CS 287: Advanced Robotics
Fall 2009

Lecture 5: Control 4: Optimal control / Reinforcement learning--- function approximation in dynamic programming

Pieter Abbeel
UC Berkeley EECS

Today

- Recap + continuation of value iteration with function approximation
- Performance boosts
- Speed-ups
- Intermezzo: Extremely crude outline of (part of) the reinforcement learning field [as it might assist when reading some of the references]

Great references:

Gordon, 1995, "Stable function approximation in dynamic programming"

Tsitsiklis and Van Roy, 1996, "Feature based methods for large scale dynamic programming"

Bertsekas and Tsitsiklis, "Neuro-dynamic programming," Chap. 6

Recall: Discounted infinite horizon

- Markov decision process (MDP) (S, A, P, γ, g)
  - γ: discount factor
- Policy $\pi = (\mu_0, \mu_1, \ldots),\ \mu_k : S \to A$
- Value of a policy π: $J^\pi(x) = \mathrm{E}[\sum_{t=0}^{\infty} \gamma^t g(x(t), u(t)) | x_0 = x, \pi]$
- Goal: find $\pi^* \in \arg\min_{\pi \in \Pi} J^\pi$

Recall: Discounted infinite horizon

- **Dynamic programming (DP) aka Value iteration (VI):**

  For i=0,1, …

  For all s $\in$ S

$$J^{(i+1)}(s) \quad \leftarrow \quad \min_{u \in A} g(s,u) + \gamma \sum_{s'} P(s'|s,u) J^{(i)}(s')$$

- **Facts:**

  $J^{(i)} \to J^*$ for $i \to \infty$

  There is an optimal stationary policy: $\pi^* = (\mu^*, \mu^*, \ldots)$ which satisfies:

$$\mu^*(s) = \arg\min_u g(s,u) + \gamma \sum_{s'} P(s'|s,u) J^*(s)$$

- **Issue in practice: Bellman's curse of dimensionality: number of states grows exponentially in the dimensionality of the state space**

DP/VI with function approximation

Pick some $S' \subseteq S$ [typically the idea is that $|S'| << |S|$].
Iterate for $i = 0, 1, 2, \ldots$:

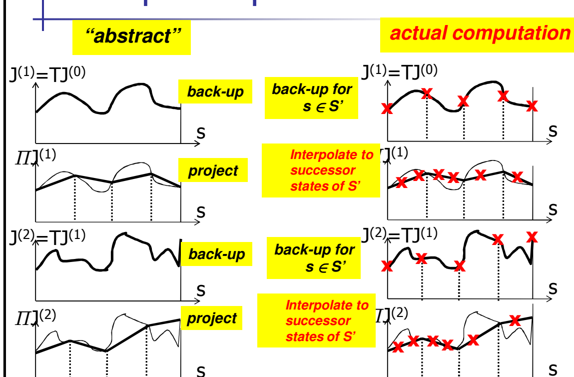
back-ups: $\forall s \in S' : \bar{J}^{(i+1)}(s) \leftarrow \min_{u \in A} g(s,u) + \gamma \sum_{s'} P(s'|s,u) \hat{J}_{\theta^{(i)}}(s')$

projection: find some $\theta^{(i+1)}$ such that $\forall s \in S' \ \ \hat{J}_{\theta^{(i+1)}}(s) = (\Pi \bar{J}^{(i+1)})(s) \approx \bar{J}^{(i+1)}(s)$

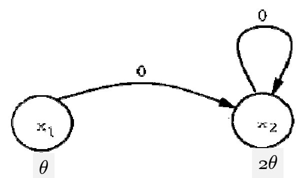
Projection enables generalization to $s \in S \setminus S'$, which in turn enables the Bellman back-ups in the next iteration.

θ parameterizes the class of functions used for approximation of the cost-to-go function

Example --- piecewise linear



Recall: VI with function approximation need not converge!



P(x2|x1,u) = 1; P(x2|x2,u) = 1
g(x1,u) = 0; g(x2,u) = 0;

Function approximator: [1 2] * θ

VI w/ least squares function approximation diverges for γ > 5/6 [see last lecture for details]

Contractions

- **Fact.** *The Bellman operator, T, is a* γ *-contraction w.r.t. the infinity norm, i.e.,*

$$\forall J_1, J_2 : \|TJ_1 - TJ_2\|_\infty \leq \gamma \|J_1 - J_2\|_\infty$$

- **Theorem.** *The Bellman operator has a unique fixed point J* = TJ* and for all J we have that $T^{(k)}$J converges to J* for* k *going to infinity.*
- Note:

$$\begin{aligned}
\|T^{(k)}J - J^*\|_\infty &= \|T^{(k)}J - T^{(k)}J^*\|_\infty \\
&\leq \gamma \|T^{(k-1)}J - T^{(k-1)}J^*\|_\infty \\
&\leq \gamma^k \|J - J^*\|_\infty
\end{aligned}$$

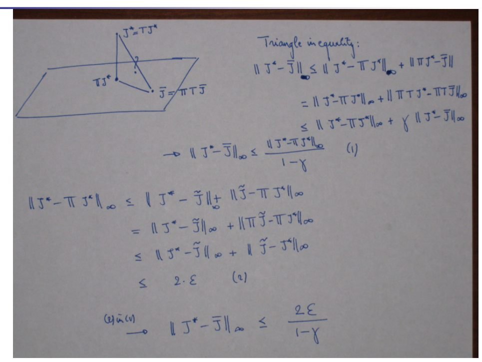
I.e., with every back-up, the infinity norm distance to J^* decreases.

Guarantees for fixed point

Theorem. Let J^* be the optimal value function for a finite MDP with discount factor γ. Let the projection operator Π be a non-expansion w.r.t. the infinity norm and let $\tilde{J}$ be any fixed point of Π. Suppose $\|\tilde{J} - J^*\|_\infty \leq \epsilon$. Then ΠT converges to a value function $\bar{J}$ such that:

$$\|\bar{J} - J^*\| \leq \frac{2\epsilon}{1-\gamma}$$

Proof



[See also Gordon 1995]

Can we generally verify goodness of some estimate J despite not having access to J*

Fact. Assume we have some $\hat{J}$ for which we have that $\|\hat{J} - T\hat{J}\|_\infty \leq \epsilon$. Then we have that $\|\hat{J} - J^*\|_\infty \leq \frac{\epsilon}{1-\gamma}$.
Proof:

$$\begin{aligned}
\|\hat{J} - J^*\|_\infty &= \|\hat{J} - T\hat{J} + T\hat{J} - T^2\hat{J} + T^2\hat{J} - T^3\hat{J} + \ldots - J^*\|_\infty \\
&\leq \|\hat{J} - T\hat{J}\|_\infty + \|T\hat{J} - T^2\hat{J}\|_\infty + \|T^2\hat{J} - T^3\hat{J}\|_\infty + \ldots + \|T^\infty\hat{J} - J^*\|_\infty \\
&\leq \epsilon + \gamma\epsilon + \gamma^2\epsilon + \ldots
\end{aligned}$$

$$= \frac{\epsilon}{1-\gamma}$$

- Of course, in most (perhaps all) large scale settings in which function approximation is desirable, it will be hard to compute the bound on the infinity norm …

What if the projection fails to be a non-expansion

- Assume Π only introduces a little bit of noise, i.e.,

$$\forall \text{ iterations } i : \|T\bar{J}^{(i)} - \Pi T\bar{J}^{(i)}\|_\infty \leq \epsilon$$

Or, more generally, we have a noisy sequence of back-ups:

$$J^{(i+1)} \leftarrow TJ^{(i)} + w^{(i)} \text{ with the noise } w^{(i)} \text{ satisfying: } \|w^{(i)}\|_\infty \leq \epsilon$$

Fact. $\|J^{(i)} - T^iJ\| \leq \epsilon(1+\gamma+\ldots+\gamma^{i-1})$ and as a consequence $\limsup_{i\rightarrow\infty} \|J^{(i)} - J^*\| \leq \frac{\epsilon}{1-\gamma}$.

Proof by induction:
Base case: We have $\|J^{(1)} - TJ^{(0)}\|_\infty \leq \epsilon$.
Induction: We also have for any $i > 1$:

$$\begin{aligned}
\|T^iJ^{(0)} - J^{(i)}\|_\infty &= \|TT^{i-1}J^{(0)} - TJ^{(i-1)} - w^{(i-1)}\|_\infty \\
&\leq \epsilon + \gamma\|T^{i-1}J^{(0)} - J^{(i-1)}\|_\infty \\
&\leq \epsilon + \gamma(\epsilon(1+\gamma+\gamma^2+\ldots+\gamma^{(i-2)}))
\end{aligned}$$

Guarantees for greedy policy w.r.t. approximate value function

Definition. μ is the greedy policy w.r.t. J if for all states s:

$$\mu(s) \in \arg\min_u g(s,u) + \gamma \sum_{s'} P(s'|s,u)J(s')$$

Fact. Suppose that J satisfies $\|J - J^*\|_\infty \le \epsilon$. If μ is a greedy policy based on J, then

$$\|J^\mu - J^*\|_\infty \le \frac{2\gamma\epsilon}{1-\gamma}$$

Here $J^\mu = \mathrm{E}[\sum_{t=0}^\infty \gamma^t g(s_t, \mu(s_t))]$.

[See also Bertsekas and Ttsitsiklis, 6.1.1]

Proof

Recall:

$$(TJ)(s) = \min_u g(s,u) + \gamma \sum_{s'} P(s'|s,u)J(s')$$

Similarly define:

$$(T_\mu J)(s) = g(s,\mu(s)) + \gamma \sum_{s'} P(s'|s,\mu(s))J(s')$$

We have $TJ^* = J^*$ and (same result for MDP with only 1 policy available) $T_\mu J^\mu = J^\mu$.

A very typical proof follows, with the main ingredients adding and subtracting the same terms to make terms pairwise easier to compare/bound:

$$\begin{aligned}
\|J^\mu - J^*\|_\infty &= \|T_\mu J^\mu - J^*\|_\infty \\
&\le \|T_\mu J^\mu - T_\mu J\|\infty + \|T_\mu J - J^*\|_\infty \\
&\le \gamma\|J^\mu - J\|_\infty + \|TJ - J*\|_\infty \\
&\le \gamma\|J^\mu - J^*\|_\infty + \gamma\|J^* - J\|_\infty + \gamma\|J - J^*\|_\infty \\
&= \gamma\|J^\mu - J^*\|_\infty + 2\gamma\epsilon,
\end{aligned}$$

and the result follows.

Recap function approximation

- DP/VI with function approximation:
  - Iterate: $J \leftarrow \Pi\, T\, J$
- Need not converge!
- Guarantees when:
  - The projection is an infinity norm non-expansion
  - Bounded error in each projection/function approximation step
- In later lectures we will also study the policy iteration and linear programming approaches

Reinforcement learning---very crude map

- Exact methods w/full model available (e.g. Value iteration/DP, policy iteration, LP)
- Approximate DP w/model available
- Sample states:
  - Use all sampled data in batch → often reducible to "exact methods" on an approximate transition model
  - Use incremental updates → stochastic approximation techniques might prove convergence to desired solution

Improving performance with a given value function

1. **Multi-stage lookahead aka Receding/Moving horizon**
   - Rather than using greedy policy μ w.r.t. approximate value function, with

     $$\mu(s_t) = \arg\min_u g(s,u) + \gamma \textstyle\sum_{s'} P(s'|s,u)\hat{J}_\theta(s')$$
   - Two-stage lookahead:
     - At time t perform back-ups for all s' which are successor states of s_t
     - Then use these backed up values to perform the back-up for s_t
   - N stage lookahead: similarly,perform back-ups to N-stages of successor states of s_t backward in time
   - Can't guarantee N-stage lookahead provides better performance [Can guarantee tighter infinity norm bound on attained value function estimates by N-stage lookahead.]
   - Example application areas in which it has improved performance chess, backgammon

See also Bertsekas and Tsitsiklis, 6.1.2

Improving performance with a given value function

2. Roll-out policies

- Given a policy π, choose the current action u by evaluating the cost encurred by taking action u followed by executing the policy π from then onwards
- Guaranteed to perform better than the baseline policy on top of which it builds (thanks to general guarantees of policy iteration algorithm)
- Baseline policy could be obtained with any method
- Practicalities
  - Todo --- fill in

See also Bertsekas and Tsitsiklis, 6.1.3

Speed-ups

- Parallelization
  - VI lends itself to parallellization
- Multi-grid, Coarse-to-fine grid, Variable resolution grid
- Prioritized sweeping
- Richardson extrapolation
- Kuhn triangulation

Prioritized sweeping

- **Dynamic programming (DP) / Value iteration (VI):**

  For i=0,1, …

  For all s $\in$ S

$$J^{(i+1)}(s) \quad \leftarrow \quad \min_{u \in A} g(s,u) + \gamma \sum_{s'} P(s'|s,u) J^{(i)}(s')$$

- Prioritized sweeping idea: focus updates on states for which the update is expected to be most significant
- Place states into priority queue and perform updates accordingly
  - For every Bellman update: compute the difference J^{(i+1)} – J^{(i)}
  - Then update the priority of the states s' from which one could transition into s based upon the above difference and the transition probability of transitioning into s'
- For details: See Moore and Atkeson, 1993, "Prioritized sweeping: RL with less data and less real time"

Richardson extrapolation

- Generic method to improve the rate of convergence of a sequence
- Assume h is the grid-size parameter in a discretization scheme
- Assume we can approximate $J^{(h)}(x)$ as follows:

$$J^{(h)}(x) = J(x) + J_1(x)h + o(h)$$

- Similarly:

$$J^{(h/2)}(x) = J(x) + J_1(x)h/2 + o(h)$$

- Then we can get rid of the order h error term by using the following approximation which combines both:

$$2J^{(h/2)}(x) - J^{(h)}(x) = J(x) + o(h)$$

Kuhn triangulation

- Allows efficient computation of the vertices participating in a point's barycentric coordinate system and of the convex interpolation weights (aka the barycentric coordinates)

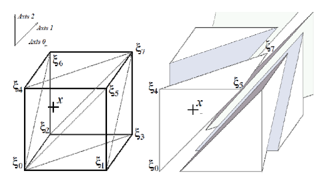


Figure 2. The Kuhn triangulation of a (3d) rectangle. The point x satisfying $1 \geq x_2 \geq x_0 \geq x_1 \geq 0$ is in the simplex $(\xi_0, \xi_4, \xi_5, \xi_7)$.

- See Munos and Moore, 2001 for further details.

Kuhn triangulation (from Munos and Moore)

3.1. Computational issues

Although the number of simplexes inside a rectangle is factorial with the dimension d, the computation time for interpolating the value at any point inside a rectangle is only of order $(d \ln d)$, which corresponds to a sorting of the d relative coordinates $(x_0, ..., x_{d-1})$ of the point inside the rectangle.

Assume we want to compute the indexes $i_0, ..., i_d$ of the $(d+1)$ vertices of the simplex containing a point defined by its relative coordinates $(x_0, ..., x_{d-1})$ with respect to the rectangle in which it belongs to. Let $\{\xi_0, ..., \xi_{2^d}\}$ be the corners of this d-rectangle. The indexes of the corners use the binary decomposition in dimension d, as illustrated in Figure 2. Computing these indexes is achieved by sorting the coordinates from the highest to the smallest: there exist indices $j_0, ..., j_{d-1}$, permutation of $\{0, .., d-1\}$, such that $1 \geq x_{j_0} \geq x_{j_1} \geq ... \geq x_{j_{d-1}} \geq 0$. Then the indices $i_0, ..., i_d$ of the $(d+1)$ vertices of the simplex containing the point are: $i_0 = 0$, $i_1 = i_0 + 2^{j_0}$, ..., $i_k = i_{k-1} + 2^{j_{k-1}}$, ..., $i_d = i_{d-1} + 2^{j_{d-1}} = 2^d - 1$. For example, if the coordinates satisfy: $1 \geq x_2 \geq x_0 \geq x_1 \geq 0$ (illustrated by the point x in Figure 2) then the vertices are: ξ_0 (every simplex contains this vertex, as well as $\xi_{2^d-1} = \xi_7$), ξ_4 (we added 2^2), ξ_5 (we added 2^0) and ξ_7 (we added 2^1).

Let us define the *barycentric coordinates* $\lambda_0, ..., \lambda_d$ of the point x inside the simplex $\xi_{i_0}, ..., \xi_{i_d}$ as the positive coefficients (uniquely) defined by: $\sum_{k=0}^{d} \lambda_k = 1$ and $\sum_{k=0}^{d} \lambda_k \xi_{i_k} = x$. Usually, these barycentric coordinates are expensive to compute; however, in the case of Kuhn triangulation these coefficients are simply: $\lambda_0 = 1 - x_{j_0}$, $\lambda_1 = x_{j_0} - x_{j_1}$, ..., $\lambda_k = x_{j_{k-1}} - x_{j_k}$, ..., $\lambda_d = x_{j_{d-1}} - 0 = x_{j_{d-1}}$. In the previous example, the barycentric coordinates are: $\lambda_0 = 1 - x_2$, $\lambda_1 = x_2 - x_0$, $\lambda_2 = x_0 - x_1$, $\lambda_3 = x_1$.