EECS 182 Deep Neural Networks
Fall 2025 Anant Sahai and Gireeja Ranade

Homework 8

This homework is due on October 31, at 10:59PM.

1. SSM Convolution Kernel

Background and Setup: Consider a discrete-time State-Space Model (SSM) of the form

$$x_{k+1} = Ax_k + Bu_k,$$
$$y_k = Cx_k + Du_k,$$

(a) **Convolution Kernel and the Output Equation.** Given that the sequence length is L (input: $(u_0, \cdots, u_L)$, output: $(y_0, \cdots, y_L)$) and assume $x_0 = 0$, **show** that the output y_k can be expressed as a *convolution* of the input sequence $\{u_\ell\}_0^L$ with a kernel $K = \{K_\ell\}_0^L$:

$$y_k = \sum_{\ell=0}^{L} K_\ell u_{k-\ell},$$

where any $u_{\leq 0}$ with a negative index is set to 0 (zero-padding). Also, **find** K.

(b) **Concrete Examples.**

i. *Scalar Case:* Let $n = 1$, and set $A = \alpha$, $B = \beta$, $C = \gamma$, $D = \delta$. Use $\alpha = 0.8, \beta = 1, \gamma = 1.5$ and compute the kernel up to $L = 4$.

ii. *2D Case:* Let $A \in \mathbb{R}^{2\times 2}$ be, for instance,

$$A = \begin{pmatrix} 0.7 & 0.1 \\ 0.2 & 0.6 \end{pmatrix}, \quad B = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad C = \begin{pmatrix} 1 & 0 \end{pmatrix}, \quad D = 0$$

Compute kernels up to $L = 3$ and briefly discuss how the kernel captures the "impulse response".

(c) **Efficient Computation with Convolutions.** If we already know the kernel K, how much can we parallelize the computation of the output sequence $\{y_k\}$ for an input sequence $\{u_k\} \in \mathbb{R}^d$ of length L? What is the minimum critical path length of the computation? What about a naive, direct computation of y_k from the unrolled recursion?

(d) **Efficient Kernel Computation.** Given A, B, C, how can we compute the kernel, K, efficiently? What are some strategies to parallelize kernel computation? You may assume $L = 2^N$ for some N for simplicity.

(e) **Adding structure to A.** Suppose A is a diagonal matrix. How can we leverage this structure to compute the kernel K more efficiently?

(f) **Diagonal-plus-low-rank (DPLR) structure** Now if A has the following form:

$$A = I_n + pp^\top,$$

where $A \in \mathbb{R}^{n\times n}, p \in \mathbb{R}^n$. How can we leverage this structure to compute the kernel K more efficiently?

2. Coding SSM Forward

You showed that SSMs forward pass can be implemented using recurrent and convolution approaches in the previous problem. In this problem, we will implement the two approaches and compare their efficiency. Particularly, you are given the following update equation:

$$\mathbf{h_{t+1}} = (W \cdot \mathbf{h_t} + U \cdot \mathbf{x}_t + \mathbf{b})$$

You are given all the input $\mathbf{x}_t$ for $t = 1, 2, \ldots, T$ and the initial hidden state $\mathbf{h}_0 = 0$. You need to implement the forward pass for the SSM computing all the hidden states $\mathbf{h}_t$ for $t = 1, 2, \ldots, T$.

We will complete the notebooks q_coding_ssm_forward_cpu.ipynb and q_coding_ssm_forward_gpu.ipynb on the CPU and GPU respectively.

CPU Implementation:

(a) Implement the forward pass by unrolling the SSM in time in a recurrent manner.

(b) Implement the forward pass by using a convolution-based approach. You will create the kernel K and use `nn.Conv1d` to apply the kernel to the input x. Hint: You can create the kernel using a divide-and-conquer method.

(c) Compare the runtime of the two approaches and explain your findings.

GPU Implementation:

(d) Now implement the forward pass on the GPU. You can copy your existing implementation to the GPU notebook.

(e) Compare the runtime of the two approaches and explain your findings.

(f) To optimize the convolution-based approach, we will constrain W to be diagonal and leverage depth-wise convolutions. Implement the new forward pass under this constraint and now compare the runtimes.

3. 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 $W^{(\alpha)}$ get higher loss $\mathcal{L}_2$ 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:

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

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?)
(Hint: The Frobenius norm is equal to $\|A\|_F^2 := tr(AA^T)$, and it is invariant under orthogonal transform. That is, $\|A\|_F^2 = \|UAV^T\|_F^2$ for any orthogonal matrices U, V, and any rectangular matrix A, as long as U, A, V have compatible shapes.)

(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?)

4. Ridge-Attention

In lecture, you saw how the standard softmax-attention mechanism can be viewed as a softened version of something akin to a nearest-neighbor model in which the value returned for a query reflects a weighted combination of the values that correspond to the keys closest to the query. In this view, the (key, value) pairs in the memory represent a kind of in-context "training data" and the query is a test input for which we want to predict the right output given that data.

(a) To start, let's think about why it is possible to efficiently update simple averaging. Let $m = \frac{1}{n}\sum_{i=1}^{n} x_i$ be the average of n points. **Use m, x_{n+1}, n and simple arithmetic operations to compute $m' = \frac{1}{n+1}\sum_{i=1}^{n+1} x_i$ — the average of all points including the new point x_{n+1}.**
(HINT: Start by multiplying m by n.)

(b) Let us now shift to thinking about traditional ridge-regression with n training pairs $(\mathbf{x}_i, y_i)$ where $\mathbf{x}_i$ are d-dimensional vectors and y_i are scalars. Let the matrix $A = \begin{bmatrix} \mathbf{x}_1^\top \\ \mathbf{x}_2^\top \\ \vdots \\ \mathbf{x}_n^\top \end{bmatrix}$ and vector $\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$ so that we can find the familiar closed-form solution

$$\mathbf{w}_* = (A^\top A + \lambda I)^{-1} A^\top \mathbf{y} \tag{6}$$

that allows us to make scalar predictions on any new test input $\mathbf{x}$ by computing $\mathbf{w}_*^\top \mathbf{x}$.
First, **write the two terms $(A^\top A + \lambda I)$ and $A^\top \mathbf{y}$ as sums involving the $\mathbf{x}_i$ and y_i.**
i.e. Complete:
$(A^\top A + \lambda I) = \lambda I + \sum_{i=1}^{n}$
$A^\top \mathbf{y} = \sum_{i=1}^{n}$

(c) Suppose we wanted to do ridge-self-attention (non-causal – "encoder-style") with a context length of n and d-dimensional query, key, and value vectors. Recall that this is the style of attention where each of the n queries is applied to the same pool of n (key, value) pairs. The goal is to calculate multi-dimensional ridge-regression predictions, after "training" on the pool of (key, value) pairs, and given the query as a kind of "test" input. (Note: the keys are playing the role of the A matrix in ridge-regression, each query is like the $\mathbf{x}$ we are testing on, and in place of the scalar y_i, we have an entire value vector for a multi-dimensional ridge-regression problem so the $\mathbf{y}$ vector is replaced with a matrix that has a row for each value vector.)

Assume that the cost of inverting a $d \times d$ matrix is $O(d^3)$ and the cost of multiplying two such matrices is also $O(d^3)$. Assume that a $d \times d$ matrix times either a d-dimensional row or column vector costs d^2 operations. You should assume $d < n$.

What is the computational cost of a non-causal ridge self-attention layer?

◯ $O(d^4)$
◯ $O(nd^2)$

◯ $O(n^2d^3)$
◯ $O(n^2d^2)$
◯ $O(n^2)$
◯ $O(1)$

(HINT: Do not forget that for a single d-dimensional query vector $\mathbf{q}$, attention needs to return a d-dimensional result.)

(d) Assume that a ridge self-attention layer is used in a Transformer architecture and there is a downstream loss. **For which of these will backprop successfully pass gradients if we use ridge self-attention?**

☐ The ridge λ viewed as a learnable parameter for the self-attention layer.
☐ The keys
☐ The values
☐ The queries

(e) Now step back. There is a nice trick (called the Sherman–Morrison Formula) by which one can update the inverse of an invertible matrix to which you make a rank-1 update. Let M be an invertible square $d \times d$ matrix and let $\mathbf{u}, \mathbf{v}$ be two d-dimensional vectors. Then:

$$(M + \mathbf{u}\mathbf{v}^\top)^{-1} = M^{-1} - \frac{1}{1 + \mathbf{v}^\top M^{-1}\mathbf{u}}(M^{-1}\mathbf{u})(\mathbf{v}^\top M^{-1}) \tag{7}$$

Assume that a $d \times d$ matrix times either a d-dimensional row or column vector costs d^2 operations, and so does the evaluation of a dyad $\mathbf{u}\mathbf{v}^\top$. Assume that computing a Euclidean inner-product costs d operations. **Assuming that you already had M^{-1} in hand, what is the computational cost of one application of (7)?**

◯ $O(d^4)$
◯ $O(d^3)$
◯ $O(d^2)$
◯ $O(d)$
◯ $O(1)$

(f) Consider implementing causal ridge-self-attention with a context length of n but where the pool of (key, value) vectors that one is querying at position t consists only of the t (key, value) pairs so far.

Describe explicitly how you would compute causal ridge-self-attention in a computationally efficient manner. Leverage your decomposition of the ridge-regression formula in part (b) of this problem together with the Sherman-Morrison formula from (7) to avoid having to do $O(n^2)$ computations while still calculating causal ridge-self-attention outputs correctly for all n positions in the context.

(HINT: Think recursively. What do you need to track from one time step to the next to avoid repeating work?)

(g) Many people consider important the ability to visualize the attention weights. For traditional softmax-attention, these are the outputs of the softmax for a given query vector. They tell you the exact amounts by which the attention outputs at this position are linear combinations of the values being fed in at this and other positions.

For ridge-attention and a given query vector, how would you compute the (possibly negative) weights associated to each of the value vectors in the context?

5. 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:

- Naman Jain.
- Qiyang Li.
- Dhruv Shah.
- Anant Sahai.

