Jessica Wang

Machine Learning and Data Mining: Set 5

I. SVD and PCA

Problem A.

If X is a $N \times N$ matrix, for the singular value decomposition (SVD) $X = U \sum V^T$, the columns of U are the principal components of X:

Applying SVD to matrix X gives:

$$X = U\sum V^T$$

To construct the covariance matrix:

$$XX^T = (U\sum V^T)(U\sum V^T)^T$$
$$XX^T = (U\sum V^T)(V\sum U^T)$$

Recall that $\sum$ is a diagonal matrix and U and V are orthogonal matrices which means:

$$V^T V = I$$

Therefore:

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

The columns of U are the principal components of X and the square roots of the eigenvalues of XX^T are the singular values of X.

Problem B.

The eigenvalues of the PCA of X or rather XX^T are non-negative:

Recall that the squared norm of a vector $\vec{s}$ is given by the dot product:

$$\vec{s} \cdot \vec{s} = \vec{s}^T\vec{s}$$

Let $\vec{t}$ be an eigenvector of XX^T with eigenvalue λ_t. To compute the norm squared of $X^T\ \vec{t}$:

$$\left\|X^T\ \vec{t}\right\|^2 = \left(X^T\ \vec{t}\right)^T (X\ \vec{t})$$
$$\left\|X^T\ \vec{t}\right\|^2 = \left(\vec{t}^T X^{T^T}\right)(X\ \vec{t})$$
$$\left\|X^T\ \vec{t}\right\|^2 = \left(\vec{t}^T X^T\right)(X\ \vec{t})$$
$$\left\|X^T\ \vec{t}\right\|^2 = \vec{t}^T (X^T X)\vec{t}$$

Simplifying:

$$\left\|X^T\ \vec{t}\right\|^2 = \lambda\vec{t}^T\vec{t}$$
$$\left\|X^T\ \vec{t}\right\|^2 = \lambda\left\|\vec{t}\right\|^2$$

Observe that $\left\|X^T\ \vec{t}\right\|^2$ and $\left\|\vec{t}\right\|^2$ are non-negative since squared norms must be non-negative.

As such, for the equation

$$\left\|X^T\ \vec{t}\right\|^2 = \lambda\left\|\vec{t}\right\|^2$$

to hold, λ must also be non-negative. This provides mathematical justification that the eigenvalues of the PCA of X or rather XX^T are non-negative.

An intuitive explanation for why the eigenvalues of the PCA of X or rather XX^T are non-negative: in principal component analysis, an eigenvalue is the variance of its corresponding eigenvector and variance is always non-negative. The length of an eigenvector is related to its corresponding eigenvalue and length is always non-negative too. The covariance matrix is positive semidefinite.

Problem C.

The trace is invariant under cyclic permutations (i.e. $Tr(ABC) = Tr(BCA) = Tr(CAB)$) for any number of square matrices:

We will begin by proving that the identity hold for two matrices and then generalize to any number of square matrices.

For two square matrices, let A be a $N \times N$ matrix and B be a $N \times N$ matrix:

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

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

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

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

$$Tr(AB) = Tr(BA)$$

We can replace the square matrix B by a square matrix of the same dimension formed from matrix multiplication such that $B = CD$ or $B = CDE$ or $B = CDE$ ... and the same with the square matrix A and find that the property holds for any number of square matrices. There is nothing special about the arrangement of the matrices so clearly, we could prove by induction that rearranging the order of the matrices still allows for the property to hold true. Therefore, the trace is invariant under cyclic permutations for any number of square matrices.

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 $k * (2N + 1)$. We need to store a column of U, a column of V, and one entry of $\sum$ for each of the singular values so the number of values

needed to store the truncated SVD would be $k * (2N + 1)$. When k is less than $N^2/(k * (2N + 1))$, storing the truncated SVD is more efficient that storing the whole matrix.

Problem E.

i. We assume that $D > N$ and that X has rank N. We can show that $U \sum = U' \sum'$ where $\sum'$ is the $N \times N$ matrix consisting of the first N rows of $\sum$ and U' is the $D \times N$ matrix consisting of the first N columns of U:

We observe that if we denote each i, j entry in $\sum$ as $\sum$(i, j), then when D > N, each entry of $\sum$(i, j) where D > i > N is equal to zero which means $U \sum = U' \sum'$

ii. Since U' is not square, it cannot be orthogonal since a matrix is orthogonal if $U'(U')^T = (U')^T U'$. Therefore, if U' is a $M \times N$ matrix, $U'(U')^T$ is a $M \times M$ matrix while $(U')^T U'$ is a $N \times N$ matrix.

iii. Even though U' is not orthogonal, the columns of U' are still orthonormal. Since the columns of U' are orthonormal, we know that the dot product of each column with itself is 1 and the dot product of each column with a different column is 0. Therefore, $(U')^T U' = I_{N \times N}$ since the resulting matrix is simply one's along the diagonal and zero's everywhere else. This is simply the identity matrix.

It is not true that $(U')^T U' = I_{D \times D}$ since the rank of U' is at most N and we know if two matrices A and B are multiplied with matrix multiplication, then rank(AB) is less than or equal to min(rank(A), rank(B)). However, since $D > N$ we know that $(U')^T U' = I_{D \times D}$ is not true.

Problem F.

i. We let X be a matrix of size $D \times N$ where $D > N$ with "thin" SVD $X = U \sum V^T$. We assume that X has rank N. We would simply like to show that the pseudoinverse $X^+ = V \sum^+ U^T$. We know that a diagonal matrix is invertible if and only if the entries along the diagonal are not zero such that it's inverse is given by the diagonal matrix where $\sigma_{ii}^{-1} = \frac{1}{\sigma_{ii}}$ but we observe that this is simply the definition of $\sum^+$ when $\sum$ is invertible. In other words, $\sum^+ = \sum^{-1}$ which means the pseudoinverse $X^+ = V \sum^+ U^T$ is equivalent to $V \sum^{-1} U^T$

ii. Another expression for the pseudoinverse is the least squares solution $X^{+\prime} = (X^T X)^{-1} X^T$. We can show that, assuming $\sum$ is invertible:

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

iii. The least squares solution is highly prone to numerical errors: higher condition numbers implies greater prone to numerical errors. The condition numbers for $X^T X$ is the square of those from part (i) and since they are so much larger the least squares solution is not as good.

II. Matrix Factorization

Problem A.

$$\underset{U,V}{\arg min} \quad \frac{\lambda}{2}(\|U\|_F^2 + \|V\|_F^2) + \frac{1}{2}\sum_{i,j}(y_{ij} - u_i^T v_j)^2$$

Gradients of the above regularized square error with respect to u_i and v_j:

$$\delta_{u_i} = \lambda U + \sum_{i,j}\left(y_{ij} - u_i^T v_j\right) * (-v_j)$$

$$\delta_{v_j} = \lambda V + \sum_{i,j}\left(y_{ij} - u_i^T v_j\right) * (-u_i)$$

Problem B.

$$\underset{U,V}{\arg min} \quad \frac{\lambda}{2}(\|U\|_F^2 + \|V\|_F^2) + \frac{1}{2}\sum_{i,j}(y_{ij} - u_i^T v_j)^2$$

ALS solves the problem by first fixing U and solving for the optimal V, then fixing this new V and solving for the optimal U. We observe that from before:

$$\underset{U,V}{\arg min} \quad \frac{\lambda}{2}(\|U\|_F^2 + \|V\|_F^2) + \frac{1}{2}\sum_{i,j}(y_{ij} - u_i^T v_j)^2$$

$$\delta_{u_i} = \lambda U + \sum_{i,j}\left(y_{ij} - u_i^T v_j\right) * (-v_j)$$

$$\delta_{v_j} = \lambda V + \sum_{i,j}\left(y_{ij} - u_i^T v_j\right) * (-u_i)$$

We set the gradients to zero and solve:

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

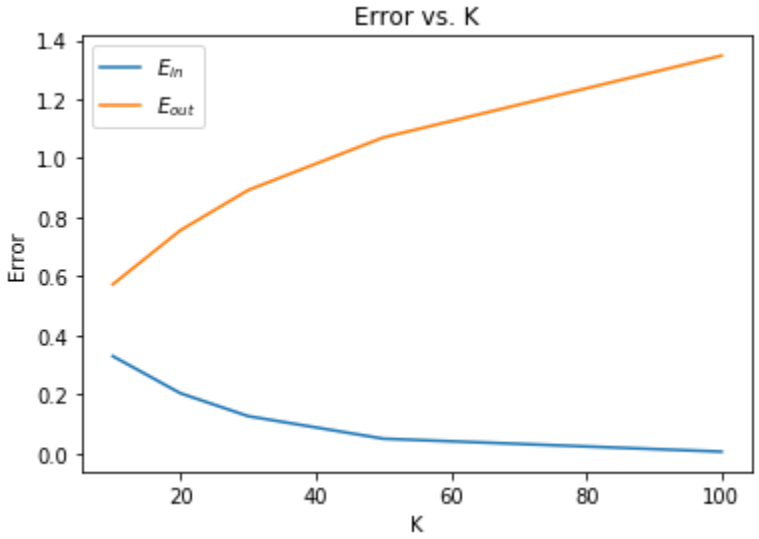
$$v_j = \left(\lambda I_k + \sum_i u_i u_i^T\right)^{-1} \left(\sum_i y_{ij} u_i\right)$$

Problem C.

See Jupyter notebook for code.

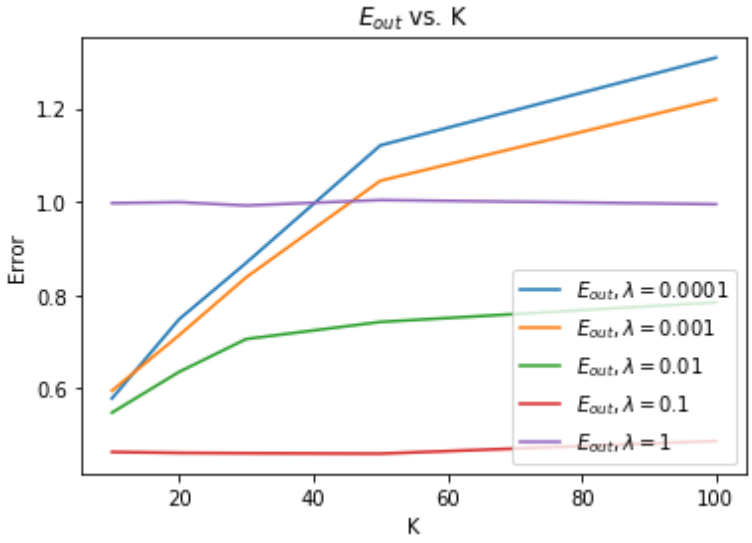
Problem D.

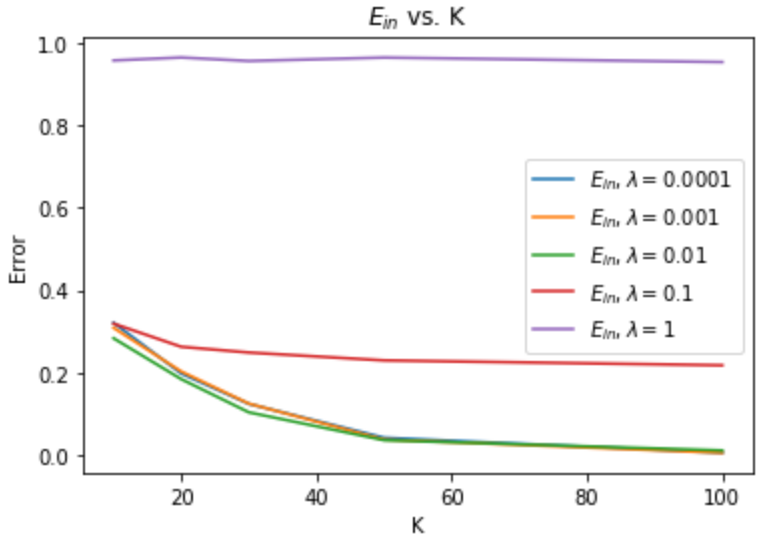
See Jupyter notebook for code. The graph is shown below. The training error decreases as the dimension K increases while the testing error increases as the dimension K increases. With a larger K, the reconstruction improves so the training error is lower but as the dimension K increases, the testing error increases, most likely due to overfitting.



Problem E.

See Jupyter notebook for code. The graphs are shown below.
We observe that for the most part, increasing the regularization term decreases the testing error and increases the training error. This is because increasing the regularization term helps to prevent overfitting. However, we see a trade off because the model is not able to learn as well when the regularization term is very high, resulting in underfitting for when lambda = 1.





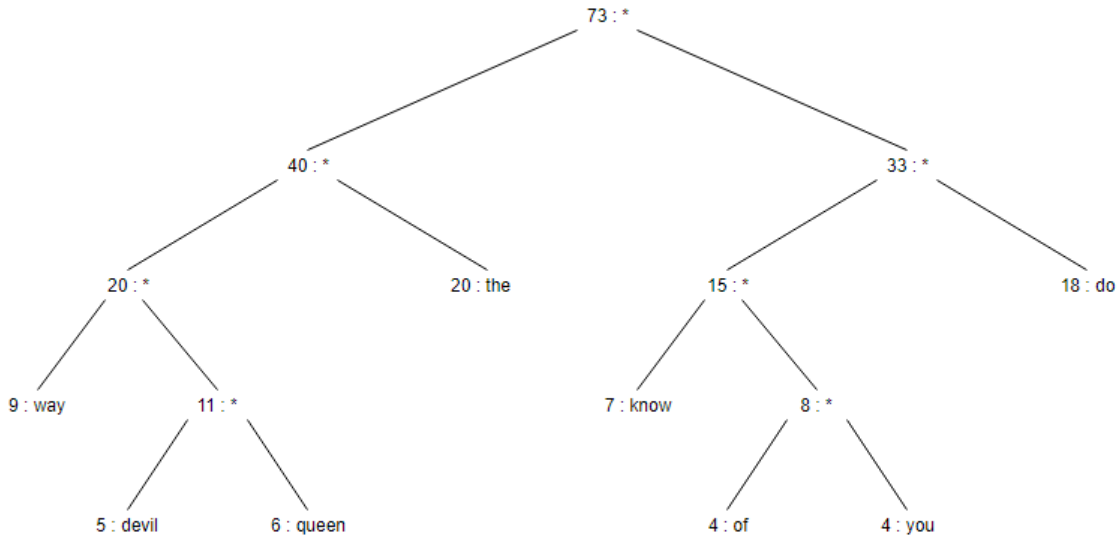
III. Word2Vec Principles

Problem A.

Computing the gradients scales W^2 with W, the number of words in the vocabulary, since we must compute ∇log p(wO|wI) for each wO, wI pair and computing these gradients scales linearly D, with D, the dimension of the embedding space.

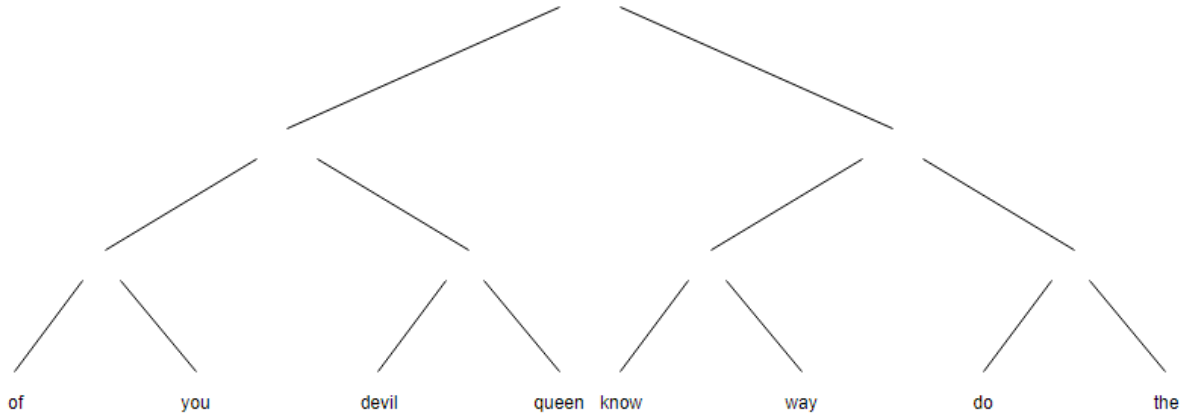
Problem B.

The Huffman tree for the words and frequencies given:



The expected representation length = [(3 * 9) + (4 * 5) + (4 * 6) + (2 * 20) + (2 * 18) + (3 * 7) + (4 * 4) + (4 * 4)] / 73 = 2.7397

The balanced binary tree of depth 3 to store the words given:



The expected representation length = 3

Problem C.

As D increases, the value of the training objective increases as the training accuracy improves when the dimension of the embedding space increases. However, one might not want to use very large D so as to prevent overfitting.

Problem D.

See Jupyter notebook for code.

Problem E.

i. Dimension of the weight matrix of hidden layer: (308, 10)
ii. Dimension of the weight matrix of output layer: (10, 308)
iii. Top 30 pairs of most similar words:

A pattern to notice is that it's common that each pair that is listed has the tuple in reverse listed as well with similar if not exact similarity. Running it for multiple epochs shows that each model trained will produce slightly different pairs although ones that show up a lot tend to rhyme and this is probably because this is Dr. Seuss.

Pair(socks, park), Similarity: 0.947406
Pair(park, socks), Similarity: 0.947406
Pair(good, pop), Similarity: 0.925787
Pair(pop, good), Similarity: 0.925787
Pair(all, upon), Similarity: 0.921253
Pair(upon, all), Similarity: 0.921253
Pair(blue, bird), Similarity: 0.919321
Pair(bird, blue), Similarity: 0.919321
Pair(zeds, need), Similarity: 0.90906
Pair(need, zeds), Similarity: 0.90906
Pair(with, told), Similarity: 0.907472

Pair(told, with), Similarity: 0.907472
Pair(they, ham), Similarity: 0.905727
Pair(ham, they), Similarity: 0.905727
Pair(say, pull), Similarity: 0.905089
Pair(pull, say), Similarity: 0.905089
Pair(one, make), Similarity: 0.902443
Pair(make, one), Similarity: 0.902443
Pair(when, thing), Similarity: 0.902274
Pair(thing, when), Similarity: 0.902274
Pair(likes, time), Similarity: 0.900297
Pair(time, likes), Similarity: 0.900297
Pair(fat, wump), Similarity: 0.897301
Pair(wump, fat), Similarity: 0.897301
Pair(he, dark), Similarity: 0.890201
Pair(dark, he), Similarity: 0.890201
Pair(was, fox), Similarity: 0.886819
Pair(fox, was), Similarity: 0.886819
Pair(things, finger), Similarity: 0.882929
Pair(finger, things), Similarity: 0.882929