EECS 182 Deep Neural Networks
Fall 2023 Anant Sahai

Homework 9

This homework is due on October 28, at 10:59PM.

1. Self-Supervised Linear Purification

Consider a linear encoder — *square* weight matrix $W \in \mathbb{R}^{m\times m}$ — that we want to be a "purification" operation on $m-$dimensional feature vectors from a particular problem domain. We do this by using self-supervised learning to reconstruct n points of training data $\mathbf{X} \in \mathbb{R}^{m\times n}$ by minimizing the loss:

$$\mathcal{L}_1(W;\mathbf{X}) = \|\mathbf{X} - W\mathbf{X}\|_F^2 \tag{1}$$

While the trivial solution $W = \mathbf{I}$ can minimize the reconstruction loss (1), we will now see how weight-decay (or equivalently in this case, ridge-style regularization) can help us achieve non-trivial purification.

$$\mathcal{L}_2(W;\mathbf{X},\lambda) = \underbrace{\|\mathbf{X} - W\mathbf{X}\|_F^2}_{\text{Reconstruction Loss}} + \lambda \underbrace{\|W\|_F^2}_{\text{Regularization Loss}} \tag{2}$$

Note above that λ controls the relative weighting of the two losses in the optimization.

(a) Consider the simplified case for $m = 2$ with the following two candidate weight matrices:

$$W^{(\alpha)} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad W^{(\beta)} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \tag{3}$$

The training data matrix $\mathbf{X}$ is also given to you as follows:

$$\mathbf{X} = \begin{bmatrix} -2.17 & 1.98 & 2.41 & -2.03 \\ 0.02 & -0.01 & 0.01 & -0.02 \end{bmatrix} \tag{4}$$

i. **Compute the reconstruction loss and the regularization loss for the two encoders, and fill in the missing entries in the table below.**

Encoder	Reconstruction Loss	Regularization Loss
α	________	________
β	0.001	________

ii. **For what values of the regularization parameter λ is the identity matrix $W^{(\alpha)}$ *not* a better solution for the objective $\mathcal{L}_2$ in (2), as compared to $W^{(\beta)}$?**

(b) Now consider a generic square linear encoder $W \in \mathbb{R}^{m\times m}$ and the regularized objective $\mathcal{L}_2$ reproduced below for your convenience:

$$\mathcal{L}_2(W;\mathbf{X},\lambda) = \underbrace{\|\mathbf{X} - W\mathbf{X}\|_F^2}_{\text{Reconstruction Loss}} + \lambda \underbrace{\|W\|_F^2}_{\text{Regularization Loss}}$$

Assume $\sigma_1 > \cdots > \sigma_m \geq 0$ are the m singular values in $\mathbf{X}$, that the number of training points n is larger than the number of features m, and that $\mathbf{X}$ can be expressed in SVD coordinates as $\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^\top$.

i. You are given that the optimizing weight matrix for the regularized objective $\mathcal{L}_2$ above takes the following form. **Fill in the empty matrices below**.

$$\widehat{W} = \begin{bmatrix} & & \\ & & \\ & & \end{bmatrix} \cdot \begin{bmatrix} \frac{\sigma_1^2}{\sigma_1^2+\lambda} & & & \\ & \frac{\sigma_2^2}{\sigma_2^2+\lambda} & & \\ & & \ddots & \\ & & & \frac{\sigma_m^2}{\sigma_m^2+\lambda} \end{bmatrix} \cdot \begin{bmatrix} & & \\ & & \\ & & \end{bmatrix} \tag{5}$$

ii. **Derive the above expression.**
(Hint: Can you understand $\mathcal{L}_2(W; \mathbf{X}, \lambda)$ as a sum of m completely decoupled ridge-regression problems?)

(c) You are given that the data matrix $\mathbf{X} \in \mathbb{R}^{8\times n}$ has the following singular values:

$$\{\sigma_i\} = \{10, 8, 4, 1, 0.5, 0.36, 0.16, 0.01\}$$

For what set of hyperparameter values λ can we guarantee that the learned purifier $\widehat{W}$ will preserve at least 80% of the feature directions corresponding to the first 3 singular vectors of X, while attenuating components in the remaining directions to at most 50% of their original strength?
(Hint: What are the two critical singular values to focus on?)

2. Hand-Design Transformers

Please follow the instructions in this notebook. You will implement a simple transformer model (with a single attention head) from scratch and then create a hand-designed the attention heads of the transformer model capable of solving a basic problem. Once you finished with the notebook,

- Download `submission_log.json` and submit it to "Homework 8 (Code)" in Gradescope.
- Answer the following questions in your submission of the written assignment:

(a) Design a transformer that selects by contents. Compare the variables of your hand-designed Transformer with those of the learned Transformer. **Identify the similarities and differences between the two sets of variables and provide a brief explanation for each difference.**

(b) Design a transformer that selects by positions. Compare the variables of your hand-designed Transformer with those of the learned Transformer. **Identify the similarities and differences between the two sets of variables and provide a brief explanation for each difference.**

3. Argmax Attention

Recall from lecture that we can think about attention as being *queryable softmax pooling*. In this problem, we ask you to consider a hypothetical argmax version of attention where it returns exactly the value corresponding to the key that is most similar to the query, where similarity is measured using the traditional inner-product.

(a) Perform **argmax attention** with the following keys and values:

Keys:

$$\left\{ \begin{bmatrix} 1 \\ 2 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 3 \\ 4 \end{bmatrix}, \begin{bmatrix} 5 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right\}$$

Corresponding Values:

$$\left\{ \begin{bmatrix} 2 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 4 \\ 3 \end{bmatrix}, \begin{bmatrix} 0 \\ -1 \\ 4 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} \right\}$$

using the following query:

$$\mathbf{q} = \begin{bmatrix} 1 \\ 1 \\ 2 \end{bmatrix}$$

What would be the output of the attention layer for this query?

Hint: For example, $\text{argmax}([1, 3, 2]) = [0, 1, 0]$

(b) Note that instead of using *softmax* we used `argmax` to generate outputs from the attention layer. **How does this design choice affect our ability to usefully train models involving attention?**

(Hint: think about how the gradients flow through the network in the backward pass. Can we learn to improve our queries or keys during the training process?)

4. Justifying Scaled-Dot Product Attention

Suppose $q, k \in \mathbb{R}^d$ are two random vectors with $q, k\ N(\mu, \sigma^2 I)$, where $\mu \in \mathbb{R}^d$ and $\sigma \in \mathbb{R}^+$. In other words, each component q_i of q is drawn from a normal distribution with mean μ and stand deviation σ, and the same if true for k.

(a) Define $\mathbb{E}[q^T k]$ in terms of μ, σ and d.

(b) Considering a practical case where $\mu = 0$ and $\sigma = 1$, define $\text{Var}(q^T k)$ in terms of d.

(c) Continue to use the setting in part (b), where $\mu = 0$ and $\sigma = 1$. Let s be the scaling factor on the dot product. Suppose we want $\mathbb{E}[\frac{q^T k}{s}]$ to be 0, and $\text{Var}(\frac{q^T k}{s})$ to be $\sigma = 1$. What should s be in terms of d?

5. Kernelized Linear Attention

The softmax attention is widely adopted in transformers, however the $\mathcal{O}\left(N^2\right)$ (N stands for the sequence length) complexity in memory and computation often makes it less desirable for processing long document like a book or a passage, where the N could be beyond thousands. There is a large body of the research studying how to resolve this [1].

Under this context, this question presents a formulation of attention via the lens of the kernel. A large portion of the context is adopted from [2]. In particular, attention can be seen as applying a kernel over the inputs

[1] `https://huggingface.co/blog/long-range-transformers`

[2] Tsai, Yao-Hung Hubert, et al. "Transformer dissection: a unified understanding of transformer's attention via the lens of kernel" (2019).

with the kernel scores being the similarities between inputs. This formulation sheds light on individual components of the transformer's attention, and helps introduce some alternative attention mechanisms that replaces the "softmax" with linearized kernel functions, thus reducing the $\mathcal{O}\left(N^2\right)$ complexity in memory and computation.

We first review the building block in the transformer. Let $x \in \mathbb{R}^{N\times F}$ denote a sequence of N feature vectors of dimensions F. A transformer[3] is a function $T : \mathbb{R}^{N\times F} \to \mathbb{R}^{N\times F}$ defined by the composition of L transformer layers $T_1(\cdot), \ldots, T_L(\cdot)$ as follows,

$$T_l(x) = f_l(A_l(x) + x). \tag{6}$$

The function $f_l(\cdot)$ transforms each feature independently of the others and is usually implemented with a small two-layer feedforward network. $A_l(\cdot)$ is the self attention function and is the only part of the transformer that acts across sequences.

We now focus on the the self attention module which involves softmax. The self attention function $A_l(\cdot)$ computes, for every position, a weighted average of the feature representations of all other positions with a weight proportional to a similarity score between the representations. Formally, the input sequence x is projected by three matrices $W_Q \in \mathbb{R}^{F\times D}$, $W_K \in \mathbb{R}^{F\times D}$ and $W_V \in \mathbb{R}^{F\times M}$ to corresponding representations Q, K and V. The output for all positions, $A_l(x) = V'$, is computed as follows,

$$\begin{aligned} Q = xW_Q, K = xW_K, V = xW_V, \\ A_l(x) = V' = \text{softmax}(\frac{QK^T}{\sqrt{D}})V. \end{aligned} \tag{7}$$

Note that in the previous equation, the softmax function is applied rowwise to QK^T. Following common terminology, the Q, K and V are referred to as the "queries", "keys" and "values" respectively.

Equation 7 implements a specific form of self-attention called softmax attention where the similarity score is the exponential of the dot product between a query and a key. Given that subscripting a matrix with i returns the i-th row as a vector, we can write a generalized attention equation for any similarity function as follows,

$$V_i' = \frac{\sum_{j=1}^N \text{sim}\left(Q_i, K_j\right) V_j}{\sum_{j=1}^N \text{sim}\left(Q_i, K_j\right)}. \tag{8}$$

Equation 11 is equivalent to equation 7 if we substitute the similarity function with $\text{sim}_{\text{softmax}}(q, k) = \exp(\frac{q^T k}{\sqrt{D}})$. This can lead to

$$V_i' = \frac{\sum_{j=1}^N \exp(\frac{Q_i^T K_j}{\sqrt{D}})V_j}{\sum_{j=1}^N \exp(\frac{Q_i^T K_j}{\sqrt{D}})}. \tag{9}$$

For computing the resulting self-attended feature $A_l(x) = V'$, we need to compute all V_i' $i \in 1, ..., N$ in equation 13.

(a) **Identify the conditions that needs to be met by the sim function to ensure that V_i in Equation 11 remains finite (the denominator never reaches zero).**

(b) The definition of attention in equation 11 is generic and can be used to define several other attention implementations.

[3] Vaswani, Ashish, et al. "Attention is all you need" (2017).

(i) One potential attention variant is the "polynomial kernel attention", where the similarity function as $\text{sim}(q, k)$ is measured by polynomial kernel $\mathcal{K}$ [4]. **Considering a special case for a "quadratic kernel attention" that the degree of "polynomial kernel attention" is set to be 2, derive the $\text{sim}(q, k)$ for "quadratic kernel attention". (NOTE: any constant factor is set to be 1.)** .

(ii) One benefit of using kernelized attention is that we can represent a kernel using a feature map $\phi(\cdot)$ [5]. **Derive the corresponding feature map $\phi(\cdot)$ for the quadratic kernel.**

(iii) **Considering a general kernel attention, where the kernel can be represented using feature map that $\mathcal{K}(q, k) = (\phi(q)^T \phi(k))$, rewrite kernel attention of equation 11 with feature map $\phi(\cdot)$.**

(c) We can rewrite the softmax attention in terms of equation 11 as equation 13. **For all the V_i' ($i \in \{1, ..., N\}$), derive the time complexity (asymptotic computational cost) and space complexity (asymptotic memory requirement) of the above softmax attention in terms of sequence length N, D and M.**

NOTE: for memory requirement, we need to store any intermediate results for backpropagation, including all Q, K, V

(d) Assume we have a kernel $\mathcal{K}$ as the similarity function and the kernel can be represented with a feature map $\phi(\cdot)$, we can rewrite equation 11 with $\text{sim}(x, y) = \mathcal{K}(x, y) = (\phi(Q_i)^T \phi(K_j))$ in part . We can then further simplify it by making use of the associative property of matrix multiplication to

$$V_i' = \frac{\phi(Q_i)^T \sum_{j=1}^N \phi(K_j) V_j^T}{\phi(Q_i)^T \sum_{j=1}^N \phi(K_j)}. \tag{10}$$

Note that the feature map $\phi(\cdot)$ is applied row-wise to the matrices Q and K.

Considering using a linearized polynomial kernel $\phi(x)$ of degree 2, and assume $M \approx D$, derive the computational cost and memory requirement of this kernel attention as in (12).

6. Kernelized Linear Attention (Part II)

This is a continuation of "Kernelized Linear Attention" in HW 7. Please refer to this part for notation and context. In Part I of this problem, we considered ways to efficiently express the attention operation when sequences are long (e.g. a long document). Attention uses the following equation:

$$V_i' = \frac{\sum_{j=1}^N \text{sim}(Q_i, K_j) V_j}{\sum_{j=1}^N \text{sim}(Q_i, K_j)}. \tag{11}$$

We saw that when the similarity function is a kernel function (i.e. if we can write $\text{sim}(Q_i, K_j) = \Phi(Q_i)^T \Phi(K_j)$ for some function Φ), then we can use the associative property of matrix multiplication to simplify the formula to

$$V_i' = \frac{\phi(Q_i)^T \sum_{j=1}^N \phi(K_j) V_j^T}{\phi(Q_i)^T \sum_{j=1}^N \phi(K_j)}. \tag{12}$$

[4] https://en.wikipedia.org/wiki/Polynomial_kernel

[5] https://en.wikipedia.org/wiki/Kernel_method

If we use a polynomial kernel with degree 2, this gives a computational cost of $\mathcal{O}\left(ND^2M\right)$, which for very large N is favorable to the softmax attention computational cost of $\mathcal{O}\left(N^2 \max\left(D, M\right)\right)$. ($N$ is the sequence length, D is the feature dimension of the queries and keys, and M is the feature dimension of the values. Now, we will see whether we can use kernel attention to directly approximate the softmax attention:

$$V_i' = \frac{\sum_{j=1}^N \exp(\frac{Q_i^T K_j}{\sqrt{D}}) V_j}{\sum_{j=1}^N \exp(\frac{Q_i^T K_j}{\sqrt{D}})}. \tag{13}$$

(a) Approximating softmax attention with linearized kernel attention

i. As a first step, we can use Gaussian Kernel $\mathcal{K}_{\text{Gauss}}(q, k) = \exp(\frac{-||q-k||_2^2}{2\sigma^2})$ to rewrite the softmax similarity function, where $\text{sim}_{\text{softmax}}(q, k) = \exp(\frac{q^T k}{\sqrt{D}})$. Assuming we can have $\sigma^2 = \sqrt{D}$, **rewrite the softmax similarity function using Gaussian Kernel.**. (*Hint: You can write the softmax* $\exp(\frac{-||q-k||_2^2}{2\sigma^2})$ *as the product of the Gaussian Kernel and two other terms.)*

ii. However, writing softmax attention using a Gaussian kernel does not directly enjoy the benefits of the reduced complexity using the feature map. This is because the feature map of Guassian kernel usually comes from the Taylor expression of $\exp(\cdot)$, whose computation is still expensive [6]. However, we can approximate the Guassian kernel using random feature map and then reduce the computation cost. (Rahimi and Recht, 2007)[7] proposed random Fourier features to approximate a desired shift-invariant kernel. The method nonlinearly transforms a pair of vectors q and k using a **random feature map** $\phi_{\text{random}}()$; the inner product between $\phi\left(q\right)$ and $\phi\left(k\right)$ approximates the kernel evaluation on q and k. More precisely:

$$\phi_{\text{random}}(q) = \sqrt{\frac{1}{D_{\text{random}}}} \left[\sin\left(\mathbf{w}_1 q\right), \ldots, \sin\left(\mathbf{w}_{D_{\text{random}}} q\right), \cos\left(\mathbf{w}_1 q\right), \ldots, \cos\left(\mathbf{w}_{D_{\text{random}}} q\right)\right]^\top. \tag{14}$$

Where we have D_{random} of D-dimensional random vectors $\mathbf{w}_i$ independently sampled from $\mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I}_D)$,

$$\mathbb{E}_{\mathbf{w}_i}\left[\phi\left(q\right) \cdot \phi\left(k\right)\right] = \exp\left(-\|q-k\|^2 / 2\sigma^2\right). \tag{15}$$

Use ϕ_{random} to approximate the above softmax similarity function with Gaussian Kernel and derive the computation cost for computing all the V' here.

(b) (Optional) Beyond self-attention, an autoregressive case will be masking the attention computation such that the i-th position can only be influenced by a position j if and only if $j \leq i$, namely a position cannot be influenced by the subsequent positions. This type of attention masking is called *causal masking*.

Derive how this causal masking changes equation 11 and 12. **Write equation 11 and 12 in terms of S_i and Z_i**, which are defined as:

$$S_i = \sum_{j=1}^{i} \phi\left(K_j\right) V_j^T, \; Z_i = \sum_{j=1}^{i} \phi\left(K_j\right), \tag{16}$$

[6] https://www.csie.ntu.edu.tw/~cjlin/talks/kuleuven_svm.pdf
[7] https://people.eecs.berkeley.edu/~brecht/papers/07.rah.rec.nips.pdf

to simplify the causal masking kernel attention and derive the computational complexity of this new causal masking formulation scheme.

7. Read a Blog Post: How to train your Resnet

In previous homeworks, we saw how memory and compute constraints on GPUs put limits on the architecture and the hyperparameters (e.g., batch size) we can use to train our models. To train better models, we could scale up by using multiple GPUs, but most distributed training techniques scale sub-linearly and often we simply don't have as many GPU resources at our disposal. This raises a natural question - how can we make model training more efficient on a single GPU?

The blog series How to train your Resnet (https://myrtle.ai/learn/how-to-train-your-resnet/) explores how to train ResNet models efficiently on a single GPU. It covers a range of topics, including architecture, weight decay, batch normalization, and hyperparameter tuning. In doing so, it provides valuable insights into the training dynamics of neural networks and offers lessons that can be applied in other settings.

Read the blog series and answer the questions below.

(a) **What is the baseline training time and accuracy the authors started with? What was the final training time and accuracy achieved by the authors?**

(b) **Comment on what you have learnt.** ($\approx$ 100 words)

(c) **Which approach taken by the authors interested you the most? Why?** ($\approx$ 100 words)

8. Homework Process and Study Group

Citing sources and collaborators are an important part of life, including being a student!
We also want to understand what resources you find helpful and how much time homework is taking, so we can change things in the future if possible.

(a) **What sources (if any) did you use as you worked through the homework?**

(b) **If you worked with someone on this homework, who did you work with?**
List names and student ID's. (In case of homework party, you can also just describe the group.)

(c) **Roughly how many total hours did you work on this homework?**

Contributors:

- Dhruv Shah.
- Anant Sahai.
- Olivia Watkins.
- Jake Austin.
- Linyuan Gong.
- Saagar Sanghavi.
- Sheng Shen.
- David M. Chan.

- Shaojie Bai.
- Angelos Katharopoulos.
- Hao Peng.
- Romil Bhardwaj.