CHAPTER 17

Model-free Policy Search

17.1 INTRODUCTION

In chapter 12, we talked about policy search as a nonlinear optimization problem, and discussed efficient ways to calculate the gradient of the long-term cost:

$$J^\alpha(\mathbf{x}_0) = \int_0^T g(\mathbf{x}, \pi_\alpha(\mathbf{x}, t))dt, \text{ s.t. } \mathbf{x}(0) = \mathbf{x}_0.$$

Assuming $J^\alpha(\mathbf{x}_0)$ is smooth over $\mathbf{w}$, we derived updates of the form:

$$\Delta\alpha = -\eta \left[\frac{\partial J^\alpha}{\partial \alpha}\right]^T_{\mathbf{x}_0,\alpha}.$$

But it may not always be possible, or practical, to compute the gradients exactly.

A standard technique for estimating the gradients, in lieu of analytical gradient information, is the method of finite differences[67]. The finite differences approach to estimating the gradient involves making the same small perturbation, ϵ to the input parameters in every dimension independently, and using:

$$\frac{\partial J^\alpha}{\partial \alpha_i} \approx \frac{J^{\alpha+\epsilon_i}(\mathbf{x}_0) - J^\alpha(\mathbf{x}_0)}{\epsilon},$$

where $\epsilon_\mathbf{i}$ is the column vector with ϵ in the ith row, and zeros everywhere else. Finite difference methods can be computationally very expensive, requiring $n+1$ evaluations of the function for every gradient step, where n is the length of the input vector.

In this chapter we will develop *stochastic* gradient descent algorithms which can, in expectation, descend a policy gradient with considerably less evaluations than those required for finite differences. In addition, we will develop methods that are robust to stochasticity in the function evaluation, which is inevitable if one is trying to descend a policy gradient on a real robot. This allows for the exciting possibility of optimizing a control policy for a system without requiring any model of the plant.

17.2 STOCHASTIC GRADIENT DESCENT

Requirements and guarantees. For a formal treatment, see Bertsekas.

17.3 THE WEIGHT PERTUBATION ALGORITHM

Instead of sampling each dimension independently, consider making a single small random change, β, to the parameter vector, α. Assuming that the function is locally smooth, we can approximate the result of this experiment using a Taylor expansion:

$$J^{\alpha+\beta}(\mathbf{x}_0) \approx J^\alpha(\mathbf{x}_0) + \frac{\partial J^\alpha(\mathbf{x}_0)}{\partial \alpha}\beta.$$

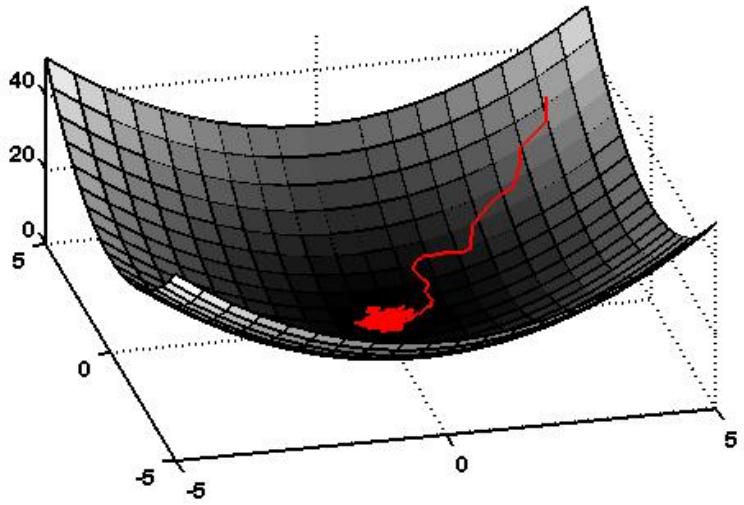

FIGURE 17.1 Stochastic gradient descent of a quadratic cost function. See problem 1 for details.

Now consider the update of the form:

$$\Delta\alpha = -\eta[J^{\alpha+\beta}(\mathbf{x}_0) - J^{\alpha}(\mathbf{x}_0)]\beta.$$

Since $J^{\alpha+\beta} - J^{\alpha}$ is (approximately) the dot product between $\frac{\partial J^{\alpha}}{\partial \alpha}$ and β, the sign of this term ensures that $\Delta\alpha$ is always within 90 degrees of true gradient descent.

Furthermore, on average, the update is in the direction of the true gradient:

$$\Delta\alpha \approx -\eta\beta\beta^T \frac{\partial J}{\partial \alpha}^T$$
$$E\left[\Delta\alpha\right] \approx -\eta E\left[\beta\beta^T\right] \frac{\partial J}{\partial \alpha}^T$$

If we select each element of β independently from a distribution with zero mean and variance σ_β^2, or $E\left[\beta_i\right] = 0, E\left[\beta_i\beta_j\right] = \sigma_\beta^2\delta_{ij}$, then we have

$$E\left[\Delta\alpha\right] \approx -\eta\sigma_\beta^2 \frac{\partial J}{\partial \alpha}^T.$$

Note that the distribution $f_{\alpha_i}(\alpha_i)$ need not be Gaussian, but it is the variance of the distribution which determines the scaling on the gradient. Bringing the σ_β^2 into the update, we have the weight perturbation algorithm:

$$\boxed{\Delta\alpha = -\tfrac{\eta}{\sigma_\beta^2}[J^{\alpha+\beta}(\mathbf{x}_0) - J^{\alpha}(\mathbf{x}_0)]\beta} \tag{17.1}$$

17.3.1 Performance of Weight Perturbation

The simplicity of the weight perturbation update makes it tempting to apply it to problems of arbitrary complexity. But a major concern for the algorithm is its performance - although we have shown that the update is in the direction of the true gradient *on average*, it may still require a prohibitive number of computations to obtain a local minima.

In this section, we will investigate the performance of the weight perturbation algorithm by investigating its *signal-to-noise* ratio (SNR). The SNR is the ratio of the power in the signal (here desired update in the direction of the true gradient) and the power in the noise (the remaining component of the update, so that $\Delta\alpha = -\eta\frac{\partial J}{\partial \alpha}^T + \text{noise}$)[1]

$$\text{SNR} = \frac{\left|-\eta\frac{\partial J}{\partial \alpha}^T\right|^2}{E\left[\left|\Delta\alpha + \eta\frac{\partial J}{\partial \alpha}^T\right|^2\right]}.$$

In the special case of the unbiased update, the equation reduces to:

$$\text{SNR} = \frac{E\left[\Delta\alpha\right]^T E\left[\Delta\alpha\right]}{E\left[(\Delta\alpha)^T(\Delta\alpha)\right] - E\left[\Delta\alpha\right]^T E\left[\Delta\alpha\right]}.$$

For the weight perturbation update we have:

$$E\left[\Delta\alpha\right]^T E\left[\Delta\alpha\right] = \eta^2 \frac{\partial J}{\partial \alpha}\frac{\partial J}{\partial \alpha}^T = \eta^2 \sum_{i=1}^{N}\left(\frac{\partial J}{\partial \alpha_i}\right)^2,$$

[1] SNR could alternatively be defined as the ratio of the power of the component of the update in the direction of the signal and the component orthogonal to the signal (does not penalize the magnitudes of the updates):

$$\frac{E[a^2]}{E\left[|\Delta\alpha - a\mathbf{v}|^2\right]}, \text{ where } \mathbf{v} = \frac{\frac{\partial J}{\partial \alpha}^T}{\left|\frac{\partial J}{\partial \alpha}^T\right|}, a = \Delta\alpha \cdot \mathbf{v}.$$

This form is probably a better definition, but is more difficult to work with.

$$\begin{aligned}
E\left[(\Delta\alpha)^T(\Delta\alpha)\right] =& \frac{\eta^2}{\sigma_\beta^4} E\left[\left[J^{\alpha+\beta}-J^\alpha\right]^2 \beta^T\beta\right] \\
\approx& \frac{\eta^2}{\sigma_\beta^4} E\left[\left[\frac{\partial J}{\partial \alpha}\beta\right]^2 \beta^T\beta\right] \\
=& \frac{\eta^2}{\sigma_\beta^4} E\left[\left(\sum_i \frac{\partial J}{\partial \alpha_i}\beta_i\right)\left(\sum_j \frac{\partial J}{\partial \alpha_j}\beta_j\right)\left(\sum_k \beta_k^2\right)\right] \\
=& \frac{\eta^2}{\sigma_\beta^4} E\left[\sum_i \frac{\partial J}{\partial \alpha_i}\beta_i \sum_j \frac{\partial J}{\partial \alpha_j}\beta_j \sum_k \beta_k^2\right] \\
=& \frac{\eta^2}{\sigma_\beta^4} \sum_{i,j,k} \frac{\partial J}{\partial \alpha_i}\frac{\partial J}{\partial \alpha_j} E\left[\beta_i\beta_j\beta_k^2\right]
\end{aligned}$$

$$E\left[\beta_i\beta_j\beta_k^2\right] = \begin{cases} 0 & i \neq j \\ \sigma_\beta^4 & i = j \neq k \\ \mu_4(\beta) & i = j = k \end{cases}$$

$$= \eta^2(N-1)\sum_i \left(\frac{\partial J}{\partial \alpha_i}\right)^2 + \frac{\eta^2\mu_4(z)}{\sigma_\beta^4}\sum_i \left(\frac{\partial J}{\partial \alpha_i}\right)^2$$

where $\mu_n(z)$ is the nth central moment of z:

$$\mu_n(z) = E\left[(z - E[z])^n\right].$$

Putting it all together, we have:

$$\text{SNR} = \frac{1}{N - 2 + \frac{\mu_4(\beta_i)}{\sigma_\beta^4}}.$$

EXAMPLE 17.1 Signal-to-noise ratio for additive Gaussian noise

For β_i drawn from a Gaussian distribution, we have $\mu_1 = 0, \mu_2 = \sigma_\beta^2, \mu_3 = 0, \mu_4 = 3\sigma_\beta^4$, simplifying the above expression to:

$$\text{SNR} = \frac{1}{N+1}.$$

EXAMPLE 17.2 Signal-to-noise ratio for additive uniform noise

For β_i drawn from a uniform distribution over [-a,a], we have $\mu_1 = 0, \mu_2 = \frac{a^2}{3} = \sigma_\beta^2, \mu_3 = 0, \mu_4 = \frac{a^4}{5} = \frac{9}{5}\sigma_\beta^4$, simplifying the above to:

$$\text{SNR} = \frac{1}{N - \frac{1}{5}}.$$

Performance calculations using the SNR can be used to design the parameters of the algorithm in practice. For example, based on these results it is apparent that noise added through a uniform distribution yields better gradient estimates than the Gaussian noise case for very small N, but that these differences are neglibile for large N.

Similar calculations can yield insight into picking the size of the additive noise (scaled by σ_β). The calculations in this section seem to imply that larger σ_β can only reduce the variance, overcoming errors or noise in the baseline estimator, $\tilde{b}$; this is a shortcoming of our first-order Taylor expansion. If the cost function is not linear in the parameters, then an examination of higher-order terms reveals that large σ_β can increase the SNR. The derivation with a second-order Taylor expansion is left for an exercise (Problem 3).

17.3.2 Weight Perturbation with an Estimated Baseline

The weight perturbation update above requires us to evaluate the function J twice for every update of the parameters. This is low compared to the method of finite differences, but let us see if we can do better. What if, instead of evaluating the function twice per update [$J^{\alpha+\beta}$ and J^α], we replace the evaluation of J^α with an estimator, $b = \hat{J}(\alpha)$, obtained from previous trials? In other words, consider an update of the form:

$$\boxed{\Delta\alpha = -\frac{\eta}{\sigma_\beta^2}[J^{\alpha+\beta} - b]\beta} \tag{17.2}$$

The estimator can take any of a number of forms, but perhaps the simplest is based on the update (after the nth trial):

$$b[n+1] = \gamma J[n] + (1-\gamma)b[n], \quad b[0] = 0, 0 \leq \gamma \leq 1,$$

where γ parameterizes the moving average. Let's compute the expected value of the update, using the same Taylor expansion that we used above:

$$\begin{aligned} E[\Delta\alpha] &= -\frac{\eta}{\sigma_\beta^2} E\left[[J^\alpha + \frac{\partial J}{\partial \alpha}\beta - b]\beta\right] \\ &= -\frac{\eta}{\sigma_\beta^2} E\left[J^\alpha - b\right] E\left[\beta\right] + E\left[\beta\beta^T\right] \frac{\partial J}{\partial \alpha}^T \\ &= -\eta \frac{\partial J}{\partial \alpha}^T \end{aligned}$$

In other words, the baseline does not effect our basic result - that the expected update is in the direction of the gradient. Note that this computation works for any baseline estimator which is uncorrelated with β, as it should be if the estimator is a function of the performance in previous trials.

Although using an estimated baseline doesn't effect the average update, it can have a dramatic effect on the performance of the algorithm. Although it is left as an exercise for the reader (see Problem 2), the results are...

As we will see in a later section, if the evalution of J is stochastic, then the update with a baseline estimator can actually outperform an update using straight function evaluations.

17.4 THE REINFORCE ALGORITHM

The weight perturbation algorithm used only additive noise in the parameters. It also assumed that this noise was small and that J was smoothly differientable. More generally, we can minimize the expected value of a scalar deterministic function $y(\mathbf{z})$, where $\mathbf{z}$ is a vector random variable which depends on α through the conditional probability density function $f_{\mathbf{z}|\alpha}(\mathbf{z}|\alpha)$. Note that $\mathbf{z} = \alpha + \beta$, is a special case of this general form. Consider the gradient of the expected value of y:

$$\begin{aligned}\frac{\partial}{\partial\alpha}E\left[y\right] &= \int_{-\infty}^{+\infty} y(\mathbf{z})\frac{\partial}{\partial\alpha}f_{\mathbf{z}|\alpha}(\mathbf{z}|\alpha)dz \\ &= \int_{-\infty}^{+\infty} y(\mathbf{z})f_{\mathbf{z}|\alpha}(\mathbf{z}|\alpha)\frac{\partial}{\partial\alpha}\ln f_{\mathbf{z}|\alpha}(\mathbf{z}|\alpha)dz \\ &= E\left[y(\mathbf{z})\left[\frac{\partial}{\partial\alpha}\ln f_{\mathbf{z}|\alpha}(\mathbf{z}|\alpha)\right]^T\right],\end{aligned}$$

where the second line used the identity $\frac{\partial}{\partial z}\ln(z) = \frac{1}{z}$. Therefore, the update for the REINFORCE algorithm is:

$$\boxed{\Delta\alpha = -\eta\left[y(\mathbf{x}) - b\right]\left[\frac{\partial}{\partial\alpha}\ln f_{\mathbf{x}|\alpha}(\mathbf{x}|\alpha)\right]^T} \tag{17.3}$$

which will have the property that

$$E\left[\Delta\alpha\right] = -\eta\frac{\partial}{\partial\alpha}E\left[y\right].$$

This derivation generalizes our weight perturbation algorithm, removing the assumption on small β and allowing more general forms for $f_{\mathbf{x}|\alpha}(\mathbf{x}|\alpha)$. The caveat is that the update is only well-defined for probability densities which are smoothly differentiable in α.

EXAMPLE 17.3 REINFORCE with additive Gaussian noise

When $\mathbf{x} = \alpha + \beta$, $\beta \in N(0, \sigma^2)$, we have

$$f_{\mathbf{x}|\alpha}(\mathbf{x}|\alpha) = \frac{1}{(2\pi\sigma^2)^N}e^{\frac{-(\mathbf{x}-\alpha)^T(\mathbf{x}-\alpha)}{2\sigma^2}}$$

$$\ln f_{\mathbf{x}|\alpha}(\mathbf{x}|\alpha) = \frac{-(\mathbf{x}-\alpha)^T(\mathbf{x}-\alpha)}{2\sigma^2} + \text{terms that don't depend on } \alpha$$

$$\frac{\partial}{\partial\alpha}\ln f_{\mathbf{x}|\alpha}(\mathbf{x}|\alpha) = \frac{1}{\sigma^2}(\alpha - \mathbf{x})^T = \frac{1}{\sigma^2}\beta^T.$$

Therefore, the update

$$\Delta\alpha = -\frac{\eta}{\sigma^2}y(\alpha+\beta)\beta$$

will perform gradient descent (in expectation). Since

$$E\left[y(\alpha)\beta\right] = y(\alpha)E\left[\beta\right] = \mathbf{0},$$

the update from Equation 17.1 will also perform stochastic gradient descent on y.

17.4.1 Optimizing a stochastic function

To generalize our weight perturbation results, now consider the case where y is not a deterministic function of $\mathbf{x}$, but rather is described by the joint density function:

$$f_{y,\mathbf{x}|\alpha}(y,\mathbf{x}|\alpha) = f_{y|\mathbf{x}}(y|\mathbf{x})f_{\mathbf{x}|\alpha}(\mathbf{x}|\alpha).$$

Repeating the recipe above, we have:

$$\begin{aligned}\frac{\partial}{\partial\alpha}E\left[y\right] =& \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} y(\mathbf{x})f_{y|\mathbf{x}}(y|\mathbf{x})\frac{\partial}{\partial\alpha}f_{\mathbf{x}|\alpha}(\mathbf{x},\alpha)dxdy \\ =& \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} y(\mathbf{x})f_{y|\mathbf{x}}(y|\mathbf{x})f_{\mathbf{x}|\alpha}(\mathbf{x}|\alpha)\frac{\partial}{\partial\alpha}\ln f_{\mathbf{x}|\alpha}(\mathbf{x}|\alpha)dxdy \\ =& E\left[y(\mathbf{x})\left[\frac{\partial}{\partial\alpha}\ln f_{\mathbf{x}|\alpha}(\mathbf{x}|\alpha)\right]^T\right],\end{aligned}$$

Therefore, the weight perturbation algorithm derived still achieves stochastic gradient descent despite stochasticity in y.

17.4.2 Adding noise to the outputs

17.5 EPISODIC REINFORCE

Evaluating our control system many times per trial. Can we do better by performing multiple perturbation "tests" during a single trial? Stochastic processes are harder (but not impossible) to define in continuous time, and our control systems are (mostly) implemented on digital machines, so let's discretize time.

$$E_\alpha = \sum_{n=0}^{N} g(\mathbf{x}[n],\mathbf{u}[n]),$$

subject to

$$\mathbf{u}[n] = \pi_\alpha(\mathbf{x}[n],n), \mathbf{x}[n+1] = f(\mathbf{x}[n],\mathbf{u}[n],n), \mathbf{x}[0] = \mathbf{x}_0.$$

To match our previous derivation, let us rewrite this as

$$E_\alpha(\bar{\mathbf{x}},\bar{\mathbf{u}}) = \sum g(\mathbf{x}[n],\mathbf{u}[n]), \text{ and } P_\alpha(\bar{\mathbf{x}},\bar{\mathbf{u}})$$

where $\bar{\mathbf{x}} = \{x_0, x_1, ...x_N\}$ and $\bar{\mathbf{u}} = \{u_0, u_1, ...u_N\}$. The first task is to compute $\frac{\partial}{\partial\alpha}\log P_\alpha(\bar{\mathbf{x}},\bar{\mathbf{u}})$.

End up with a rule of the form

$$\Delta w = -\eta\sum_{n=0}^{N} g(x,u)\mathbf{e}(k),$$

where

$$\mathbf{e}(0) = 0, \mathbf{e}(k+1) = \left[\frac{\partial}{\partial\alpha}\log P_\alpha(x(k))\right]^T + \mathbf{e}(k).$$

17.6 **INFINITE-HORIZON REINFORCE**

17.7 **LTI REINFORCE**

17.8 **BETTER BASELINES WITH IMPORTANCE SAMPLING**

PROBLEMS

17.1. (MATLAB) *Weight perturbation on a quadratic cost.*

In this problem we will optimize a cost quadratic cost function using weight perturbation.

$$y(\alpha) = \frac{1}{2}\alpha^T \mathbf{A}\alpha, \text{ with } \mathbf{A} = \mathbf{A}^T$$

(a) Simulate the weight perturbation algorithm for $\mathbf{A} = \mathbf{I}_{2\times 2}$. Plot the values of α at each iteration of the algorithm on top of the cost function, y. Your results should resemble figure 17.1, which was made with these parameters. Try using additive Gaussian noise, and additive uniform noise. How does the performance of convergence depend on η? on σ_β^2?

(b) (*Signal-to-noise ratio.*) Estimate the signal-to-noise ratio of your weight pertubation update by holding α fixed and computing $\Delta\alpha$ for as many trials as are required for your SNR statistics to converge. Using $\mathbf{A} = \mathbf{I}_{N\times N}$, make a plot of the SNR for Gaussian and uniform additive noise as a function of N. Compare these plots with the theoretical predictions.

17.2. *Weight perturbation performance with a baseline estimator.*

17.3. *Weight perturbation - optimal noise calculations.*

Use a second-order Taylor expansion...

$$y(\alpha + \beta) = y(\alpha) + \frac{\partial J}{\partial \alpha}\beta + \frac{1}{2}\beta^T \mathbf{h}\beta,$$

where $\mathbf{h}$ is the Hermitian of the function evaluated at α:

$$\mathbf{h} = \frac{\partial}{\partial \alpha}\left[\frac{\partial J}{\partial \alpha}^T\right] = \begin{bmatrix} \frac{\partial^2 y}{\partial \alpha_1 \partial \alpha_1} & \cdots & \frac{\partial^2 y}{\partial \alpha_1 \partial \alpha_N} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 y}{\partial \alpha_N \partial \alpha_1} & \cdots & \frac{\partial^2 y}{\partial \alpha_N \partial \alpha_N} \end{bmatrix}.$$

You may assume that the distribution on β_i is symmetric about the mean, which zeros all odd central-moments starting at $\mu_3(\beta_i) = 0$.

6.832 Underactuated Robotics
Spring 2009