Reinforcement Learning — Homework 2

Tarun Krishna **Dhruba Pujary**
11593040 11576200
University of Amsterdam
{dhruba.pujary, tarun.krishna}@student.uva.nl

1 Gradient Descent Methods

1. We know that returns are definde as:

$$G_t \doteq \sum_{k=0}^{\tau-t-1} \gamma^k R_{t+k+1}$$

Also value of a state under policy by π denoted as $v_\pi(s)$ is defined as:

$$\begin{aligned} v_\pi(s) &\doteq \mathbb{E}_\pi[G_t | s_t = s] \\ &= \mathbb{E}_\pi[\sum_{k=0}^{\tau-t-1} \gamma^k R_{t+k+1} | s_t = s] \end{aligned}$$

The true value function that we want to estimate is the expected return in state $s_t = s$ and following policy π. The Monte Carlo target is clearly a direct sample of this value, and has the same expected value that we are looking for. Hence it is not biased.

2. Well this is not only the case with DP methods, in general it applies to all the bootstrapping methods such as n-step returns $G_{t:t+n}$ etc. Bootstrapping targets such as DP target all depend upon the current value of the weight vector $\mathbf{w}_t$, which implies that they will be biased and that they will not produce a true gradient-descent method. One way to look at this is that the key step in the equation below relies on the target being independent of $\mathbf{w}_t$. This step would not be valid if a bootstrapping estimate were used in place of $v_\pi(S_t)$. Bootstrapping methods are not in fact instances of true gradient descent. They take into account the effect of changing the weight vector $\mathbf{w}_t$ on the estimate, but ignore its effect on the target. They include only a part of the gradient and, accordingly, we call them semi-gradient methods.

$$\begin{aligned} \mathbf{w}_{t+1} &\doteq \mathbf{w}_t - \frac{1}{2}\alpha\nabla\Big[v_\pi(S_t) - \hat{v}_\pi(S_t, \mathbf{w}_t)\Big]^2 \\ &= \mathbf{w}_t + \alpha\Big[v_\pi(S_t) - \hat{v}_\pi(S_t, \mathbf{w}_t)\Big]\nabla\hat{v}_\pi(S_t, \mathbf{w}_t) \end{aligned}$$

3. An important aspect of bootstrapping is that they enable learning to be continual and online, without waiting for the end of an episode. This enables them to be used on continuing problems and provides computational advantages. Mountain car problem is a continuous control problem where an episode ends when car reaches at the top of the mountain. This could be solved using MC methods but reaching the end of an episode or getting true estimate out of a single episode is highly unlikely and inefficient. The agents needs to identify the sequence of action such that it first moves away from the goal to gather momentum and climb the mountain. Thus, one would want to learn at every time step using bootstrapping methods.

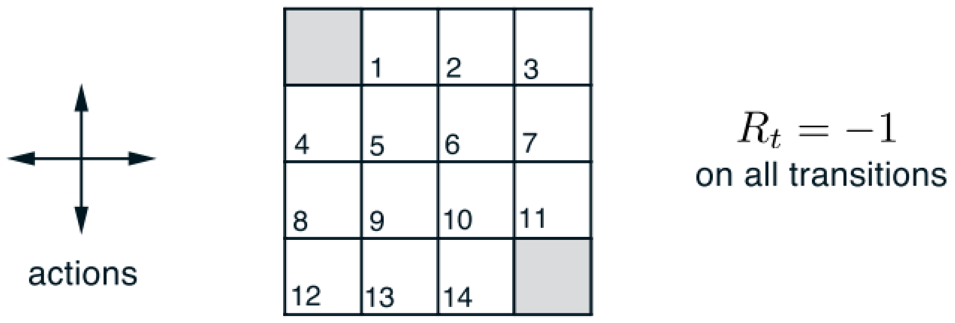

Figure 1

2 Basis functions

1. Using table lookup features:

$$\mathbf{x}^{table}(S) = \begin{pmatrix} \mathbf{1}(S = s_1) \\ \mathbf{1}(S = s_2) \\ \vdots \\ \mathbf{1}(S = s_n) \end{pmatrix}$$
$$\hat{v}(S, \mathbf{w}) = \mathbf{w}^T \mathbf{x}^{table}(S)$$

   Parameter vector $\mathbf{w}$ gives value of each individual state. This can be validated with a simple example from Figure 1, which consists of 14 non-terminal states. We can calculate $\mathbf{x}^{table}(14)$ as $[0,0,0,0,0,0,0,0,0,0,0,0,0,1]^T$ and $\hat{v}(14, \mathbf{w}) = \mathbf{w}_{14}$

2. To form a polynomial feature vector we can simply formalize each feature vector as $x_i(s) = \prod_{j=1}^{k} s_j^{c_{i,j}}$. Where $c_{i,j}$ is an integer in the set $0, 1, 2...n$ for an integer $n \geq 0$. For each state s corresponds to k numbers, $s_1, s_2...s_k$. For $s = [x, y]$ we can formalize $\mathbf{x}(s) = [1, x, y, xy, x_1^2, y_2^2, xy^2, x_1^2 y, x_1^2 y_2^2]^T$.

3. The number of features in an order-n polynomial basis grows exponentially with the dimension k of the natural state space (if $n > 0$).

4. Given that one of the state dimension has more performance impact than the other. This implies the effect of changing the values in former dimension is more than changing on the other. For example, let $s = [s_0, s_1]^T$ and changing s_0 has more impact than s_1. This can be encoded by defining the basis function as $s' = [s_0, \alpha(s_1 - s_0) + s_1]$ where α is some constant. Similar higher-order polynomial could be defined encoding the prior information about the task at hand.

5. In coarse coding, we define a coding vector as each features being present or absent. These features could be defined using different geometrical shapes and if the state lies withing the boundary of any of the shape, then that feature is marked present. One restriction to this approach is that the fixed boundary or region of influence of the geometrical shapes. The feature value can only be present or absent; binary. The Radial Basis function is generalization of coarse coding with feature values can be within interval [0,1]. This can also be interpreted as the geometrical shapes no more has a fixed boundary and is rather continuous in space. Each state can be represented in a vector with each dimension telling how close it is to all other features.

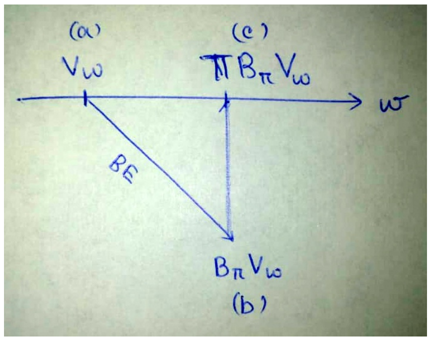

Figure 2

3 Geometry of linear value-function approximation

1.

$$\begin{aligned}\bar{\delta}_w &= \mathbb{B}^\pi v_w - v_w \\ &= \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)[r+\gamma v_w(s')] - v_w(s_0)\end{aligned}$$

$$\begin{bmatrix}\bar{\delta}_w(s_0)\\ \bar{\delta}_w(s_1)\end{bmatrix} = \begin{bmatrix}1\times 1\times[0+1\times 2]-1\\ 1\times 1\times[0+1\times 1]-2\end{bmatrix} = \begin{bmatrix}1\\ -1\end{bmatrix}$$

2. Mean squared Bellmen error, $\frac{1}{2}\times(1^2+(-1)^2)=1$
3. For w equals 0, we can have v_w as close to $\mathbb{B}^\pi v_w$ with zero least-square error. This is true in the sense that all the rewards of the MDP is zero, in this case, so the true value function v_π is also zero.
4. Figure 2 plots the $v_w, \mathbb{B}^\pi v_w$ and $\Pi\mathbb{B}^\pi v_w$.
   The value of v_w in w dimension is 1 as noted in Figure 2 point (a). The value in the 2-D state space is $(1,2)$ is we consider $(x,y)=(s_0,s_1)$. Following the Bellmen error, $[1,-1]^T$, we reach point (b),$\mathbb{B}^\pi v_w$ and its projection back in the w's 1-D space as shown in Figure 2 point (c). The value of $\mathbb{B}^\pi v_w$ in state space is $(2,1)$ and its projection, point (c), would be approximately at $(1,1)$.

4 Neural Networks

1. $\mu(s)$ is defined as *on-policy distribution*, which is created while following the policy being evaluated (describes the frequency with which states are encountered). This often changes as we change value functions as it depends on the parameters. As a result indirectly changing the policy affects one-policy distribution which depends on parameters taken into consideration.
2. Unsupervised learning is a geometric problem. Meaning, it tries to understand the structure of the problem in order to discover how to partition the data meaningfully.

   Supervised learning (SL) is all about how to partition the data based on labels. In a sense, the structure is much less important, and the labels are the critical hints of how we should partition the data.

   Between them there is semi-supervised learning, where not all samples have labels. Thus, it is something between geometric problem structure to constraining the samples partition by the labels.

   In principle, we can use any method for supervised learning from examples, including artificial neural networks, decision trees, and various kinds of multivariate regression. However, not all function approximation methods are equally well suited for use in reinforcement learning. The most sophisticated artificial neural network and statistical methods all assume a static training set over which multiple passes are made. In reinforcement learning, however, it is important that learning be able to occur online, while the agent interacts with its environment or with a model of its environment. To do this requires methods that are able to learn efficiently from incrementally acquired data. In addition, reinforcement learning generally requires function approximation methods able to handle non-stationary target functions (target functions that change over time). For example, in control methods based on GPI (generalized policy iteration) we often seek to learn q_π while π changes. Even if the policy remains the same, the target values of training examples are nonstationary if they are generated by bootstrapping methods (DP and TD learning). Methods that cannot easily handle such nonstationarity are less suitable for reinforcement learning.

3. With function approximation, an update at one state affects many others, and it is not possible to get the values of all states exactly correct. By assumption we have far more states than weights, so making one state's estimate more accurate invariably means making others' less accurate. We are obligated then to say which states we care most about. As a result we must specify how much we care about the state by their state distribution $\mu(s)$. As a result when we calculate objective we weight error with respect to the importance of that state given by it's state distribution. Also it is like how much we care about the error in particular state i.e if the errors are high for a particular state and state occurs too often then we need to penalize it accordingly. Basically, $\mu(s)$ distribution specifies how these trade-offs should be made.

4. Stochastic gradient descent works best with independent and identically distributed samples. But in reinforcement learning, we receive sequential samples from interactions with the environment.

   That causes the networks to see too many samples of one kind and forget the others. For example, the agent might reach the second level of a game. It looks or behaves totally different, so it forgets how to play the first level.

   Storing all experience in a replay buffer allows us to train on more independent samples. We just draw a batch of transitions from the buffer at random and train on that. This helps break the temporal correlation of training samples.

5. If we don't use a target network: The issue is that at every step of training, the Q-network's values shift, and if we are using a constantly shifting set of values to adjust our network values, then the value estimations can easily spiral out of control. The network can become destabilized by falling into feedback loops between the target and estimated Q-values. In order to mitigate that risk, the target network's weights are fixed, and only periodically or slowly updated to the primary Q-networks values. In this way training can proceed in a more stable manner.

5 Policy Gradient

5.1 Reinforce

$$\nabla_\theta J = \mathbb{E}_\tau\left[\big(G(\tau) - b\big)\nabla \log p_\theta(\tau)\right]$$

1.

$$\mathrm{Var}\big[(\nabla_\theta J)^2\big] = \mathbb{E}\big[\nabla_\theta J)^2\big] - \mathbb{E}\big[\nabla_\theta J\big]^2$$

$$\frac{\partial}{\partial b}\mathrm{Var}\big[(\nabla_\theta J)^2\big] = 0$$

$$\frac{\partial}{\partial b}\left(\mathbb{E}\big[(\nabla_\theta J)^2\big] - \mathbb{E}\big[\nabla_\theta J\big]^2\right) = \frac{\partial}{\partial b}\mathbb{E}\big[(\nabla_\theta J)^2\big] - \frac{\partial}{\partial b}\mathbb{E}\big[\nabla_\theta J\big]^2 = 0$$

$$
\begin{aligned}
\mathbb{E}[\frac{\partial}{\partial b}(\nabla_\theta J)^2] - 2\underbrace{\mathbb{E}[[\nabla_\theta J]\frac{\partial}{\partial b}\mathbb{E}[\nabla_\theta J]}_{\mathbf{E}=0} &= 0\\
\mathbb{E}[\frac{\partial}{\partial b}((G(\tau)-b)\nabla\log p_\theta(\tau))^2] &= 0\\
\mathbb{E}[-2(G(\tau)-b)\nabla\log p_\theta(\tau))\nabla\log p_\theta(\tau)] &= 0\\
\mathbb{E}[G(\tau)(\nabla\log p_\theta(\tau))^2] - b\mathbb{E}[(\nabla\log p_\theta(\tau))^2] &= 0\\
\frac{\mathbb{E}[G(\tau)(\nabla\log p_\theta(\tau))^2]}{\mathbb{E}[(\nabla\log p_\theta(\tau))^2]} &= b
\end{aligned}
$$

E=0 because $\mathbb{E}\big[\nabla_\theta J\big]$ is an unbiased estimate of b (which we will prove later in this question) hence it's derivative is 0 with respect to b.
The optimal value of b minimizes the variance of gradient of J. It evaluates to be ratio of expected reward of a trajectory times the square of the gradient of log probability of that trajectory to expected value of square of gradients of log probability. In other words, baseline is how much is the expected rewards by following all the trajectory on an average.

2.

$$
\begin{aligned}
b &= \frac{\mathbb{E}[G(\tau)(\nabla\log p_\theta(\tau))^2]}{\mathbb{E}[(\nabla\log p_\theta(\tau))^2]}\\
&= \frac{\mathbb{E}[(a+2)(a-\theta)^2]}{\mathbb{E}[(a-\theta)^2]}\\
&= \frac{\mathbb{E}[a^3-2a^2\theta+a\theta^2+2a^2-4a\theta+2\theta^2]}{\mathbb{E}[a^2-2a\theta+\theta^2]}
\end{aligned}
$$

using raw moments of univariate distributions,

$$
\begin{aligned}
\mathbb{E}_x[x^2] &= \mu^2+\sigma^2, \mathbb{E}_x[x^3] = \mu^3+3\mu\sigma^2\\
b &= \frac{\theta^3+3\theta-2(\theta^2+1)\theta+\theta^3+2(\theta^2+1)-4\theta^2+2\theta^2}{\theta^2+1-2\theta^2+\theta^2}\\
&= \theta+2
\end{aligned}
$$

3. **NOTE**: It doesn't make sense to have summation upto T, if my trajectory is of length T. It can be verified from [1] as well.

Due to linearity of expectation, all we need to show is that for any single time t, the gradient of $\log\pi_\theta(a_t|s_t)$ multiplied by $b(s_t)$ is zero.

$$
\begin{aligned}
\mathbb{E}_\tau\big[\nabla_\theta\log\pi(a_t|s_t)b(s_t)\big] &= \mathbb{E}_{s_{0:t},a_{0:t-1}}\Bigg[\mathbb{E}_{s_{t+1:T},a_{t:T-1}}\Bigg[\nabla_\theta\log\pi(a_t|s_t)b(s_t)\Bigg]\Bigg]\\
&= \mathbb{E}_{s_{0:t},a_{0:t-1}}\Bigg[b(s_t)\underbrace{\mathbb{E}_{s_{t+1:T},a_{t:T-1}}\Bigg[\nabla_\theta\log\pi(a_t|s_t)\Bigg]}_{\text{E}}\Bigg]\\
&= \mathbb{E}_{s_{0:t},a_{0:t-1}}\Bigg[b(s_t)\mathbb{E}_{a_t}\big[\nabla_\theta\log\pi(a_t|s_t)\big]\Bigg]\\
&= \mathbb{E}_{s_{0:t},a_{0:t-1}}\Bigg[b(s_t)\int\frac{\nabla_\theta\pi_\theta(a_t|s_t)}{\pi_\theta(a_t|s_t)}\pi_\theta(a_t|s_t)da_t\Bigg]\\
&= \mathbb{E}_{s_{0:t},a_{0:t-1}}\Bigg[b(s_t)\nabla_\theta\int\pi_\theta(a_t|s_t)da_t\Bigg]\\
&= \mathbb{E}_{s_{0:t},a_{0:t-1}}\Bigg[b(s_t)\nabla_\theta\cdot 1\Bigg]\\
&= 0
\end{aligned}
$$

E can be expanded as :

$$\begin{aligned}
E &= \sum_{a_t \in \mathcal{A}} \sum_{s_{t+1} \in \mathcal{S}} \cdots \sum_{s_T \in \mathcal{S}} \underbrace{\pi_\theta(a_t|s_t) P(s_{t+1}|s_t, a_t) \cdots P(s_T|s_{T-1}, a_{T-1})}_{p((a_t, s_{t+1}, a_{t+1}, \ldots, a_{T-1}, s_T))} (\nabla_\theta \log \pi_\theta(a_t|s_t)) \\
&= \sum_{a_t \in \mathcal{A}} \pi_\theta(a_t|s_t) \nabla_\theta \log \pi_\theta(a_t|s_t) \sum_{s_{t+1} \in \mathcal{S}} P(s_{t+1}|s_t, a_t) \sum_{a_{t+1} \in \mathcal{A}} \cdots \sum_{s_T \in \mathcal{S}} P(s_T|s_{T-1}, a_{T-1}) \\
&= \sum_{a_t \in \mathcal{A}} \pi_\theta(a_t|s_t) \nabla_\theta \log \pi_\theta(a_t|s_t)
\end{aligned}$$

Using summation just to avoid integrals in order to avoid writing $da_t ds_t \ldots . ds_T$.

5.2 Compatible Function Approximation Theorem

$$\nabla J = \mathbb{E}_\tau \big[\sum_{t=1}^{T} \nabla_\theta \log \pi(a_t|s_t) q_\pi(s_t, a_t) \big] \tag{1}$$

Again in this question it doesn't make sense to have summation upto T.

1. For a given state s:

$$\begin{aligned}
\mathbb{E}_{a \sim \pi_\theta}[\hat{q}_w(s, a)] &= \int_A \hat{q}_w(s, a) \pi_\theta(a|s) da \\
&= \int_A \pi_\theta(a|s) w^T \nabla_\theta \log \pi_\theta(a|s) da \\
&= \int_A \pi_\theta(a|s) w^T \frac{\nabla_\theta \pi_\theta(a|s)}{\pi_\theta(a|s)} da \\
&= \int_A w^T \nabla_\theta \pi_\theta(a|s) da \\
&= w^T \nabla_\theta \int_A \pi_\theta(a|s) da \\
&= w^T \nabla_\theta \cdot 1 \\
&= 0
\end{aligned}$$

   It seems that approximation of q_π has a zero mean.

2. For a given state s:

$$\begin{aligned}
\mathbb{E}_{a \sim \pi_\theta}[q(s, a) - v(s)] &= \mathbb{E}_{a \sim \pi_\theta}[q(s, a)] - \mathbb{E}_{a \sim \pi_\theta}[v(s)] \\
&= \mathbb{E}_{a \sim \pi_\theta}[q(s, a)] - \int_A \pi(a|s) v(s) da \\
&= \mathbb{E}_{a \sim \pi_\theta}[q(s, a)] - v(s)
\end{aligned}$$

3. From 2, we have $\mathbb{E}_{a \sim \pi_\theta}[q_\pi(s, a) - v(s)] = \mathbb{E}_{a \sim \pi_\theta}[A(s, a)] = \mathbb{E}_{a \sim \pi_\theta}[q_\pi(s, a)] - v(s)$. We can approximate $q_\pi(s, a)$ by $\hat{q}_w(s, a)$ and by considering result of 1, we get $\mathbb{E}_{a \sim \pi_\theta}[A(s, a)] = -v(s)$.

4.

$$\pi_\theta(a|s) = \frac{e^{\theta^T\phi_{sa}}}{\sum_b e^{\theta^T\phi_{sb}}}$$

$$\begin{aligned}
\hat{q}_w(s,a) &= w^T\Big[\nabla_\theta \log \pi_\theta(a|s)\Big] \\
&= w^T\Big[\nabla_\theta \log\Big[\frac{e^{\theta^T\phi_{sa}}}{\sum_b e^{\theta^T\phi_{sb}}}\Big]\Big] \\
&= w^T\Big[\nabla_\theta \log e^{\theta^T\phi_{sa}} - \nabla_\theta \log \sum_b e^{\theta^T\phi_{sb}}\Big] \\
&= w^T\Big[\nabla_\theta \theta^T\phi_{sa} - \nabla_\theta \log \sum_b e^{\theta^T\phi_{sb}}\Big] \\
&= w^T\Big[\phi_{sa} - \frac{\sum_b \nabla_\theta e^{\theta^T\phi_{sb}}}{\sum_b e^{\theta^T\phi_{sb}}}\Big] \\
&= w^T\Big[\phi_{sa} - \frac{\sum_b e^{\theta^T\phi_{sb}}\phi_{sb}}{\sum_b e^{\theta^T\phi_{sb}}}\Big] \\
&= w^T\Big[\phi_{sa} - \sum_b \frac{e^{\theta^T\phi_{sb}}\phi_{sb}}{\sum_b e^{\theta^T\phi_{sb}}}\Big] \\
&= w^T\Big[\phi_{sa} - \sum_b \pi_\theta(b|s)\phi_{sb}\Big]
\end{aligned}$$

5.3 Natural Gradient

1. (a)

$$\begin{aligned}
\frac{\partial \log \pi(a|\theta)}{\partial \theta_\mu} &= \frac{\partial}{\partial \theta_\mu}\log\left(\frac{1}{\exp(\theta_\sigma)\sqrt{2\pi}}\exp\big(-\frac{(a-\theta_\mu)^2}{2\exp{(\theta_\sigma)}^2}\big)\right) \\
&= \frac{\partial}{\partial \theta_\mu}\log\left(\frac{1}{\exp{(\theta_\sigma)}\sqrt{2\pi}}\right) + \frac{\partial}{\partial \theta_\mu}\left(-\frac{(a-\theta_\mu)^2}{2\exp{(\theta_\sigma)}^2}\right) \\
&= 0 + \frac{a-\theta_\mu}{\exp{(2\theta_\sigma)}}
\end{aligned}$$

(b)

$$\begin{aligned}
\frac{\partial \log \pi(a|\theta)}{\partial \theta_\sigma} &= \frac{\partial}{\partial \theta_\sigma}\log\left(\frac{1}{\exp(\theta_\sigma)\sqrt{2\pi}}\exp\big(-\frac{(a-\theta_\mu)^2}{2\exp{(\theta_\sigma)}^2}\big)\right) \\
&= \frac{\partial}{\partial \theta_\sigma}\log\left(\frac{1}{\exp{(\theta_\sigma)}\sqrt{2\pi}}\right) + \frac{\partial}{\partial \theta_\sigma}\left(-\frac{(a-\theta_\mu)^2}{2\exp{(\theta_\sigma)}^2}\right) \\
&= -1\left(\frac{\partial}{\partial \theta_\sigma}(\theta_\sigma) + \frac{\partial}{\partial \theta_\sigma}(\log\sqrt{2\pi})\right) + \frac{(a-\theta_\mu)^2}{\exp{(\theta_\sigma)}} \\
&= \frac{(a-\theta_\mu)^2}{\exp{(2\theta_\sigma)}} - 1
\end{aligned}$$

2. Vanilla policy gradients takes small steps in the direction of the best policy. It maximizes the Taylor expansion of the performance function J such that the updates lie on the norm sphere. However, the euclidean norm is sensitive to parameterisation. Natural policy gradients are covariant that it they are independent of the choice of parameterisation. This reduces the time consumed for parameter tuning. The vanilla gradient reduces the variance of the policy quickly and hence stops exploring where as natural gradient decreases the variance gradually, and in the end finds the optimal solution fast.

3.

$$\begin{aligned}
[\nabla_\theta \log \pi(a|\theta_0 + \partial\theta)] &= \begin{bmatrix} \frac{\partial \log \pi(a|\theta)}{\partial \theta_\mu} \\ \frac{\partial \log \pi(a|\theta)}{\partial \theta_\sigma} \end{bmatrix} \\
&= \begin{bmatrix} \frac{a-\theta_\mu}{\exp(2\theta_\sigma)} \\ \frac{(a-\theta_\mu)^2}{\exp(2\theta_\sigma)} - 1 \end{bmatrix} \\
\left[\nabla_\theta \log \pi(a|\theta_0 + \partial\theta) \nabla_\theta \log \pi(a|\theta_0 + \partial\theta)^T\right] &= \begin{bmatrix} \frac{a-\theta_\mu}{\exp(2\theta_\sigma)} \\ \frac{(a-\theta_\mu)^2}{\exp(2\theta_\sigma)} - 1 \end{bmatrix} \begin{bmatrix} \frac{a-\theta_\mu}{\exp(2\theta_\sigma)} & \frac{(a-\theta_\mu)^2}{\exp(2\theta_\sigma)} - 1 \end{bmatrix} \\
&= \begin{bmatrix} (\frac{a-\theta_\mu}{\exp(2\theta_\sigma)})^2 & (\frac{a-\theta_\mu}{\exp(2\theta_\sigma)})(\frac{(a-\theta_\mu)^2}{\exp(2\theta_\sigma)} - 1) \\ (\frac{a-\theta_\mu}{\exp(2\theta_\sigma)})(\frac{(a-\theta_\mu)^2}{\exp(2\theta_\sigma)} - 1) & (\frac{(a-\theta_\mu)^2}{\exp(2\theta_\sigma)} - 1)^2 \end{bmatrix} \\
&= \begin{bmatrix} g_{11} & g_{12} \\ g_{21} & g_{22} \end{bmatrix} \\
\mathrm{E}_a[g_{11}] &= \mathrm{E}_a\left[(\frac{a-\theta_\mu}{\exp(2\theta_\sigma)})^2\right] \\
&= \frac{1}{\exp(\theta_\sigma)^4}\mathrm{E}_a\left[(a^2 - 2a\theta_\mu + \theta_\mu^2)\right] \\
&= \frac{1}{\exp(\theta_\sigma)^4}\left[\mathrm{E}_a[a]^2 - 2\mathrm{E}_a[a]\theta_\mu + \theta_\mu^2\right]
\end{aligned}$$

$$\begin{aligned}
\text{Using raw moments, } \mathrm{E}_x[x^2] &= \mu^2 + \sigma^2 \text{ for univariate normal distribution,} \\
&= \frac{1}{\exp(\theta_\sigma)^4}\left[\theta_\mu^2 + \exp(\theta_\sigma)^2 - 2\theta_\mu^2 + \theta_\mu^2\right] \\
&= \frac{1}{\exp(\theta_\sigma)^2} \\
\mathrm{E}_a[g_{12}] &= \mathrm{E}_a\left[(\frac{a-\theta_\mu}{\exp(2\theta_\sigma)})(\frac{(a-\theta_\mu)^2}{\exp(2\theta_\sigma)} - 1)\right] \\
&= \mathrm{E}_a\left[\frac{(a-\theta_\mu)^3}{\exp(\theta_\sigma)^4} - \frac{(a-\theta_\mu)}{\exp(\theta_\sigma)^2}\right] \\
&= \mathrm{E}_a\left[\frac{a^3 - 3a^2\theta_\mu + 3a\theta_\mu^2 - \theta_\mu^3}{\exp(\theta_\sigma)^4}\right] - \mathrm{E}_a\left[\frac{a-\theta_\mu}{\exp(\theta)^2}\right] \\
&= \frac{\mathrm{E}_a[a^3] - 3\mathrm{E}_a[a^2]\theta_\mu + 3\mathrm{E}_a[a]\theta_\mu^2 - \theta_\mu^3}{\exp(\theta_\sigma)^4} - \frac{\mathrm{E}_a[a] - \theta_\mu}{\exp(\theta_\sigma)^2} \\
&= \theta_\mu^3 + 3\theta_\mu \exp(\theta_\sigma)^2 - 3[\theta_\mu^2 + \exp(\theta_\sigma)^2]\theta_\mu + 3\theta_\mu^3 - \theta_\mu^3 \\
&= 0 \\
\mathrm{E}_a[g_{22}] &= \mathrm{E}_a\left[\left(\frac{(a-\theta_\mu)^2}{\exp(\theta_\sigma)^2} - 1\right)^2\right] \\
&= \mathrm{E}_a\left[\frac{(a-\theta_\mu)^4}{\exp(\theta_\sigma)^4} - 2\frac{(a-\theta_\mu)^2}{\exp(\theta_\sigma)^2} + 1\right]
\end{aligned}$$

$$= \frac{\mathrm{E}_a[(a-\theta_\mu)^4]}{\exp{(\theta_\sigma)}^4} - 2\frac{\mathrm{E}_a[(a-\theta_\sigma)^2]}{\exp{(\theta_\sigma)}^2} + 1$$

Evaluating the first term,

$$\begin{aligned}
\frac{\mathrm{E}_a[(a-\theta_\mu)^4]}{\exp{(\theta_\sigma)}^4} &= \frac{\mathrm{E}_a[a^4 - 4a^3\theta_\mu + 6a^2\theta_\mu^2 - 4a\theta_\mu^3 + \theta_\mu^4]}{\exp{(\theta_\sigma)}^4} \\
&= \frac{1}{\exp{(\theta_\sigma)}^4}\Big[\theta_\mu^4 + 6\theta_\mu^2\exp{(\theta_\sigma)}^2 + 3\exp{(\theta_\sigma)}^4 - 4[\theta_\mu^3 + 3\theta_\mu\exp{(\theta_\sigma)}^2]\theta_\mu + \\
&\quad 6[\theta_\mu^2 + \exp{(\theta_\sigma)}^2]\theta_\mu^2 - 4\theta_\mu^4 + \theta_\mu^4\Big] \\
&= 3
\end{aligned}$$

Evaluating the remaining terms, $= -$

$$\begin{aligned}
&= -2\frac{\theta_\mu^2 + \exp(\theta_\sigma)^2 - 2\theta_\mu^2 + \theta_\mu^2}{\exp\theta^2} + 1 \\
&= -1 \\
\mathrm{E}_a[g_{22}] = 3 - 1 &= 2 \\
\mathrm{F} &= \begin{bmatrix} \frac{1}{\exp{(\theta_\sigma)}^2} & 0 \\ 0 & 2 \end{bmatrix}
\end{aligned}$$

4. The inverse of Fisher Information Matrix is, $\mathrm{F}^{-1} = \begin{bmatrix} \exp(2\theta_\sigma) & 0 \\ 0 & \frac{1}{2} \end{bmatrix}$, in which the component that will change the θ_μ is $\exp{(2\theta_\sigma)}$. Thus, instead of directly updating in the direction of the gradient of the θ_μ, the updates will be scaled and will happen in the direction depending on $\exp{(2\theta_\sigma)}$.

5.

$$\begin{aligned}
\frac{\partial \log \pi(a|\theta)}{\partial \theta_\mu} &= \frac{a-\theta_\mu}{\theta_\sigma^2} \\
\frac{\partial \log \pi(a|\theta)}{\partial \theta_\sigma} &= \frac{\partial}{\partial \theta_\sigma}\log\left(\frac{1}{\theta_\sigma}\right) + \frac{\partial}{\partial \theta_\sigma}\left(\log\frac{1}{\sqrt{2\pi}}\right) + \frac{\partial}{\partial \theta_\sigma}\left(-\frac{(a-\theta_\mu)^2}{2\theta_\sigma^2}\right) \\
&= \frac{(a-\theta_\mu)^2}{\theta_\sigma^3} - 1
\end{aligned}$$

$$\begin{aligned}
\left[\nabla_\theta \log \pi(a|\theta_0 + \partial\theta)\nabla_\theta \log \pi(a|\theta_0 + \partial\theta)^T\right] &= \begin{bmatrix} \frac{a-\theta_\mu}{\theta_\sigma^2} \\ \frac{(a-\theta_\mu)^2}{\theta_\sigma^3} - 1 \end{bmatrix}\begin{bmatrix} \frac{a-\theta_\mu}{\theta_\sigma^2} & \frac{(a-\theta_\mu)^2}{\theta_\sigma^3} - 1 \end{bmatrix} \\
&= \begin{bmatrix} (\frac{a-\theta_\mu}{\theta_\sigma^2})^2 & (\frac{a-\theta_\mu}{\theta_\sigma^2})(\frac{(a-\theta_\mu)^2}{\theta_\sigma^3} - 1) \\ (\frac{a-\theta_\mu}{\theta_\sigma^2})(\frac{(a-\theta_\mu)^2}{\theta_\sigma^3} - 1) & (\frac{(a-\theta_\mu)^2}{\theta_\sigma^3} - 1)^2 \end{bmatrix} \\
&= \begin{bmatrix} g_{11} & g_{12} \\ g_{21} & g_{22} \end{bmatrix}
\end{aligned}$$

$$
\begin{aligned}
\mathrm{E}_a[g_{11}] &= \frac{1}{\theta_\sigma^2} \\
\mathrm{E}_a[g_{12}] &= \mathrm{E}_a\left[(\frac{a-\theta_\mu}{\theta_\sigma^2})(\frac{(a-\theta_\mu)^2}{\theta_\sigma^3}-1)\right] \\
&= \mathrm{E}_a\left[\frac{(a-\theta_\mu)^3}{\theta_\sigma^5}\right] + \mathrm{E}_a\left[\frac{(a-\theta_\mu)}{\theta_\sigma^2}\right] \\
&= \mathrm{E}_a\left[\frac{a^3-3a^2\theta_\mu+3a\theta_\mu^2-\theta_\mu^3}{\theta_\sigma^5}\right] + 0 \\
&= \frac{\theta_\mu^3+3\theta_\mu\theta_\sigma^2-3(\theta_\mu^2+\theta_\sigma^2)\theta_\mu+3\theta_\mu^3-\theta_\mu^3}{\theta_\sigma^5} \\
&= 0 \\
\mathrm{E}_a[g_{22}] &= \mathrm{E}_a\left[\left(\frac{(a-\theta_\mu)^2}{\theta_\sigma^3}-1\right)^2\right] \\
&= \mathrm{E}_a\left[\left(\frac{(a-\theta_\mu)^2}{\theta_\sigma^3}\right)^2 - 2\left(\frac{(a-\theta_\mu)^2}{\theta_\sigma^3}\right)+1\right]
\end{aligned}
$$

Evaluating first term,

$$
\begin{aligned}
\mathrm{E}_a\left[\left(\frac{(a-\theta_\mu)^2}{\theta_\sigma^3}\right)^2\right] &= \frac{\mathrm{E}_a[a^4-4a^3\theta_\mu+6a^2\theta_\mu^2-4a\theta_\mu^3+\theta_\mu^4]}{\theta_\sigma^6} \\
&= \frac{\theta_\mu^4+6\theta_\mu^2\theta_\sigma^2+3\theta_\sigma^4-4(\theta_\mu^3+3\theta_\mu\theta_\sigma^2)\theta_\mu+6(\theta_\mu^2+\theta_\sigma^2)\theta_\mu^2-4\theta_\mu^4+\theta_\mu^4}{\theta_\sigma^6} \\
&= \frac{3}{\theta_\sigma^2}
\end{aligned}
$$

Evaluating the remaining terms,

$$
\begin{aligned}
-2\mathrm{E}_a\left[\frac{(a-\theta_\mu)^2}{\theta_\sigma^3}+1\right] &= -2\frac{\mathrm{E}_a\left[a^2-2a\theta_\mu+\theta_\mu^2\right]}{\theta_\sigma^3}+1 \\
&= -2\frac{\mathrm{E}_a\left[\theta_\mu^2+\theta_\sigma^2-2\theta_\mu^2+\theta^2\right]}{\theta_\sigma^3}+1 \\
&= \frac{-2}{\theta_\sigma}+1
\end{aligned}
$$

$$
\mathrm{F} = \begin{bmatrix} \frac{1}{\theta_\sigma^2} & 0 \\ 0 & \frac{3}{\theta_\sigma^2}-\frac{2}{\theta_\sigma}+1 \end{bmatrix}
$$

6. The inverse of Fisher Information Matrix(FIM), F^{-1} for parameterization $\sigma = \exp(\theta_\sigma)$ is $\begin{bmatrix} \exp(2\theta_\sigma) & 0 \\ 0 & \frac{1}{2} \end{bmatrix}$ where as for parameterization as $\sigma = \theta_\sigma$ is, $\mathrm{F}^{-1} = \begin{bmatrix} \theta_\sigma^2 & 0 \\ 0 & \frac{\theta_\sigma^2}{3-2\theta_\sigma+\theta_\sigma^2} \end{bmatrix}$. In the first case, the gradient of θ_σ is multiplied by a constant that is FIM has no effect other than changing the gradients by constant factor. In the other parameterization, the gradients is multiplied by $\frac{\theta_\sigma^2}{3-2\theta_\sigma+\theta_\sigma^2}$. If θ_σ is greater than $\frac{3}{2}$ than the gradients are scaled by value greater than 1, or less than 1 otherwise. Thus, the scaling of gradients depends on the parameter of θ_σ.

7.

Vanilla: $\sigma = \exp(\theta_\sigma)$

$$\nabla J = \begin{bmatrix} \frac{a-\theta_\mu}{\exp(2\theta_\sigma)} \\ \frac{(a-\theta_\mu)^2}{\exp(2\theta_\sigma)} - 1 \end{bmatrix} = \begin{bmatrix} \frac{8}{\exp(8)} \\ \frac{(8)^2}{\exp(8)} - 1 \end{bmatrix} = \begin{bmatrix} 2.68 \times 10^{-3} \\ -0.97 \end{bmatrix}$$

$$\begin{bmatrix} \theta_\mu \\ \theta_\sigma \end{bmatrix} = \begin{bmatrix} 0 \\ 4 \end{bmatrix} + 0.01 \begin{bmatrix} 2.68 \times 10^{-3} \\ -0.97 \end{bmatrix} = \begin{bmatrix} 2.68 \times 10^{-5} \\ 3.9903 \end{bmatrix}$$

Natural: $\sigma = \exp(\theta_\sigma)$

$$\mathrm{F}^{-1}\nabla J = \begin{bmatrix} \exp(2\theta_\sigma) & 0 \\ 0 & \frac{1}{2} \end{bmatrix} \begin{bmatrix} \frac{a-\theta_\mu}{\exp(2\theta_\sigma)} \\ \frac{(a-\theta_\mu)^2}{\exp(2\theta_\sigma)} - 1 \end{bmatrix} = \begin{bmatrix} a - \theta_\mu \\ \frac{1}{2}\left(\frac{(a-\theta_\mu)^2}{\exp(2\theta_\sigma)} - 1 \right) \end{bmatrix}$$

$$= \begin{bmatrix} 8 \\ -0.485 \end{bmatrix}$$

$$\begin{bmatrix} \theta_\mu \\ \theta_\sigma \end{bmatrix} = \begin{bmatrix} 0 \\ 4 \end{bmatrix} + 0.01 \begin{bmatrix} 8 \\ -0.485 \end{bmatrix} = \begin{bmatrix} 0.08 \\ 3.99515 \end{bmatrix}$$

Vanilla: $\sigma = \theta_\sigma$

$$\nabla J = \begin{bmatrix} \frac{a-\theta_\mu}{\theta_\sigma^2} \\ \frac{(a-\theta_\mu)^2}{2\theta_\sigma^3} - 1 \end{bmatrix} = \begin{bmatrix} \frac{8}{16} \\ \frac{8}{64} - 1 \end{bmatrix} = \begin{bmatrix} 0.5 \\ -0.875 \end{bmatrix}$$

$$\begin{bmatrix} \theta_\mu \\ \theta_\sigma \end{bmatrix} = \begin{bmatrix} 0 \\ 4 \end{bmatrix} + 0.01 \begin{bmatrix} 0.5 \\ -0.875 \end{bmatrix} = \begin{bmatrix} 0.005 \\ 3.99125 \end{bmatrix}$$

Natural: $\sigma = \theta_\sigma$

$$\mathrm{F}^{-1}\nabla J = \begin{bmatrix} \theta_\sigma^2 & 0 \\ 0 & \frac{\theta_\sigma^2}{3-2\theta_\sigma+\theta_\sigma^2} \end{bmatrix} \begin{bmatrix} \frac{a-\theta_\mu}{\theta_\sigma^2} \\ \frac{(a-\theta_\mu)^2}{2\theta_\sigma^3} - 1 \end{bmatrix} = \begin{bmatrix} a - \theta_\mu \\ \frac{(a-\theta_\mu)^2}{(3-2\theta_\sigma+\theta_\sigma^2)\theta_\sigma} - \frac{\theta_\sigma^2}{3-2\theta_\sigma+\theta_\sigma^2} \end{bmatrix}$$

$$= \begin{bmatrix} 8 \\ \frac{(8)^2}{(3-8+16)4} - \frac{16}{3-8+16} \end{bmatrix} = \begin{bmatrix} 8 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} \theta_\mu \\ \theta_\sigma \end{bmatrix} = \begin{bmatrix} 0 \\ 4 \end{bmatrix} + 0.01 \begin{bmatrix} 8 \\ 0 \end{bmatrix} = \begin{bmatrix} 0.08 \\ 4 \end{bmatrix}$$

In the first case of parameterization, the vanilla gradient update of σ is twice as of natural gradient. But, in the second case of parameterization the natural gradient update is zero as compared to the vanilla gradient.

References

[1] Marc Peter Deisenroth, Gerhard Neumann, and Jan Peters. A survey on policy search for robotics. *Found. Trends Robot*, 2(1&#8211;2):1–142, August 2013.