Homework 9

Due: Thursday, April 30, 2026 at 12:00pm **(Noon)**

Written Assignment

Problem 1: Principal Component Analysis

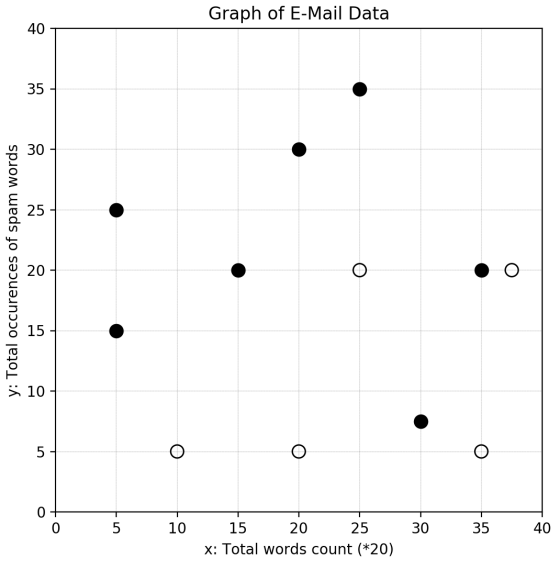
(20 points)

In this question, you will build a spam filter using using Principal Component Analysis (PCA) and the K-Nearest Neighbour algorithm.

The K-Nearest Neighbour algorithm is a simple-to-implement classifier in which a an example is labeled with the most common label among the K closest examples in the training data. (Ties can be broken randomly.) Here, "closest" refers to Euclidean distance.

You will be using PCA to transform the e-mail data from 2 dimensions (and a label) to 1 dimension (and a label). Each e-mail has two features: number of words in the email and total occurrences of the spam words "credit" and "dollars". The following graph shows the training data where filled circles are spam e-mails and unfilled circles are non-spam emails. The x-axis (Total words count) has been scaled for your convenience.

We have found the first principal component: $(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}})$. Therefore the linear subspace that PCA will project the data to is the line $y = x$. (Assume that we don't need to do any centering of the data.)



a. Draw the first principal component as a line ($y = x$ in this case) on the graph. Also draw the data points projected to this subspace with lines connecting them to the corresponding original data points.

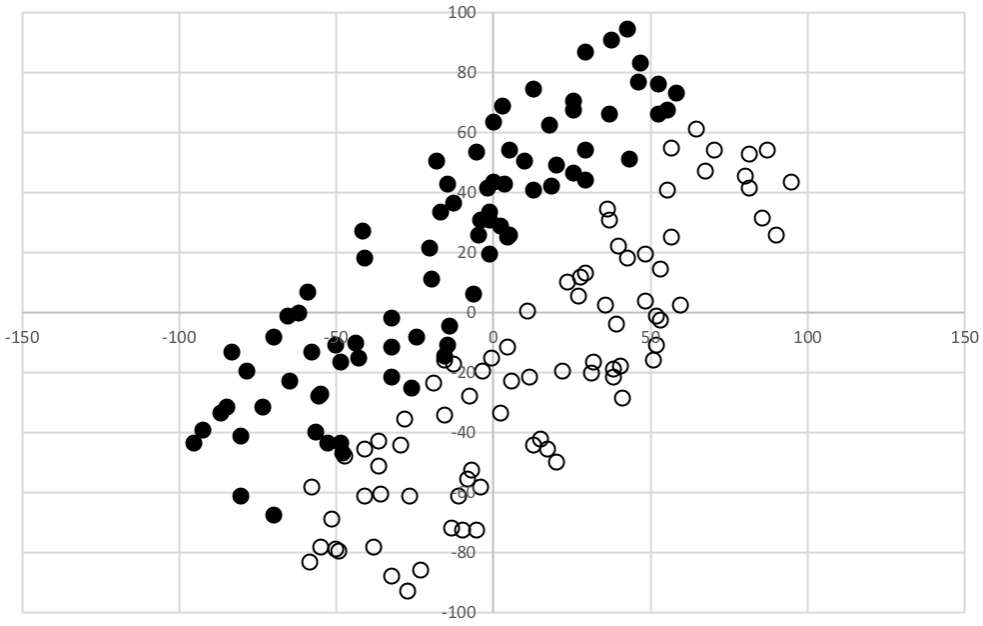
b. How would **1**-Nearest Neighbour algorithm classify an e-mail with 21 spam words in a total of 100 (x=5) words? Explain your answer.

c. How would **1**-Nearest Neighbour algorithm classify the same e-mail given in the previous question after transforming the data to 1-dimension? Explain your answer.

d. Are your answers same for part b and c? Why or why not?

Problem 2: Limitations of PCA (20 points)

Consider the following two-dimensional dataset:



a. Describe one type of machine learning model that would classify this data well. Explain your reasoning.

b. Draw the first principal component on the graph above. If PCA was used to reduce this data to one dimension, would the machine learning model from part (a) still classify the data well? Why or why not?

c. Is it possible to project the above data into a one-dimensional linear subspace in which the data remains linearly separable? If so, draw the subspace on the graph above. If not, explain your reasoning.

d. What does this example tell you about the limitations of PCA when used to pre-process data before classification?

Programming Assignment

Introduction

Professor Bach is on a mission to crown the best cafe in Providence. At each shop, he's left a jar of feedback slips for customers to drop in a quick rating from 0–9 on their way out. However after just a single day, he's ended up with a mountain of handwritten scraps that are impossible to sort by hand. You are tasked with developing a handwritten-digit identification system to automate the process.

In this assignment, you will implement an iterative method for clustering: k-means. Your implementation will be used for a k-means classifier, which will be trained and tested on handwritten digits dataset to classify an exact digit (0 to 9).

K-means Classifier

K-means is a clustering algorithm most often used for unsupervised machine learning. In unsupervised learning, the learner is given a dataset with no labels and attempts to learn some useful representation of the dataset. You may be wondering how K-means can used for classification in this assignment, as the training data is unlabeled. To address this, given a dataset $S = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$

- You will run K-means clustering on the unlabeled training data and plot the pixel representations of different cluster centers (centroids): $M = \{\mu_1 \ldots \mu_{10}\}$, for clusters $C = \{C_1 \ldots C_{10}\}$. With K = 10, these cluster centers should vaguely resemble the 10 digits (0-9).
- Using the pixel plots of the centroids, you will manually assign which digit each centroid represents.: $A = a_1 \ldots a_{10}$.
- To predict the label, y_{m+1} of a new datapoint $\mathbf{x}_{m+1}$, find the cluster center nearest to $\mathbf{x}_{m+1}$, μ_i, and predict using your assignment, a_i.

Note: You don't need to worry about changing your centroid assignments in between runs, as we've set the random seed in the stencil.

Stencil Code & Data

You can find and download the assignment here: `HW9 on Github`. If you have any problems, please consult the `Download and Handin Guide` .

You have been provided with the following stencil files:

- `main.py`: contains a main function to read data, run classifier and print/visualize results.

  To run your `main.py` program, run the command `python main.py` in the directory where `main.py` resides.
- `models.py`: contains the K-means classifier class that you will need to fill in.
- `kmeans.py`: contains helper functions for K-means clustering via iterative improvement that you will need to fill in.

8x8 Hand-written digits

In the `digits.csv` file, each row is an observation of a 8 x 8 hand-written digit (0 - 9), containing a label in the first column and 8 x 8 = 64 features (pixel values) in the rest of columns.

Data Format

We have written all the preprocessing code for you. The dataset is represented by a `namedtuple` with two fields:

- `data.inputs` is a $m \times p$ NumPy array that contains the binary features of the m examples, where p is the number of pixels in each example (64).
- `data.labels` is a m-dimensional NumPy array that contains the labels of the m examples.

You can find more infomation on `namedtuple` **here**.

Functions

1. `models.py`

   In this file, you will implement two functions. They are:

   - `KmeansClassifier`
     - **train()**: Learn K=10 cluster centroids (representatives) from the data that are robust (because they are estimated using a lot of data). Store cluster centroids as a Numpy array in *model attribute*
     - **predict()**: predict label of inputs using the label of closest centroid's assignment

2. `kmeans.py`:

   In this file, you will implement four functions:

   - **init_centroids()**: pick **K** random data points as cluster centers called centroids.
   - **assign_step()**: assign each data instance to its nearest cluster centroid using Euclidean distance measure.
   - **update_step()**: find the new cluster centroids by taking the average of its assigned data points.
   - **kmeans()**: run the K-means algorithm: initialize centroids, then repeat the assignment step and update step until the proportion the centroids [defined below] change between two iterations is below a tolerance threshold or the maximum iteration time is met. The tolerance threshold is passed into `kmeans()` as `tol` and tolerance is compared against the ratio of the norm of the difference between centroids and the norm of the original centroids.

   Note: You might also want to create a separate function that calculates the Euclidean distance between two data points in the `kmeans.py` file. Please feel free to do so.

3. `main.py`:

   You will not need to implement any functions in this file. However, you will need to do two things:

   - Uncomment the call to `plot_Kmeans` in main. This function will allow you to see the centroids that your k-means model learns.

     Please note: to complete the report you will need access to graphics on the machine you are working on. If you are running locally or through FastX/XQuartz, you do not have to worry about this. If you have been working exclusively through ssh, please read about how to set up remote work that is compatible with this assignment **here**. If there are any limitations to you doing this (e.g. not having access to a personal computer), please email Steve.
   - Fill in the `centroid_assignments` array using the results of `plot_Kmeans` in your call to `test_Kmeans`.

Project Report

1. Display your output of `plot_Kmeans()`. Does your plot match your expectations?
2. In this assignment, you implemented k-means using a Euclidean distance metric. Describe other distance metrics that can be used and how they cluster inputs.
3. What would you expect the cluster centers (centroids) to look like if use $K < 10$? $K > 10$?

Grading Breakdown

The grading breakdown for the assignment is as follows:

Written	40 points
Programming	50 points
Report	25 points
Total	140 points

Handing In

You will hand in both the written assignment and the coding portion on Gradescope, separately.

1. Your written assignment should be uploaded to Gradescope under "Homework 9."
2. Submit your HW9 GitHub repo containing all your source code and your project report named report.pdf on Gradescope under "Homework 9 Code." report.pdf should live in the root directory of your code folder; the autograder will check for the existence of this file and inform you if it is not found.

Please consult the `Download and Handin Guide` for additional guidance.

Anonymous Grading

You need to be graded anonymously, so do not write your name anywhere on your handin.

Academic Integrity

The CSCI 1420 Collaboration Policy applies to this assignment. As outlined in the Brown Academic Code and the Brown Academic Code, Graduate Student Edition, attempting to pass off another's work or AI-generated work as your own can result in failing the assignment, failing this course, or even dismissal or expulsion from Brown. More than that, you will be missing out on the goal of your education, which is the cultivation of your own mind, thoughts, and abilities. Please review this course's collaboration policy and if you have any questions, please contact the instructor.