UNIVERSITY OF MICHIGAN
Department of Electrical Engineering and Computer Science
EECS 445 — Introduction to Machine Learning
Fall 2022

Homework 3 (100 points)
Due: Tuesday, November 22nd at 10 pm

Submission: Please upload your completed assignment to Gradescope. Your submission can be either handwritten or typed. Assign all pages (including pages containing code) to their respective questions. Incorrectly assigned pages will not be graded.

0 Code Prerequisites

This homework will require you to write Python code and analyze results. Please setup a new Python 3.10 virtual environment and install code requirements for this homework using `requirements.txt`.

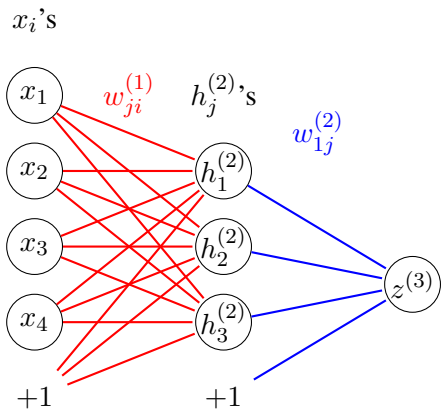
```
$ python3 -m pip install -r requirements.txt
```

Verify your installation by running `pip list` and comparing the installed verisions to `requirements.txt`.

- Confirm that you have `scikit-learn` version 1.0.2 version installed. Certain questions will not work if any later version is installed.
- If you are currently using `scipy` version 1.9.3 (or later), you should be sufficiently up-to-date.

1 Backpropagation [20 pts]

Consider the fully connected neural network given below. We name the output vectors and weights of these layers using the convention specified in the figure: Here $w_{ji}^{(1)}$ is the weight on the edge connecting x_i to $h_j^{(2)}$ and $w_{1j}^{(2)}$ is the weight connecting $h_j^{(2)}$ to $z^{(3)}$. As before, interpret $w_{j0}^{(1)}$ and $w_{10}^{(2)}$ as the weights on edges connecting the bias nodes to $h_j^{(2)}$ and $z^{(3)}$ respectively.



Suppose the loss function is squared loss: $L(y, z) = \frac{1}{2}(y - z)^2$ and the hidden neurons apply the ReLU activation function. Assume the output neuron **does not** use an activation function.

(a) [6 pts] Write out expressions for $h_1^{(2)}$, $h_2^{(2)}$, and $h_3^{(2)}$ in terms of the weights of the first layer of weights $w_{ji}^{(1)}$ and the inputs x_i (including $w_{j0}^{(1)}$). Then, write out the expression for $z^{(3)}$ in terms of the second layer of weights $w_{1j}^{(2)}$ (including $w_{10}^{(2)}$) and hidden neurons $h_j^{(2)}$.

(b) [4 pts] Consider an arbitrary weight in the first layer $w_{ji}^{(1)}$ and suppose we wanted to update this weight via back propagation and stochastic gradient descent. Write out the update step for this weight in terms of $w_{ji_{\text{new}}}^{(1)}$, $w_{ji_{\text{old}}}^{(1)}$, η, and the expanded form of $\frac{\partial L}{\partial w_{ji}^{(1)}}$. There is no need to compute the partial derivatives in this question.

(c) [10 pts] Now we will perform an update. First, use forward propagation to calculate all neuron outputs in the network (including hidden layer outputs). Then, calculate the updated weight $w_{21_{\text{new}}}^{(1)}$ after a single SGD iteration with the step size $\eta = 1$ using a training input $\bar{x} = [1, -1, 1, 0]^T$ and a label $y = 3$ and show your work.
The current weights are given in the tables below:

Weights	$h_1^{(2)}$	$h_2^{(2)}$	$h_3^{(2)}$
x_1	$+1$	-1	0
x_2	-1	0	$+1$
x_3	0	$+1$	$+1$
x_4	-1	0	-1
$+1$	$+1$	-1	$+1$

Weights	$z^{(3)}$
$h_1^{(2)}$	$+1$
$h_2^{(2)}$	-1
$h_3^{(2)}$	$+1$
$+1$	$+1$

2 Clustering [36 pts]

In this problem, we will implement spectral and k-means clustering and compare the results of the two algorithms. For all conceptual questions, assume that the number of clusters $k \leq n$, the number of points.

2.1 Spectral Clustering [14 pts]

In this problem, we will be exploring spectral clustering. You are allowed to use `scikit-learn` functions in your implementation. You will write your code in the file `spectral.py` included in the starter code. Consider the dataset `mickey.csv` that we have provided, consisting of 400 data points in a two-dimensional feature space.

(a) [2 pts] **Implement** the `plot_spectral_num_cluster` function in `spectral.py` by following the instructions in the code comments. Include a screenshot (or equivalent) of your implemented function as your solution to this question.

(b) [2 pts] **Implement** the `visualize_spectral` function in `spectral.py` by following the instructions in the code comments. Include a screenshot (or equivalent) of your implemented function as your solution to this question.

(c) [10 pts] **Run** `spectral.py` and **answer** the following questions. **Please include all generated plots in your write-up!**

i. [4 pts] The `plot_spectral_num_cluster` function will **plot** the smallest 10 eigenvalues of the Laplacian matrix. **Include** this plot in your write-up. **Explain** which number of clusters you would choose based on the plot.

ii. [6 pts] The `visualize_spectral` function creates a plot for **visualizing** the spectral clustering assignments for `gammas = [0.1, 1, 10]`. **Include** the generated visualization in your write-up. **Explain** which value of `gamma` you would choose based on the visualization and why this value of `gamma` works well for this dataset.

2.2 k-Means Clustering [16 pts]

In this problem, you will be exploring the k-means clustering algorithm as well as a variant of the algorithm, k-means++, which has the additional property of encouraging good cluster centroid initializations. These are simple yet powerful unsupervised methods for grouping similar examples within a dataset. Consider the dataset `unbalanced.csv`, consisting of 6500 points with two features each.

(a) [2 pts] Why are poor centroid initializations often a problem?

(b) [2 pts] **Implement** `visualize_kmeans` function in `kmeans.py` by following the instructions in the code comments. Include a screenshot (or equivalent) of your implemented function as your solution to this question.

(c) [12 pts] **Run** `kmeans.py` and **answer** the following questions. The program takes in three command-line arguments:

- Data file: a `.csv` file

- k: an integer number of clusters
- Initialization method: either `k-means++` or `random`

In this part, use the following command, where `<INIT_METHOD>` is replaced with the appropriate initialization method, as described above.

```
$ python3 kmeans.py unbalanced.csv 8 <INIT_METHOD>
```

i. [4 pts] **Print** the final value of the objective function using random initialization as well as k-means++. Make sure to set `random_state = 20` in the `KMeans` object to allow for proper comparison. **Compare** the two values and explain any discrepancies you might see.

ii. [4 pts] **Visualize** the resulting clusters using both random initialization as well as k-means++. **Include** your plots below.

iii. [4 pts] **Compare** the two generated plots. Which method provides a better solution, and why does that method work better?

2.3 Comparison [6 pts]

We now revisit the `mickey.csv` dataset to compare the performance of both the k-means and spectral clustering algorithms.

(a) [2 pts] **Run** `kmeans.py` on `mickey.csv` with 3 clusters and `k-means++` initialization. **Include** your plot below - you may want to change the filename the plot is saved under so that your earlier plot for Question 2.2(c)(ii) is not overwritten. Then, run the following command:

```
$ python3 kmeans.py mickey.csv 3 k-means++
```

(b) [4 pts] **Compare** the graph you chose in 2.1(c)(ii) to the one above. Which method provides a better solution, and why does that method work better?

3 Hierarchical Clustering [12 pts]

In this problem, we will use `scikit-learn` functions to implement hierarchical clustering. Consider the dataset `campus.csv` which represents the expected walking time between five buildings on campus as seen in Table 1.

Table 1: The expected time of arrival between five campus buildings

ETA (min.)	Angell Hall	Mason Hall	GGBL	BBB	Chrysler	FXB
Angell Hall	0	9	47	44	40	48
Mason Hall	9	0	48	45	40	49
GGBL	47	48	0	4	10	2
BBB	44	45	4	0	8	5
Chrysler	40	40	10	8	0	8
FXB	48	49	2	5	8	0

(a) [6 pts] **Implement** `plot_hierarchical_dendrogram` function in `hierarchical.py` by following the instructions in the code comments.

(b) [6 pts] Run `hierarchical.py`. This will call the `plot_hierarchical_dendrogram` function to **plot** the dendrogram. **List** the cluster assignments when there are 2 and 4 total clusters, respectively, based on your dendrogram.

4 Investigating Spectral Clustering [10 pts]

Here you will investigate how the spectral clustering algorithm makes cluster assignments. The file `spectralq4.py` provides starter code for parts b through e. Figure 7 shows the graph that we will be working with.

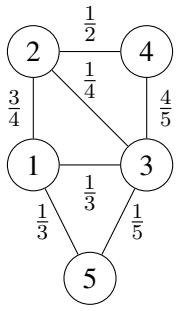


Figure 7: Spectral Clustering graph

(a) [3 pts] Compute the similarity matrix W, degree matrix D, and the corresponding graph Laplacian L based on the graph in Figure 7.

(b) [2 pts] For the remaining subparts please use the starter code provided in `spectralq4.py`. Using the `scipy.linalg.eigh` method, solve for the eigenvectors corresponding to the $k = 2$ lowest eigenvalues. List these eigenvalues and eigenvectors (round to four decimal places). Refer to the documentation for more details: https://docs.scipy.org/doc/scipy/reference/generated/scipy.linalg.eigh.html. Hint: Note the subset_by_index argument.

(c) [2 pts] Based on the eigenvalues and the eigenvectors computed in part b, what points will be grouped together in the cluster assignments? (Hint: plot the rows of the eigenvectors.)

(d) [1 pt] Using similar methods as in part (b), solve for the eigenvectors corresponding to the $k = 3$ lowest eigenvalues. List these eigenvalues and eigenvectors (round to four decimal places).

(e) [2 pts] Now that we have set k=3, what points will be grouped together in the cluster assignments?

5 UV Decomposition [22 pts]

We can think of collaborative filtering as matrix factorization. Given a binary $n \times m$ matrix Y in which there are empty cells, we want to use the low rank matrix UV^T to approximate the observed binary labels

in Y. Let $Y_{ij} \in \{-1, 1\}$ if the (i, j) entry is observed, and $Y_{ij} = 0$ otherwise. Then we would like to find U and V such that the following condition is satisfied:

$$Y_{ij}[UV^T]_{ij} = Y_{ij}(\bar{u}^{(i)} \cdot \bar{v}^{(j)}) > 0 \text{ whenever } Y_{ij} \neq 0$$

It is often advantageous to potentially avoid satisfying all the constraints, e.g., if some of the labels may be mistakes. We might also suspect that we gain something by turning the estimation problem into multiple maximum margin problems, for $\bar{u}^{(i)}$'s given $\{\bar{v}^{(j)}\}$ and for $\bar{v}^{(j)}$'s given $\{\bar{u}^{(i)}\}$. The formulation with slack is given by:

$$\begin{aligned} \min_{U,V,\xi} \quad & \sum_{i=1}^{n} \frac{1}{2}\|\bar{u}^{(i)}\|^2 + \sum_{j=1}^{m} \frac{1}{2}\|\bar{v}^{(j)}\|^2 + C \sum_{i,j:Y_{ij}\neq 0} \xi_{ij} \\ \text{subject to} \quad & Y_{ij}(\bar{u}^{(i)} \cdot \bar{v}^{(j)}) \geq 1 - \xi_{ij}, \\ & \xi_{ij} \geq 0 \text{ for all } (i, j) \text{ where } Y_{ij} \neq 0 \end{aligned}$$

(a) [6 pts] If we fix V (i.e., fix all $\bar{v}^{(1)}, ..., \bar{v}^{(m)}$), the optimization problem reduces to n independent optimization problems. Define these optimization problems for $\bar{u}^{(i)}$ where $i = 1, ..., n$. What previous algorithm that we have learned does this optimization problem resemble? (Hint: write out minimization problems with respect to each $\bar{u}^{(i)}$.)

(b) [16 pts] Consider a simple two-user and two-movie example and let the observed rating matrix and our initial guess for V be given by:

$$Y = \begin{bmatrix} 1 & 1 \\ -1 & 0 \end{bmatrix} \quad V = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

i. [4 pts] Now we would like to solve by hand $\hat{U} = [\bar{u}^{(1)}, \bar{u}^{(2)}]^T$, given the initial V. First, **write out** the explicit form of the optimization problems for $\bar{u}^{(1)}$ and $\bar{u}^{(2)}$ (i.e. no i, j indices and summation symbol and substitute in the relevant values from the Y and V matrices).

ii. [8 pts] Let's take $C = \infty$. **Solve** the optimization problems for $\bar{u}^{(1)}$ and $\bar{u}^{(2)}$.

iii. [2 pts] Given the initial V and your solution $\hat{U}$ = [$\bar{u}^{(1)*}$, $\bar{u}^{(2)*}$], can you predict Y_{22} as label -1 or 1? If so, what's your prediction? If not, please explain.

iv. [2 pts] In real-life recommender system development, we sometimes run into the issue where the predictions of some labels in $\hat{Y}$ are never clear to us; a specific instance of this problem is the Cold Start Problem. If you were a recommender system developer, what would you do if you encountered such a problem (i.e., no initial label and the algorithm you are using does not give a reasonable prediction or does not give a prediction at all)?

REMEMBER to submit your completed assignment to Gradescope by 10:00pm ET on Tuesday, Nov. 22. Assign all pages (including pages containing code) to their respective questions. Incorrectly assigned pages will not be graded.