

UIUC-CS512 “Data Mining: Principles and Algorithms” (Spring 2021)
Second Midterm Exam

(Friday, April 30, 2021, 100 marks)

IMPORTANT Notes

- Please provide brief explanations of your answers.
- Please sign the honor code in page 2. Your exam will not be graded unless the above agreement is signed. Please attach the signed honor code to your answer sheet.
- You can either (1) type in your answers in Latex/Word and submit your answer sheet in pdf, or (2) provide hand-written answers and submit a scanned version (pdf). For (2), Please make sure that the scanned version is clear and recognizable. Otherwise, you might loose points.

:

Name: **Daniel Campos**

NetID: **dcampos3**

Score:

1	2	3	4	5	6	Total

CS512 Spring 2021 Exam Honor Code


I understand that the rules of the CS512 second mid-term exam in Spring 2021. That is, (1) the exams are “open book”, and (2) I am not allowed to confer with other people about the questions or solutions to the exam (to either give or receive aid).

I have neither given nor received inappropriate aid during this exam.

I understand that my exam will not be graded unless the above agreement is signed.

NetID (print): dcampos3

Name (print): Daniel Campos

Name (signed): 

Date: 04/30/2021

CS 512 Midterm 2

Daniel Campos

April 30th, 2021

1 Short Questions

1.1 True Or False

1.1.1 For the Susceptible-Infected-Susceptible (SIS) model on network, whether there will be an epidemic is related with the selection of initial infected nodes.

False, initial selection of infected nodes can effect if there is an epidemic but an epidemic for SIS is determined by the infection rate β and the recovery rate δ if the β is high and the δ is low few initializations can effect epidemic or not and alternatively if β is low and δ is high few initialization can cause an epidemic.

1.1.2 Node attribute can provide effective assistance for network alignment task

True. Methods such as Final: Attributed Network Alignment leverages the intuition that similar node-pairs share similar node attributes to predict alignment.

1.1.3 Initializing all the weights with 0 during training will prevent the neural network from learning.

False. If all weights are initialized to zero then the derivative for each weight in the network will be identical. Since each weight is identical each weight will receive the same update and the values in the network will always be equal $w_i = w_j \forall j \in N$. As a result, the network will produce poor results for any constant initialization. The network is technically learning as weight values are updating but it is very unlikely to learn a good representation of the data.

1.1.4 Stochastic gradient descent (SGD) can help getting out of local minimums of the loss function

True. SGD is more able to avoid getting stuck at local minima for the entire dataset because it is randomly sampling a batch from the dataset. Since it is not optimizing on the entire dataset at each step it is possible that the network has a loss on the entire dataset it cannot escape but eventually a random batch of the data could be different causing the network to leave the local minimum.

1.1.5 If we randomly shuffle the order of nodes in a graph, we can have different results (e.g., node classification) from the GCN model.

True. In a graph random shuffling of a graph does not change the graph but since GCN's rely on convolution shuffling node order will cause different outcomes. It worth noting that there are solutions to such as Simple Permutation-Invariant Graph Convolutional Network and more regularly applied node ordering.

1.2 Short Answer Questions

1.2.1 What problems might occur if the learning rate is fixed during training?

The learning rate may be either too big or too small. Too big of a learning rate will cause the model to overshoot the global minimum during training. Too small will cause the model to move very slowly and can cause it to get stuck at local minima. While there could be a theoretically optimal fixed learning rate for each task without knowing that having some variability in learning rate will allow optimal exploration.

1.2.2 List at least three challenges for network alignment.

Complexity, variety, and Heterogeneity. Network alignment can be too complex in terms of space and time complexity which causes issues as network size grows. Network alignment can lack variety as networks can be rich in terms of nodes and edge attributes which can be forgotten in alignment. Network alignment can not be heterogeneous enough as networks may have different behaviors for each type of node but the network is aligning to the average.

1.2.3 From the formula of Netshield, intuitively explain that why Netshield does not only select nodes with high eigen-scores to be vaccinated.

Given the formula $Sv(s) = \sum_{i \in S} 2\lambda u(i)^2 - \sum_{i,j \in S} A(i,j)u(i)u(j)$ we can pay attention to the second summation as it measures diversity between nodes. We do want to select the nodes with the highest eigen-scores but want to make sure they are diverse amongst themselves to maximize the impact on the overall network.

1.2.4 Based on the idea of IONE (Lecture 8, before and including page 35), explain with your own words that whether we can learn node embeddings from two networks independently and align node pairs based on a distance measure (say cosine similarity)

We can learn the alignment between two networks independently as long as we assume that their distribution and relation is similar. Given the KL-Divergence we can learn some form of alignment between the two networks such that we minimize the divergence between the two networks. At its core the idea is to transform each network into a vector space and minimize the distance between the two. This makes perfect sense as we can see the real effect in how languages work. Each language is effectively a network of concepts which are connected by shared attributes (Cats and dogs are animals and pets but only cats are felines). If we transform and align the embeddings of the words to recognize that cats in English should be close to gato in Spanish then we can find an effective way of align the networks using embedding similarity vs alignment.

Unit,j	Net Input, I_j	Output, O_j
4	$0.6+0-0.3= 0.3$	$\max\{0, (0.3 - 0.2)\} = 0.1$
5	$0.3+0+0.2 = 0.5$	$\max\{0, (0.5 + 0.3)\} = 0.8$
6	$0.03 + 0.56 = 0.59$	$\max\{0, (0.59 - 0.1)\} = 0.49$

Table 1: Inputs and outputs of neurons 4,5,6

1.2.5 In order to capture long-term dependency, what modifications does LSTM introduce to RNN? Briefly explain how it works to improve learning.

LSTM's introduce learn able gating parameters to control the information that passes in the hidden state. RNNs suffer from a vanishing gradient where information from previous states disappears the as more information comes into the hidden state. LSTM's have a forget, input, and output gate which control what information is updated for the hidden state. This means the network learns what information to keep and what information to discard and as a result is able to learn longer distance dependencies.

2 Deep Learning

2.1 Show how tanh can be transformed from sigmoid through shifting and re-scaling.

As the sigmoid function is symmetric at origin and return $[0,1]$ we can write $1 - \sigma(I) = \sigma(-I)$ which is then simplified into $1 - \frac{1}{1+e^{-I}} = \frac{1}{1+e^I}$.

Similarly we can rearrange tanh function becomes $0 = \frac{e^I - e^{-I} - 2e^{-I}}{e^I + e^{-I}}$ which becomes $= 1 + \frac{-2e^{-I}}{e^I + e^{-I}} = 1 - \frac{2}{e^{2I} + 1}$.

If we simplify from the sigmoid perspective $\tanh(I) = 1 - \frac{2}{e^{2I} + 1} = 1 - 2\sigma(-2I)$ which in turn $\tanh(I) = 1 - 2(1 - \sigma(2I)) = 1 - 2 + 2\sigma(2I) = 2\sigma(2I) - 1$ which means tanh is shifted (the constant -1) and rescaled ($2\sigma(2I)$) sigmoid.

2.2 How many possible distinct and connected networks are there?

Mathematically there could be as many as 2^N networks where N is the number of non output neurons. Since we have 3 layers that would mean that possible network combinations are $2^{(d_0)*(d_0)(d_1)(d_1)(d_2)(d_2)}$ but since the output layer must be fully connected we drop the final d_2 . Additionally, since in each layer there must be at least 1 neuron connecting layers we update combinations to $2^{((d_0)-1)*((d_0)(d_1)-1)((d_1)(d_2)-1)}$ which becomes $2^{d_0^2 d_1^2 d_2 - d_0^2 d_1 - d_0 d_1 d_2 - d_0 - d_0 d_1^2 d_2 + d_0 d_1 - d_1 d_2 - 1}$ unique subnetworks with dropout.

2.3 Compute the input and the output of units 4, 5 and 6

Results can be found in table 1

2.4 Compute the error of units 4, 5 and 6

Results can be found in table 2

Unit,j	Error δ_j
6	$0.49(0-0.49)(0-0.49) = 0.49(0.2401) = 0.117649$
5	$0.8(0-0.8)(0.117649)(0.7) = -0.052706752$
4	$0.1(0-0.1)(0.117649)(0.3) = -0.000352947$

Table 2: Errors of 4,5,6

weight	New Value
$w_{4,6}$	$0.3 - (0.1)(0.117649)(0.1) = 0.29882351$
$w_{1,5}$	$0.3 - (0.1)(-0.052706752)(1) = 0.3052706752$

Table 3: New Weights for 4,5,6

2.5 compute the new weights, $w_{1,5}$ and $w_{4,6}$, after one-step update.

Results can be found in table 3

2.6 Suppose we do not apply zero-paddings, what are the dimensions of the feature maps after hidden layer 4? Please show the key steps of calculation

After the first layer our feature map becomes $28*28*16$ due to the the 16 filters and losing 2 on each side of the $5*5$ convolution. After max pool the size is $14*14*16$ as max pool reduces $2*2$ to $1*1$. After the second convolutional layer the feature map is $12*12*32$ because our $3*3$ convolutions loses 1 on each dimension. Layer 4 then applies max pool which gives a size of $6*6*32$ which is the answer to the question. Layer 5 produces a $1*1*152$

2.7 How many weights does the architecture have? What advantage of CNN do you observe compared with fully-connected layer during the computation?

The parameters we have are $p(CCN1) + p(CNN2) + p(FFN) + output$ since max pool and flatten have no parameters. This means $params = 16 * 5 * 5 * 3 + 32 * 3 * 3 * 16 + 1152 * 256 + 256 * 1 = 1200 + 4608 + 294912 + 256 = 300,976$ parameters.

The advantages of a CNN vs FFN is faster training/inference because the relative sparsity of the network. Instead of having fully connected layers at each level there are sparsely connected layers which means the number of parameters and computations is orders of magnitude less.

2.8 Compute the output features after applying one spectral-based GCN layer

$Z = D^{-0.5}AD^{-0.5}X\theta$ where $D = [[2, 0, 0, 0, 0], [0, 3, 0, 0, 0], [0, 0, 3, 0, 0], [0, 0, 0, 3, 0], [0, 0, 0, 0, 3]]$ and $A = [[0, 1, 0, 0, 1], [1, 0, 1, 1, 0], [0, 1, 0, 1, 1], [0, 1, 1, 0, 1], [1, 0, 1, 1, 0]]$ so $Z = [[-0. , 0.6], [-0.3, 0.], [-0.3, 0.6], [-0.3, 0.6], [-0.3, 0.]]$

2.9 Show how the categorical cross-entropy loss is calculated given T and O

Starting with the Binary cross entropy $L(T, O) = -T \log O - (1 - T) \log(1 - O)$ we can see that this is actually 2 class cross entropy losses but since we know that T and O have to be 0 or one we can write $T' = 1 - T$. Knowing that lets rewrite this as a 2 class cross entropy loss. First, our

T goes becomes a one hot vector (T_v) as we make $T_v = [T, 1 - T]$ and we do the same for our Output. As a result our loss function $L(T_v, O_v) = \sum_{i=0}^1 -T_i \log(O_i)$. Now, since we have to ensure that $\sum_{i=0}^1 T_i \log(O_i) = 1$ irregardless of how many classes we add we leverage softmax function and rewrite our CE loss as $CE(T, O) = - \sum t_i \log \frac{e^{O_i}}{\sum_j e^{O_j}}$

3 Outlier Detection

3.1 which value is a potential outlier

First we calculate $\mu = 16.33636363636364$ as μ is the mean. Next we calculate $\sigma = \sqrt{\frac{\sum_i (x_i - \mu)^2}{N}} = 1.4846320188534794$

Next we calculate the z scores

$$= [0.24493, 0.1592, 0.3122, 0.37964, 3.0555, 0.37964, 0.1775, 0.3122, 0.851, 0.0244, 0.581].$$

Looking at Z scores we can see that 11.8 is an outlier with a z score of over 3.

3.2 Compute the local reach ability density of B and F

First off we calculate the $dist_3 = [2, 3, 2, 3, 3, 5, 5]$ and the neighborhoods are $[[b, c, d], [a, c, e], [a, b, d], [a, c, e, g], [b, c, d], [c, d, e], [a, c, d]]$.

Using this we calculate the Local reachability density for each item with results:

$$= [0.13636, 0.11538, 0.16666, 0.2, 0.12, 0.08823, 0.08571 \text{ which means :}$$

$$lrd_3(b) = 0.11538 \text{ and } lrd_3(f) = 0.08823$$

3.3 compute the local outlier factor of A and F

$$LOF_3(A) = 1.1783475783475785 \text{ and } LOF_3(F) = 1.8385185185185182$$

3.4 what are the possible outliers from the perspective of local outlier factor and why?

The LOF for all points are

$$= 1.1783, 1.222, 0.903, 0.6359, 1.339, 1.838, 1.956$$

which makes G and F likely outliers as their coefficient is highest.

4 Graph Connectivity Optimization

4.1 Report ζ_0 and p_1

Since probability all starts the same ζ_0 is $0.5 * (1 - .2) + (1 - 0.5) = (0.5)0.8 + 0.5 = 0.9$ raised to their degree which is $[2, 3, 2, 3, 4]$ giving $\zeta_0 = [0.81, 0.729, 0.81, 0.729, 0.6561]$.

Then the infection probability $1 - p_{t+1}[i] = \sigma p_t[i] + (1 - p_t[i])\zeta_t[i]$ can be rewritten as $p_{t+1}[i] = 1 - (\sigma p_t[i] + (1 - p_t[i])\zeta_t[i])$ which gives $p_1 = [0.545, 0.5855, 0.545, 0.5855, 0.62195]$.

4.2 If we have a vaccination to immune one node which is best. Intuitively explain the answer of (1) with your own words

If we don't vaccinate any node the infection probability is 0.5766. If we vaccinate 0 the infection probability is 0.5652, for 1 it is 0.5439, for 2 it is 0.5652, for 3 it is 0.5439, and for 4 it is 0.5225. As a result, our best node to vaccinate is node 4. Intuitively this makes sense as 4 is the node that has the highest degree in our graph. By removing the node that is most connected in the network we essentially remove one 0.5 from our non infection probability which ensures that it stays $\frac{1}{0.9}$ bigger for all. This follows much of what we studied in class where the highest eigen value node is the optimal vaccination as long as the high degree nodes are separated. In the case of this problem since we have few nodes and are only vaccinating one it is not a concern.

5 Network Alignment

5.1 What is the time complexity (in big O notation) for the computation

First off, the kronecker product of A_1 and A_2 takes $O(n^2m^2)$ to run as each element must be multiplied. Next, as the power method has linear convergence we must run this algorithm K times. Since this runs just like random walk algorithm but with two sets of nodes we have N which represents the nodes in each of our matrixes of $n \times m$. Thus the complexity is $O(K * 2N) = O(KN)$ where N is number of nodes in each adjacency matrix.

5.2 Explain with your own words that why this component can capture the topology consistency

Topology consistency means the same node has a consistent connectivity structure across different networks. This is captured by the equation for Final because it is trying to minimize the value that J has with S . Since the main term in the equation is $\frac{S(x,a)}{\sqrt{f(x,a)}} - \frac{S(y,b)}{\sqrt{f(y,b)}}$ The goal of this equation is to minimize the difference between $S(x,a)$ and $S(y,b)$ an alignment that minimizes this has connected nodes that share network structure and attributes. Since this equation not only places value on the attributes but what they are connected to, it thus captures and maximizes an alignment that preserves topology consistency.

5.2.1 what is the approximated closed-form solution of s? What is the time complexity of the approximated solution ?

First, we include the approximations in the closed form formula $s = (1 - \alpha(I - \alpha D_n^{-\frac{1}{2}} N (U_1 \Lambda_1 U_1^T \otimes U_2 \Lambda_2 U_2^T) N D_N^{-\frac{1}{2}})^{-1} h$ by using the Sherman Morrison Lemma. We then move things around to simply to $s = (1 - \alpha(I - \alpha D_n^{-\frac{1}{2}} N (U_1 \otimes U_2 * \Lambda_1 \otimes \Lambda_2 * U_1^T \otimes U_2^T) N D_N^{-\frac{1}{2}})^{-1} h$ which is further improved to $s = (1 - \alpha(I + \alpha D_n^{-\frac{1}{2}} N (U_1 \otimes U_2 * \Lambda_1 \otimes \Lambda_2 * U_1^T \otimes U_2^T)^{-1} N D_N^{-\frac{1}{2}}) h$. This is finalized with $s = (1 - \alpha(I - \alpha D_n^{-\frac{1}{2}} N * U_1 \otimes U_2 ((\Lambda_1 \otimes \Lambda_2)^{-1} * \alpha U_1^T \otimes U_2^T N D_N^{-1} N U_1 \otimes U_2)^{-1} * U_1^T \otimes U_2^T * N D_N^{-\frac{1}{2}}) h$. The complexity of the approximate solution from one is $O(n^2r^4)$ but since we do not have to calculate the approximations our solution is $O(N^2)$ as only random walk needs to be executed.