Least squares policy iteration (LSPI)

Charles Elkan
elkan@cs.ucsd.edu

December 6, 2012

1 Policy evaluation and policy improvement

Let π be a non-deterministic but stationary policy, so $p(a|s;\pi)$ is the probability of action a given state s, according to π. The function $Q^\pi : S \times A \to \mathbb{R}$, which is called the state-action value function of π, gives the expected total reward achieved by starting in state s, doing action a, and then acting according to the policy π afterwards. "Afterwards" means in every state after s.

The policy iteration method begins with an arbitrary policy π and repeats two steps until convergence:

1. Compute Q^π given π (policy evaluation).

2. Find a new, better π' based on Q^π (policy improvement).

Computing Q^π in Step 1 is formulated and solved as a set of simultaneous linear equations. Suppose that the state space S is finite, with $|S| = n$. Similarly, suppose that $|A| = m$. For each s and a, the value of $Q^\pi(s,a)$ is one unknown; there are nm unknowns in total. There are also nm equations. For a given s and a, the equation is

$$Q^\pi(s,a) = r(s,a) + \gamma \sum_{s'} [p(s'|s,a) \sum_{a'} p(a'|s';\pi) Q^\pi(s',a')]. \tag{1}$$

This equation is linear in the unknowns, because on the righthand side, the coefficient of $Q^\pi(s',a')$ is $p(s'|s,a)p(a'|s';\pi)$ which is a known constant. Since there are nm linear equations in nm unknowns, the solution is well-defined and, in general, unique.

In Step 2, the improved policy π' is defined as

$$p(a|s;\pi') = I(a = \text{argmax}_b\ Q^\pi(s,b)). \tag{2}$$

Note that π' is deterministic, and the equation above is equivalent to

$$\pi'(s) = \text{argmax}_b\ Q^\pi(s,b).$$

Policy iteration has the advantage that there is a clear stopping condition: when the function Q^π does not change after performing Step 1, then the algorithm terminates. The algorithm outputs both an optimal policy and its Q function. Either of these can be obtained easily from the other one. In general, the optimal policy is deterministic, but not unique. The optimal Q function is unique.

2 Restricting the space of policies

When the state space and/or action space is infinite, doing policy evaluation by solving a system of linear equations, as above, is impossible. One way to make progress is to consider only a restricted space of value functions. The most common choice is to consider only linear combinations of a fixed set of basis functions. In this restricted space, every state-action value function is of the form

$$Q(s,a) = \sum_{j=1}^{d} w_j \phi_j(s,a)$$

where each $\phi_j : S \times A \to \mathbb{R}$ is a basis function, each w_j is a real-valued coefficient, and d is the dimensionality of the restricted space. The basis functions are predefined and fixed, not learned. The final output of the LSPI algorithm is the coefficients w_j of a good value function. The value function of the true optimal policy is likely not in the restricted space, so in general LSPI can learn a good policy and Q function, but not optimal ones.

The righthand side of Equation (1) can be viewed as an operator T^π applied to the function Q^π. Given any state-action value function Q, $T^\pi[Q]$ is also a state-action value function. For s and a,

$$T^\pi[Q](s,a) = r(s,a) + \gamma \sum_{s'} p(s'|s,a) \sum_{a'} p(a'|s';\pi) Q(s',a').$$

The value function Q^π of π is the solution of the equation $Q = T^\pi[Q]$.

Most Q functions are not in the restricted space. Even if a particular Q is inside this space, the result $T^\pi[Q]$ of applying the operator is typically outside it. Therefore, the equation $Q = T^\pi[Q]$ cannot be solved inside the restricted space. Consider solving instead the equation $Q = \Pi[T^\pi[Q]]$, where Π is a projection operator. Being a projection operator means that for any function Q, $\Pi[Q]$ is the element of the restricted space that is closest to Q in the L_2 (least squares) sense.

Solving the equation $Q = \Pi[T^\pi[Q]]$ will yield a function that is approximately the value function of π, and is within the restricted space of representable value functions. Solving this equation is an approach to approximate policy evaluation. We want to solve the equation when the state and action spaces are infinite, but we start with the situation where they are finite, with cardinalities n and m as above.

We need to write the operator T^π as a matrix expression. We can view a Q function as a column vector of length nm containing a value for each state-action pair. Let $i = 1$ to $i = nm$ index these pairs, and write $(s, a) = i$ to indicate that s and a are the state and action corresponding to the index i.

- With this notation, R is a column vector with nm entries $r(i)$ where $i = (s, a)$ ranges over $S \times A$.
- Second, define a matrix P with nm rows and n columns. This matrix contains the transition probabilities of the MDP. Let j range over the state space S and let i range over $S \times A$ as before. The ij entry of the matrix P is $P_{ij} = p(j|i)$.
- Third, define a matrix L^π to represent the nondeterministic policy π. This matrix has n rows and nm columns. Let j range over S and let k range over $S \times A$. The jk entry of L^π is

$$L^\pi_{jk} = p((s, a) = k|j; \pi) = I(s = j)p(a|j; \pi).$$

The matrix L^π is an expanded, redundant, representation of the policy π. Each column of L^π corresponds to a specific state-action pair $k = (s, a)$. Each row of L^π corresponds to a state j. $L^\pi_{jk} = 0$ if the states j and s are not the same. If they are the same, then L^π_{jk} is the probability of a given s according to π. Note that if the policy π is deterministic, then the matrix L^π is all zero except for n entries that equal one. With the definitions above, the operator T^π is $T^\pi[Q] = R + \gamma P L^\pi Q$.

3 Solving the matrix equation

We want to solve the equation

$$Q = \Pi[T^\pi[Q]] = \Pi[R + \gamma P L^\pi Q].$$

Note that R and P do not depend on the given policy π, but L^π does.

Let Q be a value function that is inside the restricted space. We have $Q(s,a) = \sum_{j=1}^{d} w_j \phi_j(s,a)$. Let Φ be a matrix with nm rows and d columns where $\Phi_{ij} = \phi_j(i)$, the jth feature value of the ith state-action pair. We assume that the columns of this matrix are linearly independent, meaning that no basis function is a linear combination of other basis functions. Then $Q = \Phi w$ where w is the column vector of coefficients. Thus we want to solve the equation $\Phi w = \Pi[R + \gamma P L^\pi \Phi w]$.

To make progress, we need an explicit formula for the projection operator Π. Given any Q, that is a column vector of length nm, what is the weight vector w of the optimal projection of Q into the restricted space? We want w to minimize $||Q - \Phi w||_2$, the L_2 norm of the difference between Q and Φw. In other words, we want the solution with smallest squared error of the system of linear equations $\Phi w = Q$. This system has nm equations and d unknowns, so it is overdetermined, and no solution with zero error exists. The solution with smallest L_2 error is $w = \Phi^+ Q$ where Φ^+ is the pseudo inverse of Φ. The projection itself is the restricted value function $\Phi w = \Phi(\Phi^+ Q)$.

So, we want to find w that solves

$$\Phi w = \Pi[R + \gamma P L^\pi \Phi w] = \Phi \Phi^+ [R + \gamma P L^\pi \Phi w].$$

We can cancel Φ on both sides, because we have assumed that the columns of Φ are linearly independent, which implies that $\Phi x = 0$ if and only if the vector $x = 0$. We get

$$w = \Phi^+ [R + \gamma P L^\pi \Phi w].$$

The definition of the pseudo inverse is $\Phi^+ = (\Phi^T \Phi)^{-1} \Phi^T$. Putting this in gives

$$\begin{aligned} w &= (\Phi^T \Phi)^{-1} \Phi^T [R + \gamma P L^\pi \Phi w] \\ (\Phi^T \Phi) w &= \Phi^T [R + \gamma P L^\pi \Phi w] \\ \Phi^T (\Phi - \gamma P L^\pi \Phi) w &= \Phi^T R \\ w &= [\Phi^T (\Phi - \gamma P L^\pi \Phi)]^{-1} \Phi^T R. \end{aligned} \tag{3}$$

The matrix to be inverted has size d by d where d is the number of basis functions. The inverse exists unless the determinant is zero. The determinant is a polynomial in γ, so it is non-zero except for a finite number of values of γ.

4 Using training data

Equation (3) can be used to compute w if the matrices Φ, P, L^π and the vector R are known. However, the essence of reinforcement learning is that all that is known is training tuples. A straightforward idea is to use empirical versions of these matrices and vector. Suppose that training examples (s_t, a_t, r_t, s'_t) for $t = 1$ to $t = T$ are available. The empirical version of R is a column vector of length T with $R_t = r_t$. The empirical version of Φ is a T by d matrix, with $\Phi_{tj} = \phi_j(s_t, a_t)$.

We do not in fact need empirical versions of P and L^π. Instead, we only need an empirical version of $PL^\pi\Phi$, which is a matrix of the same shape as Φ. Consider Equation (1) again. Given (s, a) and j, the corresponding entry of $PL^\pi\Phi$ is

$$[PL^\pi\Phi]_{(s,a),j} = \sum_{s'} p(s'|s, a) \sum_{a'} p(a'|s'; \pi)\phi_j(s', a').$$

The corresponding entry of the empirical version of the matrix is obtained from (s_t, a_t, r_t, s'_t) as

$$[PL^\pi\Phi]_{tj} = \sum_{s'} p(s'|s_t, a_t) \sum_{a'} p(a'|s'; \pi)\phi_j(s', a').$$

This is the average value of the jth feature ϕ_j where the average is over all states s' and all actions a', with each s' weighted according to the probability that it follows (s_t, a_t), and a' weighted by its probability according to policy π. We can approximate this probability distribution by putting all the mass on the particular state s'_t observed to follow (s_t, a_t) in the ith training tuple. Moreover, let the policy π be deterministic and write $\pi(s')$ for the unique action specified by the policy for state s'. This gives the approximation $[PL^\pi\Phi]_{tj} = \phi_j(s'_t, \pi(s'_t))$.

With the approximation just made, we can evaluate Equation (3) using training tuples, without direct knowledge of the MDP. This is called the LSTDQ (least squares temporal differences Q function) algorithm. What this algorithm does is find the weights w of a restricted function Φw that approximates the state-action value function Q^π of the policy π. The algorithm has two remarkable advantages.

1. Empirical versions of R, Φ, and $PL^\pi\Phi$ can be computed from training tuples even when states and/or actions are continuous and multidimensional. Given these empirical versions, Equation (3) is applicable. The matrix to be inverted is d by d, where d is the number of basis functions, regardless of the number of training tuples, and regardless of the true dimensionality of the state space or of the action space.

2. Remember that the policy being evaluated is π. Given a fixed set of tuples (s_t, a_t, r_t, s'_t), the empirical vector R and matrix Φ are the same, regardless of π. The empirical version of $PL^\pi\Phi$ is computed using s'_t and π only. Since the a_t actions are not used in this computation, it does not matter what policy was used to produce them. Therefore, the same training set can be used to evaluate different policies π.

An algorithm with the second property just described is called an "off policy" learning algorithm. Such an algorithm can learn the value of a new policy based on data collected while using an old policy.

5 From LSTDQ to LSPI

Consider doing policy iteration with LSTDQ for the policy evaluation step. Finding the improved π' given Q^π is straightforward, using Equation (2) with no change. Concretely, the LSPI algorithm is as follows.

1. Obtain training tuples (s_t, a_t, r_t, s'_t) for $t = 1$ to $t = T$ using any policy to select each a_t action.

2. Initialize the weight vector $w \in \mathbb{R}^d$ randomly.

3. Policy improvement: Let π be the policy that is implicit in the Q function defined by w:
$$\pi(s) = \text{argmax}_a \; w \cdot \phi(s, a).$$

4. Policy evaluation: Use LSTDQ to compute a new vector w defining the approximate Q function of the policy π.

5. Repeat from Step 3, until convergence of w.

Step 4 involves solving the equation $\Phi^T(\Phi - \gamma PL^\pi\Phi)w = \Phi^T R$ where each training example yields one entry in the vector R and one row in each of the matrices Φ and $PL^\pi\Phi$. As explained above, R and Φ are fixed and can be computed just once, after Step 1, while $[PL^\pi\Phi]_{tj} = \phi_j(s'_t, b)$ where

$$b = \pi(s'_t) = \text{argmax}_a \; w \cdot \phi(s'_t, a).$$

A different way of describing the LSPI algorithm is as follows:

1. Obtain T training tuples (s_t, a_t, r_t, s'_t) generated using any policy. Set $R_t = r_t$ and $\Phi_{tj} = \phi_j(s_t, a_t)$.
2. Initialize the weight vector w randomly.
3. For each t, compute $a'_t = \text{argmax}_a \sum_{j=1}^{d} w_j \phi_j(s'_t, a)$.
4. For all t and k, compute $[PL^\pi\Phi]_{tk} = \phi_k(s'_t, a'_t)$. Find the solution w of the system of d linear equations $[\Phi^T(\Phi - \gamma PL^\pi\Phi)]w = \Phi^T R$.
5. Repeat from Step 3, until w converges.

As stated before, Step 4 is policy evaluation. Policy improvement is implicit in Step 3.