E0-270 (Machine Learning)

Indian Institute of Science

MNIST Digit Recognition using Principal Component Analysis and Support Vector Machines

Susmit Das

Abstract:

In this project, we use have used Principal Component Analysis and Support Vector Machines on MNIST Dataset. 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. Each pixel of the image is represented by a single integer between 0 and 255, and so each image is a 784-dimensional vector. PCA has been implemented to reduce the dimensionality of this vector and a multi-class SVM model, consisting of ten binary SVM models, is trained using the training images with optimal values for hyperparameters, learning rate, number of iterations and regularization factor. Then the test images are classified using the trained multi-class SVM model. Then, an analysis is performed to find the minimum optimal number of principal components, which is sufficient for accurately predicting the digits.

Methodology:

1. First, the training and testing data is **normalized** by changing base and scale of the pixel data to fit between -1 and 1. The original grayscale values of each pixel lies between 0 and 255.

$$X_i \in (0,255)$$

$$X\_norm_i = \frac{X_i}{255/2} - 1$$

$$X\_norm_i \in (-1,1)$$

2. Then, **Principal Component Analysis** is used to reduce the dimensionality of the 784-dimensional vector of the training and testing images.
   Workflow of Principal Component Analysis:
   i) The feature matrix is standardized.
   ii) Covariance matrix of the features is computed.
   iii) The eigenvectors and eigenvalues of the covariance matrix is calculated.
   iv) The eigenvectors are sorted in the descending order by their corresponding eigenvalues.
   v) The first k (chosen number of components) eigenvectors is taken as components
   vi) The dot product of the standardized feature matrix and components is computed.
   vii) This dot product is the new reduced feature matrix.
3. Next, **Binary Support Vector Machine** model has been implemented as follows:

a. **Initialization**: – The weight vector is initialized with zero for all features and initial zero bias. For convenience of code, a constant feature vector with value of 1 has added to the feature matrix and bias is included directly to the weight vector.

$$\sum_{i=1}^{n}(w_i\, x_i + b) \;= \sum_{i=0}^{n}(w_i\, x_i) \qquad | \; where, x_0 = 1, w_0 = b$$

b. **Training/Fitting**: -

The training algorithm is run in a loop for a set number of iterations to gain the optimal values of the weights till the convergence occurs. The algorithm is as follows.

i. A random sample is chosen from the training dataset.
ii. The loss function value is computed with current weight values.

$$J(w) = \frac{\left|\left|w\right|\right|^2}{2} + c\sum_{i=0}^{n}\max\{0, 1 - y_i(w^T x_i)\}$$

iii. The gradient vector, $\nabla J(w)$ is initialized with zero sub-gradients.
iv. When $c\sum_{i=0}^{n}\max\{0, 1 - y_i(w^T x_i)\} > 0$, the gradient is defined. Hence,

$$\nabla J(w) = w + c\sum_{i=0}^{n} y_i x_i$$

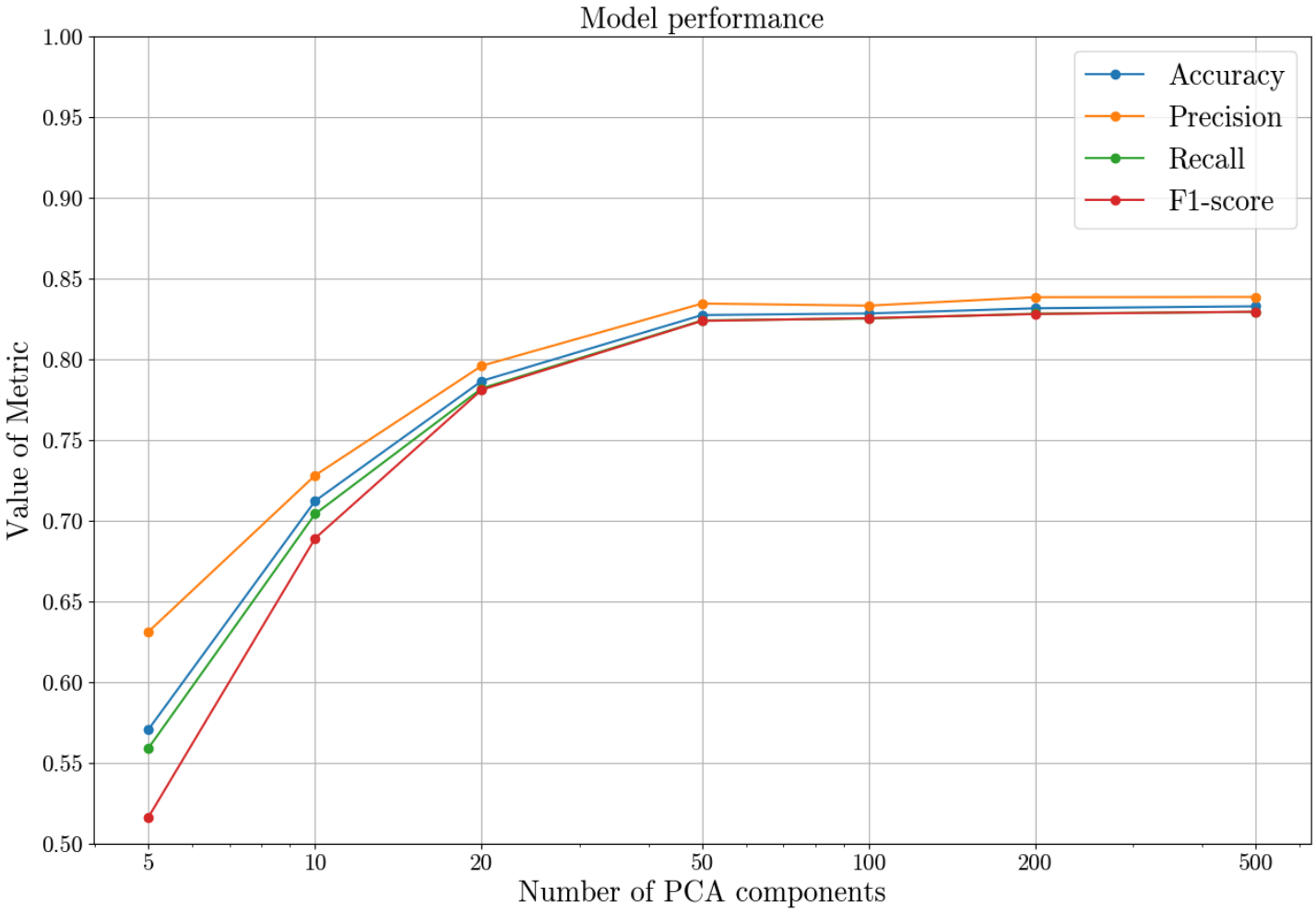
v. Weights are updated as follows:

$$w_{t+1} = w_t - \gamma \nabla J(w_t) \qquad | \; where\ \gamma\ is\ the\ learning\ rate$$

c. **Prediction Score**: - $y_{predicted} = w^T x$ is used by the model for binary classification. $y_{predicted} = 0$ is the decision boundary. Larger values indicate higher confidence on each side of the decision boundary.
d. **Prediction**: - This method checks the sign of the prediction score for any test data to make prediction of +1 for positive predictions scores and -1 for negative prediction scores.
e. **Accuracy, Precision, Recall, F1-score**: - The confusion matrix is created and used to give us the performance metrics of the binary SVM model.

4. **Multi-Class Support Vector Machine model** for classifying digits has been implemented as follows.
   i. Ten different binary SVM models are created, each to predict one of the digits from 0-9.
   ii. For each digit i, the labels of the training data are pre-processed such that label of each row becomes 1 for all labels corresponding to the digit i, -1 for all labels corresponding to the other digits, resulting in 10 different sets of training label data, one for each model.
   iii. Each model is trained with set hyper-parameters using its own pre-processed training data for all ten digits.
   iv. Prediction is done on the test data to evaluate Accuracy of the combined model. The binary SVM model giving the highest prediction score for each row of test data is chosen, because higher score indicates higher level of confidence.
   v. Separate confusion matrices are created for each digit from 0-9, which is used to get Precision, Recall, F1-score of each class. The mean of all classes gives us the macro-averaged Precision, Recall, F1-score.

Results:



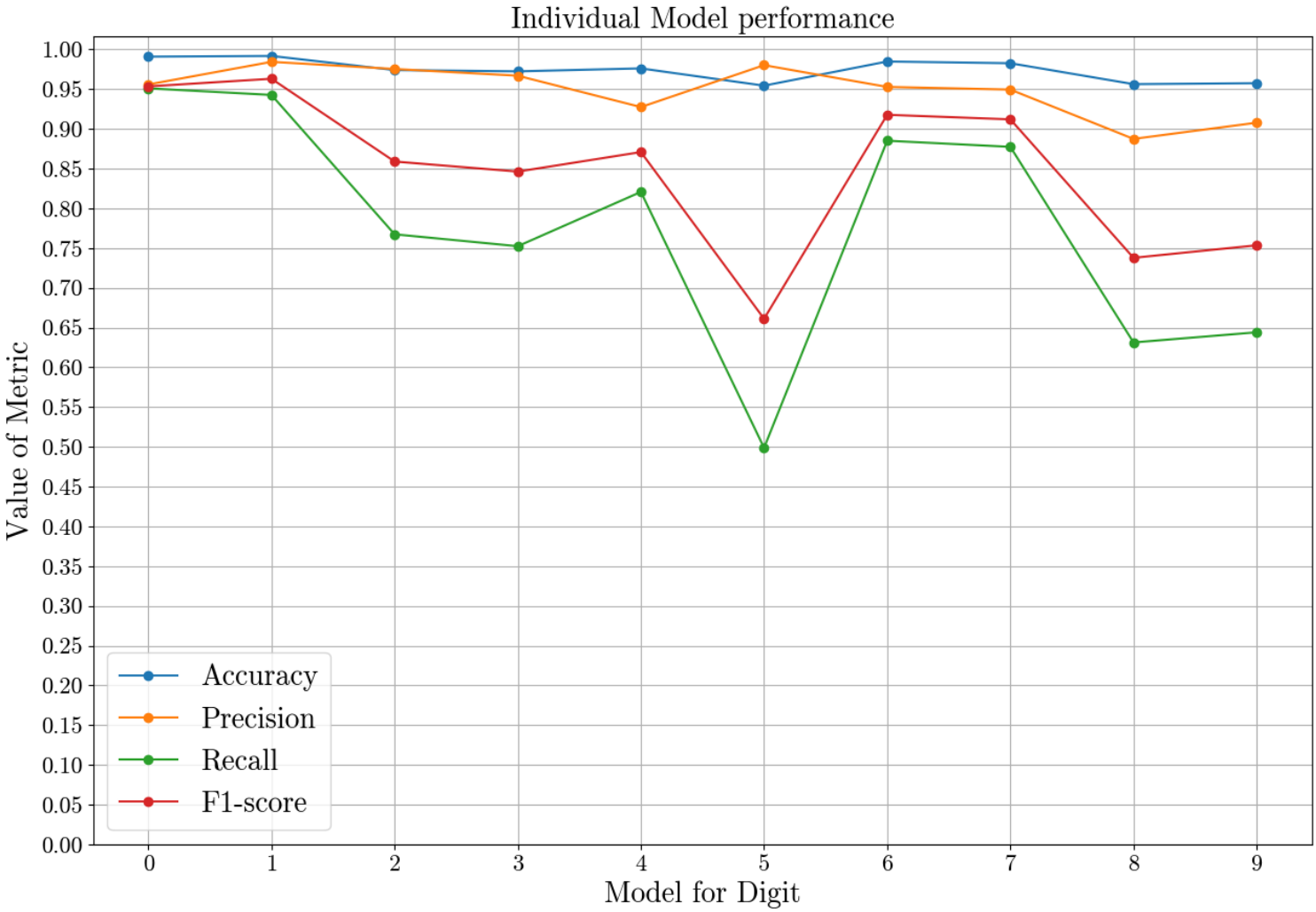
The process is repeated using different number of components for PCA and the performance metrics of the multi-class SVM model is compared using learning rate of 0.0001 at 10000 iterations and regularization factor as 1.0.

Number of Components	Accuracy	Precision	Recall	F1-score
5	0.5708	0.6313	0.5592	0.5165
10	0.7123	0.7281	0.7042	0.6891
20	0.7866	0.7960	0.7820	0.7811
50	0.8274	0.8345	0.8240	0.8238
100	0.8284	0.8332	0.8252	0.8255
200	0.8316	0.8384	0.8282	0.8280
500	0.8328	0.8386	0.8293	0.8295

Analysis:

1. It is evident from the plot that **50 is the optimal number of components**, after which all the performance metrics plateau so increasing the number of components further does not result in any significant improvement.
2. **Precision is the highest metric** for all values of k so it can be inferred that the model has lesser number of false positives compared to false negatives, especially at lower number of components.
3. When the model was run at lower number of iterations, the precision remained relatively high, but recall and f1-score was much lesser, indicating that **prevalence of false negatives decreases with higher number of iterations** at the cost of required training time.
4. Higher learning rates with constant number of iterations was resulting worse values for all metrics apart from precision.

Checking the performance metrics of each individual binary SVM model of the multiclass SVM model without using PCA gives us a clearer understanding of its overall performance.



Model of Digit	Accuracy	Precision	Recall	F1-score
0	0.9909	0.9559	0.9510	0.9534
1	0.9918	0.9844	0.9427	0.9630
2	0.9740	0.9754	0.7674	0.8590
3	0.9724	0.9669	0.7524	0.8463
4	0.9761	0.9275	0.8207	0.8708
5	0.9544	0.9801	0.4988	0.6612
6	0.9848	0.9528	0.8851	0.9177
7	0.9826	0.9494	0.8774	0.9120
8	0.9563	0.8874	0.6314	0.7378
9	0.9575	0.9078	0.6442	0.7536

1. The **accuracy** of all the individual models is very high (>**95**%) indicating good general performance.
2. Precision of each model is quite high, indicative of a very low number of false positives.
3. It can be seen that many of the models are struggling with relatively low recall, indicating the prevalence of a significant number of false negatives in their predictions. This is especially seen in the digits 5, 8, 9, which in turn affect the multi-class model, explaining the lower recall especially at small number of components.
4. Increasing the number of iterations significantly improves the recall of the models of 5, 8, 9, at the cost of computational power requirement. But it has also been observed that beyond 10000 iterations, the macro-averaged recall of the multi-class SVM model does not improve even though the recall of its individual binary SVM models improve.

References:

[1] Kaggle - Evaluation Metrics for Multi-class Classification

[2] Scikit-Learning Documentation and Source Code

[3] Support Vector Machines: Training with Stochastic Gradient Descent [University of Utah]