1 SVD and PCA

Problem A

- X is $N \times N$ matrix, $X = U \Sigma V^T$

* Want to show that columns of U are principal components of X:

$$X = U \Sigma V^T \Rightarrow XX^T = (U\Sigma V^T)(U \Sigma V^T)^T$$
$$= U \Sigma V^T V \Sigma U^T$$
$$= U \Sigma^2 U^T$$

- We know PCA property: $XX^T = U \Lambda U^T$ (zero mean) where each column of U is an eigenvector of XX^T
- So, for SVD, each column of U is a principal component of X
- Relationship: eigenvalues of XX^T are the singular values of X squared (considering diagonal of Σ^2)

Problem B

Intuitively, with eigenvalues being the variance of their corresponding eigenvectors, we can see how these eigenvalues of the PCA of X are non-negative because variance is always non-negative. On the other hand, in **problem 1a**, we just showed that these eigenvalues are the squares of the singular values of X, and since squared values are non-negative, we have reached another explanation for why these eigenvalues must be non-negative.

Problem C

Want to prove $Tr(ABC) = Tr(BCA) = Tr(CAB)$ for any num. of square matrices:

- First, let us prove this identity holds for two matrices

$\Rightarrow Tr(AB) = \sum_{i=1}^{N} (AB)_{ii} = \sum_{i=1}^{N} \sum_{j=1}^{N} A_{ij} B_{ji} = \sum_{j=1}^{N} \sum_{i=1}^{N} B_{ji} A_{ij} = \sum_{j=1}^{N} (BA)_{jj} = Tr(BA)$ ✓

- We can generalize this for 3 or more matrices:

$\Rightarrow Tr(ABC) = \sum_{i=1}^{N} (ABC)_{ii} = \sum_{i=1}^{N} \sum_{j=1}^{N} (AB)_{ij} C_{ji} = \sum_{j=1}^{N} \sum_{i=1}^{N} C_{ji} (AB)_{ij} = \sum_{j=1}^{N} (CAB)_{jj}$

$\Rightarrow Tr(ABC) = Tr(CAB)$

- This holds for $Tr(ABC) = Tr(BCA)$ if (BC) are grouped instead of (AB)

$\Rightarrow$ Holds for 3 matrices $\Rightarrow$ Holds for 3+ matrices

(More than 3 matrices can always be grouped appropriately)

Problem D

Compared to the N^2 values needed to store X, the number of values needed to store a truncated SVD with k singular values is $k + 2kN$. This is because we would take k values from Σ and k columns with length N from both U and V. We know that storing the whole matrix is storing N^2 values, so we can solve for the values of k that make storing the truncated SVD more efficient as follows:

$$k + 2kN < N^2 \implies k \cdot (1 + 2N) < N^2 \implies k < \frac{N^2}{1 + 2N}$$

Problem E

We assume $D > N$ and that X has rank N. Let us show $U\Sigma = U'\Sigma'$ such that Σ' is the NxN matrix consisting of the first N rows of Σ, and U' is the D x N matrix consisting of the first columns of U. Denoting the entries of Σ as $(\Sigma)_{ij}$, we know that the diagonal values $(\Sigma)_{ii}$ when $i \in \{1, ..., N\}$ where N is the rank of X are nonzero. So, with this and knowing $D > N$, we can see that Σ' is Σ with the rows of zeros after the last diagonal value removed. Thus, $U\Sigma = U'\Sigma'$ since these rows of zero in Σ essentially remove any columns after the N column in U. So, these products must be equivalent.

Problem F

We know that a matrix A is orthogonal is $AA^T = A^TA = I$. With U' being a D x N matrix (not a square), we can see that $U'U'^T$ is a D x D matrix and U'^TU' is a N x N matrix. Thus, $U'U'^T \neq U'^TU'$, meaning U' is not orthogonal.

Problem G

Let us show that $U'^TU' = I_{N\times N}$. With the columns of U' being orthonormal, U'^TU' has values of 1 in its diagonal and 0 elsewhere. Thus, $U'^TU' = I_{N\times N}$. However, $U'^TU' = I_{D\times D}$ does not hold because because if this were true, then U' would be orthogonal which we already have shown to be false in **problem F**. This makes sense because U' has more rows than columns, so all of the rows are not orthonormal and thus not linearly independent.

Problem H

* Want to show $X^+ = V\Sigma^+U^T = V\Sigma^{-1}U^T$

• Σ is invertible, X = matrix $D\times N$, V is orthogonal

• Let us show that $\Sigma^+ = \Sigma^{-1}$

$$\Rightarrow \Sigma = \begin{bmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & & \vdots \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & & \sigma_0 \end{bmatrix} \Rightarrow \Sigma^{-1} = \begin{bmatrix} \frac{1}{\sigma_1} & 0 & \cdots & 0 \\ 0 & \frac{1}{\sigma_2} & \ddots & \vdots \\ \vdots & & & \\ 0 & \cdots & & \frac{1}{\sigma_D} \end{bmatrix}$$

• Applying pseudoinverse: $\Sigma^+ = \begin{bmatrix} \frac{1}{\sigma_1} & 0 & \cdots & 0 \\ 0 & \frac{1}{\sigma_2} & \ddots & \vdots \\ \vdots & & & \\ 0 & \cdots & & \frac{1}{\sigma_D} \end{bmatrix}$

$$\Rightarrow \Sigma^{-1} = \Sigma^+ \Rightarrow X^+ = V\Sigma^+U^T = V\Sigma^{-1}U^T$$

Problem I

Want to show $X^{+'} = (X^TX)^{-1}X^T = V\Sigma^{-1}U^T$

- $X = U\Sigma V^T$

$$\Rightarrow X^{+'} = (X^TX)^{-1}X^T = \left((U\Sigma V^T)^T(U\Sigma V^T)\right)^{-1}(U\Sigma V^T)^T$$

$$= (V\Sigma U^T U\Sigma V^T)^{-1}(V\Sigma U^T)$$

$$= (V\Sigma^2 V^T)^{-1}(V\Sigma U^T)$$

$$= V\Sigma^{-2}\Sigma U^T$$

$$X^{+'} = V\Sigma^{-1}U^T \quad \checkmark$$

Problem J

The expression in **problem I** highly prone to numerical errors because the condition number of Σ will always be much smaller than the condition number of X^TX. We can also imagine how computing the inverse of Σ would be much more computationally straightforward since we would just be dealing with the diagonal values which would not be the case with X. Thus, we see how computing the inverse of X^TX would be more prone to numerical errors.

2 Matrix Factorization

Question 2 Code

Problem A

$$\Rightarrow \partial_{u_i} = \frac{\lambda}{2}(2u_i) + \frac{1}{2}\sum_j (-2v_j)(y_{ij} - u_i^T v_j)$$

$$\partial_{u_i} = \lambda u_i - \sum_j v_j (y_{ij} - u_i^T v_j)$$

$$\Rightarrow \partial_{v_j} = \lambda v_j - \sum_i u_i (y_{ij} - u_i^T v_j)$$

Problem B

$$\partial u_i = 0 \Rightarrow 0 = \lambda u_i - \sum_j v_j (y_{ij} - u_i^T v_j)$$

$$0 = \lambda u_i - \sum_j v_j y_{ij} + \sum_j v_j u_i^T v_j$$

$$0 = \lambda u_i - \sum_j v_j y_{ij} + u_i \sum_j v_j v_j^T$$

$$\sum_j v_j y_{ij} = u_i \left(\lambda I + \sum_j v_j v_j^T\right)$$

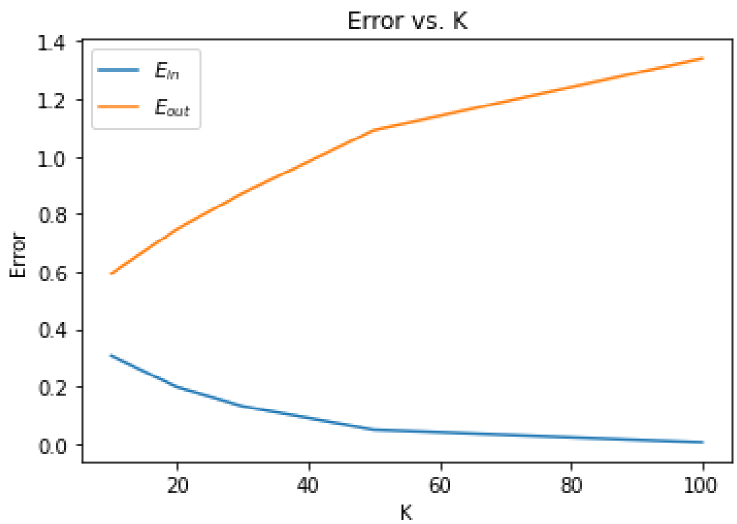
$$\Rightarrow u_i = \sum_j v_j y_{ij} \left(\lambda I + \sum_j v_j v_j^T\right)^{-1}$$

$$\Rightarrow v_j = \sum_i u_i y_{ij} \left(\lambda I + \sum_i u_i u_i^T\right)^{-1}$$

Problem C

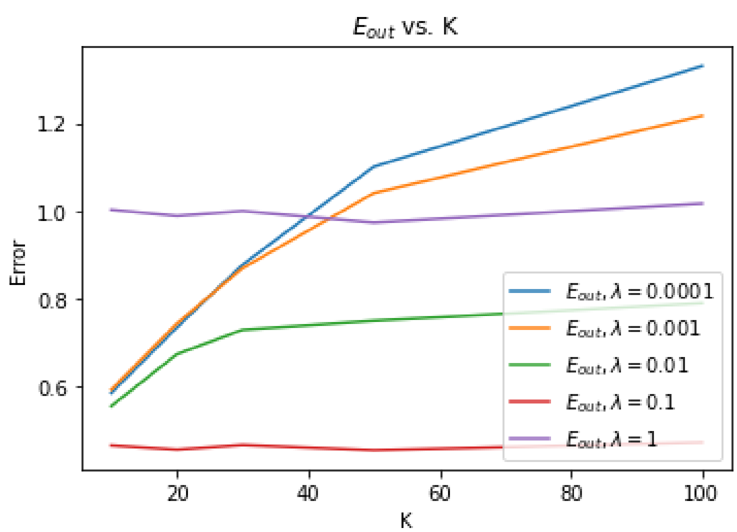
See Question 2 Code

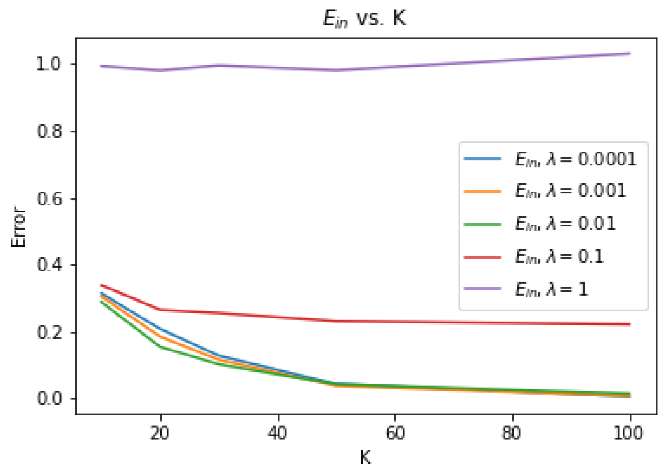
Problem D



Clearly, we see that E_{in} decreases and E_{out} increases as k increases. Thus, we see evidence of overfitting. This makes sense because with the number of latent factors (k) increasing, there is less generalization and the training data is fit more closely.

Problem E





With these two plots, we see the same general trends for E_{in} and E_{out} as k increases such that the training error generally decreases with increasing k while the testing error generally increases. Considering regularization, we see that increasing regularization λ generally flattens the increasing/decreasing curves. We see that with $\lambda = 1$, both E_{in} and E_{out} are essentially constant at 1.0. Overall, we see that increasing regularization helps to combat overfitting, but too much regularization leads to poor performance and overfitting.

3 Word2Vec Principles

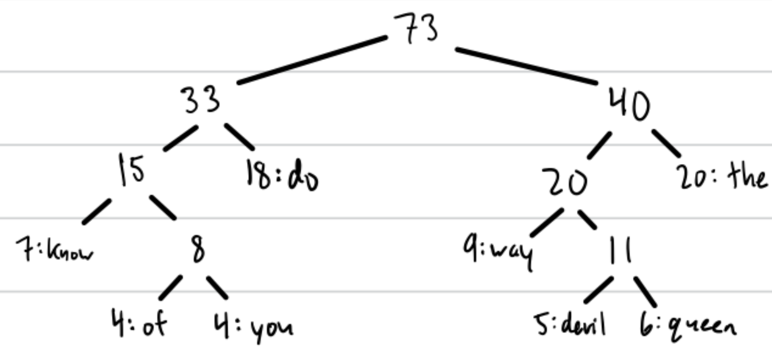
Question 3 Code

Problem A

For a single w_O, w_I pair, we can see calculating $\nabla \log\ p(w_O|w_I)$ has a time complexity of $O(WD)$. This is because this calculation essentially involves iterating over the number of words (W) as we compute the dot products such that for each step in this iteration, we are performing a computation with vectors of dimensions D. Thus, we have $O(WD)$ for a single pair.

Problem B

Huffman tree:

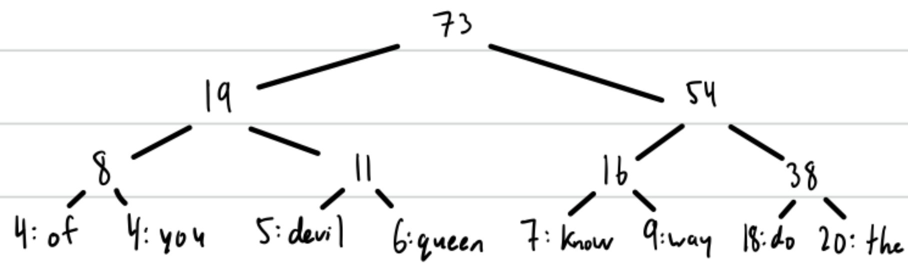


expected representation length

$$= \frac{\left(7(3)+4(4)+4(4)+18(2)+9(3)+5(4)+6(4)+20(2)\right)}{73}$$

$$= 2.74$$

Balanced binary tree:



expected representation length = 3

Problem C

As D increases, the value of the training objective increases as the training accuracy is improved. One might not want to use very large D in order to prevent overfitting.

Problem D

See Question 3 Code

Problem E

The dimension of the weight matrix of the hidden layer is 308 x 10. There are 308 unique words in the textfile.

Problem F

The dimension of the weight matrix of the output layer is 10 x 308.

Problem G

```
Textfile contains 308 unique words
torch.Size([10, 308])
torch.Size([308, 10])
Pair(star, kite), Similarity: 0.9312736
Pair(kite, star), Similarity: 0.9312736
Pair(at, swish), Similarity: 0.92859495
Pair(swish, at), Similarity: 0.92859495
Pair(be, fox), Similarity: 0.92831886
Pair(fox, be), Similarity: 0.92831886
Pair(took, yop), Similarity: 0.914207
Pair(yop, took), Similarity: 0.914207
Pair(about, open), Similarity: 0.90022033
Pair(open, about), Similarity: 0.90022033
Pair(cannot, home), Similarity: 0.8887963
Pair(home, cannot), Similarity: 0.8887963
Pair(put, yop), Similarity: 0.88486445
Pair(tell, yop), Similarity: 0.8841802
Pair(left, grows), Similarity: 0.8830075
Pair(grows, left), Similarity: 0.8830075
Pair(just, goat), Similarity: 0.8800384
Pair(goat, just), Similarity: 0.8800384
Pair(look, now), Similarity: 0.8790283
Pair(now, look), Similarity: 0.8790283
Pair(you, think), Similarity: 0.8780294
Pair(think, you), Similarity: 0.8780294
Pair(live, kite), Similarity: 0.8768652
Pair(day, brush), Similarity: 0.876771
Pair(brush, day), Similarity: 0.876771
Pair(bed, no), Similarity: 0.8740352
Pair(no, bed), Similarity: 0.8740352
Pair(seven, wink), Similarity: 0.87338483
Pair(wink, seven), Similarity: 0.87338483
Pair(was, kind), Similarity: 0.87187773
```

Problem H

A clear pattern is that almost each pairing of words shows up twice with the ordering flipped. We also see that the similarities between these flipped pairs are equivalent. Considering this file is named dr_suess.txt, we see a lot of words that would be written in a Dr. Seuss book, and we also notice that the pairs of words are either slightly rhyming or related in context.