

## Problem Set 1

CS 395T: Foundations of Modern Generative AI

Prof. Noah Golowich

Fall 2026

**Due: September 30, 11:59pm CT**

**AI Use.** It is recommended that you *not* rely on AI to complete this problem set, to further your learning. Problem sets in this course are all optional (and may be turned in for a small amount of extra credit, graded for completion; if you do so, we ask that you please do not use AI).

### Problem 1. Permutations, masks, and positional information

Represent a sequence by columns,  $X = [x_1, \dots, x_T] \in \mathbb{R}^{d \times T}$ . For a  $T \times T$  permutation matrix  $\Pi$ ,  $X\Pi$  permutes the tokens. A map  $f$  is *permutation equivariant* if  $f(X\Pi) = f(X)\Pi$  for every  $X, \Pi$ . For one self-attention head, let

$$Q = W_Q X, \quad K = W_K X, \quad V = W_V X, \quad \text{Attn}(X) = V \text{softmax}(K^\top Q / \sqrt{d_k}),$$

where  $W_Q, W_K \in \mathbb{R}^{d_k \times d}$ ,  $W_V \in \mathbb{R}^{d_v \times d}$ , and softmax normalizes each column. Multi-head attention concatenates the head outputs along the feature dimension and applies the same output projection at every position.

- (a) Prove that an *unmasked* transformer with no positional information is permutation equivariant. (This includes multi-head attention, residual connections, position-wise MLPs, and layer normalization applied independently to each column.)
- (b) In causal attention, query  $i$  may attend exactly to keys  $j \leq i$ ; all other scores are set to  $-\infty$  before softmax. Give explicit weights, an input, and a permutation for which this causal attention map is not permutation equivariant.
- (c) Let  $d_k$  be even. For fixed frequencies  $\omega_1, \dots, \omega_{d_k/2}$ , let

$$R_t = \text{diag}(R(t\omega_1), \dots, R(t\omega_{d_k/2})), \quad R(u) = \begin{pmatrix} \cos u & -\sin u \\ \sin u & \cos u \end{pmatrix}.$$

RoPE uses  $q_i = R_{p_i} \bar{q}_i$ ,  $k_j = R_{p_j} \bar{k}_j$ , where  $p_i, p_j$  are integer position IDs and  $\bar{q}_i, \bar{k}_j$  are fixed content vectors. Prove that all attention scores are unchanged by replacing every position ID  $p_t$  by  $p_t + c$  for the same integer  $c$ . Keep the content vectors and attention mask fixed.

## Problem 2. Normalized linear attention as a recurrence

For  $t = 1, \dots, T$ , suppose we are given  $q_t, k_t \in \mathbb{R}^m$  and  $v_t \in \mathbb{R}^d$ . Let  $\phi : \mathbb{R}^m \rightarrow \mathbb{R}^r$  satisfy  $\langle \phi(q_t), \phi(k_s) \rangle > 0$  whenever  $s \leq t$ . Define normalized causal linear attention by

$$y_t = \frac{\sum_{s=1}^t \langle \phi(q_t), \phi(k_s) \rangle v_s}{\sum_{s=1}^t \langle \phi(q_t), \phi(k_s) \rangle}.$$

Represent this computation by a state-space model with state  $H_t \in \mathbb{R}^{r \times (d+1)}$  and augmented input  $\tilde{v}_t = (v_t^\top, 1)^\top \in \mathbb{R}^{d+1}$ , of the form

$$H_0 = 0, \quad H_t = A(k_t)H_{t-1} + B(k_t)\tilde{v}_t^\top, \quad u_t = H_t^\top C(q_t) \in \mathbb{R}^{d+1}, \quad y_t = \frac{u_t[1:d]}{u_t[d+1]}.$$

Here  $u_t[1:d]$  denotes the vector of the first  $d$  coordinates. In particular, you should choose fixed functions  $A : \mathbb{R}^m \rightarrow \mathbb{R}^{r \times r}$  and  $B, C : \mathbb{R}^m \rightarrow \mathbb{R}^r$ , which may depend on  $\phi$  but not on  $T$  or earlier inputs. Thus the update matrices may depend on the current key, and the readout vector may depend on the current query. For your choice of  $A, B, C$ , prove that its output equals the above normalized attention for every admissible input sequence.

## Problem 3. Recurrent memory needed for copying

Fix an integer  $m \geq 1$ . A scalar-input, scalar-output linear state-space model with state dimension  $N$  processes arbitrary real inputs  $x_1, \dots, x_{2m}$  by

$$h_0 = 0 \in \mathbb{R}^N, \quad h_t = A_t h_{t-1} + B_t x_t, \quad y_t = C_t^\top h_t,$$

where  $A_t \in \mathbb{R}^{N \times N}$  and  $B_t, C_t \in \mathbb{R}^N$ . These coefficients may depend on  $m$  and the time index  $t$ , but are fixed independently of every input value. The above state space model is said to satisfy the *copying requirement* if

$$y_{m+i} = x_i \quad \text{for all } i = 1, \dots, m \quad \text{and every } (x_1, \dots, x_{2m}) \in \mathbb{R}^{2m}.$$

There is no requirement on the first  $m$  outputs.

- Set the final  $m$  inputs to zero and express the last  $m$  outputs as a linear map of the first  $m$  inputs. Show that the matrix of this map has rank at most  $N$ .
- Deduce that any model satisfying the copying requirement has  $N \geq m$ .
- Construct a model with  $N = m$  satisfying the requirement for *arbitrary* values of the final  $m$  inputs as well. Specify every  $A_t, B_t, C_t$  and verify the outputs.

## Problem 4. SGD, AdaGrad, and Adam on two quadratics

This problem is empirical in nature. *It is recommended you not use coding agents to solve this problem (if you want to learn something).* Let

$$H_1 = \begin{pmatrix} 100 & 0 \\ 0 & 1 \end{pmatrix}, \quad R = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}, \quad H_2 = RH_1R^\top.$$

Consider the objectives

$$F_1(\theta) = \frac{1}{2}\theta^\top H_1\theta, \quad F_2(\theta) = \frac{1}{2}\theta^\top H_2\theta.$$

The minimum of each objective is zero. Initialize two training runs at

$$\theta_0^{(1)} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \quad \theta_0^{(2)} = R \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ \sqrt{2} \end{pmatrix}.$$

At step  $t \geq 1$ , draw  $z_t \stackrel{\text{iid}}{\sim} \mathcal{N}(0, I_2)$  and use the stochastic gradients

$$g_t^{(1)} = H_1\theta_{t-1}^{(1)} + 0.1z_t, \quad g_t^{(2)} = H_2\theta_{t-1}^{(2)} + 0.1Rz_t.$$

**Algorithms.** Implement the following updates, with a constant base learning rate  $\eta > 0$  and coordinatewise squares, divisions, and square roots. Set  $\epsilon = 10^{-8}$  *inside* the square root for this problem.

$$\text{SGD: } \theta_t = \theta_{t-1} - \eta g_t.$$

$$\text{AdaGrad: } s_0 = 0, \quad s_t = s_{t-1} + g_t^2, \quad \theta_t = \theta_{t-1} - \eta g_t / \sqrt{s_t + \epsilon \mathbf{1}}.$$

$$\begin{aligned} \text{Adam: } m_0 = v_0 = 0, \quad m_t &= 0.9m_{t-1} + 0.1g_t, \quad v_t = 0.999v_{t-1} + 0.001g_t^2, \\ \hat{m}_t &= m_t / (1 - 0.9^t), \quad \hat{v}_t = v_t / (1 - 0.999^t), \quad \theta_t = \theta_{t-1} - \eta \hat{m}_t / \sqrt{\hat{v}_t + \epsilon \mathbf{1}}. \end{aligned}$$

- (a) Run each method for  $T = 2000$  updates on each objective. Independently select its learning rate for each objective from

$$\{10^{-4}, 3 \cdot 10^{-4}, 10^{-3}, 3 \cdot 10^{-3}, 10^{-2}, 3 \cdot 10^{-2}, 10^{-1}, 3 \cdot 10^{-1}, 1\}.$$

- (b) Evaluate on 10 random seeds. For each objective, plot the mean objective against iteration for all three methods, using a logarithmic vertical axis.
- (c) Discuss the observed effects of rotating the objective. Which methods are sensitive to the coordinate system in this experiment? Relate your observations to coordinatewise adaptation.

## Problem 5. A tiny transformer that counts

*It is recommended you not use coding agents to solve this problem (if you want to learn something).* Implement and train a decoder-only transformer from basic tensor operations and automatic differentiation. It is recommended to use Pytorch. You may use Pytorch’s embeddings module, linear layers module, layer normalization module, and optimizers, but implement attention, MLP, and the decoder blocks yourself.

**Data.** The vocabulary consists of exactly the ten digits  $\{0, 1, \dots, 9\}$ . All arithmetic on digits is modulo 10. For every starting digit  $i$  and direction  $s \in \{+1, -1\}$ , include the length-five sequence

$$(i, i + s, i + 2s, i + 3s, i + 4s) \pmod{10}.$$

Thus the training set has exactly 20 equally weighted sequences; for example,  $(8, 9, 0, 1, 2)$  and  $(1, 0, 9, 8, 7)$ . Repeatedly train on this entire set.

**Architecture.** Use learned digit embeddings of dimension 6, one decoder block, one attention head (query/key/value dimension 6), and a position-wise MLP of hidden width 12 with ReLU activation. Explicitly,

$$\text{MLP}(x) = W_2 \text{ReLU}(W_1 x + b_1) + b_2, \quad \text{ReLU}(u) = \max\{0, u\},$$

where ReLU is applied coordinatewise, between the two affine maps. The second affine map has no activation. The block has the pre-normalization form

$$\tilde{X} = X + \text{MHA}(\text{LN}_1(X)), \quad X_{\text{out}} = \tilde{X} + \text{MLP}(\text{LN}_2(\tilde{X})).$$

Attention uses scaled dot products and the causal mask  $j \leq i$ , including the current position. Apply one final layer normalization and a learned affine map to ten logits at each position. Use standard layer normalization with learned scale and bias and stabilizer  $10^{-5}$ ; include biases in all linear maps. Use separate embedding and output weights. (You don’t need to use positional embeddings.)

**Training.** For a sequence  $(x_1, \dots, x_5)$ , feed  $(x_1, \dots, x_4)$  and predict  $(x_2, \dots, x_5)$  at the four corresponding positions. Minimize mean cross-entropy over the  $20 \times 4 = 80$  predictions. Train for 100 full-batch updates with Adam at learning rate  $10^{-2}$ ,  $(\beta_1, \beta_2) = (0.9, 0.999)$ , bias correction, and  $\epsilon = 10^{-8}$  *outside* the square root (the usual PyTorch convention). Use no learning-rate schedule or gradient clipping.

- Implement the data generator, model, and training loop. Report the number of trainable parameters and plot training cross-entropy against update number for each of 3 random seeds.
- Evaluate all 20 length-five sequences by *teacher forcing*: supply the true prefix for every next-token prediction. Report cross-entropy and accuracy separately for each of the four prediction positions.
- For every  $(i, s)$  pair, prompt the trained model with the two digits  $(i, i + s)$  and generate three more digits by repeatedly taking the highest-probability next token (break ties by the smaller digit). Report the fraction of all 20 generated sequences that exactly match the required length-five sequence. Include at least one forward and one backward example that wrap around 0 or 9.
- Without further training, repeat two-digit-prompt generation to total length 12 and report exact-sequence accuracy over all 20 prompts. Also plot teacher-forced next-token accuracy as a function of the true prefix length  $\ell = 1, 2, \dots, 40$ , with a separate curve for each of your three random seeds. (In particular: For each  $\ell$  and each  $(i, s) \in \{0, \dots, 9\} \times \{+1, -1\}$ , feed the true prefix  $(i, i + s, \dots, i + (\ell - 1)s) \pmod{10}$  and predict only the next digit  $(i + \ell s) \pmod{10}$ . Define accuracy at length  $\ell$  as the fraction of these 20 predictions that are correct.)