6 LATE HOURS

1 SVD and PCA [35 Points]

Problem A [3 points]:

Solution A:

A) We have X, an $N \times N$ matrix. We also have $X = U \Sigma V^T$

So, $X^T = V \Sigma U^T$

Thus, we have $XX^T = U \Sigma V^T V \Sigma U^T$

$$XX^T = U \Sigma^2 U^T$$

We recognize this as the definition of PCA.

So by definition U are the principal components of X and Σ^2 is the diagonal matrix whose entries are the eigenvalues XX^T.

Σ contains the singular values of X. So, because Σ^2 are the eigen values of XX^T, the singular values of X are the square root of the eigenvalues of XX^T

Problem B [4 points]:

Solution B:

B. Intuitively, the eigenvalues of the PCA of XX^T are non-negative because the entries are variance of components and variance is always non-negative. Mathematically, we know the eigenvalues of XX^T are the square of the singular values of X and any number squared is non-negative.

Problem C [5 points]:

Solution C:

C. We want to show that the trace is invariant under cyclic permutations for any number of square matrices.
We will show this for 2 matrices at first as the hint suggests and then generalize.
Note that $tr(AB) = \sum_{i=1}^{N} (AB)_{ii}$

$$\Rightarrow tr(AB) = \sum_{i=1}^{N} \sum_{j=1}^{N} A_{ij} B_{ji}$$

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

$$= \sum_{j=1}^{N} (BA)_{jj}$$

$$tr(AB) = tr(BA)$$

Now we can take $tr(ABC)$ and describe $D = BC$.
So, we have $tr(AD) = tr(DA)$ which exands to $tr(ABC) = tr(BCA)$, so we see we can generalize any number of square matrices to the 2 matrix case. So we have shown that trace is invariant under cyclic permutations of any number of square matrices.

Problem D [3 points]:

Solution D:

D. We need to store the truncated SVD with the k largest singular values of Σ and the corresponding columns of U and V. We need k values from Σ, the k largest which are in the diagnol. We need to store $N \times k$ values from U and $N \times k$ values from V to get the data from the corresponding columns. This requires storage of $k(2N+1)$ values in total. For values of $k \leq \frac{N^2}{2N+1}$ it is more efficient to store the truncated SVD.

Problem E [3 points]: .

Solution E:

E. Assuming $D > N$ and X has rank N, we want to show $U\Sigma = U'\Sigma'$.

U is $D \times D$, Σ is $D \times N$

U' is $D \times N$, Σ' is $N \times N$

So we can see $U\Sigma$ and $U'\Sigma'$ are both $D \times N$.

So we have

$$U = \begin{bmatrix} u_{00} & \cdots & u_{0D} \\ \vdots & \ddots & \\ u_{D0} & \cdots & u_{DD} \end{bmatrix} \qquad \Sigma = \begin{bmatrix} s_{00} & 0 & 0 & \cdots \\ 0 & s_{11} & 0 & \cdots \\ 0 & & \ddots & \vdots \\ \vdots & & & s_{NN} \\ \vdots & & & 0 \\ 0 & \cdots & & 0 \end{bmatrix}$$

$$U' = \begin{bmatrix} u'_{00} & \cdots & u'_{0N} \\ \vdots & \ddots & \vdots \\ u'_{DN} & & u_{DN} \end{bmatrix} \qquad \Sigma' = \begin{bmatrix} s_{00} & & & 0 \\ & s_{11} & & \\ & & \ddots & \\ 0 & & & s_{NN} \end{bmatrix}$$

Thus, $U\Sigma = U'\Sigma'$

Problem F [3 points]:

Solution F:

F. A Matrix U' is orthogonal if $U'U'^T = U'^T U' = I$

We know U' is $D \times N$ so U'^T is $N \times D$.

Thus $U'U'^T$ is $D \times D$ and $U'^T U'$ is $N \times N$

so $U'U'^T$ cannot be equal to $U'^T U'$.

Problem G [4 points]:

Solution G:

G. We know the columns of U' are orthonormal and the rows of U'^T are orthonormal and when two corresponding orthonormal vectors are multiplied together the result is 1 and a orthonormal column with a non corresponding row results in 0. Thus, clearly, $U'^T U' = I_{N \times N}$. $U' U'^T \neq I_{D \times D}$ because the columns of U', not the rows, are orthonormal. Thus, we are not multiplying orthonormal vectors together so we won't get $I_{D \times D}$.

Problem H [4 points]:

Solution H:

H. Assuming that Σ is invertible we want to show $X^+ = V\Sigma^+U^T$ is equivalent to $V\Sigma^{-1}U^T$. Note that Σ is a diagonal matrix with σ_i entries on the diagonal and 0 everywhere else. Note then that Σ^{-1} will have $\frac{1}{\sigma_i}$ at every diagonal entry and 0 everywhere else. From lecture we know σ^+ is when the entry is $1/\sigma$ if $\sigma > 0$ and 0 otherwise and thus, we see that $\Sigma^{-1} = \Sigma^+$. So, $V\Sigma^+U^T = V\Sigma^{-1}U^T$ as desired.

Problem I [4 points]:

Solution I:

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

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

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

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

Problem J [2 points]:

Solution J:

J. Part (I) is more prone to numerical error, $X^T X$ can be descibed as $V \Sigma^2 V^T$ that means the condition number of I is higher than II because it's the square of II. Because

$$\left(\frac{\sigma_{max}(A)}{\sigma_{min}(A)}\right)^2 > \frac{\sigma_{max}(A)}{\sigma_{min}(A)}$$

Part I is therefore more unstable and harder to calculate. This makes sense because calculating inverses is difficult.

2 Matrix Factorization [30 Points]

Problem A [5 points]:

Solution A:

B. To find the optimal u_i we set $\partial u_i = 0$ and solve for u_i. So, we have

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

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

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

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

For optimal v_j we set $\partial v_j = 0$

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

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

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

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

Problem B [5 points]:

Solution B:

B. To find the optimal u_i we set $\partial u_i = 0$ and solve for u_i. So, we have

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

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

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

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

For optimal v_j we set $\partial v_j = 0$

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

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

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

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

Problem C [10 points]:

Solution C: *See 2D.py and prob2utils.py for the solution code.*

Problem D [5 points]:

Solution D:

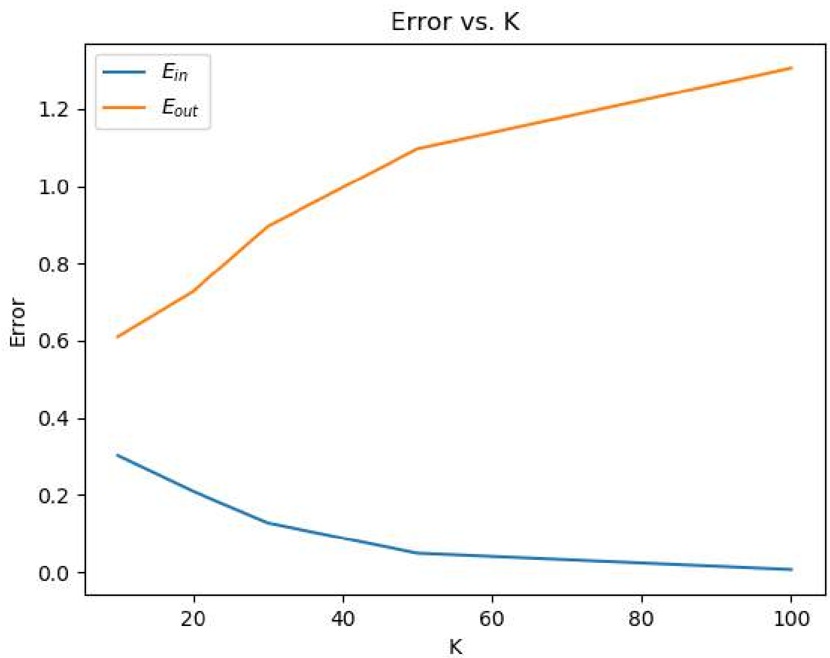


Figure 1: Unregularized factorization

As K increases Eout goes up and Ein goes down. This makes sense because as the number of latent factors increases, the more the complex the model becomes. The model increases in complexity because K incresases the dimensionality of the problem. This clearly leads to overfiting that we can see in the graph.

Problem E [5 points]:

Solution E:

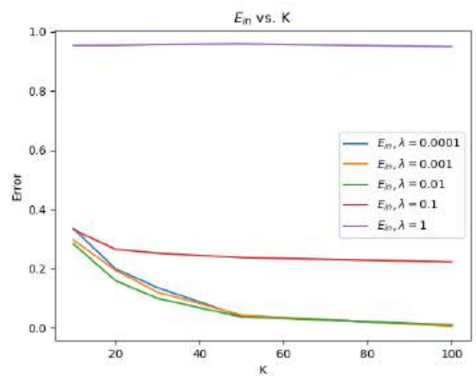


Figure 2: E_{in} vs k for different λ

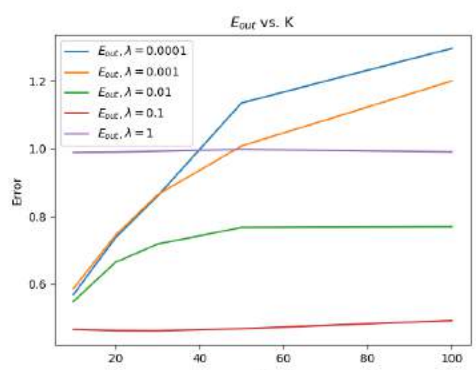


Figure 3: E_{out} vs k for different λ

With the Ein graphs we see regularization of 1 causes underfitting and that the rest see a decrease in Ein as the complexity of the model increases. With Eout we see that regularization of 1 is super underfit and that regularization of .1 produces the best results. As regularization decreses we get a plot that looks more and more similar to the Eout from the previous part which was unregularized. For regularization under .1 we see that as the number of latent factors increases overfitting starts to occur.

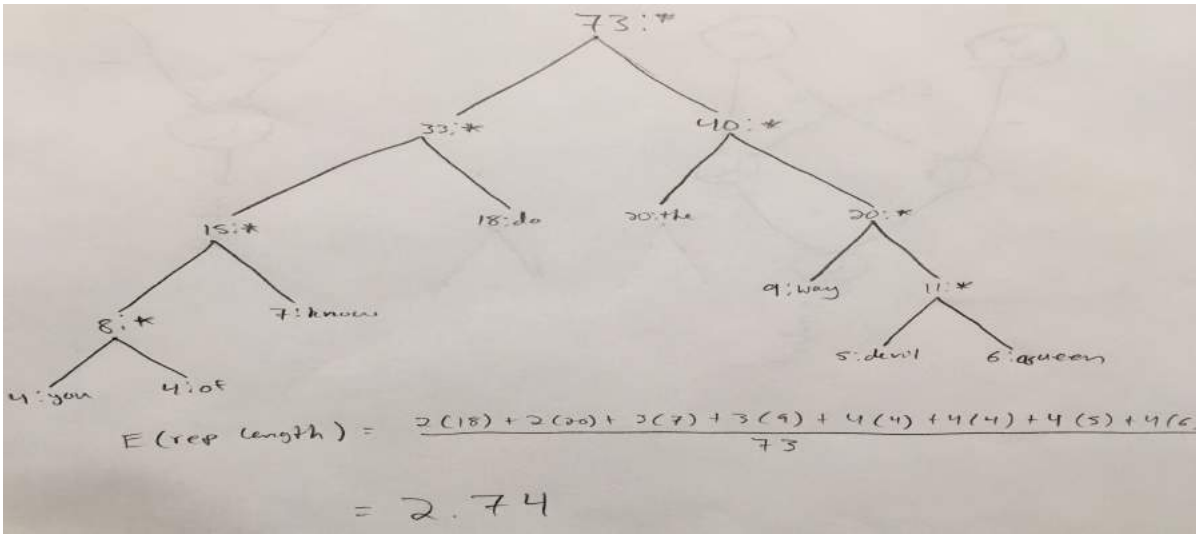
3 Word2Vec Principles [35 Points]

Problem A [5 points]:

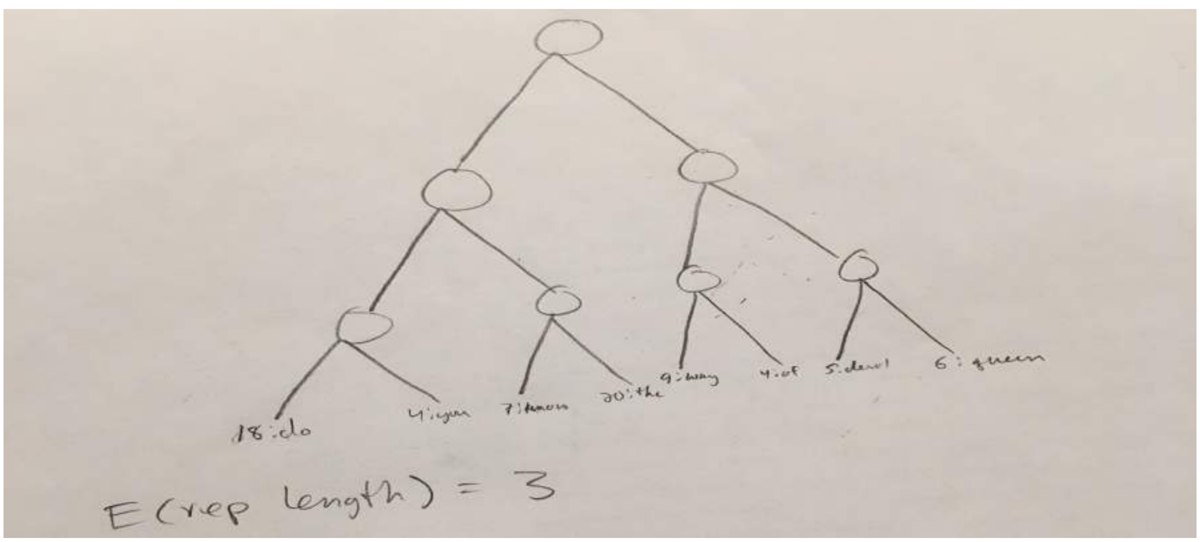
Solution A: *Computing these gradients scales approximately linearly with W and D. Each new word will be an input word once and an output approximately 2*D times. So 2*D more pairs will get made. With an increase in D we see that it will create 2 * W more pairs because it increases the window by one on each side of the input word.*

Problem B [10 points]:

Solution B: *Huffman tree*



Binary tree



Problem C [3 points]:

Solution C: *I would expect the calue of the training objective to increase as D increases. Too large of a D would be computationally expensive and inefficient and could lead to overfitting which is undesirable.*

Problem D [10 points]:

Solution D: *See solution code in P3C.py*

Problem E [2 points]:

Solution E: *308 x 10*

Problem F [2 points]:

Solution F: *10 x 308*

Problem G [1 points]:

Solution G:

```
Pair(green, eggs), Similarity: 0.99271554
Pair(eggs, green), Similarity: 0.99271554
Pair(likes, drink), Similarity: 0.98298174
Pair(drink, likes), Similarity: 0.98298174
Pair(box, goat), Similarity: 0.9789746
Pair(goat, box), Similarity: 0.9789746
Pair(fox, goat), Similarity: 0.976961
Pair(boat, goat), Similarity: 0.9768292
Pair(some, slow), Similarity: 0.974435
Pair(slow, some), Similarity: 0.974435
Pair(there, here), Similarity: 0.97167295
Pair(here, there), Similarity: 0.97167295
Pair(tree, goat), Similarity: 0.9714184
Pair(samiam, anywhere), Similarity: 0.9709146
Pair(anywhere, samiam), Similarity: 0.9709146
Pair(brush, comb), Similarity: 0.9708585
Pair(comb, brush), Similarity: 0.9708585
Pair(wink, drink), Similarity: 0.9693825
Pair(not, box), Similarity: 0.96915597
Pair(five, eight), Similarity: 0.96764493
Pair(eight, five), Similarity: 0.96764493
Pair(mouse, fox), Similarity: 0.9672581
Pair(them, or), Similarity: 0.96560454
Pair(or, them), Similarity: 0.96560454
Pair(ham, green), Similarity: 0.9652538
Pair(blue, old), Similarity: 0.9648802
Pair(old, blue), Similarity: 0.9648802
Pair(heads, grows), Similarity: 0.9645252
Pair(grows, heads), Similarity: 0.9645252
Pair(in, dark), Similarity: 0.96441853
```

Problem H [2 points]:

Solution H: *One pattern is that if (x, y) shows up as similar then it is very likely that (y, x) also shows up as a pair right after it. Another thing to notice is that common and well know Dr. Seuss phrases like green eggs and anywhere samiam occur in this set. Some odd pairs like (blue, old) appear which could be a function of it being Dr. Seuss writing again.*