**Header:**

My Name: Qi Zhao

Major resources: A laptop, java libraries, MNIST Dataset

Programming Language: java.

**Dataset details:**

I choose some examples to show what the data look like:



The total 42000 data set are divided into three parts:

Validation: 1~4000 (9.5%)

Test: 4001~8000(9.5%)

Train set: 8000~42000(81%)

**Algorithm description**

K-NN

No scaling.

Characteristics of this algorithms:

1. First, import the training data, create an “example” class for each line of data. Use an arrayList to store the example objects.
2. Use a priority queue to store the k nearest points away from the test point.

**Algorithm results:**

Confusion Matrix:

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Actual  Predict | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | total |
| 0 | 398 | 0 | 2 | 1 | 0 | 0 | 2 | 0 | 2 | 1 | 406 |
| 1 | 0 | 440 | 10 | 2 | 3 | 1 | 0 | 7 | 6 | 1 | 470 |
| 2 | 0 | 1 | 403 | 2 | 0 | 0 | 0 | 0 | 2 | 1 | 409 |
| 3 | 0 | 0 | 1 | 358 | 0 | 5 | 0 | 0 | 12 | 4 | 380 |
| 4 | 0 | 0 | 0 | 0 | 376 | 0 | 1 | 0 | 0 | 4 | 381 |
| 5 | 0 | 0 | 1 | 3 | 0 | 365 | 0 | 0 | 5 | 1 | 375 |
| 6 | 1 | 2 | 0 | 0 | 0 | 4 | 410 | 0 | 3 | 1 | 421 |
| 7 | 1 | 1 | 8 | 2 | 1 | 0 | 0 | 401 | 0 | 7 | 421 |
| 8 | 0 | 1 | 4 | 1 | 0 | 0 | 0 | 0 | 348 | 0 | 354 |
| 9 | 0 | 1 | 1 | 1 | 9 | 4 | 0 | 1 | 5 | 361 | 383 |
| total | 400 | 446 | 430 | 370 | 389 | 379 | 413 | 409 | 383 | 381 | 4000 |

Choose number 9 as the measure number.

For number 9, the accuracy of the classifier: k = 10: accuracy: 361/381 = 94.75%;

K = 1 : accuracy = 360/381 = 94.49%

K =2: accuracy = 340/381 = 89.24%

K =4: accuracy = 358/381 = 93.96%

K=8: accuracy = 361/381 = 94.75%

K=16: accuracy = 358/381 = 93.96%

K =32: accuracy = 354/381 = 92.91%

K=64: accuracy = 351/381 = 92.13%

When the k equals 10, the accuracy is highest. So, let’s set the k as 10.

**Run Time**

**Analysis**

First, convert the dataset from line to line to an arraylist that store the “example” objects. O (m \* n).

Then, for each test case, iterate the whole arraylist and find the K nearest point. O(m\*n logk).

Finally, we assume the amount of test case is t;

So, the total time complexity is O(t \* m \* n \* logk).

**Actual:**

The size of test set is 4000 (divided from the train set), k =10; run time = 2.6 min

The size of the test set( without label) is 28000, run time = 32.28 min