Implementation of K Nearest Neighbors with cross validation and sliding window on MNIST dataset

Rajat Jain(PID- 4446187) Author[1]

Abstract— **One of the oldest and simplest classifiers is the k-Nearest Neighbors (k-NN). In this assignment we will program this method and apply it to handwritten digit recognition. The dataset being used is MNIST, which is one of the most popular datasets in machine learning community.**

I. INTRODUCTION

KNN Algorithm : K nearest neighbor algorithm is a simple and lazy algorithm which works based on the minimum distance between the training samples and the query instance to find the K-nearest neighbors. In this assignment we will be using euclidean distance function to calculate distance between images.

Cross Validation : sometimes called rotation estimation, or out-of-sample testing is any of various similar model validation techniques for assessing how the results of a statistical analysis will generalize to an independent data set. In this assignment, we are doing 10 cross fold validation.

Sliding Window:we use training data which is well cropped data, and we expect to get good accuracy on test data, which may not be as well cropped as our training set. Sliding window comes here for rescue. In this assignment we converted our 28*28 image to 30*30 and again cropped it to get 9 new images, which are slightly shifting in every direction.

II. KNN FOR K=1

A. Pseudo Code

```
Classify(X,Y,x) //Here X is training data, Y is class labels, x is unknown sample
for i=1 to m do
    ComputeDistance d(Xi,x)
endfor
    Compute set I containing indices for k smallest distances d(Xi,x)
return majority label for Yi , where i is element of I
```

B. Explaination and result of implementation

1) Convert the datasets into CSV file and read the CSV file.
2) Separate the images and labels from csv file
3) Calculate the Euclidean distance using the formula sqrt(sum(square(a-b)))=a2+b2-2(a.b), using matrix manipulation for calculating euclidean distance for all testing images with training images.
4) Find the indices of k nearest neighbors using the distance obtained from calculateEuclideanDistance function.
5) Use the indices to get the label from train labels.
6) Calculate accuracy using formula (sum(predictions==testLabel))/len(predictions)*100
7) **Accuracy of k=1 using knn is 96.91 percent**
8) **Execution time is 6.49 minutes**

III. 10 FOLD CROSS VALIDATION

A. Pseudo code

```
setSize= trainset/10
for i=1 to 10 do
    x= bucket[i] //contains 6k train images as test images
    X=bucket-bucket[i] //contains 54k training images
    Y=bucketLabel-bucketLabel[i]
//Above X is training data, Y is class labels, x is unknown sample
    for i=1 to m do
        ComputeDistance d(Xi,x)
    endfor
    for i=1 to 10 do // calculate indices for k=1 to k=10
        Compute set I containing indices for k smallest distances d(Xi,x)
    endfor
getNeighbors return majority label for Yi , where i is element of I
```

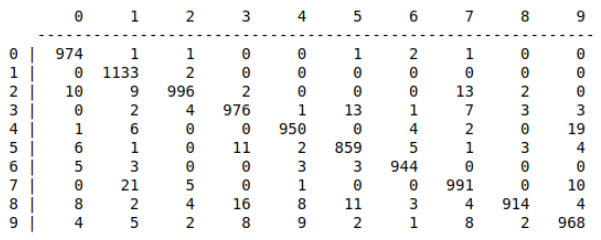
B. Explanation and result of implementation

Cross-validation is a method that is used to evaluate predictive models by dividing the original training sample into test set and training set. Training set is used to train the model and test set is to evaluate them. In K-fold cross validation, the sample is divided into k equal subsample, in which single subsample is taken as testing data and k-1 subsamples are taken as training data. The main advantage if this method is that all observations are used for both testing and training and each observation is used for testing exactly once.

1) Set the size of the subsamples by dividing them equally(6000 in our case).
2) Set one subsample(6000) as test set and remaining subsamples(54000) as training set and find the Euclidean distance.

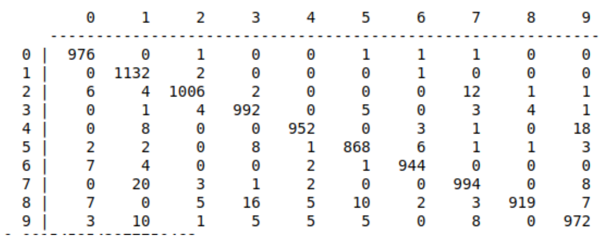
*This work is for ucf machine learning assignment

[1]Rajat Jain is student of Computer Science, University of central florida,Orlando, Florida bohrarajat@knights.ucf.edu



```
        0     1     2     3     4     5     6     7     8     9
     -----------------------------------------------------------
0 |   974     1     1     0     0     1     2     1     0     0
1 |     0  1133     2     0     0     0     0     0     0     0
2 |    10     9   996     2     0     0     0    13     2     0
3 |     0     2     4   976     1    13     1     7     3     3
4 |     1     6     0     0   950     0     4     2     0    19
5 |     6     1     0    11     2   859     5     1     3     4
6 |     5     3     0     0     3     3   944     0     0     0
7 |     0    21     5     0     1     0     0   991     0    10
8 |     8     2     4    16     8    11     3     4   914     4
9 |     4     5     2     8     9     2     1     8     2   968
```

Fig. 1. Confusion Matrix : Optimal K



```
        0     1     2     3     4     5     6     7     8     9
     -----------------------------------------------------------
0 |   976     0     1     0     0     1     1     1     0     0
1 |     0  1132     2     0     0     0     1     0     0     0
2 |     6     4  1006     2     0     0     0    12     1     1
3 |     0     1     4   992     0     5     0     3     4     1
4 |     0     8     0     0   952     0     3     1     0    18
5 |     2     2     0     8     1   868     6     1     1     3
6 |     7     4     0     0     2     1   944     0     0     0
7 |     0    20     3     1     2     0     0   994     0     8
8 |     7     0     5    16     5    10     2     3   919     7
9 |     3    10     1     5     5     5     0     8     0   972
```

Fig. 2. Confusion Matrix :Sliding Window

3) **Found Optimal k as 3 with accuracy of 97.116 percent**
4) **Execution time for cross validation was 70 minutes**

C. *Cross Validation Confusion Matrix and Confidence Interval for cross validation*

Confusion Matrix: is a matrix that is used to measure the performance of a machine learning algorithm. it used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known.
Results of confusion matrix is shown in Fig.1.
Confidence Interval for optimal k is 0.9678868438591223, 0.9744464894742112)

IV. Sliding Window Algorithm

The images are invariant to small translations and improve our distance metric by a method called sliding windows.
The sliding window is used to keep track of some function whose domain is the set of intervals and the functions value can be efficiently recomputed after one of the edges moves a small amount.

A. *Pseudo Code*

```
We are using k=3 as found optimal in cross validation.
Create empty matrix of euclidean distance
**for** i=1 **to** 60000 **do**
padTrainImg1=padding(train[i]) // padding images and converting it to 30*30
extract9Img=crop(padTrainTmag1)
**Classify(X,Y,x)** //Here X is 9 images of 1 training image, Y is class labels, x is unknown sample
**for** i=1 **to** m **do**ComputeDistance d(**Xi**,x)
EuclideanDistanceMatrix=CalculateMinmumDistance between 9 images and 10000 testing image
  **endfor**
Compute set I containing indices for k smallest distances d(**Xi**,x)
**return** majority label for **Yi** , where i is element of I
```

B. *Explanation and result of implementation*

1) I have create a zeros matrix of size 10000,60000 for keeping final euclidean matrix
2) Loop over 60000 images
3) Each image is converted into 30*30 size and than cropped it again to get 9 images
4) Calculated euclidean distance for those 9 images with 10000 test images and kept distance which is minimum
5) Inserted that euclidean distance in matrix which we initialized in start
6) After looping over 60k training images, we get final matrix of euclidean distance
7) After that we find the indices of all labels with k=3
8) Finally we predict it with training labels and calculate accuracy with test labels
9) **Accuracy after Window Sliding algorithm was 97.55**
10) **Execution time for sliding window is 120min**

C. *Confusion Matrix and Confidence Interval for sliding window*

Confusion Matrix: is a matrix that is used to measure the performance of a machine learning algorithm. it used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known.
Results of confusion matrix is shown in Fig.2.
Confidence Interval for optimal k is (0.9724699293803608, 0.978530070619639)

V. Significance Testing and Difference Rule

1) The difference between the population proportion of two knn classifiers (standard and sliding window is calculated.
2) The calculation accuracy of the standard is p0 and sliding windows is p1
3) Null hypothesis H0 is the hypothesis in which sample observations results purely from chance. H0 is true when H0:po-p1=0 , i.e. p0=p1
4) Alternative hypothesis H1 is the hypothesis in which the sample observations are not influenced by random cause, which is the exact opposite of null hypothesis.

In our case **Significance Testing:**

- p0=97.116 percent =0.97116 and p1=97.55 percent = 0.9755
- **diff = p1-p0 =0 // This is null hypothesis**
- **Alternate Hypothesis is p1!=p0**
- **Actual diff = p1-p0= 97.55-97.116 =0.0043399**——————————-equation 1

- AverageAccuracy a= (p1+p0)/2= 97.333 percent = 0.97333
- Standard Deviation SD=np.sqrt((a*(1-a))/20000) = 0.001139
- **So the range is -1.96*SD to 1.96*SD = (-0.0022329,0.0022329)**——————-equation 2

Difference Rule : The statement is that some classifier A is better than B if A's accuracy is at least 0.2 percent more than actually holds true. It is because of the fact that for null hypothesis with confidence of 95 percent we have z score of 1.96. Lets call it zn. Anything greater than this would cause the difference to be outside of 95 percent confidence interval.So,
zn=1.96————————equation 3
P(d¿ zn* sd)= alpha =0.05
where alpha nothing but 1-0.95=0.05
z = diff/sd =0.0043399/0.001139 =3.81———equation 4

Conclusion

1) From equation 1 and 2 : We found **difference to be outside the range**
2) From equation 3 and 4 : We found **z¿ zn i.e. our z value exceeds 1.96 threshold for 95 percent confidence interval**
3) **We can conclude Slide Window Method along knn is just better than KNN**