Lecture 12 - Point-Based Value Iteration

OBJECTIVE: In this lecture we introduce approximate value iteration algorithms for POMDPS. These algorithms make use of a reduced set of sampled beliefs. Our presentation will follow the PBVI algorithm of Joelle Pineau and the PERSEUS algorithm of Matthijs Spaan.

◇ POINT-BASED BACKUPS

Approximate PBVI methods backup only a finite set of belief points. The intuition here is that the filtering beliefs, obtained by following an arbitrary policy at the start, provide a good set of points to approximate the PWLC value function.

The following picture illustrates how the value function (for a two-state problem) is updated

⋆ Perseus:

The computation of the value function for a single belief follows from the previous lecture. Here we will use the notation of Spaan and Vlassis. In particular, it uses vectors and makes the index i over the α-vectors explicit:

$$\begin{aligned} b \cdot r_a &= \sum_s b(s) r(s,a) \\ \alpha^{ay}(s) &= g^i_{ay}(s) \end{aligned}$$

The value update (backup) is:

$$\begin{aligned} V_{t+1}(b) &= \max_a \left\{ b \cdot r_a + \gamma \sum_y p(y|a,b) V_t(b^y_a) \right\} \\ &= \max_a \left\{ b \cdot r_a + \gamma \sum_y p(y|a,b) \max_{\{\alpha^i_t\}_i} \sum_{s'} b^y_a(s') \alpha^i_t(s') \right\} \\ &= \max_a \left\{ b \cdot r_a + \gamma \sum_y \max_{\{\alpha^i_t\}_i} \sum_{s'} p(y|a,s') \sum_s b(s) p(s'|a,s) \alpha^i_t(s') \right\} \\ &= \max_a \left\{ b \cdot r_a + \gamma \sum_y \max_{\{g^i_{ay}\}_i} b \cdot g^i_{ay} \right\} \\ &= \max_a \left\{ b \cdot \left[r_a + \gamma \sum_y \arg\max_{\{g^i_{ay}\}_i} b \cdot g^i_{ay} \right] \right\} = \max_a \left\{ b \cdot g^b_a \right\} \end{aligned}$$

The new α-vector is then:

$$\alpha^b_{t+1} = \texttt{backup}(b) = \arg\max_{\{g^b_a\}_a} b \cdot g^b_a$$

The Perseus algorithm makes use of these recursions as shown in the following pseudo-code:

1. Set $V_{t+1} = \emptyset$. Initialize available sample beliefs $\widetilde{B}$ to B.
2. Sample a belief point b u.a.r. from $\widetilde{B}$ and compute $\alpha^b_{t+1} = \texttt{backup}(b)$.
3. If $b \cdot \alpha^b_{t+1} \geq V_t(b)$ then add α^b_{t+1} to V_{t+1}. Otherwise add $\alpha' = \arg\max_{\{\alpha^i_t\}_i} b \cdot \alpha^i_t$ to V_{t+1}.
4. Compute $\widetilde{B} = \{b \in B : V_{t+1}(b) < V_t(b)\}$.
5. If $\widetilde{B} = \emptyset$ then stop, else go to 2.

In matlab, we can use the following routines (from Perseus) to compute g^i_{ay} and the new α-vectors.

```
function [GammaAO] = computeGammaAO(V0)
% computeGammaAO - compute the backprojected vectors from t+1
% V0 - struct array of alpha vectors
% GammaAO  - size(V0){nrA}{nrO} backprojected copies from V0

% Author: Matthijs Spaan
% Copyright (c) 2003,2004 Universiteit van Amsterdam.

global problem;
nrA=problem.nrActions;
nrO=problem.nrObservations;
nrS=problem.nrStates;

[nrInV,foo]=size(V0); % number of alpha vectors.
V0v=vertcat(V0.v);

for a=1:nrA
  for o=1:nrO
    GammaAO{a}{o}=zeros(nrInV,nrS);
    for k=1:nrInV
      for s1=1:nrS
        GammaAO{a}{o}(k,:)=GammaAO{a}{o}(k,:) + ...
        gamma*transition(s1,:,a) * observation(s1,a,o)*V0v(k,s1);
      end
    end
  end
end
```



```
function Alpha = pbSingleBackup(B,GammaAO)
% pbSingleBackup - backup a single belief point
% B         - (1 x d) single belief point to be backed up
% GammaAO   - backprojected vectors from computeGammaAO.m
% Alpha     - the alpha vector of B given GammaAO[S]
% Author: Matthijs Spaan. Copyright (c) 2003,2004 U. van Amsterdam.
global problem; nrA=problem.nrActions; nrO=problem.nrObservations;
nrS=problem.nrStates; [nrInV,foo]=size(GammaAO{1}{1});

for a=1:nrA
  for o=1:nrO
    alphaDotB{a}{o}=zeros(nrInV,1);
    alphaDotB{a}{o}=GammaAOS{a}{o}*B';
  end
end

for a=1:nrA
  tmpAlpha=zeros(1,nrS);
  for o=1:nrO
    [foo,iMax]=max(alphaDotB{a}{o});
    tmpAlpha=tmpAlpha+GammaAO{a}{o}(iMax,:);
  end
  GammaAB{a}=problem.reward(:,a)'+tmpAlpha;
end

val=zeros(nrA,1);
for a=1:nrA
  val(a)=GammaAB{a}*B';
end
maxActions=find(val==max(val));
aMax=maxActions(ceil(rand*length(maxActions)));
Alpha.v=GammaAB{aMax}; Alpha.a=aMax;
```