E0-270: Assignment 1

Due date: March 29, 2023

Task

Principal Component Analysis and Support Vector Machines on MNIST Dataset

Dataset

MNIST dataset (already provided in the assignment folder). The dataset consists of $60,000$ training images and $10,000$ test images. Each image is a 28×28 grayscale image of a handwritten digit, and the task is to classify what digit it is. So, it is a 10-class classification problem.

Each pixel of the image is represented by a single integer between 0 and 255, and so each image is a 784-dimensional vector.

The dataset is already split into training and test sets. Each image is a row in the given csv files. The first column is the label and the remaining 784 columns are the flattened image.

Task 1: Principal Component Analysis

1. Fill in the normalize function in 'utils.py' to normalize the pixels of the images between -1 and 1.
2. Implement PCA to reduce the dimensionality of the images from 784 to a lower number of principal components ($k = 5, 10, 20, 50, 100, 200, 500$). For this, fill in the class methods in 'model.py' for the class 'PCA'.

Task 2: Multi-Class SVM

Since there are 10 classes in MNIST, you have to perform 1-vs-rest classification for each class separately (thereby training 10 SVM models, one for each class). For multi-class prediction, choose the class corresponding to the model that has the highest value of $w^T x + b$.

1. Implement soft SVM from scratch (linear kernel) using stochiastic gradient descent (use the subgradients when the gradient is undefined). Essentially, fill in the class methods in 'model.py' for the class 'SupportVectorModel'.
2. Now, use the above trained models to create a multi-class classifier. Fill in the class methods in 'model.py' for the class 'MultiClassSVM'.
3. Evaluate the performance of the model on the test set and report the accuracy, precision, recall, and F1 score.
4. Plot the performance metrics (accuracy, precision, recall, and F1 score) with respect to the number of principal components used. Fill in the codes for plotting in 'utils.py'.

Deliverables

- Code for implementing PCA and SVM without any libraries other than the aforementioned.
- A report detailing the methodology, results, and analysis of the experiments.
- A plot of the performance metrics (accuracy, precision, recall, and F1 score) with respect to the number of principal components used.

Submission: Email a **single zip file** of the format - `Asst1_FirstName_Last5DigitsOfSRNo.zip` to `e0.270.iisc.2023@gmail.com` with Subject: `Asst1_FirstName_Last5DigitsOfSRNo` on or before the due date. The zip file should contain the following four files: main.py, utils.py, model.py and report.pdf.

Details on SVM training

The soft SVM primal problem is given by

$$\begin{aligned} \min_{w,b,\xi} f(w,b,\xi) &= \frac{||w||^2}{2} + c\sum_{n=1}^{N} \xi_n \\ \text{s.t. } y_n \left(w^T x_n + b\right) &\geq 1 - \xi_n \\ \xi_n &\geq 0 \end{aligned}$$

So, substituting ξ_n in the objective function gives the new unconstrained objective:

$$\min_{w,b} f(w,b) = \frac{||w||^2}{2} + c\sum_{n=1}^{N} \max\left\{0, 1 - y_n \left(w^T x_n + b\right)\right\}.$$

Since this is an unconstrained problem, it can be solved directly using Stochastic Gradient Descent (SGD).

Since the second term is discontinuous, the gradient might not be defined everywhere. When the gradient is not defined, you can take it to be zero (since zero is a sub-gradient in this case).