Solutions to Transformer Architecture Exercises

Solution Manual

November 16, 2025

Contents

1 Exercise 1: Upper Bound on Attention Coefficients

Problem

Given $a_{nm} \geq 0$ and $\sum_m a_{nm} = 1$, show that $a_{nm} \leq 1$.

Approach

Since non-negative numbers sum to 1, no single number can exceed 1 (like sharing a pizza—no one gets more than the whole pizza).

Solution

For any coefficient a_{nk}:

$$a_{nk} + \sum_{m \neq k} a_{nm} = 1 \tag{1}$$

Since all terms are non-negative: $\sum_{m \neq k} a_{nm} \geq 0$, therefore:

$$a_{nk} = 1 - \sum_{m \neq k} a_{nm} \leq 1 \quad \checkmark \tag{2}$$

2 Exercise 2: Softmax Satisfies Constraints

Problem

Verify that $a_{nm} = \frac{\exp(\mathbf{x}_n^T \mathbf{x}_m)}{\sum_{m'} \exp(\mathbf{x}_n^T \mathbf{x}_{m'})}$ satisfies: (1) $a_{nm} \geq 0$ and (2) $\sum_m a_{nm} = 1$.

Approach

Exponential is always positive (ensures non-negativity), and we normalize by the sum (ensures sum to 1).

Solution

Non-negativity: Since $\exp(z) > 0$ for all z, both numerator and denominator are positive, so $a_{nm} > 0$. ✓
Normalization: The denominator is constant for fixed n:

$$\sum_m a_{nm} = \frac{\sum_m \exp(\mathbf{x}_n^T \mathbf{x}_m)}{\sum_{m'} \exp(\mathbf{x}_n^T \mathbf{x}_{m'})} = 1 \quad \checkmark \tag{3}$$

3 Exercise 3: Orthogonal Input Vectors

Problem

For orthogonal vectors ($\mathbf{x}_n^T \mathbf{x}_m = 0$ for $n \neq m$), show that self-attention returns the input unchanged: $\mathbf{Y} = \mathbf{X}$.

Approach

For orthogonal vectors, $\mathbf{X}\mathbf{X}^T$ is diagonal, so each token's query only has high similarity with itself. But softmax still spreads some probability mass to other tokens unless the diagonal values are very large.

Solution

Self-attention: $\mathbf{Y} = \text{Softmax}[\mathbf{X}\mathbf{X}^T]\mathbf{X}$
For orthogonal vectors:

$$(\mathbf{X}\mathbf{X}^T)_{nm} = \begin{cases} \|\mathbf{x}_n\|^2 & \text{if } n = m \text{ (matching token)} \\ 0 & \text{if } n \neq m \text{ (all other tokens)} \end{cases} \tag{4}$$

So $\mathbf{X}\mathbf{X}^T$ is diagonal. Applying softmax to row n:

$$\text{Row } n: \quad [\|\mathbf{x}_n\|^2,\ 0,\ 0,\ \ldots,\ 0] \tag{5}$$

After softmax:

$$a_{nn} = \frac{e^{\|\mathbf{x}_n\|^2}}{e^{\|\mathbf{x}_n\|^2} + (N-1)}, \quad a_{nm} = \frac{1}{e^{\|\mathbf{x}_n\|^2} + (N-1)} \text{ for } m \neq n \tag{6}$$

Key insight: Softmax always normalizes to sum to 1, so even when other entries are 0, they contribute $\exp(0) = 1$ each!
Concrete example: For $\|\mathbf{x}_n\|^2 = 1$ and $N = 10$ tokens:

$$a_{nn} = \frac{e^1}{e^1 + 9} = \frac{2.718}{11.718} \approx 0.232 \tag{7}$$

$$a_{nm} = \frac{1}{11.718} \approx 0.085 \text{ for each of the 9 other tokens} \tag{8}$$

Check: $0.232 + 9 \times 0.085 = 0.232 + 0.765 \approx 1$
So the output is:

$$\mathbf{y}_n = 0.232\,\mathbf{x}_n + 0.085\,\mathbf{x}_1 + \cdots + 0.085\,\mathbf{x}_{n-1} + 0.085\,\mathbf{x}_{n+1} + \cdots \tag{9}$$

This is **not** equal to $\mathbf{x}_n$—it's a weighted average where token n gets 23% weight and the other 9 tokens share 77%.
When does $\mathbf{Y} = \mathbf{X}$ actually hold?
Only when diagonal entries are much larger. For example, if $\|\mathbf{x}_n\|^2 = 10$:

$$a_{nn} = \frac{e^{10}}{e^{10} + 9} = \frac{22{,}026}{22{,}035} \approx 0.9996 \quad \text{(almost 1!)} \tag{10}$$

In the limit $\|\mathbf{x}_n\|^2 \to \infty$, we get $a_{nn} \to 1$ and $a_{nm} \to 0$, giving $\mathbf{Y} = \mathbf{X}$.
Intuition: Think of softmax as a "soft" version of picking the maximum. When you have scores $[5, 0, 0, 0]$, softmax heavily favors the first entry but still gives small probabilities to others. Only when the gap is huge (like $[100, 0, 0, 0]$) does it become nearly one-hot.
For this exercise: The statement "$\mathbf{Y} = \mathbf{X}$" is true *in the limit* or requires assuming unit vectors are implicitly scaled. For typical unit vectors, we get $\mathbf{Y} \approx \mathbf{X}$ with some mixing. ✓

4 Exercise 4: Expected Value of Dot Product Squared

Problem

For $\mathbf{a}, \mathbf{b} \sim \mathcal{N}(0, \mathbf{I}_D)$ independent, show $E[(\mathbf{a}^T\mathbf{b})^2] = D$.

Approach

Expand the square, use independence. Only diagonal terms survive in the double sum.

Solution

Each component: $a_i, b_i \sim \mathcal{N}(0,1)$ with $E[a_i] = 0$, $E[a_i^2] = 1$.
Mean: $E[\mathbf{a}^T\mathbf{b}] = \sum_i E[a_i]E[b_i] = 0$
Second moment:

$$(\mathbf{a}^T\mathbf{b})^2 = \left(\sum_i a_i b_i\right)^2 = \sum_{i,j} a_i a_j b_i b_j \tag{11}$$

Taking expectations and using independence:

$$E[(\mathbf{a}^T\mathbf{b})^2] = \sum_{i,j} E[a_i a_j]E[b_i b_j] \tag{12}$$

Since $E[a_i a_j] = \delta_{ij}$ (equals 1 if $i = j$, else 0), only diagonal terms ($i = j$) survive:

$$E[(\mathbf{a}^T\mathbf{b})^2] = \sum_{i=1}^{D} 1 \cdot 1 = D \quad \checkmark \tag{13}$$

Implication: $\mathrm{Var}(\mathbf{a}^T\mathbf{b}) = D$, so $\mathrm{SD} = \sqrt{D}$. For $D = 512$, $\mathrm{SD} \approx 23$—huge! This motivates the $1/\sqrt{D_k}$ scaling in attention to keep variance at 1.

5 Exercise 5: Multi-Head Attention Reformulation

Problem

Show $\mathbf{Y} = \mathrm{Concat}[\mathbf{H}_1, \ldots, \mathbf{H}_H]\mathbf{W}^{(o)} = \sum_{h=1}^{H} \mathbf{H}_h \mathbf{W}_h^{(o)}$.

Approach

Block matrix multiplication: $[A|B]\begin{bmatrix} C \\ D \end{bmatrix} = AC + BD$.

Solution

Partition $\mathbf{W}^{(o)}$ vertically into H blocks: $\mathbf{W}^{(o)} = \begin{bmatrix} \mathbf{W}_1^{(o)} \\ \vdots \\ \mathbf{W}_H^{(o)} \end{bmatrix}$

Then:

$$[\mathbf{H}_1 | \cdots | \mathbf{H}_H] \begin{bmatrix} \mathbf{W}_1^{(o)} \\ \vdots \\ \mathbf{W}_H^{(o)} \end{bmatrix} = \sum_{h=1}^{H} \mathbf{H}_h \mathbf{W}_h^{(o)} \quad \checkmark \tag{14}$$

This is standard block matrix multiplication.

6 Exercise 6: Self-Attention as Parameter-Efficient Network

Problem

Express self-attention as a fully-connected network and explain its parameter structure.

Approach

Self-attention connects every input token to every output token (fully connected), but achieves this with far fewer parameters than a standard fully-connected layer. The key is that the connection weights are computed from the input rather than being separate learned parameters.

Solution

Standard fully-connected baseline:
To map all N input tokens (total size ND) to all N output tokens, you'd need a matrix $\mathbf{W}_{\text{full}} \in \mathbb{R}^{ND \times ND}$ with N^2D^2 independent parameters.
Self-attention approach:
Self-attention computes $\mathbf{Y} = \mathbf{AV}$ where:

- $\mathbf{V} = \mathbf{XW}^{(v)}$ (value transformation)
- $\mathbf{A} = \text{Softmax}[\mathbf{QK}^T]$ (attention matrix, computed from input)

We can write this as a block matrix operating on all tokens:

$$\mathbf{Y} = \begin{bmatrix} a_{11}\mathbf{W}^{(v)} & a_{12}\mathbf{W}^{(v)} & \cdots & a_{1N}\mathbf{W}^{(v)} \\ a_{21}\mathbf{W}^{(v)} & a_{22}\mathbf{W}^{(v)} & \cdots & a_{2N}\mathbf{W}^{(v)} \\ \vdots & \vdots & \ddots & \vdots \\ a_{N1}\mathbf{W}^{(v)} & a_{N2}\mathbf{W}^{(v)} & \cdots & a_{NN}\mathbf{W}^{(v)} \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_N \end{bmatrix} \tag{15}$$

This is an $ND \times ND$ matrix—fully connected, not sparse!
Key insight—where the efficiency comes from:

1. **Parameter sharing:** Every block uses the *same* matrix $\mathbf{W}^{(v)}$, just scaled by different a_{nm} values. That's only D^2 parameters for $\mathbf{W}^{(v)}$, not N^2D^2!
2. **Input-dependent routing:** The scalars a_{nm} are *computed from the input* using $\mathbf{QK}^T = \mathbf{XW}^{(q)}\mathbf{W}^{(k)T}\mathbf{X}^T$. This adds $2D^2$ parameters for $\mathbf{W}^{(q)}$ and $\mathbf{W}^{(k)}$, but is shared across all positions.
3. **Total learned parameters:** $3D^2$ (query + key + value matrices) instead of N^2D^2.

Analogy: Imagine a switchboard where:

- **Fully-connected:** Each pair of phones has its own dedicated wire (needs N^2 wires)
- **Self-attention:** All pairs share the same wire, but each call has a volume knob (the a_{nm}) that's automatically adjusted based on who's calling whom. You only need one wire!

Concrete numbers: For $N = 100$ tokens, $D = 512$ dimensions:

Approach	Parameters
Fully-connected	$N^2D^2 = 2{,}621{,}440{,}000$ (2.6 billion)
Self-attention	$3D^2 = 786{,}432$
Reduction	$3330\times$ fewer parameters!

Summary: Self-attention achieves full connectivity between all tokens with a *factorized, parameter-efficient* structure. The transformation is dense (all-to-all connections), but the parameterization is low-rank (shared weights + input-dependent routing). ✓

7 Exercise 7: Equivariance Without Positional Encoding

Problem

Show that without positional encoding, permuting inputs permutes outputs the same way.

Approach

Show that permuting input rows causes Q, K, V to be permuted identically, leading to permuted outputs.

Solution

Let $\mathbf{X}^\pi$ be $\mathbf{X}$ with rows permuted by π: $(\mathbf{X}^\pi)_n = \mathbf{x}_{\pi(n)}$.
Since $\mathbf{Q} = \mathbf{X}\mathbf{W}^{(q)}$ operates row-wise:

$$(\mathbf{Q}^\pi)_n = \mathbf{x}_{\pi(n)}\mathbf{W}^{(q)} = (\mathbf{Q})_{\pi(n)} \tag{16}$$

Same for $\mathbf{K}^\pi, \mathbf{V}^\pi$. Therefore attention weights: $a^\pi_{nm} = a_{\pi(n),\pi(m)}$
The output:

$$(\mathbf{Y}^\pi)_n = \sum_m a^\pi_{nm}(\mathbf{V}^\pi)_m = \sum_m a_{\pi(n),\pi(m)}(\mathbf{V})_{\pi(m)} = \sum_\ell a_{\pi(n),\ell}(\mathbf{V})_\ell = (\mathbf{Y})_{\pi(n)} \tag{17}$$

Therefore $(\mathbf{Y}^\pi)_n = (\mathbf{Y})_{\pi(n)}$. ✓
Example: "The dog chased the cat" vs "cat the chased dog The" would be processed identically (just reordered). This is why positional encoding is essential!

8 Exercise 8: High-Dimensional Orthogonality

Problem

Show random unit vectors in high dimensions are nearly orthogonal.

Approach

The dot product has mean 0 and variance $1/D$, so it concentrates near 0 as $D \to \infty$.

Solution

For random unit vectors $\mathbf{a}, \mathbf{b}$ in $\mathbb{R}^D$, express in orthonormal basis:

$$\mathbf{a} = \sum_i \alpha_i \mathbf{u}_i, \quad \mathbf{b} = \sum_i \beta_i \mathbf{u}_i \tag{18}$$

with $\sum_i \alpha_i^2 = \sum_i \beta_i^2 = 1$.
For isotropic distribution: $E[\alpha_i] = 0$, $E[\alpha_i^2] = 1/D$.
Dot product:

$$\mathbf{a}^T\mathbf{b} = \sum_i \alpha_i \beta_i \tag{19}$$

Mean: $E[\mathbf{a}^T\mathbf{b}] = \sum_i E[\alpha_i]E[\beta_i] = 0$
Variance: Using independence and that only $i = j$ terms survive:

$$\mathrm{Var}(\mathbf{a}^T\mathbf{b}) = \sum_i E[\alpha_i^2]E[\beta_i^2] = D \cdot \frac{1}{D^2} = \frac{1}{D} \tag{20}$$

Therefore $\mathrm{SD}(\cos\theta) = 1/\sqrt{D} \to 0$ as $D \to \infty$. ✓
Examples:

D	SD	Typical angle
10	0.316	72
512	0.044	87.5

In high dimensions, random vectors are almost always perpendicular.

9 Exercise 9: Linear Transformation of Concatenated Vectors

Problem

Show that $\mathbf{W}[\mathbf{x}; \mathbf{e}] = \mathbf{W}_x\mathbf{x} + \mathbf{W}_e\mathbf{e}$.

Approach

Block matrix multiplication: partition $\mathbf{W} = [\mathbf{W}_x | \mathbf{W}_e]$ to match the concatenation.

Solution

$$\mathbf{W}\begin{bmatrix}\mathbf{x}\\ \mathbf{e}\end{bmatrix} = [\mathbf{W}_x | \mathbf{W}_e]\begin{bmatrix}\mathbf{x}\\ \mathbf{e}\end{bmatrix} = \mathbf{W}_x\mathbf{x} + \mathbf{W}_e\mathbf{e} \quad ✓ \tag{21}$$

Implication: Concatenation + linear layer ≈ addition + linear layer. Transformers use addition because it's simpler (no dimension increase, fewer parameters).

10 Exercise 10: Sinusoidal Encoding Properties

Problem

Show that $\mathbf{r}_{n+k} = \mathbf{T}_k\mathbf{r}_n$ where $\mathbf{T}_k$ depends only on k, not n.

Approach

Use trig identities: $\sin(A+B) = \sin A \cos B + \cos A \sin B$ and $\cos(A+B) = \cos A \cos B - \sin A \sin B$.

Solution

Let $\omega_i = 1/L^{i/D}$. The encoding pairs sine and cosine at each frequency.
For even i (sine):

$$r_{n+k,i} = \sin((n+k)\omega_i) = \sin(n\omega_i)\cos(k\omega_i) + \cos(n\omega_i)\sin(k\omega_i) \tag{22}$$
$$= r_{n,i}\cos(k\omega_i) + r_{n,i+1}\sin(k\omega_i) \tag{23}$$

For odd i (cosine):

$$r_{n+k,i} = \cos((n+k)\omega_{i-1}) = \cos(n\omega_{i-1})\cos(k\omega_{i-1}) - \sin(n\omega_{i-1})\sin(k\omega_{i-1}) \tag{24}$$
$$= r_{n,i}\cos(k\omega_{i-1}) - r_{n,i-1}\sin(k\omega_{i-1}) \tag{25}$$

Matrix form:

$$\begin{bmatrix} r_{n+k,i} \\ r_{n+k,i+1} \end{bmatrix} = \begin{bmatrix} \cos(k\omega) & \sin(k\omega) \\ -\sin(k\omega) & \cos(k\omega) \end{bmatrix} \begin{bmatrix} r_{n,i} \\ r_{n,i+1} \end{bmatrix} \tag{26}$$

This is a rotation matrix! The angle $k\omega$ depends only on k, not n.
The full transformation $\mathbf{r}_{n+k} = \mathbf{T}_k \mathbf{r}_n$ is block-diagonal with these rotation matrices. ✓
Why sine-only fails: $\sin((n+k)\omega) = \sin(n\omega)\cos(k\omega) + \cos(n\omega)\sin(k\omega)$ requires $\cos(n\omega)$, which isn't available in sine-only encoding.
Why this matters: The network can learn to attend to relative positions (e.g., "2 tokens back") uniformly across the sequence through linear transformations.