CSE 250B | HOMEWORK 1 |

Prototype Selection for Nearest Neighbour

Gopal Rander (grander)

1. *A short, high level description of your idea for prototype selection.*

   Aim is to reduce the size of training set to its subset in such a way that this subset can be used for classification of test set without considerable fall in accuracy. The approach is similar to that of Condensed Nearest Neighbour(CNN) *Hart (1968)*.

   The idea is to start with a small random subset (size p) of training set, let's call it SST. We label every item in training set using 1-NN with only the items in SST. Then we incrementally add a fixed number(q) of randomly selected misclassified instances from training set to SST and repeat the above procedure until we have no misclassification in training set or we have filled SST to predefined upper bound (M). At every step, as we add only the misclassified elements so that in next run, it is guaranteed that all of those are classified correctly.

   The numbers p, q and M are defined and selected in advance.

2. *Concise and unambiguous pseudocode.*

```
Terms:
      1-NN: First Nearest Neighbour.
      TRS: Training set. Images and labels.
      TS: Test set. Images and labels.
      SST: Subset of Training set, to be populated at runtime.
      MCS: Set of images misclassified at runtime.

Given:
      p = Number of items in SST to start with. [type = integer]
      M = Size of SST. [type = integer]
      q = Incremental rate for SST. [type = integer]
      MNIST Training and Test Data.

TRS = Load_MNIST_TrainingData ()
TS = Load_MNIST_TestData ()

errorRate = 100

SST = select p random items from TRS

runsLeft = (M-p)/q
epsilon = 0.1

while runsLeft > 0 and errorRate > epsilon, repeat:
      MCS = Ø
      ∀ trs ∈ TRS
            find 1-NN in SST.
            If misclassified, add trs to MCS set.

      Calculate the new errorRate.

      Select q random items from MCS and add them to SST.
      runsLeft = runsLeft - 1;


SST is the prototype of TRS we will use to classify TS.

∀ ts ∈ TS
      find 1-NN in SST
Calculate the error rate.
```

3. *Experimental results.*

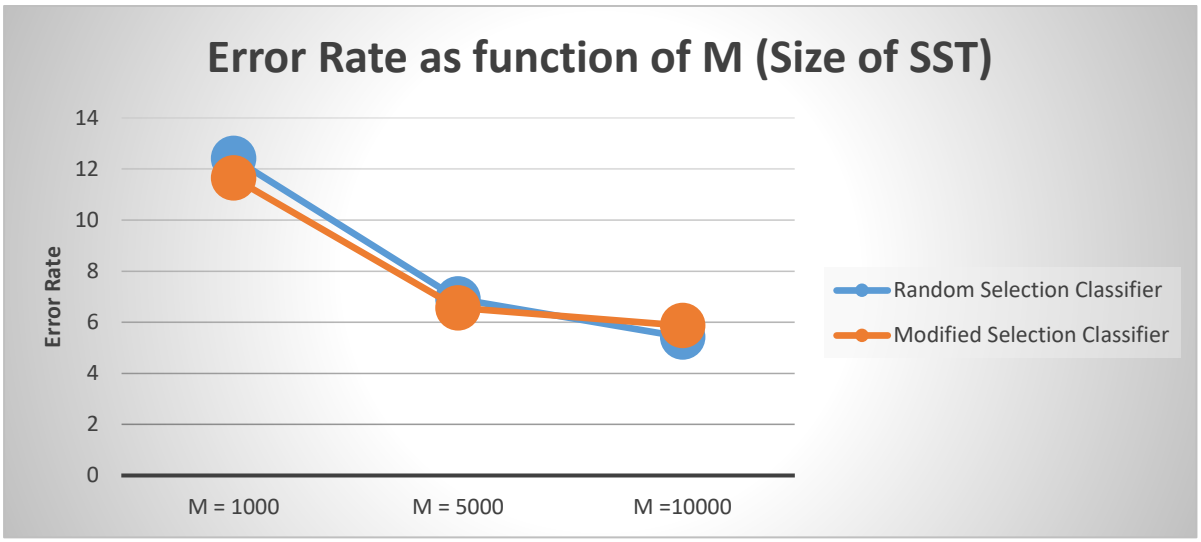
For experimental purpose, three configurations of {M, p, q} are used.
{M=10000, p=1000, q =100}
{M=5000, p=500, q =50}
{M=1000, p=100, q =10}

For each configuration, modified classifier is run 30 times and error rate in each run is calculated. As the modified classifier starts with random subset and increments constantly with random instances, confidence intervals for each setup are calculated. Furthermore, for every setup of modified classifier, comparison of error rate with that of random selection classifier is done.

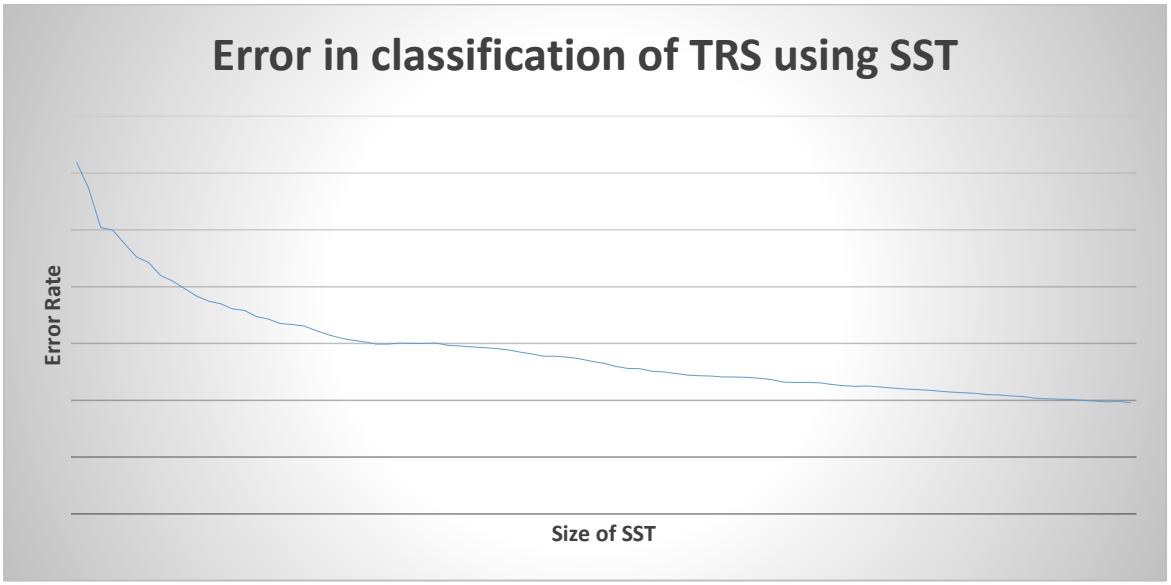
Plot of average error rate in each configuration of modified selection classifier, compared to average error rate of random selection classifier. (Average taken over 30 runs).

Average Error rate for 30 runs.

Error Rate for Modified Classifier as M increases.



To visualize the selection process for SST, the plot below shows the trend of error rate in classification of TRS using SST with the increase in size of SST for one run. As expected, the error rate decreases.



Snap taken from configuration {M=1000, p = 100, q=10}

Calculating confidence interval for error rates in each setup.

We have to estimate the standard error based on the data for 30 runs. For the whole population distribution of error rate, we do not have much information about standard deviation of error or mean. So, we calculate the estimated standard error. Given 30 error rates, we first calculate the sample mean and sample variance.

N: Number of samples.
S: Set of N Samples from error rates
μ = Sample mean
σ = Sample standard deviation.

Then, we calculate estimate of Standard Error as:

$$S_M = \frac{\sigma}{\sqrt{N}}$$

Then, the confidence level is calculated by formula:

$$Confidence\ Level\ Range = \mu \pm (S_M) * (t_{val(ConfidenceLevel)})$$

where $(t_{val(ConfidenceLevel)})$ is taken from the t-value table. For example, if we want to calculate 95% confidence interval for 30 error rates, we take the value t_{95}, which is 2.042.s

Here is the table showing the confidence level of errors for both the classifier in each configuration of run.

Over N = 30 runs	Configuration 1 {M=1000, p=100, q=10}		Configuration 2 {M=5000, p=500, q=50}		Configuration 3 {M=10000, p=1000, q=100}	
Average	11.434	10.95966667	6.43733333	6.251333	5.178333	5.544
Minimum	10.47	10.44	6.07	5.96	4.84	5.29
Maximum	12.42	11.65	6.91	6.57	5.42	5.87
Standard Deviation	0.47238	0.324584438	0.18275541	0.128244	0.13519	0.165
Standard Error	0.08624	0.05926074	0.03336642	0.023414	0.024682	0.03
Upper limit	11.6104	11.08085488	6.50556766	6.299215	5.228809	5.605
Lower limit	11.2576	10.83847845	6.369099	6.203452	5.127858	5.482

4. *Critical Evaluation*

Random Selection v/s Prototype

As it is evident from the experiment's results, the modified selection method is clearly **not** an improvement over the random selection method, especially for larger values of M. This behaviour can be explained due to the selection method in the modified version.

a. At every incremental step, we add instances which were misclassified. It might be possible that the instances which we are adding is noisy data. These noisy data points hinder the smooth behaviour of the function and can adversely affect the classification by rest of the instances.
b. If any noisy instance is selected during random selection, it will be retained in the final set. Our model does not eliminate the noisy instances from selection. Moreover, if one noisy instance exists, there are more chances that more of them are added in the final set. So we need a way to filter out noisy instances and avoid adding them.
c. For larger values of M, after considerable selection runs, the set (TRS – SST) will contain very few instances which are not noisy. Generally, as M grows, the percentage of noisy instances in the set (TRS-SST) increases. So if M is large, the classifier ends up selecting those instances with higher probability and hence the error rate is adversely affected as it might affect the instances which were previously classified correctly.

Scope for Improvement

There is definitely a scope for improvement in the modified classifier as it is comparable to random classifier. Following are the possible improvements:

a. One clear direction is to avoid selecting noisy data. We need to think on how to change the selection routine from random to a heuristic based selection, so that only relevant instances are added in the SST.
b. Instead of incremental approach, we can think of a decrement approach in which we remove the instances from SST which are redundant and noisy.
c. Values of p, q and M can be changed and best suited after observations. In the experiments, the values were fixed for a particular configuration. For the same configuration, we can compare the prototype with other values of p, q and M.

Try Next

On the similar lines of scope for improvement, I would like to try an incremental selection approach with more heuristics, and also a decrement approach. Also, we can use locality based approach to define clusters and then define the subset SST as one point from each cluster or average of each cluster. We can ignore the clusters which have very few data points because they are likely to be noisy.

To improve overall accuracy of the classifier, we can also think of other classification techniques than 1-NN and compare the error rates.