EECS 182 Homework 7 Report

Faiaz Khan

November 2025

Instructions

Your task is to produce the solution to my machine learning homework showing intermediate reasoning. Avoid hallucinating or making up definitions if information is missing, explicitly say so. Answer with clear math notation and concise justification. Do not skip derivation steps. If the problem contains multiple subparts (a), (b), (c), answer each clearly.

Here's the first question: (I gave the question as an image input)

3(b) PCA & Linear Autoencoders

Let $X \in R^{m\times n}$ be zero–centered and $C := XX^\top \in R^{m\times m}$. The linear autoencoder has encoder $W_1 \in R^{k\times m}$ and decoder $W_2 \in R^{m\times k}$; it minimizes

$$\mathcal{L}(W_1, W_2; X) = X - W_2W_1X_F^2.$$

Write $E = X - W_2W_1X$. Using $\mathrm{tr}(A^\top A) = A_F^2$ and cyclicity of trace: $\mathrm{L} = \mathrm{tr}\big[(\mathrm{X\text{-}W_2W_1X})^\top (X - W_2W_1X)\big]$
$= \mathrm{tr}(X^\top X) - 2\,\mathrm{tr}(X^\top W_2W_1X) + \mathrm{tr}\big(X^\top W_2W_1XX^\top W_1^\top W_2^\top\big)$
$= \mathrm{tr}(X^\top X) - 2\,\mathrm{tr}(W_2W_1C) + \mathrm{tr}\big(W_2W_1CW_1^\top W_2^\top\big)$
$= \mathrm{tr}(X^\top X) - 2\,\mathrm{tr}(W_2W_1C) + \mathrm{tr}\big(W_1CW_1^\top W_2^\top W_2\big).$

(i) First-order optimality conditions. We use (for compatible sizes)

$$\frac{\partial}{\partial W}\mathrm{tr}(AW) = A^\top, \qquad \frac{\partial}{\partial W}\mathrm{tr}(WAW^\top B) = BW(A + A^\top).$$

Note $C = C^\top$ and $W_1CW_1^\top$ is symmetric.

Gradient w.r.t. W_2:

$$\nabla_{W_2}\mathcal{L} = -2\,CW_1^\top + 2\,W_2\,(W_1CW_1^\top).$$

Setting to 0:

$$W_2W_1CW_1^\top = CW_1^\top \quad \Longleftrightarrow \quad (W_2W_1 - I)\,CW_1^\top = 0\,.FOC - 2$$

Gradient w.r.t. W_1:

$$\nabla_{W_1}\mathcal{L} = -2\,W_2^\top C + 2\,(W_2^\top W_2)\,W_1\,C.$$

Setting to 0:

$$(W_2^\top W_2)\,W_1C = W_2^\top C \quad \Longleftrightarrow \quad W_2^\top (W_2W_1 - I)\,C = 0\,.FOC - 1$$

(ii) Principal components satisfy the FOCs. Let $C = U\Sigma^2 U^\top$ with $U = [U_k\, U_\perp]$, $\Sigma_k = \mathrm{diag}(\sigma_1, \ldots, \sigma_k)$ and $CU_k = U_k\Sigma_k^2$. Choose $W_1^\star = U_k^\top$, $W_2^\star = U_k$ so that $W_2^\star W_1^\star = U_k U_k^\top$ (orthogonal projector).

Check (FOC-2):

$$(W_2^\star W_1^\star - I)\, C W_1^{\star\top} = (U_k U_k^\top - I)\, C U_k = (U_k U_k^\top - I)\, U_k \Sigma_k^2 = 0.$$

Check (FOC-1):

$$W_2^{\star\top}(W_2^\star W_1^\star - I)\, C = U_k^\top (U_k U_k^\top - I)\, C = 0.$$

Thus the principal components provide a stationary point; indeed they give the global optimum (unique up to sign/rotation of the k components).

Here's the next question: (I gave the question as an image input)

4. Read a Blog Post: *How to train your ResNet*

(a) Baseline vs. final. Baseline (single GPU): $\approx 341s$ (replication $\sim 356s$) to reach 94% CIFAR-10 test accuracy with ResNet-18. Final (single V100): $\approx 34s$ to 94%; with test-time augmentation they reported $\approx 26s$ to 94%. (As reported in the series at https://myrtle.ai/how-to-train-your-resnet-1-baseline/ and follow-ups.)

(b) What I learned (≈100 words). Early on, the wall-clock was dominated by data loading/augmentation rather than convolutions; reducing that overhead (e.g., precomputing common transforms) cut tens of seconds. Increasing batch size with appropriate learning-rate/weight-decay scaling reduced time further without hurting accuracy. Their "catastrophic forgetting" experiments clarified why small batches limit usable learning rates for reasons *other than* curvature, and when large batches are safe. Low-level engineering mattered: e.g., keeping BatchNorm in FP32 triggers faster cuDNN code paths; small regularization (e.g., Cutout) allowed shorter schedules while preserving 94%. Overall: remove input bottlenecks, scale batches judiciously, keep numerics friendly, and shorten schedules once regularization is tuned.

(c) Most interesting approach (≈100 words). The catastrophic-forgetting analysis. Although curvature arguments suggest large-batch instability at high learning rates, they showed time-to-accuracy can still drop with carefully scaled batches while maintaining accuracy. By separating curvature-limited from forgetting-limited regimes via controlled experiments (varying dataset size and LR factors), they gave a mechanistic view of when models forget earlier mini-batches within an epoch. This yields concrete guidance: when to push batch size (and LR) and when to fight curvature/regularization instead of heuristic knob-twiddling.

Here's the next question: (I gave the question as an image input)

7. Machine Translation

Assume target tokens include ⟨sos⟩ and ⟨eos⟩. The decoder is a 1-layer RNN initialized with the encoder's last state.

(a) Feeding encoder h_t to decoder step t forces a *time-synchronous* alignment (one source step per target step). This breaks translation in general: lengths need not match ($T_t \neq T_s$), and reordering is common. If $T_t > T_s$ we run out of encoder states; if $T_t < T_s$ some h_t are unused; and each target word can depend only on the source prefix, not all positions.

(b) Training (teacher forcing). Gold: "I see a dog". The inputs to the decoder are

$$w_1 = \langle\text{sos}\rangle, \quad w_2 = \text{“}I\text{”}, \quad w_3 = \text{“}see\text{”}, \quad w_4 = \text{“}a\text{”}, \quad w_5 = \text{“}dog\text{”},$$

and the model is trained to predict $\langle \text{eos} \rangle$ at the next step.

(c) Evaluation (greedy). With $w_1 = \langle \text{sos} \rangle$ and $w_{t+1} = \hat{y}_t$, if the model outputs "I saw a dog", then

$$w_1 = \langle \text{sos} \rangle, \quad w_2 = \text{“}I\text{”}, \quad w_3 = \text{“}saw\text{”}, \quad w_4 = \text{“}a\text{”}, \quad w_5 = \text{“}dog\text{”},$$

and typically $\hat{y}_5 = \langle \text{eos} \rangle$.

Here's the next question: (I gave the question as an image input)

8. Self-supervised Linear Autoencoders

We minimize

$$\mathcal{L}_\lambda(W_1, W_2; X) = \frac{1}{n} X - W_2 W_1 X_F^2 + \lambda\big(W_1{}_F^2 + W_2{}_F^2\big).$$

(a)(i) Number of linear layers. *Two*: encoder $x \mapsto W_1 x$ and decoder $z \mapsto W_2 z$ (use `bias=False`).

(a)(ii) Loss. `nn.MSELoss` (matches $x - W_2 W_1 x_2^2$ for each sample).

(a)(iii) What is needed to optimize (6) exactly as written? *SGD optimizer* and *Weight Decay* (to implement $\lambda W_1{}_F^2 + \lambda W_2{}_F^2$). Dropout/LayerNorm/BatchNorm are not part of (6).

(b) Inductive bias toward orthonormal columns in W_2. For small $\lambda > 0$, the reconstruction term pushes $P^\star := W_2 W_1$ toward the rank-k projector $\Pi_k = U_k U_k^\top$ onto the PCA subspace. Among all factorizations $P^\star = W_2 W_1$, the penalty $\Phi(W_1, W_2) = W_1{}_F^2 + W_2{}_F^2$ selects minimal Frobenius norms.

Let

$$W_2 = U_k \, \Sigma_2 R^\top, \qquad W_1 = R \, \Sigma_1 U_k^\top, \qquad R \in R^{k \times k} orthogonal.$$

Then $W_2 W_1 = U_k(\Sigma_2 \Sigma_1) U_k^\top = P^\star$ implies, for $i = 1, \ldots, k$,

$$(\Sigma_2)_{ii} \, (\Sigma_1)_{ii} = 1.$$

$*$ The regularizer becomes

$$\Phi = \sum_{i=1}^{k} \Big((\Sigma_1)_{ii}^2 + (\Sigma_2)_{ii}^2 \Big).$$

For each i, minimize $a^2 + b^2$ subject to $ab = 1$. Setting $b = 1/a$ gives $f(a) = a^2 + a^{-2}$ with $f'(a) = 2a - 2a^{-3} = 0 \Rightarrow a^4 = 1 \Rightarrow a = 1$, and $f''(1) = 8 > 0$. Hence the unique minimum per i is $(a, b) = (1, 1)$. Therefore $(\Sigma_1)_{ii} = (\Sigma_2)_{ii} = 1$ for all i, yielding

$$W_2^\top W_2 = I_k, \qquad W_1 W_1^\top = I_k.$$

Thus a small nonzero λ induces a bias toward *approximately orthonormal columns* in W_2 (and orthonormal rows in W_1). The solution is non-unique up to a shared rotation $Q \in O(k)$: $(W_2 Q^\top, \; Q W_1)$ has the same product and penalty.