Bookkeeping

Final exam is Tuesday in class

- Final exam review today
- Project Phase II due 12/7 at 11:59 PM
- Project final paper due 12/15 at 11:59 PM

1

Exam Topics

- 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
- Ethics
- Applications
  - Robotics
  - Natural Language

2

What Will the Exam Be Like?

- Closed book, no calculator needed
- Broadly:
  - Turn a problem description into a formulation
  - Work through a problem to reach a solution
  - Demonstrate a conceptual grasp of the material
- Be able to go from concepts to/from algorithms and implementations
- Basic idea: you need to **understand the ideas** behind the material we have covered, and be **able to apply** them to solving problems.
- **Generally** easier than the homeworks (but please don't get complacent)

3

What Kind of Questions?

- T/F, multiple choice, fill in the blank
- Work through an {algorithm|solution type|problem}
- Draw something – search trees, states, Bayes nets, paths through a map, …
- Write a **short** answer to English questions
  - E.g.: "What approach would you use to solve this problem?"
  - E.g.: "We know these are independent. Why?"
- **Write a medium length essay (half a page or less)**
- No coding questions

4

What Do I Need To Do?

- Look at **homeworks**
- Look at **sample problems** in lectures
- Look at lectures' "Why?" questions
- Think (and talk about!) what we've discussed in class
- Review slides, book, background readings

5

Scoring

- Follow directions.
- You start with a perfect score that is marked down for mistakes
  - What this means: If I ask for 2 examples, and you give 3, one of which is wrong, it's -1/2
- Read carefully.
  - You have time – my exams aren't super long
    - "I didn't see the part that said…" ← ☹
- You will need to know the terminology – we will not define things you should know to understand questions

6

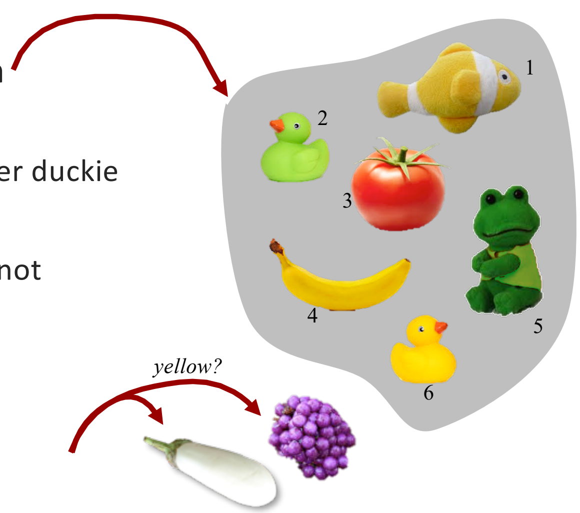
Machine Learning

- Optimize a performance criterion using example data or past experience
- Many varieties…
  - Classification
  - Regression
  - Unsupervised learning
  - Reinforcement learning
- **The Big Idea:** given some data, you learn a model of how the world works that lets you predict new data

7

Machine Learning Terminology

- What we have:
- **Data:** examples of our problem
  - Processed to produce **features**
    - Can't give a computer a rubber duckie
  - Turned into a feature **vector**
  - Sometimes labeled, sometimes not
- What we want:
- A **prediction** over new data



8

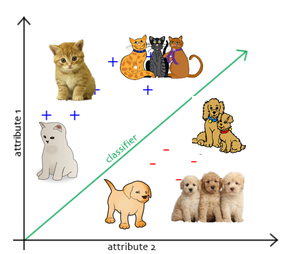
Machine Learning

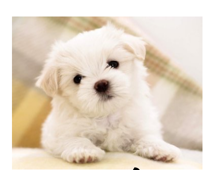
- Supervised vs. Unsupervised
  - What is classification?
  - What is clustering?
  - Exploitation v. Exploration
  - K-Means, EM, and failure modes

9

Classification

- Classification or concept learning (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: **positive example**
  - If it is not: **negative example**
  - Or we can make a probabilistic prediction (e.g., using a Bayes net)



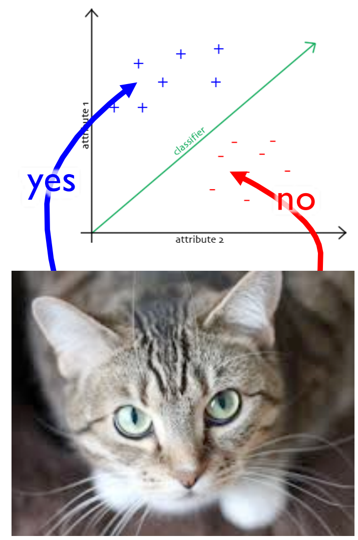


cat?

10

More on the Classification Problem

- Extrapolate from **examples** to make accurate **predictions** about future data points
  - Examples are called **training data**
- Predict into **classes**, based on attributes ("**features**")
  - Example: it has tomato sauce, cheese, and no bread. Is it pizza?
  - Example: does this image contain a cat?



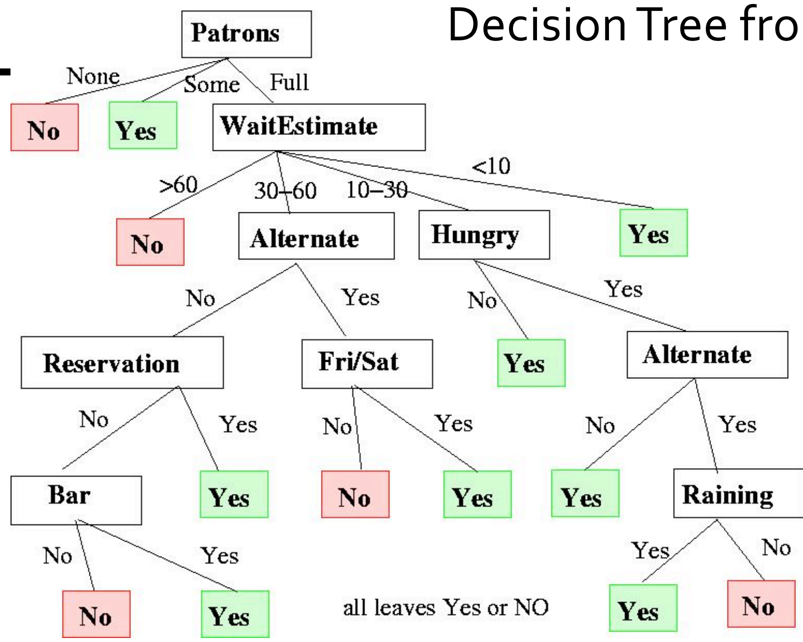
11

Decision Trees

- Goal: Build a tree to classify examples as positive or negative instances of a concept using supervised learning from a training set
- A decision tree is a tree where:
  - Each **non-leaf** node is an attribute (feature)
  - Each **leaf** node is a classification (+ or -)
    - Positive and negative data points
  - Each **arc** is one possible value of the attribute at the node from which the arc is directed
- Generalization: allow for >2 classes
  - e.g., {sell, hold, buy}

12

Decision Tree from Inspection

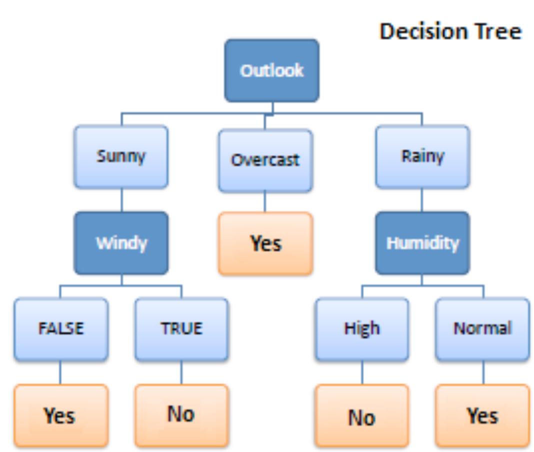


Problem from R&N, table from Dr. Manfred Kerber @ Birmingham, with thanks – www.cs.bham.ac.uk/~mmk/Teaching/AI/l3.html

13

Exercise: draw a decision tree

Outlook	Temp	Humidity	Windy	Play golf?
Rainy	Hot	High	False	No
Rainy	Hot	High	True	No
Overcast	Hot	High	False	Yes
Sunny	Mild	High	False	Yes
Sunny	Cool	Normal	False	Yes
Sunny	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Rainy	Mild	High	False	No
Rainy	Cool	Normal	False	Yes
Sunny	Mild	Normal	False	Yes
Rainy	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Sunny	Mild	High	True	No



www.saedsayad.com/decision_tree.htm

14

Choosing the Attribute to Split On

- **Information gain**: how much entropy decreases (homogeneity increases) when a dataset is split on an attribute.
  - High homogeneity → high likelihood samples will have the same class
- Constructing a decision tree is all about finding attribute that returns the highest information gain (i.e., the most homogeneous branches)

15

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

16

Representation, Reasoning, and Logic

- Point of knowledge representation is to express knowledge in a **computer usable** form
  - Needed for agents to act on it!
- **Logics** are formal languages for representing information such that conclusions can be drawn
- **Syntax** defines how symbols can be put together to form the sentences in the language
- **Semantics** define the "meaning" of sentences;
  - i.e., define truth of a sentence in a world (given an interpretation)
- Knowledge is stored in a Knowledge Base, or KB

17

PL Proofs

YOUR MISSION

- Prove that the Wumpus is in (1,3) and there is a pit in (3,1), given the observations shown and these rules:

Rules

- If there is no stench in a cell, then there is no wumpus in any adjacent cell
- If there is a stench in a cell, then there is a wumpus in some adjacent cell
- If there is no breeze in a cell, then there is no pit in any adjacent cell
- If there is a breeze in a cell, then there is a pit in some adjacent cell
- If a cell has been visited, it has neither a wumpus nor a pit
  - **FIRST** write the propositional rules for the relevant cells
  - **NEXT** write the proof steps and indicate what inference rules you used in each step

A = *Agent*
B = *Breeze*
G = *Glitter, Gold*
OK = *Safe square*
P = *Pit*
S = *Stench*
V = *Visited*
W = *Wumpus*

V12 S12 -B12	V22 -S22 -B22		
V11 -S11 -B11	V21 B21 -S21		

INFERENCE RULES

Modus Ponens
A, A → B
ergo B

And Introduction
A, B
ergo A ∧ B

And Elimination
A ∧ B
ergo A

Double Negation
¬¬A
ergo A

Unit Resolution
A ∨ B, ¬B
ergo A

Resolution
A ∨ B, ¬B ∨ C
ergo A ∨ C

18

First-Order Logic

- First-order logic (FOL) models the world in terms of
  - **Objects**, 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 ...

19

Translating English to FOL

- **Every gardener likes the sun.**
  - $\forall x \ gardener(x) \Rightarrow likes(x,Sun)$
- **You can fool some of the people all of the time.**
  - $\exists x \ \forall t \ \ person(x) \wedge time(t) \Rightarrow can\text{-}fool(x,t)$
- **You can fool all of the people some of the time.**
  - $\forall x \ \exists t \ (person(x) \Rightarrow time(t) \wedge can\text{-}fool(x,t))$
  - $\forall x \ (person(x) \Rightarrow \exists t \ (time(t) \wedge can\text{-}fool(x,t))$

  Equivalent
- **All purple mushrooms are poisonous.**
  - $\forall x \ (mushroom(x) \wedge purple(x)) \Rightarrow poisonous(x)$

20

Translating English to FOL

- **No purple mushroom is poisonous.**
  - $\neg\exists x\ purple(x) \wedge mushroom(x) \wedge poisonous(x)$
  - $\forall x\ (mushroom(x) \wedge purple(x)) \Rightarrow \neg poisonous(x)$

  Equivalent
- **There are exactly two purple mushrooms.**
  - $\exists x\ \exists y\ mushroom(x) \wedge purple(x) \wedge mushroom(y) \wedge purple(y)\ \wedge \neg(x=y) \wedge \forall z\ (mushroom(z) \wedge purple(z)) \Rightarrow ((x=z) \vee (y=z))$
- **Mary is not tall.**
  - $\neg tall(Mary)$
- **X is above Y iff X is on directly on top of Y or there is a pile of one or more other objects directly on top of one another starting with X and ending with Y.**
  - $\forall x\ \forall y\ above(x,y) \Leftrightarrow (on(x,y) \vee\ \exists z\ (on(x,z) \wedge above(z,y)))$

21

Translating FOL to English

1. $\forall x\ (bitter(x) \vee sweet(x))$
2. $\forall x\ (bitter(x)) \vee \forall x\ (sweet(x))$
3. $\exists x\ \forall y\ (loves(y,x))$
4. $\neg\exists x\ \neg\exists y\ (loves(y,x))$
5. $\exists x\ (noisy(x)) \Rightarrow \forall y(annoyed(y))$
6. $\forall x\ (frog(x) \Rightarrow green(x))$
7. $\forall x\ (frog(x) \Rightarrow \neg green(x))$
8. $\neg\exists x\ (frog(x) \wedge green(x))$
9. $\exists x\ (frog(x) \wedge \neg green(x))$
10. $\exists x\ (mech.(x) \wedge likes(x, Bob))$
11. $\exists x\ (mech.(x) \wedge likes(x, x))$
12. $\forall x\ (mech.(x) \Rightarrow likes(x, Bob))$
13. $\exists x\ \forall y\ (mech(x) \wedge nurse(y) \Rightarrow likes(x, y))$
14. $\exists x(mech(x) \wedge \forall y(nurse(y) \Rightarrow likes(y, x))$

Exercises: disi.unitn.it/~bernardi/Courses/LSNL/Slides/fl1.pdf

22

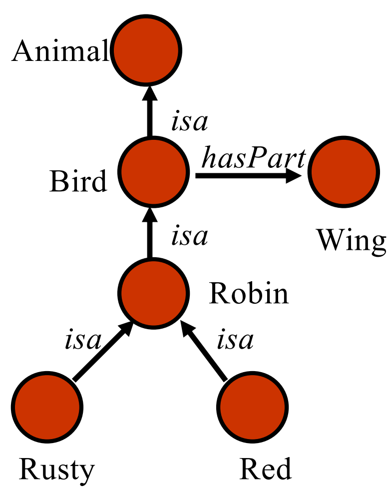
Proving Things: 5 Methods

- Inference by Enumeration
  - List all possible true worlds, check the truth value of a sentence
  - Complete but exponential in time
- Proof by Natural Deduction
  - Writing proofs from laws (e.g., modus ponens)
- Forward Chaining
- Backward Chaining
- Resolution Refutation
  - Show KB ⊨ α by proving that KB ∧ ¬α is unsatisifiable, i.e., deducing False from KB ∧ ¬α

23

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



24

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

26

Forward Chaining

sneeze(Lise) ← infer truth of (query)

- Find and apply relevant rules

variable binding

cat(Y) ∧ allergic-cats(X) → allergies(X) ∧ cat(Felix)

→

cat(Felix) ∧ allergic-cats(X) → allergies(X) ∧ allergic-cats(Lise)

→

allergies(Lise) ∧ allergies(X) → sneeze(X)

add new sentence to KB

→

sneeze(Lise) ✓

Knowledge Base

1. Allergies lead to sneezing.
   allergies(X) → sneeze(X)
2. Cats cause allergies if allergic to cats.
   cat(Y) ∧ allergic-cats(X) → allergies(X)
3. Felix is a cat.
   cat(Felix)
4. Lise is allergic to cats.
   allergic-cats(Lise)

27

Backward Chaining

sneeze(Lise) ← query

- Backward Chaining: apply rules that end with the goal

variable binding

allergies(X) → sneeze(X) + sneeze(Lise)
new query: allergies(Lise)?

cat(Y) ∧ allergic-cats(X) → allergies(X) + allergies(Lise)
new query: cat(Y) ∧ allergic-cats(Lise)?

cat(Felix) + cat(Y) ∧ allergic-cats(Lise)
new sentence: cat(Felix) ∧ allergic-cats(Lise) ✓

Knowledge Base
1. Allergies lead to sneezing.
allergies(X) → sneeze(X)
2. Cats cause allergies if allergic to cats.
cat(Y) ∧ allergic-cats(X) → allergies(X)
3. Felix is a cat.
cat(Felix)
4. Lise is allergic to cats.
allergic-cats(Lise)

28

Uses of Inference

- Ontologies
  - Conclude new information
  - Sanity check
- Semantic Networks
  - Conclude new information
  - Build out network
  - Maintain probabilities
- Planning

29

Planning

- Classical Planning
- Partial-order planning
- Probabilistic planning

30

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

31

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)))

32

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)))

33

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?

34

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 r
  - Where are drills sold?
  - Where is milk sold?
  - What actions do we have available?

<u>Knowledge Base</u>
1. We're currently home.

2. We don't have anything.

3. One has things when they are bought at *appropriate* places.

4. You have things you already have and haven't dropped.

5. Hardware stores sell drills.

6. Groceries sell milk.

7. Our actions are:

35

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)

And so on…

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]

36

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*

37

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**

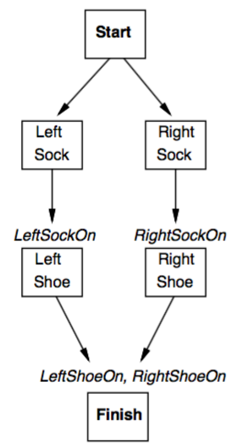
  (2) A set of **causal links** $\{ \ldots (S_i,C,S_j) \ldots\}$

  (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

Partial Order Plan:



38

Back to Milk World…

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

<u>Knowledge Base</u>

1. We're currently home.
   At(Home) ← this was not true throughout!
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]

39

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)

40

POP Constraints and Search Heuristics

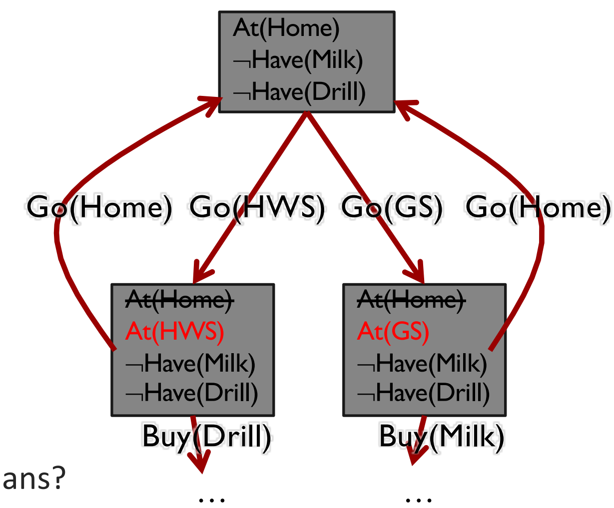
- Only add steps that reach a not-yet-achieved precondition
- Use a least-commitment approach:
  - Don't order steps unless they need to be ordered
- Honor causal links $S_1 \rightarrow S_2$ that **protect** a condition c:
  - Never add an intervening step S_3 that violates c
  - If a parallel action **threatens** c (i.e., has the effect of negating or clobbering c), resolve that threat by adding ordering links:
    - Order S_3 before S_1 (**demotion**)
    - Order S_3 after S_2 (**promotion**)

41

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?



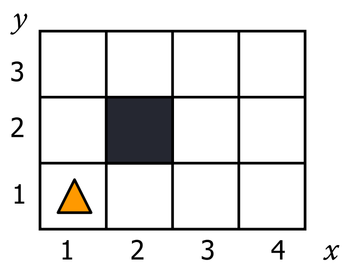
42

Probabilistic Planning

- Core idea: instead of actions having single effects:
  - a1: A → B   a2: B → C
- Actions have possible effect**s**, requiring a table:
  - a1: A → B: 80%   a2: B → C: 80%
  - a1: A → A: 20%   a2: B → B: 20%
- At each plan step, propagate probabilities forward
  - Where am I now, **with what probability?**

43

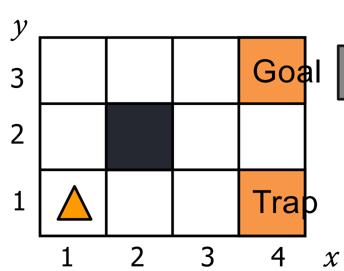
Transition Model in Practice



- 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

44

Transition Model in Practice



Plan: U, U, R, R, R

- 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?

- In each state, possible actions are U, D, R, and L
- The transition model) of U is:
  - up: 0.8
  - left: 0.1
  - right: 0.1
- D, R, and L have similar probabilistic effects

45

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

46

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

47

Reinforcement Learning

- 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

48

Learning States and Actions

- A typical approach is:
- At state S choose, some action A ← How?
- Taking us to new State S_1
  - If S_1 has a positive value: increase value of A at S.
  - If S_1 has a negative value: decrease value of A at S.
  - If S_1 is new, initial value is unknown: value of A unchanged.
- One complete learning pass or **trial** eventually gets to a terminal, deterministic state. (E.g., “win” or “lose”)
- Repeat until? Convergence? Some performance level?

49

Exploration vs. Exploitation

- Problem with naïve reinforcement learning:
  - What action to take?
  - **Best apparent action, based on learning to date** } Exploitation
    - Greedy strategy
    - Often prematurely converges to a suboptimal policy!
  - **Random (or unknown) action** } Exploration
    - Will cover entire state space
    - Very expensive and slow to learn!
    - When to stop being random?
- Balance exploration (try random actions) with exploitation (use best action so far)

50

Clustering

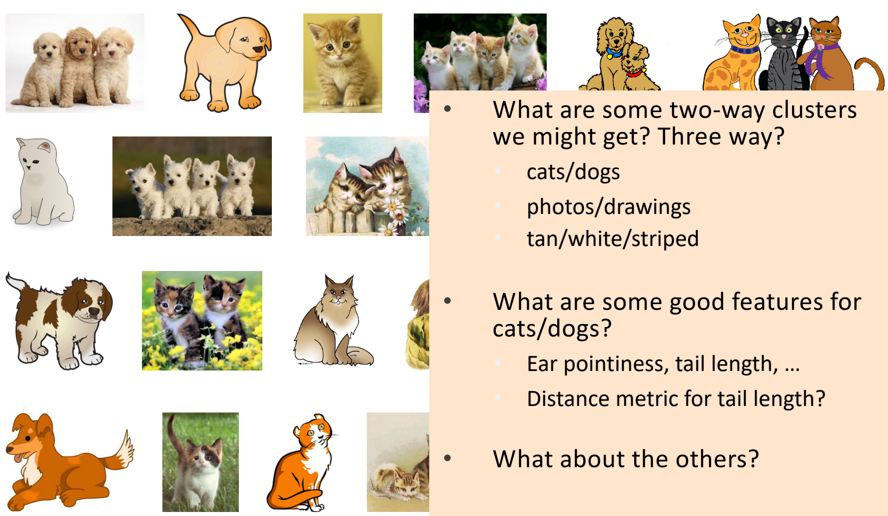
- Given some instances with examples
  - But no labels!
  - Unsupervised learning — the instances do not include a "class"
- Group instances such that:
  - Examples within a group (cluster) are similar
  - Examples in different groups (cluster) are different
- According to some *measure of similarity*, or **distance metric**.
  - Finding the right **features** and **distance metric** are important!

51

Example

52

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?

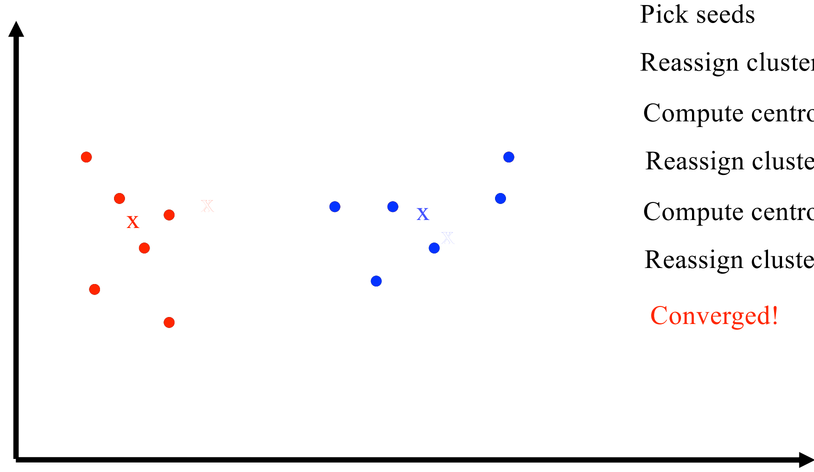
53

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.

54

K Means Example (K=2)



Pick seeds

Reassign clusters

Compute centroids

Reassign clusters

Compute centroids

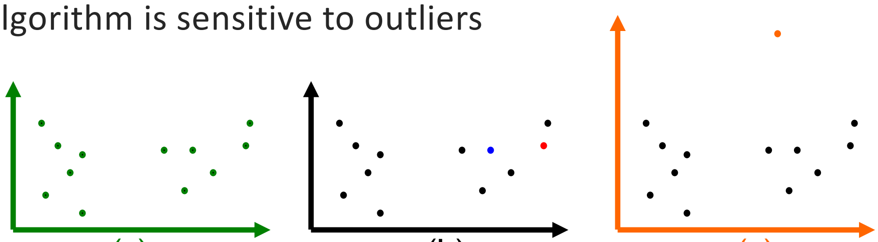
Reassign clusters

Converged!

55

K-Means

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



(a) (b) (c)

56

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

57

Ethics

- There are a lot of open areas in AI+Ethics
  - **Physical danger (e.g., self-driving cars)**
  - **Bias in artificial intelligence and machine learning**
  - **LLMs and guardrails**
  - AI and jobs
  - AI and art
  - Deepfakes
  - Privacy and surveillance

What are the risks? The potential value? Sources of problems?

- Our discussion was necessarily fairly high level, but be able to speak intelligently about any of these topics, especially AI/ML

58

Robotics

- What is a robot really, and what are they used for?
- What are sensors, actuators, effectors? What are degrees of freedom? What's a robot's "belief state"?
- How does AI tie into a robotic system?
- What about robots in human spaces?

59

Where is AI needed in robotics?

- Sensing:
  - Interpreting incoming information
    - Machine vision, signal processing
    - Language understanding
- Actuation:
  - What to do with manipulators and how
    - Motion planning and path planning
- Control:
  - Managing large search spaces and complexity
    - Accelerating masses produce vibration, elastic deformations in links.
    - Torques, stresses on end actuator
    - Feedback loops
- Firmware and software:
  - Especially with more intelligent approaches!

So, basically everywhere

60

Natural Language

- What is the history of NLP in robotics?
- Large Language Models and how they work
- Terminology
  - Semantics, syntax, morphology, phonetics, …
  - Disambiguation, reference resolution
- Applied NLP
  - What can we use NLP for?

61