Machine Problem #1
COT 5610 - Machine Learning
Fall 2014
Due 09-02-2014

Edward Aymerich

1 Problem description

A k-Nearest Neighbors (KNN) algorithm was implemented and applied to the MNIST data set. This data set consist of 60,000 training examples of handwritten digits, and 10,000 test examples.

In order to test the KNN algorithm, three test protocols were used:

- Test Protocol 1: using all 50,000 training examples from training set for training, and all 10,000 test examples from test set for testing. Different values of k were used to test the impact of k on performance and running time.
- Test Protocol 2: the training set was divided into a training subset (with 50,000 examples), and a validation set (with 10,000 examples). The validation set was used to find the optimal value for k in the range [1-10], and this value was used to test the model against the original test set.
- Test Protocol 3: only the training set was used in this protocol. The KNN algorithm was tested using 5-fold cross-validation and 10-fold cross-validation on the training set.

2 KNN implementation

The KNN algorithm was implemented on C++11, mainly due to the easy use of threads in this standard.

The Euclidean distance was the function used to estimate the distance between examples.

To find the closest neighbors, an exhaustive search was done over all the examples in the training set. For each test example, the distance to each training example is calculated, and then this distance is compared with the current k nearest neighbors already found. If the training set is composed of N examples, at worst the complexity to predict one test example will be O(k*N). In practice, only a few times a new calculated distance has to be compared with all nearest neighbors, because usually the current training example won't be a close neighbor of the test sample, so the value of k has little impact on the overall running time of the experiments, as will be shown on the next section.

3 Results

This section presents the error rates resulting from applying the implemented KNN algorithm on the MNIST data set, using the test protocols described on the Problem Description. The implemented KNN algorithm was compiled using GCC 4.9.1 on a Windows 7 64 bit platform, and the experiments were executed using a Intel Core i5 3470 processor running at 3.2GHz.

3.1 Test Protocol 1

The error rates for different values of k under Test Protocol 1 can be observed on figure 1.

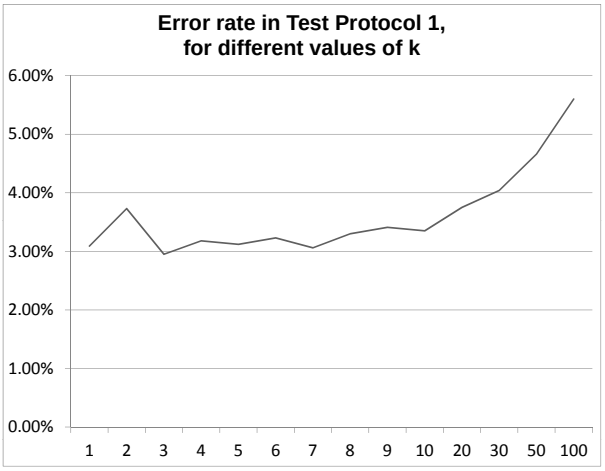

Figure 1: Error rate for Test Protocol 1.

The error rate reaches a minimum of 2.95% when k=3. For values of k from 1 to 7 the error rate seems unstable, having local minimums at 1, 3, 5 and 7.

The error rate is slightly higher when k is even, which could be due to the difficulty in reaching a consensus among the closest neighbors if there is a draw. After k=10, the error rate keeps growing, possibly because for a higher value of k more neighbors are selected which aren't that close to the test case. Another way to interpret this is that with higher values of k the neighborhood of the test example gets bigger, and so is the probability of including wrong neighbors (the ones with an label that doesn't match the label on the test example) in that neighborhood.

As for execution time, it doesn't change much with different values of k, running on average for 143.7 seconds with a standard deviation of 0.4 seconds. All the results obtained from Test Protocol 1 are shown on table 1.

k	error rate (%)	execution time (s)
1	3.09	144.96
2	3.73	143.98
3	2.95	143.46
4	3.18	143.81
5	3.12	143.61
6	3.23	143.68
7	3.06	143.56
8	3.30	143.67
9	3.41	143.49
10	3.35	143.47
20	3.75	143.36
30	4.04	143.63
50	4.66	143.41
100	5.60	143.71

Table 1: Results obtained from Test Protocol 1.

3.2 Test Protocol 2

In Test Protocol 2, first we tune the value of parameter k using a validation set, composed of 10,000 examples extracted from the original training set. The remaining 50,000 examples are used as the training subset to tune k.

The error rates for different values of k under Test Protocol 2 can be observed on figure 2.

On the validation set, the error rate reaches a minimum value of 2.74% when k=4. This value of k was later used on the original test set, obtaining an error rate of 3.42%.

Similar to Test Protocol 1, the execution time have a very small variance when k was changed. On average, the execution time was 119.54 seconds, with a standard deviation of 0.13 seconds. The smaller execution time, compared to Test Protocol 1, is easily explained because the subset used for training in Test Protocol 2 is 17% smaller than the training set used in Test Protocol 1 (50,000 compared to 60,000).

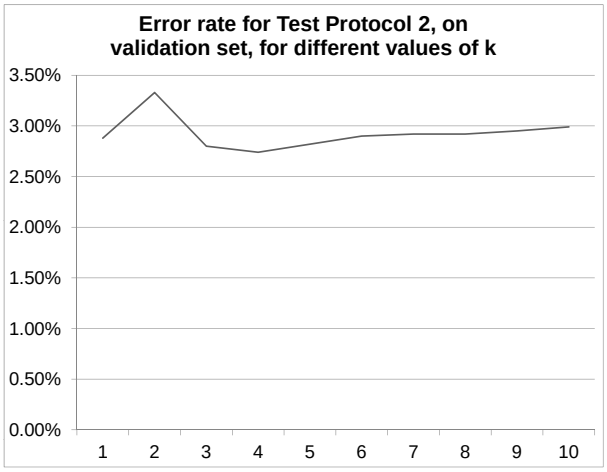

Figure 2: Error rate for Test Protocol 2.

All the results obtained from Test Protocol 2 are shown on table 2.

set	k	error rate (%)	execution time (s)
validation	1	2.88	119.50
	2	3.33	119.55
	3	2.80	119.67
	4	2.74	119.47
	5	2.82	119.78
	6	2.90	119.38
	7	2.92	119.42
	8	2.92	119.63
	9	2.95	119.41
	10	2.99	119.59
test	4	3.42	119.59

Table 2: Results obtained from Test Protocol 2.

3.3 Test Protocol 3

In test protocol 3, a 5-fold cross-validation and a 10-fold cross-validation was done over the original training set. The average error rates for both cross validations using different values of k can be observed on figure 3.

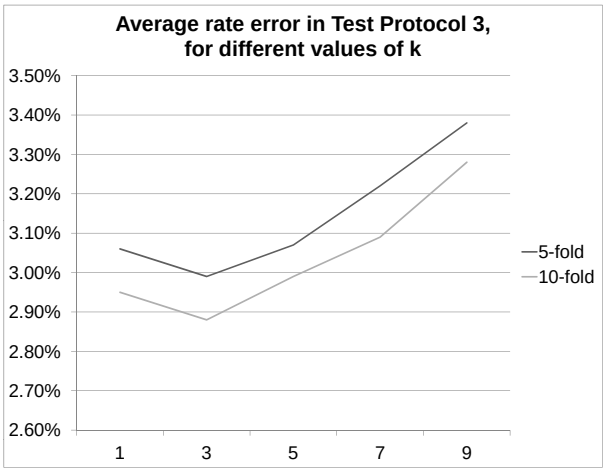

Figure 3: Error rate for Test Protocol 3.

Both cross-validations show a similar behavior, with the 10-fold cross-validation consistently archiving lower average error rates than the 5-fold cross-validation. For the 5-fold cross-validation, the minimum average error rate is 2.99% using k=3. For the 10-fold cross-validation, the minimum average error is lower, with a value of 2.88%, also using k=3.

Once again, the value of k seems to have very little effect on execution times. On average, the execution time for the 5-fold cross-validation was 689.96 seconds, with a standard deviation of 0.59 seconds. For the 10-fold cross-validation the execution time for the 5-fold cross-validation was 778.15 seconds, with a standard deviation of 2.51 seconds.

All the results obtained from Test Protocol 2 are shown on table 3.

test	k	average error rate (%)	execution time (s)
5-fold	1	3.06	690.11
	3	2.99	690.62
	5	3.07	689.26
	7	3.22	689.44
	9	3.38	690.38
10-fold	1	2.95	782.17
	3	2.88	777.63
	5	2.99	776.88
	7	3.09	775.51
	9	3.28	778.56

Table 3: Results obtained from Test Protocol 3.