Factored State Spaces
Converting POMDPs to MDPs

In a POMDP:
• Action + observation updates beliefs
• Value is a function of beliefs.

Instead we can view this as an MDP where:
• There is a state for every possible belief.
• Beliefs are probabilities, so we have a continuum.
  • There are infinitely many belief-states.
• Taking an action transitions to another belief-state.
  • Observations are random, so this transition is random.
Value Iteration in POMDPs

Value iteration in a finite MDP:
1. Initialize each state’s value to 0.
2. Compute the greedy policy for each state.
3. Update the value of each state based on this policy.
4. Goto step 2; repeat until converged.

In a POMDP, there are infinitely many states.
• We can’t loop through them.
• Value is a piecewise-linear function of belief.
  • We can do value iteration over a finite set of linear functions.
    • For a description of the algorithms continue reading pomdp.org.
POMDP value iteration is impractical

In principle, we can iterate over a finite set of linear functions to update values.

In practice, this gets out of hand very quickly.

This example is from a 2-state, 2-action MDP, computing values at horizon 3.
What can we do instead?

• Approximation
  • Don’t find optimal values for every state.
  • Instead solve for pretty-good values for groups of states.

• Online planning
  • Don’t bother devising a complete policy.
  • Instead come up with a pretty-good policy for the short term, and re-plan in the future.

• Reinforcement learning
  • Combines aspects of approximation and online planning.
  • We’ll focus on RL after spring break.
What is the running time of Value Iteration on MDPs?

repeat until convergence:
  (update values table)
  ...
  ...
  for each state:
    (update state’s value)
    ...
    ...
  for each action:
    (find the best action)
    ...
    ...
  for each next state:
    (update expected value)
    ...
    ...
Running time of Value Iteration

Value iteration isn’t *fast*, but if the state space is small, it’s manageable.

The problem is state spaces can be exponentially larger than they *seem*.

If the agent’s state has $k$ variables with domains $D_1, \ldots, D_k$ to keep track of, there can be $\prod_{i=1}^{k}|D_i|$ distinct states.

Even in the best case (binary variables), this is exponential ($2^k$ states).
How big is the state space?

In gridworld, we can track the state space with two variables *row* and *column*. This is quite manageable.

How big is the state space for realistic MDPs?
- Search and rescue robot
- Mars rover

What variables do these agents need to keep track of?
Mars rover state space

Some possible state space variables:

- position: $x, y, z$ (large domains)
- environmental factors
  - sunlight
  - wind
  - temperature
- robot controls
  - battery level
- arm position
- solar panel angle
- drill operation
- mission objectives
  - current experiment
  - samples being analyzed
- etc.

This state space is gigantic!
MDPs vs. State Space Search

• In state space search, we might also have state variables and factored state spaces.

Why didn’t we worry about the fact that the state space could be exponentially large then?
Handling huge state spaces

• Full value iteration is implausible.

• We could try to reason about the factored state space.
  • Plan in terms of variables instead of states.
  • One could spend an entire month on this topic (we won’t).

• Approximation
• Online planning
• Reinforcement learning
Online Planning

Key idea: plan for one state at a time.

• We may be able to come up with a good action for the current state without solving the entire MDP.

• We can plan ahead some, but we know that we’re going to stop and think about plans again later.

Where have we seen something like this before?
Thinking about online planning.

How can we use ideas we’ve already seen to help with online planning?

Heuristics?

Iterative deepening?

Monte Carlo simulations?

Other ideas?