ASSIGNMENT-3

"Classification Accuracy on MNIST handwritten digit data by Perceptron and Linear SVM with and without feature extraction (using mutual information)"

Submitted By

Sourabh Kumar

Stud ID: 15/8

Course Name- CPI 015 Pattern Recognition- Spring 2020

Under the guidance of:

Prof. MN Murthy

Department of Computer Science and Automation

Indian Institute of Science, Bangalore, India

Problem Statement

1. Download MNIST handwritten digit data. There are 10 classes (corresponding to digits 0, 1, ..., 9) and each digit is viewed as an image of size 28×28 (= 784) pixels; each pixel having values 0 to 255. There are around 6000 digit training patterns and around 1000 test patterns in each class and the class label is also provided for each of the digits. Visit http://yann.lecun.com/exdb/mnist/ for more details.
2. Run Perceptron and Linear SVM based on the following:

(a) Consider classes 0 (digit zero) and 1 (digit one). Compute the accuracy of Perceptron and Linear SVM classifiers on the data. Use both the Hard and Soft Margin SVMs.

(b) Repeat the experiment in step 1 with the pair of classes 7 and 9.

(c) Use mutual information to extract the best 80 features (out of 784 (28×28)) in each of the above cases and compute accuracy on the test dataset using Perceptron and Linear SVM classifiers.

3. Report your results appropriately using tables and graphs for different scenarios.
4. The report must be brief giving a page on the resources used and how they are used. Two-three pages on the results of your experiments.

Technology and Programming Resources Used

- Spyder Programming Editor
- Python Programming Language 3.7
- Following popular sklearn python libraries for machine learning
    a. sklearn.datasets for fetching MNIST data (fetches data internally from the source web site- http://yann.lecun.com/exdb/mnist/)
    b. sklearn.linear_model for perceptron classifier
    c. Sklearn for SVM
    d. sklearn.preprocessing for binarizing the data based on below logic
        1. range [0,127] – Binary value 0
        2. range [128,255] – Binary value 1
    e. sklearn.feature_selection for extracting best 80 features using mutual information method
    f. matplotlib.pyplot library for plotting charts
- MNIST hand written digit data with
    a. Total Features -784 (pixel grid size- 28x28)
    b. Total Classes- 10 (Digit 0 to Digit 9)
    c. Total Training data- 60000 (6000 per class)
    d. Total Test data- 10000 (1000 per class)
    e. Two pairs of Class data used for experiments- “Class 0 & Class 1” and “Class 7 & Class 9” together

Dataset Pre-Processing: -

Data set – Class 0 & Class 1 data

Training data - First 12,000 records (starting from 0 to 11,999 row indices) and all 784 feature fields

Training target- First 12,000 records (starting from 0 to 11,999 row indices) and last785th target field.

Test data – First 2,000 records (starting from 60,000 to 61,999 row indices) and all 784 feature fields

Test target - First 2,000 records (starting from 60,000 to 61,999 row indices) and last 785th target field

Data set – Class 7 & Class 9 data

Training data - 6,000 records (starting from 42,000 to 47,999 row indices) and

6,000 records (starting from 54,000 to 59,999 row indices) for all 784 feature fields

Training target- 6,000 records (starting from 42,000 to 47,999 row indices) and 6,000 records (starting from 54,000 to 59,999 row indices) for last785th target field.

Test data – 1,000 records (starting from 67,000 to 67,999 row indices) and

1,000 records (starting from 69,000 to 69,999 row indices) and all 784 feature fields

Test target - 1,000 records (starting from 67,000 to 67,999 row indices) and 1,000 records (starting from 69,000 to 69,999 row indices) for last 785th target field

Mutual Information process for selecting 80 or more features: -

```
# Calculating multual information and returns the value in an signle dimentsional array
mi_features_0_1 = mutual_info_classif(mnist_training_data_0_1, mnist_training_target_0_1,random_state=0)
mi_features_7_9 = mutual_info_classif(mnist_training_data_7_9, mnist_training_target_7_9,random_state=0)

top_80_features_based_0_1 = mi_features_0_1.argsort()[::-1][:80]
top_80_features_based_7_9 = mi_features_7_9.argsort()[::-1][:80]

total_indices = range(len(top_80_features_based_0_1))

# Making blank training data set and test data set for class pair- (0 & 1)
new_mnist_training_data_0_1 = np.array(np.empty((12000,))).reshape(-1,1)
new_mnist_test_data_0_1 = np.array(np.empty((2000,))).reshape(-1,1)

# Making blank training data set and test data set for class pair- (7 & 9)
new_mnist_training_data_7_9 = np.array(np.empty((12000,))).reshape(-1,1)
new_mnist_test_data_7_9 = np.array(np.empty((2000,))).reshape(-1,1)
```

Experiment-1 (a)

Check Classification Accuracy for first pair (Class 0 & Class 1) respectively using perceptron and SVM (soft and hard) with all 784 features used

Program Name- Assignment3_without MI.py

Data set – Class 0 & Class 1 data

Result Table:

Dataset	Algorithms	Accuracy %
Class 0 & 1	Perceptron	90.0
Class 0 & 1	Hard Margin SVM	95.0
Class 0 & 1	Soft Margin SVM	96.0

Experiment-1(b)

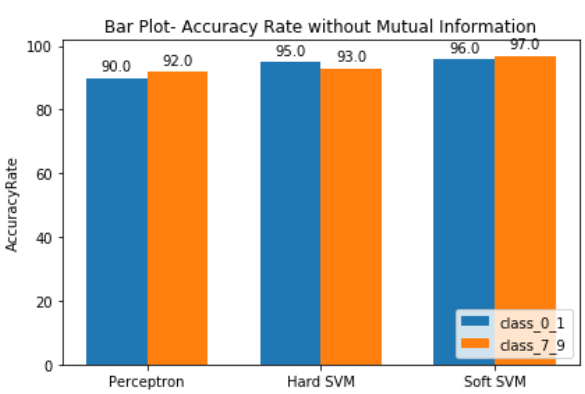
Check Classification Accuracy for second pair (Class 7 & Class 9) respectively using perceptron, and SVM (soft and hard) with all 784 features used

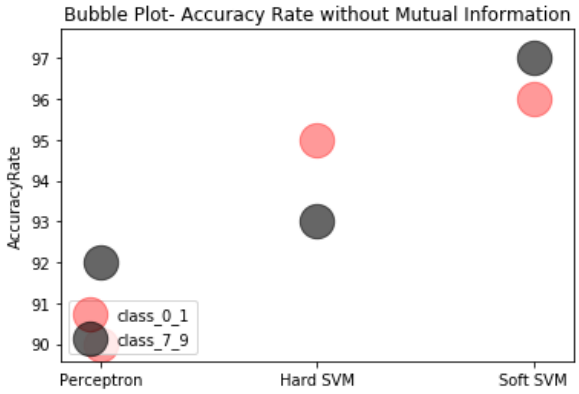
Program Name- Assignment3_without MI.py

Data set – Class 7 & Class 9 data

Dataset	Algorithms	Accuracy %
Class 7 & 9	Perceptron	92.0
Class 7 & 9	Hard Margin SVM	93.0
Class 7 & 9	Soft Margin SVM	97.0

Plots:





Experiment-2(a)

Check Classification Accuracy for first pair Class 0 & Class 1 using perceptron and SVM (soft and hard) with all 80 features used (after mutual information feature extraction pre-processing)

Program Name- Assignment3_with MI.py

Class 0 & Class 1 data

Result Table:

Dataset	Algorithms	Accuracy %
Class 0 & 1	Perceptron	66.0
Class 0 & 1	Hard Margin SVM	65.0
Class 0 & 1	Soft Margin SVM	80.0

Experiment-2(b)

Check Classification Accuracy for first pair Class 7 & Class 9 using perceptron and SVM (soft and hard) with all 80 features used (after mutual information feature extraction pre-processing)

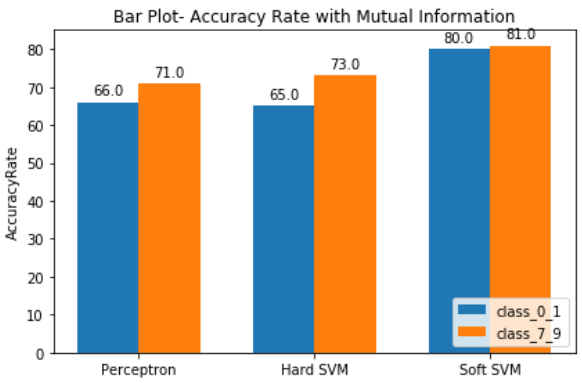
Program Name- Assignment3_with MI.py

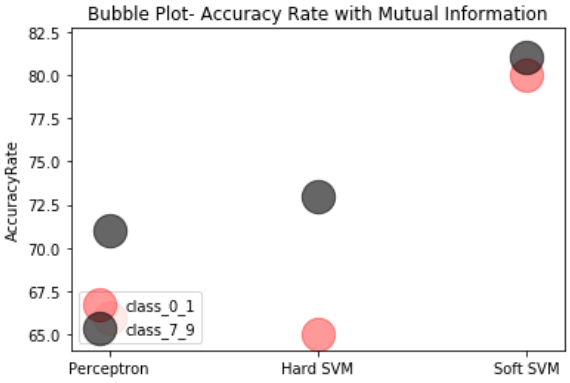
Data set – Class 7 & Class 9 data

Result Table:

Dataset	Algorithms	Accuracy %
Class 7 & 9	Perceptron	71.0
Class 7 & 9	Hard Margin SVM	73.0
Class 7 & 9	Soft Margin SVM	81.0

Plots:





Experiment-3(a)

Check Classification Accuracy for first pair Class 0 & Class 1 using perceptron and SVM (soft and hard) with all 160 features used (after mutual information feature extraction pre-processing)

Program Name- Assignment3_with MI.py

Class 0 & Class 1 data

Result Table:

Dataset	Algorithms	Accuracy %
Class 0 & 1	Perceptron	84.0
Class 0 & 1	Hard Margin SVM	81.0
Class 0 & 1	Soft Margin SVM	88.0

Experiment-3(b)

Check Classification Accuracy for first pair Class 7 & Class 9 using perceptron and SVM (soft and hard) with all 160 features used (after mutual information feature extraction pre-processing)

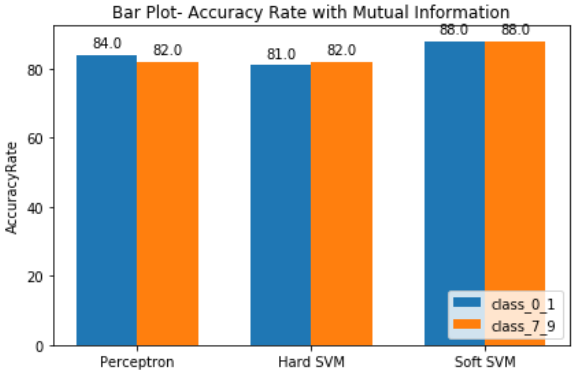
Program Name- Assignment3_with MI.py

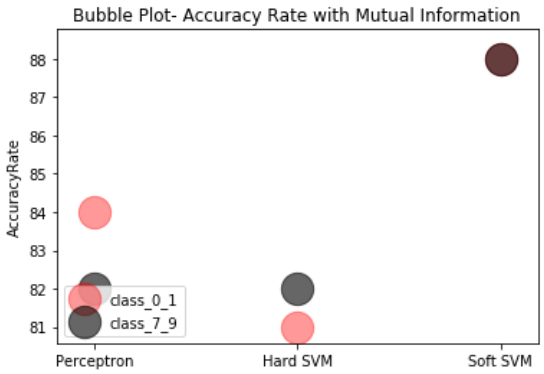
Data set – Class 7 & Class 9 data

Result Table:

Dataset	Algorithms	Accuracy %
Class 7 & 9	Perceptron	82.0
Class 7 & 9	Hard Margin SVM	82.0
Class 7 & 9	Soft Margin SVM	88.0

Plots:





Experiment-4(a)

Check Classification Accuracy for first pair Class 0 & Class 1 using perceptron and SVM (soft and hard) with all 240 features used (after mutual information feature extraction pre-processing)

Program Name- Assignment3_with MI.py

Class 0 & Class 1 data

Result Table:

Dataset	Algorithms	Accuracy %
Class 0 & 1	Perceptron	88.0
Class 0 & 1	Hard Margin SVM	90.0
Class 0 & 1	Soft Margin SVM	93.0

Experiment-4(b)

Check Classification Accuracy for first pair Class 7 & Class 9 using perceptron and SVM (soft and hard) with all 240 features used (after mutual information feature extraction pre-processing)

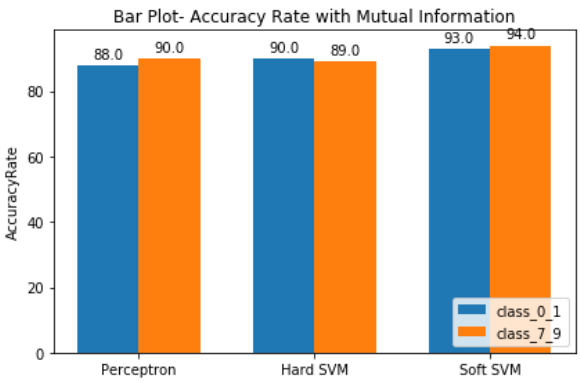
Program Name- Assignment3_with MI.py

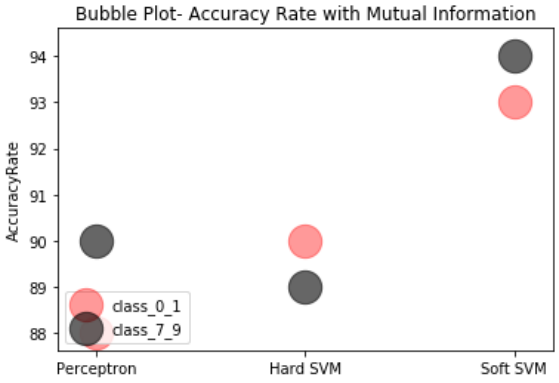
Data set – Class 7 & Class 9 data

Result Table:

Dataset	Algorithms	Accuracy %
Class 7 & 9	Perceptron	90.0
Class 7 & 9	Hard Margin SVM	89.0
Class 7 & 9	Soft Margin SVM	94.0

Plots:





Experiment-5(a)

Check Classification Accuracy for first pair Class 0 & Class 1 using perceptron and SVM (soft and hard) with all 400 features used (after mutual information feature extraction pre-processing)

Program Name- Assignment3_with MI.py

Class 0 & Class 1 data

Result Table:

Dataset	Algorithms	Accuracy %
Class 0 & 1	Perceptron	89.0
Class 0 & 1	Hard Margin SVM	91.0
Class 0 & 1	Soft Margin SVM	95.0

Experiment-5(b)

Check Classification Accuracy for first pair Class 7 & Class 9 using perceptron and SVM (soft and hard) with all 400 features used (after mutual information feature extraction pre-processing)

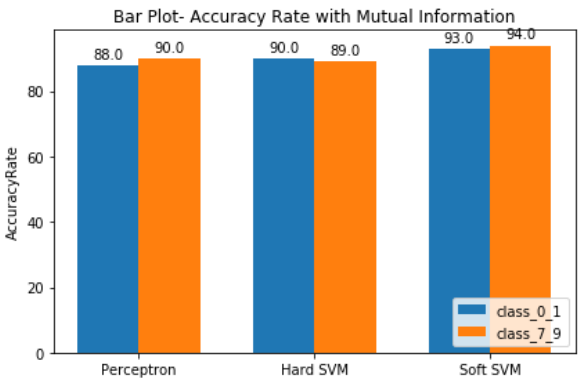
Program Name- Assignment3_with MI.py

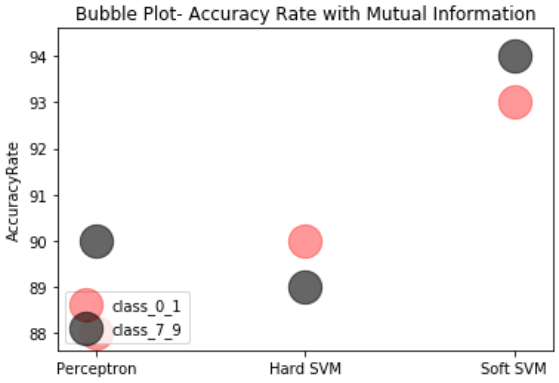
Data set – Class 7 & Class 9 data

Result Table:

Dataset	Algorithms	Accuracy %
Class 7 & 9	Perceptron	91.0
Class 7 & 9	Hard Margin SVM	93.0
Class 7 & 9	Soft Margin SVM	95.0

Plots:





Experiment-6(a)

Check Classification Accuracy for first pair Class 0 & Class 1 using perceptron and SVM (soft and hard) with all 500 features used (after mutual information feature extraction pre-processing)

Program Name- Assignment3_with MI.py

Class 0 & Class 1 data

Result Table:

Dataset	Algorithms	Accuracy %
Class 0 & 1	Perceptron	90.0
Class 0 & 1	Hard Margin SVM	95.0
Class 0 & 1	Soft Margin SVM	96.0

Experiment-6(b)

Check Classification Accuracy for first pair Class 7 & Class 9 using perceptron and SVM (soft and hard) with all 500 features used (after mutual information feature extraction pre-processing)

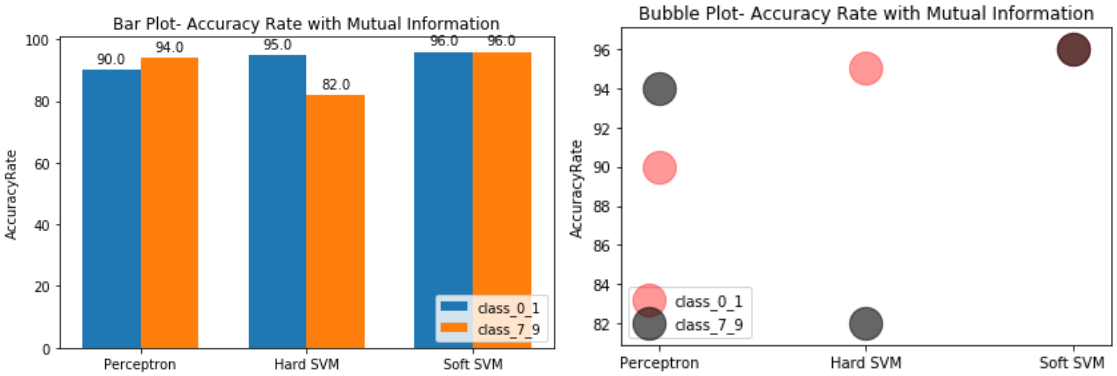
Program Name- Assignment3_with MI.py

Data set – Class 7 & Class 9 data

Result Table:

Dataset	Algorithms	Accuracy %
Class 7 & 9	Perceptron	94.0
Class 7 & 9	Hard Margin SVM	82.0
Class 7 & 9	Soft Margin SVM	96.0

Plots:



Final Conclusion (combined for all experiments based on the separate results tables and plots):

1. Experiment 1(a) (full 784 features) used for class 0 & 1 – max accuracy achieved is **96%** with Soft SVM
2. Experiment 1(b) (full 784 features) used for class 7 & 9 – max accuracy achieved is **97%** with Soft SVM
3. With other experiments done with mutual information feature extraction data pre-processing step. 80, 160, 240, 320, 400 and up to 500 mutual information features were selected in different experiments from the experiment # 2 to 6.
    - With only **80 TOP features**, max accuracy achieved for (class 0 & 1 pair) and (class 7 & 9 pair) - **80 % and 81 %** respectively
    - When features were increased up to **500** with more experiments, then max accuracy rate for the both class pairs across all algorithms were found 99% close to the experiment 1 (full features) i.e. 96% each for two class pairs
4. Based on all the experiments (2 to 6), selection of **features** based on a mutual information method look appropriate in this problem out of total 784 features, for achieving good reasonable accuracy (~ 93 % to 97 %) for all the algorithms.
5. Out of all the experiments, one thing is pretty evident that original data set in MNIST database for class pair 7 & 9 are highly distinct and relevant and less redundant compared to the class pair 0 & 1.
6. Hard Margin SVM (with C regularization parameter = 32K) is used and found less accurate compared to Soft Margin SVM (with C regularization parameter = 1) with a noticeable difference of up to 10% in 80 features experiment

Please note the role of C parameter in the linear SVC algorithm for determining Hard and Soft Margins-

The strength of the regularization is inversely proportional to C, we are doing the whole optimization exercise to maximize the margin. We convert our Maximal Margin problem to Optimal Margin problem to be able to solve it directly. So, in light of minimizing ||w||, (||w|| + C*slack) is basically working opposite. Every time there is a mis-classification, it will be incorporated by the margin will become thinner.

The higher the value of the C multiplier, the more SVM will be sensitive to noise. In the limit where C goes to positive infinite, we will have a hard-margin SVM. Hard-margin SVM will try to find a separating hyper-plane that has maximum distance from either classes' data points, such that all the data points in each side of the hyper-plane should be of the same class. This constraint might make problem unsolvable, i.e. SVM can't find a hyper-plane to separate data points perfectly, or equally, the optimization problem doesn't have an answer.

To overcome this problem, we can soften the aforementioned constraint and allow some misclassifications. Now some data points can be on the wrong side of the hyper-plane. But we penalize these misclassifications with a loss function. The C parameter is this loss function's multiplier; so, the higher the C, the more we penalize making errors.