Classifying Handwritten Digits: A Comparison Study of Random Forest and Support Vector Machine Classifiers

Taihua Li
DePaul University
tli35@mail.depaul.edu

Abstract: This paper compares the performances of Random Forest and Support Vector Machine classifiers on the standard database of handwritten digits (MNIST). Each classifier is trained on 55,000 training images, and 10,000 testing images are used to prevent over- or under-fitting the models. The final models are scored using a set of 5,000 validation images. In this study, it shows that Support Vector Machine significantly increases the accuracy, precision and recall of the handwritten digit classification task, and the best model found is a Support Vector Machine with the Gaussian Radial Basis Function kernel, which produces an accuracy of 97%.

1. Introduction

Many efforts have been made to improve the classification performance in object recognition in images, which involves the learning of pixel intensities of images, such as Convolutional Neural Network classifier, which can achieve less 1% error rate in classifying handwritten digits (Simard, Steinkraus, & Platt, 2003). However, such model has a huge complexity in runtime, and even though with nowadays machine learning tools, such as *TensorFlow* by Google, it requires users to understand the notion of a certain neural network algorithm to construct the classifier, which is much more difficult than understanding other classifiers, such as Random Forest. In this paper, two algorithms are used to compare the classification performance on the handwritten digits dataset (MNIST).

For the rest of this paper, it is organized as following: first, a description about the dataset is presented, following a Principal Component Analysis for dimensionality reduction, and then the classification models are presented. Finally, model performances are discussed.

2. Data

The dataset is acquired from *The MNIST Database of Handwritten Digits*, composed by Yann LeCun at New York University, Corinna Cortes at Google Labs, and Christopher J.C. Burges at Microsoft Research. In the dataset, it contains 60,000 examples of 28×28 handwritten digit images in grayscale, with the intensity value ranges between 0 and 1. In addition, the dataset contain 10,000 examples of same sized handwritten digit images as the testing set. Among the 60,000 examples, 55,000 examples are separated as the training dataset and the rest 5,000 examples are used as validation dataset. There are ten class labels in total, as there are ten unique digits, ranges from 0 to 9. Here is a summary of the distribution of class labels in each dataset,

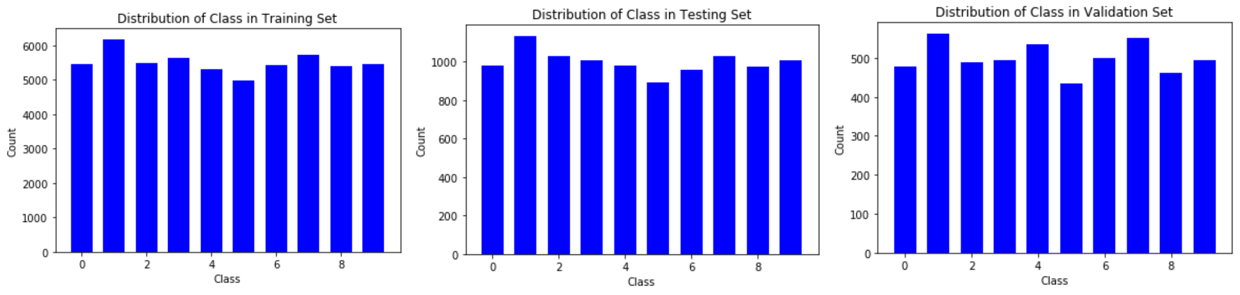

Figure 1: Bar graphs of class distributions (0 through 9) in training, testing and validation sets.

As shown above, the distribution of class labels across trainning, testing and validation sets are relatively similar to each other. This ensures that the classifiers built on the training dataset will discriminate one class well against all the other classes. In this case, there is no classs inbalance problem. As mentioned previously, each example is composed of a 28×28 handwritten digit images in grayscale, and here is a snapshot of the first 20 examples in the training dataset,

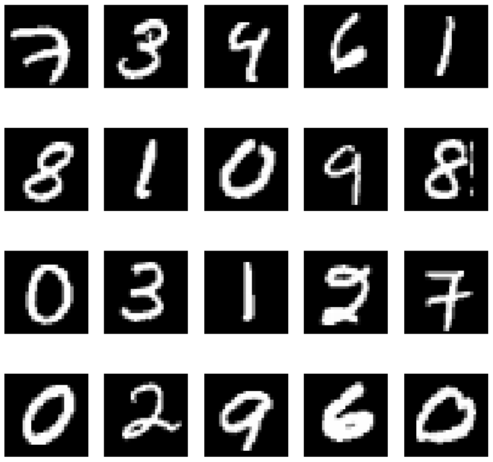

Figure 2: First twenty examples in the training set (28×28 handwritten digit images)

As shown above, images from the same class are unique from each other. For example, the images with written digit one from the first, second and third rows are not identical from each other. In addition, random noises are observed from the sample, such as that in the image with written digit eight in the second row. Since the noises do not distort the main content of the image, in this case, the handwritten digit eight, there is no need to get rid of them as they will be averaged out and ignored among all training examples. To better understand how the dataset is constructed, examples of 8×8 interpolated handwritten digit images are presented below,

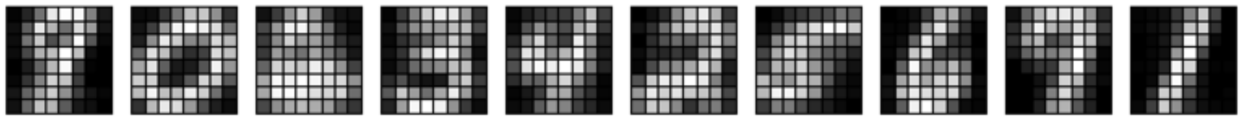

Figure 3: Examples of 8×8 handwritten digit images with interpolation

In each of these images, each pixel has a intensity value ranges between 0 and 1, where 0 represents a black pixel, 1 represents a white pixel, and values between 0 and 1 represents different gray pixels. MNIST has the same intensity range as the examples, except the image size is 28×28. To prepare the MNIST dataset for classifiers, each image is vectorized into a vector of length 784.

3. Principal Component Analysis

The training set has a dimension of 55,000×784, and it is computationally heavy for a CPU-only computer to learn and tune a classifier. In order to increase the efficiency of algorithm learning, the Principal Component Analysis (PCA) is applied. After computing eigenvectors and eigenvalues of the training image matrix, an explained variance plot is presented to find the number of component to reduce the original dimensionality to, as shown below,

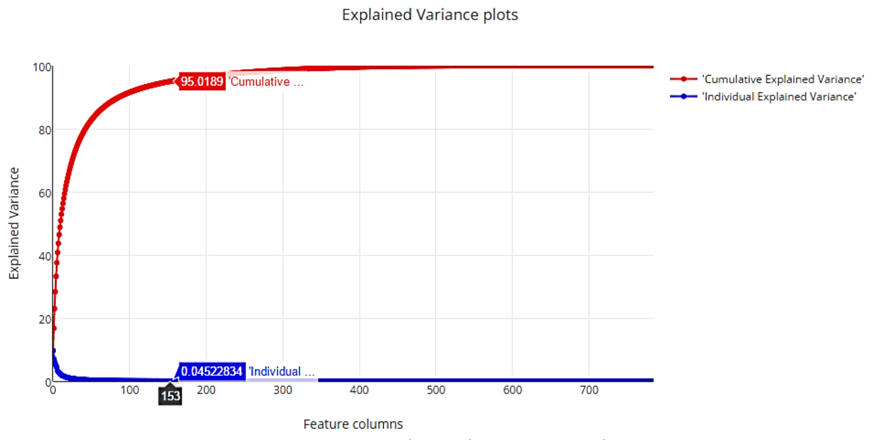

Figure 4: Explained Variance Plots

As shown in the plot, 154 (the first component has an index of 0) principal components are needed to keep at least 95% of the original dataset variance.

Eigenvectors capture and summarize the important information from the original dataset, and therefore they can be used to transform the original dataset into a reduced dimension. Each value in the eigenvector, named loadings, indicates the importance of each original dimension, which in MNIST dataset is 784. Here is a visualization of the first 20 eigenvectors,

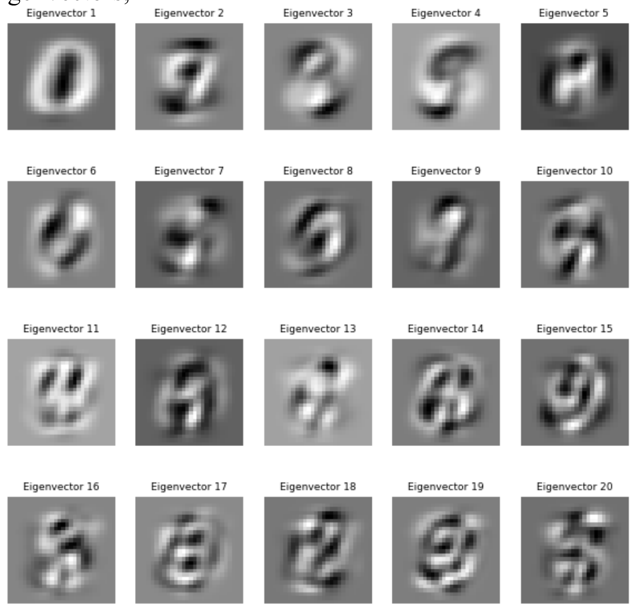

Figure 5: Frist 20 eigenvectors (reshaped from vector form to 28×28 images)

In the first eigenvector, which captures the largest variance in the original dataset, it shows that it captures the shape of a circle in handwritten digits. As more and more eigenvectors are visualized, it shows that more and more complex features from the original dataset are captured. Therefore, training, testing and validation sets can be transformed, using loadings from eigenvectors, from 784 dimensions into 154 reduced principal component dimensions. The dimension of the final training dataset is $55{,}000 \times 154$.

4. Classifiers

In this section, two classifiers are going to be presented; Random Forest and Kernel Support Vector Machine. In the Kernel Support Vector Machine section, three kernel approaches are presented.

4.1 Random Forest

To construct the Random Forest classifier, 5-fold cross validation is applied to find the best parameter, number of trees, on a small subset of the training data. After identifying a range of parameter values for grid search, the 5-fold cross validation is applied again on the full training data. By searching the Random Forest model with the parameter, number of trees, varying between 50 and 69, the best scored model is set to have 63 trees, and here is a cross validation score plot (mean and confidence interval) at each parameter tested,

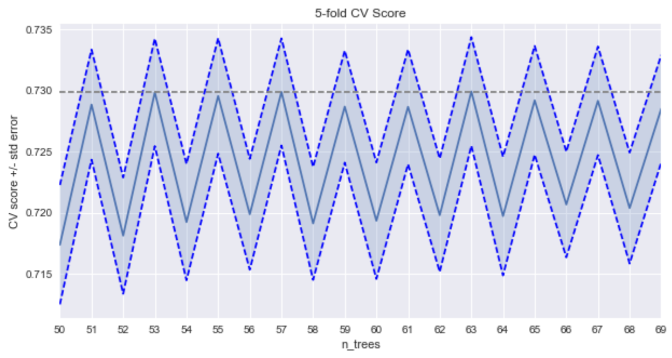

Figure 6: 5-fold cross validation score plot

4.2 Kernel Support Vector Machine (KSVM)

Kernel functions have been used to optimize the performance of Support Vector Machine. In this experiment, three kernels are tested; Gaussian Radial Basis Function (RBF), Linear and Sigmoid. Among these three kernels, Gaussian Radial Basis Function utilizes normal curves around the data points, whose sum constructs the decision boundary that can be defined as a type of topology condition such as curves where the sum is above a threshold value. Linear kernel attempts to transform non-linear vectors into linear vectors by adding or subtracting dimensions. Lastly, Sigmoid kernel performs like Gaussian Radial Basis Function, but instead of evaluating the sum, it evaluates if the logit value on the decision boundary is greater than a probability threshold. Therefore, since the image data are not linearly solvable, it makes sense to apply these three kernels to the data first before fitting Support Vector Machine model.

In the parameter tuning process, the built-in method in Python's *sklearn* has been used to tune the KSVM, with Gaussian RBF kernel, model's *gamma* value. For the error penalty parameter, a grid search of five values between 0.1 and 1 is conducted, and here is the result of the average 5-fold cross validation score for each value,

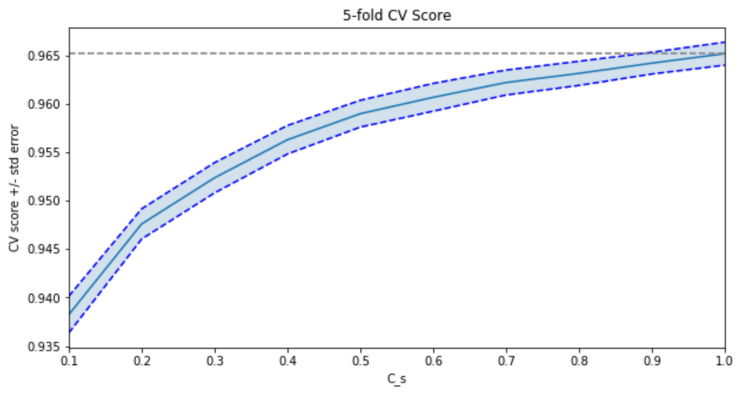

Figure 7: 5-fold cross validation score plot for error penalty paramter

As shown above, the best penalty parameter should be set to 1 to get the best Support Vector Machine result with the Gaussian kernel. For the KSVM model with Linear kernel, no parameter tuning is needed as there is no stochastic parameter in the Linear kernel function. For the KSVM model with Sigmoid kernel, the coefficient is set to be at 0.5 after several attempts of manual grid search for the ideal coefficients, where the evaluation metric is the accuracy of classification in the training set.

5. Classifier Performance

To evaluate the classifier performance, four metrics have been calculated; overall accuracy, precision, recall and F-1 score. In this section, each model, which learns solely from the training dataset, is evaluated against testing and validation datasets.

5.1 Test Set Performance

Class	Accuracy				Precision				Recall				F-1 Score			
Model	RF	G	L	S	RF	G	L	S	RF	G	L	S	RF	G	L	S
0	-	-	-	-	.96	.97	.96	.92	.98	.99	.98	.98	.97	.98	.97	.95
1	-	-	-	-	.98	.98	.98	.94	.99	.99	.99	.98	.98	.99	.98	.96
2	-	-	-	-	.94	.97	.92	.90	.93	.97	.94	.86	.94	.97	.93	.88
3	-	-	-	-	.92	.96	.91	.85	.94	.97	.93	.87	.93	.97	.92	.86
4	-	-	-	-	.92	.97	.94	.88	.95	.97	.96	.92	.94	.97	.95	.90
5	-	-	-	-	.93	.98	.92	.81	.93	.96	.90	.81	.93	.97	.91	.81
6	-	-	-	-	.96	.98	.96	.93	.97	.98	.96	.93	.97	.98	.96	.93
7	-	-	-	-	.96	.97	.95	.91	.93	.96	.94	.90	.94	.96	.95	.91
8	-	-	-	-	.93	.96	.93	.88	.90	.96	.91	.82	.91	.96	.92	.85
9	-	-	-	-	.94	.97	.95	.88	.92	.95	.92	.84	.93	.96	.94	.86
Average	.945	.970	.944	.893	.94	.97	.94	.89	.94	.97	.94	.89	.94	.97	.94	.89

5.2 Validation Set Performance

Class	Accuracy				Precision				Recall				F-1 Score			
Model	RF	G	L	S	RF	G	L	S	RF	G	L	S	RF	G	L	S
0	-	-	-	-	.97	.98	.96	.93	.99	1.0	.98	.97	.98	.99	.97	.95
1	-	-	-	-	.98	.97	.96	.90	.98	.99	.98	.98	.98	.98	.97	.94
2	-	-	-	-	.92	.97	.93	.87	.95	.97	.93	.88	.94	.97	.93	.87
3	-	-	-	-	.90	.96	.92	.84	.92	.96	.94	.84	.91	.96	.93	.84
4	-	-	-	-	.95	.98	.96	.90	.95	.98	.97	.94	.95	.98	.96	.92
5	-	-	-	-	.93	.97	.93	.79	.91	.97	.91	.76	.92	.97	.92	.78
6	-	-	-	-	.96	.99	.96	.95	.98	.99	.97	.94	.97	.99	.97	.94
7	-	-	-	-	.96	.98	.96	.92	.95	.97	.95	.90	.95	.97	.96	.91
8	-	-	-	-	.93	.98	.93	.89	.91	.96	.91	.82	.92	.97	.92	.85
9	-	-	-	-	.96	.98	.96	.90	.92	.95	.92	.86	.94	.97	.94	.88
Average	.946	.974	.947	.892	.95	.97	.95	.89	.95	.97	.95	.89	.95	.97	.95	.89

In the performance tables above, RF stands for Random Forest, G stands for Support Vector Machine with Gaussian kernel, L stands for Support Vector Machine with Linear kernel, and S stands for Support Vector Machine with Sigmoid kernel. A ROC plot has been made for each class in each set of data and it can be found in the Appendix.

6. Discussion

As shown in Section 5, Random Forest, in general, performs worse than Kernel Support Vector Machine. However, Random Forest still score over 90% overall accuracies in both training and testing datasets. The model that is slightly better than Random Forest, in accuracy, is the Kernel Support Vector Machine with Linear kernel. The Support Vector Machine model with Gaussian kernel scores the highest accuracy, precision and recall overall, where all evaluation metrics used are over 97%. The worst model experimented in this study is the Support Vector Machine with Sigmoid kernel.

Gaussian kernel approach performs the best is due to its regularization properties. A Support Vector Machine classifier with Gaussian kernel is a weighted linear combination of the kernel function computed between each data point and each of the support vectors. With its regularization properties, the estimation and approximation errors are minimized, where the estimation error is the error in the estimation model, which in this case is Support Vector Machine, and the approximation error is the error incurred by limiting the space of classification model over which the search for the support vectors are performed. Therefore, Gaussian kernel approach performs the best and scores over 97% in accuracy, precision and recall.

7. Conclusion

In this study, four models are built to classify the MNIST handwritten digit image dataset, where the predicted class contains ten classes, ranging from digit zero to digit nine. The models built include Random Forest with 63 trees, Support Vector Machines with Gaussian, Linear and Sigmoid kernels. Whilst Random Forest and Support Vector Machine with Linear kernel score very similar accuracy, precision and recall at around 95%, Support Vector Machine with Gaussian kernel performs the best in the classification task due to its error regulation properties. Support Vector Machine with Sigmoid kernel has the worst performance, with slightly below 90% accuracy, precision and recall. Overall, it shows that Kernel Support Vector Machine classifier significantly improves the classification performance of image data over Random Forest.

For future work, the kernel for Support Vector Machine should be designed according to the dataset. Other classifiers should be attempted as well, such as K-Nearest Neighbor and regression approach. However, due to the large dimensions of image data, especially when the dataset contains RGB images, computation efficiency becomes a problem, which shall be addressed in future work as well.

8. References

Simard, P. Y., Steinkraus, D., & Platt, J. C. (2003). Best Practices for Convolutional Neural Networks Applied to Visual Document Analysis. *ICDAR*, 958-962.