CSCI567: Machine Learning	USC, Spring 2024
Homework 4	
Instructor: Vatsal Sharan	*Due: April 11 by 11:59 AM PST*

We would like to thank Gregory Valiant (Stanford) for kindly sharing some of the problems with us.

A reminder on collaboration policy and academic integrity: Our goal is to maintain an optimal learning environment. You can discuss the homework problems at a high level with other groups, but you should not look at any other group's solutions. Trying to find solutions online or from any other sources for any homework or project is prohibited, will result in zero grade and will be reported. To prevent any future plagiarism, uploading any material from the course (your solutions, quizzes etc.) on the internet is prohibited, and any violations will also be reported. Please be considerate, and help us help everyone get the best out of this course.

Please remember the Student Conduct Code (Section 11.00 of the USC Student Guidebook). General principles of academic honesty include the concept of respect for the intellectual property of others, the expectation that individual work will be submitted unless otherwise allowed by an instructor, and the obligations both to protect one's own academic work from misuse by others as well as to avoid using another's work as one's own. All students are expected to understand and abide by these principles. Students will be referred to the Office of Student Judicial Affairs and Community Standards for further review, should there be any suspicion of academic dishonesty.

Notes on notation:

- Unless stated otherwise, scalars are denoted by small letter in normal font, vectors are denoted by small letters in bold font and matrices are denoted by capital letters in bold font.
- $\|.\|$ means L2-norm unless specified otherwise, *i.e.*, $\|.\| = \|.\|_2$.

Instructions

We recommend that you use LaTeX to write up your homework solution. However, you can also scan handwritten notes. The homework will need to be submitted on Gradescope.

Theory-based Questions

Problem 1: Decision Trees (12pts)

Consider a binary dataset with 400 examples, where half of them belong to class A and the rest belong to class B. Next, consider two decision stumps (i.e. trees with depth 1) $\mathcal{T}_1$ and $\mathcal{T}_2$, each with two children. For $\mathcal{T}_1$, the left child has 150 examples in class A and 50 examples in class B. For $\mathcal{T}_2$, the left child has 0 examples in class A and 100 examples in class B. (You can infer the number of examples in the right child using the total number of examples.)

1.1 (6 pts) In class, we discussed entropy and Gini impurity as measures of uncertainty at a leaf. Another possible metric is the classification error at the leaf, assuming that the prediction at the leaf is the majority class among all examples that belong to that leaf. For each leaf of $\mathcal{T}_1$ and $\mathcal{T}_2$, compute the entropy (base e), Gini impurity and classification error. You can either exactly express the final numbers in terms of fractions and logarithms, or round them to two decimal places.

Classification error:

$$\epsilon_{1,L} = \frac{50}{150+50} = 0.25 \quad \text{(0.5 point)}$$

$$\epsilon_{1,R} = \frac{50}{50+150} = 0.25 \quad \text{(0.5 point)}$$

$$\epsilon_{2,L} = \frac{0}{0+100} = 0 \quad \text{(0.5 point)}$$

$$\epsilon_{2,R} = \frac{100}{200+100} \approx 0.33 \quad \text{(0.5 point)}$$

Entropy:

$$H_{1,L} = -\frac{150}{150+50}\ln(\frac{150}{150+50}) - \frac{50}{150+50}\ln(\frac{50}{150+50}) \approx 0.56 \quad \text{(0.5 point)}$$

$$H_{1,R} = -\frac{50}{150+50}\ln(\frac{50}{150+50}) - \frac{150}{150+50}\ln(\frac{150}{150+50}) \approx 0.56 \quad \text{(0.5 point)}$$

$$H_{2,L} = -\frac{0}{0+100}\ln(\frac{0}{0+100}) - \frac{100}{0+100}\ln(\frac{100}{0+100}) = 0 \quad \text{(0.5 point)}$$

$$H_{2,R} = -\frac{200}{200+100}\ln(\frac{200}{200+100}) - \frac{100}{100+200}\ln(\frac{200}{200+100}) \approx 0.64 \quad \text{(0.5 point)}$$

Gini impurity:

$$G_{1,L} = 1 - (\frac{150}{150+50})^2 - (\frac{50}{150+50})^2 = 0.375 \approx 0.38 \quad \text{(0.5 point)}$$

$$G_{1,R} = 1 - (\frac{50}{150+50})^2 - (\frac{150}{150+50})^2 = 0.375 \approx 0.38 \quad \text{(0.5 point)}$$

$$G_{2,L} = 1 - (\frac{0}{0+100})^2 - (\frac{100}{0+100})^2 = 0 \quad \text{(0.5 point)}$$

$$G_{2,R} = 1 - (\frac{200}{200+100})^2 - (\frac{100}{200+100})^2 \approx 0.44 \quad \text{(0.5 point)}$$

1.2 (6 pts) Compare the quality of $\mathcal{T}_1$ and $\mathcal{T}_2$ (that is, the two different splits of the root) based on conditional entropy (base e), weighted Gini impurity, and total classification error. Intuitively, which of $\mathcal{T}_1$ or $\mathcal{T}_2$ appears to be a better split to you (there may not necessarily be one correct answer to this)? Based on your conditional entropy, Gini

impurity, and classification error calculations, which of the metrics appear to be more suitable choices to decide which variable to split on?

Let $p_1 = \frac{150+50}{400} = 0.5$ be the fraction of examples that belong to the left leaf of $\mathcal{T}_1$, and $p_2 = \frac{0+100}{400} = 0.25$ be the fraction of examples that belong to the left leaf of $\mathcal{T}_2$. Then the total classification error for $\mathcal{T}_1$ and $\mathcal{T}_2$ are respectively:

$$\epsilon_1 = p_1\epsilon_{1,L} + (1-p_1)\epsilon_{1,R} = 0.25 \quad \text{(0.5 point)}$$
$$\epsilon_2 = p_2\epsilon_{2,L} + (1-p_2)\epsilon_{2,R} = 0.25 \quad \text{(0.5 point)}$$

So they are as good in terms of classification error. (0.5 points)

The conditional entropy for $\mathcal{T}_1$ and $\mathcal{T}_2$ are respectively:

$$\epsilon_1 = p_1 H_{1,L} + (1-p_1)H_{1,R} \approx 0.56 \quad \text{(0.5 point)}$$
$$\epsilon_2 = p_2 H_{2,L} + (1-p_2)H_{2,R} \approx 0.48 \quad \text{(0.5 point)}$$

So $\mathcal{T}_2$ is better in terms of conditional entropy. (0.5 points)

The weighted Gini impurity for $\mathcal{T}_1$ and $\mathcal{T}_2$ are respectively:

$$\epsilon_1 = p_1 G_{1,L} + (1-p_1)G_{1,R} \approx 0.38 \quad \text{(0.5 point)}$$
$$\epsilon_2 = p_2 G_{2,L} + (1-p_2)G_{2,R} \approx 0.33 \quad \text{(0.5 point)}$$

So $\mathcal{T}_2$ is also better in terms of weighted Gini impurity. (0.5 points)

Since $\mathcal{T}_2$ leads to a pure (100% certain) node, it is probably preferable to $\mathcal{T}_1$. Therefore conditional entropy and Gini impurity appear to be more suitable measures to decide which variable to split on. They are more sensitive to changes in the node probabilities than classification error. (1.5 points)

Problem 2: Gaussian Mixture Model and EM (10pts+5pts Bonus)

In class, we applied EM to learn Gaussian Mixture Models (GMMs) and showed the M-Step without proof. Now, it is time that you prove it.

Consider a GMM with the following PDF of $\mathbf{x}_i$:

$$p(\mathbf{x}_i) = \sum_{j=1}^{k} \pi_j N(\mathbf{x}_i \mid \mu_j, \Sigma_j) = \sum_{j=1}^{k} \frac{\pi_j}{(\sqrt{2\pi})^d |\Sigma_j|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(\mathbf{x}_i - \mu_j)^T \Sigma_j^{-1} (\mathbf{x}_i - \mu_j)\right)$$

where k is the number of Gaussian components, d is dimension of a data point $\mathbf{x}_i$ and N is the usual Gaussian pdf ($|\Sigma|$ in the pdf denotes the determinant of matrix Σ). This GMM has k tuples of model parameters $\{(\mu_j, \Sigma_j, \pi_j)\}_{j=1}^k$, where the parameters represent the mean vector, covariance matrix, and component weight of the j-th Gaussian component. For simplicity, we further assume that all components are isotropic Gaussian, i.e., $\Sigma_j = \sigma_j^2 I$.

2.1 (10 pts) Find the MLE of *the expected complete log-likelihood*. Equivalently, find the optimal solution to the following optimization problem.

$$\begin{aligned} \underset{\pi_j, \mu_j, \Sigma_j}{\operatorname{argmax}} & \sum_i \sum_j \gamma_{ij} \ln \pi_j + \sum_i \sum_j \gamma_{ij} \ln N(\mathbf{x}_i \mid \mu_j, \Sigma_j) \\ \text{s.t. } & \pi_j \geq 0 \\ & \sum_{j=1}^{k} \pi_j = 1 \end{aligned}$$

where γ_{ij} is the posterior of latent variables computed from the E-Step.

You can use the following fact: Given $a_1, \ldots, a_k \in \mathbb{R}^+$, the solution to the following optimization problem over $q_1, \ldots, q_k$:

$$\begin{aligned} \underset{q_j}{\operatorname{argmax}} & \sum_{j=1}^{k} a_j \ln q_j, \\ \text{s.t. } & q_j \geq 0, \\ & \sum_{j=1}^{k} q_j = 1. \end{aligned}$$

is given by:

$$q_j^* = \frac{a_j}{\sum_{k'} a_{k'}}$$

To find $\pi_1, \ldots, \pi_k$, we simply solve

$$\begin{aligned} \underset{\boldsymbol{\pi}}{\operatorname{argmax}} & \sum_j \sum_i \gamma_{ij} \ln \pi_j. \\ \text{s.t. } & \pi_j \geq 0 \\ & \sum_{j=1}^{k} \pi_j = 1 \end{aligned} \qquad \text{(2 points)}$$

The solution is

$$\pi_j^* = \frac{\sum_i \gamma_{ij}}{\sum_j \sum_i \gamma_{ij}} = \frac{\sum_i \gamma_{ij}}{\sum_i 1} = \frac{\sum_i \gamma_{ij}}{n}. \qquad \text{(1 points)}$$

To find μ_j and σ_j, we solve for each j

$$\begin{aligned}\operatorname*{argmax}_{\mu_j,\sigma_j} \sum_i \gamma_{ij} \ln N(\mathbf{x}_i \mid \mu_j, \sigma_j) &= \operatorname*{argmax}_{\mu_j,\sigma_j} \sum_i \gamma_{ij} \ln \left[\frac{1}{(\sqrt{2\pi}\sigma_j)^d} \exp\left(-\frac{1}{2\sigma_j^2} \|\mathbf{x}_i - \mu_j\|^2 \right) \right] \\ &= \operatorname*{argmax}_{\mu_j,\sigma_j} \sum_i \gamma_{ij} \left(-d \ln \sigma_j - \frac{\|\mathbf{x}_i - \mu_j\|^2}{2\sigma_j^2} \right).\end{aligned}$$

(3 points)

First we set the derivative w.r.t. μ_j to 0:

$$\frac{1}{\sigma_j^2} \sum_i \gamma_{ij} (\mathbf{x}_i - \mu_j) = 0,$$

which gives

$$\mu_j^* = \frac{\sum_i \gamma_{ij} \mathbf{x}_i}{\sum_i \gamma_{ij}} \qquad \text{(2 points)}$$

Next we set the derivative w.r.t. σ_j to 0:

$$\sum_i \gamma_{ij} \left(-\frac{d}{\sigma_j} + \frac{\|\mathbf{x}_i - \mu_j\|^2}{\sigma_j^3} \right) = 0.$$

Solving for σ_j gives

$$(\sigma_j^*)^2 = \frac{\sum_i \gamma_{ij} \|\mathbf{x}_i - \mu_j^*\|^2}{d \sum_i \gamma_{ij}}. \qquad \text{(2 points)}$$

2.2 (Bonus) (5 pts) The posterior probability of z in GMM can be seen as a *soft* assignment to the clusters; in contrast, k-means assign each data point to one cluster at each iteration (*hard* assignment). Show that if we set $\{\sigma_j, \pi_j\}_{j=1}^k$ in a particular way in the GMM model, then the cluster assignments given by the GMM reduce in the limit to the k-means clusters assignment (where the cluster centers $\{\mu_j\}_{j=1}^k$ remain the same for both the models). To verify your answer, you should derive $p(z_i = j|\boldsymbol{x}_i)$ for your choice.

Set all $\sigma_j = \sigma$ and $\pi_j = \frac{1}{k}$ (equal weights to all clusters), we have

$$p(\boldsymbol{x}_i, z_i = j) \propto \exp\left(-\frac{1}{2\sigma^2} \|\mathbf{x}_i - \mu_j\|^2 \right),$$

where constant terms are ignored. (2 points)

The posterior then becomes

$$p(z_i = j|\mathbf{x}_i) = \lim_{\sigma \to 0} \frac{\exp\{-\frac{1}{2\sigma^2}\|\mathbf{x}_i - \mu_j\|^2\}}{\sum_j \exp\{-\frac{1}{2\sigma^2}\|\mathbf{x}_i - \mu_j\|^2\}} \to \begin{cases} 1, & \text{if } j = \arg_c \min \|\mathbf{x}_i - \mu_c\|^2 \\ 0, & \text{otherwise.} \end{cases}$$

(3 points)

Programming-based Questions

As in previous homeworks, you need to have your coding environment setup for this part. We use python3 (version $\geq$ 3.7) in our programming-based questions. There are multiple ways you can install python3, for example:

- You can use conda to configure a python3 environment for all programming assignments.
- Alternatively, you can also use virtualenv to configure a python3 environment for all programming assignments

After you have a python3 environment, you will need to install the following python packages:

- numpy
- matplotlib (for plotting figures)

Note: You are **not allowed** to use other packages such as *tensorflow, pytorch, keras, scikit-learn, scipy*, etc. for 3.1-3.2. If you have other package requests, please ask first before using them. You are **allowed** to use any packages for 3.3-3.5.

Download the files for the programming part from `https://vatsalsharan.github.io/fall22/hw3.zip`.

Problem 3: Exploring Decision Trees and Random Forests (12pts)

In this question, we will observe the effect of different hyperparameters in training decision trees and random forests, and also visualize what features the random forest model is using to make predictions. We will do this on a Colab notebook `HW4-Exploring-Random-Forests.ipynb`.

Instructions to run the notebook: Upload this file to your USC Google Drive. Then, add Google Colab to your Google App by New $\rightarrow$ More $\rightarrow$ Connect more apps $\rightarrow$ Type in Google Colab and install it, as in Fig. 1. After that, you can run the notebook with your browser.

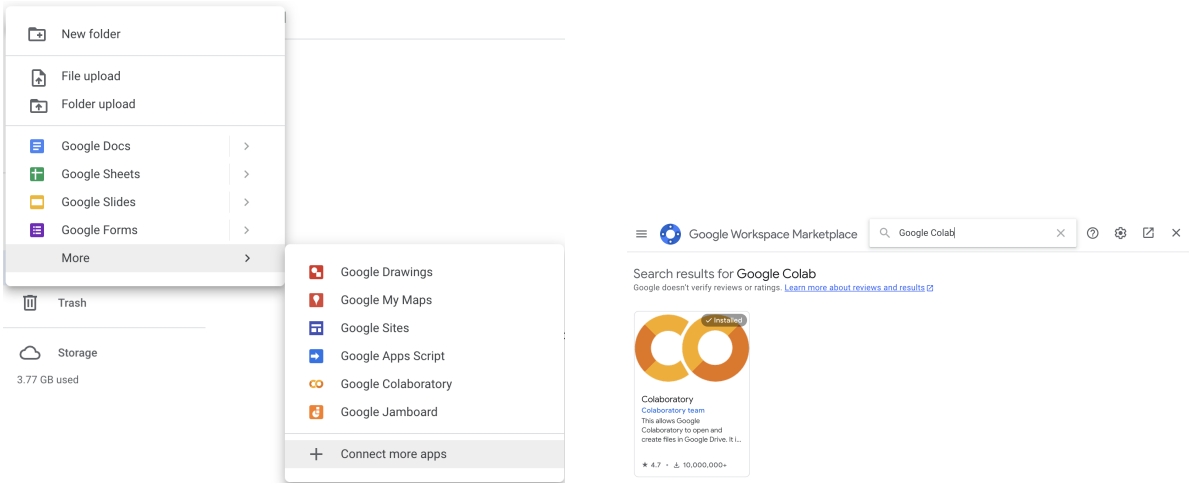

Figure 1: Screenshots showing how to install Colab.

We use a variant of the MNIST dataset for this problem. As mentioned in HW 3, the MNIST dataset contains images of handwritten digits (0, 1, 2, ..., 9) and is generally used for the (10) digit classification problem. Here, we work with a binary classfication task of predicting whether the digit is less than 5 or not. Fig. 2 shows some samples images from the dataset with original and binary labels, respectively.

You should go through the code we provide and understand what's happening, but for the purpose of answering the questions you will mainly have to run the code and understand the results. All the models (decision trees, random forests, etc.) are imported from the `sklearn` library. You should feel free to explore the role of other hyperparameters and other methods (such as bagging, boosting), and go through the documentation to better understand these things, but for this question, we will focus on decision trees and random forests.

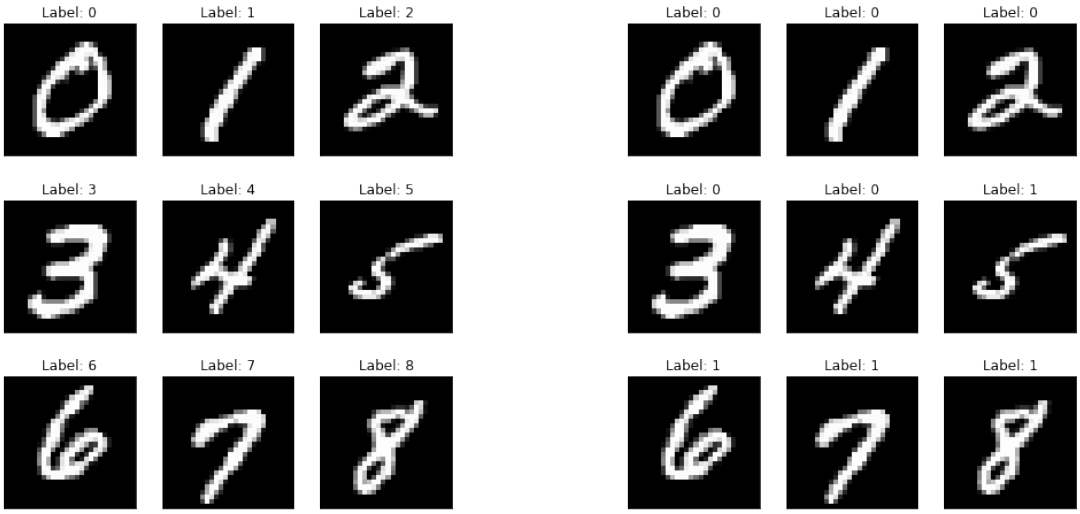

Figure 2: Some samples images from the MNIST dataset with original (left) and binary (right) labels, respectively.

We will look at the effect of 5 parameters:

- `max_depth`: The maximum depth of the decision tree.
- `n_estimators`: The number of decision trees in the forest.
- `min_samples_leaf`: The minimum number of samples required to be a leaf node.
- `max_samples`: The number of samples to draw from the training set to train each decision tree in the forest.
- `max_features`: The number of features to consider when looking for the best split for any node.

To observe the effect of a parameter, we look at train and test accuracy for different values of that parameter while keeping the rest of the parameters fixed.

3.1 (2 pts) The first set of plots shows the train and test accuracy for different values of `max_depth` using a decision tree and a random forest, respectively. How do the accuracies and the generalization gap vary with `max_depth` in these cases? For a particular value of `max_depth`, how does the generalization gap for the decision tree compare with that of the random forest? Explain your observations.

In both cases, as `max_depth` increases, the training accuracy keeps increasing, while the test accuracy increases upto a point and then saturates. The generalization gap also keeps increasing as `max_depth` is increased. (1pt)

For a fixed value of `max_depth`, the generalization gap for the decision tree is much higher than the random forest. This is because a single decision tree is trained using all the features and its predictions depend on a particular set of features, whereas the random forest leverages multiple decision trees to make its predictions, each of which is trained using only a subset of the features. This helps avoid the risk of overfitting onto a specific set of features. (1pt)

3.2 (2 pts) The next plot shows the train and test accuracy for different values of `n_estimators` for a random forest. Comment on your observations regarding the accuracies and the generalization gap. What value(s) (range or approximate values are enough) would you prefer to use for this parameter? Give reasons why. Hint: Are there any drawbacks to using very high values of `n_estimators`?

As `n_estimators` increases, the train and test accuracies increase upto a point and then saturate. The generalization gap remains similar throughout. (1pt)

As the accuracies saturate around n_estimators$= 50$, it seems like values in the range $50 - 100$ would work well. Although using higher values of n_estimators would give a similar performance, it comes at the cost of increased training time. (1pt)

For the next three parts of this question, we will also look at how the size of the training set influences the accuracy trends for a given parameter. In each case, for the first plot, the training set consists of 4000 samples whereas for the second plot, it contains 1000 samples.

3.3 (2 pts) The next set of plots shows the train and test accuracy for different values of min_samples_leaf for a random forest. Taking into account the behaviors for different training set sizes, explain your observations for very low and very high values of min_samples_leaf. What do you conclude from this trend? What could be the reasons for such a behavior?

In both cases, for low values of min_samples_leaf, the train/test accuracies as well as the generalization gap is high, whereas for high values of min_samples_leaf, the train/test accuracies, as well as the generalization gap, is low. (1pt)

There is a tradeoff between high train/test accuracy and a low generalization gap in both cases. Very low values of min_samples_leaf can make the model prone to overfitting on the training data, whereas, for very high values, the trees might be too shallow and thus, not learn a very complex decision boundary. (1pt)

3.4 (2 pts) The next set of plots shows the train and test accuracy for different values of max_samples for a random forest. What do you observe for very low and very high values of max_samples? Would you prefer to use low, intermediate, or high values for this parameter in both cases? Give reasons why. Hint: How does the size of the training set influence the choice of this parameter?

In both cases, for low values of max_samples, the train/test accuracies as well as the generalization gap is low, whereas for high values of max_samples, the train/test accuracies as well as the generalization gap is high. (1pt)

In the first case, both intermediate and high values seem fine, while in the second case, it is better to use intermediate values. In the first case, even if all the trees in the forest rely on all the samples, it doesn't hurt the accuracy as the training set is larger, so there are fewer chances of overfitting. However, in the second case, the smaller training set size makes the setting prone to overfitting when the trees use all the samples. Using different subsets of the training set for different trees helps avoid that in this case. (1pt)

3.5 (2 pts) The next set of plots shows the train and test accuracy for different values of max_features for a random forest. Comment on your observations regarding the accuracies and the generalization gap for the two training set sizes. What is the best range of values for this parameter in both cases? Is it similar/different? Explain your observations.

In the first plot, as max_features increases, both train and test accuracies increase upto a point, fluctuate a little, and then saturate. The generalization gap seems to be increasing (at a slow rate) throughout. In the second plot, as max_features increases, both train and test accuracies increase upto a point, fluctuate a little, and then start decreasing rapidly. The generalization gap increases throughout. (1pt)

In the first case, any value greater than 25 seems fine, while in the second case, values in the range $25 - 100$ seem to work well. In the first case, relying on a very large number of features doesn't seem to hurt as much as in the second case. In the first case, the training set is larger, so the sample distribution of the features is closer to the population distribution than in the second case. Thus, the second case is more prone to overfitting when using a large value of max_features. (1pt)

3.6 (2 pts) In class, we discussed how ensembles are usually not as interpretable as a single decision tree. While this is true, there are still ways to explore which features are used the most by our ensemble. We will explore one such technique in this part.

We visualize the *feature importances* of a random forest model trained for the binary classification task on the MNIST dataset. Intuitively, features with higher importance are the pixels which are used more often in the decision trees in the forest and which lead to better splits, i.e. which contribute more to improving the performance of the model. For more details, you can see Section 18.6.1 of the PML book. The last plot shows the importance of different pixels/portions of the image for a trained random forest model to make its predictions. What portions of the image does the model seem to be focusing on? In other words, can you think of reasons why the pixels with higher importance are indeed important for the prediction task (classifying whether the digit is smaller than 5 or not)? As is usually true for such open-ended questions, there can be multiple correct answers here and we're looking more for your reasoning than a specific answer.

The model seems to be focusing on portions of the image which are common in digits 5, 6, 8, and 9. As the prediction task is classifying whether the digit is smaller than 5 or not, this seems expected as the digits 0-4 have less common portions with this image. Hence, focusing on these portions allows the model to distinguish between these two classes. (2pts)

Problem 4: PCA for Learning Word Embeddings (35pts+10pts Bonus)

This question is about *word embeddings*. We saw word embeddings in class in lecture 7. As we discussed then, a word embedding is simply a vector space representation of words which captures some of the semantic and syntactic structures in the language—for example, words similar in meaning have representations which are close to each other in the vector space. Word embeddings have taken natural language processing (NLP) by storm in the past decade or so, and have become the backbone for numerous NLP tasks such as question answering and machine translation. There are neural approaches to learning word embeddings, but in this question we will study a simple PCA-based scheme which does a surprisingly good job at learning word embeddings.

We have created a word co-occurrence matrix $\mathbf{M}$ of the 10000 most frequent words from a Wikipedia corpus with 1.5 billion words. The co-occurrences were obtained by using a sliding window of length 5 across the Wikipedia corpus. Entry M_{ij} of the matrix denotes the number of times words i and j occur in the corpus within the same sliding window. The file `co_occur.csv` contains the symmetric co-occurence matrix. `dictionary.txt` contains the dictionary for interpreting this matrix, the ith row of the dictionary is the word corresponding to the ith row or column of $\mathbf{M}$. The dictionary is sorted according to the word frequencies. Therefore the first word in the dictionary—"the" is the most common word in the corpus and the first row or column of $\mathbf{M}$ contains the co-occurrence counts of "the" with every other word in the dictionary.

We provide some starter code in the file `hw4-pca.py` with some useful functions (e.g. to read files, generate plots, etc.) which can be used directly, and instructions on how to complete the functions required for this problem.

4.1 (6 pts) First, read the co-occurrence matrix and the list of all words from the given files. Let the matrix $\mathbf{M}$ be the $n \times d$ ($n = d = 10000$) matrix of word co-occurrences. As we discussed in class, a suitable normalization or scaling is often very helpful to get PCA to work well. In light of the power law distribution of word occurrences, in this case, we will work with the normalized matrix $\tilde{\mathbf{M}}$ such that each entry $\tilde{M}_{ij} = \log(1 + M_{ij})$. We regard the i-th row of $\mathbf{M}$ as the datapoint for the i-th word.

We will use PCA to find the first 100 principal components of the data. Let $\tilde{\mathbf{M}}_c$ be the centered version of $\tilde{\mathbf{M}}$. Use the `PCA` function from the `sklearn` library (refer `https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html`) to get $\mathbf{V}$, *i.e.* the set of first 100 principal components or eigenvectors of $\tilde{\mathbf{M}}_c$. Note that you can directly use the `fit` method with $\tilde{\mathbf{M}}_c$ as the input. Also, get the eigenvalues of the covariance matrix (check the documentation of the function) and plot all the 100 eigenvalues. Do the eigenvalues seem to decay? What percent of the variance in the data is explained by the first 100 eigenvalues we calculated (note that there are 10,000 eigenvalues in total)?

The graph shows that after the first eigenvalue, there is an enormous drop in the value of the eigenvalues. The values decrease very quickly and around 15-20, the graph starts to flatten out and the eigenvalues seem to tend to zero. 75.57% variance is explained by these eigenvalues.

Rubrics: 3 pts for the graph (2 pts for the plot and 1 pt for the correct range of values on the y-axis), 3 pts for correct answers (1.5 pts each).

4.2 (6 pts) In this question, we find embeddings for all 10,000 words in the dictionary using the principal components $\mathbf{V}$. Then, we will use word embeddings to find word(s) which are 'similar' to a given word.

To obtain the embeddings of the words, we will project the co-occurrences of the words onto the 100-dimensional space spanned by the first 100 principal components, similar to the general approach we laid down in class. Here are the two steps you should follow. Recall that we regard the i^{th} row of $\tilde{\mathbf{M}}_c$ as the datapoint for the i^{th} word. Given the 100 PCs ($\mathbf{V}$), we now project each datapoint (row of $\tilde{\mathbf{M}}_c$) onto these PCs. Denote this $n \times k$ (10000×100) matrix as $\mathbf{P}$ (you should write $\mathbf{P}$ as a matrix operation using $\tilde{\mathbf{M}}_c$ and $\mathbf{V}$). Next, to ensure that each PC gets equal importance, we normalize the vector of the projections of all the words onto the j^{th} PC (*i.e.* the j^{th} column of $\mathbf{P}$) to have unit norm, for all $j = \{1, ..., 100\}$. Denote this $n \times k$ (10000×100) matrix as $\mathbf{E}$. Finally, normalize the rows of $\mathbf{E}$ such that each row has unit ℓ_2 norm, to get a new matrix $\tilde{\mathbf{E}}$.

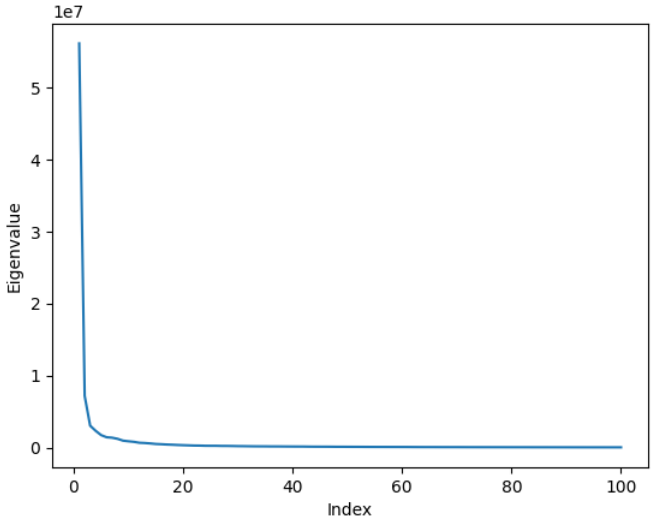

Figure 3: Plot showing the first 100 eigenvalues.

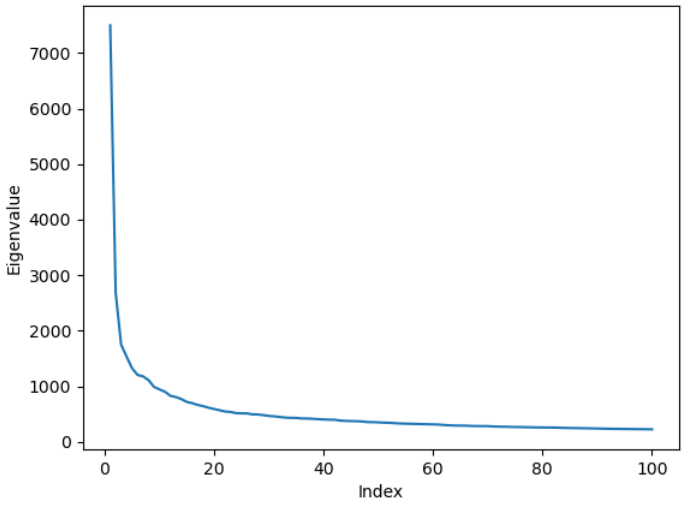

Figure 4: Plot 1 showing the first 100 eigenvalues.

We regard the ith row of $\tilde{\mathbf{E}}$ as the embedding of the ith word. Next, we will define a similarity metric for the word embeddings. We will use the cosine-similarity as the similarity metric. As all the vectors have unit ℓ_2 norm, the cosine similarity between two words i and j with embeddings $\mathbf{w}_i$ and $\mathbf{w}_j$ is equal to the inner product $\langle \mathbf{w}_i, \mathbf{w}_j \rangle$. Now that we have a similarity metric defined, we can have some fun with these embeddings by querying for the closest word to any word we like! Try finding the closest words to some common words, such as "learning", "university", "california", and comment on your observations.

The closest words to "learning", "university", "california" were "teaching", "college", "florida", respectively.

It seems like words that are related to each other have high similarity in the embedding space that we get using the PCA approximation.

Rubrics: 4 pts for getting similar words, 2 pts for observation(s).

4.3 (9 pts) We'll now interpret the principal components/eigenvectors (columns of $\mathbf{V}$). For any i, denote $\mathbf{v}_i$ as the eigenvector corresponding to the ith largest eigenvalue. Note that the entries of this vector correspond to the 10000 words in our dictionary, we'll call these our 10000 variables. By sorting the entries of $\mathbf{v}_i$ by absolute value, and observing what the top 10 variables and their (signed) entries are, we can infer what information the ith eigenvector roughly captures. Can you find 5 interesting eigenvectors, and point out what semantic or syntactic structures they capture? Can you do this for all 100 eigenvectors? Hint: What do you observe about PCs or eigenvectors with small eigenvalues?

Sample PCs and the top 10 words for each of them:
PC 1 gives: ['born' 'john' 'james' 'david' 'robert' 'william' 'george' 'jr' 'thomas' 'michael']
PC 5 gives: ['took' 'against' 'went' 'did' 'would' 'won' 'came' 'led' 'allowed' 'returned']
PC 6 gives: ['ancient' 'famous' 'greek' 'medieval' 'contains' 'contemporary' 'engine' 'aircraft' 'popular' 'known']
PC 7 gives: ['league' 'games' 'team' 'teams' 'tournament' 'championship' 'game' 'players' 'season' 'located']
PC 8 gives: ['championship' 'championships' 'cup' 'scored' 'him' 'her' 'them' 'finals' 'she' 'what']
PC 9 gives: ['german' 'army' 'aircraft' 'russian' 'soviet' 'command' 'forces' 'squadron' 'commander' 'ii']
PC 20 gives: ['television' 'vol' 'tv' 'character' 'lp' 'album' 'characters' 'actor' 'actress' 'channel']
PC 22 gives: ['australian' 'wales' 'scotland' 'zealand' 'ireland' 'british' 'chicago' 'scottish' 'australia' 'coastal']

It looks like PC 1 captures male first names, PC 5 captures action verbs, PC 6 has words related to history, the ones in PCs 7 and 8 are related to sports and competition, PC 9 captures words related to armed forces, while PC 20 captures words related to television and PC 22 captures names of places.

Not all PCs make sense because as the strongest PCs are covered, like the ones described above and vectors that describe concepts like grammar or articles, the remaining PCs might describe concepts which are not as clearly recognizable to the human eye. Additionally, as we discussed in class enforcing the PCs to be orthogonal makes it difficult to have too many interpretable components (since there likely aren't too many concepts which can be captured here which are also completely orthogonal to each other).

Rubrics: 5 pts for getting the top 10 words for 5 eigenvectors/PCs (1 each) (these can be different from the ones shown here, these are just examples), 2.5 pts for mentioning the semantic or syntactic structures (0.5 each), 1.5 pt for the answer and explanation.

4.4 (14 pts) In this question, we will explore a curious property of the word embeddings—that certain directions in the embedded space correspond to specific syntactic or semantic concepts. Let $\mathbf{w}_1$ be the word embedding for "woman" and $\mathbf{w}_2$ be the word embedding for "man". Let $\mathbf{w} = \mathbf{w}_1 - \mathbf{w}_2$, and $\hat{\mathbf{w}} = \frac{\mathbf{w}}{\|\mathbf{w}\|}$.

4.4.1 (6 pts) Project the embeddings of the following words onto $\hat{\mathbf{w}}$: *boy, girl, brother, sister, king, queen, he, she, john, mary, wall, tree*. Present a plot of projections of the embeddings of these words marked on a line. For example, if the projection of the embedding for "girl" onto $\hat{\mathbf{w}}$ is 0.1, then you should label 0.1 on the line with "girl". What do you observe?

It can be seen that words used primarily with males like "John" or "brother" are on the negative projection of $\hat{\mathbf{w}}$ and words like "queen" and "sister" have a positive projection onto $\hat{\mathbf{w}}$. This makes sense since $\mathbf{w} = \mathbf{w}_1$(woman) $-$ $\mathbf{w}_2$(man) and this positive projection imply a positive correlation with feminine gender alignment and negative means less feminine and as a result more male.

Rubrics: 4 pts for correct plot, 2 pts for correct observation. Full points if using -$\mathbf{w}$ and getting a plot flipped about the y-axis.

4.4.2 (8 pts) Present a similar plot of the projections of the embeddings of the following words onto $\hat{\mathbf{w}}$: *math, history, nurse, doctor, pilot, teacher, engineer, science, arts, literature, bob, alice.* What do you observe? Why do you

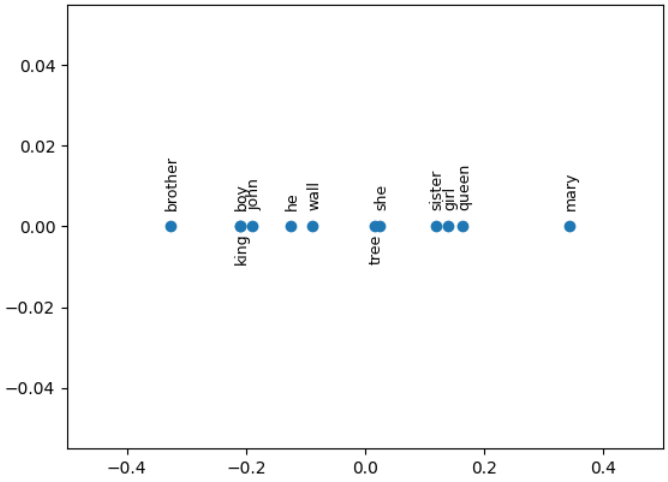

Figure 5: Plot showing projections of words in part **4.4.1**.

think this is the case? Do you see a potential problem with this? Remember that word embeddings are extensively used across NLP. Suppose LinkedIn used such word embeddings to find suitable candidates for a job or to find candidates who best match a search term or job description. What might be the result of this?

If you want to learn more about this, you might find it interesting to read the original paper which pointed out this issue in word embeddings.

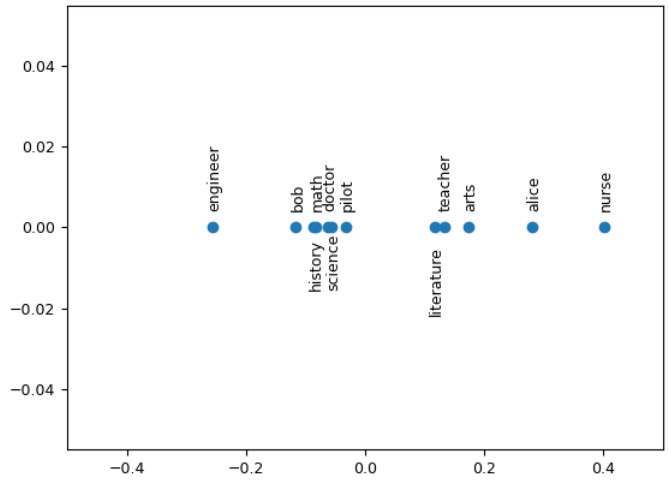

Figure 6: Plot showing projections of words in part **4.4.2**.

It can be seen that words like ”engineer” and ”bob” and ”math” have a less feminine gender alignment and more of a male one, and words like ”nurse” or ”literature” have a more feminine alignment, since it has a positive projection onto $\hat{\mathbf{v}}$.
This could be the case because there is a gender bias within occupations from the Wikipedia corpus that people associate more technical roles with males and more artistic roles with females, so in other words the language shows a

gender bias.
This is a bad idea for something like LinkedIn because depending on language that shows a gender bias in occupations will alienate people so that they are associated with their respective gendered roles. So male nurses would get marginalized since the embedding for "nurse" has more of a female orientation and female engineers would also get marginalized on search queries because "engineer" has more of a male orientation.

Rubrics: 3 pts for correct plot, 1.5 pts for correct observation, 3.5 pts for correct answers and explanation (1.5+2). Full points if using -**w** and getting a plot flipped about the y-axis.

4.5 (Bonus) (10 pts) In this question, we will explore the property that directions in the embedded space correspond to semantic or syntactic concepts in more depth .

Because word embeddings capture semantic and syntactic concepts, they can be used to solve word analogy tasks. For example, consider an analogy question— *"man is to woman as king is to ___"*, where the goal is to fill in the blank space. This can be solved by finding the word whose embedding is closest to $\mathbf{w}_{\text{woman}} - \mathbf{w}_{\text{man}} + \mathbf{w}_{\text{king}}$ in cosine similarity. You can do this by a nearest neighbor search across the entire dictionary—excluding the three words *man, woman, king* which already appear in the analogy as they cannot be valid answers to the question. Here $\mathbf{w}_i$ represents the word embedding for the word i. Refer to Fig. 7 for why this makes sense because directions in the embedded space correspond to semantic/syntactic concepts.

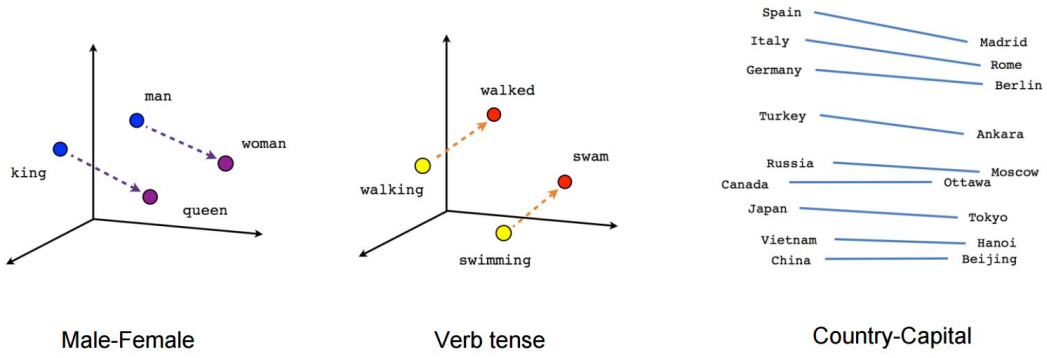

Figure 7: Figure denoting how word embeddings might encode gender, tense or country-capital relationships. Image source.

We have provided a dataset, `analogy_task.txt`, which tests the ability of word embeddings to answer analogy questions, such as those represented in Fig. 7. Using the cosine similarity metric, find and report the accuracy of the word embeddings you have constructed on the word analogy task. Look at the incorrect/correct answers of the approach and comment on the results. For example, what types of analogy questions seem to be harder to answer correctly for this approach?

The accuracy obtained is around 55.34%. The most abundant incorrect analogies were mainly verb-(some form of verb) and adjective-adverb. Analogies involving adjective-(some form of adjective) were correct more often and gender-based analogies were mostly correct.

Rubrics: 6 pts for correct implementation and accuracy, 3 pts if attempted but got low accuracy. 4 pts for (correctly) commenting on the results.

Deliverables: Discussions for all the parts. Plots for parts **4.1**, **4.4.1**, **4.4.2**. Code for all the parts as a separate Python file `hw4-pca.py`.