Homework 4

Version: 1.0
Version Release Date: 2021-03-25
Deadline: Thursday, April 8th, at 11:59pm.

Submission: You must submit your solutions as a PDF file through MarkUs[1]. You can produce the file however you like (e.g. LaTeX, Microsoft Word, scanner), as long as it is readable.

See the syllabus on the course website[2] for detailed policies. You may ask questions about the assignment on Piazza[3]. *Note that 10% of the homework mark (worth 1 pt) may be removed for a lack of neatness.*

The teaching assistants for this assignment are Keiran Paster and Irene Zhang.
`mailto:csc413-2021-01-tas@cs.toronto.edu`

Clarifications

All the modifications to the original assignment will be marked with red in the main body of the assignment. A complete list of the modifications can be found below:

-

[1] `https://markus.teach.cs.toronto.edu/csc413-2021-01/main`

[2] `https://csc413-uoft.github.io/2021/assets/misc/syllabus.pdf`

[3] `https://piazza.com/class/kjt32fc0f7y3kb`

1 RNNs and Self Attention

For any successful deep learning system, choosing the right network architecture is as important as choosing a good learning algorithm. In this question, we will explore how various architectural choices can have a significant impact on learning. We will analyze the learning performance from the perspective of vanishing /exploding gradients as they are backpropagated from the final layer to the first.

1.1 Warmup: A Single Neuron RNN

Consider an n layered fully connected network that has scalar inputs and outputs. For now, assume that all the hidden layers have a single unit, and that the weight matrices are set to 1 (because each hidden layer has a single unit, the weight matrices have a dimensionality of $\mathbb{R}^{1\times 1}$).

1.1.1 Effect of Activation - Sigmoid [0.25pt]

Lets say we're using the sigmoid activation. Let x be the input to the network and let $f : \mathbb{R}^1 \to \mathbb{R}^1$ be the function the network is computing. Do the gradients necessarily have to vanish or explode as they are backpropagated? Answer this by showing that $0 \le |\frac{\partial f(x)}{\partial x}| \le (\frac{1}{4})^n$, where n is the number of layers in the network.

Solution:

- The gradients necessarily have to vanish.
- The maximum gradient magnitude sigmoid can possibly take is $\frac{1}{4}$. The minimum is 0.
- Since the gradients are chained multiplicatively, we obtain the bound given in the question prompt.

1.1.2 Effect of Activation - Tanh [0.25pt]

Instead of sigmoid, now lets say we're using the tanh activation (otherwise the same setup as in 1.1.1). Do the gradients necessarily have to vanish or explode this time? Answer this by deriving a similar bound as in Sec 1.1.1 for the magnitude of the gradient.

Solution:

- The gradients definitely cannot explode. The gradient does not necessarily have to vanish either. When the inputs to tanh units are concentrated near 0, the gradient is preserved. Otherwise, the gradient vanishes.
- We can derive $0 \le |tanh'(x)| \le 1$ for a single tanh operation.
- Since the gradients are chained multiplicatively, we obtain $0 \le |\frac{\partial f(x)}{\partial x}| \le (1)^n = 1$ as the requested bound.

1.2 Matrices and RNN

We will now analyze the recurrent weight matrices under Singular Value Decomposition. SVD is one of the most important results in all of linear algebra. It says that any real matrix $M \in \mathbb{R}^{mxn}$ can be written as $M = U\Sigma V^T$ where $U \in \mathbb{R}^{mxm}$ and $V \in \mathbb{R}^{nxn}$ are square orthogonal matrices, and $\Sigma \in \mathbb{R}^{mxn}$ is a rectangular diagonal matrix with nonnegative entries on the diagonal (i.e. $\Sigma_{ii} \geq 0$ for $i \in \{1, \ldots, \min(m, n)\}$ and 0 otherwise). Geometrically, this means any linear transformation can be decomposed into a rotation/flip, followed by scaling along orthogonal directions, followed by another rotation/flip.

1.2.1 Gradient through RNN [0.5pt]

Let say we have a very simple RNN-like architecture that computes $x_{t+1} = tanh(Wx_t)$. You can view this architecture as a deep fully connected network that uses the same weight matrix at each layer. Suppose the largest singular value of the weight matrix is $\sigma_{max}(W) = \frac{1}{2}$. Show that the largest singular value of the input-output Jacobian has the following bound: $0 \leq \sigma_{max}(\frac{\partial x_n}{\partial x_1}) \leq (\frac{1}{2})^n$. (Hint: if $C = AB$, then $\sigma_{max}(C) \leq \sigma_{max}(A)\sigma_{max}(B)$. Also, the input-output Jacobian is the multiplication of layerwise Jacobians).

Solution:

We know that

$$\frac{\partial x_n}{\partial x_1} = \frac{\partial x_n}{\partial x_{n-1}} \frac{\partial x_{n-1}}{\partial x_{n-2}} ... \frac{\partial x_2}{\partial x_1}$$

We also know that

$$\begin{aligned}\frac{\partial x_n}{\partial x_{n-1}} &= \frac{\partial \tanh W x_{n-1}}{\partial x_{n-1}} \\ &= \frac{\partial \tanh W x_{n-1}}{\partial W x_{n-1}} \frac{\partial W x_{n-1}}{\partial x_{n-1}}\end{aligned}$$

We know that the derivative of tanh is bounded by 1. This is equivalent to saying the most scaling its Jacobian does is 1. Therefore, the first term will have singular values bounded by 1 (spectral norm is equal to the maximum singular value). The second term by assumption has bounded singular values of $\frac{1}{2}$.

Using these two facts, we conclude using the hint the required bound.

Marking scheme:

1. 0.5 point for applying the chain rule correctly and understanding the weight matrix singular value is repeated every time step
2. 0.5 point for putting everything together (using the bound, and using the derivative bounding on element wise tanh)
3. Deductions for handwaving too many steps or not being accurate or clear. For instance, the optimal solution will show the Jacobian of element-wise tanh, and then make an argument about the SVD of that diagonal matrix.

1.3 Self-Attention

In a self-attention layer (using scaled dot-product attention), the matrix of outputs is computed as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V$$

where $Q, K, V \in \mathbb{R}^{n\times d}$ are the query, key, and value matrices, n is the sequence length, and d_m is the embedding dimension.

1.3.1 Complexity of Self-Attention [0.25pt]

Show that the complexity of a self-attention layer at test-time scales quadratically with the sequence length n.

Solution:

$Q, K \in \mathbb{R}^{n\times d_m}$ and multiplying $QK^\top$ results in an $n \times n$ matrix and takes $\mathcal{O}\big(n^2 d\big)$ time.

1.3.2 Linear Attention with SVD [1pt]

It has been empirically shown in Transformer models that the context mapping matrix $P = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)$ often has a low rank. Show that if the rank of P is k and we already have access to the SVD of P, then it is possible to compute self-attention in $\mathcal{O}(nkd)$ time.

Solution:

We assume we have access to the SVD of $P = \sum_{i=1}^{k} \sigma_i u_i v_i^\top = \begin{bmatrix} u_i & \dots & u_k \end{bmatrix} \text{diag}\{\sigma_1, \dots, \sigma_k\} \begin{bmatrix} v_1 \\ \vdots \\ v_k \end{bmatrix}$.

Computing $\begin{bmatrix} v_1 \\ \vdots \\ v_k \end{bmatrix} V$ involves multiplying a $k \times n$ matrix with a $n \times d$ matrix, which takes $\mathcal{O}(knd)$ time and results in a $k \times d$ matrix. Likewise, computing $\begin{bmatrix} u_i & \dots & u_k \end{bmatrix} \text{diag}\{\sigma_1, \dots, \sigma_k\}$ involves multiplying k n-dimensional vectors by the k singular values, which takes $\mathcal{O}(nk)$ time. Finally, multiplying the $n \times k$ matrix by the $k \times d$ matrix takes $\mathcal{O}(nkd)$ time. Therefore, the entire computation can be done in $\mathcal{O}(nkd)$ time.

1.3.3 Linear Attention by Projecting [1pt]

Suppose we ignore the Softmax and scaling and let $P = QK^\top \in \mathbb{R}^{n\times n}$. Assume P is rank k. Show that there exist two linear projection matrices $C, D \in \mathbb{R}^{k\times n}$ such that $PV = Q(CK)^\top DV$ and the right hand side can be computed in $\mathcal{O}(nkd)$ time. *Hint: Consider using SVD in your proof.*

Solution:

Since P is rank k, its SVD can be written as $P = \begin{bmatrix} u_i & \dots & u_k \end{bmatrix} \text{diag}\{\sigma_1, \dots, \sigma_k\} \begin{bmatrix} v_1 \\ \vdots \\ v_k \end{bmatrix} = \tilde{U}\Sigma\tilde{V}^\top$, with $\tilde{U}, \tilde{V} \in \mathbb{R}^{n\times k}$. Let $C = D = \tilde{V}^\top$. Then,

$$\begin{aligned} Q(CK)^\top DV &= QK^\top C^\top DV \\ &= \tilde{U}\Sigma\tilde{V}^\top C^\top DV \\ &= \tilde{U}\Sigma\tilde{V}^\top \tilde{V}\tilde{V}^\top V \\ &= \tilde{U}\Sigma\tilde{V}^\top V \\ &= PV \end{aligned}$$

For the complexity, CK and DV take $\mathcal{O}(nkd)$ time, $Q(CK)^\top$ takes $\mathcal{O}(nkd)$ time, and finally multiplying $[Q(CK)^\top][DV]$ takes $\mathcal{O}(nkd)$ time. Therefore, the entire quantity can be computed in $\mathcal{O}(nkd)$ time.

2 Reversible Models and Variational AutoEncoders

In lecture, you saw how the change-of-variables formula can be used to get the likelihood of a datapoint in a reversible model. You also saw how Variational AutoEncoders can be trained by optimizing a lower bound on this likelihood.

In this problem, you will show one relationship between these two approaches. Specifically, you will show how for a class of distributions for the encoder and decoder of a VAE that correspond to deterministic random variables, you can derive the change-of-variables formula. This connection makes it clear how to calculate a bound on the likelihood of a composition both stochastic and bijective transformations. This question is inspired by SurVAE Flows [Nielsen et al., 2020], which uses this connection to derive how to optimize models with surjective transformations.

2.1 Variational Lower Bound [1pt]

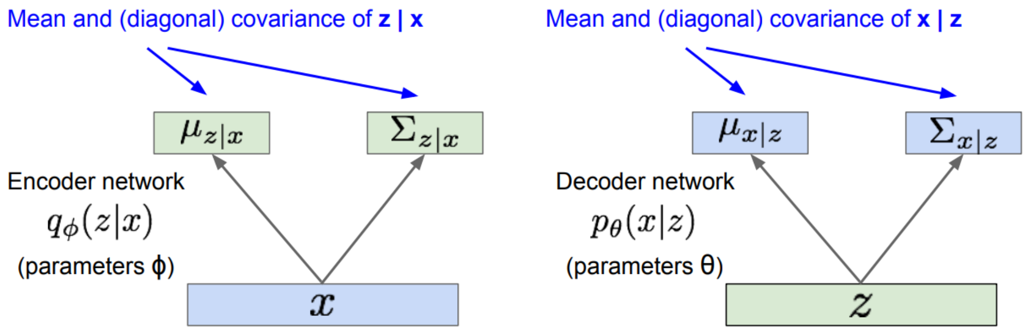

Figure 1: Figure from http://cs231n.stanford.edu/slides/2019/cs231n_2019_lecture11.pdf

You saw in class that the log probability of data x can be lower bounded by the variational lower bound using the latent prior $p(z)$, decoder $p(x|z)$ and encoder $q(z|x)$. In this problem, you will derive a different way of writing this bound. Show that $\log p(x)$ can be rewritten as:

$$\log p(x) = \underbrace{\mathbb{E}_{q(z|x)}[\log p(z)] + \mathbb{E}_{q(z|x)}\left[\log \frac{p(x|z)}{q(z|x)}\right]}_{\text{variational lower bound}} + \underbrace{\mathbb{E}_{q(z|x)}\left[\log \frac{q(z|x)}{p(z|x)}\right]}_{\text{gap in lower bound}} \tag{2.1}$$

Solution:

$$\begin{aligned}
\log p(x) &= \mathbb{E}_{q(z|x)}[\log p(x)] \\
&= \mathbb{E}_{q(z|x)}\left[\log\left[\frac{p(x,z)}{p(z|x)}\right]\right] \\
&= \mathbb{E}_{q(z|x)}\left[\log\left[\frac{p(x,z)}{q(z|x)}\frac{q(z|x)}{p(z|x)}\right]\right] \\
&= \mathbb{E}_{q(z|x)}\left[\log\frac{p(x,z)}{q(z|x)}\right] + \mathbb{E}_{q(z|x)}\log\left[\frac{q(z|x)}{p(z|x)}\right] \\
&= \mathbb{E}_{q(z|x)}[\log p(z)] + \mathbb{E}_{q(z|x)}\left[\log\frac{p(x|z)}{q(z|x)}\right] + \mathbb{E}_{q(z|x)}\left[\log\frac{q(z|x)}{p(z|x)}\right]
\end{aligned}$$

2.2 Connecting Reversible Models with VAEs [1pt]

Suppose we have an invertible, differentiable function $f : \mathcal{Z} \to \mathcal{X}$ which maps samples from some prior distribution $p(z)$ to data space. Let $z = f^{-1}(x)$. To compute the log probability of a datapoint x, one should use the change-of-variables formula:

$$\log p(x) = \log p(z) + \log\left|\det\frac{\partial f(z)}{\partial z}\right|^{-1} \tag{2.2}$$

In this problem, you will show that the change of variables formula can be derived as a special case of Equation 2.1. In order to write f with probability distributions, we will use Dirac δ-functions. A Dirac delta is defined as followed:

$$\delta(x) = \begin{cases} +\infty, & x = 0 \\ 0, & x \neq 0 \end{cases}$$

with the additional property that it integrates to 1:

$$\int_{-\infty}^{\infty} \delta(x)dx = 1$$

Suppose we define deterministic conditionals for our function f:

$$p(x|z) = \delta(x - f(z))$$
$$p(z|x) = \delta(z - f^{-1}(x))$$

Show that when using the above distributions and when $q(z|x) = p(z|x)$, Equation 2.1 can be rewritten as Equation 2.2.

Hint: use the following property to write one conditional in terms of the other:

Let z_0 be a root of the function $g : \mathcal{Z} \to \mathcal{X}$. The composition of a Dirac δ-function with a smooth function g is defined as:

$$\delta(g(z)) = \left|\det\frac{\partial g(z)}{\partial z}\right|_{z=z_0}^{-1} \delta(z - z_0) \tag{2.3}$$

Solution:

Let $g(z) = x - f(z)$ and note that $z_0 = f^{-1}(x)$. We can use Equation 2.3 to show:

$$\begin{aligned}
p(x|z) &= \delta(x - f(z)) \\
&= \delta(z - f^{-1}(x))\left|\det \frac{\partial f(z)}{\partial z}\right|^{-1}_{z=f^{-1}(x)} \\
&= p(z|x)\left|\det \frac{\partial f(z)}{\partial z}\right|^{-1}_{z=f^{-1}(x)}
\end{aligned}$$

We can then substitute into Equation 2.1:

$$\begin{aligned}
\log p(x) &= \mathbb{E}_{q(z|x)}[\log p(z)] + \mathbb{E}_{q(z|x)}\left[\log \frac{p(x|z)}{q(z|x)}\right] + \mathbb{E}_{q(z|x)}\left[\log \frac{q(z|x)}{p(z|x)}\right] \\
&= \log p(z) + \log\left|\det \frac{\partial f(z)}{\partial z}\right|^{-1}_{z=f^{-1}(x)}
\end{aligned}$$

2.3 Computing the Likelihood of a Composition of Transformations [1pt]

Consider the problem of computing a lower-bound on the likelihood of a datapoint x when using some transformation f that maps latent samples $z \sim p(z)$ some $x \in \mathcal{X}$. When f is smooth and bijective, we can use Equation 2.2, the change of variables formula, to get the exact log-likelihood of any x. When f is a stochastic transformation with $p(x|z)$ and $q(z|x)$, we can use Equation 2.1, the variational lower bound, to get a bound on the likelihood.

Suppose the function $f \coloneqq f_T \circ f_{T-1} \circ \ldots \circ f_1$ is a composition of several transformations which are either stochastic (VAE) or bijective (reversible models).

We can write a bound on the likelihood as:

$$\log p(x) \geq \log p(z) + \sum_{t=1}^{T} \mathcal{V}_t$$

How would we find each of the $\mathcal{V}_t$? *Hint: these terms may be different depending on the type of transformations.*

Solution:

If f_t is bijective:

$$z = f_t^{-1}(x)$$

$$\mathcal{V}_t = \log\left|\det \frac{\partial f(z)}{\partial z}\right|^{-1}$$

If f_t is stochastic:

$$z \sim q_t(z|x)$$

$$\mathcal{V}_t = \log \frac{p_t(x|z)}{q_t(z|x)}$$

3 Reinforcement Learning

3.1 Bellman Equation

The Bellman equation can be seen as a fix point equation to what's called the Bellman Operator. Given a policy π, the Bellman operators T^π for $V : \mathcal{B}(\mathcal{S}) \to \mathcal{B}(\mathcal{S})$ and T^π for $Q : \mathcal{B}(\mathcal{S} \times \mathcal{A}) \to \mathcal{B}(\mathcal{S} \times \mathcal{A})$ are defined as follows:

$$(T^\pi V)(s) \triangleq r^\pi(S) + \gamma \int \mathcal{P}(s' \mid s, a)\pi(a \mid s)V(s')$$
$$(T^\pi Q)(s,a) \triangleq r(s,a) + \gamma \int \mathcal{P}(s' \mid s, a)\pi(a' \mid s')Q(s', a')$$

for all $s \in \mathcal{S}($ for $V)$ or all $(s,a) \in \mathcal{S} \times \mathcal{A}($ for $Q)$.

The Bellman operators have two important properties, 1) monotononicity and 2) γ-contraction. These properties give us many guarantees such as applying the operator repeatedly will converge to an unique and optimal solution, which is what allow us to show RL algorithms such as Q-Learning converges (under certain additional assumptions, but we won't go over them here). In this section, we will show that the Bellman operator indeed have these two properties.

a) [0.5pt] Show that the Bellman operator (on V) has the monotonicity property. i.e., show that for a fixed policy π, if $V_1, V_2 \in \mathcal{B}(\mathcal{S})$, and $V_1 \le V_2$, then we have

$$T^\pi V_1 \le T^\pi V_2$$

Solution:

$$\begin{aligned}(T^\pi V_1)(x) &= r^\pi(x) + \gamma \int \mathcal{P}^\pi(\mathrm{d}x' \mid x)\underbrace{V_1(x')}_{\le V_2(x')} \\ &\le r^\pi(x) + \gamma \int \mathcal{P}^\pi(\mathrm{d}x' \mid x)V_2(x') = (T^\pi V_2)(x)\end{aligned}$$

b) [1pt] Show that the Bellman operator is a γ-contraction mapping with the supremum norm (on Q). i.e. show that for a discount factor γ and $Q_1, Q_2 \in \mathcal{B}(\mathcal{S} \times \mathcal{A})$, we have

$$||T^\pi(Q_1) - T^\pi(Q_2)||_\infty \le \gamma||Q_1 - Q_2||_\infty$$

Recall from your math classes, the supremum norm (on Q) is as follows:

$$\|Q\|_\infty = \sup_{(s,a)\in\mathcal{S}\times\mathcal{A}} |Q(s,a)| \tag{3.1}$$

Hint: for some function f, we have the following. For this question, you can think about what is P and f in our case.

$$\begin{aligned}\left|\int P(x)f(x)\right| &\le \int |P(x)f(x)| = \int |P(x)| \cdot |f(x)| \\ &\le \int P(x) \cdot \sup_{x\in\mathcal{X}} |f(x)| \\ &= \sup_{x\in\mathcal{X}} |f(x)| \int P(x) = \|f\|_\infty\end{aligned} \tag{3.2}$$

where in the last line we used the fact $\int P(x) = 1$

Solution:

For any $(x,a) \in \mathcal{X} \times \mathcal{A}$, we have

$$\begin{aligned}
|(T^\pi Q_1)(x,a) - (T^\pi Q_2)(x,a)| &= \left|\left[r(x,a) + \gamma \int \mathcal{P}(\mathrm{d}x' \mid x,a)\pi(\mathrm{d}a' \mid x')Q_1(x',a')\right] - \right. \\
&\left[r(x,a) + \gamma \int \mathcal{P}(\mathrm{d}x' \mid x,a)\pi(\mathrm{d}a' \mid x')Q_2(x',a')\right] \\
&= \gamma\left|\int \mathcal{P}(\mathrm{d}x' \mid x,a)\pi(\mathrm{d}a' \mid x')\big(Q_1(x',a') - Q_2(x',a')\big)\right|
\end{aligned}$$

$$\begin{aligned}
\left|\int P(\,\mathrm{d}x)f(x)\right| &\le \int |P(\,\mathrm{d}x)f(x)| = \int |P(\,\mathrm{d}x)| \cdot |f(x)| \\
&\le \int P(\,\mathrm{d}x) \cdot \sup_{x\in\mathcal{X}} |f(x)| \\
&= \sup_{x\in\mathcal{X}} |f(x)| \int P(\,\mathrm{d}x) = \|f\|_\infty |(T^\pi Q_1)(x,a) - (T^\pi Q_2)(x,a)|
\end{aligned}$$

$$\begin{aligned}
\gamma\left|\int \mathcal{P}(\mathrm{d}x' \mid x,a)\pi(\mathrm{d}a' \mid x')\big(Q_1(x',a') - Q_2(x',a')\big)\right| \\
&\le \gamma \int \mathcal{P}(\mathrm{d}x' \mid x,a)\pi(\mathrm{d}a' \mid x')\big|Q_1(x',a') - Q_2(x',a')\big| \\
&\le \gamma\|Q_1 - Q_2\|_\infty \int \mathcal{P}(\mathrm{d}x' \mid x,a)\pi(\mathrm{d}a' \mid x') \\
&= \gamma\|Q_1 - Q_2\|_\infty
\end{aligned}$$

c) **[0.25pt]** For this question, you may assume knowledge of the reward function $r(s,a)$ and transition probability function $p(s'|s,a)$.

c.1) Give a definition of $v_*(s)$ in terms of $q_*(s,a)$.

Solution:

$v_*(s) = \max_a(q_*(s,a))$

c.2) Give a definition of $q_*(s,a)$ in terms of $v_*(s)$.

Solution:

$q_*(s,a) = \sum_{s',r} p(s',r \mid s,a)[r + \gamma v_*(s')]$

or

$q_*(s,a) = r(s,a) + \gamma \sum_{s'} p(s' \mid s,a) v_*(s')$

c.3) Give a definition of a_* in terms of $q_*(s,a)$.

Solution:

$a_* \doteq \arg \pi_*(a_* \mid s) = \arg\max_a q_*(s,a)$

c.4) Give a definition of a_* in terms of $v_*(s)$.

Solution:

$a_* \doteq \arg \pi_*(a_* \mid s) = \arg\max_a \sum_{s',r} p(s', r \mid s, a)[r + \gamma v_*(s')]$

or

$a_* = \arg\max_a r(s,a) + \gamma \sum_{s'} p(s' \mid s, a) v_*(s')$

3.2 the REINFORCE Gradient Estimator

3.2.1 Choice of Baseline [1pt]

In class/tutorial we saw that the reinforce gradient can be written as follows using the log derivative trick:

$$\nabla_{\boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta}) = \mathbb{E}_{\substack{s_1 \sim \rho_0(s) \\ a_t \sim \pi_\theta(a_t|s_t) \\ s_{t+1} \sim p(s_{t+1}|s_t,a_t)}} \left[\sum_{t=1}^{T} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_t \mid s_t) \left[\sum_{t'=1}^{T} R(s_{t'}, a_{t'}) \right] \right] \tag{3.3}$$

In tutorial, we also saw that subtracting a baseline $b \in \mathbb{R}$ as follows is a variance reduction technique:

$$\nabla_{\boldsymbol{\theta}} \mathcal{J}_b(\boldsymbol{\theta}) = \mathbb{E}_{\substack{s_1 \sim \rho_0(s) \\ a_t \sim \pi_\theta(a_t|s_t) \\ s_{t+1} \sim p(s_{t+1}|s_t,a_t)}} \left[\sum_{t=1}^{T} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_t \mid s_t) \left[(\sum_{t'=1}^{T} R(s_{t'}, a_{t'}) - b) \right] \right] \tag{3.4}$$

In practice, to estimate the gradient, we can execute the policy in the environment N times, and the average of N runs we obtain for both (3.3) and (3.4) are unbiased estimators. For this question, derive an optimal baseline $b \in \mathbb{R}^M$ in terms of π_θ and R that would minimize the variance of this estimator, where M is the size of the gradient.

Hint: To do this, we can follow three steps, for each k-th element in the gradient term:

1) Write down the variance term $\mathrm{Var}[\sum_{t=1}^{T} \nabla_{\boldsymbol{\theta_k}} \log \pi_{\boldsymbol{\theta}} \left(a_t^{(i)} \mid s_t^{(i)} \right) \left[\left(\sum_{t'=t}^{T} R\left(s_{t'}^{(i)}, a_{t'}^{(i)} \right) \right) - b_k \right]]$. Recall from your probability class that for a random variable x, we have $\mathrm{Var}[x] = E\left[x^2\right] - E[x]^2$

2) To minimize the variance, find $\frac{d}{db_k}$ Var, set this to zero, and solve for b_k.

3) Finally, to show that what we find is a minima, we show that the second derivative is positive.

Solution:

$$\nabla_\theta J(\theta) = E_{\tau \sim \pi_\theta(\tau)}[\nabla_\theta \log \pi_\theta(\tau)(r(\tau) - b)]$$
$$\mathrm{Var}[x] = E\left[x^2\right] - E[x]^2$$
$$\mathrm{Var} = E_{\tau \sim \pi_\theta(\tau)}\left[(\nabla_{\theta_k} \log \pi_\theta(\tau)(r(\tau) - b_k))^2 \right] - E_{\tau \sim \pi_\theta(\tau)}[\nabla_{\theta_k} \log \pi_\theta(\tau)(r(\tau) - b_k)]^2$$

Note that when we take the derivative of the variance, we can ignore the second expectation term, because as seen in tutorial, the estimator with baseline is unbiased, hence does not depend on b. let $g(\tau) = \nabla_{\theta_k} \log \pi_\theta(\tau)$

$$\begin{aligned}
\frac{d\,\mathrm{Var}}{db_k} &= \frac{d}{db_k}E\left[g(\tau)^2(r(\tau)-b_k)^2\right] = \frac{d}{db_k}\left(E\left[g(\tau)^2r(\tau)^2\right] - 2E\left[g(\tau)^2r(\tau)b_k\right] + b_k^2E\left[g(\tau)^2\right]\right) \\
&= -2E\left[g(\tau)^2r(\tau)\right] + 2b_kE\left[g(\tau)^2\right] = 0 \\
b_k &= \frac{E\left[g(\tau)^2r(\tau)\right]}{E[g(\tau)^2]} \\
&= \frac{E\left[\left(\sum_{t=1}^{T}\nabla_{\theta_k}\log\pi_\theta(a_t \mid s_t)\right)^2\sum_{t'=1}^{T}r(s_{t'},a_{t'})\right]}{E\left[\left(\sum_{t=1}^{T}\nabla_{\theta_k}\log\pi_\theta(a_t|s_t)\right)^2\right]}
\end{aligned}$$

2005/06/28*ver*

3.2.2 Importance Sampling [1pt]

For a general function f and a probability measure P, computing the following will be hard if we do not have access to samples from P, or if P have very low probability where f takes its largest values:

$$\mathrm{E}_{P(X)}[f(X)] = \int_x P(x)f(x)dx \approx \frac{1}{m}\sum_{i=1}^{m}P\left(x^{(i)}\right)f\left(x^{(i)}\right) \text{ with } x^{(i)} \sim P$$

Importance sampling provides an alternative solution, where instead of sampling from P, we sample from some distribution Q of our choice. Given samples from Q, we can then estimate the expectation w.r.t. P as follows:

$$\begin{aligned}
\mathrm{E}_{P(X)}[f(X)] &= \mathrm{E}_{Q(X)}\left[\frac{P(X)}{Q(X)}f(X)\right] \\
&\approx \frac{1}{m}\sum_{i=1}^{m}\frac{P\left(x^{(i)}\right)}{Q\left(x^{(i)}\right)}f\left(x^{(i)}\right) \text{ with } x^{(i)} \sim Q
\end{aligned}$$

When we derive the REINFORCE estimator in the lecture/tutorial, we are assuming getting access to samples from π_θ. Suppose we no long have this assumption, and we only have samples from another distribution, $\pi_{\theta'}$ instead. Derive the REINFORCE estimator for policy π_θ under this new setting with importance sampling, i.e. Derive $\nabla_\theta J(\theta)$ with $a_t \sim \pi_{\theta'(a|s_t)}$. Your solution should be only in terms of $\pi_\theta, \pi_{\theta'}$, and r. Show your steps explicitly. What happens if the two policies π_θ and $\pi_{\theta'}$are close to each other?

Hint1: you can use the following property: for some function f, $\nabla_\theta\mathbb{E}[f(\theta)] = \mathbb{E}[\nabla_\theta f(\theta)]$

Hint2: Use the importance sampling equation given, where P should be the probability of trajectory sampling actions from π_θ, and Q should be the probability of trajectory sampling actions from $\pi_{\theta'}$.

Solution:

$$\begin{aligned}
P(\tau;\theta) &= p(\mathbf{s}_1)\prod_{t=1}^{T}\pi_\theta(\mathbf{a}_t \mid \mathbf{s}_t)p(\mathbf{s}_{t+1} \mid \mathbf{s}_t,\mathbf{a}_t) \\
\frac{P(\tau;\theta)}{P(\tau;\theta')} &= \frac{p(s_1)\prod_{t=1}^{T}\pi_\theta(\mathbf{a}_t \mid \mathbf{s}_t)p(\mathbf{s}_{t+1}|\mathbf{s}_t,\mathbf{a}_t)}{p(s_1)\prod_{t=1}^{T}\pi_{\theta'}(\mathbf{a}_t|\mathbf{s}_t)p(\mathbf{s}_{t+1}|\mathbf{s}_t,\mathbf{a}_t)} = \frac{\prod_{t=1}^{T}\pi_\theta(\mathbf{a}_t \mid \mathbf{s}_t)}{\prod_{t=1}^{T}\pi_{\theta'}(\mathbf{a}_t \mid \mathbf{s}_t)} \\
\nabla_{\boldsymbol{\theta}}\mathcal{J}(\boldsymbol{\theta}) &= \nabla_{\boldsymbol{\theta}} \mathop{\mathbb{E}}_{\boldsymbol{\tau}\sim p(\boldsymbol{\tau};\boldsymbol{\theta}')}\left[\frac{p(\tau;\theta)}{p(\tau;\theta')}R(\tau)\right] \\
\nabla_\theta J(\theta) &= E_{\tau\sim\pi_{\theta'}(\tau)}\left[\frac{\nabla_\theta\pi_\theta(\tau)}{\pi_{\theta'}(\tau)}r(\tau)\right] \\
&= E_{\tau\sim\pi_{\theta'}(\tau)}\left[\sum_{t=1}^{T}\nabla_\theta\log\pi_\theta(\mathbf{a}_t \mid \mathbf{s}_t)\left(\prod_{t'=1}^{t}\frac{\pi_\theta(\mathbf{a}_{t'} \mid \mathbf{s}_{t'})}{\pi_{\theta'}(\mathbf{a}_{t'} \mid \mathbf{s}_{t'})}\right)\left(\sum_{t'=t}^{T}r(\mathbf{s}_{t'},\mathbf{a}_{t'})\right)\right]
\end{aligned}$$

If the polices are close together, we get the original reinforce estimator again.

References

Didrik Nielsen, Priyank Jaini, Emiel Hoogeboom, Ole Winther, and Max Welling. SurVAE flows: Surjections to bridge the gap between vaes and flows. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL `https://proceedings.neurips.cc/paper/2020/hash/9578a63fbe545bd82cc5bbe749636af1-Abstract.html`.