Final Project: MNIST classification
ECE 5450: Machine Learning
Fall 2023

- The main focus of the final project is to apply the machine-learning tools we learned in the course to practical applications.
- You will be mainly working with the Modified National Institute of Standards and Technology database (MNIST) dataset, which is a large database of handwritten digits that is commonly used for training various image processing systems.

- The project consists of two components. These components are
  - **Component 1:** Develop Gaussian and Logistic Regression classifiers to classify digit 0 from 1. You should use your own implementations of the above classifiers. While you may use scikit learn to double check your answers, we will be grading your python implementations.
  - **Component 2:** Extend the above Gaussian and Logistic Regression classifiers to multiclass classification of all digits.

INSTRUCTIONS

- This is an individual project. You may discuss the project with your peers, but please do not share code or work on the code together.
- We can help with the concepts, but we will not be able to debug your code and make it work. Please refer to the mini-assignments and documentation available online to develop the code.

Component 1:

Develop Gaussian and Logistic Regression classifiers to classify digit 0 from 1. Please note that both these classifiers use the discriminant

$$y(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0 = \underbrace{\begin{bmatrix}\mathbf{w}^T & w_0\end{bmatrix}}_{\mathbf{a}^T}\underbrace{\begin{bmatrix}\mathbf{x}\\1\end{bmatrix}}_{\tilde{\mathbf{x}}} \tag{1}$$

Please try to write your code as modules (see details below) to make the code easy to understand and test.

1. Load the MNIST dataset and extract a subset with zeros and ones. Split this subset into training (70%), validation (10%), and test dataset (20%).

2. Write a function `testLinearClassifier` to test the performance of a classifier specified by its weights $\mathbf{a}$. This function accepts the test dataset, labels, and $\mathbf{a}$ and would return the number of misclassified points.

3. Adapt the Gaussian classifier, assuming the common covariance matrix $\mathbf{\Sigma} = \mathbf{I}$ you developed in previous mini-assignments and homeworks to the classification of this dataset. Estimate the parameters $\boldsymbol{\mu_1}$ and $\boldsymbol{\mu_2}$ from the training dataset. Develop a python function called `gaussianClassifierWithIdentityCovariance` that implements the above classifier. This function accepts the training dataset and the labels, and returns the vector $\mathbf{a}_1$ that corresponds to the classifier with discriminant $y_1(x)$.

4. Adapt the Gaussian classifier, assuming a common covariance matrix $\mathbf{\Sigma}$ you developed in previous mini-assignments and homeworks to the classification of this dataset. Estimate the parameters $\boldsymbol{\mu_1}$, $\boldsymbol{\mu_2}$, and $\mathbf{\Sigma}$. Develop a python function called `gaussianClassifierWithCommonCovariance` that implements the above classifier. This function accepts the training dataset and the labels, and returns the vector $\mathbf{a}_2$ that corresponds to the classifier with discriminant $y_2(x)$.

5. Adapt the two class Logistic regression algorithm to this dataset. Train the algorithm on the training dataset. Develop a python function called `logisticRegression` that implements the above classifier. This function accepts the training dataset and the labels, and returns the vector $\mathbf{a}_3$ that corresponds to the classifier with discriminant $y_3(x)$.

6. Implement a regularized version of logistic regression. Determine the gradient of the cost function in slide 20 of logistic regression and develop the corresponding iterative algorithm. Develop a python function called `logisticRegressionWithRegularization` that implements the above classifier. This function accepts the training dataset, labels, and λ and returns the vector $\mathbf{a}$. You may test this function independently with a specific λ value.

7. Write a function `optimizeHyperparameters` to determine the optimal λ of `logisticRegressionWithRegularization`. This function would call `logisticRegressionWithRegularization` with multiple λ values and determine the optimal λ using the validation dataset. For each solution (corresponding to a specific λ value), determine the # misclassifications on the validation subset. Choose the λ value that gives the lowest classification error on the validation dataset. Determine the accuracy of the classification algorithm corresponding to the above optimal λ on the test subset. This function accepts the training dataset, validation dataset, and the labels, and returns the vector $\mathbf{a}_4$ that corresponds to the optimal logistic regression classifier with discriminant $y_4(x)$.

8. Write a script that would compare the performance all the above algorithms on the test subset. This script calls the above functions to train the classifiers. It then uses `testLinearClassifier` with each of the vectors $\mathbf{a}_1$, $\mathbf{a}_2$, $\mathbf{a}_3$ and $\mathbf{a}_4$ and report the misclassifications in each case.

Component 2:

Develop Gaussian and Logistic Regression classifiers for multi-class classification. Please note that these classifiers use n discriminants, where n is the number of classes

$$y_i(\mathbf{x}) = \mathbf{w}_i^T\mathbf{x} + w_0 = \underbrace{\begin{bmatrix}\mathbf{w}^T & w_0\end{bmatrix}}_{\mathbf{a}_i^T}\underbrace{\begin{bmatrix}\mathbf{x}\\1\end{bmatrix}}_{\tilde{\mathbf{x}}}, \quad i = 0, .., n-1 \tag{2}$$

You should use your own implementations of the above classifiers. While you may use scikit learn to double check your answers, we will be grading your implementations. Please try to write your code as modules to make the code easy to understand and test

1. Load the MNIST dataset. Split this dataset into training (70%), validation (10%), and test dataset (20%).

2. Consider the multi-class Gaussian classifier in page 23 of Logistic regression slide deck. Assume a common covariance matrix $\mathbf{\Sigma} = \mathbf{I}$. Estimate the parameters $\boldsymbol{\mu_i}; i = 0, .., n-1$ from the training dataset. Develop a python function called `gaussianMultiChannelClassifier` that implements the above classifier. This function accepts the training dataset and the labels, and returns the matrix $\mathbf{A}_1$, whose columns are $\mathbf{a}_i$ that corresponds to discriminants $y_i(x)$.

3. Implement a multiclass logistic regression algorithm to this dataset. Develop a python function called `logisticRegressionMultiClassClassifier` that implements the above classifier. This function accepts the training dataset and the labels, and returns the matrix $\mathbf{A}_2$.

4. Implement a regularized version of multi-class logistic regression. Develop a python function called `logisticRegressionMultiClassClassifierWithRegularization` that implements the above classifier. This function accept the training dataset, labels, and λ and return the matrix $\mathbf{A}_3$. You may test this function independently with a specific λ value.

5. Write a function `Optimize_MC_Hyperparameters` to determine the optimal λ of `logisticRegressionMultiClassClassifierWithRegularization`. This function accepts the training dataset, validation dataset, and the labels, and returns the matrix $\mathbf{A}_4$ that corresponds to the optimal logistic regression classifier.

6. Write a function `testLinearMCClassifier` to test the performance of a multiclass classifier specified by its weights $\mathbf{A}$. This function would accept the test dataset, labels, and $\mathbf{A}$ and would return the number of misclassified points.

7. Write a script that would compare the performance all the above algorithms on the test subset. This script would call `testLinearMCClassifier` with each $\mathbf{A}_1$, $\mathbf{A}_2$, $\mathbf{A}_3$, $\mathbf{A}_4$ and report the misclassifications in each case.