Programming-1 : Prototype Selection for Nearest Neighbor

Aman Raj, PID: A53247556

January 23, 2019

1 Nearest Neighbor Prototype Selection

Our dataset consist of MNIST samples, where each sample is a 28x28 grayscale image with class label between 0 to 9. Dataset has standard split of training set of 60,000 samples and test set of 10,000 samples. Each sample is converted into 1-Dimensional feature vector of 784 features. In order to create a baseline accuracy score, we trained a 1-NN classifier on the full standard training set and validated on test set, resulting in an accuracy of 96.8%. Now, we use two different methods for prototypes selection for 1-Nearest Neighbor classifier which are outlined in the following subsections.

1.1 Uniform Random Prototype Selection

We randomly select M number of prototypes from training set without replacement, such that the probability of picking any sample is same. Also, we make sure, we have $M/10$ samples for each class, since we have total 10 classes. For each value of M, we repeat the experiment several times and report the average accuracy with error margin with 95% confidence on test set in Table 1.

1.2 Strategic Prototype Selection

In this method, we iterate over the training set one sample at a time sequentially and strategically take decision whether to select the sample as prototype or not. We explain the full algorithm in next section and compare the result with uniform random selection method on test set in Table 1.

2 Pseudocode For Strategic Prototype Selection

Let's say we have to choose M number of prototypes from the training dataset.

Step 1 Create a set of basic prototypes S that contains mean image of each class calculated using samples of training set only. Now, we will increase the size of this set S in next steps.

Step 2 Choose a sample in sequence from training dataset x, predict its class by using 1-NN on set S, if it is misclassified then continue or skip to Step 4.

Step 3 Remove the missclassified sample x from training dataset and add it to the prototype set S.

Step 4 Repeat the steps from Step-2 to Step-4, until the number of prototypes in S is equal to M or we exhaust the training dataset.

Step 5 Now, if the number of prototypes in S is less than M, then with uniform-random selection add D number of samples from training set to S, where $D = M - |(S)|$.

Step 6 Return S, that contains M prototypes.

3 Experimental Results

For uniform-random prototype selection method, we did 100 runs for each choice of M in $100, 500, 1000, 5000, 10000$. The performance reported in Table 1, are within 95% confidence level. The margin of error for mean accuracy is calculated by:

$$error = \frac{Z_p * \sigma}{\sqrt{N}}$$

where $Z_p = 1.96$ for 95% confidence level, σ is standard deviation of sample space having N samples.

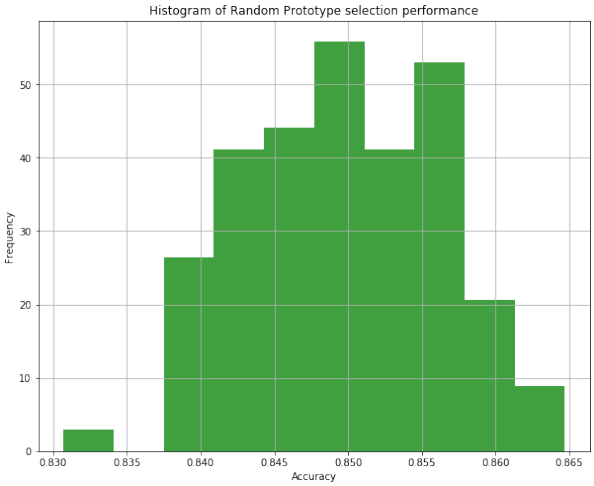

Figure 1: Accuracy bars for 100-runs for $M = 500$ prototypes on test set of MNIST.

Prototypes(M)	Strategic Selection	Uniform-Random Selection
100	74.05	70.80 ±0.38
500	85.13	84.91 ±0.12
1000	88.88	88.45 ±0.08
5000	93.60	93.53 ±0.03
10000	94.84	94.83 ±0.03

Table 1: Comparison of performance of Strategic selection method with Unifrom-random selection method on different number of M prototypes.

4 Critical Evaluation

Since the gap between the accuracy is negligible at higher values of M in Table 1, so we can say strategic selection method is slightly better than uniform selection method. This coan be explained by the fact that the training data is highly balanced and clean so even a random selection gives strong candidates as prototypes. There seems to be a little scope of improvement as we are already close to our baseline score of 96.8% at $M = 1000$. However, if the dataset is made more challenging with introduction of image distortions, noise(like salt pepper noise), imbalance in data, rotation of digits at different angle, and digits at different scale, then strategic selection method should be able to find stronger prototypes than random selection method. This will be future work of this method.