# HW1

**Submission instructions:** Submit a PDF with your solutions, and a Gist if including any code

## 1 Linear Autoencoders and PCA

In this question we will study linear autoencoders from a loss landscape perspective.

### Setup

Suppose we have a data distribution on $x \in \mathbb{R}^d$. Suppose $\mathbb{E}_x[x] = 0$, and let $\Sigma = \mathbb{E}_x[xx^T]$ be the covariance of $x$. Let $\Sigma = USU^T$ be the spectral decomposition of $\Sigma$, with $U$ orthonormal and $S$ diagonal (ordered by magnitude).

We want to recover the principle components of $x$. Let us try to train a linear auto-encoder, of the form:

$$f_{A,B}(x) := A(Bx)$$

where $B \in \mathbb{R}^{k \times d}$ is a rank-$k$ "encoder" matrix, and $A \in \mathbb{R}^{d \times k}$ is the "decoder" matrix. We use the loss:

$$L_1(A, B) := \mathop{\mathbb{E}}_{x}[||f_{A,B}(x) - x||_2^2]$$

which measures the reconstruction-loss of the autoencoder, in expectation over the true distribution of $x$. (Note that in practice, we would approximate this loss via samples from $x$. Here we consider the exact test loss for simplicity). Intuitively, we may expect that this autoencoder recovers the best $k$-dimentional projection of $x$, since we force it to "compress" down to a $k$-dimentional code space.

### Part 1

Let $U_k \in \mathbb{R}^{d \times k}$ be the first $k$ columns of $U$. That is, $U_k$ is the matrix of the top-$k$ eigenvectors of $\Sigma$.

Prove that the optimal test loss $L_1$ is achieved for $f_{A,B}(x) = U_k U_k^T$.

That is:

$$A^*, B^* \in \mathop{\text{argmin}}_{A,B} L_1(A, B) \implies f_{A^*,B^*}(x) = U_k U_k^T.$$

This implies the optimal linear autoencoder projects onto the top $k$ principle components of the covariance of $x$.

Hint: Apply or adapt the Eckart-Young-Mirsky theorem.

## Part 2

Part 1 shows that the learnt encoder and decoder matrices satisfy $AB = U_k U_k^T$. But do we always have $A = U_k$ and $B = U_k^T$? Is this the unique solution? That is, will the encoder always project into the principle-component basis?

Prove that this is not guaranteed to happen. That is, prove that there is a set of minima $(A^*, B^*)$ which equivalently minimize the loss $L_1$. What can you say about this set? Describe an invariant of this set (i.e. a transformation which leaves the set invariant).

## Part 3

In Part 2 you should have shown that there are many minimas for the loss $L_1$. But are these minimas reachable by gradient descent?

Specifically, consider the distribution produced by:

1. Sample initial matrices $A_0, B_0$ by sampling each coordinate independently $\mathcal{N}(0, 1)$.
2. Run gradient flow on the loss $L_1(A, B)$ until reaching a stationary point.
3. Output $(A, B)$ at the stationary point.

Prove that this distribution is supported on more than one point. That is, show that gradient flow from random initialization will not converge to a single unique minima.

What can you say about the distribution of stationary points of the above procedure? (Hint: think about symmetries that exist at initialization)

## Part 4

Suppose we try to reduce the symmetries in the loss, by adding a regularizer. Formally, a symmetry of $L$ is a function $T : (A, B) \mapsto (A', B')$ such that $L(A, B) = L(T(A, B))$.

Consider the loss function:

$$L_2(A, B) := \mathop{\mathbb{E}}_{x}[||f_{A,B}(x) - x||_2^2] + \lambda ||A||_F^2 + \lambda ||B||_F^2$$

Does $L_2$ have the same symmetries as $L_1$? Comment on the symmetries of $L_2$.

## Part 5

Can you design a regularizer which will remove further symmetries? Try to think of a regularized loss which will have a unique solution (perhaps by modifying the regularizer in Part 4).

This part is open-ended and does not require formal proofs. You can simply write down your proposed regularizer, and describe intuitions for why you expect it will work. (Optionally, you can prove it or show it empirically).

## 2 Deep Linear Networks

In this question, we will study the dynamics of deep linear networks. We will try to understand how the depth changes the dynamics of the network compared to that simple linear regression. We will provide a the high-level steps in the proof sketch, and you are required to fill in the blanks.

### Setup

We are given a dataset $\{x_i, y_i\}_{i=1}^N$ with $x \in \mathbb{R}^D$ and $y \in \mathbb{R}$. The inputs are gathered in a weight matrix $X \in \mathbb{R}^{D\times N}$ and normalized such that $XX^T = I$. The outputs are gathered in a vector $Y \in \mathbb{R}^{1\times N}$ and the input-output correlations are written as $YX^T = U\Sigma V$ where $\Sigma = \begin{bmatrix} s_1 & & \\ & \ddots & \end{bmatrix}$.

We use this dataset to train a depth-L linear network of the form $\hat{y}(x) = \hat{w} \cdot x = w_{tot}x = \left(\prod_{l=1}^L W_l\right) \cdot x$ where $W_l \in \mathbb{R}^{D_l \times D_{l-1}}$ is the weight matrix at layer $l$, $D_l$ is the output dimension of layer $l$ and $L$ is the depth of the network. The network is trained by minimizing the mean-squared loss

$$\mathcal{L}(W_1 ... W_L) = \frac{1}{N}\sum_{i=1}^N \left(y_i - \prod_{l=1}^L W_l \cdot x_i\right)^2 \tag{1}$$

The weights are updated using gradient flow (Gradient descent with small learning rate $\eta$)

$$\dot{W}_l = -\eta \frac{\partial \mathcal{L}}{\partial W_l} \tag{2}$$

We will assume the *orthogonal initialization* for this network: There are $L$ orthogonal matrices $R_l$ that diagonalize the weight matrices as

$$R_{l+1}^T W_l(t_0) R_l = D_l(t_0) \tag{3}$$

for some diagonal matrix $D_l(t_0) = \begin{bmatrix} d_1^l & & \\ & \ddots & \end{bmatrix}$. To simply things, we will also assume that $R_1 = V$ and $R_L = U$.

### Part 1

Write the expression for $\frac{\partial \mathcal{L}}{\partial W_l}$ i.e. the gradient of the loss wrt the weight matrix at the $l$-th layer

### Part 2

Prove the following interesting property of the orthogonal initialization: If the network starts with orthogonal initialization, then it stays in this condition throughout optimization

$$R_{l+1}^T W_l(t) R_l = D_l(t) \quad \forall t \geq t_0 \tag{4}$$

for some diagonal matrix $D_l(t)$.

### Part 3

Show that under the orthogonality condition, each each "singular mode" (i.e. the diagonal values $d_i^l$) evolves independently of the other diagonal elements $d_{\neq i}^l$. Show that this gives us the following scalar differential equation:

$$\frac{\partial d_j^l}{\partial t} = \prod_{m \neq l} d_j^m \left( s - \prod_n d_j^n \right) \tag{5}$$

*Optional* One can further assume that $d_j^1 = d_j^2 ... = d_j^L$ to get closed-form solutions for the above differential equations. Qualitatively, compare it to the dynamics of a depth-1 linear network (simple linear regression). What do we learn from these solutions?

## 3 CLIP

In this question, we will play the recently released CLIP model. This model learns a joint embedding for text and images. Thus, it can be used for image-image or text-image search. Read the blog/paper and familiarize yourself with the basic functioning of this model.

Now, we will investigate the representation space of the CLIP embeddings. It has been found that text embeddings (for eg: word2vec) or image embeddings (for eg: GANs https://arxiv.org/pdf/1511.06434.pdf Figure 7) often have nice vector properties. For example, they satisfy the following example

$$Embedding(King) - Embedding(Male) + Embedding(Female) = Embedding(Queen)$$

We want to test if CLIP also displays such *linearity* in its joint image-text embedding space. Specifically, we want to test if the embedding can do the following

$$\begin{aligned} Embedding(Image : King) - Embedding(Text : Male) + Embedding(Text : Female) \\ = Embedding(Image : Queen) \end{aligned}$$

To test this, you can use the text-image retrieval engine created by Vladimir Haltakov. The code is open source here, with a Colab you can play with here. You will want to register as a developer on Unsplash (free), make a test App, and copy your Access Key to use in that Colab.

You can play around with the normalization or scaling of the respective embeddings. Describe what you did, and post 5 failure/success cases of such linearity in CLIP.

*Hint:* A recent follow up to the CLIP paper shows that this is possible https://arxiv.org/pdf/2102.05918.pdf Figure 5

### CLIP: Open-Ended

Read the CLIP paper. Then, pick one aspect of the paper that you are interested in (or that surprised you), and write a 1 page "research proposal" on how you would study this aspect. The proposal should include:

1. A description of the question (eg, *"What factors influence robustness to natural distribution shifts?"*).
2. A description of how the CLIP paper informed your thoughts on the question.

3. A survey of prior work relevant to the question (mention how these works compare to CLIP).
4. A detailed proposal of how you would study your research question, if you had infinite compute and resources. (In particular, this could involve collecting new datasets, re-training CLIP, etc).
5. A proposal of how you might start attacking this question given a reasonable amount of resources. How would you design experiments to best understand your research question?

You do not need to execute on this research proposal for your final project – this is just an exercise in posing good research questions and designing experiments.