RL 11: POMDPs: Partially Observable Markov Decision Processes

Michael Herrmann

University of Edinburgh, School of Informatics

25/02/2014

A trace through an MDP

Environment:	You are in state 65. You have 4 possible actions.
Agent:	I'll take action 2.
Environment:	You received a reinforcement of 7 units. You are now in state 15. You have 3 possible actions.
Agent:	I'll take action 1.
Environment:	You received a reinforcement of -4 units. You are now in state 16. You have 2 possible actions.
Agent:	I'll take action 2.
Environment:	You received a reinforcement of 8 units. You are now in state 15. You have 3 possible actions.
⋮	⋮

How is this different for a POMDP?

Types of Planning Problems

	State	Action Model
Classical Planning	observable	deterministic accurate
MDP	observable	stochastic
POMDP	partially observable	stochastic

Two types of uncertainty

- Stochasticity: Only parameters of a distribution can be known
- Partial observability:
  - There is an underlying deterministic process that in principle can be inferred
  - This deterministic process may govern the parameters of a stochastic process

Background: COMDPs vs POMDPs

Same: set of states and actions, transitions and immediate rewards.

Different:

- Previously: (regular) discrete MDPs $\longrightarrow$ completely observable (COMDPs)
- Value iteration algorithm for COMDPs gives a value per state
- accurate state information is available
- Markovian

- POMDPs are also discrete MDPs.
- No certainty about the current state
- How represent values and actions?
- Probabilitistic observations replace explicit state information
- Observation model needed: Bayesian estimation of states
- Taking into account information about previous states: Non-Markovian for states, but Markovian in terms of belief states.

Reminder (Bellmann for MDPs)

Bellman optimality equation

$$V^*(s) = \max_a \sum_{s'} P^a_{ss'} \left(R^a_{ss'} + \gamma V^* \left(s' \right) \right)$$

Usually we can assume that $R^a_{ss'} = r\left(s, a\right)$ (i.e. independent on next state)

$$V^*(s) = \max_a \left(r\left(s, a\right) + \gamma \sum_{s'} P^a_{ss'} V^* \left(s' \right) \right)$$

For continuous states x and actions u this becomes

$$V^*(x) = \max_u \left(r\left(x, u\right) + \gamma \int p\left(x' | s, u\right) V^* \left(x' \right) dx' \right)$$

Value Iteration (for MDPs)

Bellman optimality equation:

$$V^*(x) = \max_u \left(r(x,u) + \gamma \int p\left(x'|s,u\right) V^*\left(x'\right) dx' \right)$$

Value iteration

$$V_t(x) = \max_u \left(r(x,u) + \gamma \int p\left(x'|s,u\right) V_{t-1}\left(x'\right) dx' \right)$$

Note: $V_{t-1}(x')$ is the previous iterate of the value function which is evaluated at the next state x'

Initialisation by

$$V_1(x) = \max_u r(x,u)$$

POMDPs: Beliefs instead of state information

- Generalisation of MDPs: POMDPs
- State is not observable: the agent relies on beliefs about its state
- Define the belief b of the agent about the its state and formulate a POMDP with a value function over a belief space:

$$V_t(b) = \max_u \left(r(b,u) + \int V_{t+1}(b')\, p(b'|u,b)\, db' \right)$$

- Belief is a posterior distribution over states (see below)

$$b'(s') \propto \Omega(o \mid s', a) \sum_{s \in S} T(s' \mid s, a) b(s)$$

Consequences

- As beliefs are probability distributions POMDPs involve functions on (continuous) probability distributions
- Belief spaces are generally large
- Because of continuity, belief spaces have a relatively simple structure
- If we assume
  - finite state space
  - finite action spaces
  - finite horizons
- Then we can represent value functions by piecewise linear functions

POMDP: Formal definition and belief propogation

(S, A, O, T, Ω, R), where

S is a set of states,

A is a set of actions,

O is a set of observations,

T is a set of conditional transition probabilities,

Ω is a set of conditional observation probabilities,

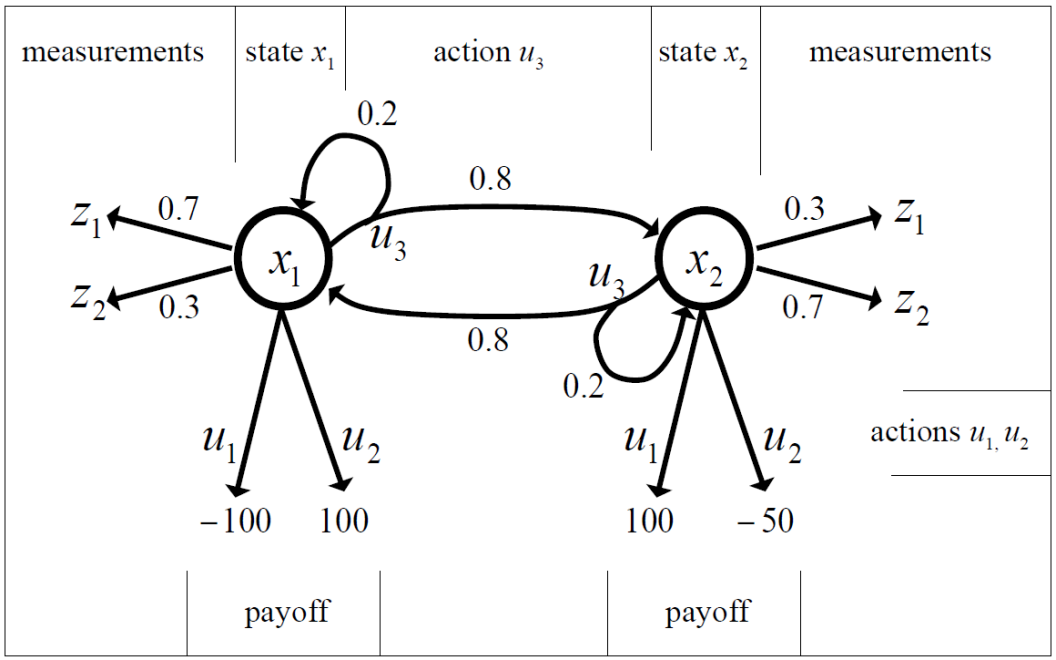
$R : A \times S \to \mathbb{R}$ is the reward function

Bayesian belief propogation:

$$b'(s') = \frac{\Omega(o \mid s', a) \sum_{s \in S} T(s' \mid s, a) b(s)}{\sum_{s'' \in S} \Omega(o \mid s'', a) \sum_{s \in S} T(s'' \mid s, a) b(s)}$$

An Illustrative Example

- Actions u_1, u_2 are terminal actions
- Action u_3 is a sensing action that may lead to a state transition.
- The horizon is finite
- $\gamma = 1$.
- Reward is a probability-weighted integral



$$r(x_1, u_1) = -100 \qquad r(x_2, u_1) = +100$$
$$r(x_1, u_2) = +100 \qquad r(x_2, u_2) = -50$$
$$r(x_1, u_3) = -1 \qquad r(x_2, u_3) = -1$$
$$p(x_1'|x_1, u_3) = 0.2 \qquad p(x_2'|x_1, u_3) = 0.8$$
$$p(x_1'|x_2, u_3) = 0.8 \qquad p(x_2'|x_2, u_3) = 0.2$$
$$p(z_1|x_1) = 0.7 \qquad p(z_2|x_1) = 0.3$$
$$p(z_1|x_2) = 0.3 \qquad p(z_2|x_2) = 0.7$$

Credit assignment in POMDPs

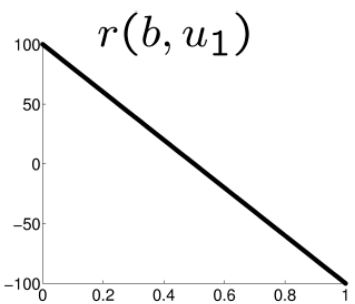
If we are totally certain that we are in state x_1 and execute action u_1, we receive a reward of -100

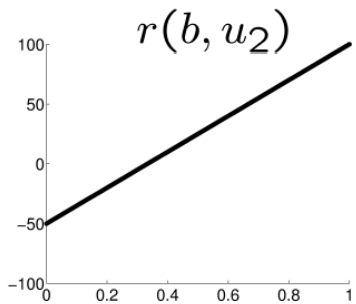
If, on the other hand, we definitely know that we are in x_2 and execute u_1, the reward is +100.

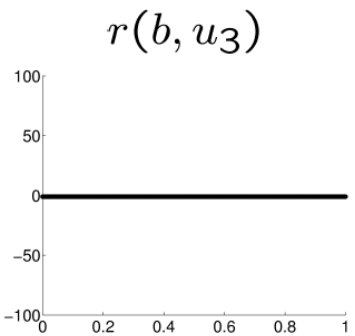
If the 'probability' that we are in x_1 is p_1 and in x_2 is $p_2 = 1 - p_1$ then the belief b can be parametrised by p_1.

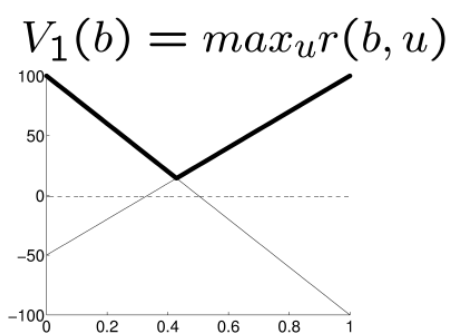
r is a linear combination:

$$
\begin{aligned}
r(b, u_1) &= -100 p_1 + 100 p_2 \\
&= -100 p_1 + 100 (1 - p_1) \\
r(b, u_2) &= 100 p_1 - 50 (1 - p_1) \\
r(b, u_3) &= -1
\end{aligned}
$$









Choice of a Policy (just for one time step)

At $t = 1$, we can use $V_1(b)$ to determine the optimal policy.

$$\pi_1(b) = \begin{cases} u_1 & \text{if } p_1 \leq \frac{3}{7} \\ u_2 & \text{if } p_1 > \frac{3}{7} \end{cases}$$

This is the upper thick graph in the diagram on the previous slide

The obtained value function $V_1(b)$ is the maximum of the three functions $r(b, u_i)$ at each point p_1

$V_1(b)$ is piecewise linear and convex.

We see the $r(b, u_3)$ does not contribute to the result. When considering the first time step only, this component can be pruned, i.e.

$$\begin{aligned} V_1(b) &= \max\{-100p_1 + 100(1 - p_1), 100p_1 - 50(1 - p_1), -1\} \\ &= \max\{-100p_1 + 100(1 - p_1), 100p_1 - 50(1 - p_1)\} \end{aligned}$$

Choice of a Policy (further time steps)

Considering one more time step the agent can also apply the sensing action u_3 (and finish then by taking u_1 or u_2)

Action u_3 leads to an observation, say z_1, the probability of which depends on the (unknown) state

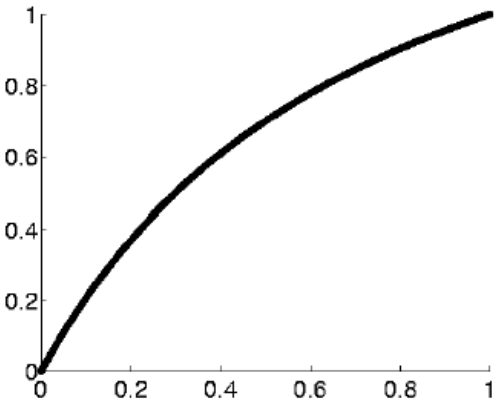
The observation can be used to update the belief using Bayes rule:

Prior: $p_1 = p(x_1)$ and $(1 - p_1)$;
Evidence $p(z_1|x_1) = 0.7$ and $p(z_1|x_2) = 0.3$;
Normalisation $p(z_1) = 0.7p_1 + 0.3(1 - p_1) = 0.4p_1 + 0.3$

Inserting this into the value function calculation

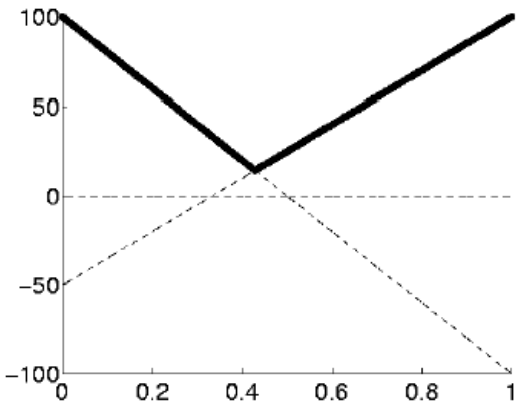
$$
\begin{aligned}
V_1(b|z_1) &= \max \left\{ \begin{array}{c} -100\frac{0.7}{p(z_1)}p_1 + 100\frac{0.3(1-p_1)}{p(z_1)} \\ +100\frac{0.7}{p(z_1)}p_1 - 50\frac{0.3(1-p_1)}{p(z_1)} \end{array} \right\} \\
&= \max \frac{1}{p(z_1)} \left\{ \begin{array}{c} -70p_1 + 30(1 - p_1) \\ +70p_1 - 15(1 - p_1) \end{array} \right\}
\end{aligned}
$$

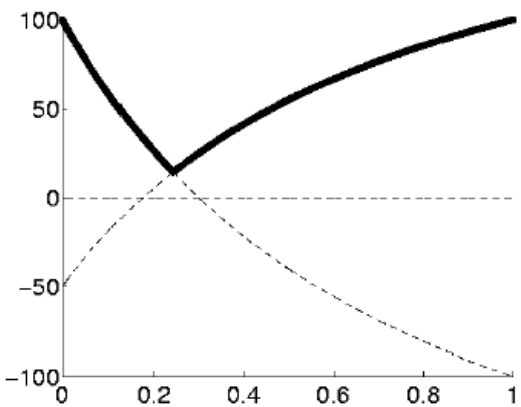
Value function



$b'(b|z_1)$

$V_1(b)$





$V_1(b|z_1)$

We should try to use the information given by the observation in such a way that the value function remains picewise linear and convex.

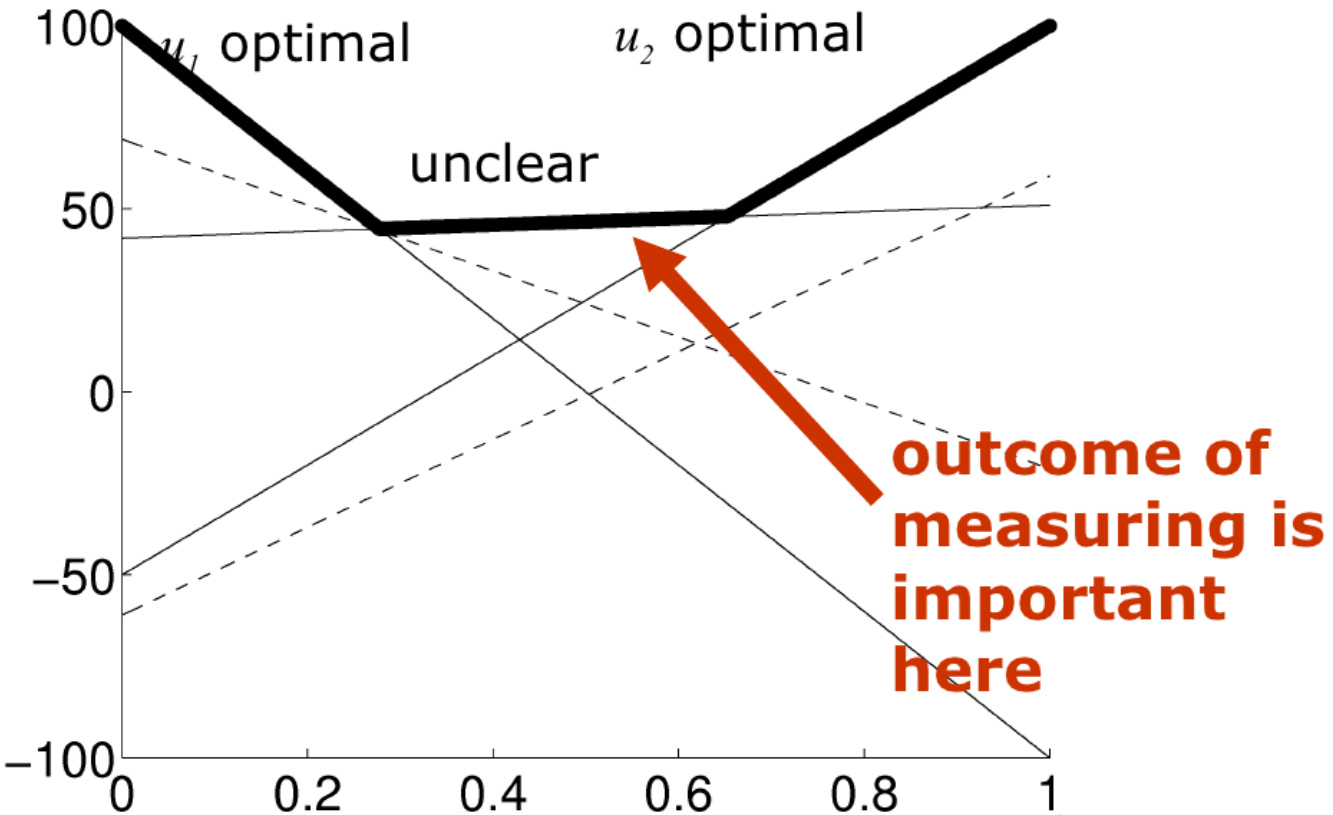
Choice of a Policy (further time steps)

Expected belief before measuring z_i

$$
\begin{aligned}
\bar{V}_1(b|u_3) &= V_1(b|z_1)\,p(z_1) + V_1(b|z_2)\,p(z_2) \\
&= \max\begin{Bmatrix} -70p_1 + 30(1-p_1) \\ +70p_1 - 15(1-p_1) \end{Bmatrix} + \max\begin{Bmatrix} -30p_1 + 70(1-p_1) \\ +30p_1 - 35(1-p_1) \end{Bmatrix} \\
&= \max\begin{Bmatrix} -70p_1 + 30(1-p_1) - 30p_1 + 70(1-p_1) \\ -70p_1 + 30(1-p_1) + 30p_1 - 35(1-p_1) \\ +70p_1 - 15(1-p_1) - 30p_1 + 70(1-p_1) \\ +70p_1 - 15(1-p_1) + 30p_1 - 35(1-p_1) \end{Bmatrix} \\
&= \max\begin{Bmatrix} -100p_1 + 100(1-p_1) \\ +40p_1 + 55(1-p_1) \\ +100p_1 + 50(1-p_1) \end{Bmatrix}
\end{aligned}
$$

(last step after pruning)

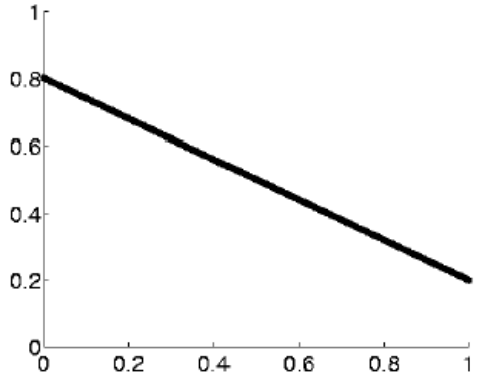
Graphical solution



State Transitions (Prediction)

Selecting u_3 may lead to a state change $\Longrightarrow$ belief update

$$\begin{aligned} p_1' &= \mathbb{E}\left[p\left(x_1|x,u_3\right)\right] \\ &= p\left(x_1|x_1,u_3\right)p_1 + p\left(x_1|x_2,u_3\right)p_2 \\ &= 0.2p_1 + 0.8\left(1-p_1\right) = 0.8 - 0.6p_1 \end{aligned}$$



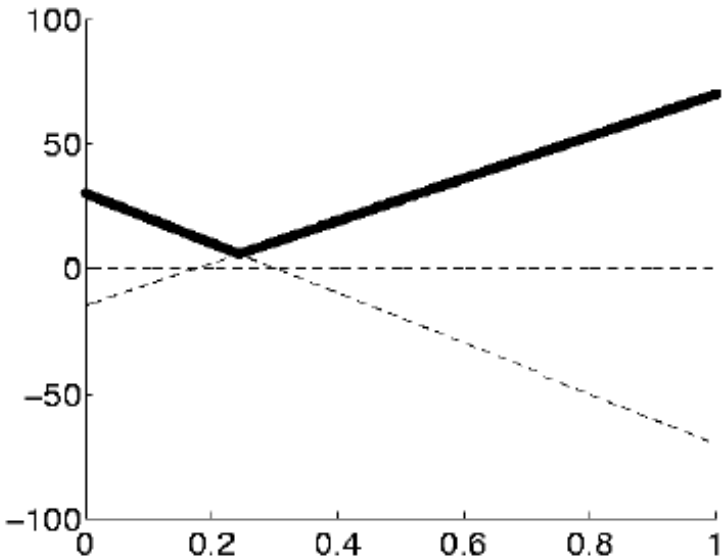
Action u_3 is taken into account when computing the value function

$$\bar{V}_1\left(b|u_3\right) = \max\left\{\begin{array}{c} -100p_1 + 100\left(1-p_1\right) \\ +40p_1 + 55\left(1-p_1\right) \\ +100p_1 + 50\left(1-p_1\right) \end{array}\right\}$$

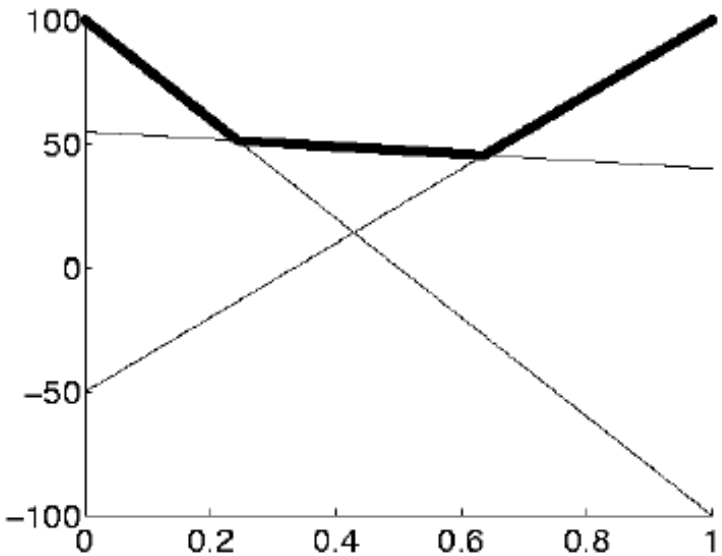
The agent can either perform u_1 or u_2, or first u_3 and then u_1 or u_2 (the latter including belief update)

$$\bar{V}_1\left(b\right) = \max\left\{\begin{array}{c} -100p_1 + 100\left(1-p_1\right) \\ 100p_1 + 50\left(1-p_1\right) \\ +51p_1 + 42\left(1-p_1\right) \end{array}\right\}$$

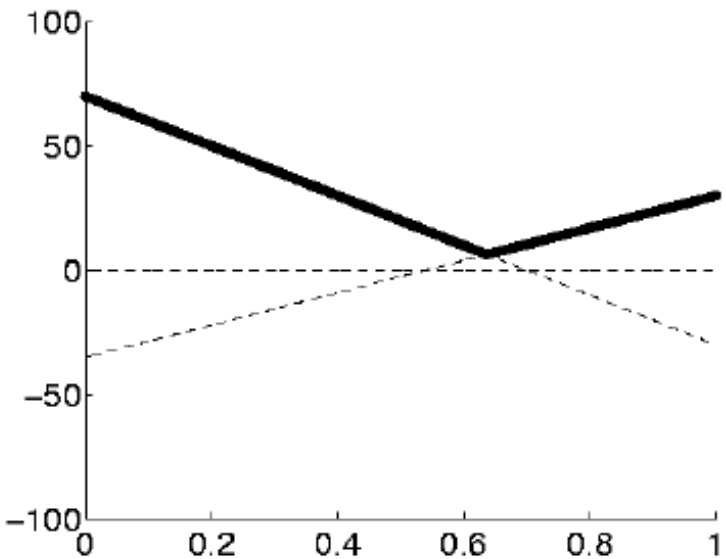
Value function: If in doubt ($p_1 \approx 0.5$) take an observation



$p(z_1)V_1(b|z_1)$



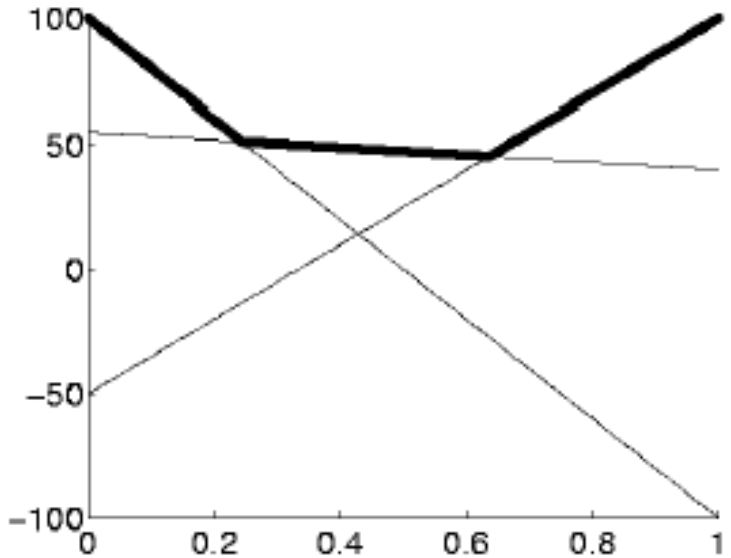
$\bar{V}_1(b)$



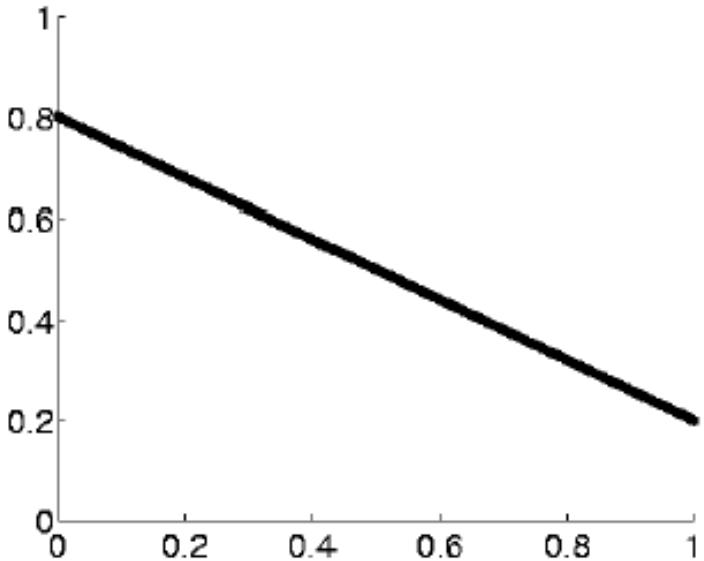
$p(z_2)V_1(b|z_2)$

Value function after executing u_3

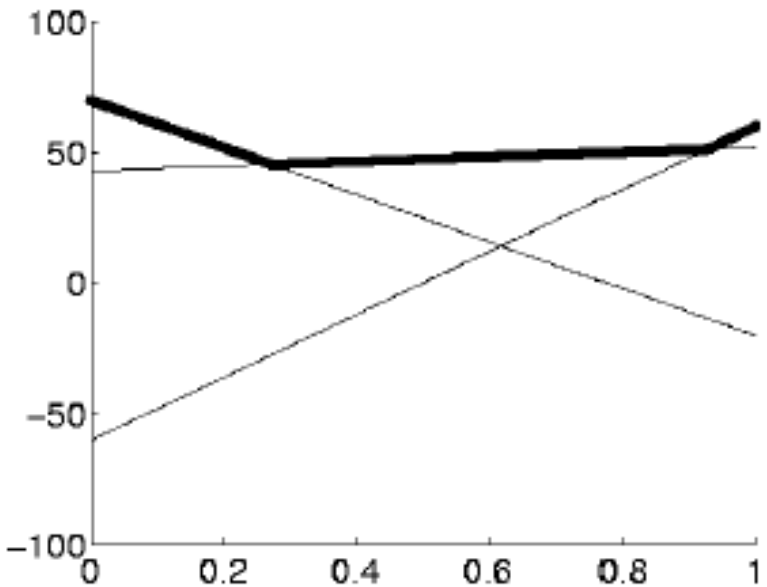
$$\bar{V}_1(b) \longrightarrow$$



Belief update



$$\bar{V}_1(b|u_3) \longrightarrow$$



Why Pruning is Essential

Each update introduces additional linear components to V.

Each measurement squares the number of linear components.

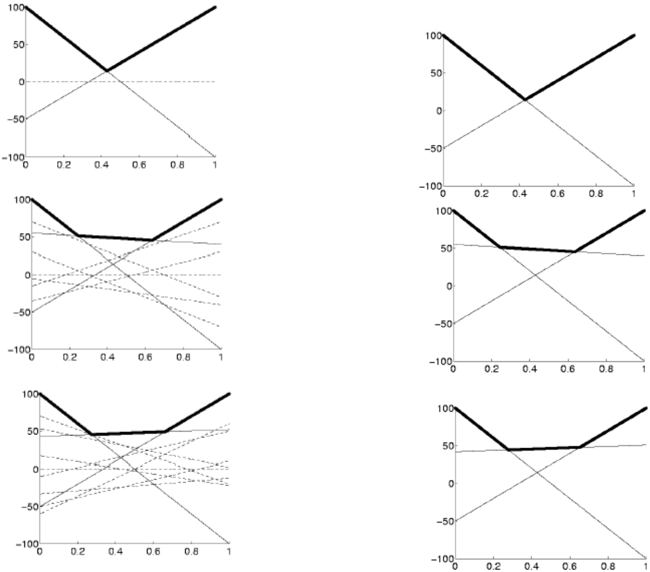
Thus, an unpruned value function for $t = 20$ includes more than $10^{547,864}$ linear functions.

At $t = 30$ we have $10^{561,012,337}$ linear functions.

The pruned value functions at $t = 20$, in comparison, contains only 12 linear components.

The combinatorial explosion of linear components in the value function are the major reason why POMDPs are impractical for most applications.

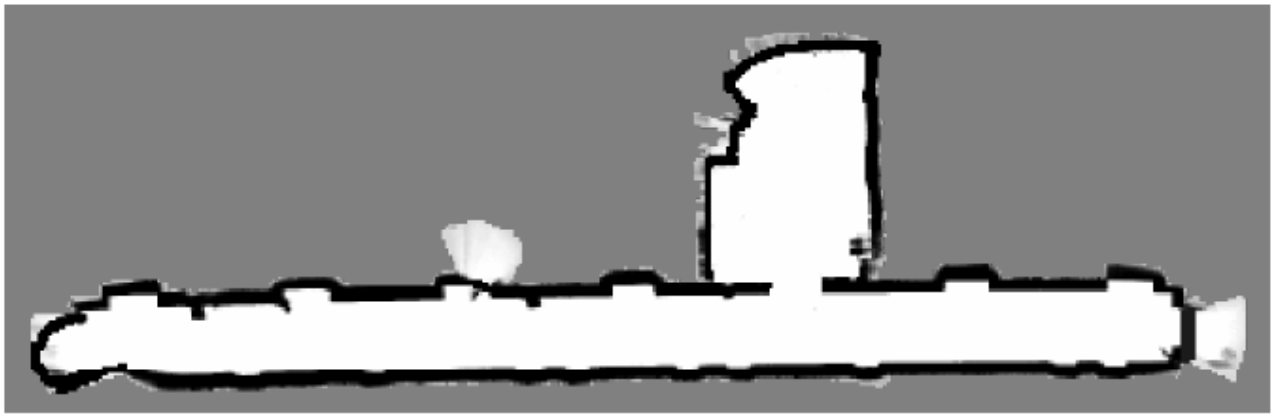
Why Pruning is Essential

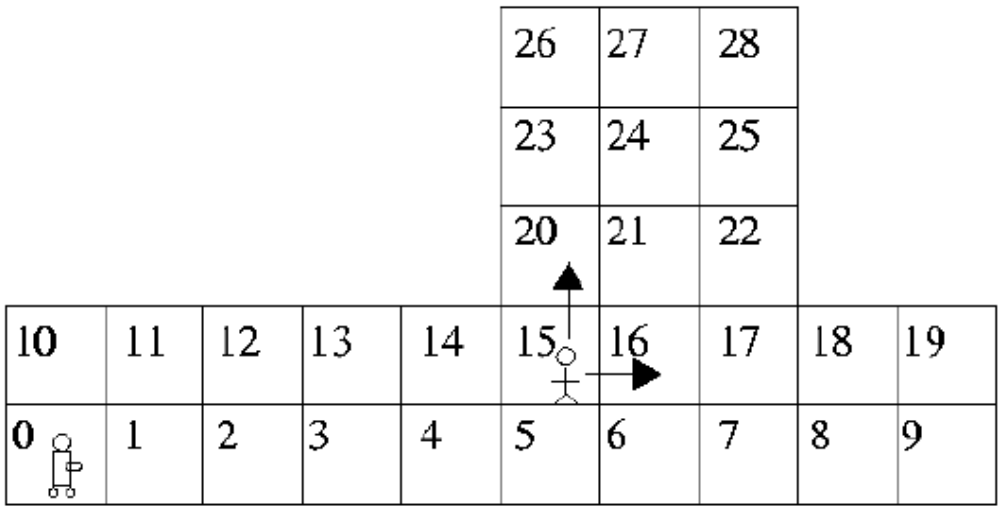


Algorithm POMDP(T):

$\Upsilon = (0, \ldots, 0)$
for $\tau = 1$ *to* T *do*
  $\Upsilon' = \emptyset$
  for all $(u'; v_1^k, \ldots, v_N^k)$ *in* Υ *do*
    for all control actions u *do*
      for all measurements z *do*
        for $j = 1$ *to* N *do*
          $$v_{j,u,z}^k = \sum_{i=1}^{N} v_i^k \; p(z \mid x_i) \; p(x_i \mid u, x_j)$$
        endfor
      endfor
    endfor
  endfor
  for all control actions u *do*
    for all $k(1), \ldots, k(M) = (1, \ldots, 1)$ *to* $(|\Upsilon|, \ldots, |\Upsilon|)$ *do*
      for $i = 1$ *to* N *do*
        $$v_i' = \gamma \left[r(x_i, u) + \sum_z v_{u,z,i}^{k(z)} \right]$$
      endfor
      add $(u; v_1', \ldots, v_N')$ to Υ'
    endfor
  endfor
  optional: prune Υ'
  $\Upsilon = \Upsilon'$
endfor
return Υ

Example Application





Preliminary Summary on POMDPs

- POMDPs compute the optimal action in partially observable, stochastic domains.
- For finite horizon problems, the resulting value functions are piecewise linear and convex.
- In each iteration the number of linear constraints grows exponentially.
- POMDPs so far have only been applied successfully to very small state spaces with small numbers of possible observations and actions.

Remaining lectures (outlook)

- 12 More on POMDPs, Bayes filters, robot experiments
- 13 RL in continuous space and time, i.e. RL and function approximation
- 14 Policy gradient methods (Natural actor critic)
- 15 Distributed RL
- 16 Model-based reinforcement learning
- 17 Apreticeship learnign and inverse RL
- 18 Self-motivated RL
- 19 Mathematical aspects, complexity of RL
- 20 RL in neural network models, in biology and psychology

(not part of the exam)

Acknowledgements

Some material was adapted from web resources associated with Sutton and Barto's Reinforcement Learning book

. . . before being used by Dr. Subramanian Ramamoorthy in this course in the last three years.

The main example of todays lecture has been used before also elsewhere e.g. in Advanced AI by W. Burgard (Freiburg)

see also:

cs.brown.edu/research/ai/pomdp/index.html