CE 273
Markov Decision Processes

Lecture 21

Approximation in Policy Space

Previously on Markov Decision Processes

We might sometimes be interested in the expected amount of time spent by the system in different states up to time n (e.g., parking).

Such metrics are called occupancy times. Let $V_j^{(n)}$ be the number of visits to j over $\{0, 1, \dots, n\}$. Mathematically, occupancy time of j up to time n starting from i is

$$m_{ij}^{(n)} = \mathbb{E}\big[V_j^{(n)} | X_0 = i\big], \forall\, i, j \in S, n \geq 0$$

The matrix of $m_{ij}^{(n)}$ values, also called the occupancy time matrix, is represented by

$$M^{(n)} = \big[m_{ij}^{(n)}\big]_{|S| \times |S|}$$

The occupancy times matrix can be computed from the transition matrix!

Theorem

Let $P^0 = I$. For a fixed n, $M^{(n)} = \sum_{r=0}^{n} P^r$

Previously on Markov Decision Processes

The D and Q matrices are shown in blue and green respectively.

$$P = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 1/4 & 0 & 1/2 & 0 & 1/4 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 1/16 & 1/16 & 1/4 & 1/8 & 1/4 & 1/4 \\ 0 & 1/4 & 0 & 0 & 1/4 & 1/2 \end{bmatrix} \end{array}$$

$$D^{(n)} \to \begin{array}{c|cc} & 1 & 2 \\ \hline 3 & 3/4 & 1/4 \\ 4 & 1/2 & 1/2 \\ 5 & 1/2 & 1/2 \\ 6 & 1/4 & 3/4 \end{array} \quad P^{(n)} \to \begin{array}{c|cccccc} & 1 & 2 & 3 & 4 & 5 & 6 \\ \hline 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 1 & 0 & 0 & 0 & 0 \\ 3 & 3/4 & 1/4 & 0 & 0 & 0 & 0 \\ 4 & 1/2 & 1/2 & 0 & 0 & 0 & 0 \\ 5 & 1/2 & 1/2 & 0 & 0 & 0 & 0 \\ 6 & 1/4 & 3/4 & 0 & 0 & 0 & 0 \end{array}$$

Previously on Markov Decision Processes

Given a policy μ, J_μ was the long run discounted cost of using the policy.

Similarly, we can define $Q_\mu(i,u)$ as the cost of using control u in state i and thereafter following policy μ.

The equations for finding $Q_\mu(i,u)$ parallel that of J_μ but without the minimization operator.

$$Q_\mu(i,u) = \sum_{j=1}^{n} p_{ij}(u)\Big(g(i,u,j) + \alpha J_\mu(j)\Big) \; \forall\, i = 1,\dots,n, u \in U(i)$$

The earlier expression $J_{k+1}(i) = \min_{u \in U(i)} Q_{k+1}(i,u)$ now becomes

$$J_\mu(j) = Q_\mu(j, \mu(j))$$

Can you interpret both sides of the the above equation in words?

Previously on Markov Decision Processes

Suppose for each state i, we extract m features. Let k represent a generic feature. Then, the vector of approximate value functions can be written as

$$\tilde{J} = \Phi r$$

where Φ is

$$\Phi = \begin{bmatrix} \phi_1(1) & \dots & \phi_m(1) \\ \phi_1(2) & \dots & \phi_m(2) \\ \vdots & \vdots & \vdots \\ \phi_1(n) & \dots & \phi_m(n) \end{bmatrix}_{n\times m} \qquad r = \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_m \end{bmatrix}_{m\times 1}$$

The rows of the Φ matrix are features and the columns can be interpreted as basis functions/vectors.

Thus, we can think of the subspace $S = \{\Phi r | r \in \mathbb{R}^m\}$ as the subspace spanned by the basis vectors (columns of Φ).

Previously on Markov Decision Processes

To address these issues, a Monte Carlo simulation method like the one discussed in the previous class can be used.

Suppose $\xi = (\xi_1, \xi_2, \ldots, \xi_n)$ is a probability distribution. Then the sum $\sum_{i=1}^{n} \xi_i a_i$ can be interpreted as the expectation of a random variable whose support is $a_1, \ldots, a_n$ with a pmf ξ.

A simulation-based approach to compute the above expectation is to sample from $a_1, \ldots, a_n$ according to the distribution ξ and form Monte Carlo averages.

Suppose $k = 1, \ldots, K$ are a set of states (not episodes or state transitions) sampled according to the distribution ξ, then

$$\sum_{i=1}^{n} \xi_i a_i \approx \frac{1}{K} \sum_{k=1}^{K} a_k$$

Previously on Markov Decision Processes

Consider a policy μ. Suppose we simulate S trajectories and each trajectory is indexed by s. A trajectory s can be written as

$$i_0, \mu(i_0), i_1, \mu(i_1), \ldots, i_t, \mu(i_t), \ldots, i_{t(s)}$$

where $i_{t(s)}$ represents the terminal state of trajectory s. Given this trajectory, we can compute the sample future cost at every time step $t = 0, \ldots, t(s)$ as follows

$$G_t(s) = g(i_t, \mu(i_t), i_{t+1}) + \alpha g(i_{t+1}, \mu(i_{t+1}), i_{t+2}) + \ldots \\ + \alpha^{t(s)-1-t} g(i_{t(s)-1}, \mu(i_{t(s)-1}), i_{t(s)})$$

For every i_t in the trajectory $i_0, \mu(i_0), i_1, \mu(i_1), \ldots, i_{t(s)}$, update

$$\text{numVisits}(i_t) \leftarrow \text{numVisits}(i_t) + 1$$
$$\tilde{J}_\mu(i_t) \leftarrow \tilde{J}_\mu(i_t) + \frac{1}{\text{numVisits}(i_t)}\Big(G_t(s) - \tilde{J}_\mu(i_t)\Big)$$

This method can be generalized as

$$\tilde{J}_\mu(i_t) \leftarrow \tilde{J}_\mu(i_t) + \gamma\Big(G_t(s) - \tilde{J}_\mu(i_t)\Big)$$

Previously on Markov Decision Processes

Mathematically, the MC update

$$\tilde{J}_\mu(i_t) \leftarrow \tilde{J}_\mu(i_t) + \gamma\Big(G_t(s) - \tilde{J}_\mu(i_t)\Big)$$

is transformed to

$$\tilde{J}_\mu(i_t) \leftarrow \tilde{J}_\mu(i_t) + \gamma\Big(g(i_t, \mu(i_t), i_{t+1}) + \alpha \tilde{J}_\mu(i_{t+1}) - \tilde{J}_\mu(i_t)\Big)$$

This method is also called the TD(0) algorithm, and

- $g(i_t, \mu(i_t), i_{t+1}) + \alpha \tilde{J}_\mu(i_{t+1})$ is called the TD target
- $g(i_t, \mu(i_t), i_{t+1}) + \alpha \tilde{J}_\mu(i_{t+1}) - \tilde{J}_\mu(i_t)$ is called the TD error

Lecture Outline

Lecture Outline

Introduction

Introduction
Motivation

So far we have looked at value-based approximations in which

- We parameterize the value function for a given policy and use 'policy improvement' and repeat.
- Try to find the value closest to J^* or Q^* using Q-learning or ALP.

Alternately, we can parametrize the policy directly and optimize some objective with respect to the parameters.

This is useful when for instance, threshold policies are optimal or if we seek a policy that is easier to explain and implement (e.g., (s, S)-policies for inventory and dynamic pricing for parking and airlines).

Introduction

Objective

Which objective among discounted, total, average cost problems is most amenable to such parametric policy optimization approach?

The value function of discounted and total cost problems is dependent on the initial state.

$$J_{\mu}(x_0) = \lim_{N\to\infty} \mathbb{E}_w \left\{ \sum_{k=0}^{N-1} \alpha^k g(x_k, \mu(x_k), w_k) \right\}$$

One can solve a series of parametric policy optimization problems for each initial state, but such an exercise is not scaleable.

Introduction

Objective

For this lecture, we will assume that the system always starts in some initial state and there is a zero-cost terminal state(s), i.e., it is an episodic MDP with $\alpha = 1$.

For reference, think of Tetris which always starts with an empty board and there are many states at which the game terminates.

Luckily, the results extend to the discounted cost case and average cost problems but deriving the exact expressions is a bit tedious.

Introduction

Stochastic Policies

Imagine a state x_0 from which the system always starts. We are interested in minimizing over all admissible policies, the following function

$$J_\mu(x_0) = \lim_{N \to \infty} \mathbb{E}_w \left\{ \sum_{k=0}^{N-1} g(x_k, \mu(x_k), w_k) \right\}$$

When we optimize $J_\mu(x_0)$ over all possible policies, for finite-state and finite-action problems, we have a discrete optimization problem with potentially an exponential number of options.

Suppose, we have a parameterized policy $\mu_\theta \in \mathbb{R}^m$, then we can write the objective as $\min_{\theta \in \mathbb{R}^m} J_{\mu_\theta}(x_0)$. This is identical to searching for rs instead of the optimal value functions J. How can we solve this problem?

The problem is now reduced to optimization of a function with respect to fewer variables (typically unconstrained). So one could use standard gradient methods. What are some potential issues with this approach?

Introduction
Stochastic Policies

If the actions are discrete, restrictions must be imposed on θ as well.

- For example, consider an inventory policy which suggests to order $S + i\theta$, when the state of the system is i.
- If we are dealing with indivisble goods, we cannot let θ be continuous.

For this reason, we smoothen the objective by considering stochastic policies instead of deterministic ones.

Introduction

Stochastic Policies

Specifically, we use any class of stochastic policies which guarantee that $\nabla_\theta \mu_\theta$ is differentiable and belongs to $(0, 1)$.

But there always exists a deterministic policy which we have been trying to find? How do we find such a policy using the above method? We don't. Since policies are anyways being approximated, we hope to weigh the optimal ones more.

There are however instances such as POMDPs in which the optimal policies are stochastic where this method is ideal. For example, rock paper scissors and Poker. Stochastic policies also help in establishing theoretical results.

Introduction

Softmax Policies

We modify the notation $\mu(i)$ to $\mu(i,u)$ to represent the probability of choosing u when in state i. To denote the dependence on the parameter θ, we write $\mu_\theta(i,u)$.

A standard way to encode such stochastic policies is to use a softmax policy. Imagine $f(i,u;\theta)$ denotes some kind of numerical preference/utility of u over all the other $u' \in U(i)$. The softmax policy is defined as

$$\mu_\theta(i,u) = \frac{\exp\big(f(i,u;\theta)\big)}{\sum_{u'} \exp\big(f(i,u;\theta)\big)}$$

This is identical to the multinomial logit model! We also explicilty parameterize $f(i,u;\theta)$ using a linear architecture

$$f(i,u;\theta) = \theta^T \psi(i,u)$$

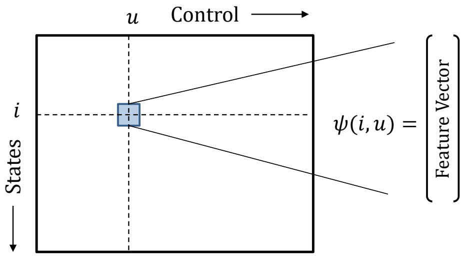
where $\psi(i,u)$ is a feature vector for state-action pair (i,u). Notice that the order of writing parameters and features is flipped. It is more convenient this way since we will calculate derivatives.

Introduction

Softmax Policies

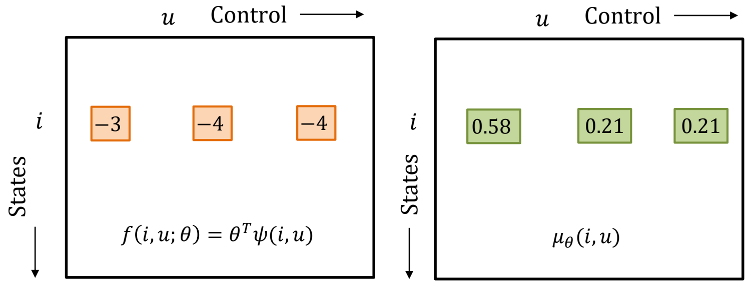
For instance a vector of features for Tetris could be

- Constant
- Number of lines cleared because of choosing u.
- Difference in column heights before and after taking control u in state i.



Introduction

Softmax Policies



Notice that higher the value of f of an action, the odds of choosing it are greater.

Why not use the negative of the Q-values instead of f?

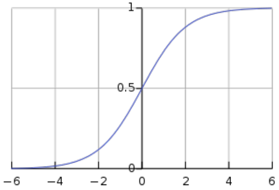
- On the outset, it appears that they could be used but the Q-factors have a definite meaning.
- The one with the lowest value was choosen earlier. But this doesn't mean its probability gets driven to 1 in the softmax policy.
- In the earlier example, if the Q-factors were 3, 4 and 4, We should choose action 1, but softmax places a sizeable weight on the other two options.
- The function f on the other hand can be pushed to large values to make the probability close to 1 or 0. For example, in the above problem, we might be able find a θ during the course of optimization for which f takes values 9999, 1, and 1.

Thus, using f can mimic deterministic policies but Q-factors cannot.

Introduction

Softmax Policies

Softmax policies are between $(0, 1)$ but are they differentiable?



That is, does the following derivative exist?

$$\begin{bmatrix} \frac{\partial}{\partial \theta_1} \mu_\theta(i, u) \\ \frac{\partial}{\partial \theta_2} \mu_\theta(i, u) \\ \vdots \\ \frac{\partial}{\partial \theta_m} \mu_\theta(i, u) \end{bmatrix}$$

Substituting the preferences in the expression for the softmax distriution,

$$\mu_\theta(i, u) = \frac{\exp\left(\theta^T \psi(i, u)\right)}{\sum_{u'} \exp\left(\theta^T \psi(i, u')\right)}$$

Using the above expression calculate $\nabla_\theta \ln \mu_\theta(i, u)$? We will use this later.

Introduction

Softmax Policies

$$
\begin{aligned}
\nabla_\theta \ln \mu_\theta(i,u) &= \nabla_\theta \ln \frac{\exp\big(\theta^T \psi(i,u)\big)}{\sum_{u'} \big(\exp \theta^T \psi(i,u')\big)} \\
&= \nabla_\theta \ln \exp\big(\theta^T \psi(i,u)\big) - \nabla_\theta \ln \Big(\sum_{u'} \big(\exp \theta^T \psi(i,u')\big)\Big) \\
&= \psi(i,u) - \frac{1}{\sum_{u'} \big(\exp \theta^T \psi(i,u')\big)} \nabla_\theta \Big(\sum_{u'} \big(\exp \theta^T \psi(i,u')\big)\Big) \\
&= \psi(i,u) - \frac{1}{\sum_{u'} \big(\exp \theta^T \psi(i,u')\big)} \Big(\sum_{u'} \psi(i,u') \big(\exp \theta^T \psi(i,u')\big)\Big) \\
&= \psi(i,u) - \sum_{u'} \mu_\theta(i,u') \psi(i,u')
\end{aligned}
$$

Lecture Outline

Policy Gradient

Policy Gradient

Introduction

Let us revisit our original problem. We want to find θ to minimize

$$\min_{\theta \in \mathbb{R}^m} J_{\mu_\theta}(x_0)$$

Standard gradient descent approaches involve finding iterates

$$\theta^{k+1} = \theta^k - \eta_k \nabla_\theta J_{\mu_\theta}(x_0)$$

Expanding $J_{\mu_\theta}(x_0)$, in $\nabla_\theta J_{\mu_\theta}(x_0)$,

$$\nabla_\theta J_{\mu_\theta}(x_0) = \nabla_\theta \mathbb{E}_w \left\{ \sum_{k=0}^{\infty} g(x_k, \mu_\theta(x_k), w_k) \right\}$$

As before, we can think of replacing the expectation with simulated samples, but how do we deal with the ∇_θ operator that appears before the expectation?

Can we make it look like the expectation of something else? Yes.

Policy Gradient

Introduction

For brevity, we'll use i to denote the starting state instead of x_0.

Proposition

Let ξ_j denote the average amount of time spent in state j before terminating.

$$\nabla_\theta J_{\mu_\theta}(i) \propto \sum_{j=1}^{n} \xi_j \sum_{v \in U(j)} Q_{\mu_\theta}(j, v) \nabla_\theta \mu_\theta(j, v)$$

Why is this an expectation? The RHS can be written as

$$\mathbb{E}_j \left\{ \sum_{v \in U(j)} Q_{\mu_\theta}(j, v) \nabla_\theta \mu_\theta(j, v) \right\}$$

We can evaluate this using simulation, but we still have a sum to take care of.

$$\begin{aligned} \mathbb{E}_j \left\{ \sum_{v \in U(j)} \mu_\theta(j, v) Q_{\mu_\theta}(j, v) \frac{\nabla_\theta \mu_\theta(j, v)}{\mu_\theta(j, v)} \right\} &= \mathbb{E}_{j,v} \left\{ Q_{\mu_\theta}(j, v) \frac{\nabla_\theta \mu_\theta(j, v)}{\mu_\theta(j, v)} \right\} \\ &= \mathbb{E}_{j,v} \left\{ Q_{\mu_\theta}(j, v) \nabla_\theta \ln \mu_\theta(j, v) \right\} \end{aligned}$$

Policy Gradient

Policy Gradient Theorem

$$\mathbb{E}_{j,v}\left\{ Q_{\mu_\theta}(j,v)\nabla_\theta \ln \mu_\theta(i,u) \right\}$$

Given a policy, we know how to simulate its value function $Q_{\mu_\theta}(j,v)$ and from our previous discussion $\nabla_\theta \ln \mu_\theta(i,u)$ is known in closed form!

Proof.

Recall from the definitions of Q-factors,

$$J_\mu(i) = Q_\mu(i, \mu(i))$$

Since we are using a stochastic policy μ_θ,

$$J_{\mu_\theta}(i) = \left\{ \sum_{u \in U(i)} \mu_\theta(i,u) Q_{\mu_\theta}(i,u) \right\}$$

Differentiating both sides wrt θ,

$$\nabla_\theta J_{\mu_\theta}(i) = \sum_{u \in U(i)} \left\{ Q_{\mu_\theta}(i,u)\nabla_\theta \mu_\theta(i,u) + \mu_\theta(i,u) \nabla_\theta Q_{\mu_\theta}(i,u) \right\}$$

Policy Gradient

Policy Gradient Theorem

Proof.

Again, by definition, $Q_\mu(i,u) = \sum_{j=1}^n p_{ij}(u)\big(g(i,u,j) + J_\mu(j)\big)$. Hence,

$$\Rightarrow Q_{\mu_\theta}(i,u) = \sum_{j=1}^{n} p_{ij}(u)\big(g(i,u,j) + J_{\mu_\theta}(j)$$

$$\Rightarrow \nabla_\theta Q_{\mu_\theta}(i,u) = \sum_{j=1}^{n} p_{ij}(u)\nabla_\theta J_{\mu_\theta}(j)$$

Substituting the above expression in the earlier equation,

$$\nabla_\theta J_{\mu_\theta}(i) = \sum_{u\in U(i)} \left\{ Q_{\mu_\theta}(i,u)\nabla_\theta \mu_\theta(i,u) + \mu_\theta(i,u)\sum_{j=1}^{n} p_{ij}(u)\nabla_\theta J_{\mu_\theta}(j)\right\}$$

The same equation can be used recursively to expand J_{μ_θ}.

Policy Gradient

Policy Gradient Theorem

Proof.

$$\nabla_\theta J_{\mu_\theta}(i) = \sum_{u \in U(i)} \Big\{ Q_{\mu_\theta}(i,u) \nabla_\theta \mu_\theta(i,u) + \mu_\theta(i,u) \sum_{j=1}^{n} p_{ij}(u)$$

$$\sum_{v \in U(j)} \Big\{ Q_{\mu_\theta}(j,v) \nabla_\theta \mu_\theta(j,v) + \mu_\theta(j,v) \sum_{k=1}^{n} p_{jk}(v) J_{\mu_\theta}(k) \Big\} \Big\}$$

Policy Gradient

Policy Gradient Theorem

Proof.

Let's recolor this to discover some structure

$$\nabla_\theta J_{\mu_\theta}(i) = \sum_{u\in U(i)} \Big\{ Q_{\mu_\theta}(i,u)\nabla_\theta \mu_\theta(i,u) + \mu_\theta(i,u)\sum_{j=1}^{n} p_{ij}(u)$$

$$\sum_{v\in U(j)} \Big\{ Q_{\mu_\theta}(j,v)\nabla_\theta \mu_\theta(j,v) + \mu_\theta(j,v)\sum_{k=1}^{n} p_{jk}(v) J_{\mu_\theta}(k) \Big\}\Big\}$$

The purple and teal parts can be rewritten as

$$\sum_{j=1}^{n} p_{ij}^{(0)}(\mu_\theta) \sum_{v\in U(j)} \Big\{ Q_{\mu_\theta}(j,v)\nabla_\theta \mu_\theta(j,v) \Big\}$$

$$\sum_{j=1}^{n} \sum_{u\in U(i)} \mu_\theta(i,u) p_{ij}(u) \sum_{v\in U(j)} \Big\{ Q_{\mu_\theta}(j,v)\nabla_\theta \mu_\theta(j,v) \Big\}$$

$$= \sum_{j=1}^{n} p_{ij}^{(1)}(\mu_\theta) \sum_{v\in U(j)} \Big\{ Q_{\mu_\theta}(j,v)\nabla_\theta \mu_\theta(j,v) \Big\}$$

Policy Gradient

Policy Gradient Theorem

Proof.

Generalizing these ideas,

$$\begin{aligned}\nabla_\theta J_{\mu_\theta}(i) &= \sum_{j=1}^{n}\sum_{k=0}^{\infty} p_{ij}^{(k)}(\mu_\theta) \sum_{v\in U(j)} \left\{ Q_{\mu_\theta}(j,v)\nabla_\theta \mu_\theta(j,v) \right\} \\ &= \sum_{j=1}^{n} \eta_j \sum_{v\in U(j)} \left\{ Q_{\mu_\theta}(j,v)\nabla_\theta \mu_\theta(j,v) \right\}\end{aligned}$$

where η_j is the expected number of visits to state j over the infinite horizon starting from i. (Same as $m_{ij}^{(\infty)}$ that we saw in DTMCs).

The sum on the RHS includes all states including terminal states. η for this states could be ∞. Is this okay? η_j is proportional to ξ_j. Hence,

$$\nabla_\theta J_{\mu_\theta}(i) \propto \sum_{j=1}^{n} \xi_j \sum_{v\in U(j)} Q_{\mu_\theta}(j,v)\nabla_\theta \mu_\theta(j,v)$$

■

Why is the proportionality constant not important?

Policy Gradient
REINFORCE

In summary, θ can be updated using a gradient-descent method

$$\theta^{k+1} = \theta^k - \eta_k \nabla_\theta J_{\mu_\theta}(x_0)$$

and the policy gradient theorem guarantees that

$$\nabla_\theta J_{\mu_\theta}(i) \propto \mathbb{E}_{j,v}\left\{ Q_{\mu_\theta}(j,v) \nabla_\theta \ln \mu_\theta(j,v) \right\}$$

Recall that for the softmax policy,

$$\nabla_\theta \ln \mu_\theta(j,v) = \psi(j,v) - \sum_{v'} \mu_\theta(j,v')\psi(j,v')$$

To simulate the above expectation, we can use a MC-like method in which we follow policy μ_θ for a full sample episode and for the state-action pair visited at step k, we calculate G_k (the sampled future cost) and $\nabla_\theta \ln_{\mu_\theta}(j_k, v_k)$ (using the above formula) and update θ

$$\theta^{k+1} = \theta^k - \eta_k G_k \nabla_\theta \ln \mu_\theta(j_k, v_k)$$

Policy Gradient
REINFORCE

The above algorithm is popularly called REINFORCE. The term $\nabla_\theta \ln \mu_\theta(i, u)$ is also called the *eligibility* or *score* vector.

- Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine learning, 8(3-4), 229-256.

Lecture Outline

Extensions

Extensions

Introduction

The policy gradient theorem is theoretically appealing and can be extended to the discounted and average cost problems. The REINFORCE algorithm however has a few disadvantages:

- The algorithm may get stuck at a local optima and has all the drawbacks of gradient descent.
- MC methods introduce large variance and the learning rate is usually slow.

Extensions

Options

There have been several improvements to address the issues of vanilla-policy gradient methods.

- Derivative-free methods:
  Start with a guess θ and perturb it to $\theta + \epsilon$ and estimate the value functions of the Markov chains and construct approximations to the derivative.
- Cross-entropy methods:
  The parameters are drawn from a Gaussian distribution and using simulation the mean and variance of the distribution are adjusted. Think of them as being similar to mixed-logit models.
- Natural Gradient:
  This method is like Newton's descent where the gradient vector is adjusted with the inverse of another matrix.

Extensions

Applications

Policy gradient methods have been widely used in robotics.

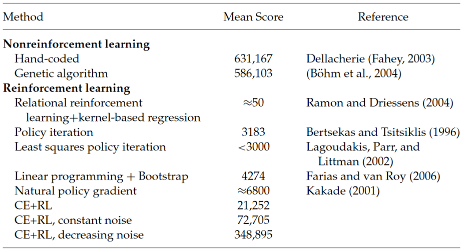
References:

- Kohl, N., & Stone, P. (2004, May). Policy gradient reinforcement learning for fast quadrupedal locomotion. In Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on (Vol. 3, pp. 2619-2624). IEEE.
- Kober, J., & Peters, J. R. (2009). Policy search for motor primitives in robotics. In Advances in neural information processing systems (pp. 849-856).

Extensions

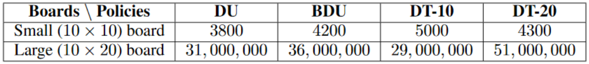
Applications

- Szita, I., & Lörincz, A. (2006). Learning Tetris using the noisy cross-entropy method. Neural computation, 18(12), 2936-2941.



Method	Mean Score	Reference
Nonreinforcement learning		
Hand-coded	631,167	Dellacherie (Fahey, 2003)
Genetic algorithm	586,103	(Böhm et al., 2004)
Reinforcement learning		
Relational reinforcement learning+kernel-based regression	≈50	Ramon and Driessens (2004)
Policy iteration	3183	Bertsekas and Tsitsiklis (1996)
Least squares policy iteration	<3000	Lagoudakis, Parr, and Littman (2002)
Linear programming + Bootstrap	4274	Farias and van Roy (2006)
Natural policy gradient	≈6800	Kakade (2001)
CE+RL	21,252	
CE+RL, constant noise	72,705	
CE+RL, decreasing noise	348,895	

- Gabillon, V., Ghavamzadeh, M., & Scherrer, B. (2013). Approximate dynamic programming finally performs well in the game of Tetris. In Advances in neural information processing systems (pp. 1754-1762).



Boards \ Policies	DU	BDU	DT-10	DT-20
Small (10×10) board	3800	4200	5000	4300
Large (10×20) board	$31,000,000$	$36,000,000$	$29,000,000$	$51,000,000$

Table 1: Average (over $10,000$ games) score of DU, BDU, DT-10, and DT-20 policies.

Extensions

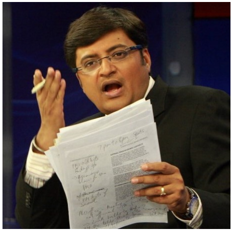
Actor-Critic Methods

The REINFORCE algorithm can be modified using a TD(0) method by replacing G_k with one-step costs and an approximate value function at the future step

$$\theta^{k+1} = \theta^k - \eta_k G_k \nabla_\theta \ln \mu_\theta(j_k, v_k)$$

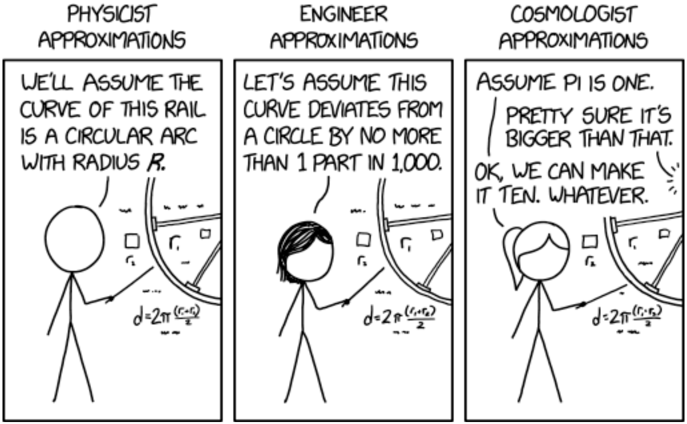
The approximate value function depends on the current policy and hence can be parameterized and simulated. Specifially, we can define $Q_{\mu_\theta}(i, u) = \phi(i, u)^T r$, where r and θ can be completely different set of features.

(a) Policy Gradient


(b) Value Function Approximator

The policy μ_θ is called the **actor** who updates θ by policy gradient and the value function evaluator is called the **critic** who uses TD methods.

Your Moment of Zen


PHYSICIST APPROXIMATIONS
WE'LL ASSUME THE CURVE OF THIS RAIL IS A CIRCULAR ARC WITH RADIUS R.
ENGINEER APPROXIMATIONS
LET'S ASSUME THIS CURVE DEVIATES FROM A CIRCLE BY NO MORE THAN 1 PART IN 1,000.
COSMOLOGIST APPROXIMATIONS
ASSUME PI IS ONE.
PRETTY SURE IT'S BIGGER THAN THAT.
OK, WE CAN MAKE IT TEN. WHATEVER.
d=2π(r₁+r₂)/2