

Applied Machine Learning

assignment 1

Yihan Chen

Programming Exercise

1. Digit Recognizer

1.1 Dataset Description

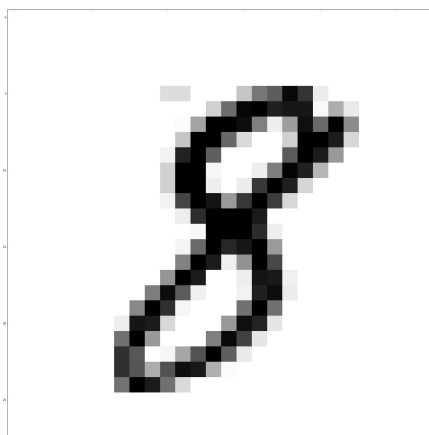
In this exercise, I'm given the data of Digit Recognizer on Kaggle. The data files train.csv and test.csv contain gray-scale images of hand-drawn digits, from zero through nine. The training data set, (train.csv), has 785 columns. The first column, called "label", is the digit that was drawn by the user. The rest of the columns contain the pixel-values of the associated image.

One of the tasks is to calculate the distances of each images of digit 0 and digit1 and generate ROC curve based on the calculated results.

The other task is to implement an KNN classifier and and apply cross-validation test to examine its validity and draw confusion matrix.

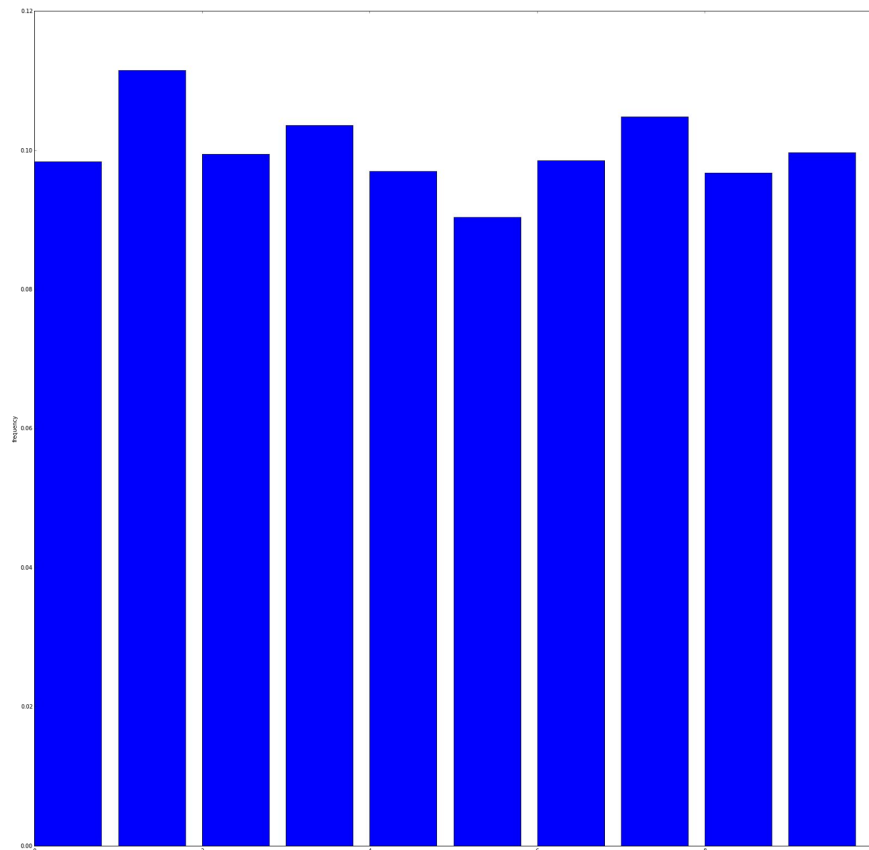
1.2 Display the images

To display the images from pixel data, I need to extract each row from the dataset, reshape it into 28*28 size from which the image is constructed. And then I use plt.imshow() function to display the image.



1.3 Examine the Prior Probability

The prior probability of each digits can be calculated as their (frequency)/(frequency of all). Generated a histogram where the x-axis measured by classes and y-axis measured by prior probability.



According to the histogram, the images are mostly even distributed in each each digits.

1.4 L2 Best Match

Pick up the first 10 examples of each digit in the training set, and use L2 distance to measure the similarity between the example and the rest of training data. The result of pairing is as below:

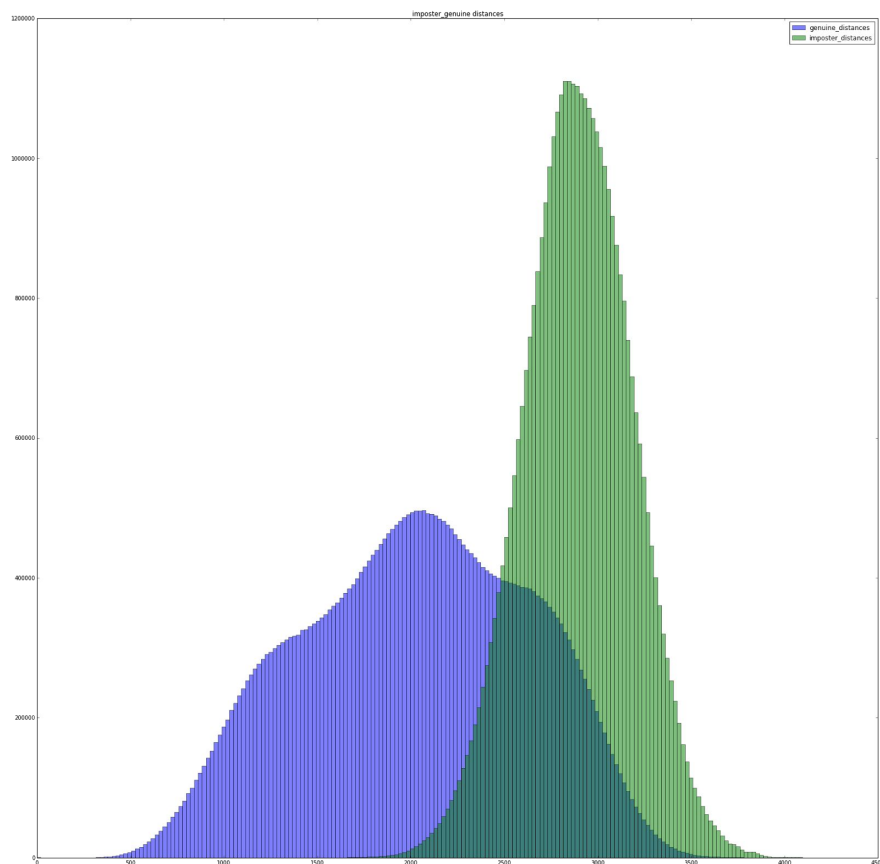
examples	best_match
1	1
0	0
4	4
7	7

3	5
5	5
8	8
9	9
2	2
6	6

Actually, I get a good result by simply calculate the distances between each examples and the training data. From the table we can see, there is a mismatch between digit 3 and digit 5. That make sense because the lower part of 3 and 5 are similar.

1.5 0 and 1 Pairing

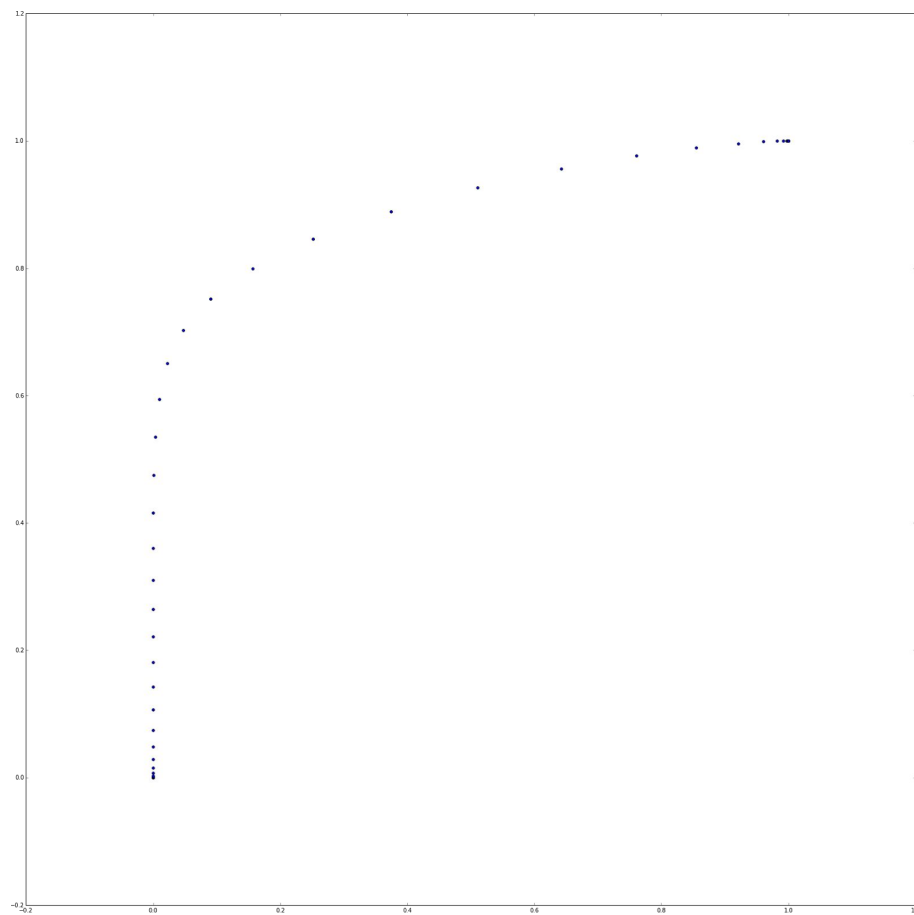
Pick up all the images that with label of 0 or 1. To calculate the distances between each 0 and 1, I imported euclidean distances function from sklearn.metrics.pairwise. And then generated a histogram to plot all the distances.



From the picture we can see, the imposter matches(green) and the genuine matches(blue) are basically under the normal distribution. And imposter distances are averagely larger than genuine distances which prove the validity of using L2 distances to classify.

1.6 ROC Curve

ROC curve is created by plotting true positive rate versus false negative rate. It can be used to illustrate the performance of binary classification. In this case, to generate ROC curve from the imposter and genuine distances, I set up several thresholds on the distances. For every threshold, I assume the distances smaller than threshold to be genuine pair and the other side to be imposter and therefore generate the TPR and FPR of every threshold. The trajectory of the ROC curve is like this



1.7 KNN - Classifier

There are three steps to implement KNN classifier: 1) calculate the distances 2) choose the k nearest neighbors 3) k-nearest neighbors voting. Basically, KNN need to calculate the distances of every two examples. In this case, there are 42000 elements, so the calculation would be 42000×28000 which take a really long time to run the code.

```

def findminK(k,a):
    a_sorted = np.argsort(a,axis=0)
    # print(a_sorted)
    k_min = a_sorted[0:k].flatten(|)
    # print(k_min)
    return k_min
#findminK(3,[1,2,5,4,9,0])
def voting(k_nbs,l):
    k_l = []
    # print(k_nbs[1])
    for i in range(0,len(k_nbs)):
        t=l[k_nbs[i]]
        k_l.append(t)
    return max(set(k_l), key=k_l.count)
#voting(np.arange(0,9))
#print(label[0:9])
def KNN(k_train,k_labels,k_test,k):
    results = []
    for i in range(0,k_test.shape[0]):
        distances = []
        for j in range(0,k_train.shape[0]):
            ds = euclidean_distances(k_test[i],k_train[j])
            distances.extend(ds)
        # print(distances)
        k_nbs=findminK(k,distances)
        # print(k_nbs)
        results.append(voting(k_nbs,k_labels))
        # print(results)
    return results
#KNN(train,label,test,5)

```

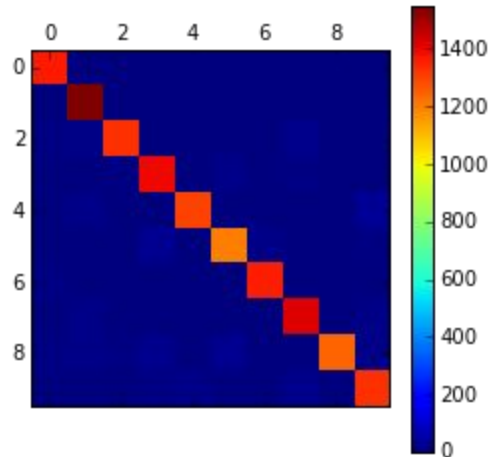
So I try the KNeighborsClassifier in sklearn.neighbors. Which only take about 10mins with a come out of 96% accuracy on Kaggle(The name of submission is yeehan). The reason sklearn being much faster than basic KNN is that it implement kd_tree algorithm in calculation, which will significantly reduce the times of distances calculation.

1.8 3-fold cross_validation

Divided the training dataset into 3 parts, take out each part as test set, and use the rest to train data. The average accuracy is 96.5%.

1.9 Confusion Matrix

Based on the result of 3-fold cross-validation, I draw the confusion matrix.



From the picture, 5 and 8 are the most tricky digits to classify.

2. The Titanic Disaster

2.1 Dataset Description

Kaggle provides a set of passenger data. The task is to predict whether a passenger will survive in the accident based on the passenger information. The point in this task is to select which features to use because not all features will contribute to the classification performance.

2.2 Logistic Regression

Before running the data into logistic regression, I need to preprocess the data to make it meet the requirement of model.

I used the logistic regression model in `sklearn.linear_model` as the classifier. Since this model can't deal with string data, I need to transform the data into float type.

When it comes to feature selection, based on analysis, I eliminated the attributes like name, id, ticket number, price which are not relative to the survival of a person. The attribute cabin should be relative to the classification, however there are too many missing values in the column which make prediction being difficult. One way to deal with the missing data is to use data we have to predict the missing value. But the problem is, there are more missing data than provided data. So it's not appropriate to predict the missing value based on given value. I choose to delete it.

One more thing to mention, age data is believed to be linked with people's survival. However, the minus age difference like 21 and 23 should not make significant difference. So I group age into 3 groups: below 15, 15 to 50, above 50.

2.3 Submission

I submitted my data to kaggle with the name Yeehan, the score is 77.033%.

Writing Exercise

Exe 1.

$$\begin{aligned} E[X+Y] &= E[(X+Y)^2] + (E[X+Y])^2 \\ &= E[X^2 + 2XY + Y^2] + E[X]^2 + 2E[X]E[Y] + E[Y]^2 \\ &= E[X^2] - (E[X])^2 + E[Y^2] - (E[Y])^2 + 2E[XY] + 2E[X]E[Y] \\ &= V[X] + V[Y] + 2Cov[X, Y] \end{aligned}$$

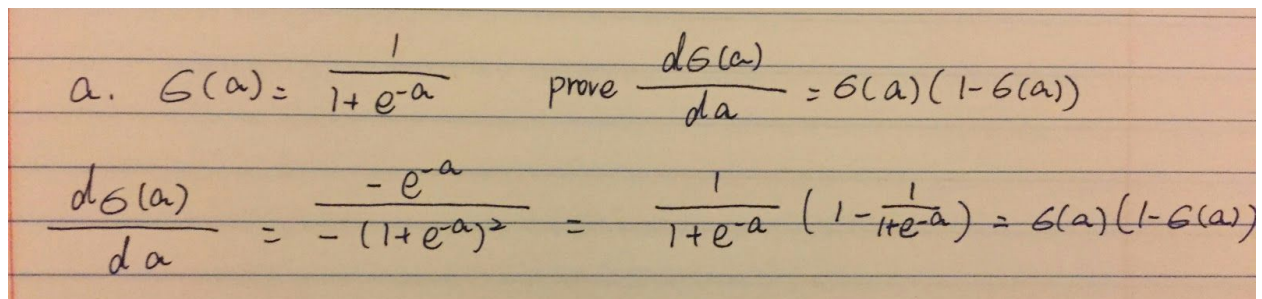
Exe 2.

In this question, the Prior Probability $\Pr(B)$ is the prob to get disease among population, which is $1/10000$. And $\Pr(A)$ is the prob of being tested positive.

So the $\Pr(B) = (1/10000 * 99\%) / ((1/10000)*99\% + (9999/10000)*1\%) = 0.98\%$

Exe 3.

a)



Handwritten mathematical derivation for Exe 3a:

$$\begin{aligned} \text{a. } G(a) &= \frac{1}{1+e^{-a}} \quad \text{prove } \frac{dG(a)}{da} = G(a)(1-G(a)) \\ \frac{dG(a)}{da} &= \frac{-e^{-a}}{(1+e^{-a})^2} = \frac{1}{1+e^{-a}} \left(1 - \frac{1}{1+e^{-a}}\right) = G(a)(1-G(a)) \end{aligned}$$

b)

$$\begin{aligned}
 \frac{\partial l(\beta)}{\partial \beta \partial \beta^T} &= \frac{\partial \left(\sum_{i=1}^N x_i (y_i - \sigma(\beta^T x_i)) \right)}{\partial \beta^T} \\
 &= \sum_{i=1}^N \frac{-x_i (\partial \sigma(\beta^T x_i))}{\partial \beta^T} \\
 &= \sum_{i=1}^N -x_i x_i^T \sigma'(\beta^T x_i) (1 - \sigma(\beta^T x_i)) \\
 &= \sum_{i=1}^N -x_i x_i^T p(x_i; \beta) (1 - p(x_i; \beta))
 \end{aligned}$$

c)

$$\begin{aligned}
 X^T W X &= \sum_{i=1}^N x_i x_i^T p(x_i; \beta) (1 - p(x_i; \beta)) \\
 \text{prove: } \forall U \in n \times 1 &\Rightarrow U(X^T W X)U^T > 0. \\
 U(X^T W X)U^T &= \cancel{U(X^T X)U^T} \sum_{i=1}^N U x x^T U^T p(x_i; \beta) (1 - p(x_i; \beta))
 \end{aligned}$$

Since the inner product of matrix X and U are positive and the prob is always positive, we can see the product of this matrix and any matrix U is always positive. Hence, this matrix is positive definite.