CSCI 145 Problem 16: The Subspace, Not the Axes

July 20, 2026

Submission Instructions

Please upload your work to Gradescope by **11:59pm on DATE**.

- Turn in a single PDF of *handwritten* solutions. Typed (including LaTeX) solutions will not be accepted.
- Problems are grouped into three-week **cycles**. Once per cycle, you will meet with the grader and present the solution to *one* problem from that cycle, selected at random when you arrive — so bring your work and be ready to explain any of them.

Problem 16: The Subspace, Not the Axes

The demo's GloVe vectors give every word 50 numbers. Ask anyone who works with embeddings what the seventh number measures and you will not get an answer, which is strange for a quantity a model computes millions of times a second. This problem works out, for the linear autoencoder from lecture, exactly what training pins down and what it leaves free.

Throughout, $\mathbf{X} \in \mathbb{R}^{n \times d}$ is the centered data matrix from lecture (so $\mathbf{X} = \mathbf{X}_c$ and we drop the subscript), $\mathbf{W}_0 \in \mathbb{R}^{d \times k}$ is the encoder, $\mathbf{W}_1 \in \mathbb{R}^{k \times d}$ is the decoder, and the reconstruction is $\hat{\mathbf{X}} = \mathbf{X}\mathbf{W}_0\mathbf{W}_1$, trained to minimize $\|\mathbf{X} - \hat{\mathbf{X}}\|_F^2$. You may assume the decoder has orthonormal rows, $\mathbf{W}_1\mathbf{W}_1^\top = \mathbf{I}_k$; this costs nothing, since any decoder can be replaced by one with orthonormal rows without changing the set of achievable reconstructions.

Part A: Tied Weights Are Optimal

Fix the decoder $\mathbf{W}_1$, and let $\mathbf{Z} = \mathbf{X}\mathbf{W}_0 \in \mathbb{R}^{n \times k}$ collect the codes, one per row. Minimizing $\|\mathbf{X} - \mathbf{Z}\mathbf{W}_1\|_F^2$ over $\mathbf{Z}$ is the normal-equations problem from the Optimization lecture, transposed: $\mathbf{W}_1$ plays the role of the design matrix and $\mathbf{Z}$ the coefficients being solved for. Set the gradient with respect to $\mathbf{Z}$ to zero and use $\mathbf{W}_1\mathbf{W}_1^\top = \mathbf{I}_k$ to show the optimal code is $\mathbf{Z}^\star = \mathbf{X}\mathbf{W}_1^\top$. Conclude that for this fixed decoder the best encoder is $\mathbf{W}_0^\star = \mathbf{W}_1^\top$: at the optimum the encoder and decoder are transposes of each other, which practitioners call "tied weights."

Part B: The Bottleneck Spans the Top-k Principal Subspace

Lecture showed two things we now use: $\hat{\mathbf{X}} = \mathbf{X}\mathbf{W}_0\mathbf{W}_1$ always has rank at most k, and no matrix of rank at most k beats the truncated SVD $\mathbf{X}_k = \mathbf{X}\mathbf{V}_k\mathbf{V}_k^\top$, where $\mathbf{V}_k = [\mathbf{v}_1 \cdots \mathbf{v}_k]$ collects $\mathbf{X}$'s top k right singular vectors. Exhibit a pair $\mathbf{W}_0, \mathbf{W}_1$ achieving $\hat{\mathbf{X}} = \mathbf{X}_k$ exactly, and explain why the two facts above make that pair a *global* minimizer of the autoencoder loss, even though the loss is not convex in $(\mathbf{W}_0, \mathbf{W}_1)$ jointly. Check your pair against Part A's tied-weights fact, and say which subspace of $\mathbb{R}^d$ the columns of your $\mathbf{W}_0$ span.

Part C: Rotating the Latent Space Changes Nothing

Let $\mathbf{R} \in \mathbb{R}^{k \times k}$ be any orthogonal matrix ($\mathbf{R}\mathbf{R}^\top = \mathbf{I}_k$), and let $(\mathbf{W}_0, \mathbf{W}_1)$ be the optimal pair from Part B. Compute the reconstruction produced by $(\mathbf{W}_0\mathbf{R}, \mathbf{R}^\top\mathbf{W}_1)$ and show it is identical to the original, so this rotated pair is equally optimal for every orthogonal $\mathbf{R}$: an entire $k(k-1)/2$-parameter family of encoders ties for first place. Now say what that family does and does not change. Writing $\mathbf{z}^{(i)} \in \mathbb{R}^k$ for the code of the ith data point (the ith row of $\mathbf{Z}$), which of the following are the same for every member of the family: the reconstruction $\hat{\mathbf{X}}$, the subspace spanned by the encoder's columns, the individual coordinates of $\mathbf{z}^{(i)}$, the pairwise distances $\|\mathbf{z}^{(i)} - \mathbf{z}^{(j)}\|$? Conclude in one sentence what a trained embedding's coordinates do and do not mean, and connect it to the question in the opening paragraph.

Where this goes next. The freedom you found here appears in every method built on a low-rank factorization, including the one the next lecture introduces. LoRA freezes a pretrained weight matrix and learns a low-rank *update* $\Delta\mathbf{W} = \mathbf{B}\mathbf{A}$ to it, so the same many-factorizations-one-product ambiguity returns, this time as something the designer exploits rather than a curiosity.