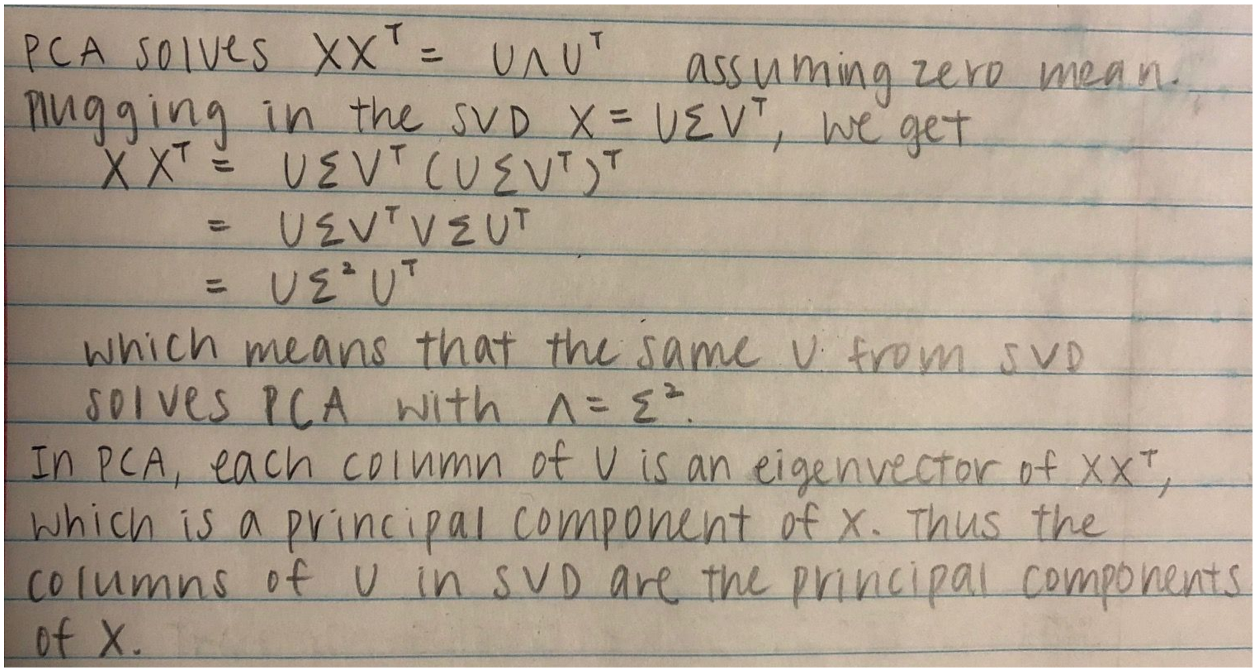
* 1 hour late token used

CS 155 PS 5

1 SVD and PCA

Problem A



PCA solves $XX^T = U \Lambda U^T$ assuming zero mean.
Plugging in the SVD $X = U\Sigma V^T$, we get

$$\begin{aligned} XX^T &= U\Sigma V^T (U\Sigma V^T)^T \\ &= U\Sigma V^T V \Sigma U^T \\ &= U\Sigma^2 U^T \end{aligned}$$

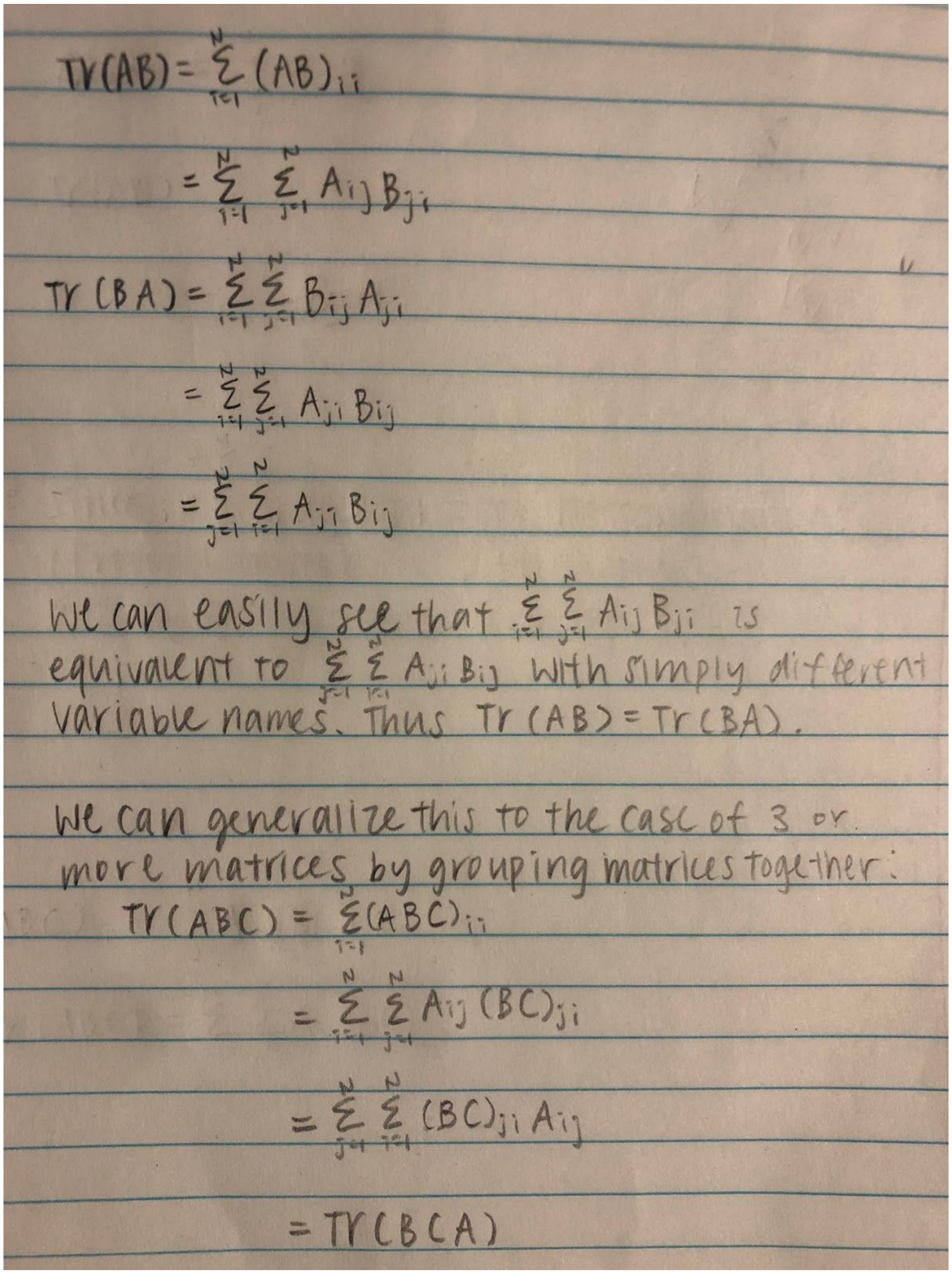
which means that the same U from SVD solves PCA with $\Lambda = \Sigma^2$.
In PCA, each column of U is an eigenvector of XX^T, which is a principal component of X. Thus the columns of U in SVD are the principal components of X.

The eigenvalues of XX^T are squares of singular values of X.

Problem B

The eigenvalues represent the amount of variance in directions of the corresponding eigenvectors, and variance is always non-negative. We also know that they're squares of the singular values of X, and squares can't be negative.

Problem C



$$\mathrm{Tr}(AB) = \sum_{i=1}^{N} (AB)_{ii}$$

$$= \sum_{i=1}^{N} \sum_{j=1}^{N} A_{ij} B_{ji}$$

$$\mathrm{Tr}(BA) = \sum_{i=1}^{N} \sum_{j=1}^{N} B_{ij} A_{ji}$$

$$= \sum_{i=1}^{N} \sum_{j=1}^{N} A_{ji} B_{ij}$$

$$= \sum_{j=1}^{N} \sum_{i=1}^{N} A_{ji} B_{ij}$$

We can easily see that $\sum_{i=1}^{N} \sum_{j=1}^{N} A_{ij} B_{ji}$ is equivalent to $\sum_{j=1}^{N} \sum_{i=1}^{N} A_{ji} B_{ij}$ with simply different variable names. Thus $\mathrm{Tr}(AB) = \mathrm{Tr}(BA)$.

We can generalize this to the case of 3 or more matrices by grouping matrices together:

$$\mathrm{Tr}(ABC) = \sum_{i=1}^{N} (ABC)_{ii}$$

$$= \sum_{i=1}^{N} \sum_{j=1}^{N} A_{ij} (BC)_{ji}$$

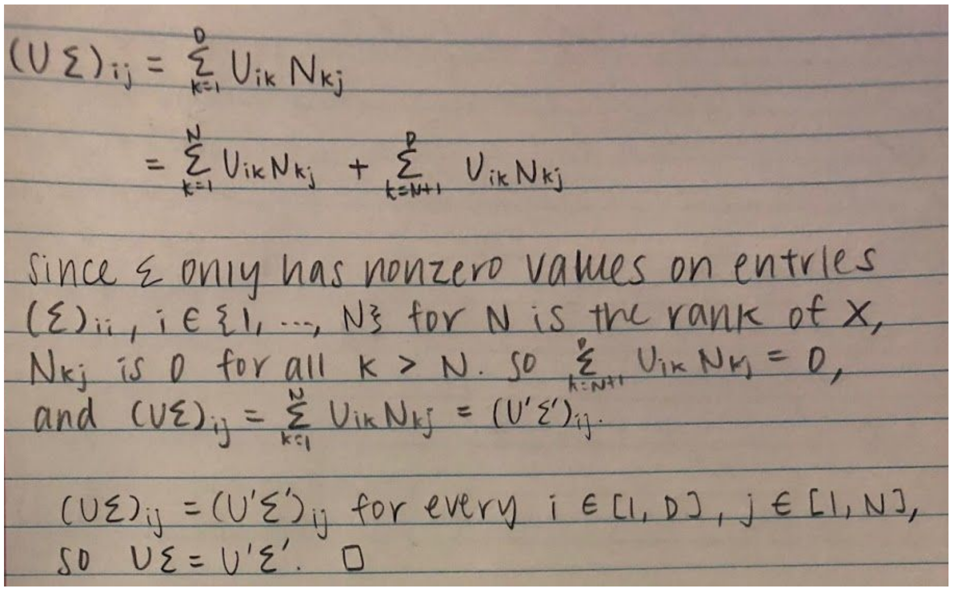
$$= \sum_{j=1}^{N} \sum_{i=1}^{N} (BC)_{ji} A_{ij}$$

$$= \mathrm{Tr}(BCA)$$

Problem D

We need to store k columns each of U and V with length N, for a total of 2 * N * k values, and k values from the diagonal matrix Σ. Thus we need 2 (N * k) + k values total to store a truncated SVD with k singular values. Storing the truncated SVD is more efficient than storing the whole matrix when k is less than N^2 / (2 N + 1).

Problem E



$$(U\Sigma)_{ij} = \sum_{k=1}^{D} U_{ik} N_{kj}$$

$$= \sum_{k=1}^{N} U_{ik}N_{kj} + \sum_{k=N+1}^{D} U_{ik}N_{kj}$$

Since Σ only has nonzero values on entries $(\Sigma)_{ii}$, $i \in \{1, \dots, N\}$ for N is the rank of X, N_{kj} is 0 for all $k > N$. So $\sum_{k=N+1}^{D} U_{ik}N_{kj} = 0$, and $(U\Sigma)_{ij} = \sum_{k=1}^{N} U_{ik}N_{kj} = (U'\Sigma')_{ij}$.

$(U\Sigma)_{ij} = (U'\Sigma')_{ij}$ for every $i \in [1, D]$, $j \in [1, N]$, so $U\Sigma = U'\Sigma'$. $\square$

Problem F

U' is a D x N matrix, so $U'U'^T$ is a D x D matrix, while U'^TU' is an N x N matrix. Matrices of different dimensions cannot be equal, so the condition for orthogonality $AA^T = A^TA$ could never hold true for a non-square matrix U'.

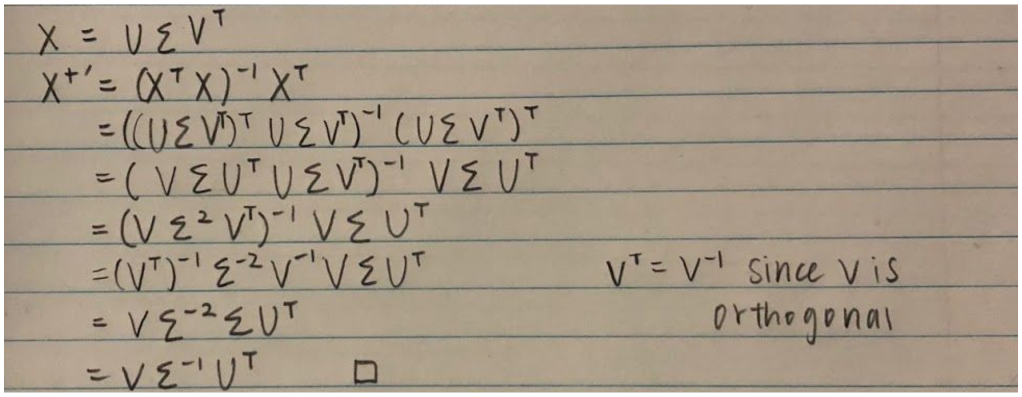
Problem G

Since the columns of U' are orthonormal, for any column u in U', <u, u> = 1, and for any two columns u_1 and u_2 in U', $<u_1, u_2> = 0$. Thus U'^TU' has a value of 1 for all diagonal entries, and a value of 0 for all other entries, which means that it is $I_{N \times N}$. Because U' has more rows than columns, the rows can't all be linearly independent and thus can't be orthonormal. Therefore a dot product of two different rows in U' can be a nonzero value, and thus it is not true that $U'U'^T = I_{D \times D}$.

Problem H

The inverse of a diagonal matrix A is a matrix with each entry as the reciprocal of the corresponding matrix in A. Thus we can easily see that Σ^+ defined in lecture 10, with σ^+ being 1 / σ if $\sigma < 0$ and 0 otherwise, is equivalent to Σ^{-1}, where all the entries are reciprocals of entries in Σ, with the assumption that σ is nonnegative. Thus $V\Sigma^+U^T$ is equivalent to $V\Sigma^{-1}U^T$.

Problem I



$X = U\Sigma V^T$

$X^{+'} = (X^T X)^{-1} X^T$

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

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

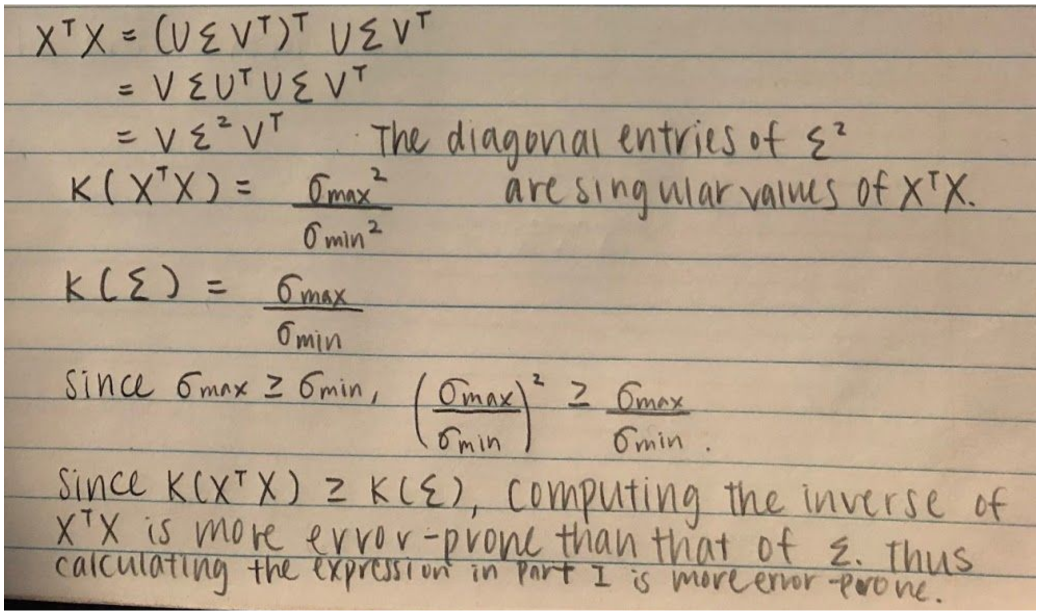
$= (V\Sigma^2 V^T)^{-1} V\Sigma U^T$

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

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

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

Problem J



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

The diagonal entries of Σ^2 are singular values of X^TX.

$$K(X^TX) = \frac{\sigma_{max}^2}{\sigma_{min}^2}$$

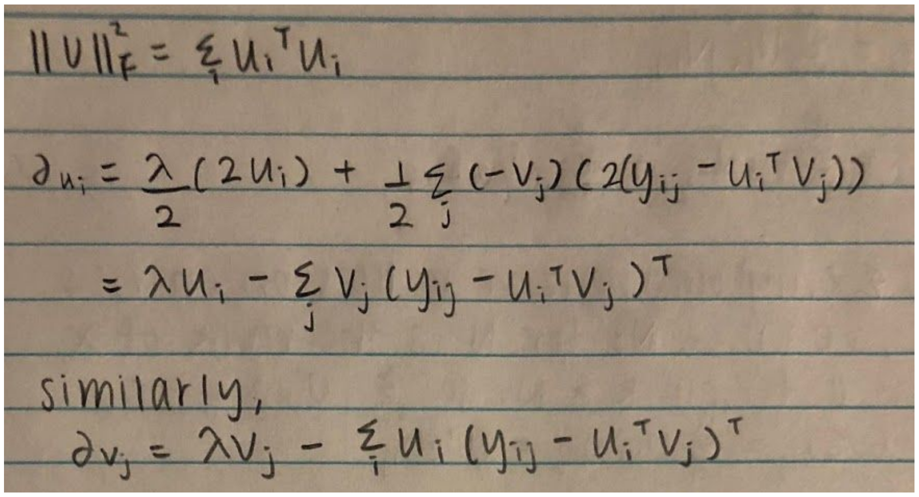
$$K(\Sigma) = \frac{\sigma_{max}}{\sigma_{min}}$$

Since $\sigma_{max} \geq \sigma_{min}$, $\left(\frac{\sigma_{max}}{\sigma_{min}}\right)^2 \geq \frac{\sigma_{max}}{\sigma_{min}}$.

Since $K(X^TX) \geq K(\Sigma)$, computing the inverse of X^TX is more error-prone than that of Σ. Thus calculating the expression in part I is more error-prone.

2 Matrix Factorization

Problem A



$$\|U\|_F^2 = \sum_i u_i^T u_i$$

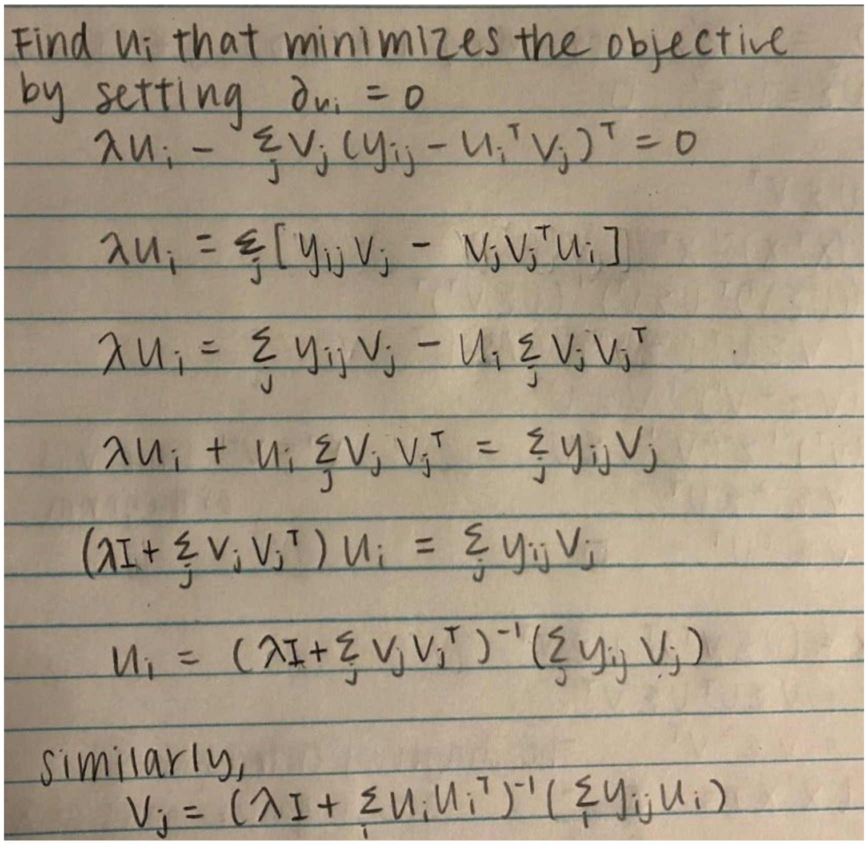
$$\partial_{u_i} = \frac{\lambda}{2}(2u_i) + \frac{1}{2}\sum_j (-v_j)(2(y_{ij} - u_i^T v_j))$$

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

similarly,

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

Problem B



Find u_i that minimizes the objective by setting $\partial_{u_i} = 0$

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

$$\lambda u_i = \sum_j [y_{ij} v_j - v_j v_j^T u_i]$$

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

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

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

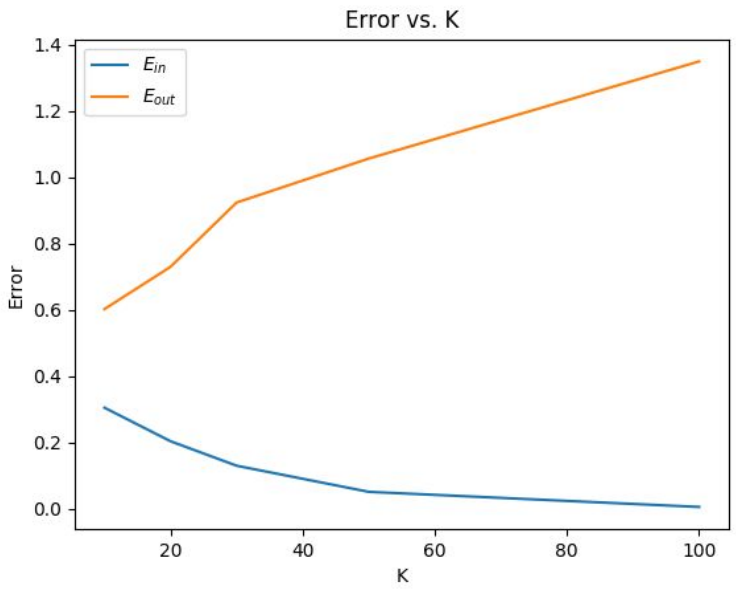
$$u_i = (\lambda I + \sum_j v_j v_j^T)^{-1} (\sum_j y_{ij} v_j)$$

similarly,

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

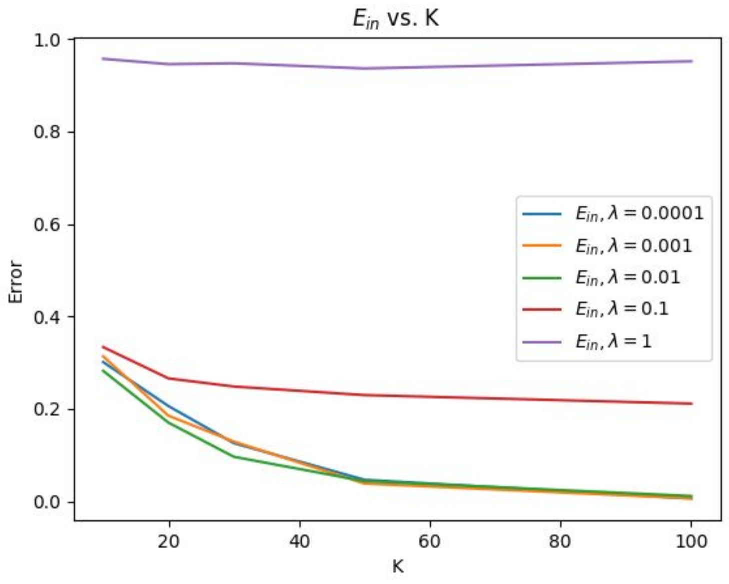
Problem C
See code

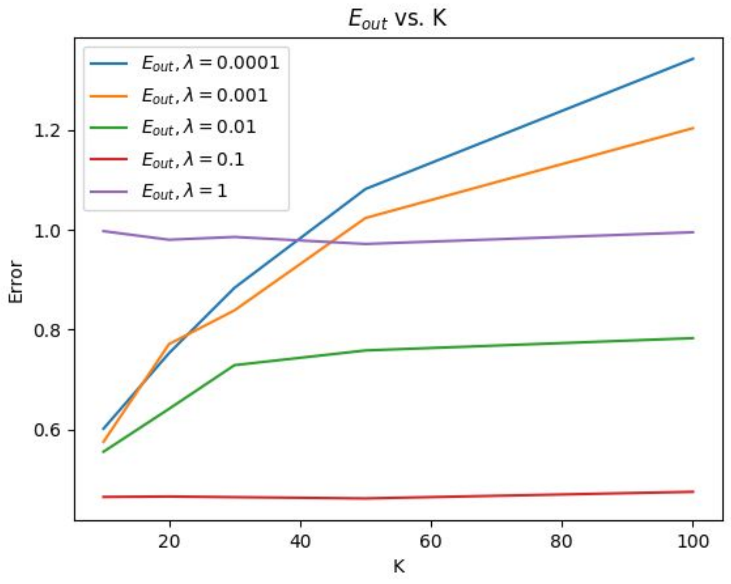
Problem D



As K increases, E_{in} goes down and E_{out} goes up. This is because using more latent factors could allow the model to fit the actual training data better (closer to the original representation), improving training accuracy, but doing so results in overfitting and causes E_{out} to go up.

Problem E





We see the same general pattern from Part D; as K increases, E_{in} goes down and E_{out} goes up. We can see that regularization slackens such increase or decrease; E_{in} goes down and E_{out} goes up at slower rates with increasing regularization strengths, such that at $\lambda = 1$, E_{in} and E_{out} stay the same at 1.0 (since the model can't learn anything) regardless of the K values.

3 Word2Vec Principles

Problem A

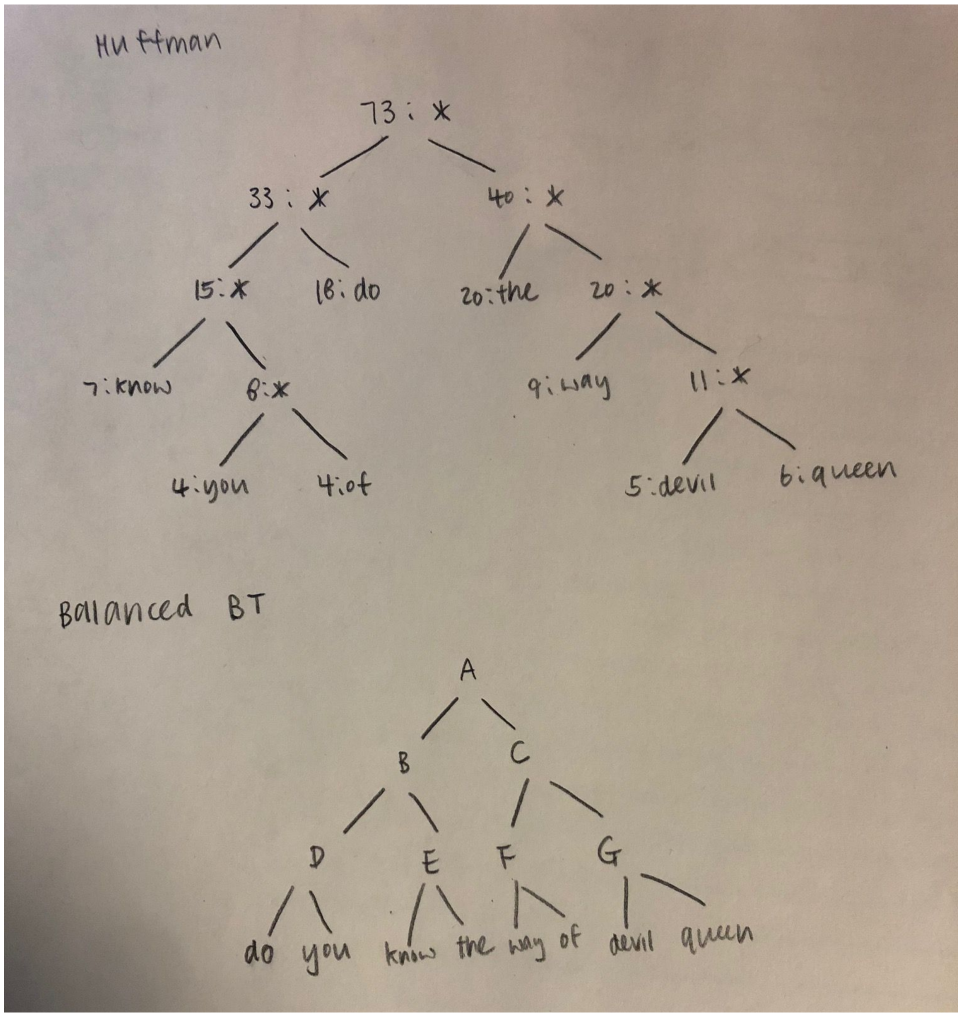
$$\partial U_{w_O}\left[\log\left(\frac{e^{U_{w_O}^T V_{w_I}}}{\sum_{w=1}^{W} e^{U_w^T V_{w_I}}}\right)\right]$$

$$= V_{w_I} - \frac{V_{w_I}}{\sum_{w=1}^{W} e^{U_w^T V_{w_I}}} \cdot e^{U_{w_O}^T V_{w_I}}$$

$$\partial V_{w_I} = U_{w_O} - \frac{1}{\sum_{w=1}^{W} e^{U_w^T V_{w_I}}}\left(\sum_{w=1}^{W} e^{U_w^T V_{w_I}} \cdot U_w\right)$$

u_w and v_w above are input and output vector representations of a word w. The asymptotic time complexity of computing a gradient for each w_O, w_I pair is dominated by the numerator term in ∂_{v_wI}; for each w in {1, …, W} (W), it calculates the dot product of u_w and v_{wI} (D) and multiplies that value by u_w (D), so the asymptotic time complexity of calculating a gradient for each pair is O (W * D). Time complexity of calculating all gradients for all W * W input and output vector pairs would scale by O (W^3 * D).

Problem B



The expected representation length is 2.74 for the Huffman tree is 2.74 and 3 for the balanced binary tree.

Problem C

The value of the training objective increases as D increases, as having larger dimension representations could improve training accuracy. However, using a very large D may be computationally expensive and cause the model to overfit the training data.

Problem D

See code

Problem E

The dimension of the weight matrix of the hidden layer is 308 x 10. 308 is the number of unique words in our text file and 10 is the number of hidden units, or the embedding dimension.

Problem F

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

Problem G

```
Pair(green, eggs), Similarity: 0.99526656
Pair(eggs, green), Similarity: 0.99526656
Pair(ham, green), Similarity: 0.98852795
Pair(likes, drink), Similarity: 0.9793115
Pair(drink, likes), Similarity: 0.9793115
Pair(there, here), Similarity: 0.976655
Pair(here, there), Similarity: 0.976655
Pair(pink, drink), Similarity: 0.9764696
Pair(ink, drink), Similarity: 0.97461367
Pair(brush, comb), Similarity: 0.9706263
Pair(comb, brush), Similarity: 0.9706263
Pair(mouse, fox), Similarity: 0.9681398
Pair(fox, mouse), Similarity: 0.9681398
Pair(shoe, off), Similarity: 0.96764153
Pair(off, shoe), Similarity: 0.96764153
Pair(wump, hump), Similarity: 0.9670307
Pair(hump, wump), Similarity: 0.9670307
Pair(with, mouse), Similarity: 0.9665417
Pair(do, like), Similarity: 0.96363837
Pair(like, do), Similarity: 0.96363837
Pair(red, blue), Similarity: 0.9569321
Pair(blue, red), Similarity: 0.9569321
Pair(foot, off), Similarity: 0.9567155
Pair(pop, hair), Similarity: 0.9558503
Pair(hair, pop), Similarity: 0.9558503
Pair(fun, another), Similarity: 0.95519215
Pair(another, fun), Similarity: 0.95519215
Pair(funny, things), Similarity: 0.95448846
Pair(things, funny), Similarity: 0.95448846
Pair(dont, sit), Similarity: 0.95405376
```

Problem H

For almost all the pairs, their reversed version (i.e. (a, b) and (b, a)) appears immediately below with the same similarity value. Some pairs have words that rhyme with each other (i.e. wump, hump), and some others have words that have similar contexts (i.e. red, blue), which makes sense based on the way Dr. Seuss poem was written.