Construction of two classifiers Logistic Regression and k-Nearest Neighbour

TongTong Liu tliu8353
Hitesh Wagle hwag3386
Dhanu Jhala djha5117

October 02 2018

Abstract

Logistic Regression and K Nearest Neighbour classifier are popularly used classifiers to solve many real life problems. In this report two classifiers namely, Logistic Regression (LR) and k-Nearest Neighbors (k-NN) classifiers are implemented. Gradient Descent was implemented as a part of forming the LR classifier. Further, performance evaluation of the two classifiers is done on evaluation metrics. The metrics used are accuracy, precision and recall . Based on the evaluation metrics results, we will depict and compare the performance of Logistic Regression and k-NN. Also, validation dataset has been formed so as to ensure building of a robust classifier. To improve time and space complexity, Singular Value Decomposition (SVD) was used for dimensionality reduction. The analysis shows that k-NN performed better than LR for some classes.

1 Introduction

Logistic Regression and the k- Nearest Neighbour Classifier are one of the most popular classifiers used in the academia and the industry. They are used to solve many real world problems. Learning a classifier is crucial aspect of automation. Today, numerous applications in real world scenario requires application of classifiers. They form a wide use in financial modelling, image mapping in search engine, climate change identification, driver less cars, natural disasters prediction, aerodynamic modelling, Robotic automation etc. In this report, Logistic Regression (LR) and K Nearest Neighbour (k-NN) classifiers have been implemented. In the later part, the performance of both the classifiers over training, validation and test datasets has been evaluated. Performance metrics like accuracy and precision have been used to do the analysis of the two classifiers.

By studying and creating a classifier having high acceptable accuracy, one can achieve the goals associated with the problem. Further, the classifier can be made more realistic over time by learning on more data. Continuous emergence of new classification algorithms and techniques in recent years necessitates such a review, which will be highly valuable for guiding or selecting a suitable classification procedure among Logistic Regression and KNN method. In the following report first, the pre-processing methods are listed along with

detailed explaination of classifier building procedure. The optimization of the LR classifier using gradient descent has been implemented by defining user defined function rather than using the internal library. Also, regularisation has been carried out to prevent overfitting. Furthermore, the training data has been partitioned into validation and training data so as to make a robust classifier. This also helps in choosing an appropriate regularization parameter lambda. In the end, the results of both the classifier has been listed. This helps in carrying out an extensive performance analysis of both the classifiers on all classes.

2 Methods

2.1 Pre-processing

In the pre-processing step, SVD and normalisation are performed. Normalisation bounds the pixel values between [0,1]. This is a standardised part of preprocessing. In next part, Singular Value Decomposition (SVD) helps to reduce the redundant data from the training data set. SVD on any matrix $\boldsymbol{X}$ is written as

$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^{\boldsymbol{T}}$$

where $\boldsymbol{X} \in \mathbb{R}^{m\times n}, \boldsymbol{U} \in \mathbb{R}^{m\times r}, \boldsymbol{S} \in \mathbb{R}^{r\times r} and\ V \in \mathbb{R}^{n\times r}$. The original data $\boldsymbol{X}$ is of rank r. Depending on the desired variance this rank can be reduced by selecting k highest eigen-values and corresponding eigen-vectors of the data matrix $\boldsymbol{X}$. We selected k such that 96% of our variance is captured in the reconstructed data. Thus the reconstructed reduced rank matrix $\boldsymbol{X}_{reconstructed}$ becomes as

$$\boldsymbol{X}_{reconstructed} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^{\boldsymbol{T}}$$

where $\boldsymbol{X}_{reconstructed} \in \mathbb{R}^{m\times n}, \boldsymbol{U} \in \mathbb{R}^{m\times k}, \boldsymbol{S} \in \mathbb{R}^{k\times k} and V \in \mathbb{R}^{n\times k}$. $k \ll r$ is the new reduced rank of the matrix [1] The effective compression ratio therefore becomes as

$$Comp\_ratio = \frac{mk + k + nk}{mn}$$

2.2 Classifier

2.2.1 Logistic Regression Classifier

Logistic Regression classifier is one of the most popular statistical model which uses a logistic function to model a binary dependent variable. It is a supervised classification algorithm and during the classification problem, the target variable(or output) takes discrete values for given set of features(or inputs) [1]. Based on the target category, LR can be classified as ordinal with 10 classes ordered from 1 to 10. LR classifier was trained using One Vs All methodology [3]. Therefore, 10 classifiers were formed, each trained for a specific class. By using Singular Value Decomposition and normalization of the data, approximately 20% higher performance was achieved but with an increase in computation time by 3 minutes. For some mathematics calculations, numpy library was used.

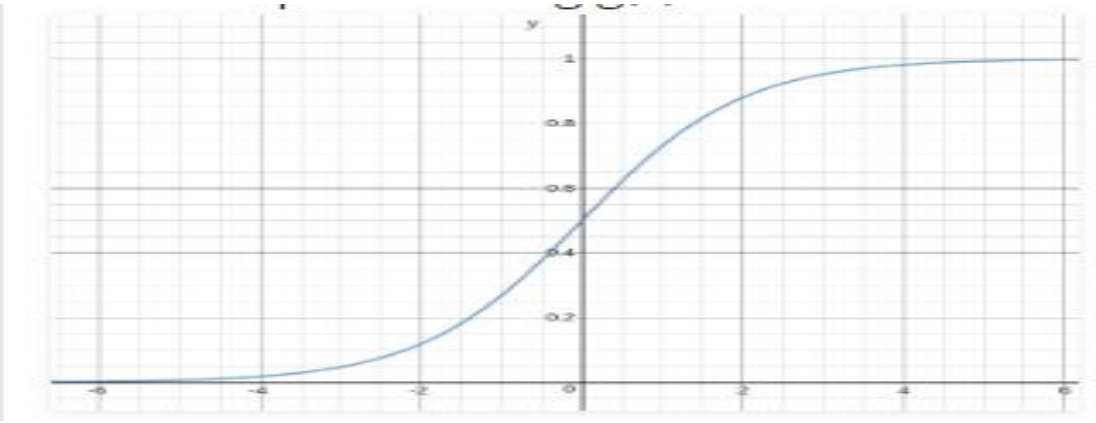

Figure 1: Sigmoidal Function

We have a multi-class data set with $\boldsymbol{Y}$ belonging to {0, 1, ..., $\boldsymbol{C-1}$} labels. Therefore, we have $\boldsymbol{C}$ number of classes. LR is defined using sigmoid function and can be explained as:

$$h_\omega(x) = \frac{1}{1 + e^{-}\omega^T x}$$

where

$$\omega^T x = \omega_0 + \omega_1 * x_1 + \omega_2 * x_2 ... + \omega_m * x_m$$

for a datapoint $\boldsymbol{x}$.

where $\omega_0, \omega_1, \omega_2 ... \omega_m$ are the regression coefficients and $\boldsymbol{m}$ is the number of features [1] [2]. Two of the important components of learning a classifier are the cost function and optimization methodology. In the case of LR we formulated our cost function for miscalculation of a data point x as below [4].

$$Cost(\boldsymbol{h(x)}_W, y) = \begin{cases} -log(\boldsymbol{h(x)}_W), \text{if } y = 1 \\ -log(1 - h(x)_W), \text{if } y = 0 \end{cases} \tag{1}$$

The aim is to minimise this cost function over all the misclassified data points. Thus one can write the above cost function over all the data points as below:

$$Cost(\boldsymbol{h(x)}_W, y) = -\frac{1}{N} \sum_{i=1}^{N} (\boldsymbol{y}_i \log(h(x_i)) + (1 - \boldsymbol{y_i}) \log(1 - h(x_i))) \tag{2}$$

To the above cost function regularization term has been added with regularisation parameter as λ. This is done to avoid overfitting on the training data. Thus, the cost function gets reformulated as below:

$$Cost(\boldsymbol{h(x)}_W, y) = -\frac{1}{N} \sum_{i=1}^{N} (\boldsymbol{y}_i \log(h(x_i)) + (1 - \boldsymbol{y_i}) \log(1 - h(x_i))) + \frac{1}{2} \lambda \|\boldsymbol{W}\|_2^2 \tag{3}$$

To minimise the cost function and find the optimum value of $\boldsymbol{w}$, Gradient descent methodology has been implemented. The steps for calculating the gradient are listed below:

$$J = -\frac{1}{N}\sum_{i=1}^{N}(\boldsymbol{y}_i \log(h(x_i)) + (1-\boldsymbol{y_i})\log(1-h(x_i))) + \frac{1}{2}\lambda\|\boldsymbol{W}\|_2^2$$

$$\frac{\partial J}{\partial w_k} = -\frac{1}{N}\sum_{i=1}^{N}(x_k(1-h_i)y_i - x_k h_i(1-y_i)) + \lambda w_k$$

$$\frac{\partial J}{\partial w_k} = \frac{1}{N}x_k\sum_{i=1}^{N}(h_i - y_i) + \lambda w_k$$

Therefore one gets the derivative with respect to $\boldsymbol{w_k}$ for k=0, 1, 2...m and thus a gradient of cost function with respect to $\boldsymbol{W}$ can be formed. This gradient can be used in gradient descent algorithm to update $\boldsymbol{W}$. The gradient descent algorithm is listed below:

Input: Data Matrix $\boldsymbol{X}_{m\times n}$, initial vector $\boldsymbol{W}^0_{1\times m+1}$, regularization parameter λ, learning rate η, convergence threshold ϵ and number of iterations (steps).

Output: The final result for $\boldsymbol{W}$.

Algorithm 1 Gradient Descent Algorithm

1: $\boldsymbol{W} \leftarrow \boldsymbol{W^0}$
2: **for** *t = 1* to *steps* **do**
3: Calculate predicted labels for all images
4: Use predicted labels and true labels to calculate the gradient ∇J
5: $\boldsymbol{W}_{t+1} \leftarrow \boldsymbol{W}_t - \eta\nabla\boldsymbol{J}$
6: Check for convergence $\|W_{t+1} - W_t\| \leq \epsilon$

This algorithm is used to learn the parameters for each class and therefore is run C times. The various stages in learning a classifier are listed below:

Dividing the data into training and validation: For this 20% of the data is randomly selected to form validation data set and the remaining 80% is used for training the classifier. This is done multiple times so as to reduce any variance in the learnt model parameters. The validation dataset is used for the fine tuning of regularisation parameter. For different values of λ, training and validation error are plotted. This helps us select the best regularisation parameter. We select a λ for which training and validation error combined to achieve a minimum. Furthermore, validation also helps a classifier to be robust. This is done by keeping an eye over the validation error and to observe the performance over validation dataset. To implement validation and training dataset divsion we write a function Validation_Dataset.

Regularisation over training dataset: To avoid overfitting a squared L2 norm over the model parameters is taken into account in the cost function. This helps avoid overfitting in the trained LR classifier. We observe the model's performance over different values of regularization parameter and select the one that performs the best [1].

Learning One vs All classifier: The learning for multiple classes using LR classifier is done using a training criteria of one vs all. For this we take a particular class (j), and create a binary label data wherein we mark the labels of that particular class (j) as one and of the rest all class labels as zero. We pass these binarised labels to a binary Logistic

regression classifier function. This helps learn the binary classifier for the particular class (j). Then after this process is carried out for all the other labels. Therefore, we learn total C classifiers charactersied by their W. This is achieved by implementing a function "One Vs All Classifier" in the code.

Testing Procedure: In order to test the data, the optimised parameters for each class is multiplied with one of the data point. The output probabilities are compared for each class for that particular datapoint. The assigned label to that datapoint is of the class for which the probability is highest. Therefore, for all the datapoints under test, a predicted class label is obtained. Furthermore, to calculate the accuracy predicted labels are compared with true labels. The testing procedure is carried over training data, validation data and test data. Therefore we get three accuracies.

Code Structure: The code for logisitc regression classifier is being summarised below. The summary contains important functions formulated to achieve different functionalities.

- Loading the data: Data is loaded using the function **loaddata**, where both image data set and label set is normalised in the range of [0,1].
- Bifurcation of data set: **ValidationDataset** fucntion is defined to bifurcate the given image data set and label set into the ration of 4:1 respectively. Model is then trained on the 80 percent of data set.
- Calculating Confusion matrix: Parameters of confusion matrix like true-positive, false-positive, false-negative and true-negative are calculated for each class and maintained in a list for further calculation of performance parameters like Accuracy, Precision, Recall and F-measure.
- Binary Logistic Regression classifier: **Binary-classifier** function is the most important function of the model as this functions trains the classifier on the input training sets, providing the number of iteration for the loop to run and train the model for each class. In this function, new set is created with respect to training data set which contains binary number for each image to depict whether an image belongs to that class (0 depicts does not belongs and 1 depict belongs). In the end of this function, the task of classifying an image with respect to a class in terms of 0 and 1 is achieved.

2.2.2 K- Nearest Neighbour Classifier

k-Nearest Neighbors (k-NN): k-NN classifier is a simple and effective non-parametric algorithm based on feature similarity. This algorithm does not build any model or have any data distribution related assumptions, i.e. there is no-explicit training procedure. The main aspect of k-NN lies in about testing phase, in which we retrieve all training samples for each query point, evaluate similarity between them. Then we decide value of k and return the majority vote of these k most similar samples from the training set [5].

There are many similarity measuring methods, like Euclidean Distance, Manhattan Distance, Minkowski Distance, Cosine Similarity and few more. But for k-NN, the most widely used is Euclidean Distance so we calculate similarity based on it [6]: Eucledian distance between two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ of m dimension is given by

$$Distance(d) = \sqrt{\sum_{i=1}^{m}(x_i - y_i)^2}$$

Effect of Data Pre-processing : Same as in Logistic Regression, Normalisation is applied to bound the pixel values between 0 and 1. Applying Normalization on all data set has no improvement on prediction accuracy (with same k) in k-NN because distance sorting order does not change, which means it has the same training samples vote as compared to data without normalization, but the running time reduces by 20-30 seconds through simplified calculations. Therefore, Normalization has been considered. SVD is applied on both training dataset with 96% variance. Number of components kept (rank) are 89. Although it affects the running time, but the accuracy is improved by 10% which signifies that SVD has reduced the noise very effectively. Below is the depiction of k-NN algorithm [1].

Input: Original matrix $\boldsymbol{X}_{m\times n}$, test point $\boldsymbol{X_q}$.

Output: The predicted label for data point $\boldsymbol{X_q}$.

Algorithm 2 KNN algorithm

1: **for** *i = 1* to *All test points* **do**
2: Calculate the eucledian distance of test point from every other training data point.
3: Sort the distance in ascending order but preserving the corresponding class index.
4: Count the k distances and observe the majority class
5: Assign the label of majority class to data point X_q

Selecting the K: This method has been tested on different values of k (3-10) on validation set to achieve higher accuracy. Value of k between (3-10) has almost no effect on the accuracy therefore value k=6 is set as default.

3 Experiments and Results

3.1 Experimental setup

3.1.1 Dataset

The dataset is of 30000 images belonging to 10 different classes. Each image data has a label ranging from 0 to 9. Around 20% of this dataset has been used as validation dataset and the remaining 80% is used as training dataset. Each image is a grey-scale image of the form 28*28 dimension thus forming a feature vector of 784 dimensions.

3.1.2 Evaluation Metrics

Performance analysis of the classifiers have been done on metrics of accuracy and time complexity. Furthermore, to analyse various classes among themselves, a confusion matrix has been formulated. Thereafter, parameters like precision, accuracy, recall, F statistic have been calculated for every class.

Accuracy: This metric is explained as the number of correct classifications over the total number of test data points taken for classification. Mathematically accuracy can be explained as:

$$Accuracy = \frac{TotalNumberofCorrectClassification}{Totalnumberoftestdatapointsused} * 100\%$$

Inter Class Evaluation Metrics: To analyse the robustness of the classifier over different classes, various performance metrics associated with binary classification are taken in to account.

Confusion Matrix: Confusion matrix gives an idea about the predictive capability of the model. Rather than focussing on time complexity or scalability, it focuses on the predictions made by the classifier for different possibilities. Figure 2 shows the confusion matrix [1] where, TP: TruePositive, FP: FalsePositive, FN:FalseNegative and TN: TrueNegative.

	PREDICTED CLASS		
ACTUAL CLASS		Class=Yes	Class=No
	Class=Yes	a (TP)	b (FN)
	Class=No	c (FP)	d (TN)

Figure 2: Confusion Matrix

Accuracy for binary classification: Accuracy is defined for a binary classification as the ratio of sum of true positive and true negative to the sum of true negative, true positive, false negative and false positive. Therefore the mathematical representation for Accuracy becomes as:

$$Accuracy = \frac{TruePositive + TrueNegative}{TruePositive + TrueNegative + FalseNegative + FalsePositive}$$

Precision: Precision is defined as the ratio of True Positive over the sum of True positive and False positive. Therefore, this metric gives the sensitivity of the model towards true positive and false positive. Precision becomes important in an application or task where False positive classification is of high cost. Precision mathematically becomes as:

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive}$$

Recall: Recall is defined as the ratio of True Positive to sum of True positive and False Negative. Therefore, recall is of great concern when the cost of False Negative is high such as tasks of disease diagnosis. Mathematically recall is defined as:

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative}$$

3.2 Experimental results

We present the results for two classifiers, LR and k-NN. We show their accuracy as well as time taken in running the algorithm. Furthermore, we present a detailed analysis over learning achieved on each class label. We also show the root mean square error and its variation with regularisation parameter lamda.

3.2.1 Accuracy and Execution time

Table 1 shows the accuracy and Table 2 shows the execution time. A point to note here is that the total execution time for Logistic Regression is 3 times the time taken for each validation (i.e. 12 minutes as mentioned in the table below). This is because we sample the validation dataset 3 times from training data and take a mean of the results. This ensures a reduced variance in the final model. However, the run time gets amplified because of this process. However the algorithm takes only the below mentioned time to run.

Table 1: Accuracy (in %)

Classifier	Training Data	Validation Data	Testing Data
Logistic Regression	76.11	75.73	75.60
KNN	89.13	85.45	85.15

Table 2: Execution Time (in minutes)

Classifier	Logistic Regression	KNN
Time	12.45	12 (& 36 mins for 3 fold validation)

3.2.2 Logistic Regression Experimental Results at the level of each class:

The Figure 3 depicts the plot of mean square error over regularisation paramter lamda ($\ln \lambda$). We can observe from the plot that the as lambda increases from 0 to 1 the training and the test error follows a similar trend. This shows that there is little overfitting taking place in the model. This may be because of high number of training data which reduces possibility of overfitting. We carried out this process to select the best regularization parameter. However, since there is no depiction of overfitting, we took a lamda for which error was not large. Therefore we took $\log \lambda$ as -50.

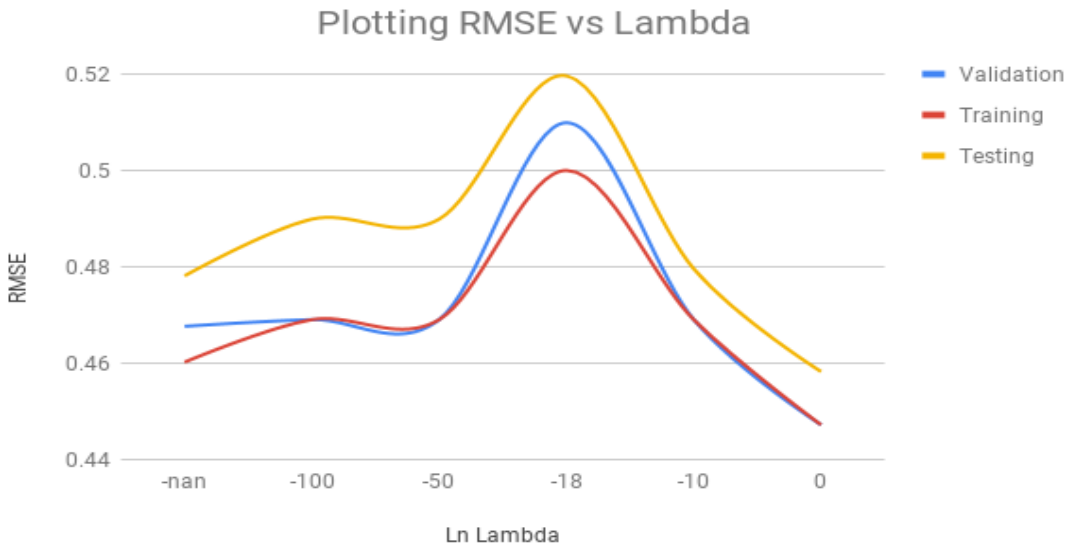

Figure 3: Plot of RMSE Vs Lamda

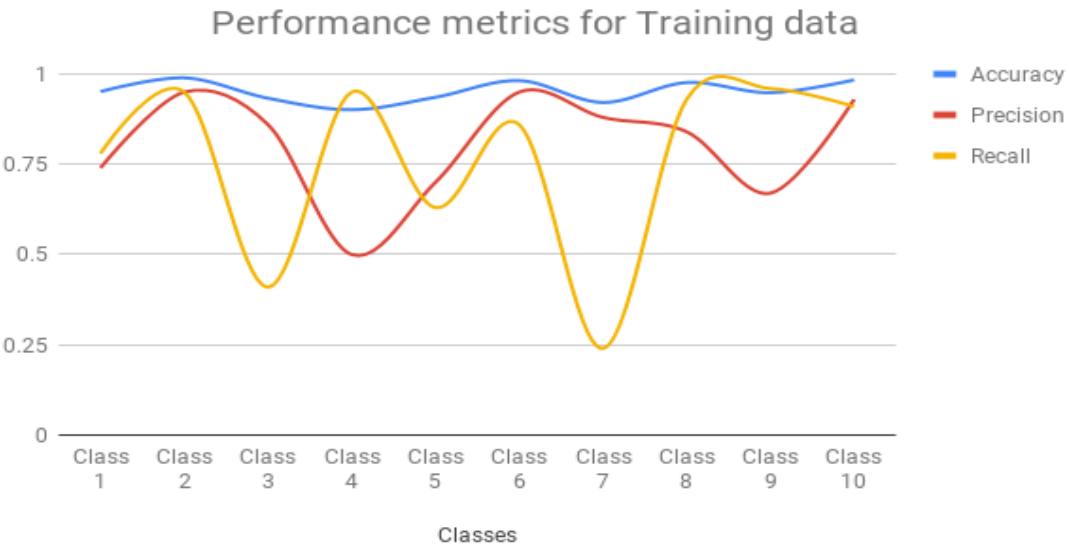

Figure 4: Plot of performance metrics Vs classes Training

From Figure 4, 5 and 6, we can observe that the LR learns on different classes differently. We observe that class 3, 5 and 7 are confused in learning the data classification. This is because, these classes have images similar to other classes. For instance class 7 has images which are a combination of other classes. We also observe high performance on distinctive classes like class 1, class 6, class 10, class 4. These classes are learnt efficiently. For eg: class 9 and 10 are of wallets and shoes which are very different from classes of clothing. Figure 7 is the confusion matrix obtained for training, testing and validation data for all the classes.

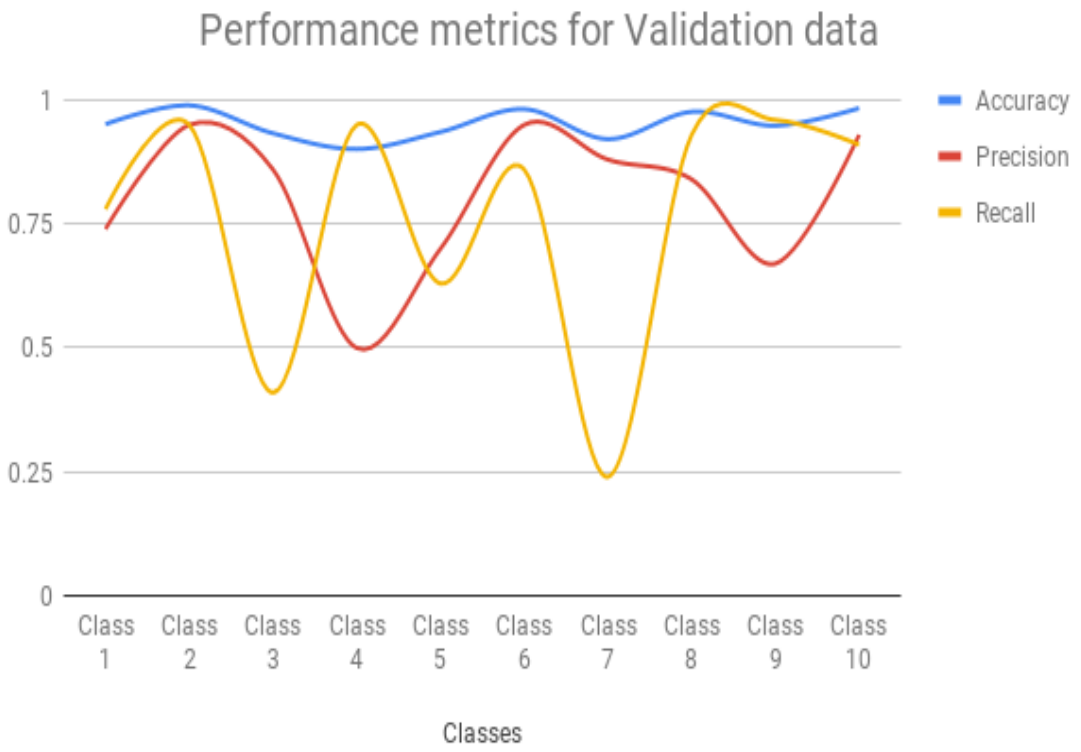

Figure 5: Plot of performance metrics Vs classes Validation

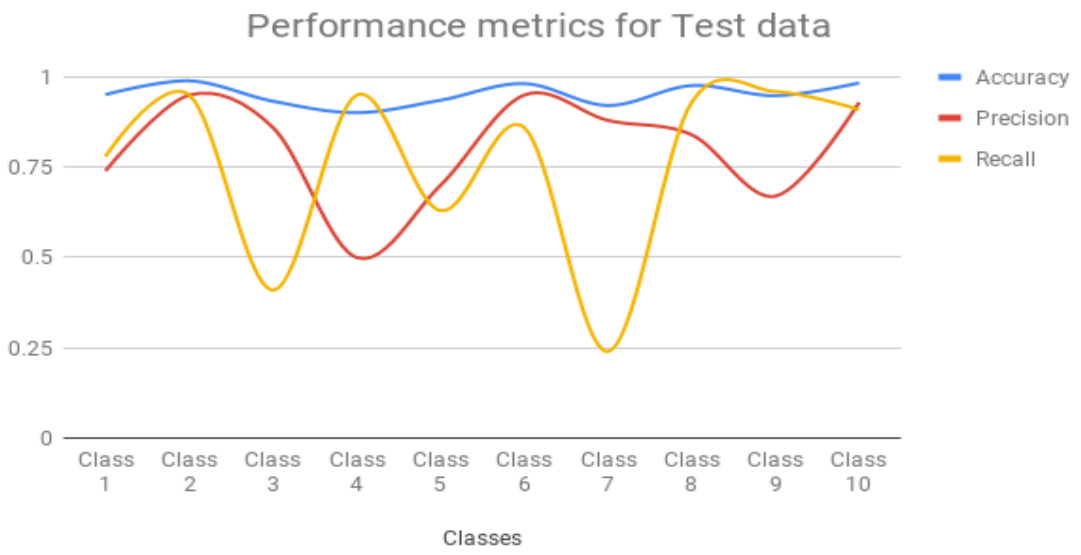

Figure 6: Plot of performance metrics Vs classes Testing

Training data confusion matrix				
	True Positive	False Positive	False Negative	True Negative
Class 1	1864	652	514	20970
Class 2	2259	131	121	21489
Class 3	1000	160	1449	21391
Class 4	2275	2250	121	19354
Class 5	1509	641	902	20948
Class 6	2095	109	329	21467
Class 7	561	80	1808	21551
Class 8	2147	396	170	21287
Class 9	2283	1135	106	20476
Class 10	2274	179	213	21334

Validation data confusion matrix				
	True Positive	False Positive	False Negative	True Negative
Class 1	487	171	146	5196
Class 2	532	41	44	5383
Class 3	239	49	332	5380
Class 4	577	604	29	4790
Class 5	370	135	248	5247
Class 6	533	41	71	5355
Class 7	142	24	456	5378
Class 8	534	85	44	5337
Class 9	584	268	29	5119
Class 10	546	38	57	5359

Test data confusion matrix				
	True Positive	False Positive	False Negative	True Negative
Class 1	136	50	42	1772
Class 2	181	9	10	1800
Class 3	86	20	124	1770
Class 4	185	191	6	1618
Class 5	129	64	83	1724
Class 6	177	6	37	1780
Class 7	46	7	154	1793
Class 8	188	27	10	1775
Class 9	208	91	11	1690
Class 10	176	23	11	1790

Figure 7: Confusion Metric for all classes for training, validation and testing

3.2.3 KNN Experimental Results at class level & for various K:

We select several values of nearest neighbours i.e. K and observe the performance of our classifier. We then select the K for which the classifier has highest accuracy. In this regard,

we select the K as 6. Figure 8 shows the plot of Accuracy vs K. It can be seen that for K=6 the accuracy is highest. Figure 9 shows the plot of time taken for various values of K.

Table 3: Accuracy and time for KNN (Over K) (in %)

K value	3	4	5	6	7	8	9	10
Accuracy (%)	85.38	85.70	85.53	85.77	85.72	85.37	85.43	85.58
Time (mins)	12.73	13.12	12.36	12.45	13.50	13.31	12.77	13.59

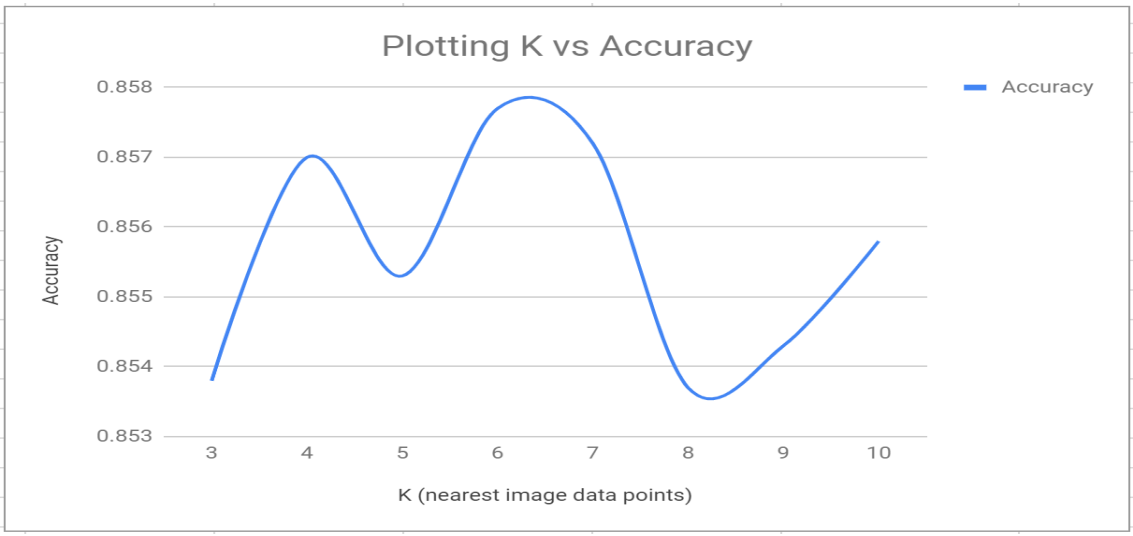

Figure 8: Accuracy of KNN over different K

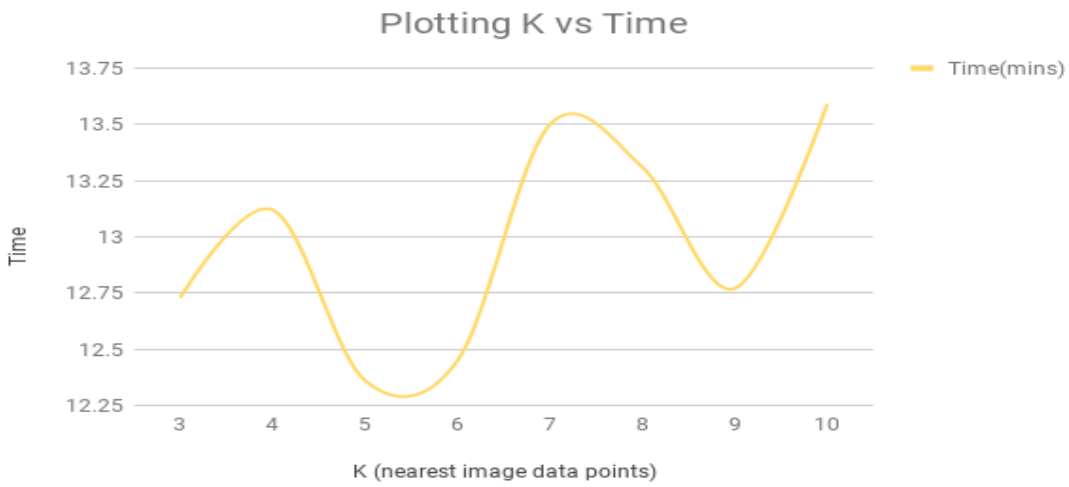

Figure 9: KNN: Time taken over different K

Figure 10: KNN: Plot of performance metrics Vs classes Training

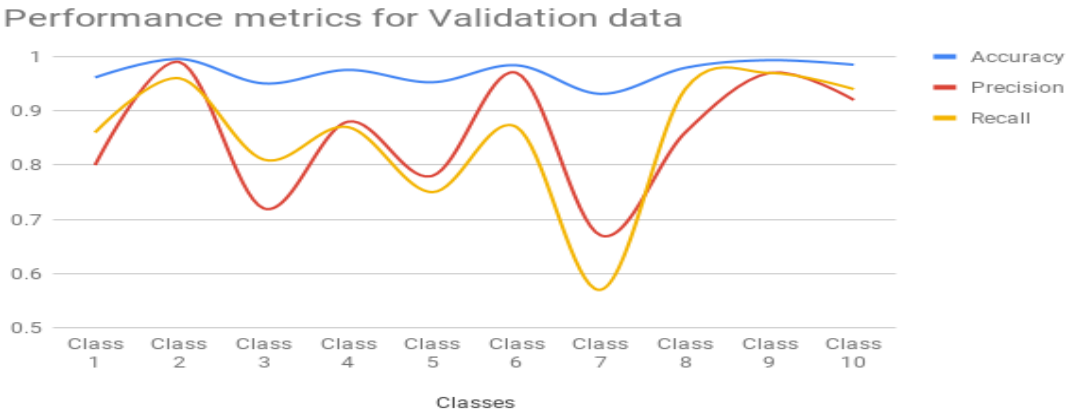

Figure 11: KNN: Plot of performance metrics Vs classes Validation

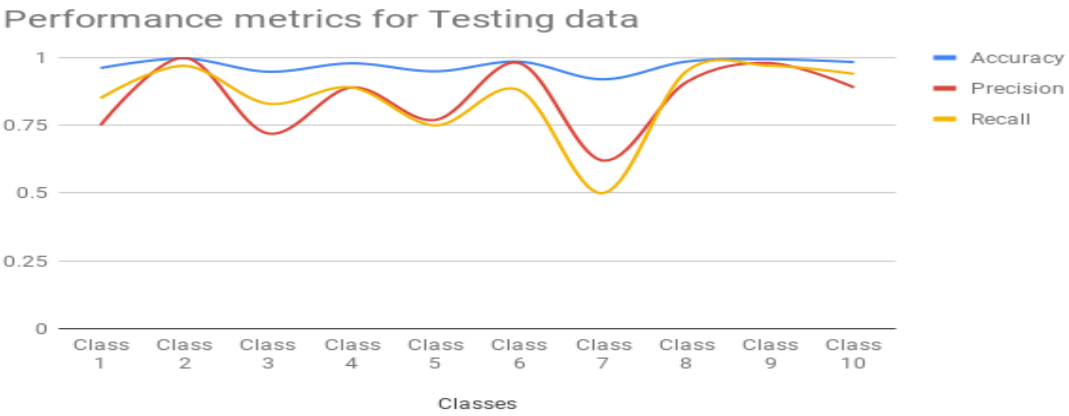

Figure 12: KNN: Plot of performance metrics Vs classes Testing

We can observe from the figure 10, 11 and 12 that the learning over some of the classes like class 3, class 5 and class 7 has been confused. The learning has not been very discriminative. This is in sync with the observations from LR classifier too. One can say that the classes 3, 5 and 7 are hybrid and are not very discriminative from other classes. On the other hand class 1, 6, 8, 9 and 10 are very distinct from other classes. This ensures good predictive capability over these classes. The confusion matrix over all the classes for k-NN classifier is show in the figure 13.

Training data confusion matrix

	True Positive	False Positive	False Negative	True Negative
Class 1	2141	955	290	20614
Class 2	2222	183	115	21480
Class 3	1755	773	665	20807
Class 4	2111	831	286	20772
Class 5	1471	525	946	21058
Class 6	2111	141	261	21487
Class 7	733	161	1653	21453
Class 8	2100	324	203	21373
Class 9	2340	644	83	20933
Class 10	2306	173	208	21313

Validation data confusion matrix

	True Positive	False Positive	False Negative	True Negative
Class 1	512	242	68	5178
Class 2	585	39	34	5342
Class 3	441	194	159	5206
Class 4	543	234	62	5161
Class 5	346	98	266	5290
Class 6	573	36	83	5308
Class 7	214	35	367	5384
Class 8	539	82	53	5326
Class 9	548	154	31	5267
Class 10	529	56	47	5368

Test data confusion matrix

	True Positive	False Positive	False Negative	True Negative
Class 1	156	76	22	1746
Class 2	184	9	7	1800
Class 3	142	66	68	1724
Class 4	174	86	17	1723
Class 5	123	50	89	1738
Class 6	180	9	34	1777
Class 7	62	14	138	1786
Class 8	184	21	14	1781
Class 9	207	54	12	1727
Class 10	176	27	11	1786

Figure 13: KNN: Table showing Confusion Matrix for all the classes

3.3 Discussion

The results for the two classifiers Logistic Regression and k- Nearest Neighbour has shown some interesting findings. First, the accuracy of both the classifiers over the classes follows the same trend. The accuracy of both classifiers drops for class 3, class 5 and class 7. On inspecting the data set we find that these classes especially class 7 is not distinctive. This leads to confusion in the classifier as its not able to discriminate between the given class and other classes. On the other hand class 2, 4, 8 and 9 are very discriminative. Both the classifier performs really well for them with little confusion. On inspecting the data set we find that these classes are remarkably different from other classes.

We also observe a higher accuracy in the case of k-NN compared to LR. This can be attributed to the exploitation of similarity metric in k-NN. Its very likely that similar elements will be located nearby. This leads to high performance of k-NN. Also, the binary nature of LR leads to using One vs All Methodology for training. As a result, 10 classifiers are being built. This prevents the LR to learn between the classes i.e. when we learn for class 1 we consider all the other classes as same. This leads to a drop in performance of LR in multi-class framework.

We plot the root mean square error for LR classifier over training, testing and validation data. We observe that all the three data set follows a similar trend for error. Hence, there is a little space for overfitting. However to make our LR classifier robust, we selected a suitable lamda for which the accuracy was high ($= -50$). We also implement gradient descent algorithm to achieve the solution. We take 200 steps to find the optimum solution. Furthermore, we carried out validation three times and build model each time, finally taking a mean of the all the three models. This helps us reduce variance in our model and makes it more robust.

In the case of KNN classifier we observe the classifier's performance over different K. We find that the classifier performs best when $K = 6$.

4 Conclusions

Here we have attempted to implement two classifiers namely, Logisitic Regression and k Nearest Neighbour. As expected both of the classifiers provided similar trends over the dataset. The performance of both the classifier dropped for class 3, 5 and 7 whereas both performed well for remaining classes. The LR classifier was optimised by implementing gradient descent algorithm. Also a regularisation term was added to the cost function so as to prevent overfitting. The k-NN classifier was fine tuned with different nearest neighbours (K) and it was observed that it performed best for k=6. Both the classifiers differed over accuracy. k-NN classifier had higher accuracy than the Logistic Regression classifier. We also observed that multiple validation (3 times) and taking mean of all the models provided us with better results.

The main challenge encountered was the limitation of computational power. The Logistic Regression classifier was trained over different sets of training data randomly sampled from the given dataset. However this increased the time taken and therefore we had to limit ourselves to 3 iterations. Also the selection of validation data and training data from original

dataset was challenging. We decided to take 20% of the data as validation data and rest as training data which is ideal in most of the cases. The overall performance of both the classifier were satisfactory and the accuracy achieved over various classes was good. Performance metrics like precision, recall also performed well for most of the classes which provides us with an optimum grey-scale image classifiers.

References

[1] Dr. Tongliang Liu, *Logistic Regression and SVM*, COMP 5318: Lecture notes/

[2] Zhang, Nikhil Kumar, *Understanding Logistic Regression*, Geeksforgeeks. https://www.geeksforgeeks.org/understanding-logistic-regression/

[3] Alexey Grigorev, *One-vs-All Classification*, ML Wiki http://mlwiki.org/index.php/One-vs-All_Classification

[4] Andrew Ng, *Supervised learning for Gradient Descent and cost function*, Stanford notes, http://cs229.stanford.edu/notes/cs229-notes1.pdf.

[5] Akshay Padmanabha, Christopher Williams, *K-nearest Neighbors*, https://brilliant.org/wiki/k-nearest-neighbors/

[6] Ahmad Basheer Hassanat, *Solving the Problem of the K Parameter in the KNN Classifier Using an Ensemble Learning Approach* , International Journal of Computer Science and Information Security, https://arxiv.org/ftp/arxiv/papers/1409/1409.0919.pdf