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.
Note that here we are going to assume that $\sigma(\boldsymbol{h}_{t-1}) = \boldsymbol{g}_{t-1}$ as suggested in the question.
From that:

$$\begin{aligned}
\boldsymbol{g}_t &= \sigma(\boldsymbol{W}\boldsymbol{g}_{t-1} + \boldsymbol{U}\boldsymbol{x}_t + \boldsymbol{b}) \\
\boldsymbol{h}_t &= \boldsymbol{W}\sigma(\boldsymbol{h}_{t-1}) + \boldsymbol{U}\boldsymbol{x}_t + \boldsymbol{b} \\
&\Leftrightarrow \boldsymbol{h}_t - \boldsymbol{W}\sigma(\boldsymbol{h}_{t-1}) = \boldsymbol{U}\boldsymbol{x}_t + \boldsymbol{b}
\end{aligned}$$

Therefore,

$$\begin{aligned}
\boldsymbol{g}_t &= \sigma(\boldsymbol{W}\boldsymbol{g}_{t-1} + \boldsymbol{U}\boldsymbol{x}_t + \boldsymbol{b}) \\
&= \sigma(\boldsymbol{W}\boldsymbol{g}_{t-1} + \boldsymbol{h}_t - \boldsymbol{W}\sigma(\boldsymbol{h}_{t-1})) \\
&= \sigma(\cancel{\boldsymbol{W}\boldsymbol{g}_{t-1}} + \boldsymbol{h}_t - \cancel{\boldsymbol{W}\sigma(\boldsymbol{h}_{t-1})})
\end{aligned}$$

$$\boxed{\boldsymbol{g}_t = \sigma(\boldsymbol{h}_t)}$$

■.

*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.
Using the chain rule:

$$\frac{\partial \boldsymbol{h}_T}{\partial \boldsymbol{h}_o} = \frac{\partial \boldsymbol{h}_T}{\partial \boldsymbol{h}_{T-1}} \frac{\partial \boldsymbol{h}_{T-1}}{\partial \boldsymbol{h}_{T-2}} \cdots \frac{\partial \boldsymbol{h}_1}{\partial \boldsymbol{h}_0}$$

Then:

$$\frac{\partial \boldsymbol{h}_T}{\partial \boldsymbol{h}_{T-1}} = \frac{\partial}{\partial \boldsymbol{h}_{T-1}} \boldsymbol{W}\sigma(\boldsymbol{h}_{T-1}) + \boldsymbol{U}\boldsymbol{x}_T + \boldsymbol{b} = \boldsymbol{W}\frac{\partial \sigma(\boldsymbol{h}_{T-1})}{\partial \boldsymbol{h}_{T-1}}$$

Using given properties of L_2 norm:

$$\begin{aligned} ||\frac{\partial \boldsymbol{h}_T}{\partial \boldsymbol{h}_{T-1}}|| &\leq ||\boldsymbol{W}|| \cdot ||\frac{\partial \sigma(\boldsymbol{h}_{T-1})}{\partial \boldsymbol{h}_{T-1}}|| \\ &\leq \sqrt{\lambda_1(\boldsymbol{W}^\top \boldsymbol{W})} \cdot ||\frac{\partial \sigma(\boldsymbol{h}_{T-1})}{\partial \boldsymbol{h}_{T-1}}|| \\ &\leq \frac{\delta}{\gamma}\gamma \leq \delta \end{aligned}$$

So:

$$||\frac{\partial \boldsymbol{h}_T}{\partial \boldsymbol{h}_0}|| \leq \prod_{i=1}^{T} ||\frac{\partial \boldsymbol{h}_i}{\partial \boldsymbol{h}_{i-1}}|| \leq \delta^T$$

Therefore, as $\in [0, 1)$ as T $\to \infty, \delta^T \to 0$. So as $||\frac{\partial \boldsymbol{h}_T}{\partial \boldsymbol{h}_0}|| \leq \delta^T$, $||\frac{\partial \boldsymbol{h}_T}{\partial \boldsymbol{h}_0}|| \to 0$. ■.

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.
This is a necessary condition but not a sufficient one for the gradient to explode. This can be shown is we have $\boldsymbol{W}$ such that $\delta \leq \sqrt{\lambda_1(\boldsymbol{W}^\top \boldsymbol{W})}\gamma \leq 1$ than the gradient will not explode (it will vanished if $\delta \leq \sqrt{\lambda_1(\boldsymbol{W}^\top \boldsymbol{W})}\gamma < 1$ and will be stabilized if $\delta = \sqrt{\lambda_1(\boldsymbol{W}^\top \boldsymbol{W})}\gamma = 1$).

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.

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.

Recursive update expressions:

- SGD with momentum:

$$\begin{aligned}\Delta\boldsymbol{\theta}_t &= -\boldsymbol{v}_t = -(\alpha\boldsymbol{v}_{t-1} + \epsilon\boldsymbol{g}_t) \\ &= -\alpha\boldsymbol{v}_{t-1} - \epsilon\boldsymbol{g}_t = \alpha\Delta\boldsymbol{\theta}_{t-1} - \epsilon\boldsymbol{g}_t\end{aligned}$$

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

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

Therefore, the two update rules are equivalent :
$\Delta\boldsymbol{\theta}_t = \alpha\Delta\boldsymbol{\theta}_{t-1} - \epsilon\boldsymbol{g}_t = \beta\Delta\boldsymbol{\theta}_{t-1} - \delta(1-\beta)\boldsymbol{g}_t$
when $\alpha = \beta$ 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.

$$\begin{aligned}
\Delta\theta_t &= -\delta \boldsymbol{v}_t = -\delta(\beta \boldsymbol{v}_{t-1} + (1-\beta)\boldsymbol{g}_t) \\
&= -\delta\beta \boldsymbol{v}_{t-1} - \delta(1-\beta)\boldsymbol{g}_t \\
&= -\delta\beta(\beta \boldsymbol{v}_{t-2} + (1-\beta)\boldsymbol{g}_{t-1}) - \delta(1-\beta)\boldsymbol{g}_t \\
&= -\delta\beta^2 \boldsymbol{v}_{t-2} - \delta\beta(1-\beta)\boldsymbol{g}_{t-1} - \delta(1-\beta)\boldsymbol{g}_t \\
&= -\delta\beta^2(\beta \boldsymbol{v}_{t-3} + (1-\beta)\boldsymbol{g}_{t-2}) - \delta\beta(1-\beta)\boldsymbol{g}_{t-1} - \delta(1-\beta)\boldsymbol{g}_t \\
&= -\delta\beta^3 \boldsymbol{v}_{t-3} - \delta\beta^2(1-\beta)\boldsymbol{g}_{t-2} - \delta\beta(1-\beta)\boldsymbol{g}_{t-1} - \delta(1-\beta)\boldsymbol{g}_t \\
&= -\delta \sum_{i=1}^{t} \beta^{i-1}(1-\beta)\boldsymbol{g}_{t-i+1} = \boxed{-\delta(1-\beta)\sum_{i=1}^{t}\beta^{t-i}\boldsymbol{g}_i}
\end{aligned}$$

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 $\boldsymbol{v}_t$.

Anwser.

$$\begin{aligned}
\mathbb{E}[\boldsymbol{v}_t] &= \mathbb{E}[(1-\beta)\sum_{i=1}^{t}\beta^{t-i}\boldsymbol{g}_i] \\
&= (1-\beta)\mathbb{E}[\sum_{i=1}^{t}\beta^{t-i}\boldsymbol{g}_i] \\
&= (1-\beta)\sum_{i=1}^{t}\mathbb{E}[\beta^{t-i}\boldsymbol{g}_i] \\
&= (1-\beta)\sum_{i=1}^{t}\beta^{t-i}\mathbb{E}[\boldsymbol{g}_i] \\
&= \mathbb{E}[\boldsymbol{g}_t](1-\beta)\sum_{i=1}^{t}\beta^{t-i} + \delta \\
&= \mathbb{E}[\boldsymbol{g}_t](1-\beta^t) + \delta
\end{aligned}$$

Therefore I just demonstrate that $\mathbb{E}[\boldsymbol{v}_t]$ is biased as $\mathbb{E}[\boldsymbol{v}_t] \neq \mathbb{E}[\boldsymbol{g}_t]$.

We can fix that by using the same trick as used in the **ADAM algorithm**:

$$\mathbb{E}[\hat{\boldsymbol{v}}_t] = \mathbb{E}[\boldsymbol{g}_t] = \frac{\mathbb{E}[\boldsymbol{v}_t] - \delta}{(1-\beta^t)}$$

Where $\mathbb{E}[\hat{\boldsymbol{v}}_t]$ is an unbiased exponential running average as is very close to 0.

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?

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.

Answer.
using n the sequence length, d the embedding dimension, and h the number of heads:

Single-head time complexity

The time complexity for a single head is $\boxed{\mathcal{O}(n^2 d)}$ as $\boldsymbol{Q}.\boldsymbol{K}^\top$ multiplies a $n \times d$ with a $d \times n$ matrix leading to a complexity of $\mathcal{O}(n^2 d)$. Then we need to compute the softmax of that resulting matrix ($\boldsymbol{Q}.\boldsymbol{K}^\top \in \mathbb{R}^{n \times n}$) which result in a time complexity of $\mathcal{O}(n^2)$ as we need to compute the softmax for each element of the matrix. Finally, we multiply the matrix obtain after applying the softmax ($\in \mathbb{R}^{n \times n}$) by $\boldsymbol{V}$ ($\in \mathbb{R}^{n \times d}$) leading to a final time complexity of $\mathcal{O}(n^2 d)$.

Single-head space complexity

The space complexity of a single head is $\boxed{\mathcal{O}(n^2 + nd)}$ as we first need to store the result of softmax($\boldsymbol{Q}.\boldsymbol{K}^\top$) resulting in the storage of n^2 values (as it is a $n \times n$ matrix). After that, wee need to store the result of softmax($\boldsymbol{Q}.\boldsymbol{K}^\top$).$\boldsymbol{V}$ ($\in \mathbb{R}^{n \times d}$) wich need to store nd values. All together those storage needs leads to a space complexity of $\mathcal{O}(n^2 + nd)$.

Multi-head time complexity

The time complexity of the multi-head dot-product is $\boxed{\mathcal{O}(n^2dh + nd^2h)}$. Firstly we can take the result from the single-head complexity wich is $\mathcal{O}(n^2d)$ but now we have h heads to compute **sequentially** leadings to a time complexity of $\mathcal{O}(n^2dh)$. Then we need to concatenate those heads ($\in \mathbb{R}^{n\times d}$) into one matrix ($\in \mathbb{R}^{n\times dh}$) and multiply that matrix with $\boldsymbol{W}^o \in \mathbb{R}^{dh\times d}$ leading to a complexity of $\mathcal{O}(nd^2h)$. Therefore we can plug to two computational part into the same complexity and get the final multi-head attention dot-prodcut time complexity which is $\mathcal{O}(n^2dh + nd^2h)$.

Multi-head space complexity

The space complexity of the multi-head dot-product is $\boxed{\mathcal{O}(n^2h + 2ndh + nd)}$. Firstly we can take the result from the single-head space complexity wich is $\mathcal{O}(n^2 + nd)$ but now we have h heads to compute **sequentially** leadings to a multi-head space complexity of $\mathcal{O}(n^2h + ndh)$. Then we need to concatenate those heads ($\in \mathbb{R}^{n\times d}$) into one matrix ($\in \mathbb{R}^{n\times dh}$) which requires to store ndh values leading to a space complexity of $\mathcal{O}(ndh)$ for the concatenation operation. Finally, we have to multiply the result of that operation ($\in \mathbb{R}^{n\times dh}$) with $\boldsymbol{W}^o \in \mathbb{R}^{dh\times d}$ leading to a space complexity of $\mathcal{O}(nd)$. Therefore we can put all those storage complexity together to finally get the multi-head dot-product space complexity which is $\mathcal{O}(n^2h + 2ndh + nd)$.

For very long sequences, where does the bottleneck lie ?

The bottleneck lie on the fact that for both multi-head dot-product time and space complexity are dependant of n^2, the sequence length squared in their computation time or storage size. That means that we can't afford having arbitrarelly long sequence length as it will quadratically increase both computation time and needed storage size.

If we want to localize what in the equation leads to such sequence-length dependant complexities we can guess that it is related to $\boldsymbol{Q}.\boldsymbol{K}^\top$ as it is always the step that create the n^2 of both time and space complexities.

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 $\text{softmax}_{\text{row}}(\boldsymbol{Q}\boldsymbol{K}^\top)$. 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.

Proof. We need to prove that $\text{softmax}_{\text{row}}(\boldsymbol{Q})\text{softmax}_{\text{col}}(\boldsymbol{K})^\top$ produces a valid categorical distribution over every row like $\text{softmax}_{\text{row}}(\boldsymbol{Q}\boldsymbol{K}^\top)$.

To do so, let fix $\boldsymbol{A} = \text{softmax}_{\text{row}}(\boldsymbol{Q})\text{softmax}_{\text{col}}(\boldsymbol{K})^\top = \boldsymbol{BC}$

$$\boldsymbol{A} = \begin{bmatrix} B_{1,1} & \dots & B_{1,d} \\ \vdots & \ddots & \vdots \\ B_{n,1} & \dots & B_{n,d} \end{bmatrix} \begin{bmatrix} C_{1,1} & \cdots & C_{1,n} \\ \vdots & \ddots & \vdots \\ C_{d,1} & \dots & C_{d,n} \end{bmatrix}$$

Notice that,

$$\sum_{j=1}^{d} B_{i,j} = 1 \qquad \forall i \in \{1, \dots, n\}$$

$$\sum_{j=1}^{n} C_{i,j} = 1 \qquad \forall i \in \{1, \dots, d\}$$

Thefore, $\boldsymbol{A} = \begin{bmatrix} B_{1,.} \cdot C_{.,1} & \dots & B_{1,.} \cdot C_{.,n} \\ \vdots & \ddots & \vdots \\ B_{n,.} \cdot C_{.,1} & \dots & B_{n,.} \cdot C_{.,n} \end{bmatrix}$

Thus,

$$\begin{aligned} \sum_{j=1}^{n} A_{i,j} &= \sum_{j=1}^{n} B_{i,.} \cdot C_{.,j} \\ &= \sum_{j=1}^{n} \sum_{k=1}^{d} B_{i,k} C_{k,j} \\ &= \sum_{k=1}^{d} B_{i,k} (\sum_{j=1}^{n} C_{k,j}) \\ &= \sum_{k=1}^{d} B_{i,k} 1 \qquad \text{using the softmax definition} \\ &= 1 \qquad \forall i \in \{1, \dots, n\} \end{aligned}$$

So I have demonstrated that the sum of any row of matrix $\boldsymbol{A}$ sums to 1. All element of $\boldsymbol{A}$ are greather or equal to 0 as we applied the softmax to $\boldsymbol{Q}$ and $\boldsymbol{K}$ before the matrix multiplication leading to have all element of each of those matrices positives (or equal to zero). Therefore multiplying or summing positives elements with positives elements will never gives negatives elements. I have so demonstrated that the result of $\text{Attention}_{\text{separable}}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V})$ produces valid categorical distrubition over every row of $\boldsymbol{A}$.

$\text{Attention}_{\text{separable}}$ can be faster and requires less space than $\text{Attention}_{\text{std}}$?
Let fix $n >> d$ with that constraint the $\text{Attention}_{\text{separable}}$ will be faster to compute the attention as the softmax will consist of 2 $\mathcal{O}(nd)$ instead of $\mathcal{O}(n^2)$ as in the $\text{Attention}_{\text{std}}$ leading to compute the softmax much faster as $n >> d$. It will also consume less storage as the result of two softmax consist of $2nd$ values for $\text{Attention}_{\text{separable}}$ instead of n^2 values for $\text{Attention}_{\text{std}}$ which will take less place in memory if $n >> d$.

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.

$\text{Attention}_{\text{std}} = \text{softmax}(\boldsymbol{Q}\boldsymbol{K}^\top)\boldsymbol{V}$

We can define $\boldsymbol{C} = \boldsymbol{Q}\boldsymbol{K}^\top$
Therefore,

$$\begin{aligned}
\text{Attention}_{\text{std}} &= \text{softmax}(\boldsymbol{Q}\boldsymbol{K}^\top)\boldsymbol{V} \\
&= \text{softmax}(\boldsymbol{C})\boldsymbol{V} \\
&= \text{diag}(\begin{bmatrix} 1/\sum_{j=1}^n e^{\boldsymbol{C}_{1,j}} & \dots & 1/\sum_{j=1}^n e^{\boldsymbol{C}_{n,j}} \end{bmatrix})e^{\boldsymbol{C}}\boldsymbol{V} \\
&= \text{diag}(\begin{bmatrix} \sum_{j=1}^n e^{\boldsymbol{C}_{1,j}} & \dots & \sum_{j=1}^n e^{\boldsymbol{C}_{n,j}} \end{bmatrix})^{-1}e^{\boldsymbol{C}}\boldsymbol{V}
\end{aligned}$$

Then, as $\boldsymbol{C} = \boldsymbol{Q}\boldsymbol{K}^\top$ we have $\boldsymbol{A} = e^{\boldsymbol{Q}\boldsymbol{K}^\top} = e^{\boldsymbol{C}}$.
We also have $\boldsymbol{D} = \text{diag}(\boldsymbol{A}\mathbf{1}) = \text{diag}(\begin{bmatrix} \sum_{j=1}^n e^{\boldsymbol{C}_{1,j}} & \dots & \sum_{j=1}^n e^{\boldsymbol{C}_{n,j}} \end{bmatrix})$.

Thus, we can show that:

$$\begin{aligned}
\text{Attention}_{\text{approx}} &= \boldsymbol{D}^{-1}\boldsymbol{A}\boldsymbol{V} \\
&= \text{diag}(\begin{bmatrix} \sum_{j=1}^n e^{\boldsymbol{C}_{1,j}} & \dots & \sum_{j=1}^n e^{\boldsymbol{C}_{n,j}} \end{bmatrix})^{-1}e^{\boldsymbol{C}}\boldsymbol{V} \\
&= \text{diag}(\begin{bmatrix} \sum_{j=1}^n e^{\boldsymbol{Q}\boldsymbol{K}^\top_{1,j}} & \dots & \sum_{j=1}^n e^{\boldsymbol{Q}\boldsymbol{K}^\top_{n,j}} \end{bmatrix})^{-1}e^{\boldsymbol{Q}\boldsymbol{K}^\top}\boldsymbol{V} \\
&= \text{diag}(\begin{bmatrix} 1/\sum_{j=1}^n e^{\boldsymbol{Q}\boldsymbol{K}^\top_{1,j}} & \dots & 1/\sum_{j=1}^n e^{\boldsymbol{Q}\boldsymbol{K}^\top_{n,j}} \end{bmatrix})e^{\boldsymbol{Q}\boldsymbol{K}^\top}\boldsymbol{V} \\
&= \text{softmax}_{row}(\boldsymbol{Q}\boldsymbol{K}^\top)\boldsymbol{V}
\end{aligned}$$

Then we can compute the time and space of the $\text{Attention}_{\text{approx}}$. As we assume that $n >> m$ and $n >> d$, we start to compute $\boldsymbol{A}\boldsymbol{V}$ which have a time complexity $\mathcal{O}(nd^2)$ as $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ and $\boldsymbol{V} \in \mathbb{R}^{n\times d}$

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

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

Use this result to devise the function $f : \mathbb{R}^d \to \mathbb{R}^m_+$ introduced in the previous part, such that $\text{Attention}_{\text{approx}}$ approximates the expectation in $\text{Attention}_{\text{std}}$ 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.

Based on the Performers's paper (`https://arxiv.org/abs/2009.14794`) and using the fact that $\text{Attention}_{\text{std}}(\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V}) = \boldsymbol{D}^{-1}\boldsymbol{A}\boldsymbol{V}$ as proven in question 3.3 :

First we can show that:

$$\boldsymbol{A}_{i,j} = \exp\left(\boldsymbol{Q}_{\boldsymbol{i},.}\boldsymbol{K}^\top_{.,\boldsymbol{j}}\right) = \exp\left(-\|\boldsymbol{Q}_{\boldsymbol{i},.}\|^2/2\right) \cdot \exp\left(\|\boldsymbol{q_i} + \boldsymbol{K}^\top_{.,\boldsymbol{j}}\|^2/2\right) \cdot \exp\left(-\|\boldsymbol{K}^\top_{.,\boldsymbol{j}}\|^2/2\right) \tag{1}$$

where $\boldsymbol{Q}_{i,.}, \boldsymbol{K}^\top_{.,j} \in \mathbb{R}^d$.

Thefore, with supposing that $\boldsymbol{x} \in \mathbb{R}^d$. Remind that for any $\mu \in \mathbb{R}^d$

$$(2\pi)^{-d/2}\int \exp\left(-\|\boldsymbol{x}-\boldsymbol{\mu}\|_2^2/2\right) d\boldsymbol{x} = 1$$

Thus,

$$\begin{aligned}
\exp(\|\boldsymbol{q}+\boldsymbol{k}\|^2/2) &= (2\pi)^{-d/2}\exp\left(\|\boldsymbol{q}+\boldsymbol{k}\|^2/2\right)\int \exp\left(-\|\boldsymbol{x}-(\boldsymbol{q}+\boldsymbol{k})\|^2/2\right) d\boldsymbol{x} \\
&= (2\pi)^{-d/2}\int \exp\left(-\|\boldsymbol{x}\|^2/2 + \boldsymbol{x}^\top(\boldsymbol{q}+\boldsymbol{k}) - \|\boldsymbol{q}+\boldsymbol{k}\|^2/2 + \|\boldsymbol{q}+\boldsymbol{k}\|^2/2\right) d\boldsymbol{x} \\
&= (2\pi)^{-d/2}\int \exp\left(-\|\boldsymbol{x}\|^2/2 + \boldsymbol{x}^\top(\boldsymbol{q}+\boldsymbol{k})\right) d\boldsymbol{x} \\
&= (2\pi)^{-d/2}\int \exp\left(-\|\boldsymbol{x}\|^2/2\right)\cdot\exp\left(\boldsymbol{x}^\top\boldsymbol{q}\right)\cdot\exp\left(\boldsymbol{x}^\top\boldsymbol{k}\right) d\boldsymbol{x} \\
&= \mathbb{E}_{x\sim\mathcal{N}(\mathbf{0}_d,\mathbf{I}_d)}\left[\exp\left(x^\top\boldsymbol{q}\right)\cdot\exp\left(x^\top\boldsymbol{k}\right)\right]
\end{aligned}$$

Finally, using (1) we can proof that in $\text{Attention}_{\text{approx}}$:

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

■.

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

Answer.
Here we are talking about choosing between accuracy of estimate of $\boldsymbol{A}$ and computation time: This is a trade-off in the sense that big m will leads to less variance on the estimation of $\boldsymbol{A}$ but a higher cost for computation, where a low m leads to more variance on estimation of $\boldsymbol{A}$ but much less computation time.

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.

Firstly the derivative of the loss and the regularizer term with respect to parameters θ_i is,

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

Therefore,

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

So, Weight decay scheme is equal to L2 regularization scheme when $\boxed{\lambda = 2\eta\gamma}$.

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.

The line that needs to be changed is the following:

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

The modification consists of adding the L2 penalty term to the loss as following (g using the modification will become g' here):

$$\begin{aligned}
\boldsymbol{g}' &\leftarrow \frac{1}{m}\nabla_{\boldsymbol{\theta}}((\sum_i L(f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), \boldsymbol{y}^{(i)}) + \gamma||\boldsymbol{\theta}||_2^2) \\
&\leftarrow \frac{1}{m}\nabla_{\boldsymbol{\theta}}(\sum_i L(f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), \boldsymbol{y}^{(i)}) + \nabla_{\boldsymbol{\theta}}\gamma||\boldsymbol{\theta}||_2^2 \\
&\leftarrow \frac{1}{m}\nabla_{\boldsymbol{\theta}}(\sum_i L(f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), \boldsymbol{y}^{(i)}) + 2\gamma\boldsymbol{\theta}
\end{aligned}$$

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 ?

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

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

Answer.

Adam L_2 regularization scheme, $\theta_{small}, \theta_{large}$?

θ_{small} will be push further 0 as r is computed using the gradient squared. Then as, $\hat{r}$ is at the denominator of $\Delta\theta$, a larger $\hat{r}$ will leads to small changes where small $\hat{r}$ will changed much more θ.

Adam weight decay scheme, $\theta_{small}, \theta_{large}$?

The answer to that question depends on the value of θ as it is directly proportional to the weights updates $\Delta\theta$ which is given by $\Delta\theta = -\epsilon\frac{\hat{\mathbf{s}}}{\sqrt{\hat{r}}+\delta} - \lambda\theta$. Thus, $\theta_{\text{large}}, \theta_{\text{small}}$ will change depending on the value of θ.

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.
Weight decay is better than L2 regularization as it avoid sparsity in the sense of it is not pushing θ_{small} as L2 regularization does on Adam L_2 regularization scheme. This is a good news because even if in some case sparsity can be interesting pushing θ_{small} towards 0 leads to only keep high θ_{large} which will create a lot of variance between weights that are set to 0 and the other ones. Furthermore, the goal of regularization is to avoid overfitting.For instance, by penalizing large weights which Adam using L2 regularization scheme doesn't seems to do in constrast to AdamW (Adam Weight Decay).