(2a) By definition of Frobenius Norm from lec10 slide 8 we have $\|U\|_F^2 = \sum_{ij} u_{ij}^2$ & $\|V\|_F^2 = \sum_{ij} v_{ij}^2$

$$\therefore \partial u_i \left(\frac{\lambda}{2} \left(\|U\|_F^2 + \|V\|_F^2 \right) + \frac{1}{2} \sum_{ij} \left(y_{ij} - u_i^T v_j \right)^2 \right)$$

$$= \partial u_i \left(\frac{\lambda}{2} \left(\sum_{ij} u_{ij}^2 + \sum_{ij} v_{ij}^2 \right) + \frac{1}{2} \sum_{ij} \left(y_{ij} - u_i^T v_j \right)^2 \right)$$

$$= \frac{\lambda}{2} \left(\partial u_i + 0 \right) + \frac{1}{2} \sum_j \left(2 \left(y_{ij} - u_i^T v_j \right)^T \left(-v_j \right) \right)$$

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

and

$$\therefore \partial v_j \left(\frac{\lambda}{2} \left(\|U\|_F^2 + \|V\|_F^2 \right) + \frac{1}{2} \sum_{ij} \left(y_{ij} - u_i^T v_j \right)^2 \right)$$

$$= \partial v_j \left(\frac{\lambda}{2} \left(\sum_{ij} u_{ij}^2 + \sum_{ij} v_{ij}^2 \right) + \frac{1}{2} \sum_{ij} \left(y_{ij} - u_i^T v_j \right)^2 \right)$$

$$= \frac{\lambda}{2} \left(0 + \partial v_j \right) + \frac{1}{2} \sum_i 2 \left(y_{ij} - u_i^T v_j \right)^T \left(-u_i \right)$$

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

(2b) We'll find the closed form each partial derivative when each is equal to zero as that is our optimal V_j & U_i (the critical point)

$$\therefore \partial u_i \left(\frac{\lambda}{2} \left(\|U\|_F^2 + \|V\|_F^2 \right) + \frac{1}{2} \sum_{ij} \left(y_{ij} - u_i^T v_j \right)^2 \right) = 0$$

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

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

$$\rightarrow u_i \left(\lambda + \sum_j v_j v_j^T \right) = \sum_j y_{ij} v_j \quad \therefore \quad \boxed{u_i = \frac{\sum_j y_{ij} v_j}{\left(\lambda + \sum_j v_j v_j^T \right)}}$$

and for v_j:

$$= \partial v_j \left(\frac{\lambda}{2} \left(\|U\|_F^2 + \|V\|_F^2 \right) + \frac{1}{2} \sum_{ij} \left(y_{ij} - u_i^T v_j \right)^2 \right) = 0$$

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

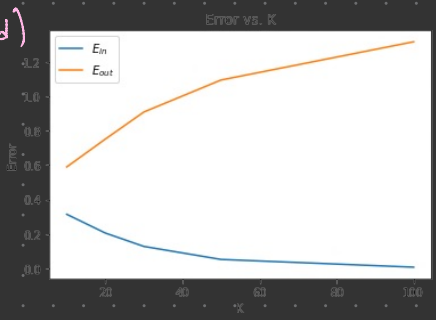
$$\rightarrow \lambda v_j = \sum_i y_{ij} u_i - \sum_i u_i v_j^T u_i \quad ; \quad \rightarrow \lambda v_j = \sum_i y_{ij} u_i - \sum_i u_i u_i^T v_j$$

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

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

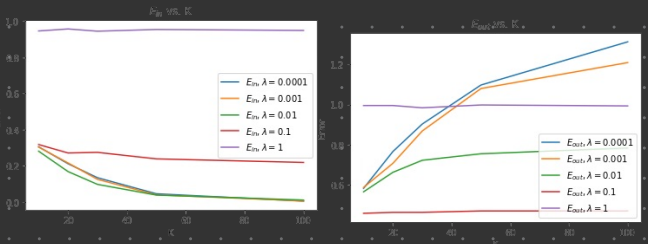
(2c) See Code For Prob2

(2d)



The Eout (out of sample error) is significantly higher than the Ein (in sample error) for all K values and as K values increase Ein decreases while Ein increases meaning we have overfitted with all models which makes sense as our regularization parameter is set to 0. This lack of regularization means our models simply memorize the training set hence fitting noise and making them bad generalizers and the higher the K value the greater this overfitting is as we are increasing model complexity hence allowing more noise to be fit in our training set.

(2e)



The trend is the same as in the previous part for most lambda values with the Eout being generally higher than Ein as a result of overfitting and this overfitting worsens for some of the lambda values (10**-4, 10**-3, 10**-2) as K increases, making them weak regularizers. For lambda equals 1, the regularization term has too much weight therefore all our models want to only minimize U and V norms, yielding relatively the same predictions for all inputs and poor performance for both training and testing error hence increasing K barely affects the models therefore consistently performing poorly for all K values as the both the Ein and Eout are consistently extremely high. Lambda equals 0.1 is our best regularizer because it yields the lowest Eout and as our values for K increase, Ein continually to slightly decreas and Eout seems to slightly decrease before stagnating to stay consistently low hence avoiding increased overfitting as seen for all other values.

(3a)

$$\log P(w_o|w_I) = \log\left(\frac{\exp(v'^{T}_{w_o} v_{w_I})}{\sum_{w=1}^{W} \exp(v'^{T}_{w} v_{w_I})}\right) = \log\left(\frac{e^{v'^{T}_{w_o} v_{w_I}}}{\sum_{w=1}^{W} e^{v'^{T}_{w} v_{w_I}}}\right)$$

$$= \log\left(e^{v'^{T}_{w_o} v_{w_I}}\right) - \log\left(\sum_{w=1}^{W} e^{v'^{T}_{w} v_{w_I}}\right)$$

$$= v'^{T}_{w_o} v_{w_I} - \log\left(\sum_{w=1}^{W} e^{v'^{T}_{w} v_{w_I}}\right)$$

Now to find the gradient:

$$\nabla_{v'_{w_o}} \log P(w_o|w_I) = v_{w_I} - \frac{v_{w_I} e^{v'^{T}_{w_o} v_{w_I}}}{\sum_{w=1}^{W} e^{v'^{T}_{w} v_{w_I}}}$$

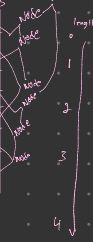
$$\nabla_{v_{w_I}} \log P(w_o|w_I) = v'_{w_o} - \frac{\sum_{w=1}^{W} v'_{w} e^{v'^{T}_{w} v_{w_I}}}{\sum_{w=1}^{W} e^{v'^{T}_{w} v_{w_I}}}$$

The time complexity of $v'^{T}_{w} v_{w_I}$ is $O(D)$ as we are taking the dot product for all D dimensions & the summation $\sum e^{v'^{T}_{w} v_{w_I}}$ has a time complexity of $O(WD)$ as we are calculating $v'^{T}_{w} v_{w_I}$ W many times :. for a sing w_o, w_I pair the time complexity is $O(WD)$.

(3b)

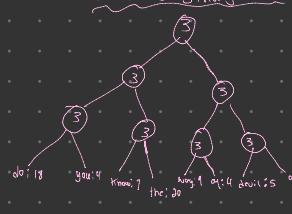
Word	occurances
do	18
you	4
know	7
the	20
way	4
of	4
devil	5
queen	6

=73



Huffman Tree

Balanced Binary Tree



The expected length of the Binary tree is 3, the depth of our tree.

The expected length of the Huffman tree is $\frac{\sum(\text{length})(\text{occurance})}{73} = \left(2(do) + 4(you) + 5(know) + 2(the) + 3(way) + 4(of) + 4(devil) + 4(queen)\right)/73$

$= \left(2(18) + 4(4) + 3(7) + 2(20) + 3(4) + 4(4) + 4(5) + 4(6)\right)/73$

$= 200/73 \approx 2.7397$ hence why we used Huffman Tree instead of binary tree.

(3c) As D increases, we expect the value of the training objectives to increase as an increased D allows for better model selections hence allowing us to fit the training data better. However, we might not want to use a very large D in order to avoid overly complex models that fit noise resulting in overfitting. This increased in dimensionality could also be very computationally expensive hence we don't want a very large D.

(3d) See Code For Prob3

(3e) The dimensions are 308×10 showing the number of words and embeding dimensions.

(3f) The dimmensions are 10×308.

(3g)

```
Pair(fear, take), Similarity: 0.9954883
Pair(take, fear), Similarity: 0.9954883
Pair(feet, low), Similarity: 0.99465114
Pair(low, feet), Similarity: 0.99465114
Pair(called, zeep), Similarity: 0.98782295
Pair(zeep, called), Similarity: 0.98782295
Pair(head, wave), Similarity: 0.98776114
Pair(wave, head), Similarity: 0.98776114
Pair(made, only), Similarity: 0.9875726
Pair(only, made), Similarity: 0.9875726
Pair(little, took), Similarity: 0.9871121
Pair(took, little), Similarity: 0.9871121
Pair(teeth, boat), Similarity: 0.9870644
Pair(boat, teeth), Similarity: 0.9870644
Pair(ned, pink), Similarity: 0.9868588
Pair(pink, ned), Similarity: 0.9868588
Pair(but, grow), Similarity: 0.98576736
Pair(grow, but), Similarity: 0.98576736
Pair(story, low), Similarity: 0.9852524
Pair(it, foot), Similarity: 0.9851571
Pair(foot, it), Similarity: 0.9851571
Pair(dad, thin), Similarity: 0.98467755
Pair(thin, dad), Similarity: 0.98467755
Pair(fat, ride), Similarity: 0.98459214
Pair(ride, fat), Similarity: 0.98459214
Pair(too, take), Similarity: 0.98443496
Pair(cant, way), Similarity: 0.9842897
Pair(way, cant), Similarity: 0.9842897
Pair(nine, boat), Similarity: 0.9842129
Pair(cans, yink), Similarity: 0.984207
```

(3h) We notice that most pairs occur twice s.t. for words x,y then we will have the pair (x,y) and (y,x) show up twice next to each other with the same similarity percentage as our model associates words with their neighbors in both directions.