HW7

Mengying Lin SID: 3038737132.

(a) Designing AutoEncoders

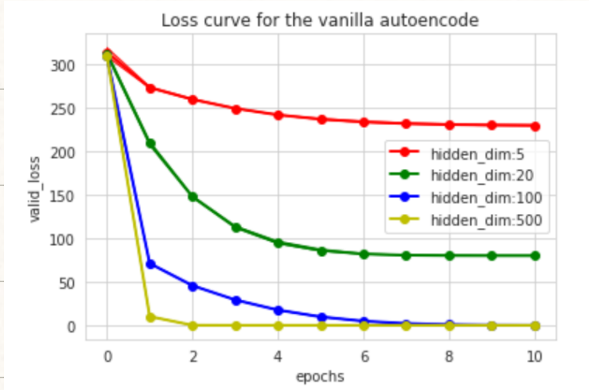
Please follow the instructions in this notebook. You will train autoencoders, denoising autoencoders, and masked autoencoders on a synthetic dataset and the MNIST dataset. Once you finished with the notebook,

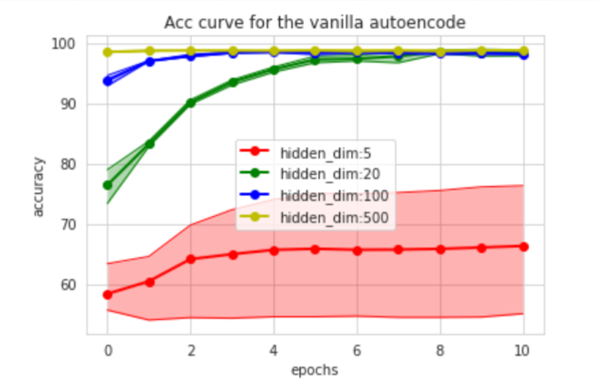
- Download submission_log.json and submit it to "Homework 7 (Code)" in Gradescope.
- Answer the following questions in your submission of the written assignment:

(i) **Show your visualization** of the vanilla autoencoder with different latent representation sizes.

(ii) Based on your previous visualizations, answer this question: **How does changing the latent representation size of the autoencoder affect the model's performance in terms of reconstruction accuracy and linear probe accuracy? Why?**

i)





ii). The larger the size, the faster the model converges to a relatively higher accuracy, because deeper models are more likely to extract useful features.

(b) PCA & AutoEncoders

In the case where the encoder f_θ, g_ϕ are linear functions, the model is termed as a *linear autoencoder*. In particular, assume that we have data $x_i \in \mathbb{R}^m$ and consider two weight matrices: an encoder $W_1 \in \mathbb{R}^{k\times m}$ and decoder $W_2 \in \mathbb{R}^{m\times k}$ (with $k < m$). Then, a linear autoencoder learns a low-dimensional embedding of the data $\mathbf{X} \in \mathbb{R}^{m\times n}$ (which we assume is zero-centered without loss of generality) by minimizing the objective,

$$\mathcal{L}(W_1, W_2; \mathbf{X}) = \frac{1}{n}||\mathbf{X} - W_2 W_1 \mathbf{X}||_F^2 \tag{1}$$

We will assume $\sigma_1^2 > \cdots > \sigma_k^2 > 0$ are the k largest eigenvalues of $\frac{1}{n}\mathbf{X}\mathbf{X}^\top$. The assumption that the $\sigma_1, \dots, \sigma_k$ are positive and distinct ensures identifiability of the principal components, and is common in this setting. Therefore, the top-k eigenvalues of $\mathbf{X}$ are $S = \text{diag}(\sigma_1, \dots, \sigma_\text{k})$, with corresponding eigenvectors are the columns of $\mathbf{U}_k \in \mathbb{R}^{m\times k}$. A well-established result from (Baldi & Hornik, 1989) shows that principal components are the unique optimal solution to linear autoencoders (up to sign changes to the projection directions). In this subpart, we take some steps towards proving this result.

(i) **Write out the first order optimality conditions that the minima of Eq. 1 would satisfy.**

(ii) **Show that the principal components $\mathbf{U}_k$ satisfy the optimality conditions outlined in (i).**

i). Let $W = W_2 W_1$.

$$\mathcal{L}(W, X) = \frac{1}{n}(X - WX)^T(X - WX)$$

$$= \frac{1}{n}(X^T - X^T W^T)(X - WX)$$

$$= \frac{1}{n}(X^T X - X^T W^T X - X^T W X + X^T W^T W X)$$

$$\frac{\partial \mathcal{L}}{\partial W} = \frac{1}{n}(-2XX^T + 2WXX^T) = -\frac{2}{n}(I - W)XX^T.$$

$$\frac{\partial \mathcal{L}}{\partial W_1} = \frac{\partial \mathcal{L}}{\partial W} \cdot \frac{\partial W}{\partial W_1} = -\frac{2}{n}(I - W)XX^T W_2.$$

$$\frac{\partial \mathcal{L}}{\partial W_2} = \frac{\partial \mathcal{L}}{\partial W} \cdot \frac{\partial W}{\partial W_2} = -\frac{2}{n}(I - W)XX^T W_1^T.$$

$$\begin{cases} (I - W_2 W_1) XX^T W_2 = 0 \\ (I - W_2 W_1) XX^T W_1^T = 0. \end{cases}$$

ii). Let $\frac{1}{n}XX^T = USV^T$.

2. Self-supervised Linear Autoencoders

We consider linear models consisting of two weight matrices: an encoder $W_1 \in \mathbb{R}^{k\times m}$ and decoder $W_2 \in \mathbb{R}^{m\times k}$ (assume $1 < k < m$). The traditional autoencoder model learns a low-dimensional embedding of the n points of training data $\mathbf{X} \in \mathbb{R}^{m\times n}$ by minimizing the objective,

$$\mathcal{L}(W_1, W_2; \mathbf{X}) = \frac{1}{n}||\mathbf{X} - W_2 W_1 \mathbf{X}||_F^2 \tag{2}$$

We will assume $\sigma_1^2 > \cdots > \sigma_k^2 > \sigma_{k+1}^2 \geq 0$ are the $k+1$ largest eigenvalues of $\frac{1}{n}\mathbf{X}\mathbf{X}^\top$. The assumption that the $\sigma_1, \ldots, \sigma_k$ are positive and distinct ensures identifiability of the principal components.

Consider an ℓ_2-regularized linear autoencoder where the objective is:

$$\mathcal{L}_\lambda(W_1, W_2; \mathbf{X}) = \tfrac{1}{n}||\mathbf{X} - W_2 W_1 \mathbf{X}||_F^2 + \lambda\|W_1\|_F^2 + \lambda\|W_2\|_F^2. \tag{3}$$

where $\|\cdot\|_F^2$ represents the Frobenius norm squared of the matrix (i.e. sum of squares of the entries).

(a) You want to use SGD-style training in PyTorch (involving the training points one at a time) and self-supervision to find W_1 and W_2 which optimize (3) by treating the problem as a neural net being trained in a supervised fashion. **Answer the following questions and briefly explain your choice:**

(i) **How many linear layers do you need?**

- ☐ 0
- ☐ 1
- ☑ 2
- ☐ 3

(ii) **What is the loss function that you will be using?**

- ☐ `nn.L1Loss`
- ☑ `nn.MSELoss`
- ☐ `nn.CrossEntropyLoss`

~~we need to estimate how close x and $\hat{x}$ are, so o with the loss containi'~~

i) Need to learn both W_2 and W_1

ii) Need to estimate how close x and $\hat{x}$ are, so we'd go with the loss containing the info of distance.

L1-Loss is not preferred because it's not differentiable in some pts.

(iii) **Which of the following would you need to optimize (3) exactly as it is written? (Select all that are needed)**

- [x] Weight Decay
- [ ] Dropout
- [ ] Layer Norm
- [x] Batch Norm
- [ ] SGD optimizer

(b) **Do you think that the solution to (3) when we use a small nonzero λ has an inductive bias towards finding a W_2 matrix with approximately orthonormal columns? Argue why or why not?**

(Hint: Think about the SVDs of $W_1 = U_1\Sigma_1V_1^\top$ and $W_2 = U_2\Sigma_2V_2^\top$. You can assume that if a $k \times m$ or $m \times k$ matrix has all k of its nonzero singular values being 1, then it must have orthonormal rows or columns. Remember that the Frobenius norm squared of a matrix is just the sum of the squares of its singular values. Further think about the minimizer of $\frac{1}{\sigma^2} + \sigma^2$. Is it unique?)

Yes. Note $\|W_2\| = \sum_i \sigma_i^2$, where σs are the singular values. By penalizing $\|W_2\|_F^2$, we are actually assuming the singular values of W_2 are more likely to be close to 0.

3. 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?

a) $E[q_e^T k] = E[\sum q_{ei} k_i] = \sum_{i=1}^{d} E[q_{ei} k_i]$

Assume q_{ei} and k_i are independent.

$E[q_e^T k] = \sum_{i=1}^{d} E[q_{ei}] E[k_i] = \sum_{i=1}^{d} \mu_i^2 = \|\mu\|_2^2$

b) $\text{Var}(q_e^T k) = E[(q_e^T k)^2] - (E[q_e^T k])^2$

$= E[k^T q_e q_e^T k] - \|\mu\|_2^4$

$= E[k^T E[q_e q_e^T] k] - \|\mu\|_2^4.$

$\text{Var}(q_e) = E[q_e q_e^T] - E[q_e] E[q_e]^T$

Hence $E[q_e q_e^T] = \text{Var}(q_e) + E[q_e] E[q_e]^T = \sigma^2 I + \mu\mu^T.$

$\text{Var}(q_e^T k) = E[k^T(\sigma^2 I + \mu\mu^T) k] - \|\mu\|_2^4.$

$= \sigma^2 E[k^T k] + E[\mu^T k]^2 - \|\mu\|_2^4.$

$= \sigma^2 E[\sum_{i=1}^{d} k_i^2] + \|\mu\|_2^4 - \|\mu\|_2^4.$

$= \sigma^2 (d\sigma^2)$

$= d\sigma^4. = d.$

c) $\begin{cases} E[\frac{q_e^T k}{s}] = \frac{1}{s} E[q_e^T k] = \frac{\|\mu\|_2^2}{s} = 0 \\ \text{Var}[\frac{q_e^T k}{s}] = \frac{1}{s^2} \text{Var}[q_e^T k] = \frac{d}{s^2} = 1. \end{cases}$ $s = \sqrt{d}.$

4. 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, $\text{argmax}([1, 3, 2]) = [0, 1, 0]$

$<k_1^T, q> = 1\times1 + 2\times1 + 0\times2 = 3.$

$<k_2^T, q> = 1\times1 + 4\times1 + 3\times2 = 11$

$<k_3^T, q> = 5\times1 = 5$

$<k_4^T, q> = 0\times1 + 0\times1 + 1\times2 = 2.$

$V \cdot \text{argmax}([3, 11, 5, 2])^T = \begin{bmatrix} 1 \\ 4 \\ 3 \end{bmatrix}.$

The output will be $\begin{bmatrix} 1 \\ 4 \\ 3 \end{bmatrix}$

(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?)

The gradient of softmax is between 0 and 1. and might vanish during backprop. However, the norm of the gradient of argmax is 1, and might better prevent dead gradient.

5. Kernelized Linear Attention

The softmax attention is widely adopted in transformers (Luong et al., 2015; Vaswani et al., 2017), however the $\mathcal{O}(N^2)$ (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 Tsai et al. (2019). In particular, attention can be seen as applying a kernel over the inputs 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 $\mathcal{O}(N^2)$ 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 Vaswani et al. (2017) 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,

$$T_l(x) = f_l(A_l(x) + x). \tag{4}$$

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,

$$\begin{aligned} Q = \overset{N\times D}{xW_Q}, K = xW_K, V = xW_V, \\ A_l(x) = V' = \text{softmax}\left(\frac{QK^T}{\sqrt{D}}\right)V. \end{aligned} \tag{5}$$

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 5 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,

$$V_i' = \frac{\sum_{j=1}^N \text{sim}(Q_i, K_j)\, V_j}{\sum_{j=1}^N \text{sim}(Q_i, K_j)}. \tag{6}$$

Equation 6 is equivalent to equation 5 if we substitute the similarity function with $\text{sim}_{\text{softmax}}(q, k) = \exp(\frac{q^T k}{\sqrt{D}})$. This can lead to

$$V_i' = \frac{\sum_{j=1}^N \exp(\frac{Q_i^T K_j}{\sqrt{D}}) V_j}{\sum_{j=1}^N \exp(\frac{Q_i^T K_j}{\sqrt{D}})}. \tag{7}$$

For computing the resulting self-attended feature $A_l(x) = V'$, we need to compute all V_i' $i \in 1, \ldots, N$ in equation 7.

(a) **Identify the conditions that needs to be met by the sim function to ensure that V_i in Equation 6 remains finite (the denominator never reaches zero).**

The sim fn should be positive all the time.

(b) The definition of attention in equation 6 is generic and can be used to define several other attention implementations.

(i) One potential attention variant is the "polynomial kernel attention", where the similarity function as $\mathrm{sim}(q, k)$ is measured by polynomial kernel $\mathcal{K}$ [2]. **Considering a special case for a "quadratic kernel attention" that the degree of "polynomial kernel attention" is set to be 2, derive the $\mathrm{sim}(q, k)$ 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 $\phi(\cdot)$ [3]. **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 6 with feature map $\phi(\cdot)$.**

i). $\mathrm{sin}(q_e, k) = (q_e^T k + 1)^2$.

ii) $\phi(x) = [x_1 \cdots x_d, x_1^2, x_1x_2, x_1x_3 \cdots x_d^2]^T$.

iii). $$V_i' = \frac{\sum_j \phi(Q_i)^T \phi(Q_j) V_j}{\sum_j \phi(Q_i)\phi(Q_j)}$$

(c) We can rewrite the softmax attention in terms of equation 6 as equation 7. **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

$$V_i' = \frac{\sum_{j=1}^N \exp(\frac{\overset{1\times D}{Q_i^T} \overset{D\times 1}{K_j}}{\sqrt{D}}) V_j}{\sum_{j=1}^N \exp(\frac{Q_i^T K_j}{\sqrt{D}})}. \quad M\times 1.$$

Time Complexity:

Computing $Q_i^T K_j$ needs $O(D)$.

Computing $\sum e^{\frac{Q_i^T K_j}{\sqrt{D}}}$ needs $O(N \cdot D)$, $\sum_{j=1}^N e^{\frac{Q_i^T K_j}{\sqrt{D}}} V_j$ needs $O(NMD)$.

We need to compute all V_i', $i \in [0, \cdots N]$.

Hence time complexity is $O(N^2M)$.

We need store $Q_i^T K_j$, $e^{\frac{Q_i^T K_j}{\sqrt{D}}}$, $e^{\frac{Q_i^T K_j}{\sqrt{D}}} V_j$ in the process.

which occupies $O(N^2 \times 1) + O(N^2 \times 1) + O(N^2 \times M) = O(N^2 M)$ space.

(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 6 with $\operatorname{sim}(x, y) = \mathcal{K}(x, y) = (\phi(Q_i)^T \phi(K_j))$ in part (b). We can then further simplify it by making use of the associative property of matrix multiplication to

$$V_i' = \frac{\phi(Q_i)^T \sum_{j=1}^N \phi(K_j) V_j^T}{\phi(Q_i)^T \sum_{j=1}^N \phi(K_j)}. \tag{8}$$

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 (8).

In terms of computing time,

$\phi(K_j)$: $O(D)$.

$\phi(K_j) V_j^T$: $O(DM)$.

$\sum_j \phi(K_j) V_j^T$: $O(NDM)$.

$\phi(Q_i)^T \sum_j \phi(K_j) V_j^T$: $O(ND^2M)$.

In terms of memory,

$\phi(K_j) \times N$: $O(D^2 N)$

$\phi(K_j) V_j^T \times N$: $O(D^3 NM)$

$\phi(Q_i) \times N$: $O(D^2 N)$

Hence space complexity is $O(D^3 NM)$.

7. 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? Write it down here where you'll need to remember it for the self-grade form.**

a) None.

b) Name : Xiang Fei

SID : 3038733024.

c) 10h.