Comparison of Multi-Classification models on the EMNIST dataset.

Introduction

Supervised image classification on the EMNIST dataset is to be carried out. The EMNIST dataset is commonly used with classification and computer vision systems. As part of this project, we will be using truncated data (26,000 images rather than 100,000 items in the original data set) rather than the original number of images. This is due to computational constraints, resource limitations, reducing bias (by using a subset, we can identify potential biases before scaling to entire data set) etc. We will train 4 models and make inferences from the truncated data. Different models have their advantages and disadvantages; therefore, it is important for us to find the most suitable model for inferring data on the EMNIST dataset.

Data and Preparation

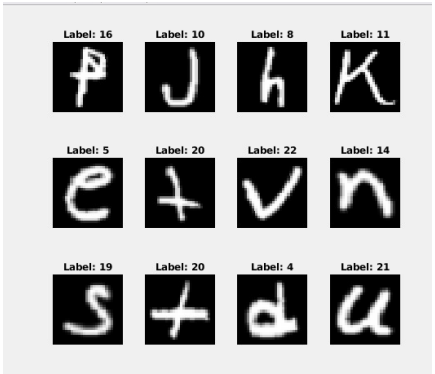
Each image is made of 28x28 pixels. These images are stored as 1x748 vectors for ease of processing and analysis. For example: storing the images as 1D arrays can significantly reduce the overall dimensionality of the data, making the computations and training of the models more tractable and quicker . Load the data into MATLAB and save the labels and features (images) in different parameters. Then we convert the image information from the dataset to a double type. We can reshape the image size to 28x28 allowing us to easily view the images or features when required. We employ a vector approach for generating a random 3x4 matrix of images with their respective labels. Next, we save to a Png File.

Additionally, we split the data into training and testing data. We do this for both the label and images. We split the training and testing set with a 50% split each. One advantage is: equal representation of training and testing. Furthermore, we have a fair assessment, as the testing set holds enough instances to fairly evaluate the model's performance without being too little or too big. Finally, we check the label distribution for the training and testing data of the labels using the hiscounts function. If the labels are distributed appropriately, this helps us understand the data better.

Methodology

Train the K-Nearest Neighbour (KNN) model using two different distance metrics (Euclidean distance and cosine distance). We classify the test data using fitcknn(). The number of K-Nearest Neighbours is defined to be 5 and passed into the fitcknn() function which is a reasonable starting point for a smaller data set. In the future, it would be more effective to find the optimal value of k in MATLAB using a reasonable range (e.g., 1 to 20). As a means of comparison, we use the SVM for multiclass and Ensembles algorithms. This step begins with selecting 2 existing classification algorithms implemented in MATLAB (e.g., SVM Model, Ensembles Model) by using their respective functions (fitcecoc(), fitcensemble(), etc.) available in MATLAB's Statistics and Machine Learning Toolbox.

The format for each model in MATLAB is the model name followed by predictions. For example: knnModelEuclidean, followed by predictEuclidean. The models are compared based on model performance (training and testing time calculated using tic toc commands) and accuracy. Comparing these models allows an evaluation of their predictive performance.



We can compare these models for other reasons too. The KNN model is chosen as it is resistant to outliers. As it considers most of its neighbours, KNN tends to be less affected by outliers. Regardless of the occasional anomalies it can perform reasonably well. The SVM for multiclass model is compared with the KNN model as it has a contrasting approach to classification. Comparing these different approaches helps in understanding their strengths and weaknesses. The Ensembles model can be more complex due to their combination of multiple models, while KNN is relatively simple. Therefore, comparison with these models can provide insight into the relationship between model complexity and performance.

Results

Classification Model	Accuracy	Training Time	Testing Time
KNN (Euclidean)	0.78262	0.086818	85.9482
KNN (Cosine)	0.78262	0.035599	84.9806
SVM for Multiclass	0.73377	43.5679	12.1879
Ensembles	0.46992	116.5003	0.7474

The model with the greatest accuracy is the KNN model, with SVM for multiclass having a similar accuracy rate. The model that had the lowest accuracy rate and highest training time is Ensembles. Thus, ruling this out as the best option (results are more extreme than others). The model with the highest testing times is the KNN models. The results for the KNN model (Euclidean distance) and KNN model (Cosine distance) produce quite similar results. However, the training time for the cosine distance is slightly lower. This could be due to the algorithmic complexity of the Euclidean distance in comparison with the cosine distance. Nonetheless, the use of the KNN model yields similar results for both cosine and Euclidean distance.

Conclusion

Taking into consideration both accuracy and computational time (training time and testing time), the best model out of the three is SVM for multiclass. Furthermore, SVMs are effective for high dimensional data, which can be the case with images. Also, they work well with small-medium sized classification. Although, the training time is still quite high compared to the KNN model, the testing time is significantly lower.

Further research with models that work will with larger datasets may be useful (save resources and time). Additionally, carrying out the K-Nearest Neighbour by calculating the optimal value of k with this dataset may result in higher accuracy rate and lower testing times.