# Problem 1

## Part A

For an matrix

We have singular values given by the diagonal matrix .

We have:

So we have the matrix diagonalization for

diagonal entries of are the eigenvalues of and we know that the diagonal entries are the squares of the singular values of . So we have found the PCA solution is where so we conclude that the columns of are the principle components of since the diagonal entries of are the singular values squared.

## Part B

Intuitively, the eigenvalues of are positive because they represent the variance upon a given dimension and variance is inherently a positive quantity

Mathematically, we showed in part a that the eigenvalues of are the squares of the singular values of which must be positive.

## Part C

We have matrices

It is clear that this generalizes to any number of matrices

Because for square matrices :

Let , then from above we know that

And the same can be done for any number of matrices and any permutation.

## Part D

We know it would take values to store the entire matrix

If we have the SVD of X

If we consider only the first singular values, We have which is and potentially dense, a truncated which is a diagonal matrix and thus only has non-zero entries, and finally a truncated which is size so altogether we need

Values.

We want to know when

## Part E

### Sub i

Let

Where each row is of dimension making size

Let

Which is of size

So we have:

Let

Where each row

And

So we have:

But since we can resubstitute and get

As desired.

### Sub ii

In class we said is orthogonal if

If is where , we have

And

Thus

So, we can’t fit our definition.

### Sub iii

If for some matrix

We know that the columns are orthonormal, we know that for any two columns , when

We also know that if then ,

Since we know how matrix multiplication works we know that some entry of the matrix

So putting it altogether we know that all non-diagonal entries of are 0 and the diagonals are 1

Which is the definition of the identity matrix which in this case is

We note for later that if

Because the sum of the entrants squared of the identity matrix is just

We can also show that it is false that for some that

Assume for the sake of contradiction that

Aside: let denote the **ROW** in

(I apologize I don’t know a more convenient notation)

So by that definition we have an entry of the resulting matrix

We therefore know that when and when since is assumed to be .

Thus we must have , but we know and , so we have a contradiction. Thus .

## Part F

i)

We know that is a diagonal square matrix

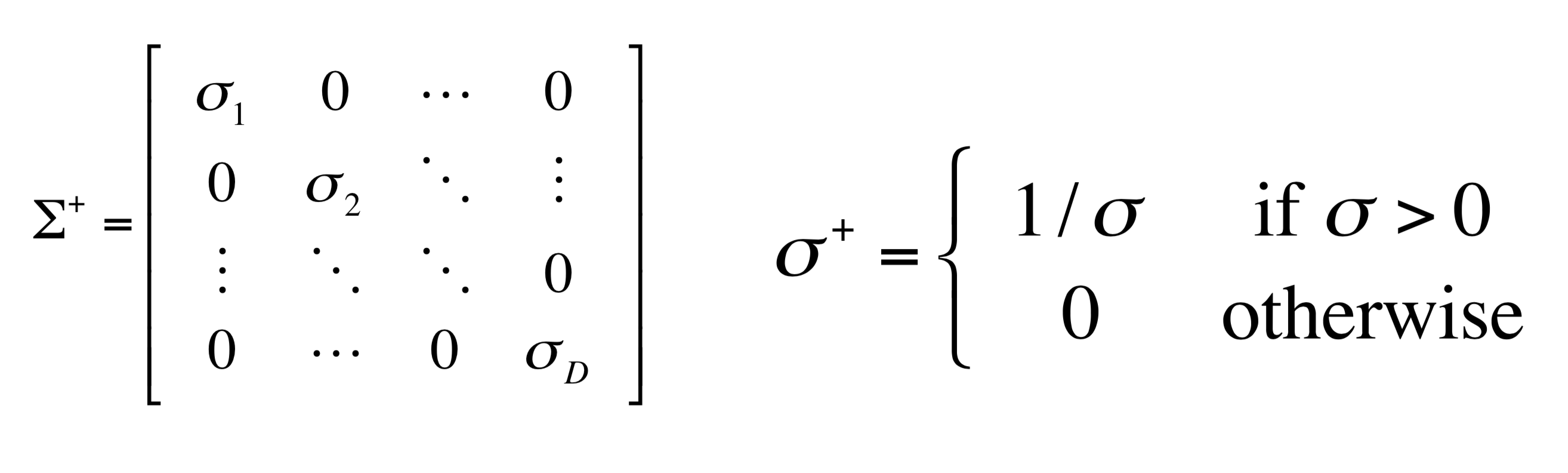
So say

We can see that

We know that all since we are told is invertible

We also know that all from part B.

is defined in the slides as follows:



Thus, as desired.

ii)

We have

We know

As desired.

iii)

The pseudoinverse of the form is the most complex and prone to numerical issues because it involves that inversion of a potentially large matrix which is computationally difficult and prone to numerical errors. The other method only needs the inversion of a diagonal matrix which is computationally easy.

# Question 2

We have the regularized square error:

Taking derivatives and applying chain rule gives:

The derivation for is symmetric so we have:

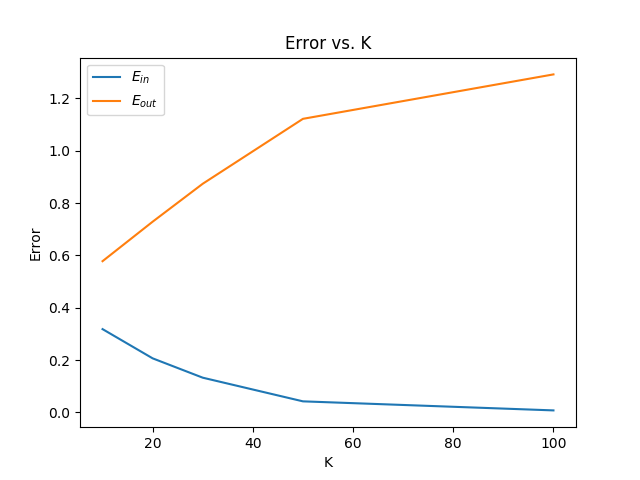
## Part B

We set gradient to 0 and look to solve for assuming the other is constant:

Similarly

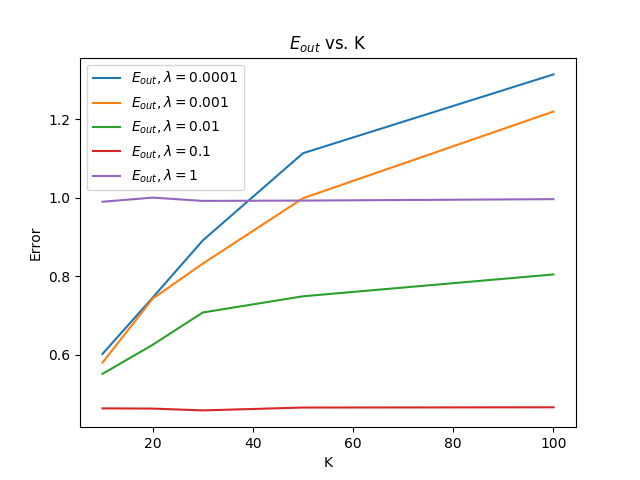
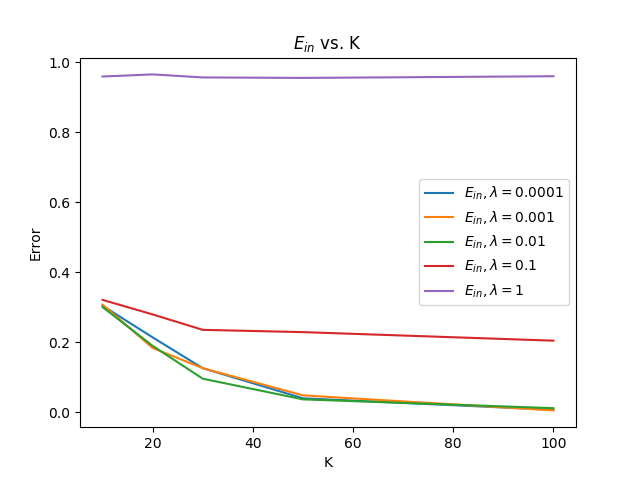
## Part D

We can see that training error decreases with which makes sense intuitively because with higher we are essentially compressing the data less which makes it easier to predict in sample. Testing error generally increases with which indicates that the model is overfitting with higher which also makes sense because we are increasing the complexity of the model class and looking to predict a lot of information given relatively sparse information.



## Part E

We can see that training error decreases with for each , however we have best training error decreasing with increasing which makes sense because a lower regularization strength means that the model is more free to fit the training data. Testing error increases with generally and there is also best performance for across all which indicates that it is the regularization strength that results in the best generalization behavior.



# Problem 3

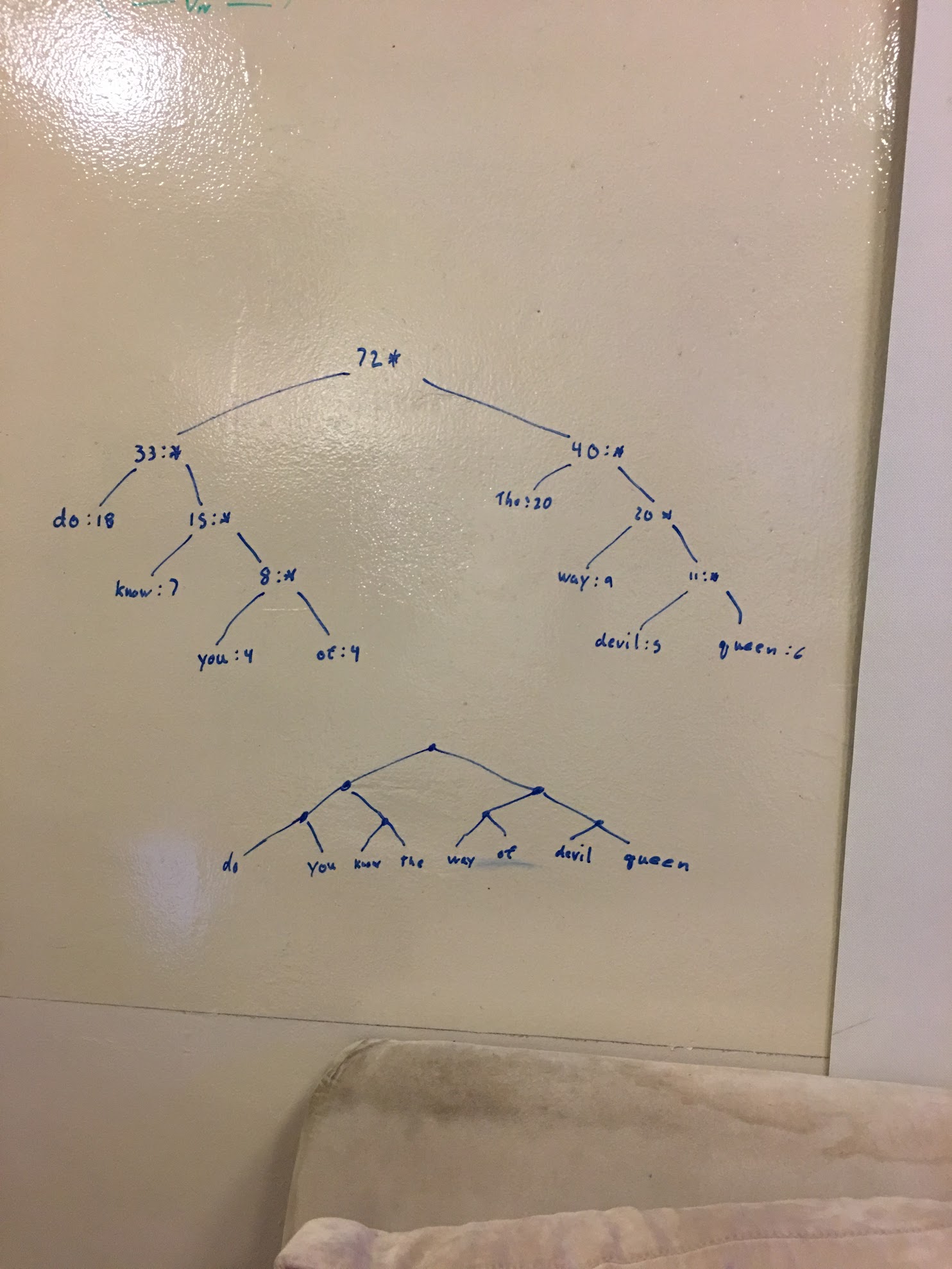
## Part A

We would like to calculate:

We must compute a gradient with respect to each term in the summation so the computation scales linearly with .

The numerator calculation scales with and the denominator scales with so calculating both scales with

## Part B



## Part C

As one increases D, the magnitude of the training objective will increase since the embedding vector will become bigger which for one thing will increase the size of the summation of the log probability. Also higher dimensionality can result in overfitting. Additionally, we pay an increased computational cost

## Part D/E

Weights shape layer 1: (308, 10)

Weights shape layer 2: (10, 308)

Pair(fox, goat), Similarity: 0.9869466

Pair(goat, fox), Similarity: 0.9869466

Pair(would, samiam), Similarity: 0.9864516

Pair(samiam, would), Similarity: 0.9864516

Pair(with, eat), Similarity: 0.98387635

Pair(eat, with), Similarity: 0.98387635

Pair(or, anywhere), Similarity: 0.98260736

Pair(anywhere, or), Similarity: 0.98260736

Pair(could, train), Similarity: 0.9825326

Pair(train, could), Similarity: 0.9825326

Pair(eggs, ham), Similarity: 0.98221785

Pair(ham, eggs), Similarity: 0.98221785

Pair(mouse, anywhere), Similarity: 0.9816431

Pair(green, ham), Similarity: 0.98051345

Pair(them, mouse), Similarity: 0.97886395

Pair(not, with), Similarity: 0.97852635

Pair(car, not), Similarity: 0.9766632

Pair(boat, with), Similarity: 0.9765761

Pair(do, eggs), Similarity: 0.9744789

Pair(tree, goat), Similarity: 0.9711042

Pair(box, with), Similarity: 0.96849716

Pair(be, not), Similarity: 0.967821

Pair(dark, tree), Similarity: 0.9654477

Pair(rain, tree), Similarity: 0.96290296

Pair(that, mouse), Similarity: 0.9579255

Pair(there, here), Similarity: 0.9575806

Pair(here, there), Similarity: 0.9575806

Pair(four, five), Similarity: 0.95323676

Pair(five, four), Similarity: 0.95323676

Pair(house, eat), Similarity: 0.9517888