions: \( \text{Have(T)} \)

• **Causal Links:** \( \text{Go(X)} \Rightarrow \text{At(X)} \)

• **Ordering Constraints:** \( \text{Go(X)} < \text{At(X)} \)
Eventually…

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

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

Probabilistic Planning

- Core idea: instead of actions having single effects:
  - a1: A \rightarrow B \quad a2: B \rightarrow C
- Actions have possible effects, requiring a table:
  - a1: A \rightarrow B: 80\% \quad a2: B \rightarrow C: 80\%
  - a1: A \rightarrow A: 20\% \quad a2: B \rightarrow B: 20\%
- At each plan step, propagate probabilities forward
  - Where am I now, with what probability?
In each state, the possible actions are U, D, R, and L.

The effect of U is as follows (transition model):
- With probability 0.8, the robot moves up one square (if the robot is already in the top row, then it does not move).
- With probability 0.1, the robot moves right one square (if the robot is already in the rightmost row, then it does not move).
- With probability 0.1, the robot moves left one square (if the robot is already in the leftmost row, then it does not move).

D, R, and L have similar probabilistic effects.

Where am I?
- Step 1: (1,2): 0.8  (1,1): 0.1  (2,1): 0.1
- Step 2: (1,2) → (1,3): 0.8
  
  (1,2) → (1,2): 0.1
  (1,2) → (1,2): 0.1
  (1,1) → (1,1): 0.1
  (1,1) → (1,2): 0.8
  (1,1) → (2,1): 0.1

Now: What are the odds I’m at 1,3? 1,2?
What does that mean?

- We must evaluate each sequence of actions
  - “Utility”
- Based on what we believe about events
  - But we can replan throughout
- In practice, we define (or learn) a policy.
  - I’m at X. What’s best at X?
    - And does it matter how I got there? No – this is a Markovian problem.
- Value Iteration?
  - 17.13, 17.17

Machine Learning

- Supervised vs. Unsupervised
  - What is classification?
  - What is clustering?
  - Exploitation v. Exploration
  - K-Means, EM, and failure modes
Reinforcement Learning

- **Reinforcement learning systems**
  - Learn series of actions or decisions, rather than a single decision
  - Based on feedback given at the end of the series

- A reinforcement learner has
  - A goal
  - Carries out trial-and-error search
  - Finds the best paths toward that goal

---

A typical reinforcement learning system is an active agent, interacting with its environment.

- It must balance
  - Exploration: trying different actions and sequences of actions to discover which ones work best
  - Exploitation (achievement): using sequences which have worked well so far

- Must learn **successful sequences of actions** in an uncertain environment
Clustering

- Given some instances with examples
  - But no labels!
  - Unsupervised learning — the instances do not include a “class”

- Group instances such that:
  - Examples within a group (cluster) are similar
  - Examples in different groups (cluster) are different

- According to some measure of similarity, or distance metric.
  - Finding the right features and distance metric are important!

Example
Example

• What are some two-way clusters we might get? Three way?
  - cats/dogs
  - photos/drawings
  - tan/white/striped

• What are some good features for cats/dogs?
  - Ear pointiness, tail length, …
  - Distance metric for tail length?

• What about the others?

K-Means Clustering

- Provide number of desired clusters, k.
- Randomly choose k instances as seeds.
- Form initial clusters based on these seeds.
- Calculate the centroid of each cluster.
- Iterate, repeatedly reallocating instances to closest centroids and calculating the new centroids
- Stop when clustering converges or after a fixed number of iterations.
K Means Example (K=2)

Pick seeds
Reassign clusters
Compute centroids
Reassign clusters
Compute centroids
Reassign clusters
Converged!

K-Means

• Tradeoff: more clusters (better focused clusters) and too many clusters (overfitting)
  a) What would we likely get for 3 clusters? 4?
• Results can vary based on random seed selection
  b) What if these were our starting points?
• The algorithm is sensitive to outliers
  c) Yike.

(a)  (b)  (c)
EM Summary

- Basically a probabilistic K-Means.
- Has many of same advantages and disadvantages
  - Results are easy to understand
  - Have to choose k ahead of time
- Useful in domains where we would prefer the likelihood that an instance can belong to more than one cluster
  - Natural language processing for instance