13
Response Surface Methods

In this lecture we introduce Response Surface Methods (RSM) for policy optimization in reinforcement learning (RL). The problem setting in this lecture makes two important assumptions:

Parametric policy the policy $\pi : \mathbb{S} \to \mathbb{A}$ depends on a parameter vector $\theta \in \mathbb{R}^d$.

Episodic learning In this setting learning proceeds by interleaving two stages in a loop:

1. Batch simulation (or execution) of one or more "episodes"[1] under the current policy.
2. Adjustment of the policy (in this case, of the current policy's parameters) based on some aggregate performance metric from the simulations (e.g. total sum of rewards).

Episodic learning contrasts with methods that continuously learn throughout the simulation.

[1] Previously described as "rollouts" in this course.

See table 13.1 for an example of episodes, policies and rewards in three problem domains.

	Domain	
	Tetris	Helicopter
Episode	Game	Simulate for 1 minute
Policy	Board $\to$ Move	$(x, y, z, \dot{x}, \dot{y}, \dot{z}) \to$ (cycle, throttle, collective)
Reward	# of lines cleared	Remain close to desired trajectory

Table 13.1: Examples of episodes, policies and rewards

13.1 Optimization with Response Surface Methods

RSM is a general purpose, "black box" optimization method for any function $f : \mathbb{X} \to \mathbb{R}$. The outline of RSM is as follows:

1. Pick an initial point $\mathbf{x}_1$.
2. For $t = 1, \ldots, T$
   (a) Obtain response $f(\mathbf{x}_t)$.
   (b) Fit a "response surface" $\hat{f}$ to all data points $\{(\mathbf{x}_1, f(\mathbf{x}_1)), \ldots, (\mathbf{x}_t, f(\mathbf{x}_t))\}$.
   (c) Use the response surface to choose a new $\mathbf{x}_{t+1}$.

Note that t here means iterations rather than steps in episodes.

Step (b) involves fitting a function $\hat{f}$ and step (c) optimizes over the response surface $\hat{f}$. These steps may be computationally expensive. Hence, RSM is most useful when evaluating f is itself very costly compared to solving the surrogate problems in steps (b) and (c), which is often the case in reinforcement learning for robotics. If evaluating f is cheap, then we may be better off using one of black box methods covered earlier, such as the cross entropy method.

13.2 Fitting the response surface: Gaussian Process Regression

In step (b) of the algorithm we fit a response surface $\hat{f}$, sometimes called the *surrogate function*, to all our data points $\{(\mathbf{x}_1, f(\mathbf{x}_1)), \ldots, (\mathbf{x}_t, f(\mathbf{x}_t))\}$. Intuitively, $\hat{f}$ is an approximation the real f, but much less expensive to evaluate, and it will guide our search for the location of the optima in f.

Gaussian processes (GPs) are a commonly used model in response surfaces methods that maintain a nonparametric prior over functions. The intuition behind the GP prior is to see functions as a point in a continuous, "infinite-dimensional" space. The joint distribution for any finite set of samples $\{f(\mathbf{x}_1), \ldots, f(\mathbf{x}_t)\}$ is Gaussian and defined by a mean function $\mu : \mathbb{X} \to \mathbb{R}$ and a covariance function $k : \mathbb{X} \times \mathbb{X} \to \mathbb{R}$:

$$\begin{aligned} \big(f(\mathbf{x}_0), \ldots, f(\mathbf{x}_t)\big)^\top &\sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \\ \boldsymbol{\mu} &= \big(\mu(\mathbf{x}_1), \ldots, \mu(\mathbf{x}_t)\big)^\top \\ \boldsymbol{\Sigma}_{ij} &= k(\mathbf{x}_i, \mathbf{x}_j) \quad \forall\, (i,j) \in [1, \ldots, t]^2 \end{aligned}$$

Without loss of generality, the mean function μ is assumed to be always 0. This is the same as preprocessing the data as $f'(\mathbf{x}) = f(\mathbf{x}) - \mu(\mathbf{x})$.

Conceptually, the kernel function is like an "infinite-dimensional" covariance matrix where $\Sigma_{ij} = k(\mathbf{x}_i, \mathbf{x}_j) = \mathrm{cov}\big(f(\mathbf{x}_i), f(\mathbf{x}_j)\big)$. Therefore it must meet the equivalent of the SPD condition:

1. $k(\mathbf{x}_i, \mathbf{x}_j) = k(\mathbf{x}_j, \mathbf{x}_i), \quad \forall\, \mathbf{x}_i, \mathbf{x}_j$
2. Any matrix K s.t. $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$, that is symmetric positive semi-definite.

While we do not go into further detail here, we should note that choosing a suitable kernel function is very important for the performance of the GP regression and by extension, the RSM. This is a model selection problem, which falls outside the scope of this lecture.

GP inference

Given a prior for $f \sim GP(\mu, k)$ and a set of samples $\big((\mathbf{x}_1, f(\mathbf{x}_1)), \ldots, (\mathbf{x}_t, f(\mathbf{x}_t))\big)$, we wish to compute a posterior over $f(\mathbf{x}^*)$ at any query $\mathbf{x}^*$. We have

$$\begin{bmatrix} f(\mathbf{x}^*) \\ f(\mathbf{x}_1) \\ \vdots \\ f(\mathbf{x}_t) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} k(\mathbf{x}^*, \mathbf{x}^*) & k(\mathbf{x}^*, \mathbf{x}_1) & \ldots & k(\mathbf{x}^*, \mathbf{x}_t) \\ k(\mathbf{x}_1, \mathbf{x}^*) & k(\mathbf{x}_1, \mathbf{x}_1) & \ldots & k(\mathbf{x}_1, \mathbf{x}_t) \\ \vdots & \vdots & \ddots & \vdots \\ k(\mathbf{x}_t, \mathbf{x}^*) & k(\mathbf{x}_t, \mathbf{x}_1) & \ldots & k(\mathbf{x}_t, \mathbf{x}_t) \end{bmatrix} \right)$$

For notational convenience, let us define

$$\begin{aligned} K_{**} &= k(\mathbf{x}^*, \mathbf{x}^*) \\ K_{x*} &= \big[k(\mathbf{x}^*, \mathbf{x}_1), \ldots, k(\mathbf{x}^*, \mathbf{x}_t)\big]^\top \\ K_{xx} &= \big[k(\mathbf{x}_i, \mathbf{x}_j)\big]_{i,j=1}^N \end{aligned}$$

Using the properties of the Normal distribution, it can be shown that the posterior of $f(\mathbf{x}^*)$ is also a Gaussian:

$$f(\mathbf{x}^*) | f(\mathbf{x}_1), \ldots, f(\mathbf{x}_t) \sim \mathcal{N}\big(\mu_t(\mathbf{x}^*), \sigma_t(\mathbf{x}^*)\big) \tag{13.2.1}$$

$$\mu_t(\mathbf{x}^*) \doteq K_{x*}^\top K_{xx}^{-1} \big[f(\mathbf{x}_1), f(\mathbf{x}_1), \ldots f(\mathbf{x}_t)\big]^\top \tag{13.2.2}$$

$$\sigma_t(\mathbf{x}^*) \doteq K_{**} - K_{x*}^\top K_{xx}^{-1} K_{x*} \tag{13.2.3}$$

This gives us the posterior expected mean of $f(\mathbf{x})$ at any point $\mathbf{x}$ in the input space $\mathbb{X}$. It also gives us a posterior variance for $f(\mathbf{x})$, which we can interpret as a measure of uncertainty about the value of f at the point. In section 13.3 we will see how to use both in optimization.

Visualization

A good way to visualize a GP is to draw samples and plot them as functions. For this one selects a set of t samples $\mathbf{x}_i$, $i = 1, \ldots, t$, constructs the corresponding covariance matrix with k, samples n-dimensional points from the Gaussian with this covariance matrix, and plots them as functions. Figure 13.2.1 shows draws from two different GP priors.

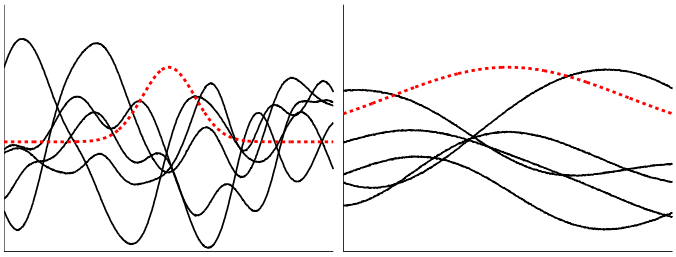

Figure 13.2.1: Draws from two different Gaussian Process Priors. Both use a squared exponential kernel $k(\mathbf{x}, \mathbf{x}') = \exp\{-(|\mathbf{x} - \mathbf{x}'|^2 / l^2)\}$, but with different characteristic length-scale l. The left figure has smaller l while the right figure has larger l. The kernel is graphically depicted by the red dotted line.

The same procedure can be applied to the posterior function, but using the posterior mean and covariance from eq. (13.2.1).

To learn more

This has been a very brief introduction to Gaussian Process regression. A good place to learn more is [1].

13.3 Choosing the next point to evaluate

For this section, we will switch notation to match the RL scenario:

- $f \to J$
- $\mathbf{x} \to \theta$

$J(\theta)$ is the "true cost-to-go" function, which is unknown but what we're trying to optimize as function of θ, the parameter vector for the policy π_θ. To evaluate $J(\theta)$ means to perform one (or more) episodes of simulation using the policy defined by θ.

Now, given all the samples $\big((\theta_1, J(\theta_1)), \dots (\theta_t, J(\theta_t))\big)$ observed so far and and a GP prior, we can estimate the posterior $J(\theta)$ for any query θ. Let us define

$$\begin{aligned} J(\theta) &\sim \mathcal{N}(\mu_t(\theta), \sigma_t(\theta)) \\ \mu_t(\theta) &\doteq \text{posterior mean for } J(\theta) \text{ according to eq. (13.2.2)} \\ \sigma_t(\theta) &\doteq \text{posterior variance of } J(\theta) \text{ according to eq. (13.2.3)} \end{aligned}$$

How do we choose the next point θ_{t+1} to evaluate (step (c) in the RSM algorithm outline)? Below we list some possible strategies, each with different advantages and disadvantages.

Maximum of posterior mean

In this strategy, we simply use

$$\theta_{t+1} = \operatorname*{argmax}_\theta \mu_t(\theta)$$

To obtain the argmax we can use any optimization algorithm, such as gradient descent, exhaustive search, Nelder-Mead, etc. (Note that this implies we are performing a nonlinear optimization step for each iteration of RSM, although on our estimated surrogate function, which is one of the reasons it is relatively slow).

This is the most "greedy" strategy, maximizing exploitation over exploration. As such it is more prone to converge in local optima. It may even choose $\theta_{t+1} = \theta_t$, in which case we will get stuck. Moreover, it ignores the uncertainty of the estimate. Nonetheless, it may be a good choice if we believe θ_t is close to a good optima.

Figure 13.3.1 shows this strategy in action. The first panel shows the true objective function $J(\theta)$, in black, and three evaluations (black dots). The rest are seven iterations of the RSM algorithm, ordered from left to right and top to bottom. The solid red line is the expected posterior mean of $J(\theta)$, $\mu_t(\theta)$. The dotted red lines are a confidence interval for $J(\theta)$, reflecting the magnitude of $\sigma_t(\theta)$. The blue line is the normalized value of the function we maximize to obtain θ_{t+1}, in this case μ_t. As we can see, in this case the strategy does nothing of interest, as the θ_{t+1} (the small black dot) equals one of the previously sampled points, and no further exploration occurs; the maximum is not found.

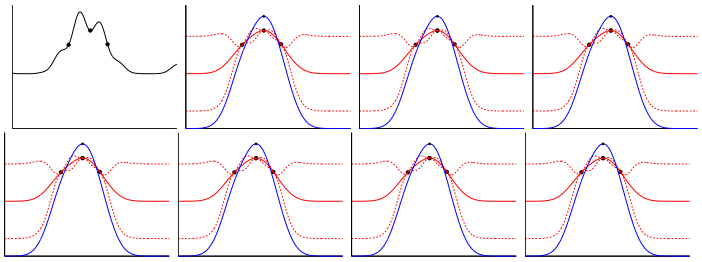

Figure 13.3.1: The "maximum of posterior mean" strategy in action. See text for explanation.

Maximum of posterior variance

In this strategy, we use

$$\theta_{t+1} = \operatorname*{argmax}_{\theta} \sigma_t(\theta)$$

That is, θ_{t+1} is placed where our uncertainty about $J(\theta)$ is greatest, regardless of the expected value. This is the "opposite" of the last strategy in the exploitation-exploration spectrum.

Figure 13.3.2 shows this strategy. We can see this strategy is good for exploring over various different locations, but fails to find the function maximum.

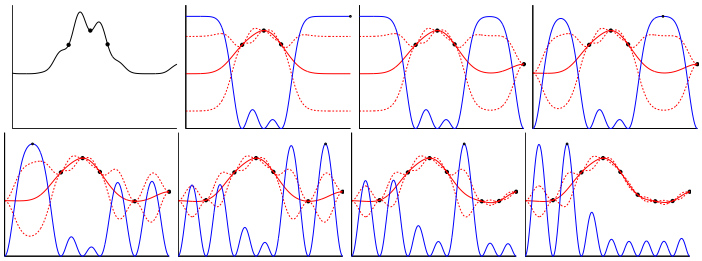

Figure 13.3.2: The "maximum of posterior variance" strategy in action. See text for explanation.

Maximum upper confidence bound

In practice, we want to balance exploitation and exploration. Therefore, a sensible strategy is to choose

$$\theta_{t+1} = \operatorname*{argmax}_{\theta} \mu_t(\theta) + \beta\, \sigma_t(\theta)$$

where β is a parameter regulating the exploitation-exploration tradeoff. $\mu_t(\theta) + \beta\, \sigma_t(\theta)$ can be interpreted as an "upper confidence bound" for $\mu_t(\theta)$. This strategy works well, but has the disadvantage of requiring a tuning parameter β.

Maximum probability of improvement

What we really want is to improve $J_{\max}$, the best value found so far.

Let us define an "improvement" function

$$I(\theta) = \max\big(J(\theta) - J_{\max}, 0\big)$$

Since $I(\cdot)$ depends on $J(\theta)$, it is also a random variable. One strategy is to maximize the *probability of improvement*:

$$\theta_{t+1} = \operatorname*{argmax}_{\theta} \Pr\big(I(\theta) > 0\big)$$

or alternatively,

$$\theta_{t+1} = \operatorname*{argmax}_{\theta} \int_{J_{\max}}^{\infty} \mathcal{N}\big(y|\mu_t(\theta), \sigma_t(\theta)\big)\mathrm{d}y$$

The main problem with this strategy is that if we ever choose $\theta_{t+1} = \theta_t$, we we will get stuck there. To avoid this we can use a modified improvement function

$$I(\theta_t) = \max\big(J(\theta_t) - (J_{\max} + \beta), 0\big)$$

But again, we have a tuning parameter β.

Maximum expected improvement

One problem with the last strategy is that we are only considering the probability of improvement, not the magnitude of the improvement. We can fix this by using the expected improvement:

$$\begin{aligned}\theta_{t+1} &= \operatorname*{argmax}_{\theta} \mathbb{E}\left[I(\theta)\right] \\ &= \operatorname*{argmax}_{\theta} \int_{J_{\max}}^{\infty} \mathcal{N}\big(y|\mu_t(\theta), \sigma_t(\theta)\big)\,\big(y - J_{\max}\big)\mathrm{d}y\end{aligned}$$

This has no tuning parameter and is one of the most popular RSM methods. One disadvantage is that it sometime tends to explore too much. Figure 13.3.3 shows this strategy in action. As we can see it has a good balance of exploration and exploitation, and finds the maximum.

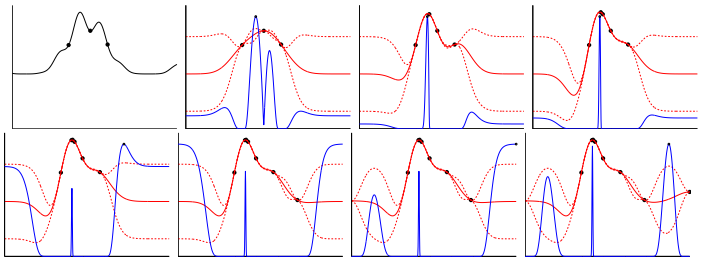

Figure 13.3.3: The "maximum of expected improvement" strategy in action. See text for explanation.

Miscellaneous

- For any strategy, at each step we can sample a random point with some probability ϵ. This encourages exploration.
- How do we take into account the stochasticity of $J(\theta)$ itself? Often it is simply ignored, which conflates it with our uncertainty about the value $J(\theta)$. Alternatively, it can be separately modelled as "process noise", which may be heteroscedastic (it varies with θ).
- We may take $J_{\max}$ to be stochastic. In this case the expected improvement requires a joint expectation calculation. Most people simply use the mean of the GP at the maximum point. Empirically it doesn't seem to matter too much.

13.4 Related Reading

[1] Carl Edward Rasmussen and Christopher K. I. Williams, *Gaussian Processes for Machine Learning*. MIT Press, 2006. www.gausianprocess.org