Due Date: March 23rd (11pm), 2022

Instructions

- *For all questions, show your work!*
- *Starred questions are* ***hard*** *questions, not* ***bonus*** *questions.*
- *Use LaTeX and the template we provide when writing your answers. You may reuse most of the notation shorthands, equations and/or tables. See the assignment policy on the course website for more details.*
- *Unless noted that questions are related, assume that notation and defintions for each question are self-contained and independent.*
- *Submit your answers electronically via Gradescope.*
- *TAs for this assignment are* ***Ankit Vani*** *and* ***Sai Aravind Sreeramadas****.*

Question 1 (6-9-6)**.** This question is about activation functions and vanishing/exploding gradients in recurrent neural networks (RNNs). Let $\sigma : \mathbb{R} \to \mathbb{R}$ be an activation function. When the argument is a vector, we apply σ element-wise. Consider the following recurrent unit:

$$\boldsymbol{h}_t = \boldsymbol{W}\sigma(\boldsymbol{h}_{t-1}) + \boldsymbol{U}\boldsymbol{x}_t + \boldsymbol{b}$$

1.1 Show that applying the activation function in this way results in an equivalent recurrence as the conventional way of applying the activation function: $\boldsymbol{g}_t = \sigma(\boldsymbol{W}\boldsymbol{g}_{t-1} + \boldsymbol{U}\boldsymbol{x}_t + \boldsymbol{b})$ (i.e. express $\boldsymbol{g}_t$ in terms of $\boldsymbol{h}_t$). More formally, you need to prove it using mathematical induction. You only need to prove the induction step in this question, assuming your expression holds for time step $t-1$.

answer:

if we assume that the expression holds for time $t-1$

$$h_{t-1} = W\sigma\left(h_{t-2}\right) + Ux_{t-1} + b$$

also h_t is like:

$$h_t = W\sigma\left(h_{t-1}\right) + Ux_t + b$$

we can replace h_{t-1} in above equation :

$$h_t = W\sigma\left(W\sigma\left(h_{t-2}\right) + Ux_{t-1} + b\right) + Ux_t + b$$

We can suppose that $\sigma(h_{t-2}) = g_{t-2}$:

$$h_t = W\sigma\left(W\sigma g_{t-2} + Ux_{t-1} + b\right) + Ux_t + b$$

We know that $g_{t-1} = \sigma\left(Wg_{t-2} + Ux_{t-1} + b\right)$ So we can write :

$$h_t = Wg_{t-1} + Ux_t + b$$

If we pass sigmoid to both side:

$$\sigma(h_t) = \sigma(Wg_{t-1} + Ux_t + b)$$

and we will get:

$$g_t = \sigma(h_t)$$

*1.2 Let $||\boldsymbol{A}||$ denote the L_2 operator norm [1] of matrix $\boldsymbol{A}$ ($||\boldsymbol{A}|| := \max_{\boldsymbol{x}:||\boldsymbol{x}||=1} ||\boldsymbol{A}\boldsymbol{x}||$). Assume $\sigma(x)$ has bounded derivative, i.e. $|\sigma'(x)| \leq \gamma$ for some $\gamma > 0$ and for all x. We denote as $\lambda_1(\cdot)$ the largest eigenvalue of a symmetric matrix. Show that if the largest eigenvalue of the weights is bounded by $\frac{\delta^2}{\gamma^2}$ for some $0 \leq \delta < 1$, gradients of the hidden state will vanish over time, i.e.

$$\lambda_1(\boldsymbol{W}^\top \boldsymbol{W}) \leq \frac{\delta^2}{\gamma^2} \quad \Longrightarrow \quad \left|\left|\frac{\partial \boldsymbol{h}_T}{\partial \boldsymbol{h}_0}\right|\right| \to 0 \text{ as } T \to \infty$$

Use the following properties of the L_2 operator norm

$$||\boldsymbol{A}\boldsymbol{B}|| \leq ||\boldsymbol{A}||\,||\boldsymbol{B}|| \qquad \text{and} \qquad ||\boldsymbol{A}|| = \sqrt{\lambda_1(\boldsymbol{A}^\top \boldsymbol{A})}$$

answer: From the previous part we have:

$$h_t = W\sigma(h_{t-1}) + Ux_t + b$$

For t in [1, ..., T] we can write:

$$\frac{\partial h_t}{\partial h_{t-1}} = \frac{\partial(W\sigma(h_{t-1} + Ux_t + b)}{\partial h_{t-1}} = W\sigma'(h_{t-1})$$

Also we can write this equation:

$$\frac{\partial h_T}{\partial h_0} = \frac{\partial h_T}{\partial h_{T-1}}\frac{\partial h_{T-1}}{\partial h_{T-2}}...\frac{\partial h_1}{\partial h_0}$$

with replacing the previous equation and take norm from the equation we can rewrite it in this form:

$$||\frac{\partial h_T}{\partial h_0}|| = ||W\sigma'(T-1)W\sigma'(T-2)...W\sigma'(h_0)||$$

Now by applying $||AB|| \leq ||A||||B||$ we will have:

$$||\frac{\partial h_T}{\partial h_0}|| \leq ||W||\,||\sigma'(T-1)||\,||W||\,||\sigma'(T-2)||\,...\,||W||\,||\sigma'(h_0)||$$

we know that $|\sigma'(x)| \leq \gamma$ so:

$$||\frac{\partial h_T}{\partial h_0}|| \leq ||W||^T\gamma^T = (||W||^2\gamma^2)^{\frac{T}{2}}$$

1. The L_2 operator norm of a matrix $\boldsymbol{A}$ is is an *induced norm* corresponding to the L_2 norm of vectors. You can try to prove the given properties as an exercise.

Also by applying $||A|| = \sqrt{\lambda_1(A^TA)}$ we can assume that $||W||^2 = \lambda_1(W^TW)$ also if we assume that $\lambda_1(W^TW) \leq \frac{\delta^2}{\gamma^2}$ we can prove that:

$$||\frac{\partial h_T}{\partial h_0}|| \leq (\frac{\delta^2}{\gamma^2}\gamma^2)^{\frac{T}{2}} = \delta^T$$

from the fact that $0 \leq \delta < 1$ we can prove that:

$$\lim_{T\to\infty} ||\frac{\partial h_T}{\partial h_0}|| \leq \delta^T = 0$$

1.3 What do you think will happen to the gradients of the hidden state if the condition in the previous question is reversed, i.e. if the largest eigenvalue of the weights is larger than $\frac{\delta^2}{\gamma^2}$? Is this condition *necessary* and/or *sufficient* for the gradient to explode ? (Answer in 1-2 sentences).

answer: For the gradient to explode we should have:

$$\lim_{T\to\infty} ||\frac{\partial h_T}{\partial h_0}|| = \infty$$

also we prove that $||\frac{\partial h_T}{\partial h_0}|| \leq ||W||^T\gamma^T$ So for the gradient to explode, the upper bound should tend to infinity over time, so if the largest eigenvalue of the weights is larger than $\frac{\delta^2}{\gamma^2}$ the upper bound will not be zero anymore. The gradient may tend to a finite number or infinity, so it is a necessary condition and not sufficient since there is still a possibility that the gradient will be limited even with this condition.

Question 2 (8-8-8)**.** In this question you will demonstrate that an estimate of the first moment of the gradient using an (exponential) running average is equivalent to using momentum, and is biased by a scaling factor. The goal of this question is for you to consider the relationship between different optimization schemes, and to practice noting and quantifying the effect (particularly in terms of bias/variance) of *estimating* a quantity.

Let $\boldsymbol{g}_t$ be an unbiased sample of gradient at time step t and $\Delta\boldsymbol{\theta}_t$ be the update to be made. Initialize $\boldsymbol{v}_0$ to be a vector of zeros.

2.1 For $t \geq 1$, consider the following update rules:

- SGD with momentum:

$$\boldsymbol{v}_t = \alpha\boldsymbol{v}_{t-1} + \epsilon\boldsymbol{g}_t \qquad \Delta\boldsymbol{\theta}_t = -\boldsymbol{v}_t$$

  where $\epsilon > 0$ and $\alpha \in (0,1)$.

- SGD with running average of $\boldsymbol{g}_t$:

$$\boldsymbol{v}_t = \beta\boldsymbol{v}_{t-1} + (1-\beta)\boldsymbol{g}_t \qquad \Delta\boldsymbol{\theta}_t = -\delta\boldsymbol{v}_t$$

  where $\beta \in (0,1)$ and $\delta > 0$.

Express the two update rules recursively ($\Delta\boldsymbol{\theta}_t$ as a function of $\Delta\boldsymbol{\theta}_{t-1}$). Show that these two update rules are equivalent ; i.e. express (α, ϵ) as a function of (β, δ).

answer: By replacing the $\Delta\theta_t$ in the momentum we will have:

$$\Delta\theta_t = \alpha\Delta\theta_{t-1} - \epsilon g_t$$

and by replacing the $\Delta\theta_t$ in running average, we arrive to:

$$\Delta\theta_{t-1} = \beta\Delta\theta_t - \delta(1-\beta)g_t$$

This two equation can be equivalent if $\alpha = \beta$ and $\epsilon = \delta(1-\beta)$

2.2 Unroll the running average update rule, i.e. express $\boldsymbol{v}_t$ as a linear combination of $\boldsymbol{g}_i$'s $(1 \leq i \leq t)$.

answer:

$$\begin{aligned} v_t &= \beta v_{t-1} + (1-\beta)g_t \\ &= \beta^2 v_{t-2} + \beta(1-\beta)g_{t-1} + (1-\beta)g_t \\ &= \beta^3 v_{t-3} + \beta^2(1-\beta)g_{t-2} + \beta(1-\beta)g_{t-1} + (1-\beta)g_t \end{aligned}$$

So we can arrive to:

$$v_t = B^t v_0 + \sum_{i=1}^{t} b^{t-i}(1-\beta)g_i$$

and if we assume the v_0 being a zero vector we will arrive to:

$$v_t = \sum_{i=1}^{t} b^{t-i}(1-\beta)g_i$$

2.3 Assume $\boldsymbol{g}_t$ has a stationary distribution independent of t. Show that the running average is biased, i.e. $\mathbb{E}[\boldsymbol{v}_t] \neq \mathbb{E}[\boldsymbol{g}_t]$. Propose a way to eliminate such a bias by rescaling $\boldsymbol{v}_t$.

answer: We can write the expectation value of last equation like this:

$$\mathbb{E}\left[v_t\right] = \sum_{i=1}^{t}(1-\beta)\beta^{t-i}\mathbb{E}\left[g_i\right]$$

$\mathbb{E}\left[g_i\right]$ is independent of t so we the expectation can be equal to:by changing the variable $t-i = j$:

$$\sum_{j=0}^{t-1}\beta^j = (1-\beta)\frac{1-\beta^t}{1-\beta} = (1-\beta^t)$$

so to eliminate such a bias we can rescaling v_t:

$$v_t = \frac{v_t}{1-\beta^t}$$

Question 3 (8-8-6-9-3)**.** In this question, you will analyze the performance of dot-product attention and derive an efficient approximation of it. Consider that *multi-head* dot-product attention for a sequence of length n is defined as follows:

$$\begin{aligned}\text{MultiHead}(\bar{\boldsymbol{Q}}, \bar{\boldsymbol{K}}, \bar{\boldsymbol{V}}) &= \text{Concat}(\text{head}_1, \ldots, \text{head}_h)\boldsymbol{W}^O \\ \text{where} \quad \text{head}_i &= \text{Attention}_{\text{std}}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) \quad (\text{here, } \boldsymbol{Q} := \bar{\boldsymbol{Q}}\boldsymbol{W}_i^Q, \boldsymbol{K} := \bar{\boldsymbol{K}}\boldsymbol{W}_i^K, \boldsymbol{V} := \bar{\boldsymbol{V}}\boldsymbol{W}_i^V) \\ &= \text{softmax}_{\text{row}}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{d_k}}\right)\boldsymbol{V}\end{aligned}$$

where $\bar{\boldsymbol{Q}}, \bar{\boldsymbol{K}}, \bar{\boldsymbol{V}} \in \mathbb{R}^{n \times d_{\text{model}}}$ are the queries, keys, and values, and $\boldsymbol{W}_i^Q, \boldsymbol{W}_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $\boldsymbol{W}_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$ $\forall i$, and $\boldsymbol{W}_O \in \mathbb{R}^{hd_v \times d_{model}}$ are the weights. The softmax subscript "row" indicates that the softmax is computed along the rows, and the Attention subscript "std" indicates that this is the standard variant (we will see other variants later in the question). For this question, you can assume that $d_k = d_v = d_{\text{model}}$ and call the value d.

For calculating the time and space complexities, you can also assume that matrix multiplications are performed naively. As an example, for $\boldsymbol{C} = \boldsymbol{A}\boldsymbol{B}$ where $\boldsymbol{A} \in \mathbb{R}^{p \times q}$, $\boldsymbol{B} \in \mathbb{R}^{q \times r}$, and $\boldsymbol{C} \in \mathbb{R}^{p \times r}$, the time complexity is $\Theta(pqr)$ due to the three nested loops, and the space complexity is $\Theta(pq + qr + pr)$ from storing the inputs and the result.

3.1 What is the time and space complexity of the attention operation carried out by a single head in Θ-notation in terms of n and d ? Use your answer to calculate the time and space complexity of multi-head dot-product attention in terms of n, d, and h, assuming that the heads are computed sequentially. For very long sequences, where does the bottleneck lie ?

answer: for a single head time complexity we can write:

multiplications	time complexity
QK^T	$\Theta(n^2d)$
$(QK^T)V$	$\Theta(n^2d)$
softmax	$\Theta(n^2)$
$(\text{head}_1)W^O$	$\Theta(nd^2)$

So we can sum all this number and we will have: time complexity for single head: $\Theta(n^2d) + \Theta(n^2d) + \Theta(n^2) + \Theta(nd^2) = \Theta(n^2d + nd^2)$
for a single head space complexity we can write:

multiplications	space complexity
Q	$\Theta(nd)$
K^T	$\Theta(dn)$
QK^T	$\Theta(n^2)$
V	$\Theta(nd)$
$(QK^T)V$	$\Theta(nd)$
W^O	$\Theta(d^2)$
$(\text{head}_1)W^O$	$\Theta(nd)$

So we can sum all this number and we will have: space complexity for single head $= \Theta(n^2 + d^2 + nd)$

for the multi-head we can just multiply our complexity by h because we repeat the each operation h time:

time complexity $= \Theta(h(n^2 d + nd^2))$ space complexity $= \Theta(h(n^2 + d^2 + nd))$

The bottleneck for long sequences(big n) is where the n^2 appears in our complexity. in both time and space complexity the bottleneck is when we multiply Q by K^T bottleneck in space and time complexity: $softmax_{row}(\frac{QK^T}{\sqrt{d_k}})$

For the remaining parts, let us focus on the attention operation carried out by a single head. Furthermore, you can omit the scaling factor $\sqrt{d}$ without loss of generality by considering that $\boldsymbol{Q}$ and $\boldsymbol{K}$ can be scaled as desired.

3.2 Let us consider an alternative form of attention, one that performs row-wise softmax on $\boldsymbol{Q}$ and column-wise softmax on $\boldsymbol{K}$ separately as follows:

$$\text{Attention}_{\text{separable}}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{softmax}_{\text{row}}(\boldsymbol{Q})\text{softmax}_{\text{col}}(\boldsymbol{K})^\top \boldsymbol{V}.$$

Prove that $\text{softmax}_{\text{row}}(\boldsymbol{Q})\text{softmax}_{\text{col}}(\boldsymbol{K})^\top$ produces valid categorical distributions in every row, like $\text{softmax}_{\text{row}}(\boldsymbol{QK}^\top)$. If $n \gg d$, show that $\text{Attention}_{\text{separable}}$ can be faster and requires less space than $\text{Attention}_{\text{std}}$. Is $\text{Attention}_{\text{separable}}$ as expressive as $\text{Attention}_{\text{std}}$?

(Hint: For a valid categorical distribution $\boldsymbol{p} \in \mathbb{R}^d$ over d categories, $p_i \geq 0 \, \forall i \in \{1, \ldots, d\}$ and $\sum_{i=1}^d p_i = 1$.)

answer: Softmax $_{\text{row}}(Q) = \begin{bmatrix} s(Q_1) \\ s(Q_2) \\ \vdots \\ s(Q_n) \end{bmatrix}_{nxd}$ $\text{softmax}_{\text{col}}(k) = [s(k_1) \quad s(k_2) \quad \cdots s(k_d)]$

$$softmax_{row}(Q)softmax_{col}(k) = \begin{bmatrix} \sum_{j=1}^d s(q_{1j})s(k_{1j}) & \cdots & \sum_{j=1}^d s(q_{1j})s(k_{nj}) \\ \vdots & \ddots & \vdots \\ \sum_{j=1}^d s(q_{nj})s(k_{1j}) & \cdots & \sum_{j=1}^d s(q_{nj})s(k_{nj}) \end{bmatrix} = \text{final}$$

to prove that this multiplication produces valid categorical distribution we should show that: 1)

$$p_i \geq 0 \text{ for each i } \in \{1, .., n\}$$

each element in the row of the final matrix is greater than zero because q and k are each greater that zero because both of them come from after applying the softmax function (we apply softmax function in each row and column of Q and K which produce each q and k elements)

2) for each row we should prove that:

$$\sum_{i=1}^{n} final_1 = 1$$

$$= \sum_{j=1}^{d} q_{1j} (\sum_{j=1}^{d} k_{1j} + \sum_{j=1}^{d} k_{2j} + ... + \sum_{j=1}^{d} k_{nj})$$

$$= \sum_{j=1}^{d} q_{1j} \sum_{i=1}^{n} k_{ij}$$

$\sum_{j=1}^{d} q_{1j}$ is the sum of all the elements in one row in $softmax_{row} Q$ which is equal to 1. and $\sum_{j=1}^{n} k_{ij}$ is the sum of all the element in one column in $softmax_{col} K$ which is equal to 1.
so we prove that $\sum_{i=1}^{n} final_1 = 1$ we can prove this for every row in final matrix.
answer for the second part: to compute time and space complexity we can consider the multiplication with this order:

$$Attention_{sep}(Q, K, V) = softmax_{row}(Q)(softmax_{col}(K)^T V)$$

so for time complexity we will have:

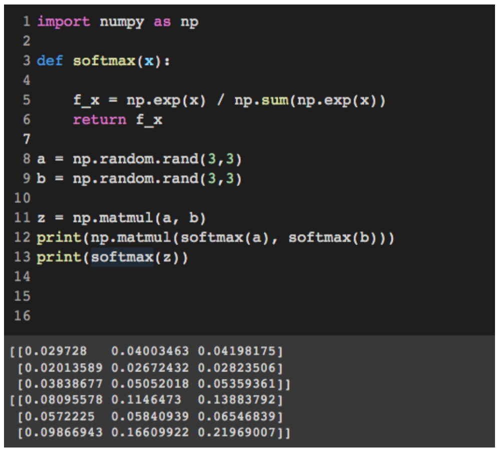
multiplications	time complexity
K$^T V$	$\Theta(nd^2)$
Q(K$^T V$)	$\Theta(nd^2)$

so the time complexity for form of attention is : $\Theta(nd^2)$
and for space complexity we can write:

multiplications	space complexity
V	$\Theta(nd)$
K^T	$\Theta(dn)$
K$^T V$	$\Theta(d^2)$
Q	$\Theta(nd)$
Q(K$^T V$)	$\Theta(nd)$

so the space complexity for form of attention is : $\Theta(d^w + nd)$
we can conclude that $attention_{sep}$ is faster and requires less space compare to $attention_{std}$ if n»d, since in it's time and space complexity there is no quadratic n.
answer for the third part:
$Attention_{sep}$ is not as expressive as $attention_{std}$ because we can not map the output of these function. which is evident in the formula of these function because $softmax(xy) \neq softmax(x)softmax(y)$

I show that softmax(xy) is not equal to softmax(x)softmax(y) by coding the formula on two random 3x3 matrices in python, which you can see in the picture in the following:



```python
import numpy as np

def softmax(x):

    f_x = np.exp(x) / np.sum(np.exp(x))
    return f_x

a = np.random.rand(3,3)
b = np.random.rand(3,3)

z = np.matmul(a, b)
print(np.matmul(softmax(a), softmax(b)))
print(softmax(z))
```

```
[[0.029728   0.04003463 0.04198175]
 [0.02013589 0.02672432 0.02823506]
 [0.03838677 0.05052018 0.05359361]]
[[0.08095578 0.1146473  0.13883792]
 [0.0572225  0.05840939 0.06546839]
 [0.09866943 0.16609922 0.21969007]]
```

3.3 Verify that the standard attention can be written as

$$\text{Attention}_{\text{std}}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \boldsymbol{D}^{-1}\boldsymbol{A}\boldsymbol{V}$$

where $\boldsymbol{A} = \exp\left(\boldsymbol{Q}\boldsymbol{K}^\top\right)$ and $\boldsymbol{D} = \text{diag}(\boldsymbol{A}\mathbf{1})$, where exp is an element-wise operation, diag creates a diagonal matrix from a vector, and $\mathbf{1}$ is a vector of ones. Note that you can store diagonal matrices in linear space and compute matrix multiplications with them in linear time.
Let us now consider a variant $\text{Attention}_{\text{approx}}$ where the elements a_{ij} of $\boldsymbol{A}$ can be represented as $a_{ij} = f(\boldsymbol{q}_i)^\top f(\boldsymbol{k}_j)$ for some $f : \mathbb{R}^d \to \mathbb{R}^m_+$, where $\boldsymbol{q}_i$ and $\boldsymbol{k}_j$ are the ith row of $\boldsymbol{Q}$ and the jth row of $\boldsymbol{K}$ respectively.
If $n \gg m$ and $n \gg d$, how can you use this formulation to make attention efficient ? What is the time and space complexity of $\text{Attention}_{\text{approx}}$? You can assume that f takes $\Theta(md)$ time and space.
(Hint: Decompose the matrix $\boldsymbol{A}$.)

answer:
we know that $D = diag(A1)$ so we can write:

$$D = \begin{cases} \sum_{j=1}^n a_{ij} & i = j \\ 0 & i \neq j \end{cases}$$

also if we take $\text{T} = D^{-1}A$:

$$\begin{aligned} T_{ij} = (D^{-1}A)_{ij} &= \frac{a_{ij}}{\sum_{l=1}^n a_{il}} \\ &= \frac{(exp(QK^T)_{ij})}{\sum_l (exp(QK^T)_{il})} \\ &= softmax_{row}(QK^T) \end{aligned}$$

So we can write for this form of attention

$$\begin{aligned} Attention_{std}(Q, K, V) &= D^{-1}AV \\ &= softmax_{row}(QK^T)V \end{aligned}$$

answer for the second part:
let's consider Q' and K' be the output of f after applying it on them
now Q' and K' both has dimensionality of (n,m).

Also we can write the $Attention_{approx}$ with this order:

$$Attention_{approx} = D^{-1}(Q'(K'^T V))$$

so with this new order we can compute the complexity of time and space:

multiplications	time complexity
apply f on Q	$\Theta(nmd)$
apply f on K	$\Theta(nmd)$
K$'V$	$\Theta(mnd)$
Q$'(K'V)$	$\Theta(nmd)$
D$^{-1}(Q'(K'V))$	$\Theta(n)$

multiplications	space complexity
Q	$\Theta(nd)$
K	$\Theta(nd)$
f	$\Theta(md)$
Q$'$	$\Theta(nm)$
K$'$	$\Theta(nm)$
V	$\Theta(nd)$
K$'V$	$\Theta(md)$
D	$\Theta(n)$
D(Q$'(K'V)$)	$\Theta(nd)$

the time complexity would be: $\Theta(nmd)$ and space complexity would be: $\Theta(nd + md + nm)$
so if we consider n»m and n»d then this form of attention would be more efficient.

*3.4 Prove that in Attention$_{\text{std}}$,

$$a_{ij} = \exp\left(\frac{-\|\boldsymbol{q}_i\|^2}{2}\right) \cdot \mathbb{E}_{\boldsymbol{x}\in\mathcal{N}(\mathbf{0},\mathbf{I})}\left[\exp(\boldsymbol{x}^\top \boldsymbol{q}_i)\exp(\boldsymbol{x}^\top \boldsymbol{k}_j)\right] \cdot \exp\left(\frac{-\|\boldsymbol{k}_j\|^2}{2}\right).$$

Use this result to devise the function $f : \mathbb{R}^d \to \mathbb{R}^m_+$ introduced in the previous part, such that Attention$_{\text{approx}}$ approximates the expectation in Attention$_{\text{std}}$ by sampling.
(Hint 1: If $\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{I})$, $p(\boldsymbol{x}) = (2\pi)^{-d/2}\exp\left(-\frac{1}{2}\|\boldsymbol{x}-\boldsymbol{\mu}\|^2\right)$ and $\int_{\boldsymbol{x}} p(\boldsymbol{x})d\boldsymbol{x} = 1$.)
(Hint 2: $\boldsymbol{x}^\top\boldsymbol{y} = -\frac{1}{2}(\boldsymbol{x}^\top\boldsymbol{x} - (\boldsymbol{x}+\boldsymbol{y})^\top(\boldsymbol{x}+\boldsymbol{y}) + \boldsymbol{y}^\top\boldsymbol{y})$.)

answer: For the fist step we want to simplify E_x :

$$E_{x\in N(0,1)}\left[\exp\left(x^\top q_i\right)\exp\left(x^\top k_j\right)\right]$$

we know that $E_x[f(x)] = \int_x p(x)f(x)dx$ So with considering Hint 1 we can write:

$$E\left[\exp\left(x^\top q_i\right)\exp\left(x^\top k_j\right)\right] = \int_x \left[\exp(x^\top q_i)\exp(x^\top k_j)\left((2\pi)^{-d/2}\exp\left(\frac{-||x||^2}{2}\right)\right)\right]dx$$

Also with the help of Hint 2 we can replace $x^T q_i$ and $x^T k_j$:

$$E[exp(x^T q_i)exp(x^T k_j)] =$$
$$\int_x [exp(-\frac{1}{2}(||x||^2 - ||x+q_i||^2 + ||q_i||^2))exp(-\frac{1}{2}(||x||^2 - ||x+k_j||^2 + ||k_j||^2))((2\pi)^{-d/2}exp(\frac{-||x||^2}{2}))]dx =$$
$$(2\pi)^{-d/2}\int_x [exp(-\frac{1}{2}(||x||^2 - 2xq_i - ||q_i||^2 + ||q_i||^2 - 2xk_j - ||k_j||^2 + ||k_j||^2)]dx =$$
$$(2\pi)^{-d/2}\int_x [exp(-\frac{1}{2}(||x||^2 - 2x(q_i+k_j) + (q_i+k_j)^2 - ||q_i||^2 - ||k_j||^2) - 2(q_ik_j)]dx =$$
$$(2\pi)^{-d/2}\int_x [exp(-\frac{1}{2}(x-(q_i+k_j)^2))exp(\frac{||k_j||^2}{2})exp(\frac{||q_i||^2}{2})exp(q_ik_j)]dx$$

as we saw in Hint 1 we can write $(2\pi)^{-d/2}\int_x [exp(-\frac{1}{2}(x-(q_i+k_j)^2))]dx = 1$
so we will have:

$$E_x = exp(\frac{||k_j||^2}{2})exp(\frac{||q_i||^2}{2})exp(q_ik_j)$$

finally we can prove that:

$$\begin{aligned} a_{ij} &= exp(\frac{-||q_i||^2}{2})exp(\frac{||k_j||^2}{2})exp(\frac{||q_i||^2}{2})exp(\frac{-||k_j||^2}{2})exp(q_ik_j) \\ &= exp(q_ik_j) \end{aligned}$$

answer for the second part:

Let's consider the probability function $p(\boldsymbol{x})$ of the discrete variables $\boldsymbol{x}$ and a function $f(\boldsymbol{x})$ which is $exp(x^T q_i)exp(x^T k_j)$ and we want to evaluate the expectation over the distribution $p(\boldsymbol{x})$ in this regard we approximate the expectation by sampling $\boldsymbol{m}$ points of x, so we can write:

$$\begin{aligned} E[f(x)] &= \sum_{l=1}^{m} f(x_l)p(x_l) \\ &= \frac{1}{m}\sum_{l=1}^{m} f(x_l) \\ &= \frac{1}{m}\sum_{l=1}^{m} exp(x_l^T q_i)exp(x_l^T k_j) \end{aligned}$$

so we will have:

$$a_{ij} = exp(\frac{-||q_i||^2}{2})\frac{1}{m}\sum_{l=1}^{m} exp(x_l^T q_i)exp(x_l^T k_j)exp(\frac{-||k_j||^2}{2})$$

based on above equation we can devise f as following which X is a matrix with all m samples of x:

$$f(y) = \frac{1}{\sqrt{m}} exp(\frac{-||y||^2}{2}) exp(X^T y)$$

3.5 Discuss the implications of the choice of m for Attention$_{\text{approx}}$. What are the trade-offs to think about ?

answer: m is the number of points (x) that is sampling ; the higher the m, the accuracy of the approximation of the expectation function would be more accurate(more precise). However, since M is present in Time and Space complexity, we will have the trade-off between the accuracy of the approximation of the expectation and the efficiency of the attention(the attention can be faster and requires less space). If we pick a big m, the approximation would be more accurate, but the attention would be slower and need more space. Moreover, if we pick small m, the approximation would be less accurate, but the attention would be faster and require less space.

Question 4 (4-5-6-6)**.** In this question, you will reconcile the relationship between L2 regularization and weight decay for the Stochastic Gradient Descent (SGD) and Adam optimizers. Imagine you are training a neural network (with learnable weights θ) with a loss function $L(f(\mathbf{x}^{(i)}, \theta), \mathbf{y}^{(i)})$, under two different schemes. The *weight decay* scheme uses a modified SGD update rule: the weights θ decay exponentially by a factor of λ. That is, the weights at iteration $i+1$ are computed as

$$\theta_{i+1} = \theta_i - \eta \frac{\partial L(f(\mathbf{x}^{(i)}, \theta_i), \mathbf{y}^{(i)})}{\partial \theta_i} - \lambda \theta_i$$

where η is the learning rate of the SGD optimizer. The *L2 regularization* scheme instead modifies the loss function (while maintaining the typical SGD or Adam update rules). The modified loss function is

$$L_{\text{reg}}(f(\mathbf{x}^{(i)}, \theta), \mathbf{y}^{(i)}) = L(f(\mathbf{x}^{(i)}, \theta), \mathbf{y}^{(i)}) + \gamma \|\theta\|_2^2$$

4.1 Prove that the *weight decay* scheme that employs the modified SGD update is identical to an *L2 regularization* scheme that employs a standard SGD update rule.

answer: In SGD we update rules like $\theta_{i+1} = \theta_i - \eta \Delta_\theta L_{reg}$ so we can write:

$$\begin{aligned}\theta_{i+1} &= \theta_i - \eta \frac{\partial}{\partial \theta_i} L(f(x^{(i)}, \theta_i), y^{(i)}) + \eta\gamma \frac{\partial}{\partial \theta_i} ||\theta_i||_2^2 \\ &= \theta_i - \eta \frac{\partial}{\partial \theta_i} L(f(x^{(i)}, \theta_i), y^{(i)}) + 2\eta\gamma\theta_i\end{aligned}$$

This two equation is identical if $\lambda = -2\eta\gamma$

4.2 This question refers to the Adam algorithm as described in the lecture slide (also identical to Algorithm 8.7 of the deep learning book). It turns out that a one-line change to this algorithms gives us Adam with an L2 regularization scheme. Identify the line of the algorithm that needs to change, and provide this one-line modification.

answer: In this regard we can change following line in the algorithm:

$$g \leftarrow \frac{1}{m}\nabla_\theta \sum_i L(f(x^{(i)};\theta), y^{(i)}) + \gamma\theta$$

4.3 Consider a "decoupled" weight decay scheme for the original Adam algorithm (see lecture slides, or equivalently, Algorithm 8.7 of the deep learning book) with the following two update rules.

- The **Adam-L2-reg** scheme computes the update by employing an L2 regularization scheme (same as the question above).
- The **Adam-weight-decay** scheme computes the update as $\boldsymbol{\Delta}\theta = -\left(\epsilon\frac{\hat{\mathbf{s}}}{\sqrt{\hat{\mathbf{r}}}+\delta} + \lambda\theta\right)$.

Now, assume that the neural network weights can be partitioned into two disjoint sets based on their gradient magnitude: $\theta = \{\theta_{\text{small}}, \theta_{\text{large}}\}$, where each weight $\theta_s \in \theta_{\text{small}}$ has a much smaller gradient magnitude than each weight $\theta_l \in \theta_{\text{large}}$. Using this information provided, answer the following questions. In each case, provide a brief explanation as to why your answer holds.

(a) Under the **Adam-L2-reg** scheme, which set of weights among θ_{small} and θ_{large} would you expect to be regularized (i.e., driven closer to zero) more strongly than the other ? Why ?

answer: in Adam optimizer we have:

$$\Delta\theta = -\eta(\frac{\hat{m}}{\sqrt{\hat{v}_t}+\epsilon})$$

where $\hat{m} = \frac{m_t}{1-\beta_1^t}$ and $\hat{v} = \frac{v_t}{1-\beta_2^t}$ also we can define m_t and v_t as:

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)[\frac{\partial L_{reg}}{\partial\theta_t}]$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2)[\frac{\partial L_{reg}}{\partial\theta_t}]^2$$

by replacing above equation and also replacing the gradient from 4.2 we arrive to:

$$\hat{m} = \frac{\beta_1 m_{t-1} + (1-\beta_1)(\nabla\theta_t L_{reg} + \gamma\theta_t)}{1-\beta_1^t}$$
$$\hat{v} = \frac{\beta_2 v_{t-1} + (1-\beta_2)(\nabla\theta_t L_{reg} + \gamma\theta_t)^2}{1-\beta_2^t}$$

By replacing the above equation to general equation:

$$\begin{aligned}
\Delta_\theta &= -\eta\frac{\frac{\beta_1 m_{t-1}+(1-\beta_1)(\nabla_{\theta_t} L_{reg}+\gamma\theta_t)}{1-\beta_1^t}}{\sqrt{\frac{\beta_2 v_{t-1}+(1-\beta_2)(\nabla\theta_t L_{reg}+\gamma\theta_t)^2}{1-\beta_2^t}}+\epsilon} \\
&= -\eta\frac{\beta_1 m_{t-1} + (1-\beta_1)(\nabla\theta_t L_{reg} + \gamma\theta_t)}{(1-\beta_1^t)\sqrt{\frac{\beta_2 v_{t-1}+(1-\beta_2)(\nabla_{\theta_t} L_{reg}+\gamma\theta_t)^2}{1-\beta_2^t}}+\epsilon} \\
&= -\eta\frac{\beta_1 m_{t-1} + (1-\beta_1)(\nabla\theta_t L_{reg})}{(1-\beta_1^t)\sqrt{\frac{\beta_2 v_{t-1}+(1-\beta_2)(\nabla_{\theta_t} L_{reg}+\gamma\theta_t)^2}{1-\beta_2^t}}+\epsilon} + -\eta\frac{(1-\beta_1)(\gamma\theta_t)}{(1-\beta_1^t)\sqrt{\frac{\beta_2 v_{t-1}+(1-\beta_2)(\nabla_{\theta_t} L_{reg}+\gamma\theta_t)^2}{1-\beta_2^t}}+\epsilon}
\end{aligned}$$

here we can see that the regularization term $\frac{(1-\beta_1)(\gamma\theta_t)}{(1-\beta_1^t)\sqrt{\frac{\beta_2 v_{t-1}+(1-\beta_2)(\nabla_{\theta_t} L_{reg}+\gamma\theta_t)^2}{1-\beta_2^t}}+\epsilon}$ is normalized with $\sqrt{v_t}$ so it results in huge decay for large θ cause we can see $\nabla^2_{delta} L_{reg}$ in nominator, which causes a larger nominator. Moreover, for small θ we will have smaller weight decay(smaller change) compared to the large weight because we will have a smaller nominator. so Adam-L2-reg will regularized more strongly θ_{small} compared to the θ_{large}

(b) Would your answer change for the **Adam-weight-decay** scheme ? Why/why not ?

answer: Yes, my answer would change because, in Adam-weight-decay, we can see that the regularization term does not normalize with moving averages, and weights(gradients) do not affect by their magnitude. So none of the θ_{large} or θ_{small} would be regularized stronger than the other.

(Note: for the two sub-parts above, we are interested in the rate at which the weights are regularized, *relative* to their initial magnitudes.)

4.4 In the context of all of the discussion above, argue that weight decay is a better scheme to employ as opposed to L2 regularization ; particularly in the context of adaptive gradient based optimizers. (Hint: think about how each of these schemes regularize each parameter, and also about what the overarching objective of regularization is).

answer: The Adam-weight-decay would be better because it would regularize the parameter without the effect of their gradient magnitude. On the other hand, in L2 regularization, we take the adaptive gradient when the regularization term is added, which leads to affecting the parameter with the magnitude of the gradient. So with weight decay regularization, we would take advantage of the same strength regularization over all the parameters with different magnitudes. This is the main objective of the l2 regularizer, which is to regularize every parameter the same and push them to zero.