

# HOMWORK 9: LEARNING PARADIGMS

10-301/10-601 Introduction to Machine Learning (Spring 2021)

<http://www.cs.cmu.edu/~mgormley/courses/10601/>

OUT: 2022-04-21

DUE: 2022-04-27

TAs: Abbey, Abhi, Alex, Neural, Shelly, Udai

This is the final homework assignment. This assignment covers **Graphical Models, Ensemble Methods, K-Means, PCA, and Recommender Systems**.

## START HERE: Instructions

- **Collaboration Policy:** Please read the collaboration policy here: <http://www.cs.cmu.edu/~mgormley/courses/10601/syllabus.html>
- **Late Submission Policy:** See the late submission policy here: <http://www.cs.cmu.edu/~mgormley/courses/10601/syllabus.html>
- **Submitting your work:** You will use Gradescope to submit answers to all questions and code. Please follow instructions at the end of this PDF to correctly submit all your code to Gradescope.
  - **Written:** For written problems such as short answer, multiple choice, derivations, proofs, or plots, please use the provided template. Submissions can be handwritten onto the template, but should be labeled and clearly legible. If your writing is not legible, you will not be awarded marks. If your scanned submission misaligns the template, there will be a 5% penalty. Alternatively, submissions can be written in LaTeX. Each derivation/proof should be completed in the boxes provided. If you do not follow the template, your assignment may not be graded correctly by our AI assisted grader.

## Instructions for Specific Problem Types

For “Select One” questions, please fill in the appropriate bubble completely:

**Select One:** Who taught this course?

- Matt Gormley
- Marie Curie
- Noam Chomsky

If you need to change your answer, you may cross out the previous answer and bubble in the new answer:

**Select One:** Who taught this course?

- Matt Gormley
- Marie Curie
- Noam Chomsky

For “Select all that apply” questions, please fill in all appropriate squares completely:

**Select all that apply:** Which are scientists?

- Stephen Hawking
- Albert Einstein
- Isaac Newton
- I don't know

Again, if you need to change your answer, you may cross out the previous answer(s) and bubble in the new answer(s):

**Select all that apply:** Which are scientists?

- Stephen Hawking
- Albert Einstein
- Isaac Newton
- I don't know

For questions where you must fill in a blank, please make sure your final answer is fully included in the given space. You may cross out answers or parts of answers, but the final answer must still be within the given space.

**Fill in the blank:** What is the course number?

10-601

10-~~6~~301

## Written Questions (47 points)

### 1 $\LaTeX$ Bonus Point (1 points)

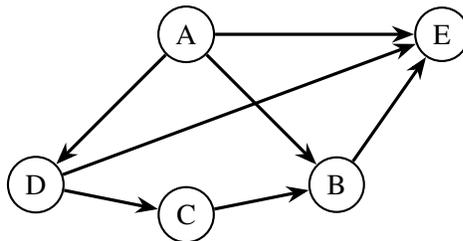
1. (1 point) **Select one:** Did you use  $\LaTeX$  for the entire written portion of this homework?

Yes

No

### 2 Graphical Models (6 points)

Consider the joint distribution over the binary random variables  $A, B, C, D, E$  represented by the Bayesian Network shown in the figure.



1. (1 point) Write the joint probability distribution for  $P(A, B, C, D, E)$  factorized as much as possible according to the standard definition of a Bayesian Network using the conditional independence assumptions expressed by the above network.

Your Answer

2. (1 point) Which nodes are in the Markov boundary of  $B$ ? Note that the Markov boundary is the smallest possible Markov blanket.

Your Answer

3. (1 point) Which nodes are in the Markov boundary of  $C$ ? Note that the Markov boundary is the smallest possible Markov blanket.

Your Answer

4. (1 point) **True or False:**  $E$  is conditionally independent of  $C$  given  $\{A, B, D\}$ . That is,  $E \perp C \mid \{A, B, D\}$ .

**Select one:**

- True  
 False

5. (1 point) How many parameters would we need to represent the joint distribution  $P(A, B, C, D, E)$  **without** the conditional independence assumptions expressed by the Bayesian Network?

Your Answer

6. (1 point) How many parameters would we need to represent the joint distribution  $P(A, B, C, D, E)$  **with** the conditional independence assumptions expressed by the Bayesian Network?

Your Answer

### 3 PCA (8 points)

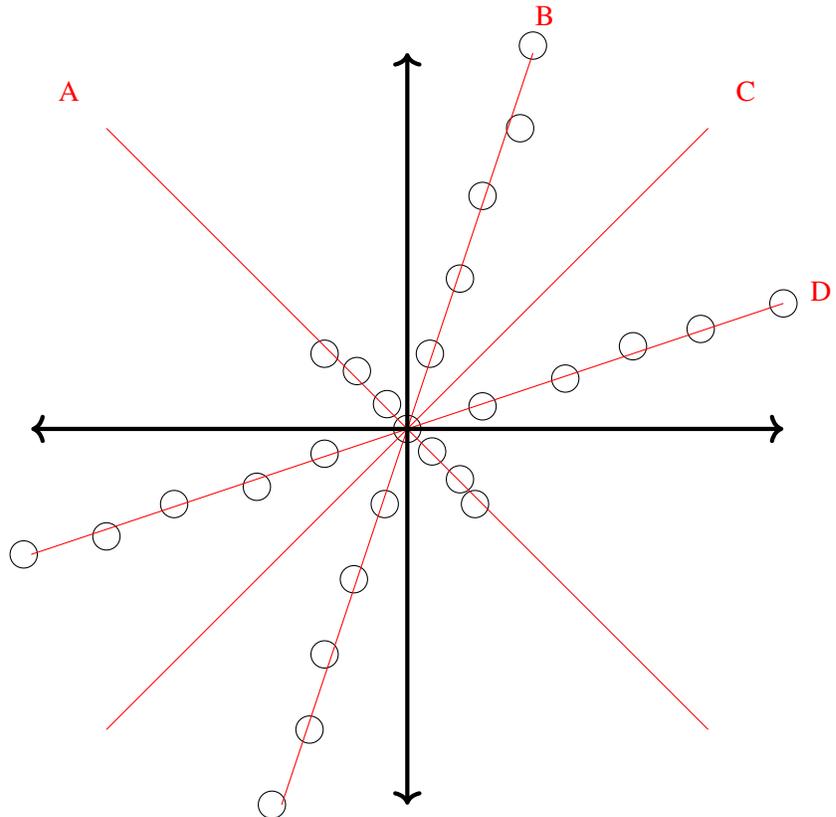
#### Some PCA Theory

1. (1 point) Assume we apply PCA to a matrix  $X \in R^{n \times 2}$  and obtain two sets of PCA feature scores,  $Z_1, Z_2 \in R^n$ , where  $Z_1$  corresponds to the first principal component and  $Z_2$  corresponds to the second principal component. Which is more common in the training data:

Select one:

- a point with small feature values in  $Z_2$  and large feature values in  $Z_1$
  - a point with large feature values in  $Z_2$  and small feature values in  $Z_1$
  - a point with large feature values in  $Z_2$  and large feature values in  $Z_1$
  - a point with small feature values in  $Z_2$  and small feature values in  $Z_1$
2. (2 points) For the data set shown below, list the principal components from first to last.

Your Answer



## PCA in Practice

For this section, refer to the PCA demo linked [here](#). In this demonstration, we have performed PCA for you on a [simple four-feature dataset](#). The questions below have also been added to the colab notebook linked for ease of access. Run the code in the notebook, then answer the questions based on the results.

3. (1 point) Do you see any special relationships between any of the features? In particular, take a look at the `petal_length` feature. How would you describe its association with each of the **other features**? Select the correct statement with appropriate justification.

**Select one:**

- The features are highly correlated: we observe linearly proportional relationships where increases in `petal_length` often correspond to increases in another feature
  - The features are highly correlated: we observe that the color classes can be separated with decision boundaries along the `petal_length` axis.
  - The features are uncorrelated: we observe random noise as if the features were generated from independent distributions
  - The features are uncorrelated: we observe the “default  $y = x$ ” relationship between features
4. (2 points) To get the principal components of the features, we calculate the eigenvectors of the covariance matrix, which are orthogonal, along with their corresponding eigenvalues. Which of the following are consequences of the principal components being orthogonal to each other?

**Select all that apply:**

- The variance of the data is maximized.
  - The reconstruction error is minimized.
  - The dot product of any two principal components will be 1.0.
  - We can attribute certain variations in the data to unique principal components.
  - In the dimensionality-reduced space, the covariance of the first and second dimensions will always be zero.
  - It ensures that our lower-dimensional data will be linearly separable.
  - None of the above.
5. (1 point) If we wanted to find  $k$  principal components such that we preserve **at least** 95% of the variance in the data, what would be the value of  $k$ ? Hint: it is helpful here to look at the cumulative variance in the first  $k$  components, which we have calculated for you.

$k$

6. (1 point) If we wanted to perform dimensionality reduction to have just two features, we could pick any two features from the dataset and train a classifier on just those. What is one reason we could prefer the PCA features to just choosing two of the original features to represent our data?

Your Answer

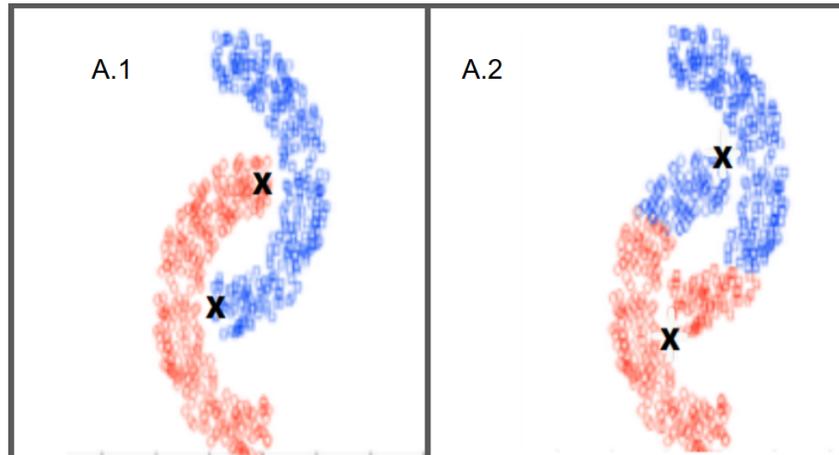
## 4 K-Means (13 points)

1. Consider the 3 datasets A, B and C. Each dataset is classified into  $k$  clusters, with centers marked  $X$  and cluster membership represented by different colors in the figure. For each dataset, exactly one clustering was generated by K-means with Euclidean distance. Select the image with clusters generated by K-means.

(a) (1 point) Dataset A

Select one:

- A.1  
 A.2



(b) (1 point) Dataset B

Select one:

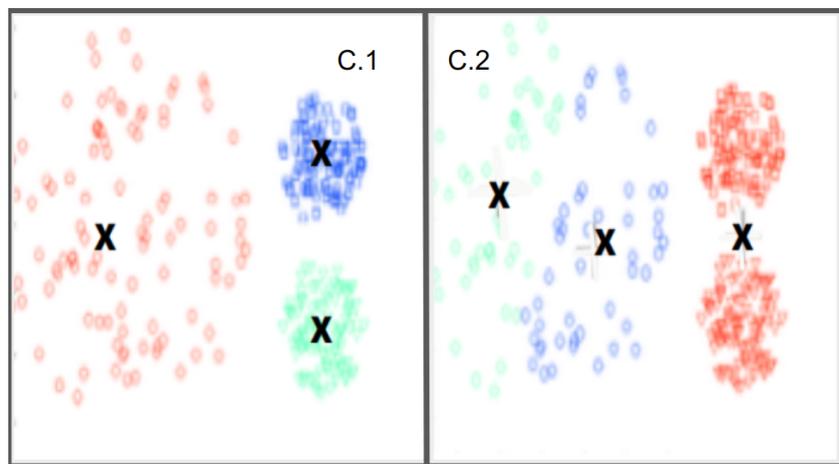
- B.1  
 B.2



(c) (1 point) Dataset C

Select one:

- C.1  
 C.2



2. Consider a dataset  $\mathcal{D}$  with 5 points as shown below. Perform a K-means clustering on this dataset with  $k = 2$  using the Euclidean distance as the distance function. Remember that in the K-means algorithm, one iteration consists of following two steps: first, we assign each data point to its nearest cluster center; second, we recompute each center as the average of the data points assigned to it. Initially, the 2 cluster centers are chosen randomly as  $\mu_0 = (5.3, 3.5)$ ,  $\mu_1 = (5.1, 4.2)$ . Parts (a) through (d) refer only to the first iteration of K-means clustering performed on  $\mathcal{D}$ .

$$\mathcal{D} = \begin{bmatrix} 5.5 & 3.1 \\ 5.1 & 4.8 \\ 6.6 & 3.0 \\ 5.5 & 4.6 \\ 6.8 & 3.8 \end{bmatrix}$$

- (a) (1 point) Which of the following points will be the new center for cluster 0?

**Select one:**

- (5.7 , 4.1)
- (5.6 , 4.8)
- (6.3 , 3.3)
- (6.7 , 3.4)

- (b) (1 point) Which of the following points will be the new center for cluster 1?

**Select one:**

- (6.1 , 3.8)
- (5.5 , 4.6)
- (5.4 , 4.7)
- (5.3 , 4.7)

- (c) (1 point) How many points will belong to cluster 0, using the new centers?

Answer

- (d) (1 point) How many points will belong to cluster 1, using the new centers?

Answer

3. Recall that in K-means clustering we attempt to find  $k$  cluster centers  $\mathbf{c}_1, \dots, \mathbf{c}_k$  such that the total distance between each point and the nearest cluster center is minimized. We thus solve

$$\operatorname{argmin}_{\mathbf{c}_1, \dots, \mathbf{c}_k} \sum_{i=1}^N \min_{j \in \{1, \dots, k\}} \|\mathbf{x}^{(i)} - \mathbf{c}_j\|_2^2$$

where  $n$  is the number of data points. Instead of holding the number of clusters  $k$  fixed, your friend John tries to also minimize the objective over  $k$ , solving

$$\operatorname{argmin}_k \operatorname{argmin}_{\mathbf{c}_1, \dots, \mathbf{c}_k} \sum_{i=1}^N \min_{j \in \{1, \dots, k\}} \|\mathbf{x}^{(i)} - \mathbf{c}_j\|_2^2$$

You found this idea to be a bad one.

- (a) (1 point) What is the minimum possible value of the objective function when minimizing over  $k$ ?

Answer

- (b) (1 point) What is a value of  $k$  for which we achieve the minimum possible value of the objective function when  $N = 100$ ?

Answer

4. Consider the following brute-force algorithm for minimizing the K-means objective: Iterate through each possible assignment of the points to  $k$  clusters,  $\mathbf{z} = [z^{(1)}, \dots, z^{(N)}]$ . For each assignment  $\mathbf{z} \in \{1, \dots, k\}^N$ , you evaluate the following objective function:

$$J(\mathbf{z}) = \operatorname{argmin}_{\mathbf{c}_1, \dots, \mathbf{c}_k} \sum_{i=1}^N \|\mathbf{x}^{(i)} - \mathbf{c}_{z^{(i)}}\|_2^2$$

At the end, you pick the assignment  $\mathbf{z}$  that had lowest  $J(\mathbf{z})$ .

- (a) (1 point) Suppose we have  $N$  points and  $k$  clusters. For how many possible assignments  $\mathbf{z}$  does the brute force algorithm have to evaluate  $J(\mathbf{z})$ ?

Answer

- (b) (1 point) Suppose  $N = 1000$ ,  $k = 10$ , and it takes us 0.01 seconds to evaluate  $J(\mathbf{z})$  for a single assignment  $\mathbf{z}$ . How many seconds will the brute force algorithm take to check all assignments?

Answer

5. Initializing the centers has a big impact on the performance of the K-means clustering algorithm. Usually, we randomly initialize  $k$  cluster centers. However, there are other methods, namely, furthest point initialization and  $k$ -means++ initialization.

(a) (1 point) Is the clustering at convergence generated by furthest point initialization sensitive to outliers? Explain why or why not.

Your Answer

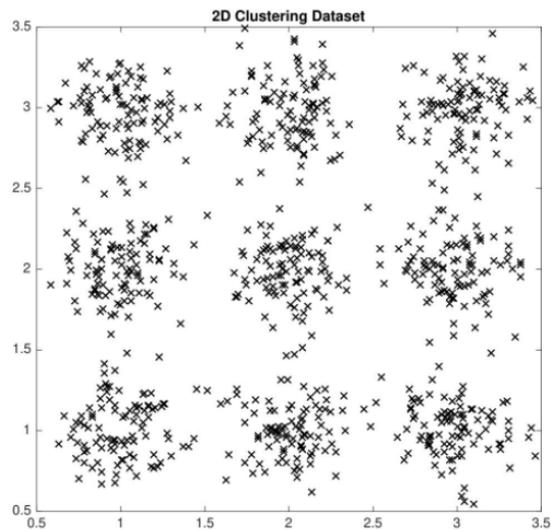


Figure 1: 2D Dataset

(b) (1 point) Explain in 1–2 sentences why using  $k$ -means++ initialization is more likely to choose one sample from each cluster than random initialization when using the dataset in Figure 1 above.

Your Answer

## 5 Ensemble Methods (10 points)

1. In the following question, we will explore two different kinds of ensemble methods: the halving algorithm and the doubling algorithm.

Consider the **halving algorithm**:

We first maintain a list of  $n$  weak classifiers  $h_1, \dots, h_n$  which have not yet made mistakes. The training labels and classifier predictions are in  $\{-1, 1\}$ . For each training sample  $(x, y)$ , we make prediction based on the majority vote of weak classifiers  $\hat{y} = \text{sign}(\sum_{i=1}^n h_i(x))$ . If the majority vote prediction does not equal to the label, we will eliminate all the  $h_i$  such that  $h_i(x) \neq y$ . The final aggregated classifier will be the ensemble of all the remaining classifiers.

- (a) (1 point) Assume there are a total of  $n$  classifiers initially, and at least one classifier left in the end. What would be the big-O bound of the total number of mistakes made in terms of  $n$ ?

Your Answer

- (b) (1 point) **Fill in the blank:** We can set the multiplicative penalty  $\beta = \underline{\hspace{2cm}}$  from the weighted majority algorithm to have the weighted majority algorithm be equivalent to the halving algorithm.

Your Answer

- (c) (1 point) What weak classifiers are guaranteed to be kept by the halving algorithm? (In one short sentence.)

Your Answer

Now we consider another algorithm, the **doubling algorithm**, which doubles the count of correct classifiers (by making identical copies of the correct classifiers each iteration) when the majority vote is incorrect.

- (d) (1 point) How could we modify the weighted majority algorithm to behave equivalently to the doubling algorithm for every input  $x$  and true label  $y$ ?

**Select all that apply:**

- Set the factor  $\beta = 2$ , and perform weight update of all classifiers  $h_i$  for which  $h_i(x) \neq y$  when the majority vote is incorrect.
- Set the factor  $\beta = \frac{1}{2}$ , and perform weight update of all classifiers  $h_i$  for which  $h_i(x) \neq y$  when the majority vote is correct.
- Set the factor  $\beta = \frac{1}{2}$ , and perform weight update of all classifiers  $h_i$  for which  $h_i(x) \neq y$  when the majority vote is incorrect.
- Set the factor  $\beta = 2$ , and perform weight update of all classifiers  $h_i$  for which  $h_i(x) = y$  when the majority vote is correct.
- Set the factor  $\beta = 2$ , and perform weight update of all classifiers  $h_i$  for which  $h_i(x) = y$  when the majority vote is incorrect.
- None of the above.

2. (1 point) Which of the following is true? **Select all that apply:**

- In the weighted majority algorithm, the weights associated with the weak learners are learned during training.
- In the AdaBoost algorithm, the weights associated with the weak learners are learned during training.
- In the weighted majority algorithm, the weak learners are learned during training.
- In the AdaBoost algorithm, the weak learners are learned during training.

3. (1 point) **True or False:** Consider some training point  $(x^{(i)}, y^{(i)})$  to the AdaBoost algorithm. If for all  $t$ , the weak learner  $h_t$  learned during training at time  $t$  correctly classifies  $h_t(x^{(i)}) = y^{(i)}$ , there will eventually be a finite time  $t$  such that the weight assigned to  $x^{(i)}$  in the training distribution  $\mathcal{D}_t$  reaches exactly 0.

**Select one:**

- True
- False

4. (1 point) **True or False:** If AdaBoost reaches perfect training accuracy, all weak learners created in subsequent iterations will be identical (i.e., they will produce the same output on any input). Assume we are using deterministically selected weak learners.

**Select one:**

- True
- False

5. In the following question, we will examine the generalization error of AdaBoost using a concept known as the *classification margin*.

Throughout the question, use the following definitions:

- $N$ : The number of training samples.
- $S = \{(x^{(1)}, y^{(1)}), \dots, (x^{(N)}, y^{(N)})\}$ : The training samples with binary labels ( $y^{(i)} \in \{-1, +1\}$ ).
- $\mathcal{D}_t(i)$ : The weight assigned to training example  $i$  at time  $t$ . Note that  $\sum_i \mathcal{D}_t(i) = 1$ .
- $h_t$ : The weak learner constructed at time  $t$ .
- $\epsilon_t$ : The error of  $h_t$  on  $\mathcal{D}_t$ .
- $Z_t$ : The normalization factor for the distribution update at time  $t$ .
- $\alpha_t$ : The weight assigned to the learner  $h_t$  in the composite hypothesis.
- $f_t(x) = (\sum_{t'=1}^t \alpha_{t'} h_{t'}(x)) / (\sum_{t'=1}^t \alpha_{t'})$ : The aggregated vote of the weak learners, rescaled based on the total weight.

For a binary classification task, assume that we use a probabilistic classifier that provides a probability distribution over the possible labels (i.e.  $p(y|x)$  for  $y \in \{+1, -1\}$ ). The classifier output is the label with highest probability. We define the *classification margin* for an input as the signed difference between the probability assigned to the correct label and the incorrect label  $p_{correct} - p_{incorrect}$ , which takes on values in the range  $[-1, 1]$ . Recall from recitation that  $\text{margin}_t(x^{(i)}, y^{(i)}) = y^{(i)} f_t(x^{(i)})$ .

(a) (1 point) Recall the update AdaBoost performs on the distribution of weights:

- $\mathcal{D}_1(i) = 1/N$
- $\mathcal{D}_{t+1}(i) = \mathcal{D}_t(i) \frac{\exp(-y^{(i)} \alpha_t h_t(x^{(i)}))}{Z_t} = \frac{1}{N} \left( \prod_{t'=1}^t \frac{1}{Z_{t'}} \right) \exp(-\sum_{t'=1}^t y^{(i)} \alpha_{t'} h_{t'}(x^{(i)}))$

We define  $C_{t+1} = \frac{1}{N} \left( \prod_{t'=1}^t \frac{1}{Z_{t'}} \right)$  and  $M_{t+1}(i) = -\sum_{t'=1}^t y^{(i)} \alpha_{t'} h_{t'}(x^{(i)})$ . We then have

$$\mathcal{D}_{t+1}(i) = C_{t+1} \exp(M_{t+1}(i))$$

Let  $\alpha = \sum_{t'=1}^t \alpha_{t'}$ . Rewrite  $M_{t+1}(i)$  in terms of  $\text{margin}_t(x^{(i)}, y^{(i)})$  and  $\alpha$ . (Hint: first rewrite  $M_{t+1}(i)$  in terms of  $y^{(i)}, \alpha, f_t, x^{(i)}$ , then apply our given formula for the margin).

Your Answer

- (b) (1 point) Note that  $C_{t+1}, \alpha$  are treated as positive constants with respect to the input points. Using the classification margin and the above formulation of the weights assigned by AdaBoost, fill in the blanks to describe which points AdaBoost assigns high weight to at time  $t$ .

At time  $t$ , AdaBoost assigns higher weight to points  $x^{(i)}$  with \_\_\_\_\_ value of margin on the current ensemble classifier (i.e.,  $\text{margin}_t(x^{(i)}, y^{(i)})$ ).

**Select one:**

- higher absolute
- higher signed
- lower absolute
- lower signed

- (c) (1 point) How does this weighting behavior explain the empirical result of test error continuing to decrease after training error has converged?

Your Answer

## 6 Recommender Systems (9 points)

1. (2 points) In which of the following situations will a collaborative filtering system be a more appropriate learning algorithm than a linear or logistic regression model?

**Select all that apply:**

- You manage an online bookstore, and you have book ratings and sales data from many users. For each user, you want to recommend other books she will enjoy, based on her own ratings and the ratings of other users.
- You manage an online bookstore, and you have book ratings and sales data from many users. You want to learn to predict the expected sales volume (number of books sold) as a function of the average rating of a book.
- You run an online news aggregator, and for every user, you know some subset of articles that the user likes and some different subset that the user dislikes. You want to use this to find other articles that the user likes.
- You've written a piece of software that downloads news articles from many news websites. In your system, you also keep track of which articles you personally like and which ones you dislike, and the system also stores away features of these articles (e.g., word counts, name of author). Using this information, you want to build a system to try to find additional new articles that you personally will like.
- None of the above

2. (2 points) What is the basic intuition behind matrix factorization?

**Select all that apply:**

- That content filtering and collaborative filtering are just two different factorizations of the same rating matrix.
- That factoring user and item matrices can partition the users and items into clusters that can be treated identically, reducing the complexity of making recommendations.
- The user-user and item-item correlations are more efficiently computed by factoring matrices, even when including the cost of factoring matrices.
- That users and items can be well described in a shared low dimensional space that can be computed from the rating matrices.
- None of the above

3. Neural the Narwhal decides to set up a friend-recommendation system for all the students in 10-301/601, of which there are  $N = 10,301,601$ . Ideally, Neural would store the full  $N \times N$  matrix  $M$ , where  $M_{ij}$  is 1 if student  $i$  and  $j$  are friends and 0 if student  $i$  and  $j$  are nemeses, or null if students  $i$  and  $j$  have never met. Assume that these are the only possible relationships between 2 people and that all relationships are symmetric (so it cannot be the case that student  $i$  thinks student  $j$  is their friend while  $j$  thinks  $i$  is their nemesis). Unfortunately, storing  $M$  in its entirety would take over 10 TB of storage, which Neural cannot afford on a TA salary. Neural instead uses the following procedure to approximate  $M$  as  $UU^T$  for some low rank  $N \times d$  matrix  $U$ , where each row of  $U$  corresponds to a student.

---

```

1: Given learning rate  $\eta$ , ground truth relationships  $M$ 
2: Randomly initialize user embedding matrix  $U \in \mathbb{R}^{N \times d}$ 
3: while not converged do
4:   Sample  $i \sim \text{Uniform}(1, N), j \sim \text{Uniform}(1, N)$ 
5:    $\hat{M}_{ij} \leftarrow \sigma(\vec{u}_i^T \vec{u}_j)$  //  $\sigma$  is sigmoid function
6:    $\mathcal{L}(\vec{u}_i, \vec{u}_j) \leftarrow -M_{ij} \log(\hat{M}_{ij}) - (1 - M_{ij}) \log(1 - \hat{M}_{ij})$  // Compute logistic loss
7:    $\vec{g}_i \leftarrow \nabla_{\vec{u}_i} \mathcal{L}(\vec{u}_i, \vec{u}_j)$  // Compute and perform gradient updates
8:    $\vec{g}_j \leftarrow \nabla_{\vec{u}_j} \mathcal{L}(\vec{u}_i, \vec{u}_j)$ 
9:    $\vec{u}_i \leftarrow \vec{u}_i - \eta \cdot \vec{g}_i$ 
10:   $\vec{u}_j \leftarrow \vec{u}_j - \eta \cdot \vec{g}_j$ 

```

---

- (a) (2 points) One of the lines in the above algorithm is incorrect. State the line number, and provide the correct line.

Line #	Fixed line of code

- (b) (1 point) Based on the loss function given on line 6, derive an expression for  $\nabla_{\vec{u}_j} \mathcal{L}(\vec{u}_i, \vec{u}_j)$  in terms of  $\vec{u}_i$ ,  $\vec{u}_j$ , and  $M_{ij}$ . Note that  $\sigma$  denotes the sigmoid function and  $\log$  is natural log.

Your Answer

- (c) (2 points) Why is it appropriate here to factorize  $M$  as the product of a single matrix by itself  $UU^T$  rather than as the product of two distinct matrices  $VW^T$ ?

**Select all that apply:**

- $UU^T$  enforces that our approximation is symmetric, while  $VW^T$  is not necessarily symmetric.
- SGD is guaranteed to converge faster and to a better optimum for  $U$  because we have fewer parameters to learn.
- We can parallelize training for  $U$ , whereas we could not parallelize training for  $V$  and  $W$ .
- We wish to model relationships between objects of a single type, not between objects of two different types.
- None of the above.

## 7 Collaboration Questions

After you have completed all other components of this assignment, report your answers to these questions regarding the collaboration policy. Details of the policy can be found [here](#).

1. Did you receive any help whatsoever from anyone in solving this assignment? If so, include full details.
2. Did you give any help whatsoever to anyone in solving this assignment? If so, include full details.
3. Did you find or come across code that implements any part of this assignment? If so, include full details.

Your Answer