Handwritten Digit Recognition

Yin Ming

2017.4.20

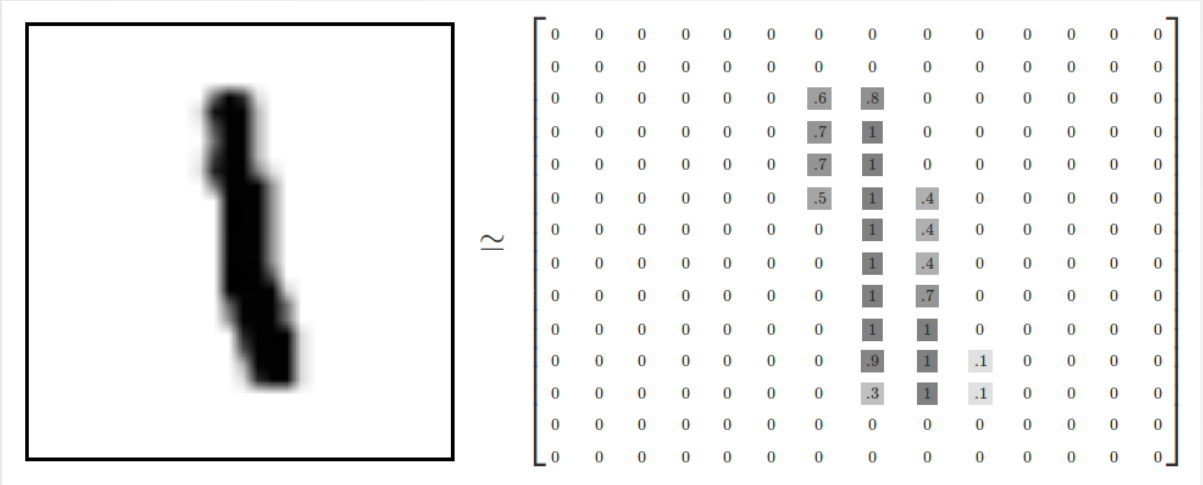
CONTENTS

1 Mnist Dataset --- Introduction

1. 70,000 handwritten digits, divided into 60,000 training examples and 10,000 testing examples.

2. Each example has a image and a label. The image is a matrix consisting of 28×28=784 values of pixels which range from 0 to 255. 0 means background white and 255 means black. Now Normalize the pixel values into [0, 1].

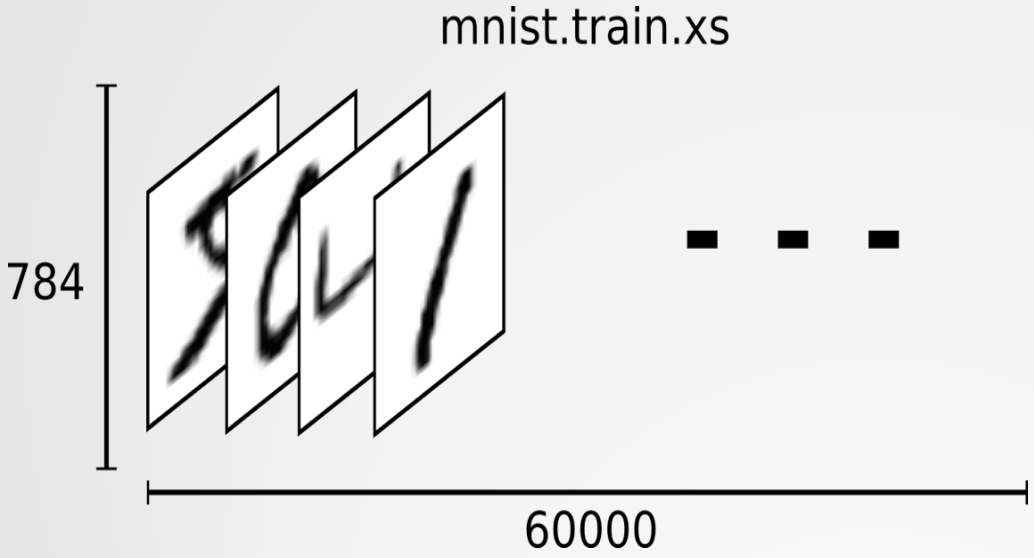
3. The label represents the number in corresponding image.



1 Mnist Dataset --- Read data

1/ The dataset file is stored in the relative file path --- datasets_dir

2/ The pixel values of training(test) image set are transformed into a matrix with 60,000(10,000) rows and 28×28=784 columns. The training labels are transformed into a matrix with 60,000(10,000) rows and 1 columns.



1 Mnist Dataset --- Read data

1. Check if the dataset is read correctly. For example, we select the first training image and the last testing image.

```
image = Xtrain[0].reshape(28,28)
image1 = Xtest[9999].reshape(28,28)
```

2 KNN Method

1/ K-Nearest Neighbors is an algorithm in which the best estimate among all the values is the value that has the maximum number of neighbors with smallest Euclidian or Hamming distance.

```
print('\nPreparing Classifier Training and Testing Data...')
Xtrain, Xtest, Ytrain, Ytest = cross_validation.train_test_split(Xtrain,Ytrain,test_size=0.1)
```

```
clf = KNeighborsClassifier(n_neighbors=5,algorithm='auto',n_jobs=10)
clf.fit(Xtrain,Ytrain)

with open('MNIST_KNN.pickle','wb') as f:
      pickle.dump(clf, f)

pickle_in = open('MNIST_KNN.pickle','rb')
clf = pickle.load(pickle_in)
```

2/

- Use cross validation to divide training data into training and testing data to train classifier.
- Provide training data and labels as input to train the classifier.
- The digit recognized using KNN is then matched with the provided training labels to get the accuracy of the trained classifier.
- The trained classifier is pickled to be used again on the testing data.

2 KNN Method

2/ The test image data is used to predict the labels of digits and is compared to the provided test labels to see for the accuracy of the algorithm.

```
KNN Classifier with n_neighbors = 5; algorithm = auto, n_jobs = 10

Pickling the Classifier for Future Use...

Calculating Accuracy of trained Classifier...

Making Predictions on Testing Data...

Calculating Accuracy of Predictions...

Calculating Confusion Matrix...

KNN Trained Classifier Accuracy: 0.978833333333

Predicted Values: [7 2 1 ..., 4 5 6]

Confusion Matrix:
[[ 974    1    1    0    0    0    2    1    1    0]
 [   0 1133    2    0    0    0    0    0    0    0]
 [  10   13  983    3    2    0    1   16    4    0]
 [   0    3    2  979    1   12    1    6    4    2]
 [   2    8    0    0  942    0    4    2    1   23]
 [   5    0    0   15    1  858    4    2    2    5]
 [   6    3    0    0    3    2  944    0    0    0]
 [   0   25    4    0    2    0    0  986    0   11]
 [   8    4    6   14    7   15    2    6  907    5]
 [   7    6    2    9    7    3    1   11    2  961]]

Accuracy of Classifier on Test Images: 0.9667
```

- KNN classifier with 5 neighbors
- Accuracy of the trained classifier is 97.88%
- Accuracy of the prediction (classifier on test images) is 96.67%
- The confusion matrix provides the percentage of accuracy with which each digit has been recognized.

3 SVM Classifier

- Use cross validation to divide the training data into training and testing data to train the classifier.
- Train the SVM Classifier. Provide training data and labels as input to train the classifier.
- The digit recognized using SVM is then matched with the provided Training Labels to get the accuracy of the trained classifier.
- This trained classifier is pickled to be used again on the testing data.
- The Test Image data is used to predict the labels of digits and is compared with the provided test labels to see for the accuracy of the algorithm.

```
SVM Classifier with gamma = 0.1; Kernel = polynomial

Pickling the Classifier for Future Use...

Calculating Accuracy of trained Classifier...

Making Predictions on Testing Data...

Calculating Accuracy of Predictions...

Calculating Confusion Matrix...

SVM Trained Classifier Accuracy: 0.999166666667

Predicted Values: [7 2 1 ..., 4 5 6]

Confusion Matrix:
[[ 972    0    1    1    0    3    1    0    2    0]
 [   0 1126    2    2    0    0    4    0    1    0]
 [   8    2 1007    0    2    0    4    5    4    0]
 [   0    2    1  987    0    6    0    4    5    5]
 [   3    0    1    0  966    0    3    0    0    9]
 [   2    0    0   12    1  864    4    1    6    2]
 [   4    5    1    0    3    5  938    0    2    0]
 [   0    9    9    1    1    0    0 1002    0    6]
 [   5    0    1    2    3    5    1    4  951    2]
 [   3    6    1    4    8    3    1    2    3  978]]

Accuracy of Classifier on Test Images: 0.9791
```

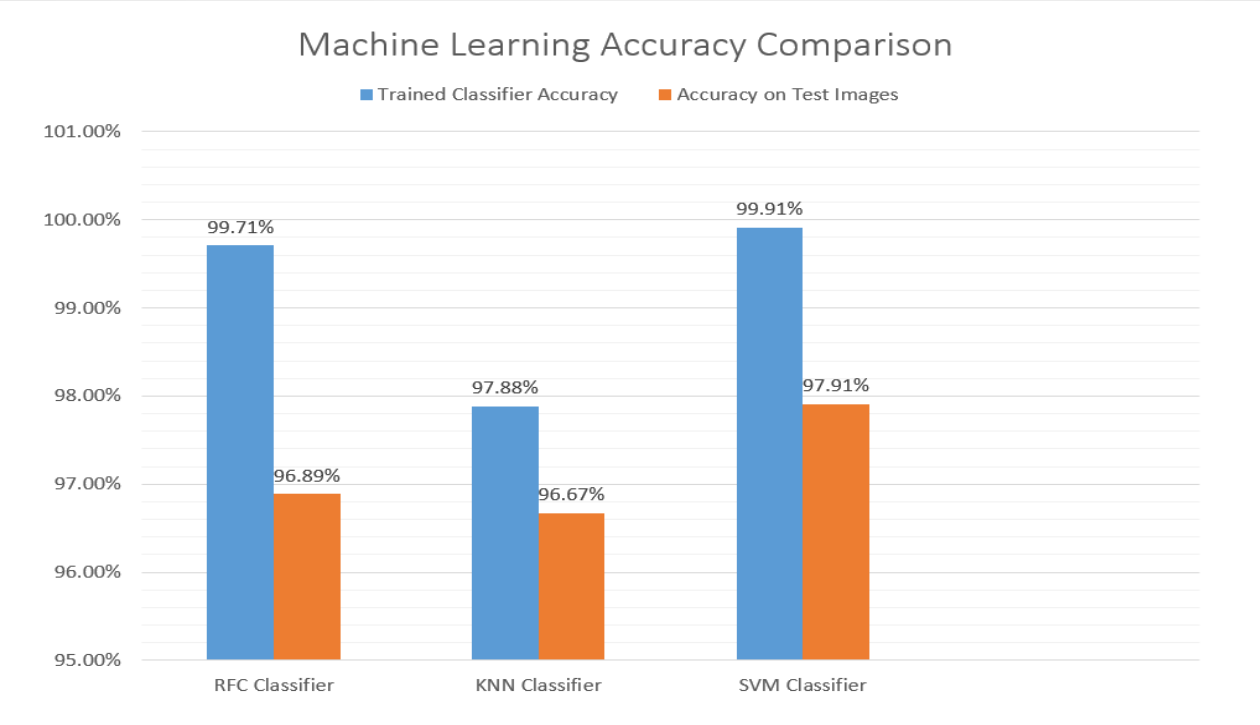
4 Random Forest Classifier

- Use cross validation to divide the training data into training and testing data to train the classifier.
- Provide training data and labels as input to train the classifier. RFC requires the number of trees in forest, number of features to look for best split, maximum depth of the tree etc. as the input.
- The digit recognized using RFC is then matched with the provided Training Labels to get the accuracy of the trained classifier.
- This trained classifier is pickled to be used again on the testing data.
- The Test Image data is used to predict the labels of digits and is compared with the provided test labels to see for the accuracy of the algorithm.

```
RFC Classifier with n_estimators = 100; n_jobs = 10

Pickling the Classifier for Future Use...

Calculating Accuracy of trained Classifier...

Making Predictions on Testing Data...

Calculating Accuracy of Predictions...

Calculating Confusion Matrix...

RFC Trained Classifier Accuracy: 0.997166666667

Predicted Values: [7 2 1 ..., 4 5 6]

Confusion Matrix:
[[ 972    0    0    0    0    2    2    1    3    0]
 [   0 1122    2    4    2    1    3    0    1    0]
 [   6    0  993    5    5    1    4   11    7    0]
 [   0    0   10  966    0   11    0    9   11    3]
 [   1    0    1    0  956    0    4    0    3   17]
 [   3    0    1   10    1  862    4    2    5    4]
 [   7    3    0    0    3    3  938    0    4    0]
 [   1    6   19    1    1    0    0  986    4   10]
 [   4    0    4   10    4    4    5    5  931    7]
 [   6    6    2   11   11    1    1    4    4  963]]

Accuracy of Classifier on Test Images: 0.9689
```

5 Comparisons

Percent Accuracy of Each Classification Technique			
	KNN	SVM	RFC
Trained Classifier Accuracy	97.88%	99.91%	99.71%
Accuracy on Test Images	96.67%	97.91%	96.89%



Possible Improvements

1. The visualization of results (e.g. confusion matrix) is not finished well.

2. Training and testing time of each method can also be measured. Aspects of comparisons will be multiple.

3. Some errors still exist in CNN code.

Thank you!

Yin Ming

2017.4.20