Modern Machine Learning
Computer Assignment #3

1. *Multiclass classification and regularization:* Using Python or MATLAB, implement the *one-vs-all* technique for multiclass classification using Support Vector Machines.

   Background: The one-vs-all approach consists of trainining K separate binary classifiers, where K is the number of classes, for each one of the different classes.

   - Train K separate binary classifiers producing a set of classification rules $G_j(\mathbf{x})$ $(j = 1, 2, \ldots, K)$
   - For every new example $\mathbf{x}$, find the predicted label as

   $$\hat{j} = \arg\max_j G_j(\mathbf{x})$$

   In this exercise, we still use the partial MINST database in Model 2. It contains handwritten digits 0-9. The size of the images is 28 × 28 pixels, these pixels are vectorized producing features of size 784 × 1. Your program will attempt to recognize these digits. Examples of the handwritten digits can be seen in Figure 1.

   Python files: The script multiclassSVM.py can be used as a template to implement your one-vs-all classifier. You can reuse useful coding information from the worked example (svm_example.py) in Model 3 and the one-vs-all classification in Model 2. However, you should finish the main function in multiclassSVM.py by your own.

   You are required to install the cvxopt package into your environment first, the command is

   ```
   conda install -c conda-forge cvxopt
   ```

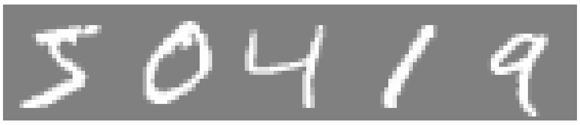

Figure 1: Example images from MINST database

You can find more information about the cvxopt package at the link `https://cvxopt.org/install/index.html`. In addition, you also need the **scipy** package as Model 2.

Submission guidelines: Your submission should include:

- A unique **zip folder**, which should include a modified version of multiclassSVM.py, plus all necessary files to run your code.
- You should complete the main function by yourself. In the main function, you need to finish three tasks: (1) use the one-vs-all method to train SVM models to get the corresponding weights, (2) use the one-vs-all method to test the learned SVM modesl to check the learned models, and (3) obtain the training accuracy and testing accuracy during the above two tasks.
- You can reuse the code in the svm_example.py file to solve SVM Wolfe dual problem for learning weights. You can reuse the code from Model 2 to solve the one-vs-all problem. MNIST dataset is non-sperable case, you can refer 3-2 slides Page 7-8 to complete the classification task. You can refer 3-1 slides Page 5-8 to design classifcation accuracy function.
- Please rename the modified file multiclassSVM.py by adding your last name. **This should be the main function**.
- A pdf file with a figure showing the classification accuracy vs the variable C in the function 'svm_dual'. Choosing three different values, e.g., $C = \{0.00005, 0.01, 10\}$ to examine the training accuracy. Explain your results.

Hint: The training accuracy should be around 80% and the testing accuracy should be around 60%..

MATLAB files: The script multiclass_svm_example.m can be used as a template to implement your one-vs-all classifier. The function multiclassSVM.m should be finished by the student to complete the assignment.

Submission guidelines: Your submission should include:

- A unique **zip folder**, which should include a modified version of multiclass_svm_example.m and multiclassSVM.m, plus all necessary files to run your code. The function multiclassSVM.m has the following structure $\hat{\mathbf{v}} = \text{multiclassSVM}(\mathbf{X}, \mathbf{y}, \mathbf{U}, m)$, where $\mathbf{X}$ are the training examples, $\mathbf{y}$ is a vector containing their corresponding class labels, $\mathbf{U}$ is a matrix containig the testing examples, and m is the number of class labels. The output $\hat{\mathbf{v}}$ is a vector with the predicted labels of the testing examples in $\mathbf{U}$. This function should train K binary SVM classifiers using the

MATLAB function $fitcsvm(\cdot)$ (using the **linear kernel**, which is the default option). You can use a *for* loop to achieve this just as in Computer Assignmet #2.

Please rename the modified file multiclass_svm_example.m replacing the word 'example' in the provided script with your last name. **This should be the main function**.

- A pdf file with a figure showing the classification accuracy vs number of training examples. The variable *train_num* in multiclass_SVM_example.m allows you to vary the number of training examples. Vary *train_num* from 2500 to 4500 examples in steps of 1000. Explain your results.