Fall 2023
ECE 236A
Professor: Christina Fragouli
TAs: Merve Karakas and Xinlin Li

ECE236A: Linear Programming

Interplay between Supervision and Learning in Classification
Due: Thursday, Nov 30$^{\text{th}}$, 2023, before 9pm.
To be completed by teams of 3-4 people.

Project Description:
In this project, you will explore the effect of labeled data in learning through comparison of clustering and classification. While you are obligated to include Linear Programming formulations as a core component, this project is open-ended, which means there are no "right" answers, and that we highly value innovative design ideas.
You can work in teams of 3 or 4 people. The project is due on **Thursday, Nov 30$^{\text{th}}$ before 9 pm** and needs to be uploaded on Gradescope.

1 Background

The proliferation of data has been accelerating the development of machine learning. However, the majority of successful applications rely on supervised learning, which requires labeled datasets to guide the training process and directly learn mappings from inputs to outputs. Data annotation (typically by humans) has high cost, resulting in limited availability of labeled data - while massive amounts of unlabeled data are left under-utilized. Unsupervised learning aims to address this, by making better use of the available unlabeled data. In this project, we will explore and compare supervised and unsupervised learning using two typical methodologies as follows.

Classification is a fundamental task in surpervised machine learning where the goal is to map a given set of data points to predefined categories or classes. The input data can be represented as a dataset $\mathcal{X}$ containing N samples of dimension M, $x \in \mathbb{R}^M$. Associated with each data point x is a label $y \in \mathcal{Z}$ indicating its class or category; $\mathcal{Z}$ consists of discrete values representing the classes, e.g., it can be an enumeration $\{1, 2, \ldots, K\}$ for K class classification. The objective of classification is to learn a predictive model or classifier, typically denoted as $f(\cdot) : \mathbb{R}^M \to \mathcal{Z}$, that can map the input data to its corresponding class labels, i.e., $f(x) = \hat{y}$ where $\hat{y}$ is an estimate of y. The classification model $f(\cdot)$ can be generally decomposed into $f(\cdot) = h(g(\cdot))$, where $g(\cdot)$ is a transformation function which extracts information from the input vector x, and $h(\cdot)$ is a decision function that takes output of $g(\cdot)$ and makes the final decision regarding the label estimate of y.

You will work with linear classifiers in this project. A linear classifier is a type of classifier for which the function $g(\cdot)$ is affine, i.e., $g(x) = w^T x + b$, where $w \in \mathbb{R}^M$ and $b \in \mathbb{R}$ are the model parameters to be determined.

Clustering is a technique used in unsupervised machine learning and data analysis that involves grouping similar data points together based on certain criteria or patterns. The primary goal of clustering is to identify natural groupings or structures within a dataset, without any predefined labels or categories. In other words, it aims to discover hidden patterns or associations in the data. More specifically, we assume that we have N data samples, where each data sample $x \in \mathbb{R}^M$ is a vector of M entries (we will refer to the entries as *features*). Clusters $C_1, C_2, \cdots, C_K$ are disjoint subsets or groups of data points that are more similar to each other within the same cluster than to those in other clusters. The goal is to create clusters that exhibit high intra-cluster similarity and low inter-cluster similarity which is usually determined by a distance or similarity metric $d(\cdot, \cdot)$. Common metrics include Euclidean distance (L2 distance) $d(x, y) = \sqrt{\sum_{i=1}^{M}(x_i - y_i)^2}$, Manhattan distance (L1 distance) $d(x, y) = \sum_{i=1}^{M} |x_i - y_i|$, and cosine similarity $d(x, y) = \frac{x \cdot y}{\|x\|_2 \|y\|_2}$. Many clustering algorithms additionally use a representative point $c_i \in \mathbb{R}^M$ within each cluster C_i, known as a centroid. The centroid can be calculated as the mean, median, or another measure of the data points in the cluster.
In Fig. 1, you can find an example of a clustering of a dataset with two features. The dataset is partitioned into 3 clusters using K-means clustering algorithm, and the red points represent the centroid of each cluster where centroids are calculated by taking the mean of data points in the cluster.
Example:

- Streaming services: Businesses use clustering to group customers with similar behavior, preferences, or purchase histories. For instance, they collect data such as the number of shows viewed, minutes watched per day over the last month, the amount spent on purchases, and categories of shows watched; and perform cluster analysis to identify high usage and low usage users. This helps in targeted marketing, advertising, and product recommendations. If streaming services had access to labels (low/high usage users) for the users, they could train a simple classifier on the data, and use it to classify any upcoming users in the system. However, data does not always come with labels and even if it does, it is labeled by humans. It is both expensive and time-consuming to label large amount of data, and human factor introduces possibility of errors (wrong labeling). Moreover, ability of your algorithm to learn becomes constrained by the capabilities of humans. Both due to time, resource, and reliability constraints; unsupervised learning has gained popularity and sometimes it is the only option for some tasks.

Assessing Performance.
• *Classification* A good classifier is a classifier that is able to correctly classify as many input vectors as possible. One common way to measure the quality of a classifier is through the percentage of

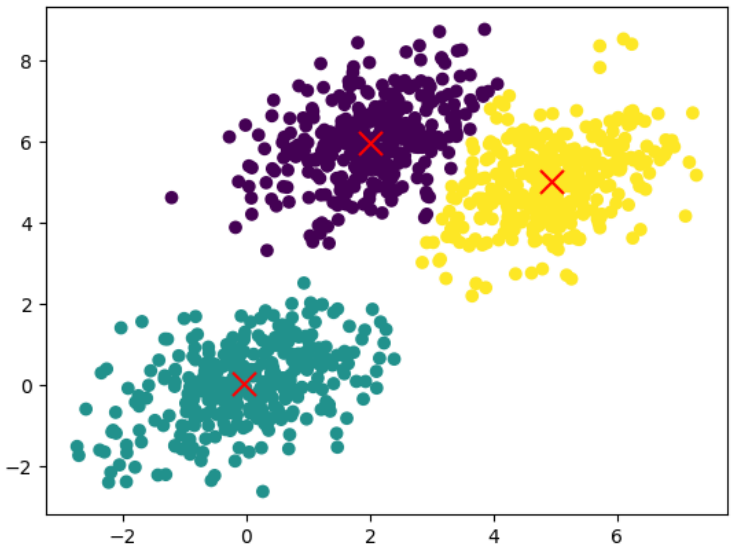

Figure 1: Two-dimensional example for clustering into 3 groups. Each color represents a cluster and red crosses are cluster centroids calculated using mean of the group.

correctly classified points, namely classification accuracy. Specifically, let $x^{(i)} \in \mathbb{R}^M$ denote the samples in the dataset $\mathcal{X}$ with N samples and $y^{(i)} \in \mathcal{Y}$ are corresponding labels. We can define classification accuracy as

$$P(\mathcal{X}, \mathcal{Y}) = \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}(y^{(i)} = f(x^{(i)})$$

where $\mathbb{1}(\cdot)$ is the indicator function.

• *Clustering* There are several methods and metrics to evaluate clustering performance. The choice of the evaluation metric should be based on the goals of the clustering task and the data characteristics. One determining factor in assessment choice is whether you have ground truth labels or not. Commonly used metrics for supervised evaluation (with ground truth) are Adjusted Rand Index (ARI), Normalized Mutual Information (NMI), confusion matrix, and visual inspection; and silhouette score, Davies-Bouldin Index, and variance ratio criterion for unsupervised evaluation.
In this project, you will use Normalized Mutual Information (NMI) to measure the agreement between the two label assignments (the implementation of NMI will be provided). Given ground truth labels $\mathcal{Y}_{\text{true}}$ and clustering labels $\hat{\mathcal{Y}}$ corresponding to a set of data points, NMI is defined as

$$NMI(\mathcal{Y}_{\text{true}}, \hat{\mathcal{Y}}) = \frac{I(\mathcal{Y}_{\text{true}}; \hat{\mathcal{Y}})}{mean(H(\mathcal{Y}_{\text{true}}), H(\hat{\mathcal{Y}}))}$$

where $I(\cdot;\cdot)$ and $H(\cdot)$ stand for mutual information and entropy respectively. The value of NMI is between 0 and 1 (i.e. perfect correlation), and it is independent of label permutation. That is, the closer NMI is to 1, the better.

Training a Model

• *Classification* The classifier is usually trained in some way on a labeled training data set $(\mathcal{X}_{\text{train}}, \mathcal{Y}_{\text{train}})$ to determine the best parameters such that these training data are correctly classified. More specifically, the model parameters w, b, and decision function $h(\cdot)$ need to be optimized to maximize the

training accuracy $P(\mathcal{X}_{\text{train}}, \mathcal{Y}_{\text{train}})$. Once trained, it can be used to predict the class labels of new, unlabeled data points. The premise is that if the training set is a good representative of each class, the classifier can generalize well on unseen data and achieve similar accuracy.

• *Clustering* The training process of clustering is to assign each data point in the given unlabeled dataset to an appropriate cluster. The first step is usually to decide on a distance/similarity metric and the total number of clusters, based on your dataset characteristics and objectives. These can be turned as other hyperparameters. Then the goal of a clustering algorithm is to group the given data and identify clusters according to the chosen metric, which usually involves iterated updates. Once trained, each data point will be assigned with a cluster label and new data points can also be assigned to identified clusters. The results can be evaluated using one of the assessing metrics mentioned above.

To give a concrete example, K-means clustering (to also be discussed in class) is one of the popular clustering algorithms: The algorithm consists of initialization, assignment, update, and repetition steps. It starts by selecting K initial cluster centroids, where K is the number of clusters you want to identify (it is a hyperparameter and can be tuned). These initial centroids can be randomly chosen from the data points or by other methods. Each data point is assigned to the nearest centroid based on a similarity or distance metric, typically using Euclidean distance. The centroids of the clusters are then recalculated as the mean (average) of all data points assigned to each cluster. Assignment and update steps are iteratively repeated until a convergence criterion is met. Common convergence criteria include a maximum number of iterations or a small change in centroids between iterations.

2 Project Goal and Details

In this project, we consider a multi-class classification task supervised learning (classification of labeled data) and unsupervised learning (clustering based on number of classes) techniques.

We divide the project into four parts: in Task 1, you assume complete knowledge of the dataset labels $\mathcal{Y}_{\text{train}}$ to do linear classification. In Task 2, you assume NO knowledge of the labels and perform unsupervised clustering. In Task 3, you are given a certain budget to label some of the training samples, and you are asked to develop an algorithm to choose which samples to label. Finally, Task 4 contains open-ended questions where you discuss your findings from previous tasks.

We also provide two datasets for evaluation; however, the algorithms you find for each task should be applicable to any number of classes and to any dataset that can be used for classification tasks. You should perform the given tasks for both datasets and provide plots for each of them in your report as described next.

Task 1: Supervised Classification

Task 1.1: Formulation For this part, you are allowed to use the whole training dataset and its labels $(\mathcal{X}_{\text{train}}, \mathcal{Y}_{\text{train}})$ for training. Solve (one or more) linear programs, and propose, using the optimization program outputs, a method to perform multi-class classification for any number of classes $K \geqslant 2$. Write down the formulation in your project report. Clearly define your variables,

constraints, and objective function. Elaborate on how you would use the linear program(s) to create a classifier.

Task 1.2: Implementation and Evaluation Implement the classifier you came up with in Task 1.1. Train your model on the synthetic dataset and MNIST dataset. Evaluate the classification accuracy on the testing set $(\mathcal{X}_{\text{test}}, \mathcal{Y}_{\text{test}})$ of your algorithm and include the results in your report. In addition, visualize (plot) the decision boundary of the classifier, along with the test data point of the synthetic dataset in a 2D plot.

Task 2: Unsupervised Clustering

Task 2.1: Formulation For this task, you assume unlabeled data, i.e., you are only allowed to use $\mathcal{X}_{\text{train}}$ for training. Formulate (integer and associated linear) program(s) to do clustering for any given dataset. Your program should be generic to create any given number of clusters K. Write down the formulation in your project report. Clearly define your variables, constraints, and objective function. If you first formulate an ILP and relax it to LP, explain your reasoning. Explain which distance metric you chose to decide which points are in the same cluster and how you enforced it in the linear program. Elaborate on how you would use the linear program to create an algorithm for clustering.

Task 2.2: Implementation and Evaluation Implement the algorithm you came up with in Task 2.1. Train your model on the synthetic dataset using $K = 3, 5, 10$ and on MNIST data using $K = 3, 10, 32$. Evaluate both the clustering and classification performance of your algorithm. For the clustering performance, we will provide the function to calculate normalized mutual information (NMI) on the training set $(\mathcal{X}_{\text{train}}, \mathcal{Y}_{\text{train}})$; and for the classification performance, you need to first assign the testing data $\mathcal{X}_{\text{test}}$ to identified clusters (without training from scratch) and calculate the accuracy using $\mathcal{Y}_{\text{test}}$ (here we assume each cluster can be assigned with a class label according to the majority true labels of training data in that cluster, a function will be provided in the given code). Compare the clustering performance among different choices of K and compare the classification results with what you obtained in Task 1 in your report. Include two bar charts to visualize the results using the given *plot_result()* function in *utils.py* file.

Task 3: Semi-supervised learning (label Selection)

Task 3.1: Formulation For this task, assume you have a budget to obtain correct labels for L samples in a given dataset. Your task is to write down an (integer or linear) program that decides which samples to choose for labeling. Your program should work for any number of samples and cannot use the labels as an input to the program – you will obtain the labels for selected samples based on your program outputs. In a way, you are trying to label the most informative L samples that will allow you to train a classifier that achieves good performance. Write down the formulation in your project report. Clearly define your variables, constraints, and objective function. If you first formulate an ILP and relax it to LP, explain your reasoning.

Task 3.2: Implementation and Evaluation Implement the program you developed in Task 3.1. Run it to obtain $5\%, 10\%, 20\%, 50\%$ labeled data for each dataset. Train the classifier you have from Task 1 on the labeled data. Compare the performance of classifiers both with each other and with the case where you uniformly at random select which points to label in your report. Plot the percentage of used labels vs testing accuracy using the given *plot_ result()* function in *utils.py* file.

Task 4. Discussion and Comparison We would like you to reflect on your findings, compare the findings in the three tasks, and potentially run additional simulations to answer the following questions:

1. Is it correct that if we use unsupervised learning, we simply need a larger number of samples to achieve the same performance as supervised learning? Check how the performance of your classifier changes, for all three tasks, as you vary the number of training samples.

2. How does the performance of clustering in particular change with the number of samples? What happens if we have too few samples?

3. What if the clustering did not give "hard decisions" but rather "soft decisions" (belongs in class * with prob *) - could we use this information in the design of the classifier? Could we then achieve a better performance with a smaller number of samples?

4. In Task 2, to use the learned clusters for classification, we use the majority vote to connect each cluster with a class label. However, this still requires all labels $\mathcal{Y}_{\text{train}}$ of the training data and is not truly unsupervised. Could you propose a better way that assign class labels to clusters using a smaller number of true labels while maintaining good performance?

3 Dataset

You will need to validate your algorithms and implementations on two datasets:

1. Synthetic Dataset: We generated $N = 1000$ samples with dimension $M = 2$. Samples are generated from one of the three multivariate Gaussian distribution. Labels $y^{(i)} \in \{0, 1, 2\}$ indicate which distribution $x^{(i)}$ belongs to.

2. MNIST Dataset[1]: We will use a reduced version of MNIST Dataset, which contains N=1000 data samples from three classes. Each sample is a 28 by 28 image (M=784) of a handwritten digit. Labels $y^{(i)} \in \{2, 5, 6\}$ indicate which number is written in the image.

Note: This part is to inform you about which datasets you will experiment on. You do not need to code anything for getting the datasets, you will just call the corresponding functions to get the datasets.

[1] http://yann.lecun.com/exdb/mnist/

4 Implementation Requirements

Please download the project_code.zip file from Bruin Learn, which contains three files:

- We provide some utility functions in **utils.py**. You do not need to do any data cleaning, preprocessing, or splitting. You can get the processed datasets using `prepare_data_synthetic()` and `prepare_data_mnist()`. You can get the requested plots for Task 2 and 3 using `plot_result()` function, for which you need to store the required experiment output into dictionaries and pass it to the function. You can find the input/dictionary format in the comment lines within the function. Do NOT change the given code in this file.

- The synthetic data and reduced MNIST data are stored in **Data.zip**. Unzip it before using `utils.prepar_data_synthetic()` and `utils.prepar_data_mnist()` functions.

- **MySolution.py** contains the skeleton implementation of Task 1,2, and 3. You are supposed to complete the missing parts according to the task instructions and submit it as **MySolution_{groupnumber}.py**. Each task is implemented as a class, and there are three classes (named `MyClassifier`, `MyClustering`, and `MyLabelSelector`) in total.

Notes:

1. You are required to complete the implementation of this project using Python. A short introduction to Python is provided in Discussion 1. You are allowed to use any libraries.

2. Do **NOT** change the given functions or code structure. You are allowed to add other auxiliary attributes/methods/functions, but each task should be completed within the specified methods. You may also consider adjusting the input (parameters) of methods. When you do so, please make sure it is generic and specify the meaning in the comments, so that we are able to reproduce the results or run verification on other datasets using your code.

3. Do **NOT** use the testing dataset $\mathcal{X}_{\text{test}}, \mathcal{Y}_{\text{test}}$ for training classifier or clustering; do **NOT** use labels $\mathcal{Y}_{\text{train}}$ for clustering. They are only used they by the given functions (i.e., *evaluate()*, *evaluate_clustering()*, and *evaluate_classification()*) to check the performance of your algorithms.

5 Report

Beyond the code, we also expect a report of up to 3 pages (excluding appendix and references), that describes in a complete way what is the rationale you used to to formulate the LPs/ILPs as well as the specific algorithm descriptions. Be concise and clear about your algorithm definitions and rationale behind. Put the required figures and discuss your observations. You are welcome to add extra plots in the appendix if they help with your illustration.

6 Grading

- 10 points: This project will be graded mainly based on LP formulations, and completeness of results. We will grade based on if your formulation is actually a linear program, how you justified the algorithms you came up with, and your ability to get results by implementing those algorithms.
- 5 bonus points: Bonus points up to 5 points will be given to at most 5 groups based on the creativity and performance of your proposed algorithms, subject to also presenting their approach on Dec. 7^{th}.
- You need to submit a folder named `'group_{groupnumber}'` (`group_5` for the group with name group 5) on Gradescope. Inside this folder there should be a file named **MySolution_{groupnumber}.py** as well as your report and experiment files (where you run the simulations).