I am using 11 late hours on this set.

1a. We have

$$XX^T = U\sum V^T V \sum V^T$$
$$= U\sum^2 U^T$$

Thus, the eigenvalues of XX^T are the singular values of X squared. We know that the columns are the principle components of X because we have the PCA solution and the diagonal of $\sum^2$ is the singular values of X squared.

1b. The eigenvalues of the PCA represent variance, so this value is nonnegative. It does not make sense to have negative variance. We also showed in part 1a that this is a squared value. Any value squared is at minimum zero, so these values are nonnegative.

1c. We can show this for two matrices:

$$Tr(AB) = \sum_{i=1}^{N}(AB)_{ii} = \sum_{i=1}^{N}(\sum_{j=1}^{N} a_{ij}b_{ji}) = \sum_{j=1}^{N}(\sum_{i=1}^{N} b_{ji}a_{ij}) = \sum_{j=1}^{N}(BA)_{jj} = Tr(BA)$$

Now, let D = AB:

$$Tr(CAB) = Tr(CD) = Tr(DC) = Tr(ABC)$$

Or E = BC:

$$Tr(ABC) = Tr(AE) = Tr(EA) = Tr(BCA)$$

This shows that the case of three matrices is equal. We can treat any many of matrices as a product to a single matrix, which always reduces to the case of two matrices that was proven.

1d. We would need the k values from $\sum$, the kN values from U and V^T each. Therefore, we would need $k + 2kN$ values for the truncated SVD. We have N^2 values when we store the whole matrix, so we need when the number of values is fewer.

$$k + 2kN < N^2$$

$$k(1 + 2N) < N^2$$

$$k < \frac{N^2}{1 + 2N}$$

1e. Since $\sum$ has only diagonal values, we can take off the extra rows of zeros after the last diagonal value to get $\sum'$. Thus, when we multiply $U\sum$, the zero rows on the bottom of $\sum$ will cancel out any values that were stored after N columns in U. Therefore, the only values left in $U\sum$ are the same values that would appear in $U'\sum'$. Thus,

$$U\sum = U'\sum'$$

1f. We have that A is orthogonal iff $AA^T = A^TA = 1$. Let us take a matrix B that has dimensions a x b where a ≠ b. If we take BB^T, we get dimensions of a x a. If we take B^TB, we get dimensions of b x b. Therefore, we have $BB^T \neq B^TB$ which contradicts the fact we have. Since U' is not a square, there is not possible way for it to be orthogonal by the logic beforehand.

1g. We know that U’ is just a slimmed down version of U. We have that U’ has some columns cut off, but it has the same number of N rows. So, this means that when we have $U'^T U'$, the same results come as when we compute $U^T U$, which is I_{NxN}. However, we do not have $U' U'^T = I_{DxD}$ or else it would be orthogonal, which we have already shown is not true.

1h. We must show that $\sum^{+} = \sum^{-1}$:

$$\sum = \begin{bmatrix} \sigma_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma_D \end{bmatrix}$$

$$\sum^{-1} = \begin{bmatrix} \frac{1}{\sigma_1} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \frac{1}{\sigma_D} \end{bmatrix}$$

We can calculate the pseudoinverse as shown in class:

$$\sum^{+} = \begin{bmatrix} \frac{1}{\sigma_1} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \frac{1}{\sigma_D} \end{bmatrix}$$

Thus, they are equal, so we have

$$V\sum^{+}U^T = V\sum^{-1}U^T$$

1i. We can show this by using the definition of X:

$$(X^T X)^{-1} X^T$$

We have

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

So,

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

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

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

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

1j. The second version is highly prone to numerical errors. Calculating the inverse of $X^T X$ is extremely computational and allows a lot of room for error. Calculating the inverse of $\sum$ is very easy to do, since the condition number is always smaller than the one given in the other case. There is only one value per row and column in $\sum$, while X can have many values by itself.

2a. We can simply compute this with the exponent and chain rule with derivatives:

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

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

2b. To do this, we can set the gradients we found in 2a equal to zero, while holding the other variable constant. This gives us our critical points:

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

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

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

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

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

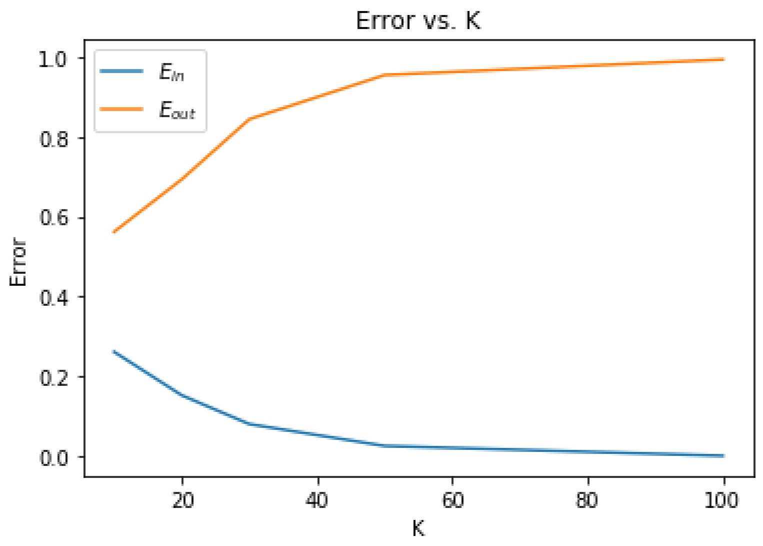
$$u_i = \sum_j v_j y_{ij} (\sum_j v_j v_j^T + \lambda I)^{-1}$$

Following the same for ∂_{v_j}:

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

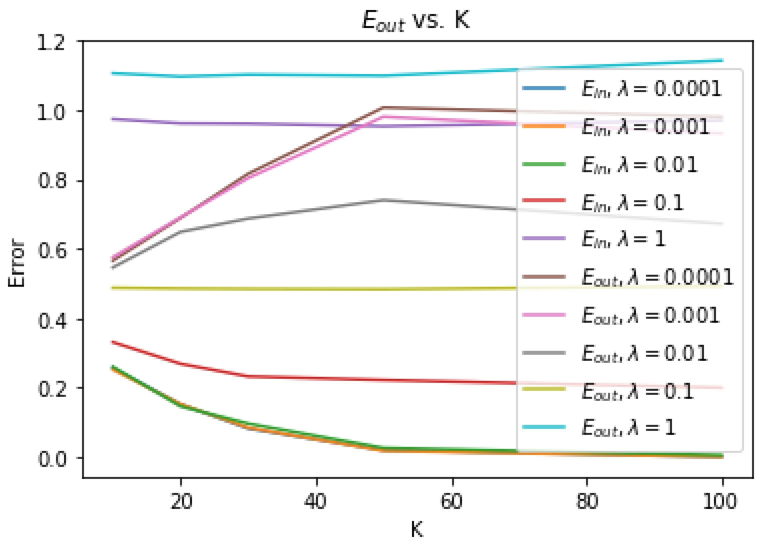
2c. Code is submitted through Moodle.

2d.



As k increases, Ein decreases while Eout increases. We know that k is the number of latent factors, and this determines the level of dimensionality reduction. Thus, when we have small k, we are summarizing the data more, which can prevent the model from overfitting. As we increase k, we generalize less, and we see that overfitting occurs.

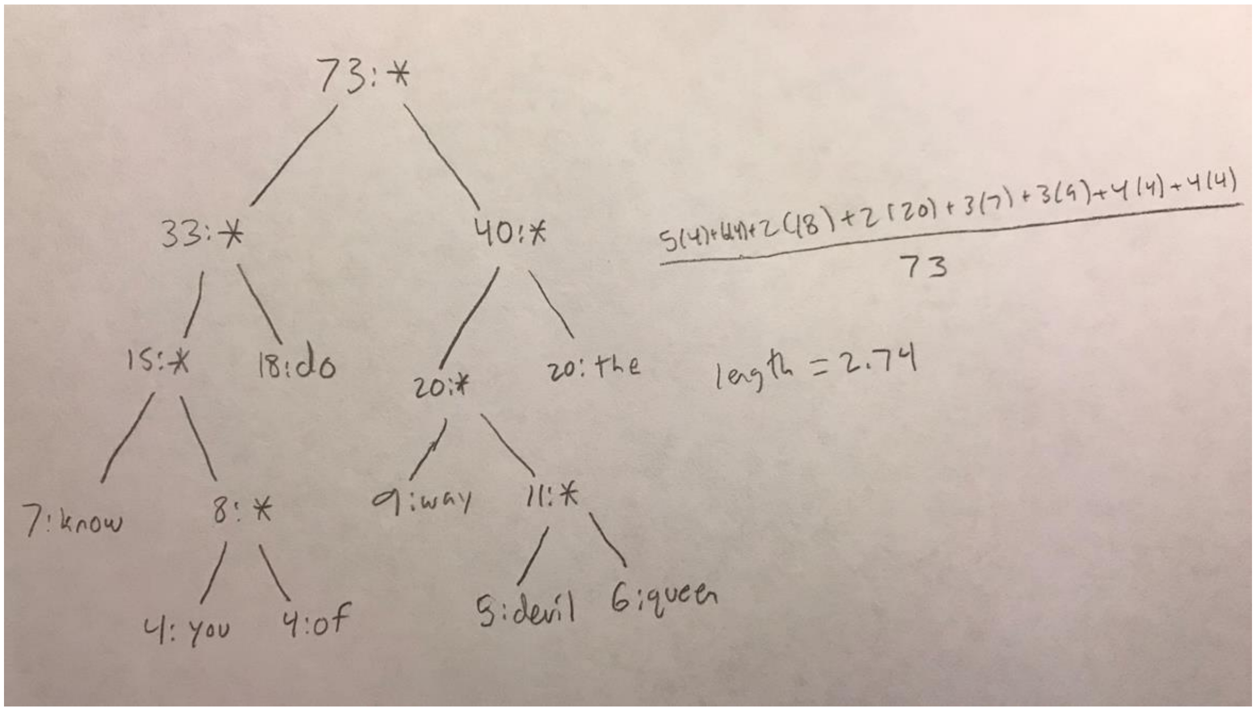
2e.

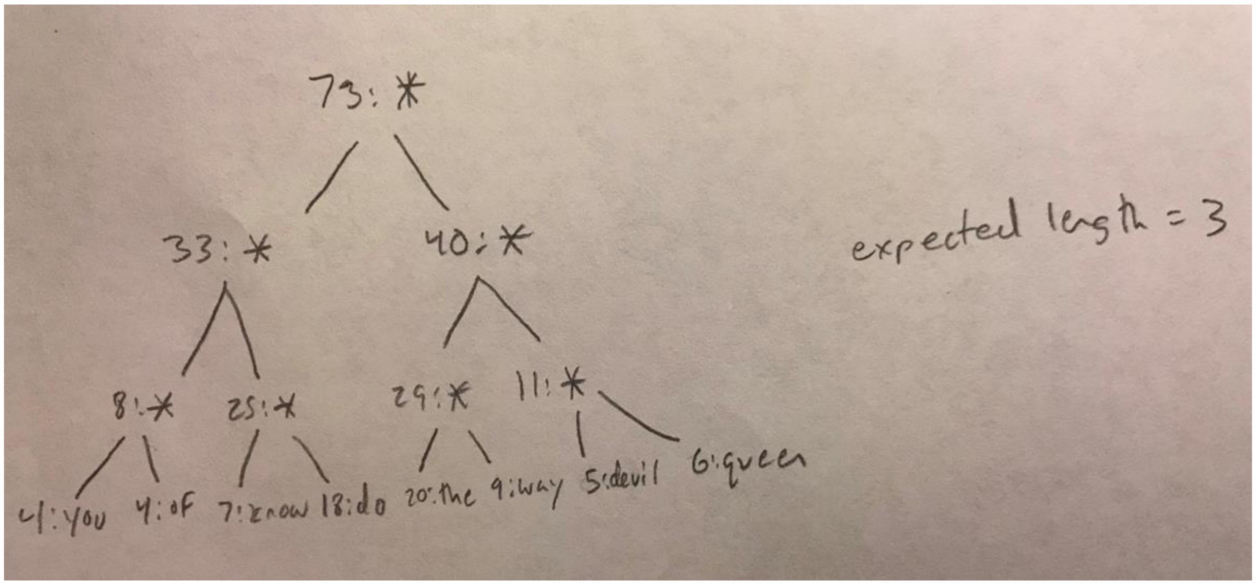


Like the previous graph, Ein decreases and Eout increases as k increases. When we have high values for lambda, we have high Ein and Eout, which is a sign of underfitting. This means that we are over regulating, which obviously has a large impact.

3a. Computing the gradients scales with W and D. If we look at the function for $p(w_O|w_I)$, we can see that we must compute a gradient for each W pair, and the gradient depends on the vectors which depend on D. These are both linear relationships, so the computation scales with O(WD) complexity.

3b.


73:*
33:*
40:*
15:*
18:do
20:*
20: the
7: know
8: *
9: way
11: *
4: you
4: of
5: devil
6: queen
5(4)+4(4)+2(18)+2(20)+3(7)+3(9)+4(4)+4(4) / 73
length = 2.74


73: *
33: *
40: *
8: *
25: *
29: *
11: *
4: you
4: of
7: know
18: do
20: the
9: way
5: devil
6: queen
expected length = 3

3c. When D increases, the value of the training objective becomes more accurate. Since $p(w_O|w_I)$ involves multiplying vectors within D, the complexity of computing the training objective increases as D increases. Once D gets large enough, it might become too complex to carry out. Also, larger values of D may cause overfitting on the training set. Since the vectors are large, generalization could become poor.

3d. See attached code

3e. Hidden layer shape: 308 x 10

3f. Output layer shape: 10 x 308

3g.

```
Layer 0 shape(308, 10)

Layer 1 shape(10, 308)

Pair(mouse, could), Similarity: 0.974375
Pair(could, mouse), Similarity: 0.974375
Pair(i, like), Similarity: 0.97124183
Pair(like, i), Similarity: 0.97124183
Pair(them, like), Similarity: 0.9618859
Pair(do, i), Similarity: 0.9539996
Pair(not, them), Similarity: 0.9531103
Pair(wump, does), Similarity: 0.95117277
Pair(does, wump), Similarity: 0.95117277
Pair(eat, them), Similarity: 0.95049214
Pair(quiet, today), Similarity: 0.94832563
Pair(today, quiet), Similarity: 0.94832563
Pair(house, like), Similarity: 0.9449521
Pair(in, i), Similarity: 0.94431114
Pair(with, anywhere), Similarity: 0.94373214
Pair(anywhere, with), Similarity: 0.94373214
Pair(box, eat), Similarity: 0.9355874
Pair(lot, see), Similarity: 0.9354582
Pair(see, lot), Similarity: 0.9354582
Pair(you, house), Similarity: 0.92692524
Pair(will, like), Similarity: 0.92413825
Pair(fox, will), Similarity: 0.9196478
Pair(five, gold), Similarity: 0.91583014
Pair(gold, five), Similarity: 0.91583014
Pair(and, think), Similarity: 0.91483045
Pair(think, and), Similarity: 0.91483045
Pair(they, hot), Similarity: 0.91198254
Pair(hot, they), Similarity: 0.91198254
Pair(would, do), Similarity: 0.91101795
Pair(hook, only), Similarity: 0.9104416
```

3h. I noticed a lot of repeats and rhyming words. Many of the results are like (mouse, could) and (could, mouse). This makes sense as the words are seen together depending on the pivot word. The other trends I saw were phrases only seen in Dr. Seuss's books. There are a lot of imaginary words like "wump," and it makes me reminisce about my childhood because I can make out the sentence from the book from those words and its pair.