Homework 6

1 SVD and PCA

Question A:

If $Y = X^T$ and SVD of $Y = U\Sigma V^T$ then $X = Y^T$ and $X = \left(U\Sigma V^T\right)^T = V\Sigma U^T$. Now we have:

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

If we perform PCA on X, we diagonalize XX^T i.e. find U and Λ such that $XX^T = U\Lambda U^T$. In this case, we see that this U is V, $\Lambda = \Sigma^2$ and so the columns of V are the principal components of X.

Question B:

We seek to prove that SVD is the best rank at-most-k approximation in the Frobenius norm.

$$A_k = argmin_{rank(B)\leq k}\|A - B\|_F = argmin_{rank(B)\leq k}\left\|U\Sigma V^T - B\right\|_F$$

Converting this by multiplying by an orthogonal matrix that keeps the Frobenious Norm the same and which does not change the rank of the matrix, we get :

$$argmin_{rank(B)\leq k}\left\|\Sigma - U^T BV\right\|_F$$

Going off the Piazza post, we know that to minimize $\left\|\Sigma - U^T BV\right\|_F$, $U^T BV$ should be a diagonal matrix with the first k elements of the diagonal of Σ and other diagonal elements 0 because adding off-diagonal elements to $U^T BV$ will not help minimize $\left\|\Sigma - U^T BV\right\|_F$, $U^T BV$. In other words, we minimized:

$$\sum_i (\Sigma_{i,i} - D_{i,i})^2$$

where $D = U^T BV$. This implies $UDV^T = B$ and thus we get that $A_k = \sum_{j=1}^k u_j \sigma_j v_j^T = \sum_{j=1}^k \sigma_j u_j v_j^T$.

2 Matrix Factorization

Question A:

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

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

Question B:
Setting $\partial_{u_i} = 0$ and $\partial_{v_j} = 0$ to find critical points of our regularized square error , we get the following:

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

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

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

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

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

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

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

Similarly, for ∂_{v_j}:

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

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

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

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

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

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

Question C:
See attached code.

Question D:

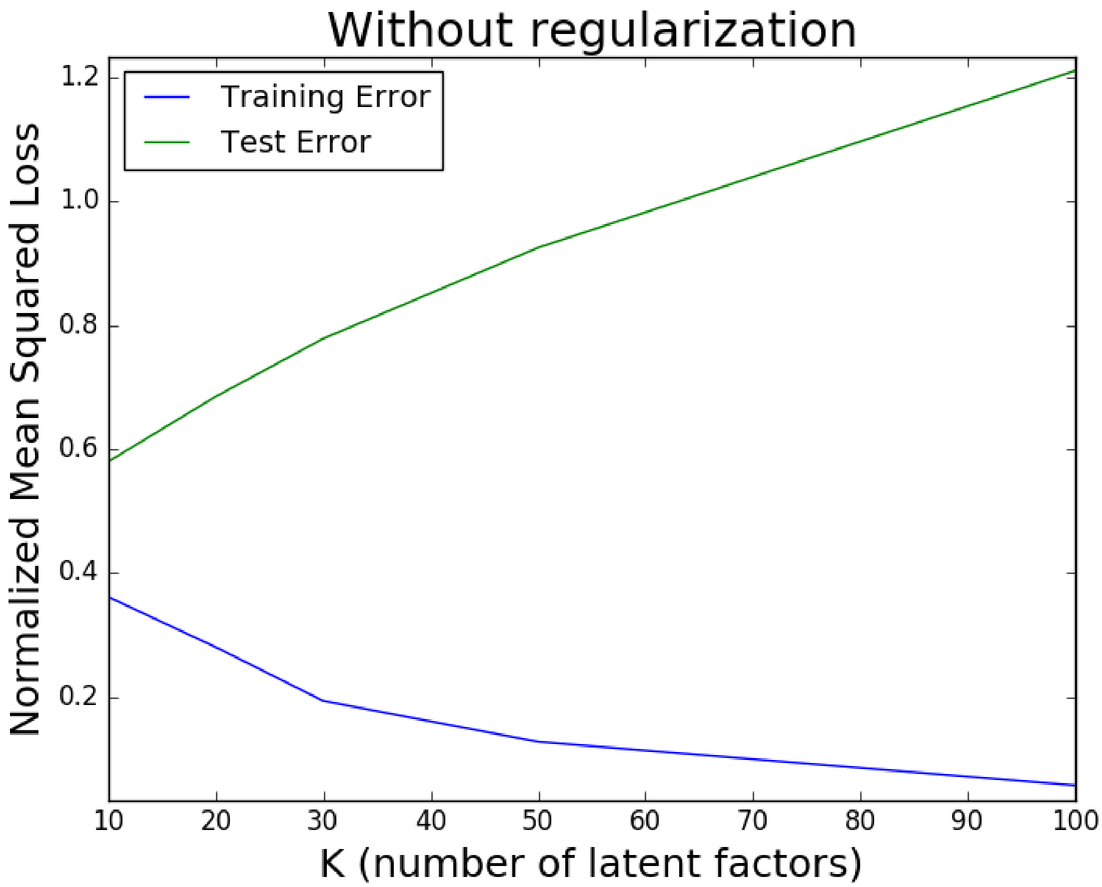

Figure 1: E_{in} and E_{out} vs. k with $\lambda = 0$

We observe that as k increases, E_{in} decreases while E_{out} increases. We know that k, the number of latent factors, determines the level of dimensionality reduction such that a small k represents small matrices U and V and therefore a large reduction in dimensionality of our original input space. In these terms, the more we lower k and the more we reduce the dimensions of the input space, the more we are summarizing the original data. This summarization can prevent the model from overfitting to the training set because in summarizing the input data, it is not fitting to the peculiarities of the data itself. Thus, we see that for low k where we summarize the input data the most, we have low E_{out} and high E_{in} and as we increase k, we are getting more likely to overfit to the training input data, so E_{in} goes down at the cost of generalization where E_{out} increases.

Question E:

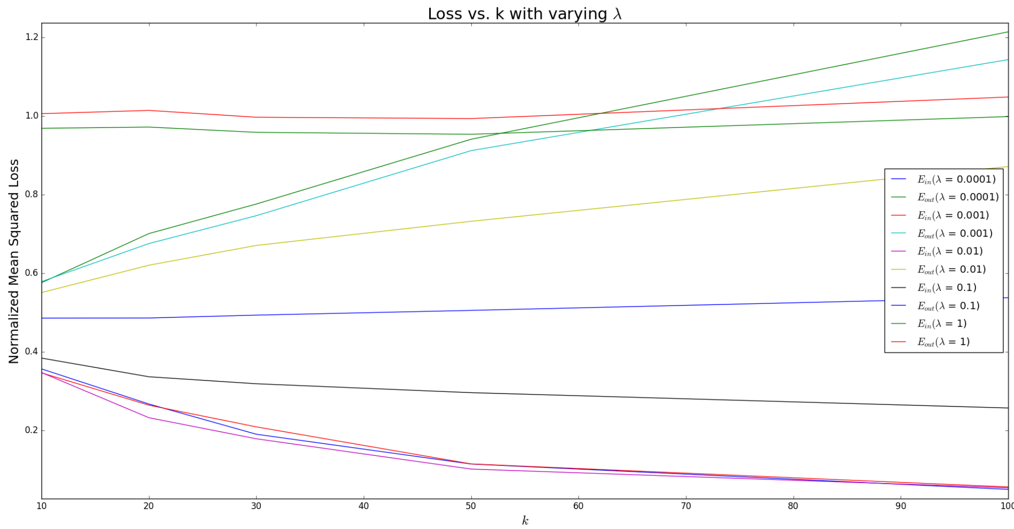

Figure 2: E_{in} and E_{out} for $k = 10, 20, 30, 50, 100$ for each $\lambda = 0.0001, 0.001, 0.01, 0.1, 1$

We observe similar trends to the previous graph where increasing k typically decreases E_{in} and increase E_{out}. When λ is too large, such as when $\lambda = 1$, we see that we have high E_{in} and E_{out}, a sign of underfitting. As we decrease λ, the closer the results resemble that of the previous graph where $\lambda = 0$. Out of the five values for λ, the best results seem to come from when $\lambda = 0.1$ where we have the lowest E_{out}. At $\lambda = 0.1$, we seem to combat both underfitting and overfitting.

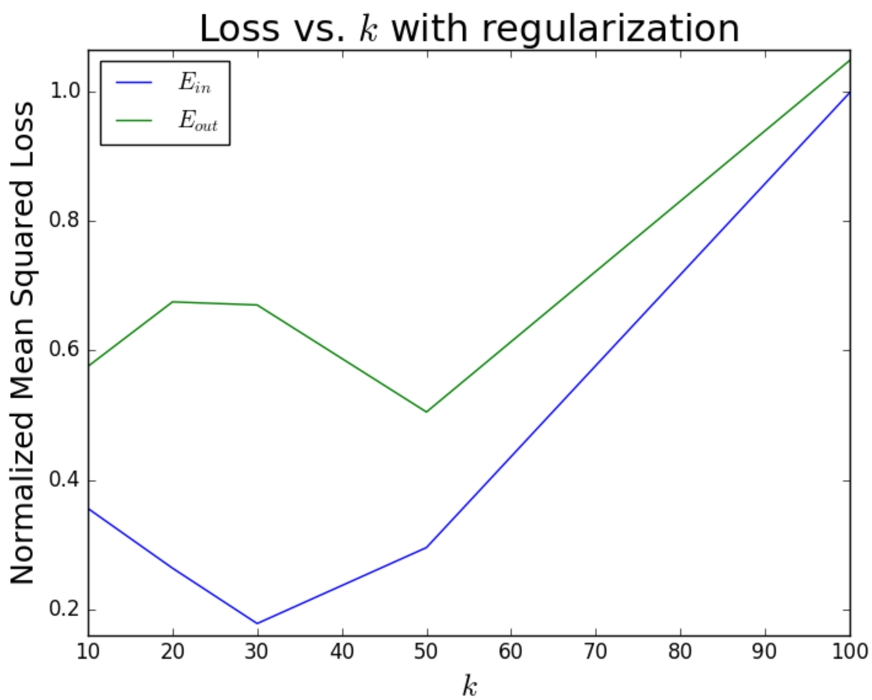

Figure 3: E_{in} and E_{out} for k = 10, 20, 30, 50, 100 for λ = 0.0001, 0.001, 0.01, 0.1, 1 respectively (as mentioned in Piazza post). Not sure if this graph is required, but just added in case.

3 Word2Vec Principles

Question A:

We have: $log\ p(w_O|w_I) = log\frac{exp(v'^T_{w_O} v_{w_I})}{\sum_{w=1}^{W} exp(v'^T_{w} v_{W_I})}$.

Computing $\nabla\ log\ p(w_O|w_I)$ scales linearly with W. First off, we note that computing the derivative of the numerator takes constant time. Now, we analyze the denominator of $p(w_O|w_I)$ and observe that as we increase W, we increase the number of terms in the summation for the denominator. Each time we increase W by 1, then we have 1 more additional term to differentiate in the denominator. Thus, computing $\nabla\ log\ p(w_O|w_I)$ is O(W).

Question B:

Because $p(w_O|w_I)$ involves multiplying v^T and v and $v \epsilon R^D$, as we increase D, we also increase the computational complexity of computing the training objective. For large D, it may be very computationally intensive to carry out such calculations. In addition, as we increase D, we are more likely to overfit to the training set, potentially generalizing poorly. This is indeed undesirable.

Question C:

See attached code.

Question D:

i. Dimensions of weight matrix of hidden layer: 311 x 10.

ii. Dimensions of weight matrix of output layer: 10 x 311.

iii.

```
Layer 0 shape(311, 10)
Layer 1 shape(10, 311)
Pair(could, would), Similarity: 0.987127
Pair(would, could), Similarity: 0.987127
Pair(house, car), Similarity: 0.982678
Pair(car, house), Similarity: 0.982678
Pair(goat, boat), Similarity: 0.980282
Pair(boat, goat), Similarity: 0.980282
Pair(may, goat), Similarity: 0.977274
Pair(fox, goat), Similarity: 0.976378
Pair(brush, comb), Similarity: 0.974029
Pair(comb, brush), Similarity: 0.974029
Pair(mouse, fox), Similarity: 0.972813
Pair(anywhere, samiam), Similarity: 0.971517
Pair(samiam, anywhere), Similarity: 0.971517
Pair(tree, boat), Similarity: 0.970141
Pair(rain, car), Similarity: 0.970048
Pair(eat, not), Similarity: 0.966097
Pair(not, eat), Similarity: 0.966097
Pair(heads, ear), Similarity: 0.95831
Pair(ear, heads), Similarity: 0.95831
Pair(slow, some), Similarity: 0.957296
Pair(some, slow), Similarity: 0.957296
Pair(you, eat), Similarity: 0.956688
Pair(eggs, green), Similarity: 0.954727
Pair(green, eggs), Similarity: 0.954727
Pair(ham, green), Similarity: 0.954214
Pair(bump, had), Similarity: 0.953407
Pair(had, bump), Similarity: 0.953407
Pair(or, anywhere), Similarity: 0.952945
Pair(here, there), Similarity: 0.951908
Pair(there, here), Similarity: 0.951908
```

Figure 4: Top 30 pairs of most similar words

I noticed that we have many repeats where Pair(a,b) and Pair(b,a) are both included. This makes sense given that if word a is often seen with word b, then word b is often seen with word a. If we look at these pairs, many of the pairing words make up well-known Dr. Seuss phrases. For example, we have the pairs ("ham", "green") and ("eggs", "green") which are words used in the famous phrase "green eggs and ham." Other examples include the pair ("here", "there") as the phrase "here and there" is repeated many times in Dr. Seuss' poems. Another common theme between these top similar pairs are that the words rhyme such as in ("could", "would") and ("goat, boat"). This also makes sense given that Dr. Seuss wrote poems that have a repetitive rhyming scheme such that words that rhyme are typically neighbored by the similar words. Additionally, many of these pairs consist of two of the same type of objects such as two animals or two body parts.

Question E:
To calculate the expected frequency of a word w_i after subsampling, we simply subtract the number of times we expect to discard the word from $f(w_i)$, the frequency of the word. This is formulated as:

$$\begin{aligned}
\mathbb{E}[f(w_i)] &= f(w_i) - P(w_i)f(w_i) \\
&= f(w_i)(1 - P(w_i)) \\
&= f(w_i)(1 - (1 - \sqrt{\frac{t}{f_{w_i}}})) \\
&= f(w_i)\sqrt{\frac{t}{f_{w_i}}} \\
&= \sqrt{tf(w_i)}
\end{aligned}$$

Because t is a constant, we have that if $f(w_i) > f(w_j)$ before subsampling, then $\sqrt{tf(w_i)} > \sqrt{tf(w_j)}$. Therefore $\mathbb{E}[f(w_i)] > \mathbb{E}[f(w_j)]$ after subsampling and thus ranking of frequencies is preserved in expectation by the subsampling policy.