ESE 680-004: Learning and Control **Fall 2019**

Lecture 20: Model Free Methods 3

Lecturer: Nikolai Matni *Scribes: Walker Gosrich*

1 Introduction

So far in the class, we have looked at several methods for solving reinforcement learning problems, including model-based and model-free approaches. Here, we will continue our discussion of model-free approaches, by considering *direct policy search.*

First, we'll look at Section 3.3 of [3] for a discussion of direct policy search and an introduction to the REINFORCE algorithm, a simple and highly versatile algorithm for derivative-free (function evaluation-based) optimization of unconstrained problems. We'll also be introduced to Policy Gradient and Pure Random Search methods, from Section 4 of the same. Then we'll briefly see how these approaches compare, applied to a simple test case: the LQR.

After this, we'll turn to [4] for a look at the theoretical properties and the "optimization landscape" of model-free methods applied to LQR. We'll analyze both the model-based and model-free cases of this problem, prove convergence to the optimal policy, and characterize the sample-complexity of this convergence.

2 REINFORCE Algorithm

First, we'll develop a simple and very general algorithm, typically called REINFORCE, that is used to solve unconstrained optimization problems through function evaluations.

Let's begin by considering the generic unconstrained optimization problem:

$$\text{minimize}_{c\in\mathbb{R}_d} C(z) \tag{1}$$

where we aim to find the value of z that minimizes our cost function. We claim that the problem of finding the minimizing value of z is equivalent to the problem of finding the minimizing *probability distribution* over z:

$$\text{minimize}_{p(z)} \mathbb{E}_p\left[C(z)\right]$$

If z^* is the minimizing argument of $C(z)$, then we can do at least as well by choosing a probability distribution: we can achieve the same value of $C(z)$ by choosing the δ-function around z^*.

Furthermore, we can do no better than a fixed policy by applying a probability distribution. To see this, note that for a fixed dsitribution p, we must have $\mathbb{E}_p[C(z)] \geq \min_z C(z)$. Because p is an arbitrary distribtion, we know that $\min_p \mathbb{E}_p[C(z)] \geq \min_z C(z)$ also holds. So we can do at least as well as our fixed value, and can do no better: the two problems are equivalent, and we can choose: optimize over z or *distributions* of z.

The space of distributions over z is infinite-dimensional, so we typically restrict ourselves to a subset of the space that we can efficiently optimize over. A typical approach is to parameterize by a vector θ, considering the family of density functions $p(z;\theta)$. We now have the following problem:

$$\text{minimize}_{\theta} \mathbb{E}_{p(z;\theta)}[C(z)] \tag{2}$$

Note that by restricting ourselves to this family of probability density functions, we have upper-bounded our performance on the problem. Our parameterization may contain all of the δ-functions, in which case we

could reach the same performance as the original optimization. But our family of distributions will likely not contain all δ-functions, and so we will likely perform sub-optimally.

But our reformulation has a purpose. Now, we can compute the derivative of $\mathbb{E}_{p(z;\theta}[C(z)]$ using a method called the *log likelihood trick*. We define the cost $J(\theta) := \mathbb{E}_{p(z;\theta)}[C(z)]$, and examine the derivative with respect to θ: $\nabla_\theta J(\theta)$. We perform the following manipulation:

$$\begin{aligned}
\nabla_\theta J(\theta) &= \int C(z) \nabla_\theta p(z;\theta) dz \\
&= \int C(z) \frac{\nabla_\theta p(z;\theta)}{p(z;\theta)} p(z;\theta) dz \\
&= \int [C(z) \nabla_\theta \log\, p(z;\theta)] p(z;\theta) dz \\
&= \mathbb{E}_{p(z;\theta)}[C(z) \nabla_\theta \log\, p(z;\theta)]
\end{aligned}$$

This tells us that the gradient of $J(\theta)$ is the expected value of the function

$$G(z,\theta) = C(z) \nabla_\theta \log\, p(z;\theta) \tag{3}$$

We can sample z from $p(z;\theta)$, and evaluate the function $G(z,\theta)$ to give us an unbiased estimate of the gradient of $J(\theta)$. Using these function evaluations, we can follow the gradient and perform stochastic gradient descent on $J(\theta)$. This approach gives us the REINFORCE algorithm [1] - Algorithm 1.

Algorithm 1: REINFORCE

Hyperparameters: step sizes $\alpha_j > 0$
Initialize: $\theta_0, k = 0$;
while *ending condition not satisfied* **do**
  sample $z_k \sim p(z;\theta_k)$;
  set $\theta_{k+1} \leftarrow \theta_k - \alpha_k C(z_k) \nabla_\theta \log\, p(z_k;\theta_k)$;
  $k \leftarrow k+1$;
end

The main benefit of Algorithm 1 is that it is incredible simple to implement. If you can sample efficiently from $p(z;\theta)$, then you can run the algorithm on essentially any problem. However, this generality comes at a cost: we now only access our cost function $C(z)$ through function evaluations. Because we've used the log-likelihood trick, we are using a derivative-free optimization method, and can not achieve the same performance as methods that compute actual gradients. This performance gap is exacerbated when the function evaluations are noisy. Another drawback to this approach is that our choice of probability distribution can lead to high variance of stochastic gradients. High variance requires more samples to be drawn in order to find a minima or maxima. In other words, sample complexity increases.

Although the approach has a few drawbacks, the simplicity of implementation is often valuable enough to justify its use. There are two primary applications of this sort of stochastic search approach in reinforcement learning: policy gradient and pure random search.

3 Policy Gradient

In reinforcement learning, our goal is typically to minimize some cost (or conversely, maximize a reward subject to system dynamics. This is stated as follows:

$$\text{minimize } \mathbb{E}\left[\sum_{t=0}^{N} C_t(x_t, u_t)\right] \text{ subject to } x_{t+1} = f_t(x_t, u_t, e_t) \tag{4}$$

We can see from Bellman's equation that the optimal policy for the above problem is always deterministic. However, for policy gradient, we relax our search to probabilistic policies. This isn't too unreasonable: these policies are optimal for other sorts of problems, such as MDPs or zero-sum games.

Let's consider parametric, randomized policies such that u_t is sampled from a distribution $p(u|\tau_t;\theta)$, where $\tau_t = \{x_{0:t}, u_{0:t-1}\}$, the observed trajectory up to time t. This probabilistic policy induces a probability distribution over trajectories:

$$p(\tau;\theta) = \prod_{t=0}^{N-1} p(x_{t+1}|x_t;u_t)p(u_t|\tau_t;\theta)$$

Further, if we overload notation, and define the cost over a trajectory as follows:

$$C(\tau) = \sum_{t=0}^{N-1} C(x_t, u_t)$$

then we can write Equation 4 as the following optimal control problem:

$$\text{minimize}_\theta \mathbb{E}_{p(\tau;\theta)}[C(\tau)] \tag{5}$$

which is identical to Equation 2! Policy gradient proceeds by sampling a trajectory using the probabilistic policy, and updating using REINFORCE to find the solution.

We can verify that the gradient of $J(\theta) = \mathbb{E}_{p(\tau,\theta)}[C(\tau)]$ is not an explicit function of the dynamics:

$$\begin{aligned}
\nabla_\theta J(\theta) &= \mathbb{E}_{p(\tau;\theta)}[C(\tau)\nabla_\theta \log\, p(\tau;\theta)] \\
&= \mathbb{E}_{p(\tau;\theta)}[C(\tau)\nabla_\theta \left[\sum_{t=0}^{N-1} \log\, p(x_{t+1}|x_t,u_t) + \sum_{t=0}^{N-1} \log\, p(u_t|\tau_t;\theta)\right] \\
&= \sum_{t=0}^{N-1} \mathbb{E}_{p(\tau;\theta)}[C(\tau)\nabla_\theta \log\, p(u_t|\tau_t;\theta)]
\end{aligned}$$

This expectation can then be approximated by applying REINFORCE. This should not be surprising though: by shifting to distribution over policies, we push the burden of optimization onto the sampling procedure.

4 Pure Random Search

An older and more widely applied approach to solve Equation 1 is to directly perturb the current decision variable z with random noise, and then update the model based on the received reward at this perturbed value. Again, we can apply the REINFORCE algorithm without any knowledge of the underlying dynamics. In effect, applying reinforce is equivalent to approximate gradient descent of $C(z)$. We consider drawing random perturbations ϵ according to some distribution - most simply a uniform or normal distribution. We compute the following approximate gradient step update:

$$\theta_{t+1} = \theta_t - \alpha g_\sigma(\theta_t)$$

where

$$g_\sigma(\theta) = \frac{C(\theta+\sigma\epsilon) - C(\theta-\sigma\epsilon)}{2\sigma}\epsilon$$

is a finite difference approximation to the gradient along direction ϵ. So this approach steps along the gradient in direction ϵ. We can reduce the variance of these gradient estimates by averaging over multiple random directions:

$$g_\sigma^{(m)} = \frac{1}{m}\sum_{i=1}^{m} \frac{C(\theta + \sigma\epsilon_i) - C(\theta - \sigma\epsilon_i)}{2\sigma}\epsilon_i$$

The pure random search method is simpler to implement than the policy search method, but uses less of the structure of the problem. It is difficult to say which approach is better without selecting a specific problem to which to apply them. In the next section, we'll look at how these algorithms perform on linear quadratic regulator problems.

Side note: in [2], Mania, Guy, and Recht used a pure random search of static linear policies to take on some of the benchmarks in OpenAI Gym. They managed to beat the state of the art, using orders of magnitude less training data. The (intended) takeaway is that these benchmark tasks in OpenAI Gym and other environments aren't all that hard - after all, one of the simplest canonical algorithms can perform really well! But a lot of the field had a different takeaway: we should be using more of random search and linear policies!

5 Application to LQR

To compare the above approaches, let's take a look at the simplest nontrivial problem that can help distinguish between them. In control, this is the linear quadratic regulator: linear dynamics, and convex rewards/costs. In fact, let's start with an even simpler example with no dynamics:

$$\text{minimize } C(u) = ||u||_2^2, \text{ for } u \in \mathbb{R}^d$$

Applying policy gradient, we parameterize our policy as

$$p(u;\theta) = \mathcal{N}(\theta, \sigma^2 I)$$

Then we can evaluate the formula in Equation 2:

$$\mathbb{E}_{p(u;\theta)}[C(u)] = ||\theta||_2^2 + \sigma^2 d$$

As we are simply minimizing the square of the norm of our input, the best place to start is clearly $\theta_* = 0$. This will still leave us $\sigma^2 d$ off from optimal, but it will give us a good starting guess.

As a function of θ, the cost is strongly convex. Also, from what we've seen in our analysis of stochastic gradient descent, we need bounds on the (expected) norm of the gradient. If we draw $u \sim \mathcal{N}(\theta_0, \sigma^2 I)$, we can compute $G(u, \theta_0)$, our approximate gradient from Equation 3.

$$G(u,\theta) = C(u)\nabla_\theta \log p(u;\theta) = -\frac{||w - \theta_0||_2^2 w}{\sigma^2}, w \sim \mathcal{N}(0, \sigma^2 I)$$

We can show that the expected norm of this gradient scales as $O(\sigma d^{3/2} + \frac{d^{1/2}||\theta_0||_2}{\sigma})$, a non-trivial scaling with dimension. Interestingly, most bounds analyzing the convergence of these methods scale with the largest magnitude that the cost function can possible take. If you start with a cost function taking values in [0,1] and add 10^6, the running time may increase by 10^6, even though you haven't fundamentally changed the problem.

As another case, let's take a look at the classic problem of a discrete-time double integrator with the dynamical model

$$x_{t+1} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} x_t + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u_t$$

Such a system could model the position (first state) and velocity (second state) of a unit mass object under force u. As an instance of LQR, we can try to steer this system to reach point - from initial condition $x_0 = [-1, 0]$ without expending much force:

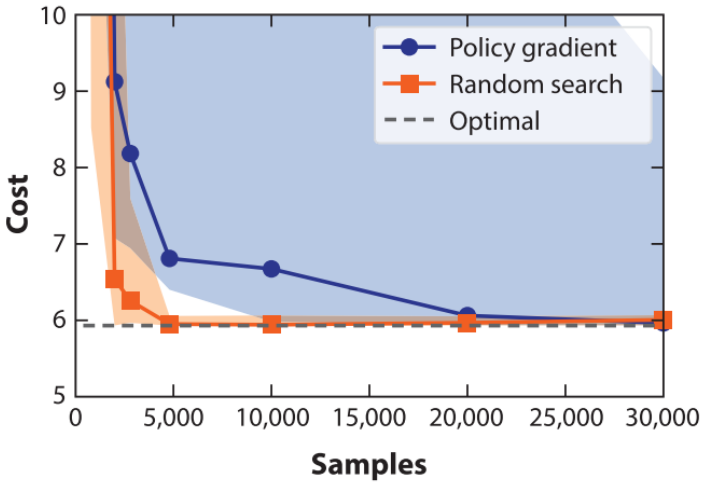

Figure 1: Cost for the double-integrator model for various reinforcement learning algorithms. The solid plots denote the median performance, and the shaded regions capture the maximum and minimum performance.

$$Q = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, R = r_0$$

for some scalar r_0. Even in this simple case, there is an element of control design: changing r_0 changes the character of the control law, balancing control energy against time required to reach the destination.

To compare the different approaches, the author of [3] ran experiments on this instance with a small amount of noise (e_t zero mean with covariance $10^{-4}I$) and training episode length $L = 10$. The goal was to design a controller that works on an arbitrarily long time horizon using the fewest number of simulations of length L.

To compare with policy search, the author restricts to policies that use a static, linear gain, as would be optimal on an infinite time horizon. The author used the Adam algorithm to shape the iterates, and subtracted the mean reward of previous iterates, a popular baseline subtraction heuristic to reduce variance. Without these algorithmic changes, the author was unable to get policy gradient to converge.

Figure 1 from [3] is reproduced here, showing the results from this experiment.

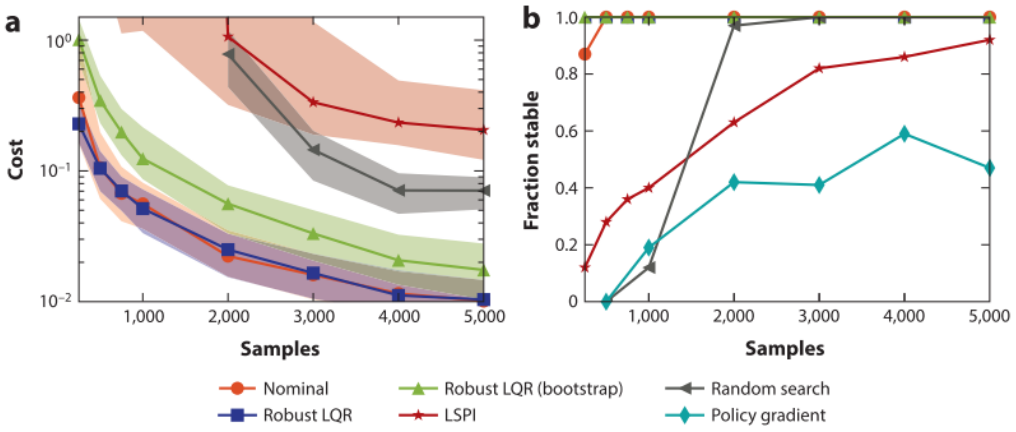

Figure 2: (a) Cost for the Laplacian model for varied models over 5,000 iterations. (b) The fraction of the time that the synthesized control strategy returns a stabilizing controller. Abbreviations: LQR, linear quadratic regulator; LSPI, least squares policy iteration

The author of [3] also considered a more complex instance of LQR: unstable Laplacian dynamics. An idealized instance of data canter cooling, a popular application of RL, with three heat sources and cooling devices, is modeled by the following linear dynamical system:

$$x_{t+1} = \begin{bmatrix} 1.01 & 1.01 & 0 \\ 0.01 & 1.01 & 0.01 \\ 0 & 0.01 & 0.01 \end{bmatrix} x_t + \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} u_t + w_t$$

The LQR problem is approached with settings $Q = I$ and $R = 1,000I$. Figure 3 from [3] is reproduced here showing the results of applying model-free and model-based methods to learn a solution to this LQR problem.

6 Theoretical Properties: Policy Gradient for LQR

Now, we turn to [4] for an in-depth theoretical analysis of the performance of policy gradient methods on an infinite-horizon LQR problem. They consider the following problem:

$$\text{minimize } \mathbb{E}\left[\sum_{t=0}^{\infty}(x_t^T Q x_t + u_t^T R u_t)\right] \text{ such that } x_{t+1} = Ax_t + Bu_t, x_0 \sim \mathcal{D} \tag{6}$$

and make a few important contributions. First, they show that gradient descent on linear policies $u_t = Kx_t$ globally converges to optimal policies, and does so efficiently. Secondly, they showed that simulated trajectories can be used in a stochastic policy gradient method with provable convergence to a globally optimal policy with polynomial computational and sample complexity. And finally, they show that natural policy gradient methods (and their stochastic counterparts) enjoy significantly improved convergence rates.

6.1 Setup

In this work, direct policy gradient methods are characterized, where the policy is linearly parameterized by a matrix K, such that the controls $u_t = -Kx_t$ are generated. The cost of this policy K is denoted as follows:

$$C(K) := \mathbb{E}_{x_0 \sim \mathcal{D}} \left[\sum_{t=0}^{\infty} (x_t^T Q x_t + u_t^T R u_t) \right] \tag{7}$$

where $\{x_t, u_t\}$ is the trajectory and control sequence resulting from following K, starting from x_0. Gradient descent updates the policy, following the update rule:

$$K \leftarrow K - \eta \nabla C(K)$$

We can more clearly write the gradient in its functional form. With P_K as the solution to

$$P_K = Q + K^T R K + (A - BK)^T P_K (A - BK)$$

we can write

$$C(K) = \mathbb{E}_{x_0 \sim \mathcal{D}} \left[x_0^T P_K x_0 \right]$$

.

Also, define Σ_K as the state correlation matrix:

$$\Sigma_K = \mathbb{E}_{x_0 \sim \mathcal{D}} \sum_{t=0}^{\infty} x_t x_t^T$$

.

Now, we can write the gradient in its functional form:

Lemma 1. *(Policy Gradient Expression):*

$$\nabla C(K) = 2 \left((R + B^T P_K B) K - B^T P_K A \right) \Sigma_K$$

for simplicity later, define:

$$E_K := \left((R + B^T P_K B) K - B^T P_K A \right)$$

so we have $\nabla C(K) = 2 E_K \Sigma_K$

Proof: The full proof can be found in [4]

6.2 Optimization Landscape

Lemma 2. *(Gradient Domination) Let K^* be an optimal policy. Suppose K has finite cost and $\sigma_{min}(\Sigma + K) > 0$. It holds that:*

$$C(K) - C(K^*) \leq \frac{||\Sigma_{K^*}||}{\sigma_{min}(\Sigma + K)^2 \sigma_{min}(R)} ||\nabla C(K)||_F^2$$

Proof: comes from analyzing the advantage of the optimal policy Σ^* over Σ in each step.

Corollary 1. *(Stationary point characterization) If $\nabla C(K) = 0$, then either K is an optimal policy or Σ_K is rank deficient.*

Note that $\Sigma_K \geq \Sigma_0 := \mathbb{E}_{x_0 \sim \mathcal{D}} x_0 x_0^T$. So if $\mathbb{E}_{x_0 \sim \mathcal{D}} x_0 x_0^T$ is full rank, then Σ_K is full rank, and we know that all stationary points are global optima (i.e. K is optimal).

Using the fact that $\Sigma_K \geq \Sigma_0$, we can prove the convergence of $C(K)$ by showing that it is gradient dominated, meaning in this case that $\exists \lambda$ s.t. $C(K) - C(K^*) \leq \lambda ||\nabla C(K)||^2$. Proving convergence this way dates back to the 60s with [5], and requires $C(K)$ to be gradient dominated **and smooth.** If this were the case, we would be able to immediately imply global convergence at a linear rate. However, our cost function is not smooth. Near the boundary between stable and unstable policies, our cost may go from convergent to ∞. In other words, we may have $C(K) - C(K + \epsilon) = \infty$. To solve this problem, we observe that as long as we aren't too close to the boundary, the cost function satisfies an "almost smoothness" condition:

Lemma 3. *("Almost" smoothness) $C(K)$ satisfies:*

$$C(K') - C(K) = -2Tr(\Sigma_{K'}(K-K')^T E_K) + Tr(\Sigma_{K'}(K-K')^T(R+B^T P_K B)(K-K'))$$

where E_K here is as defined in Lemma 1.

To see why this helps, consider when K' is sufficiently close to K such that $\Sigma_{K'} \approx \Sigma_K + O(||K-K'||_F)$. Then, considering the order with respect to $(K-K')$, Lemma 3 simplifies as follows:

$$\begin{aligned}
&-2Tr(\Sigma_{K'}(K-K')^T E_K) + O(||K-K'||_F^2) \\
&= -Tr(K-K')^T 2E_K \Sigma_K + O(||K-K'||_F^2) \\
&= -Tr(K-K')^T \nabla C(K) + O(||K-K'||_F^2)
\end{aligned}$$

This tells us that as long as small perturbations of the controller lead to small perturbations of the steady state covariance, the cost function admits a Taylor series approximation

$$C(K') - C(K) \approx \langle K-K', \nabla C(K)\rangle + O(||K-K'||_F^2)$$

Quantifying this well-behaved (i.e. smooth) first order Taylor series approximation is used to show that gradient descent converges to critical points. This, combined with a gradient-dominated cost function, is sufficient to conclude convergence to a globally optimal solution despite a non-convex landscape.

6.3 Main Results

First, results on exact gradient methods are provided. We'll work through these, establishing global convergence, and then consider model-free methods.

6.3.1 Model-Based Methods

We'll use three exact update rules. The gradient descent update is:

$$K_{n+1} = K_n - \eta \nabla C(K_n),$$

the natural gradient descent update is:

$$K_{n+1} = K_n - \eta \nabla C(K_n)\Sigma_{K_n}^{-1},$$

and the Gauss-Newton method update is:

$$K_{n+1} = K_n - \eta (R+B^T P_{K_n} B)^{-1} \nabla C(K_n)\Sigma_{K_n}^{-1}.$$

Theorem 1. *(Global Convergence of Gradient Methods)*
Suppose $C(K_0)$ is finite and $\mu = \sigma_{min}(\mathbb{E}_{x_0\sim\mathcal{D}}[x_0 x_0^T]) > 0$
With the following step sizes and lower bounds for N, we have that $C(K_N) - C(K_*) \leq \epsilon$

Gauss-Newton:

$$\eta = 1, \quad N \geq \frac{||\Sigma_{K_*}||_2}{\mu} log \frac{C(K_0) - C(K_*)}{\epsilon}$$

Natural Gradient Descent:

$$\eta = \frac{1}{||R|| + \frac{||B||^2 C(K_0)}{\mu}}, \quad N \geq \frac{||\Sigma_{K_*}||_2}{\mu} log \frac{C(K_0) - C(K_*)}{\epsilon} \left(\frac{||R||}{\sigma_{min}(R)} + \frac{||B||^2 C(K_0)}{\mu \sigma_{min}(R)} \right)$$

Gradient Descent: *for an appropriate (constant) setting of the stepsize* η*,*

$$\eta = poly\left(\frac{\mu\sigma_{min}(Q)}{C(K_0)}, \frac{1}{||A||_2}, \frac{1}{||B||_2}, \frac{1}{||R||_2}, \sigma_{min}(R)\right)$$

$$N \geq \frac{||\Sigma_{K_*}||_2}{\mu} log\frac{C(K_0) - C(K_*)}{\epsilon} poly\left(\frac{C(K_0)}{\mu\sigma_{min}(R)}, ||A||_2, ||B||_2, ||R||_2, \frac{1}{\sigma_{min}(R)}\right)$$

*Proof:*Gauss-Newton reproduced here, others see supplemental material of [4].

Lemma 4. *First, let*

$$K' = K - \eta(R + B^T P_K B)^{-1}\nabla C(K)\Sigma_K^{-1}$$

if $\eta \leq 1$*,*

$$C(K') - C(K^*) \leq \left(1 - \frac{\eta\mu}{||\Sigma_{K^*}||}\right)(C(K) - C(K^*))$$

Now, the proof: First, note that from Lemma 1, we can rewrite $K' = K - \eta(R + B^T P_K B)^{-1}E_K$*. Now, using Lemma 3 and* $\eta \leq 1$*:*

$$\begin{aligned}
C(K') - C(K) &= -2\eta Tr(\Sigma_{K'} E_K^T (R + B^T P_K B)^{-1} E_K) + \eta^2 Tr(\Sigma_{K'} E_K^T (R + B^T P_K B)^{-1} E_K) \\
&\leq -\eta Tr(\Sigma_{K'} E_K^T (R + B^T P_K B)^{-1} E_K) \\
&\leq -\eta\sigma_{min}(\Sigma_{K'}) Tr(E_K^T (R + B^T P_K B)^{-1} E_K) \\
&\leq -\eta\mu Tr(E_K^T (R + B^T P_K B)^{-1} E_K) \\
&\leq -\frac{\eta\mu}{||\Sigma_{K^*}||}(C(K) - C(K^*)),
\end{aligned}$$

where the last step uses Lemma 2. Proof of convergence rate of Gauss Newton algorithm is immediate: $\eta = 1$ *leads to a contraction of* $1 - \frac{\eta\mu}{||\Sigma_{K^*}||}$ *at every step.*

6.3.2 Model Free Methods

In this setting, the controller has only simulation access to the model - it does not know A, B, Q, or R, and can only query the simulation. So to convert our model-based formulations, we use a gradient estimation algorithm.

The step update formulae are similar to model-based, only replacing exact gradient values with algorithm calls. The policy gradient step update is:

$$K_{n+1} = K_n - \eta\widehat{\nabla C(K_n)},$$

and the natural policy gradient step update is:

$$K_{n+1} = K_n - \eta\widehat{\nabla C(K_n)}\Sigma_{K_n}^{-1}.$$

The gradient estimates $widehat\nabla C(K_n)$ are calculated using Algorithm 2.

Algorithm 2: Model-Free Policy Gradient (and Natural Policy Gradient) Estimation

Input: K, number of trajectories m, roll out length l, smoothing parameter r, dimension d
for i = *1,...m* **do**
sample a policy $\hat{K}_i = K + U_i$, where U_i is drawn uniformly at random over matrices whose Frobenius norm is r;
Simulate $\hat{K}_i$ for l steps starting from $x_0 \sim \mathcal{D}$. Let $\hat{C}_i$ and $\hat{\Sigma}_i$ be the empirical estimates:

$$\hat{C}_i = \sum_{t=1}^{l} c_t, \quad \hat{\Sigma}_i = \sum_{t=1}^{l} x_t x_t^T$$

where c_t and x_t are the costs and states on this trajectory ;
end
Return the (biased) estimates:

$$\widehat{\nabla C(K)} = \frac{1}{m}\sum_{i=1}^{m} \frac{d}{r^2}\hat{C}_i U_i, \quad \widehat{\Sigma_K} = \frac{1}{m}\sum_{i=1}^{m}\hat{\Sigma}_i$$

The choice of this algorithm over REINFORCE (Algorithm 1) is primarily for technical reasons. REINFORCE could potentially have lower variance.

Anyhow, Algorithm 2 is applied at every iteration to provide the estimates necessary for policy gradient and natural policy gradient methods. This brings us to the main result of the paper for model-free methods:

Theorem 2. *(Global Convergence in the Model Free Setting)*
Assume that $C(K_0)$ is finite, $\mu > 0$ (defined in Theorem 1), and that $x_0 \sim \mathcal{D}$ has norm bounded by L almost surely. Then, given the step sizes and lower bounds of N outlined below, we can say that with probability greater than $1 - e^{-d}$, $C(K_N) - C(K_*) \leq \epsilon$

Natural policy gradient case:

$$\eta = \frac{1}{||R|| + \frac{||B||^2 C(K_0)}{\mu}}, \quad N \geq \frac{||\Sigma_{K_*}||_2}{\mu} log \frac{2(C(K_0) - C(K_*))}{\epsilon}\left(\frac{||R||}{\sigma_{min}(R)} + \frac{||B||^2 C(K_0)}{\mu \sigma_{min}(R)}\right)$$

Gradient descent case: *for an appropriate (constant) setting of the stepsize η,*

$$\eta = poly\left(\frac{\mu\sigma_{min}(Q)}{C(K_0)}, \frac{1}{||A||_2}, \frac{1}{||B||_2}, \frac{1}{||R||_2}, \sigma_{min}(R)\right)$$

$$N \geq \frac{||\Sigma_{K_*}||_2}{\mu} log \frac{C(K_0) - C(K_*)}{\epsilon} poly\left(\frac{C(K_0)}{\mu\sigma_{min}(R)}, ||A||_2, ||B||_2, ||R||_2, \frac{1}{\sigma_{min}(R)}\right)$$

Proof sketch:

1. Show that rollout length l is long enough so that finite cost C and covariance Σ approximate their infinite counterparts
2. Show that with enough samples, Algorithm 1 can estimate both the gradient $\nabla C(K)$ and the covariance matrix Σ_K with a desired accuracy
3. Prove that gradient and natural gradient descent still converge at (about) the same rate if the gradients have slight perturbations added to them

7 Conclusion

We have been introduced to several approaches to model-free reinforcement learning: the REINFORCE algorithm, policy gradient, and pure random search. We looked at how these approaches performed on two LQR problems: the discrete-time double integrator, and the more complex idealized data center cooling problem. We saw that on both problems, at least one model-free method was able to generate stabilizing controllers even in the presence of unstable dynamics, given enough samples (on the order of a few thousand). In the double integrator, the model-free methods converged to the optimal solution. In the more complex cooling problem, the model-free methods performed roughly one order of magnitude worse than the nominal and robust model-based control methods. Random search was able to stabilize almost as quickly as the model-based method, but policy gradient was inconsistent and did not stabilize reliably. It should be noted that we do expect model-free methods to underperform compared to model-based methods tailor-built for a problem. It is in scenarios where a model is not available that model-free methods excel.

We then turned to a theoretical analysis of the performance of model-free methods - specifically, direct policy gradient - on LQR. We first developed a proof of convergence of model-based methods, and then used this framework to reason about model-free methods that only have simulation query access to the system. Under assumptions of "almost-smoothness" we were able to prove that natural policy gradient, and gradient descent model-free methods are convergent to a globally optimal policy, and characterized their sample complexity.

References

[1] Ronald J. Williams. Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Machine Learning*, 8(3):229–256, 1992.

[2] Horia Mania, Aurelia Guy, and Benjamin Recht. Simple random search provides a competitive approach to reinforcement learning. pages 1–22, 2018.

[3] Benjamin Recht. A Tour of Reinforcement Learning: The View from Continuous Control. *Annual Review of Control, Robotics, and Autonomous Systems*, 2(1):253–279, 2019.

[4] Maryam Fazel, Rong Ge, Sham M. Kakade, and Mehran Mesbahi. Global Convergence of Policy Gradient Methods for the Linear Quadratic Regulator. *35th International Conference on Machine Learning, ICML 2018*, 4:2385–2413, 2018.

[5] B. T. Polyak. Gradient methods for the minimisation of functionals. *USSR Computational Mathematics and Mathematical Physics*, 3(4):864–878, 1963.