Prototype Selection for Nearest Neighbor

Yilan Chen
Department of Computer Science and Engineering
University of California, San Diego
yilan@ucsd.edu

1 Key idea of Prototype Selection

We followed the basic ideas of Hart's Condensed Nearest Neighbour (CNN) [Hart, 1968] and Fast Condensed Nearest Neighbor (FCNN) rule [Angiulli, 2007], while applying Principle Component Analysis (PCA) with the data before the prototype selection.

First we applied PCA to reduce the dimension of the dataset D. We initialized the prototype set S as the centroids[1] of data points of every class. Then we added a point classified wrong with S and repeated until the prototype set S didn't change or the size of S equaled some M. If the size of S at the end was smaller than M, we randomly selected other prototypes from $D - S$.

2 Algorithm

Algorithm 1 shows the pseudocode of prototype selection for nearest neighbor. We used SVD to do PCA and took the first c components, which counts for 90% of variance of original data. For prototype selectioin, we followed FCNN [Angiulli, 2007] and initialized the prototype set S as the centroids of data points of every class. But due to the limit of computation power, we did not apply further complicated methods to select representative points from those classified wrong but rather took the first point classified wrong.

3 Experimental Results

Table 1 shows the results of the experiments with MNIST. For prototype selectioin and random selectioin, we repeated 20 experiments to calculate the mean and standard deviation of the test accuracy.

Due to the limit of computation power, we randomly selected $10k$ samples ($\frac{1}{6}$ of $60k$) to do PCA. We used SVD to do PCA and took the first about 344 (out of 784) components, which counts for 90% of variance of original training data. When testing, we trained a nearest neighbor classifier with the prototype set selected, and projected the test data onto the principle components, then classified the point with the nearest neighbor classifier.

We noticed that for the prototype selectioin of size $10k$, selecting about $6k$ points acheived a training-set-consistent subset [Hart, 1968]. And the remaining was taken randomly from training set.

4 Critical Evaluation

This method always has better accuracy and lower standard deviation over random selection for different prototype set size M. And also, with the increase of prototype size, the standard deviation of both prototype selectioin and random selectioin decreased.

[1]Given a set S of points having the same class label, the centroid of S is the point of S which is closest to the geometrical center of S.

Algorithm 1 Prototype selection for nearest neighbor

```
Input: Training dataset $D = \{(x_i, y_i)\}_{i=1}^N$, prototype set size $M$;
Output: Prototype set $S$ of size $M$.
  $S = \emptyset$
  $D' = PCA(D)$
  $\triangle S = Centroids(D')$
  while $|S| < M$ and $\triangle S \neq \emptyset$ do
    $S = S \cup \triangle S$
    $\triangle S = \emptyset$
    Train a nearest neighbor classifier with $S$ as $nn(\cdot, S)$
    for $(x'_i, y_i)$ in $D'$ do
      if $nn(x'_i, S) \neq y_i$ then
        $\triangle S = \{(x'_i, y_i)\}$
        break
      end if
    end for
  end while
  if $|S| < M$ then
    Randomly select $M - |S|$ samples from $(D' - S)$ as $\triangle S$
    $S = S \cup \triangle S$
  end if
  return $S$
```

Table 1: Test accuracy of different selection methods for nearest neighbor with MNIST

Method	Size (M)	Test Accuracy (mean $\pm$ std)
prototype	1000	88.786 ± 0.420
random	1000	88.504 ± 0.501
prototype	5000	93.856 ± 0.101
random	5000	93.310 ± 0.157
prototype	10000	95.248 ± 0.101
random	10000	94.848 ± 0.128

But the improvement was not significant. The main reason may be the randomness of selection scheme, which can be improved with more and better restrictions, like what is applied in FCNN [Angiulli, 2007]. But due to the limit of computation power and time, we left further improvements later.

This algorithm followed the basic ideas of Hart's Condensed Nearest Neighbour (CNN) [Hart, 1968] and Fast Condensed Nearest Neighbor (FCNN) rule [Angiulli, 2007]. This technique is very sensitive to noise and to the order of presentation of the training set instances, in fact CNN by definition will tend to preserve noisy instances [Cunningham and Delany, 2020]. The idea behind the condensation methods is to preserve the accuracy over the training set, but the generalization accuracy over the test set can be negatively affected [Garcia et al., 2012]. Further improvements may include better selection scheme with more restrictions, and better distance measure such as correlation and tangent distance. Also we may want to try kernel methods and neural networks for better representation of data.

From the view of computation, the computation needed for this algorithm is still large for a labtop. Improvements may include selecting bounch of representative points at a time rather than one point at a time.

References

F. Angiulli. Fast nearest neighbor condensation for large data sets classification. *IEEE Transactions on Knowledge and Data Engineering*, 19(11):1450–1464, 2007.

P. Cunningham and S. J. Delany. k-nearest neighbour classifiers–. *arXiv preprint arXiv:2004.04523*, 2020.

S. Garcia, J. Derrac, J. Cano, and F. Herrera. Prototype selection for nearest neighbor classification: Taxonomy and empirical study. *IEEE transactions on pattern analysis and machine intelligence*, 34(3):417–435, 2012.

P. Hart. The condensed nearest neighbor rule (corresp.). *IEEE transactions on information theory*, 14 (3):515–516, 1968.