Evaluating Predictive Performance of Statistical Models

Nischal Subedi

Applied Statistics

1 Introduction

Over the past 10 years, there has been a rapid growth in the field of artificial intelligence and machine learning which has seen great progress in image and digit classification, that includes proposal of various models and model enhancing techniques to improve predictive performance.

In this paper, the motivation is to compare six mainstream statistical models. These models are Multinomial Logistic Regression, Convolutional Neural Network, Gradient Boosting , Support Vector Machine, Multi-class Linear Discriminant Analysis and Random Forest.

For the purpose of this project, MNIST dataset has been provided with slight modification. In order to reduce the computation time and complexity, a random sample of 10,000 training images and 5,000 testing images are taken, without replacement, from the original data set. For each image, there are 28*28 pixels that contains a value from 0-255. Moreover, the data was already prepossessed and split into training and testing set. Three files were used for this analysis: `mnist_train_counts-1, mnist_test_counts-1`, and `mnist_test_counts_new.`

After the best modeling efforts, the goal is to make prediction on the testing dataset. Two separate text files containing the prediction for datasets– `mnist_test_counts-1` and `mnist_test_counts_new` will be provided along with the report. In addition, the mis-sclassification rate of the models on `mnist_test_counts-1` will be compared and the method used will be discussed.

2 Methods

2.1 Multinomial Logistic Regression

The first model that was fit to the dataset is Multinomial Logistic Regression. Multinomial Logistic Regression uses softmax function to assign final probabilities to each of the discrete outcome of digit class. Ridge regularization has been used in order to prevent the model from overfitting. 'lbfs' solver which stands

for Limited-memory Broyden–Fletcher–Goldfarb–Shanno is used for fitting this particular model. It approximates the second derivative matrix and updates it with gradient evaluations. It stores only the last few updates, so it saves memory. It isn't very fast with large data sets but can very well handle the dataset we are working with.

2.2 Random Forest

Random Forest is a bagging technique meta estimator which fits a number of decision trees classifiers on the sub-samples of the dataset and uses averaging to improve the predictive performance of the model. Since, Random Forest has a lots of hyperparameters to tune, one of the methods we used to aid in selecting good hyperparameter to obtain a better fit was Grid Search Cross Validation approach(GridSearchCV). GridSearchCV loops through a pre-defined hyperparameters and fits the best combination of hyperparameter on the training set.

2.3 Support Vector Machine

Support Vector Machine is one of the most widely used algorithm for classification tasks. The objective of support vector machine algorithm is to find a hyperplane in an N-dimensional space where N represents the number of features that distinctly classifies the data points by finding the maximum margin, i.e, the maximum distance between the data points among classes. GridSearchCv approach was used for hyperparameter selection as well. List of kernels including linear, polynomial, and sigmoid were passed via GridSearchCV to choose the best kernel that fits the training set. The 2nd and 3rd degrees of polinomial were specified. Finally, gamma was set to 'auto' allowning the model to use 1/n features while training.

2.4 Gradient Boosting

Gradient Boosting is an additive model in a forward stage wise fashion which it allows for the optimization of arbitrary differentiable loss function. GridSearch CV technique is used for optimizing the hyperparameters of this algorithm as well. The deviance loss function is used for classification with probabilistic output. Small range of learning rates were given for the model to choose the best from using the GridsearchCV algorithm. The criterion to evaluate the trees were chosen to be mean squared error and the number of boosting stages was specified as well.

2.5 Linear Discriminant Analysis

This is a classifier with linear decision boundary, generated by fitting class conditional densities to the data using Bayes' rule. The model fit Gaussian density to each class, and assumes that all the classes shares same covariance matrix.

The similar approach of GridSearchCV was used for tuning the hyperparameters for this model as well. Solvers such as singular value decomposition (svd), and least squares solutions (lsqr) were passed for the GridsearchCV to choose from. In addition, shrinkage was set to none as shrinkage would not be able to be combined with svd given that svd was choosen to be better hyperparameter to tune the model.

2.6 Convolution Neural Network

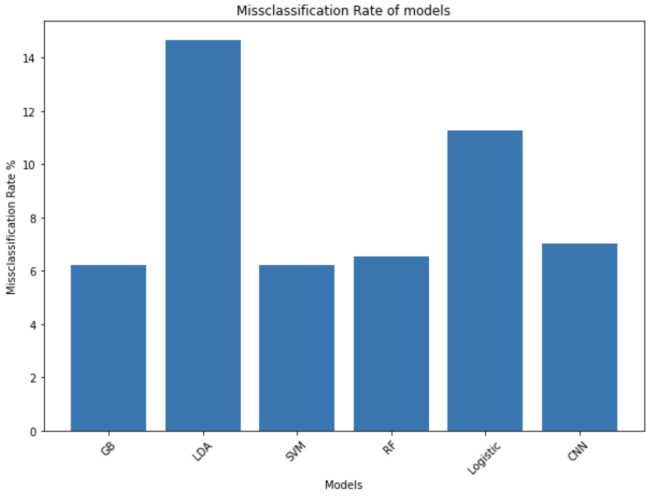
Finally, we used 1D Convolution Neural Network (CNN) to classify digits. A sequential model with 7 layers were defined which included input, hidden, max-pooling, dropouts, and output layers. Batchsize of 32 was defined to make training memory efficient. Two dropout layers were added to prevent the model from overfitting. Each hidden layers had 32 nodes. Finally, flatten layer was added prior to output layer to flatten the input prior to going through the output layers.

For this multiple classification problem, categorical cross entropy, one of the popular metrics, is used to evaluate the model. It is a combination of softmax activation plus the cross entropy loss. This metric will output probability over C classes for each digit.

Moreover, Adam is an optimization algorithm that can be used to update the network weights iteratively. Adam has been a popular optimizer choice over stochastic gradient descent because of its nice properties which combines the AdaGrad and RMSProp algorithms that can handle sparse gradients on noisy problems.

Misclassification rate function was defined and was used to evaluate the misclassfication rate of all the models. Time functions were used to keep track of the training time for each models as well.

3 Result

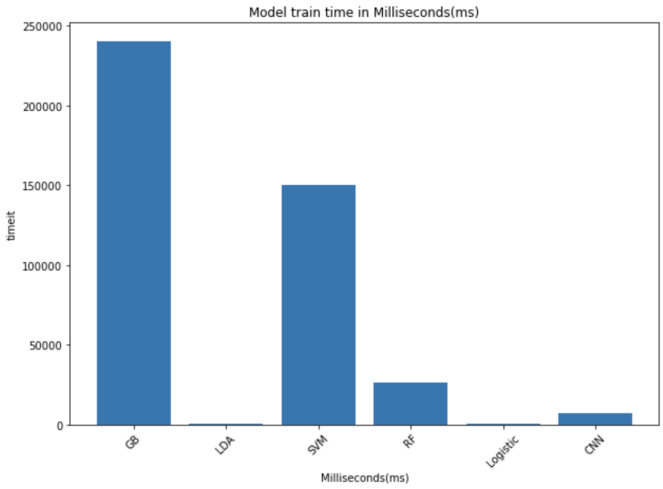


After the models were fitted, we evaluated the performance of model on the testing set using predefined metric called mis-classification rate. As mentioned earlier, the prediction labels for testing set and new testing set will be provided along with this report.

One of the reasons LDA has highest mis-classification rate is its assumption that all classes have same co-variance matrix and model fits Gaussian density to each class. However, for the dataset we used, it is more likely that each class is not Gaussian distributed.

Similarly, Gradient Boosting algorithm achieved the lowest mis-classification rate because it improves upon the error of past trees and uses several trees to fit the final model. This allows the algorithm to have very good predictive power and tend to perform well in unbalanced dataset like the one in our case.

From the bar chart, we can see that gradient boosting achieves the lowest mis-classification rate followed by support vector machine, random forest, etc respectively.



In addition, time tracking for all the model fit was also performed using prebuilt timeit function in python. From the bar chart above, we can see that the gradient boosting algorithm was slowest to train followed by support vector machine, random forest, and CNN respectively. Training in batches for CNN significantly lowered the training time. Gradient boosting took the longest time because of looping through multiple trees to produce final output.

4 Conclusion and Future Works

Overall, the missclassification rate for all the models seems to be below 15% which implies that the performance of these statistical models is very good.

With respect to the future works, I would propose to fine tune the models using additional methods that could potentially lower the missclassification rate further. Even though the GridSearchCV was used, limited search space was provided for the model to choose the best hyperparameters from due to computational complexity. A larger search space could be provided and other methods such as RandomizedSearchCV can be used which could lead to potentially a better search space. One of the

Additionally, the modeling approach assumed that the distribution of classes were more or less equal. However, in reality, it is more than likely to observe the imbalanced classes. In such cases, the models needs to be optimize to provide more weight to classify the minority samples.

Moreover, GPU could be used by building models in platform like google colab to reduce the computational time by utilizing the provided GPU access. Likewise, each algorithm could be optimized to get the reduced training time. For example, early stopping concept for gradient boosting was introduced which has been shown to achieve almost the same accuracy as compared to a model built without early stopping using many fewer estimators. Thus techniques such as early stopping could be utilized to significantly reduce training time, memory usage and prediction latency. For the algorithm like gradient boosting that takes long time to train, we can adopt solutions like adopting larger learning rates, use multi-threading for parallel processing, etc to speed up the training time.