CSE 250B - Learning Algorithms

Prototype selection for nearest neighbor

Pin Tian (PID: A53219987)
pitian@ucsd.edu
Department of Electrical and Computer Engineering

1. Introduction

For the course CSE 250B Learning Algorithms students are asked to think of a good strategy for choosing prototypes from the training set.

In the coming pages of the report the procedures and findings during the Prototype selection are described.

2. My idea for prototype selection

In this PA, I used two prototype selection methods.

The first one I come up with is that find M nearest points around each center of 10 digit classes. Let's call this method K Nearest Centers. So the total number is $10 \times M$ in the final training set.

The second one is Condensed Nearest Neighbors algorithm which uses a 1 nearest neighbor rule to iteratively decide if a sample should be removed or not.

3. Pseudocode

3.a. K Nearest Centers

Algorithm 1 centersFinder

```
1: procedure CENTERSFINDER(data, label)
2:     centers ← all-zero array with size (number of classes, number of features)
3:     numClass ← all-zero array with size(number of classes, 1)
4:     for all digit data in every class do
5:         calculate mean
```

Algorithm 2 K nearest center

```
1: procedure KNC(data, label, M)
2:     trainSet ← all-zero array with size (M, number of features)
3:     labelSet ← all-zero array with size(M, 1)
4:     for each class of digits do
5:         find M/10 points which are the nearest to the class center
```

3.b. Condensed Nearest Neighbors

Algorithm 3 Condensed Nearest Neighbors

```
procedure CNN(data, label, M)
    trainD ← empty list
    trainL ← empty list
    num ← 0
    ptr ← 0
    randInd ← M uniform random indices
    trainD ← data[randInd[ptr]]
    trainL ← label[randInd[ptr]]
    num ← 1
    ptr ← 1
    while num < M do
        fit 1 NN model using (trainD, trainL)
        pred ← predict data[randInd[ptr]]
        if pred ≠ label[randInd[ptr]] then
            add data[randInd[ptr]] to trainD
            add label[randInd[ptr]] to trainL
            num ← num+1
        ptr ← ptr+1
        if ptr == length of data then
            ptr ← 0
```

4. Experimental results

I used Jupyter Notebook Python in this programming assignment. Due to the low large data processing speed, I only got at most 6000 subset. Below are the results.
According the following confidence interval foumula:

$$(\bar{x} - z^* \frac{\sigma}{\sqrt{n}}, \bar{x} + z^* \frac{\sigma}{\sqrt{n}}) \tag{1}$$

So the confidence interval of random sample is:

$$\begin{aligned} M = 1000 \quad & (0.11588 - 0.00427\, 0.11588 + 0.00427) \\ M = 5000 \quad & (0.06502 - 0.00139\, 0.06502 + 0.00139) \\ M = 6000 \quad & (0.06114 - 0.00141\, 0.06114 + 0.00141) \end{aligned} \tag{2}$$

Because the speed of CNN algorithm is very slow, I only did several times experiments on $M = 1000$.
the confidence interval of CNN when M = 1000 is :

$$(0.10848 - 0.00268\, 0.10848 + 0.00268) \tag{3}$$

Error Rate of Random Selection			
number of experiment	M = 1000	M = 5000	M = 6000
1	0.1093	0.0672	0.0603
2	0.1144	0.0663	0.0585
3	0.1126	0.0632	0.0629
4	0.1166	0.0634	0.0626
5	0.1265	0.065	0.0614
Mean	0.11588	0.06502	0.06114
Standard deviation	0.005823195	0.001570223	0.001610714

Figure 1

Error Rate of Nearest points of centers		
M = 1000	M = 5000	M = 6000
0.2087	0.1601	0.1496

Figure 2

Error Rate of Condensed nearest neighbors			
number of experiment	M = 1000	M = 5000	M = 6000
1	0.1083	0.062	0.0593
2	0.103		
3	0.1101		
4	0.1141		
5	0.1069		
Mean	0.10848		
Standard deviation	0.003654258		

Figure 3

We can see that the results after CNN has 1% increase on classification performance. KNC has worse performance comparing to random sample, although it is a determinant method.

5. Critical evaluation

Is your method a clear improvement over random selection?
No, my two methods doesn't have an obvious improvement over random selection. I suppose the reason is that these two methods both don't solve the essential problems which includes digits rotation, scaling and translation.

Is there further scope for improvement?
Try to use another distance formula such as tangent distance. Because it can treat the scaling, rotation conditions of the handwriting digits.

What would you like to try next?
Except trying to use tangent distance. I also want to try some other prototype selection methods such as OneSidedSelection, NeighbourhoodCleaningRule which are the derived algorithms of Condensed Nearest Neighbors.