# Decision Making Under Uncertainty Ch6 Stanford AA228 course notes – Philippe Weingertner

## DMU Ch.6 State Uncertainty

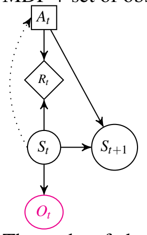
Because of sensor limitations or noise, the state might not be perfectly observable

Assumption in Ch.6: model is known

Known $T(s' \mid s,a), R(s,a), O(s,a)$ but the environement is NOT fully observable

### Formulation

**POMDP:** $< \mathscr{S}, \mathscr{A}, \mathscr{O}, T, R, O >$
MDP + set of observations $\mathscr{O}$ + observation model $O$



The proba of observing $o$ given state $s$ is written $\mathbf{O(o \mid s)}$
In some formulations, the observation can also depend on the action $a$ , and so we can write $\mathbf{O(o \mid s, a)}$
The decision in a POMDP at time $t$ can only be based on the history of observations $o_{1:t}$
Instead of keeping track of arbitrarily long histories, we keep track of the belief state
$b(s)$ is the probability assigned to being in state $s$
POMDP policy: maps belief states to actions $\mathbf{a} = \pi(\mathbf{b})$

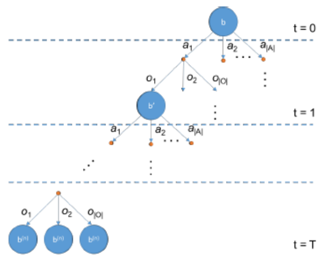
**POMDP Policy Execution**

- Execute action $a = \pi(b)$
- Observe $o$ and reward $r$
- Update belief: $b \leftarrow UpdateBelief(b,a,o)$

$\mathscr{B}$ is the set of all possible beliefs
$n$ dicrete states $\Longrightarrow \mathscr{B}$ is a subset of $\mathbb{R}^n$
POMDP is a MDP over belief-states aka belief-state MDPs. Solving belief-state MDPs is challenging bcz the state space is continuous. ADP methods can be used. But we can do better by taking advantage of the structure of the belief-state MDP

### Belief Updating

**Discrete State Filters (aka Histogram Filters)**
$b$ is a vector with proba for each state: $b_i = p(s_i)$
A POMDP Belief-Tree is represented below:



$b'(s') = P(s' \mid b,a,o)$
By Bayes Rule:
$P(s' \mid b,a,o) \propto P(o \mid s',b,a) \times P(s' \mid b,a)$
$P(s' \mid b,a,o) \propto$ Update step $\times$ Prediction step
But $P(o \mid s',b,a) = P(o \mid s',a) = O(o \mid s',a)$ as knowing the belief does not tell anything extra
$P(s' \mid b,a,o) \propto O(o \mid s',a) \times P(s' \mid b,a)$
By Law of Total probability:
$P(s' \mid b,a) = \sum_s P(s',s \mid b,a)$
$P(s' \mid b,a) = \sum_s P(s' \mid s,b,a) P(s \mid b,a)$
$P(s' \mid b,a) = \sum_s P(s' \mid s,a) b(s)$
And finally:
$\mathbf{b'(s') \propto O(o \mid s',a) \times \sum_s T(s' \mid s,a) b(s)}$

- **Prediction step:** we are summing over all states that can lead to $s'$ given the action $a$ taken in state $s$ . We get a new "prediction belief"
- **Update step:** the "prediction belief" is weighted based on our measurement likelihood in state $s'$

| Pros | Cons |
|---|---|
|  | $\mathscr{O}(\lvert S\rvert^2)$ |
| Exact if $S$ is discrete | Approximations for $S \subset \mathbb{R}^n$ |

$(s,a) \rightarrow s'$ many cases where $T(s' \mid s,a) = 0$
Complexity pretty close to $\mathscr{O}(\lvert S\rvert) \times 2$ or 3

```
function HISTOGRAMUPDATEBELIEF($b,a,o$)
    $b' \leftarrow$ zeros $(|S|)$
    for $s' \leftarrow 1$ to $|S|$ do
        $prediction = 0$
        for $s \leftarrow 1$ to $|S|$ do
            $prediction += T(s' \mid s,a) b(s)$
        $b'(s') = O(o \mid s',a) \times prediction$
    $b' \leftarrow b'/sum(b')$
    return $b'$
```

**Kalman Filters**
Linear Gaussian Models in $\mathbb{R}^n$ + Bayes Rule: that's all
Dynamics: has to be a linear function
Noise: can only be Gaussian
Parametric method: $\mu, \Sigma$ with just $2n$ parameters
Linearize with EKF

- **Prediction step:** dynamics + noise. Uncertainty grows.
- **Update step:** measurement is incorporated. Uncertainty is reduced

| Pros | Cons |
|---|---|
| FAST | Linearization can be hard |
| $\mathbb{R}^n$ easy to handle | Multimodalities |

**Particle Filters without rejection**

- **Prediction step:** dynamics + noise. This leads to particles that are more spread out.
- **Update step:** measurement is incorporated. You weight the particles by the LLH of the observation that you end up receiving and then you resample from these weighted particles. This leads to particles that are more condensed.

| Pros | Cons |
|---|---|
| Efficient: $\mathscr{O}(\lvert b\rvert)$ complexity |  |
| Non Gaussian beliefs |  |

```
function PARTICLEUPDATEBELIEF($b,a,o$)
    $b' \leftarrow 0$
    for $i \leftarrow 1$ to $|b|$ do
        $s'_i \sim G(s_i,a)$          ▷ prediction step: shift + noise
        $w_i = O(o \mid s'_i, a)$     ▷ observation weight
    for $i \leftarrow 1$ to $|b|$ do
        Sample a new particle $k$ with proba $\propto w_k$
        Add $s'_k$ to $b'$
    return $b'$
```

### Exact Solution Methods

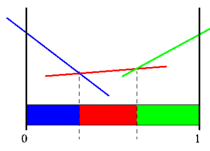
For finite horizons: Value function is Piece Wise Linear Convex (proof by induction)

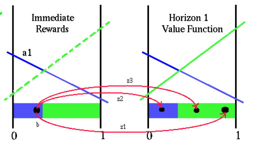
**Convex:** if you take any 2 points on that function and draw a segment, the function of interest is below or equal this segment (you may check that this is equivalent to $f'' \geq 0$ i.e. $f'$ always increasing)

A set of alpha-vectors $\alpha_\mathbf{a}$ represents a set of hyperplans

$\alpha_\mathbf{a}$ : is a vector of size $|S|$ which captures value of action $a$ in the different states $s$

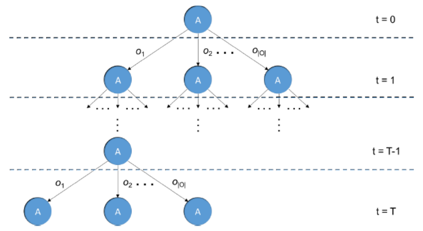
Value function: is equal to the upper part of all these hyperplans





Policy: is compactly represented by a set of alpha-vectors

**POMDP Value Iteration**
vectors are in bold: $\mathbf{v}$
1 **step horizon:**
$U^*(b) = \underset{a}{max} \sum_s R(s,a) b(s)$
$U^*(b) = \underset{a}{max}\ \alpha_\mathbf{a}^\mathbf{T} \mathbf{b}$
$t$ step: $U$ defined by a set $\Gamma_t$ of $\alpha_\mathbf{a}^\mathbf{t}$ vectors
$\alpha_\mathbf{a}^\mathbf{t}$ not optimal for any belief state are discarded
$t+1$ step: $U$ defined by a new set $\Gamma_{t+1}$ of $\alpha_\mathbf{a}^\mathbf{t+1}$
**Policy Tree**



**How many recursions ?**
The number of $\alpha$-vectors grows exponentially in the number of observations at each iteration
$|\Gamma_{t+1}| = |A| \times |\Gamma_t|^{|\mathscr{O}|}$
Each new $\alpha$-vector requires computation time in $\mathscr{O}(|O| \times |S|^2)$ (cf 14-ExactPOMDPMethods.ipynb)

**Conditional Plans (DMU 6.3.2)**
We compute the Utility of a Policy Tree in a top down way. We recurse down until we reach the horizon. When we reach horizon: the utility is 0 . So we can solve $U^P(s)$

$$\boxed{U^P(s) = R(s,a) + \gamma \sum_{s'} T(s' \mid s,a) \sum_o O(o \mid s',a) U^{P(o)}(s')}$$

$P$: a policy tree with **root action** $a$
$P(o)$: subplan (subtree) associated with observation $o$
We can then compute the expected Utility associated with a belief state as:
$\boxed{U^P(b) = \sum_s U^P(s) b(s)}$ which is a dot product
We can use the $\alpha_P$ vectors to represent the vectorized version of $U^P$ e.g. for a 2 states-space
$\alpha_\mathbf{P} = \left[U^P(s_1) \quad U^P(s_2)\right]^T$ and $\mathbf{b} = [P(s_1) \quad P(s_2)]^T$
$\boxed{U^P(b) = \alpha_\mathbf{P}^\mathbf{T} \mathbf{b}}$ (dot product with the belief vector)
If we maximize over the space of all possible plans up to the planning horizon, then we can fin
$\boxed{U^*(b) = \underset{P}{max}\ \alpha_\mathbf{P}^\mathbf{T} \mathbf{b}}$
Hence the finite horizon optimal value function is PWLC. We simply execute the action at the root node of the plan that maximizes $\alpha_\mathbf{P}^\mathbf{T} \mathbf{b}$

It is generally infeasible to enumerate every possible h-step plan to find the one that maximizes above equation from the current belief state. We have $|A|^{\frac{|O|^h - 1}{|O| - 1}}$ h-step plans.

***Value Iteration:*** iterate over all the one-step plans and toss out the plans that are not optimal for any belief state. And so on ... Discard ***dominated*** plans along the way...

Notes:

- Only in the 1st step horizon you have 1 action per alpha vector. With an arbitrary number of steps you could have more than 1
- Question at 1H16 W7 last video:
  - For finite horizons problems, alpha vectors are different for each time step. They change with time
  - If you represent your policy as a set of "the same" (I guess) alpha vectors, that is kind of assuming that you have an infinite horizon problem, because your policy does not change with time. So we have a stationnary policy w.r.t. our belief states (but of course the belief state itself is changing over time)
- Usually we are interested by stationnary policies w.r.t. our belief state
- How to adress finite horizon pbs: e.g. add time to your state space or use a tree based representation

**14-ExactPOMDPMethods.ipynb**

**Backup UpdateAlphas($\mathbf{b}, \mathbf{alphas}$):**

$\alpha_a^P(s) = R(s,a) + \gamma \sum_{s'} T(s' \mid s,a) \sum_o O(o \mid s',a) \alpha_{ao}^{P(o)}(s')$
$P$: a policy tree with **root action** $a$
$P(o)$: subplan (subtree) associated with observation $o$

We have $\alpha_{ao}^{P(o)}$ computed this way:

- $\mathbf{b_{ao}} = UpdateBelief(b,a,o)$

E.g. for a 2 states-space $\mathbf{b_{ao}} = [P(s_1) \quad P(s_2)]$

- $\alpha_{\mathbf{ao}}^{\mathbf{P(o)}} = \underset{\alpha_a^{P(o)}}{argmax}\ \alpha_{\mathbf{a}}^{\mathbf{P(o)}} \mathbf{b_{ao}}$ which means we pick the highest piece-wise linear part of $U$ w.r.t. our new belief

With 2 actions, every $UpdateAlphas$ adds 2 $\alpha$-vectors
**How to call UpdateAlphas($b, alphas$) ?**
Root belief is: $b_0^{(0)}$

Let say current beliefs are: $b_1^{(1)}, b_2^{(1)}, b_3^{(1)}, b_4^{(1)}$
For every current belief add e.g. 4 children beliefs (2 actions x 2 observations) $\Longrightarrow b_1^{(2)}, \dots, b_{16}^{(2)}$
Then for all 16 children:

- UpdateAlphas($b_i^{(2)}, alphas$)
- UpdateAlphas($b_{parent_i}^{(1)}, alphas$)
- UpdateAlphas($b_0^{(0)}, alphas$)

The tree grows exponentially
Prune the dominated alpha-vectors to limit the growth

**Offline Methods**

We start first with two very loose approximation of POMDP solution: QMDP and FIB

**QMDP**
We simply take an expectation of the MDP's Q-values, so we include our current state uncertainty, but then the MDP solution has no state uncertainty in it.

$$\pi^{QMDP}(b) = \underset{a}{argmax} \sum_s b(s) Q(s,a)$$

1 alpha-vector per action based on $Q(s,a)$
Pre-computed offline under full observability assumption: $Q_{MDP}$
It assumes uncertainty disappears at the next time step
Works well in practice when you do not have to take info gathering actions
Then approximate online:

- $U(b) = \underset{a}{max}\ \alpha_{\mathbf{a}}^{\mathbf{T}} \mathbf{b}$
- Optimal action: $\underset{a}{argmax}\ \alpha_{\mathbf{a}}^{\mathbf{T}} \mathbf{b}$
- $b' = BeliefUpdate(b,a,o)$

**Algo: VI on every $\alpha_a$**
Init: $\alpha_a^{(0)}(s) = 0$
Iter: $\alpha_a^{(k+1)}(s) = R(s,a) + \gamma \sum_{s'} T(s' \mid s,a) \underset{a'}{max}\ \alpha_{a'}^{(k)}(s')$
VI iter until CV
Complexity: $\mathscr{O}(|A|^2 |S|^2)$
Properties:

- QMDP policy is suboptimal
- Upper Bound on U: $U^*(\mathbf{b}) \le \underset{a}{max}\ \alpha_{\mathbf{a}}^{\mathbf{T}} \mathbf{b}$ for all $\mathbf{b}$

You should try to understand intuitively why QMDP is an overestimate. If we know our state, we are going to do better than if we only have a belief, or distribution over states. State uncertainty makes it harder to have a good solution. Because we are using the MDP solution, we are not accounting for the fact that we will have state uncertainty (which will likely make performance worse).

The lack of the observation function is related to QMDP's being an overestimate. One way to derive QMDP is to switch a sum and max operator in the Bellman update, which makes the updated value an overestimate. Because of this switch, the observation function falls out of the update equation. It's a bit more complex than this and if you are really interested, you should look up Hauskrecht's paper, "Value-function approximations for partially observable Markov decision processes.

**FIB**
Same idea as QMDP but takes into account partial observability
$\alpha_a^{(k+1)}(s) = R(s,a) + \gamma \sum_o \sum_{s'} O(o \mid s',a) T(s' \mid s,a) \underset{a'}{max}\ \alpha_{a'}^{(k)}(s')$
$\alpha_a^{(k+1)}(s) = R(s,a) + \gamma \sum_o \underset{a'}{max} \sum_{s'} O(o \mid s',a) T(s' \mid s,a)\ \alpha_{a'}^{(k)}(s')$
Complexity: $\mathscr{O}(|A|^2 |S|^2 |O|)$ still polynomial
Upper Bound on the value fct: $U^* \le FIB \le QMDP$

**PBVI**
Major advance 20 years ago
Enabled to solve interesting problems
Key idea:

- Backup $n$ alpha vectors associated with a point in the belief space
- Then use the $n$ alpha-vectors to approximate the value fct anywhere in the belief space via $U^\Gamma(\mathbf{b}) = \underset{\alpha \in \Gamma}{max}\ \alpha^{\mathbf{T}} \mathbf{b}$

**Algo:**

- Init: $B = \{b_1, \dots, b_n\}, \Gamma = \{\alpha_1, \dots, \alpha_n\}$
- $\Gamma$ is used to represent the value fct
- Init the $|S|$ components of all $\alpha_i$ to a lower bound value $\underline{U}(s) = \underset{a}{max} \sum_t \gamma^t \underset{s}{min}\ R(s,a) = \frac{1}{1-\gamma} \underset{a}{max} \underset{s}{min}\ R(s,a)$
- Iter: $\alpha_i \leftarrow BackupBelief(b_i, \Gamma)$ until CV

```
function BACKUPBELIEF($b, \Gamma$)
    for $a \in A$ do
        for $o \in O$ do
            $b' \leftarrow$ UPDATEBELIEF($b, a, o$)
            $\alpha_{a,o} \leftarrow argmax_{\alpha \in \Gamma}\ \alpha^T b'$
        for $s \in S$ do
            $\alpha_a(s) \leftarrow R(s,a) + \gamma \sum_{s',o} O(o \mid s',a) T(s' \mid s,a) \alpha_{a,o}(s')$
    $\alpha \leftarrow argmax_{\alpha_a}\ \alpha_a^T b$
    return $\alpha$
```

**Belief Points Selection**
2 general principles:

- Some kind of exploration strategy to choose some likely reachable belief points
  - start from b
  - choose a random action
  - sample s, s', o
  - $b' = UpdateBelief(b,a,o)$
- You want them to be spaced out so that approximation holds throughout the space
  - same as above but with all possible actions
  - resulting in $|A|$ candidate new beliefs: $b_a(s)$
  - Select and add the 1 that is most spaced out: $b' = argmax_{b_a}\ min_{b \in B} \sum_s |b(s) - b_a(s)|$

**Online Methods**

**Lookahead with Approximate Value Function**
Todo ... We compute the Utility of a Policy Tree in a top down way. We recurse down until we reach the horizon. When we reach horizon: the utility is 0 . So we can solve $U^P(s)$

**Forward Search**
Todo ... We compute the Utility of a Policy Tree in a top down way. We recurse down until we reach the horizon. When we reach horizon: the utility is 0 . So we can solve $U^P(s)$

**Branch and Bound**
Todo ... We compute the Utility of a Policy Tree in a top down way. We recurse down until we reach the horizon. When we reach horizon: the utility is 0 . So we can solve $U^P(s)$

**Monte Carlo Tree Search**
Todo ... We compute the Utility of a Policy Tree in a top down way. We recurse down until we reach the horizon. When we reach horizon: the utility is 0 . So we can solve $U^P(s)$