Final Exam Review

What Have We Covered?

Before the Midterm:

- 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

Since the midterm:

- Knowledge
  - Knowledge-Based Agents
  - First-Order Logic & Inference
- Planning
  - Classical
  - PO Planning
  - Probabilistic Planning
- Machine Learning
  - Decision Trees
  - Classification
  - Reinforcement Learning
  - Clustering
  - Bayes' Nets
- Ethics

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.

The Exam Itself

- Logistics
  - No food
  - Calculators okay (simple calculations only) *but not phones*
- What kinds of questions?
  - Definitions
  - 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 sample problems in class
  - 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, possibly a little easier.
- What's the best way to study?
  - Homeworks
  - Sample problems from class
  - Re-skim readings

State Spaces: Review

- **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: Review

- 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.**

Agents: Review

- 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

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. — Represented
Dr. M is a living person. — Represented
∴ … — Inferred

- 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

11

Knowledge-Based Agents

- Takeaways
  - What the agent can **represent**, it **knows**
  - Actions are based on knowledge of change
  - Actions can be found through **inference**

Entailment and Derivation

- **Entailment: KB ⊨ Q**
  - Q is entailed by KB (a set of premises or assumptions) 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 is 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.

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
- **Connectives**
  - Same as in PL: not (¬), and (∧), or (∨), implies (→), if and only if (biconditional ↔)
- **Quantifiers**
  - Universal **∀x** or **(Ax)**
  - Existential **∃x** or **(Ex)**

First-Order Logic

- The world in FOL:
  - **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 ...

A Common Error

- A **complex sentence** is formed from atomic sentences connected by the logical connectives:
  ¬P, P∨Q, P∧Q, P→Q, P↔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 } Make sure you understand these thoroughly (look at examples)
  - 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

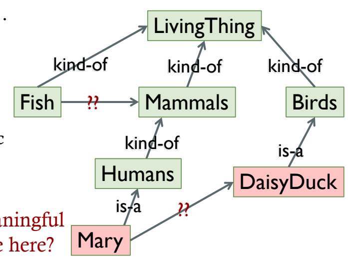
Model Checking

- 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** (What does model mean?)
  - If ∀M ¬S, then S is **provably false**
  - Otherwise (∃M1 S ∧ ∃M2 ¬S): S is **satisfiable** but neither provably true or provably false

Ontology: Living Things

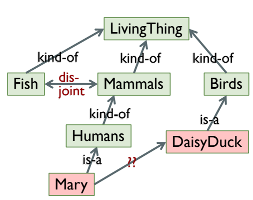
- Ontologies are…
  - Taxonomic
  - Pyramidal (generally)
  - Interconnected
  - Capture semantic (meaningful) relationships
- What other meaningful relationships are here?



Ontology as Text

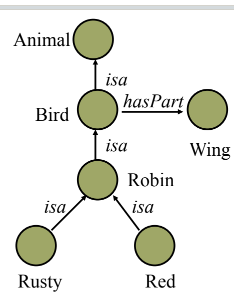
- Statements
  - kind-of(Fish, LivingThing)
  - kind-of(Humans, Mammals)
  - is-a(Mary, Human)
  - is-a(Mammals, Phylum)
  - disjoint(Fish, Mammals)
- Rules
  - disjoint(Fish, Mammals)
  - disjoint(Mammals, Birds) …

    OR…
  - is-a(X,Phylum) ^ is-a(Y,Phylum) ^ (not-equal(X,Y) → disjoint(X,Y)



Semantic Networks

- is-a or kind-of relation is often used to link instances to classes, classes to superclasses
  - But it's also usually necessary to define domain-specific relationships
- Some relationships (e.g., isa) are inherited
- Semantics can be informal or very formal



Semantic Net Example: 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

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

- 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]

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** { … (S_i,C,S_j) …}

  (One) goal of step S_i is to achieve precondition C of step S_j

  (3) A set of **ordering constraints** { … $S_i<S_j$ … }

  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…

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

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]

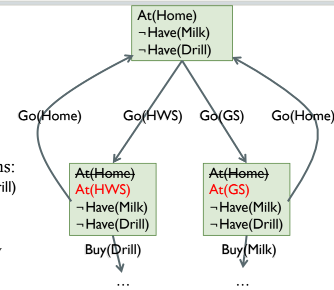
Specifying Steps and Constraints

- 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)

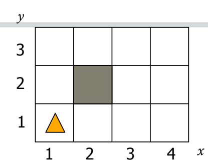
- Ordering is not strict.
- Go(HWS) preconditions:
  - ¬At(HWS) ^ ¬Have(Drill)
- So, 1<2, 3<4
- How many non-loopy paths – i.e., plans?



Probabilistic Planning

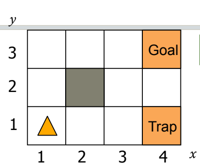
- Core idea: instead of actions having single effects:
  - a1: A → B   a2: B → C
- Actions have possible effects, requiring a table:
  - a1: A → B: 80%   a2: B → C: 80%
  - a1: A → A: 20%   a2: B → B: 20%
- At each plan step, propagate probabilities forward
  - Where am I now, **with what probability?**

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

Transition Model in Practice



- 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

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?

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?
  - **Practice problems: R&N 17.13, 17.17**

This lecture went poorly, so just make sure you understand the core ideas, get the idea of the main algorithms, and can solve problems like the one on the previous slide. ☺

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?

44

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?

45

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.

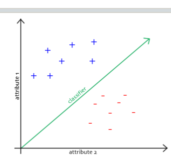
46

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

47

The Classification Problem (1)

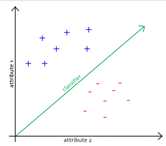


- Extrapolate from **examples** (training data) to make accurate predictions about future data
- Supervised vs. unsupervised learning
  - Learn an unknown function f(X) = Y, where
  - X is an input example
  - Y is the desired output. (*f* is the..?)
  - **Supervised learning** implies we are given a **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

48

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)

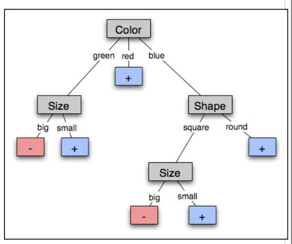


Learning Decision Trees

- 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

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
- Unsupervised Learning

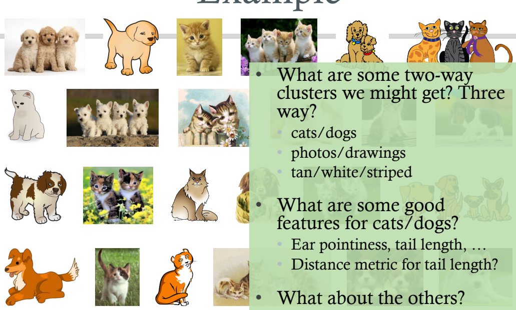
Reinforcement Learning

- 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

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



- 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

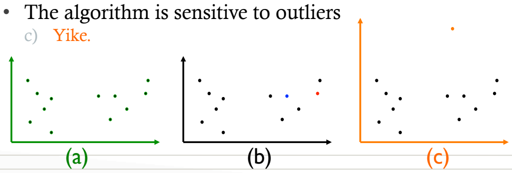
- Provide number of desired clusters, k.
- Randomly choose k instances as seeds.
- Form initial clusters based on these seeds.
- Calculate the centroid of each cluster.
- Iterate, repeatedly reallocating instances to closest centroids and calculating the new centroids
- Stop when clustering converges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

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

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.



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

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)

62

Bayesian Formulation

- The probability of class C given $F_1, \ldots, F_n$
  $p(C \mid F_1, \ldots, F_n) = p(C)\, p(F_1, \ldots, F_n \mid C) / P(F_1, \ldots, F_n)$
  $= \alpha\, p(C)\, p(F_1, \ldots, F_n \mid C)$
- Assume that each feature F_i is conditionally independent of the other features given the class C. Then:
  $p(C \mid F_1, \ldots, F_n) = \alpha\, p(C)\, \Pi_i\, p(F_i \mid C)$
- 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")

63

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?

64

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

66

Bayesian Learning: Understand…

- **MLE (Maximum Likelihood Estimate)**
  - Assume all hypotheses are equally likely *a priori*;
  - best hypothesis maximizes the **likelihood** (i.e., probability of data given hypothesis)
  - **Maximize p(D | h_i)**
- **BMA (Bayesian Model Averaging)**
  - Don't just choose one hypothesis; instead, make predictions based on the weighted average of all hypotheses (or some set of best hypotheses)
- **MAP (Maximum *A Posteriori*) hypothesis**
  - Choose hypothesis with highest *a posteriori* probability, given data
  - **Maximize p(h_i | D)**
  - Generally easier than Bayesian learning
  - Closer to Bayesian prediction as more data arrives
- **Parameter Estimation**

67

Ethics in AI

- May be on exam
  - Same ground rules: use commonly understood ideas of "wrong," and don't get into the meta-questions
  - Understand the cases we talked about in class
- Be able to…
  - Identify the different options in an ethical case
  - Discuss which you think is best **and why**
  - Identify or defend a consistent point of view
  - Answer factual questions
- You'll be graded on whether your answers are supported and internally consistent, not whether I agree!