Final Exam

- Tuesday, December 11, 5:30pm-8pm
- This classroom (I assume)
- Cumulative, but emphasizes material post-midterm.
- Study old homework assignments, including programming projects.
- I will give you some practice problems for reinforcement learning since we didn't have a homework assignment on that.

Topics

- State space search
- Constraint satisfaction problems
- Adversarial search
- Probability
- Bayes nets
- Naïve Bayes
- Hypothesis choosing
- Markov chains & Hidden Markov models
- Reinforcement learning

Models, Reasoning, and Learning

- A model is a way of representing a problem (think data structure)
  - States (used in search trees, game trees, CSPs), Bayes nets (incl. Naïve Bayes), Markov chains, HMMs, MDPs.

Models, Reasoning, and Learning

- A reasoning algorithm draws conclusions or makes inferences based on data in a model.
  - Search (uniform cost search, greedy best first search, A*, minimax, alpha-beta pruning), CSP search, AC-3, exact inference algorithm for Bayes nets, ML & MAP, inference algorithm in Markov chains, forward algorithm, backward algorithm, Viterbi algorithm, value iteration, Q-learning.

Models, Reasoning, and Learning

- A learning algorithm tries to deduce the structure or parameters of the model itself from auxiliary data.
  - Training a Naïve Bayes classifier.

State Space Search

- Represent a partial solution to the problem as a “state.”
- Use an algorithms to find the “best” path through the state space.
- Pros: Often easy to formulate the model: states and actions.
- Cons: Often slow with a mediocre heuristic, state space is often too big to store explicitly in memory.

CSPs

- Represent a partial solution to the problem as a "state," using a set of variables assigned to values.
- No notion of "actions;" move between states by assigning or re-assigning variables.
- Pros: No need for heuristic for each problem; one algorithm can solve any CSP!
- Cons: Still can be slow (uses backtracking search), can get stuck in local maxima.

Adversarial Search

- Still uses a “state,” only we aren’t usually interested in the entire “best” path, just the “best” next move.
- Can use minimax and alpha-beta pruning to search the game tree.
- Pros: “The” model & algorithm(s) for 2-player games.
- Cons: Can’t represent entire tree in memory, very slow for large games, still requires heuristics for deep trees.

Probability

- Way of representing uncertainty in a model or algorithm.
- Many modern AI techniques based on rules of probability.
  - Often can give better results than heuristic approaches, where any numbers used may not be derived from any mathematical rules.
- Algorithms for ML and MAP hypothesis choosing.

Bayesian Networks

- A representation of the conditional independences that hold among a set of random variables.
- Lets you compute the probability of any event, given any observation (setting) of a set of other variables.
- Pros: Simple representation, grounded in math
- Cons: Hard to learn, exact inference can be slow, scientist must develop set of appropriate variables.

Naïve Bayes

- Particular kind of Bayes net with nice properties.
- Assumes conditional independence among all pieces of evidence/features/data.
- Useful where you need to choose a hypothesis, but don't necessarily care about the actual posterior probability (often the conditional independence assumption messes that up).
- Pros: Very simple, parameters of model easy to learn, fast algorithms for inference and learning.
- Cons: Can make gross oversimplifications, probability estimates may not be very accurate (though hypothesis often is).

Markov chains and HMMs

- Another type of Bayes net!
- Makes Markov assumption: probability distribution of next state depends only upon current state. (Sometimes called Markov property)
- Used for sequential or temporal data.
- Pros: Only model so far that takes time into account, efficient algorithms for inference and learning.
- Cons: Again, might be overly simplistic for some applications.

Reinforcement learning

- Model: MDP
- Inference: Bellman equations
- Learning: Value iteration, Q-learning, lots of others…
- Pros: Simple representation, good for cases where you'll be in the same state many times.
- Cons: Slooooooooooow, must be able to get experience by repeating same situations over and over.

Comparison of models

- Some model-algorithm combinations can solve “any” problem:
  - State-space search/AI*, CSPs/backtracking
- But often they either require
  - lots of engineering on the human’s part
  - and/or intractable on real-world problems

Comparison of models

- Other model-algorithm combinations solve problems very quickly:
  - e.g., Naïve Bayes and HMMs
- But they only work for problems that fit the model well.