Full name : Saad Benslimane
ID : 20228273
Email adress : saad.benslimane@umontreal.ca
March 23th, 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 1.1

We have :

$$\begin{aligned}\boldsymbol{h}_t &= \boldsymbol{W}\sigma(\boldsymbol{h}_{t-1}) + \boldsymbol{U}\boldsymbol{x}_t + \boldsymbol{b}\\ &= \boldsymbol{W}\sigma(\boldsymbol{W}\sigma(\boldsymbol{h}_{t-2}) + \boldsymbol{U}\boldsymbol{x}_{t-1} + \boldsymbol{b}) + \boldsymbol{U}\boldsymbol{x}_t + \boldsymbol{b}\end{aligned}$$

If we suppose that $\sigma(\boldsymbol{h}_t) = \boldsymbol{g}_t$ holds for $t-1$, then :

$$\begin{aligned}\boldsymbol{h}_t &= \boldsymbol{W}\sigma(\boldsymbol{W}\boldsymbol{g}_{t-2} + \boldsymbol{U}\boldsymbol{x}_{t-1} + \boldsymbol{b}) + \boldsymbol{U}\boldsymbol{x}_t + \boldsymbol{b}\\ &= \boldsymbol{W}\boldsymbol{g}_{t-1} + \boldsymbol{U}\boldsymbol{x}_t + \boldsymbol{b}\end{aligned}$$

Now let's pass both sides through a sigmoid and using the equation for $\boldsymbol{g}_i$:

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

Thus we have proved the inductive bias and shown that : $\sigma(\boldsymbol{h}_t) = \boldsymbol{g}_t$.

*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 1.2

Using the chain rule, we have the derivative w.r.t $\boldsymbol{h}_0$:

$$\frac{\partial \boldsymbol{h}_T}{\partial \boldsymbol{h}_0} = \frac{\partial \boldsymbol{h}_T}{\partial \boldsymbol{h}_{T-1}} \frac{\partial \boldsymbol{h}_{T-1}}{\partial \boldsymbol{h}_{T-2}} \frac{\partial \boldsymbol{h}_{T-2}}{\partial \boldsymbol{h}_{T-3}} \frac{\partial \boldsymbol{h}_{T-3}}{\partial \boldsymbol{h}_{T-4}} .... \frac{\partial \boldsymbol{h}_1}{\partial \boldsymbol{h}_0}$$

And for every $t \in [1, ..., T]$, we have :

$$\begin{aligned}\frac{\partial \boldsymbol{h}_t}{\partial \boldsymbol{h}_{t-1}} &= \frac{\partial}{\partial \boldsymbol{h}_{t-1}} (\boldsymbol{W}\sigma(\boldsymbol{h}_{t-1}) + \boldsymbol{U}\boldsymbol{x}_t + b) \\ &= \boldsymbol{W}\frac{\partial}{\partial \boldsymbol{h}_{t-1}} (\sigma(\boldsymbol{h}_{t-1})) \\ &= \boldsymbol{W}\sigma^{'}(\boldsymbol{h}_{t-1})\end{aligned}$$

Replacing it in the first derivative, we have :

$$\frac{\partial \boldsymbol{h}_T}{\partial \boldsymbol{h}_0} = \boldsymbol{W}\sigma^{'}(\boldsymbol{h}_{T-1})\boldsymbol{W}\sigma^{'}(\boldsymbol{h}_{T-2})\boldsymbol{W}\sigma^{'}(\boldsymbol{h}_{T-3})....\boldsymbol{W}\sigma^{'}(\boldsymbol{h}_0)$$

Using the property of the L_2, $||\boldsymbol{A}\boldsymbol{B}|| \leq ||\boldsymbol{A}||\,||\boldsymbol{B}||$ multiple times, we have :

$$\begin{aligned}||\frac{\partial \boldsymbol{h}_T}{\partial \boldsymbol{h}_0}|| &= ||\boldsymbol{W}\sigma^{'}(\boldsymbol{h}_{T-1})\boldsymbol{W}\sigma^{'}(\boldsymbol{h}_{T-2})\boldsymbol{W}\sigma^{'}(\boldsymbol{h}_{T-3})....\boldsymbol{W}\sigma^{'}(\boldsymbol{h}_0)|| \\ &\leq ||\boldsymbol{W}||\,||\sigma^{'}(\boldsymbol{h}_{T-1})||....||\boldsymbol{W}||\,||\sigma^{'}(\boldsymbol{h}_0)|| \\ &\leq ||\boldsymbol{W}||^T \gamma^T = (||\boldsymbol{W}||\,\gamma)^T\end{aligned}$$

Using the second property of the L_2 $||\boldsymbol{A}|| = \sqrt{\lambda_1(\boldsymbol{A}^\top \boldsymbol{A})}$ and $\lambda_1(\boldsymbol{W}^\top \boldsymbol{W}) \leq \frac{\delta^2}{\gamma^2}$, we have :

$$||\frac{\partial \boldsymbol{h}_T}{\partial \boldsymbol{h}_0}|| \leq \left(\sqrt{\lambda_1(\boldsymbol{W}^\top \boldsymbol{W})}\,\gamma\right)^T \leq \left(\frac{\delta}{\gamma}\gamma\right)^T = \delta^T$$

Since we have $0 \leq \delta < 1$, then $\delta^T \to 0$ as $T \to \infty$. Thus : $||\frac{\partial \boldsymbol{h}_T}{\partial \boldsymbol{h}_0}|| \to 0$ as $T \to \infty$, as we can see the derivatives will vanish over time.

[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.

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 1.3

For the gardient to explode, it is necessary that the largest eigenvalue of the weights is larger than $\frac{\delta^2}{\gamma^2}$. Otherwise, the gardient will vanish as we saw in the previous question. It will become a sufficient condition when γ is infinitely close to 0 making the limit of $\frac{\delta^2}{\gamma^2}$ to be evaluated as $T \to \infty$.

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 2.1

For SGD with momentum, we have :

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

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

For SGD with running average, we have :

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

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

With $\alpha = \beta$ and $\epsilon = \delta(1-\beta)$, we can see that the two results are equivalents.

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 2.2

Since $\boldsymbol{v}_0 = 0$, we have :

$$\begin{aligned}
\boldsymbol{v}_1 &= \beta \boldsymbol{v}_0 + (1-\beta)\boldsymbol{g}_1 = (1-\beta)\boldsymbol{g}_1 \\
\boldsymbol{v}_2 &= \beta \boldsymbol{v}_1 + (1-\beta)\boldsymbol{g}_2 = \beta(1-\beta)\boldsymbol{g}_1 + (1-\beta)\boldsymbol{g}_2 = (1-\beta)(\beta\boldsymbol{g}_1 + \boldsymbol{g}_2) \\
\boldsymbol{v}_3 &= \beta \boldsymbol{v}_2 + (1-\beta)\boldsymbol{g}_3 = \beta(1-\beta)(\beta\boldsymbol{g}_1 + \boldsymbol{g}_2) + (1-\beta)\boldsymbol{g}_3 = (1-\beta)(\beta^2\boldsymbol{g}_1 + \beta\boldsymbol{g}_2 + \boldsymbol{g}_3) \\
\boldsymbol{v}_4 &= \beta \boldsymbol{v}_3 + (1-\beta)\boldsymbol{g}_4 = \beta(1-\beta)(\beta^2\boldsymbol{g}_1 + \beta\boldsymbol{g}_2 + \boldsymbol{g}_3) + (1-\beta)\boldsymbol{g}_4 \\
&= (1-\beta)(\beta^3\boldsymbol{g}_1 + \beta^2\boldsymbol{g}_2 + \beta\boldsymbol{g}_3 + \boldsymbol{g}_4)
\end{aligned}$$

from this pattern, for $\boldsymbol{g}_i$'s $(1 \leq i \leq t)$:

$$\begin{aligned}
\boldsymbol{v}_t &= (1-\beta)(\beta^{t-1}\boldsymbol{g}_1 + \beta^{t-2}\boldsymbol{g}_2 + \beta^{t-3}\boldsymbol{g}_3 + ... + \beta\boldsymbol{g}_t) \\
&= (1-\beta)\sum_{i=1}^{t} \beta^{t-i}\boldsymbol{g}_i
\end{aligned}$$

Using recurrence, let's proof this property that : $\boldsymbol{v}_t = (1-\beta)\sum_{i=1}^{t} \beta^{t-i}\boldsymbol{g}_i$
For $t = 1$:

$$\boldsymbol{v}_1 = (1-\beta)\boldsymbol{g}_1$$

Thus is true for $t = 1$.
Considering that this property is true for t, let's proof it is true for $t+1$:

$$\begin{aligned}
\boldsymbol{v}_{t+1} &= \beta\boldsymbol{v}_t + (1-\beta)\boldsymbol{g}_{t+1} \\
&= \beta(1-\beta)\sum_{i=1}^{t} \beta^{t-i}\boldsymbol{g}_i + (1-\beta)\boldsymbol{g}_{t+1} \\
&= (1-\beta)\left(\sum_{i=1}^{t} \beta^{t+1-i}\boldsymbol{g}_i + \beta^{t+1-(t+1)}\boldsymbol{g}_{t+1}\right) \\
&= (1-\beta)\sum_{i=1}^{t+1} \beta^{t+1-i}\boldsymbol{g}_i
\end{aligned}$$

Thus the property is true for $t+1$.
Finally we proved that, $\forall t \geq 1$:

$$\boldsymbol{v}_t = (1-\beta)\sum_{i=1}^{t} \beta^{t-i}\boldsymbol{g}_i$$

.

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 2.3

We have seen in the previous question that : $\boldsymbol{v}_t = (1-\beta)\sum_{i=1}^{t} \beta^{t-i}\boldsymbol{g}_i$, so :

$$\mathbb{E}[\boldsymbol{v}_t] = \mathbb{E}[(1-\beta)\sum_{i=1}^{t} \beta^{t-i}\boldsymbol{g}_i]$$

Using linearity of expectation, we have :

$$\mathbb{E}[\boldsymbol{v}_t] = (1-\beta)\sum_{i=1}^{t}\beta^{t-i}\mathbb{E}[\boldsymbol{g}_i]$$

Using stationarity, we have :

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

Using the sum of finite geometric series, we have :

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

We can observe that the running average is biased $\mathbb{E}[\boldsymbol{v}_t] \neq \mathbb{E}[\boldsymbol{g}_t]$. To eliminate the bias, we can rescale $\boldsymbol{v}_t$ this way :

$$\boldsymbol{v}_t^{new} = \frac{1}{(1-\beta^t)}\boldsymbol{v}_t$$

We can check that $\boldsymbol{v}_t^{new}$ makes the running unbiased :

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

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 3.1

For a single head we have :

$$\text{head}_i = \text{Attention}_{\text{std}}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{softmax}_{\text{row}}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{d_k}}\right)\boldsymbol{V}$$

With :

$$\begin{aligned}
\boldsymbol{Q} &= \bar{\boldsymbol{Q}}\boldsymbol{W}_i^Q \quad \text{where} \quad \bar{\boldsymbol{Q}} \in \mathbb{R}^{n\times d}, \quad \boldsymbol{W}_i^Q \in \mathbb{R}^{d\times d} \quad \text{and} \quad \boldsymbol{Q} \in \mathbb{R}^{n\times d} \\
\boldsymbol{K} &= \bar{\boldsymbol{K}}\boldsymbol{W}_i^K \quad \text{where} \quad \bar{\boldsymbol{K}} \in \mathbb{R}^{n\times d}, \quad \boldsymbol{W}_i^K \in \mathbb{R}^{d\times d} \quad \text{and} \quad \boldsymbol{K} \in \mathbb{R}^{n\times d} \\
\boldsymbol{V} &= \bar{\boldsymbol{V}}\boldsymbol{W}_i^V \quad \text{where} \quad \bar{\boldsymbol{V}} \in \mathbb{R}^{n\times d}, \quad \boldsymbol{W}_i^V \in \mathbb{R}^{d\times d} \quad \text{and} \quad \boldsymbol{V} \in \mathbb{R}^{n\times d}
\end{aligned}$$

So the linear transformation of $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}$ has the same time complexity $\Theta(nd^2)$ and the same space complexity $\Theta(nd + d^2)$. The multiplication $\text{softmax}_{\text{row}}(\boldsymbol{Q}\boldsymbol{K}^\top)\boldsymbol{V}$ has a time complexity $\Theta(n^2d)$ and a space complexity $\Theta(n^2 + nd)$. Finally, by summing we got the time and the space complexity for a single head, respectively equal to $\Theta(n^2d + nd^2)$ and $\Theta(n^2 + d^2 + nd)$.

For multi-head dot-product we have : $\text{Concat}(\text{head}_1, \ldots, \text{head}_h) \in \mathbb{R}^{n\times(h\times d)}$ and $\boldsymbol{W}^O \in \mathbb{R}^{(h\times d)\times d}$, so the multiplication will cost a time complexity $\Theta(nhd^2)$ and a space complexity $\Theta(nd + nhd + hd^2)$. Finally the time and space complexity for multi-head do-product is respectively $\Theta(n^2d + (1+h)nd^2)$ and $\Theta(n^2 + (1+h)(d^2 + nd))$.

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 $\text{softmax}_{\text{row}}(\boldsymbol{Q}\boldsymbol{K}^\top)$. 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 3.2

Let note that, $\forall i, j \in [1, n] \times [1, d]$:

$$\boldsymbol{Q} = (\boldsymbol{q}_{ij})_{1\leq i\leq n, 1\leq j\leq d} \quad \text{and} \quad \boldsymbol{K} = (\boldsymbol{k}_{ij})_{1\leq i\leq n, 1\leq j\leq d}$$

We have :

$$\text{softmax}_{\text{row}}(\boldsymbol{Q}) = \left(\frac{\exp \boldsymbol{q}_{ij}}{\sum_{a=1}^d \exp \boldsymbol{q}_{ia}}\right)_{1\leq i\leq n, 1\leq j\leq d} \qquad \text{softmax}_{\text{col}}(\boldsymbol{K})^\top = \left(\frac{\exp \boldsymbol{k}_{ji}}{\sum_{b=1}^n \exp \boldsymbol{k}_{bi}}\right)_{1\leq j\leq d, 1\leq i\leq n}$$

So the multiplication would look like :

$$\text{softmax}_{\text{row}}(\boldsymbol{Q})\text{softmax}_{\text{col}}(\boldsymbol{K})^\top = \left(\sum_{j=1}^{d} \frac{\exp \boldsymbol{q}_{ij} \exp \boldsymbol{k}_{ji}}{\sum_{a=1}^{d} \exp \boldsymbol{q}_{ia} \sum_{b=1}^{n} \exp \boldsymbol{k}_{bi}}\right)_{1\leq i \leq n}$$

As for every row, we have :

$$\sum_{i=1}^{n}\left(\sum_{j=1}^{d} \frac{\exp \boldsymbol{q}_{ij} \exp \boldsymbol{k}_{ji}}{\sum_{a=1}^{d} \exp \boldsymbol{q}_{ia} \sum_{b=1}^{n} \exp \boldsymbol{k}_{bi}}\right) = 1$$

Thus $\text{softmax}_{\text{row}}(\boldsymbol{Q})\text{softmax}_{\text{col}}(\boldsymbol{K})^\top$ produces valid categorical distributions in every row like $\text{softmax}_{\text{rox}}(\boldsymbol{QK}^\top)$.
Now the good thing in $\text{Attention}_{\text{separable}}$ is that we have the choice to start with the operation we want. For example if we start by calculating the multiplication $\text{softmax}_{\text{col}}(\boldsymbol{K})^\top\boldsymbol{V}$, then we multiply $\text{softmax}_{\text{row}}(\boldsymbol{Q})$ by it, that will cost us a time complexity $\Theta(nd^2)$ and a space complexity of $\Theta(nd + d^2)$.
So if $n \gg d$, the time and space complexity for $\text{Attention}_{\text{separable}}$ will be both equal to $\Theta(n)$, while for $\text{Attention}_{\text{std}}$ the time and space complexity will be both equal to $\Theta(n^2 + n)$. If $n \gg d$, $\text{Attention}_{\text{separable}}$ can be faster and requires less space than $\text{Attention}_{\text{std}}$.

3.3 Verify that the standard attention can be written as

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

where $\boldsymbol{A} = \exp\left(\boldsymbol{QK}^\top\right)$ and $\boldsymbol{D} = \text{diag}(\boldsymbol{A1})$, 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 $\text{Attention}_{\text{approx}}$?
(Hint: Decompose the matrix $\boldsymbol{A}$.)

Answer 3.3

Let's note $\boldsymbol{a}_{ij}$ the element of the matrix $\boldsymbol{A}$, we have :

$$\begin{aligned}\boldsymbol{D} = \text{diag}(\boldsymbol{A1}) &= \text{diag}\left(\begin{bmatrix} \boldsymbol{a}_{11} & \cdots & \boldsymbol{a}_{1n} \\ \vdots & \ddots & \vdots \\ \boldsymbol{a}_{n1} & \cdots & \boldsymbol{a}_{nn}\end{bmatrix}\begin{bmatrix} 1 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 1\end{bmatrix}\right) \\ &= \text{diag}\left(\sum_{i=1}^{n}\boldsymbol{a}_{1i}, \sum_{i=1}^{n}\boldsymbol{a}_{2i}, ..., \sum_{i=1}^{n}\boldsymbol{a}_{ni}\right)\end{aligned}$$

So the inverse of D would be :

$$\boldsymbol{D}^{-1} = \text{diag}\left(\sum_{i=1}^{n}\boldsymbol{a}_{1i}, \sum_{i=1}^{n}\boldsymbol{a}_{2i}, ..., \sum_{i=1}^{n}\boldsymbol{a}_{ni}\right)^{-1} = \text{diag}\left(\frac{1}{\sum_{i=1}^{n}\boldsymbol{a}_{1i}}, \frac{1}{\sum_{i=1}^{n}\boldsymbol{a}_{2i}}, ..., \frac{1}{\sum_{i=1}^{n}\boldsymbol{a}_{ni}}\right)$$

Now let's calculate the multiplication :

$$
\begin{aligned}
\boldsymbol{D}^{-1}\boldsymbol{A} &= \begin{bmatrix} \frac{1}{\sum_{i=1}^{n}\boldsymbol{a}_{1i}} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \frac{1}{\sum_{i=1}^{n}\boldsymbol{a}_{ni}} \end{bmatrix} \begin{bmatrix} \boldsymbol{a}_{11} & \cdots & \boldsymbol{a}_{1n} \\ \vdots & \ddots & \vdots \\ \boldsymbol{a}_{n1} & \cdots & \boldsymbol{a}_{nn} \end{bmatrix} \\
&= \begin{bmatrix} \frac{\boldsymbol{a}_{11}}{\sum_{i=1}^{n}\boldsymbol{a}_{1i}} & \cdots & \frac{\boldsymbol{a}_{1n}}{\sum_{i=1}^{n}\boldsymbol{a}_{ni}} \\ \vdots & \ddots & \vdots \\ \frac{\boldsymbol{a}_{n1}}{\sum_{i=1}^{n}\boldsymbol{a}_{1i}} & \cdots & \frac{\boldsymbol{a}_{nn}}{\sum_{i=1}^{n}\boldsymbol{a}_{ni}} \end{bmatrix} \\
&= \text{sofmax}_{\text{rox}}(\boldsymbol{Q}\boldsymbol{K}^{\top}) \\
\boldsymbol{D}^{-1}\boldsymbol{A}\boldsymbol{V} &= \text{Attention}_{\text{std}}(\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V})
\end{aligned}
$$

Now we have the representation of $\boldsymbol{A}$ as : $\boldsymbol{a}_{ij} = f(\boldsymbol{q}_i)^{\top} f(\boldsymbol{k}_j)$, we can assume that : $\boldsymbol{A} = \boldsymbol{Q}^{'}\boldsymbol{K}^{'\top}$ where $\boldsymbol{Q}^{'}, \boldsymbol{K}^{'} \in \mathbb{R}^{n\times m}$ with rows given respectively as $f(\boldsymbol{q}_i)^{\top}$ and $f(\boldsymbol{k}_j)$, the approximation would be :

$$
\text{Attention}_{\text{Approx}} = \boldsymbol{D}^{-1}(\boldsymbol{Q}^{'}(\boldsymbol{K}^{'\top}\boldsymbol{V}))
$$

As the production matrix $(\boldsymbol{K}^{'\top}\boldsymbol{V}) \in \mathbb{R}^{m\times d}$ and the output of $\text{Attention}_{\text{Approx}} \in \mathbb{R}^{n\times d}$: The time and space complexity of $\text{Attention}_{\text{Approx}}$ will be respectively $\Theta(nmd)$ and $\Theta(nm + md + nd)$.

*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 3.4

Let's first start by decomposing the elements of matrix $\boldsymbol{A}$, we have :

$$
\begin{aligned}
\boldsymbol{a}_{ij} &= \exp(\boldsymbol{q}_i^{\top}\boldsymbol{k}_j) \\
&= \exp(-\frac{1}{2}(\boldsymbol{q}_i^{\top}\boldsymbol{q}_i - (\boldsymbol{q}_i+\boldsymbol{k}_j)^{\top}(\boldsymbol{q}_i+\boldsymbol{k}_j) + \boldsymbol{k}_j^{\top}\boldsymbol{k}_j)) \\
&= \exp\left(\frac{-\|\boldsymbol{q}_i\|^2}{2}\right) \cdot \exp\left(\frac{\|\boldsymbol{q}_i+\boldsymbol{k}_j\|^2}{2}\right) \cdot \exp\left(\frac{-\|\boldsymbol{k}_j\|^2}{2}\right)
\end{aligned}
$$

Now let's show that : $\exp\left(\frac{\|\boldsymbol{q}_i+\boldsymbol{k}_j\|^2}{2}\right) = \mathbb{E}_{\boldsymbol{x}\in\mathcal{N}(\mathbf{0},\mathbf{I})}\left[\exp(\boldsymbol{x}^{\top}\boldsymbol{q}_i)\exp(\boldsymbol{x}^{\top}\boldsymbol{k}_j)\right]$

$$
\begin{aligned}
\exp\left(\frac{\|\boldsymbol{q}_i+\boldsymbol{k}_j\|^2}{2}\right) &= (2\pi)^{-d/2}\exp\left(\frac{\|\boldsymbol{q}_i+\boldsymbol{k}_j\|^2}{2}\right)\int_{\boldsymbol{x}}\exp\left(\frac{-1}{2}\|\boldsymbol{x}-(\boldsymbol{q}_i+\boldsymbol{k}_j)\|^2\right)d\boldsymbol{x} \\
&= (2\pi)^{-d/2}\int_{\boldsymbol{x}}\exp\left(-\frac{\|\boldsymbol{x}\|^2}{2} + \boldsymbol{x}^{\top}(\boldsymbol{q}_i+\boldsymbol{k}_j) - \frac{\|\boldsymbol{q}_i+\boldsymbol{k}_j\|^2}{2} + \frac{\|\boldsymbol{q}_i+\boldsymbol{k}_j\|^2}{2}\right)d\boldsymbol{x} \\
&= (2\pi)^{-d/2}\int_{\boldsymbol{x}}\exp\left(-\frac{\|\boldsymbol{x}\|^2}{2}\right)\exp\left(\boldsymbol{x}^{\top}\boldsymbol{q}_i\right)\exp\left(\boldsymbol{x}^{\top}\boldsymbol{k}_j\right)d\boldsymbol{x} \\
&= \mathbb{E}_{\boldsymbol{x}\in\mathcal{N}(\mathbf{0},\mathbf{I})}\left[\exp(\boldsymbol{x}^{\top}\boldsymbol{q}_i)\exp(\boldsymbol{x}^{\top}\boldsymbol{k}_j)\right]
\end{aligned}
$$

Thus we prove that :

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

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

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 4.1

We have :

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

The gradient would be :

$$\begin{aligned}\nabla_\theta \boldsymbol{L}_{reg}(f(\mathbf{x}^{(i)}, \theta_i), \mathbf{y}^{(i)}) &= \frac{\partial L(f(\mathbf{x}^{(i)}, \theta_i), \mathbf{y}^{(i)})}{\partial\theta_i} + \gamma\frac{\partial}{\partial\theta_i}\|\theta\|_2^2 \\ &= \frac{\partial L(f(\mathbf{x}^{(i)}, \theta_i), \mathbf{y}^{(i)})}{\partial\theta_i} + \gamma\frac{\partial}{\partial\theta_i}\left(\theta^T\theta\right) \\ &= \frac{\partial L(f(\mathbf{x}^{(i)}, \theta), \mathbf{y}^{(i)})}{\partial\theta_i} + 2\gamma\theta_i\end{aligned}$$

The standard SGD update rule would be :

$$\begin{aligned}\theta_{i+1} &= \theta_i - \eta\nabla_\theta \boldsymbol{L}_{reg}(f(\mathbf{x}^{(i)}, \theta_i), \mathbf{y}^{(i)}) \\ &= \theta_i - \eta\frac{\partial L(f(\mathbf{x}^{(i)}, \theta_i), \mathbf{y}^{(i)})}{\partial\theta_i} - 2\eta\gamma\theta_i\end{aligned}$$

As we can see the standard SGD looks identical to the modified SGD for weight decay if we set $\lambda = 2\eta\gamma$.

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 4.2

The line we should modify is the one that compute the gradients :

$$\boldsymbol{h} \longleftarrow \frac{1}{m}\nabla_\theta \sum_i \boldsymbol{L}(f(\mathbf{x}^{(i)},\theta),\mathbf{y}^{(i)})$$

A simple way would be to add $L2$ penalty to the loss, the modified line would be like :

$$\boldsymbol{h} \longleftarrow \frac{1}{m}\nabla_\theta \sum_i \boldsymbol{L}(f(\mathbf{x}^{(i)},\theta),\mathbf{y}^{(i)}) + \lambda\theta$$

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 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 (a)

We have the update rule :

$$\Delta\theta = -\epsilon\frac{\hat{\boldsymbol{s}}_t}{\sqrt{\hat{\boldsymbol{r}}_t}+\delta}$$

With :

$$\begin{aligned}
\hat{\boldsymbol{s}}_t &= \frac{\boldsymbol{s}_t}{1-\rho_1^t} = \frac{\rho_1\boldsymbol{s}_{t-1} + (1-\rho_1)\boldsymbol{h}_t}{1-\rho_1^t} \\
\hat{\boldsymbol{r}}_t &= \frac{\boldsymbol{r}_t}{1-\rho_2^t} = \frac{\rho_2\boldsymbol{r}_{t-1} + (1-\rho_2)\boldsymbol{h}_t^2}{1-\rho_2^t} \\
\boldsymbol{h}_t &= \frac{1}{m}\nabla_\theta \sum_i \boldsymbol{L}(f(\mathbf{x}^{(i)},\theta_t),\mathbf{y}^{(i)}) + \lambda\theta_t
\end{aligned}$$

Let's replace all these equations in the update rule :

$$\begin{aligned}
\theta_{t+1} &= \theta_t - \epsilon\frac{\rho_1\boldsymbol{s}_{t-1} + (1-\rho_1)(\nabla_\theta\boldsymbol{L} + \lambda\theta_t)}{(1-\beta_1^t)\left(\sqrt{\hat{\boldsymbol{r}}_t}+\delta\right)} \\
&= \theta_t - \epsilon\frac{\rho_1\boldsymbol{s}_{t-1} + (1-\rho_1)\nabla_\theta\boldsymbol{L}}{(1-\rho_1^t)\left(\sqrt{\hat{\boldsymbol{r}}_t}+\delta\right)} - \epsilon\frac{(1-\rho_1)(\lambda\theta_t)}{(1-\rho_1^t)\left(\sqrt{\hat{\boldsymbol{r}}_t}+\delta\right)}
\end{aligned}$$

For θ_{large}, we will have a large gradient, and the denominator would be huge because of $\hat{\boldsymbol{r}}_t$ which means the term is closer to zero and thus results in strong regularization.
While for θ_{small}, the gradient are smaller and thus the denominator is smaller and thus results in less regularization.

(b) Would your answer change for the **Adam-weight-decay** scheme ? Why/why not ?
Answer (b)
After all moving averages have been taken care of, if we add the weight decay term, then the decay will not depend on the moving averages. Thus the magnitude of gradients (for different set of parameters) won't affect it and the strength of regularizer would be stabilized for all parameters independently of their gradient magnitude.

(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 4.4

The objective of **L2** regularization is to decay all weights equally. Now talking about adaptive gradient based optimizer, this is not respected in **Adam-L2-reg** scheme : adding the regularization term of the loss, thus will bring in the gradient of regularization into this for all moving averages. And so the strength of the regularization will depend on the magnitude of the gradient.
While in **Adam-weight-decay** scheme : we add the decay to the last update, after that we use this weight decay and restore the same effect as balanced L2 penalty with an equal strength over all parameters. That concludes the argument for why using weight-decay over L2 regularization.