HW3 Report

Gong Lixue

November 5, 2017

1 Neural Networks

The implementation of feedforward and backpropagation process is shown in code folder. At the end of training iterations, $loss = 1.717e-01$, $accuracy = 0.9400$, which is not the optimal actually. And for testing, $loss = 2.522e-01$, and $accuracy = 0.9230$.

2 K-Neareast Neighbor

(a) boundary figures is shown below:

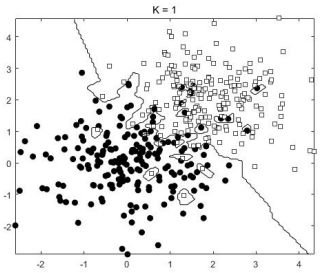

Figure 1: K = 1

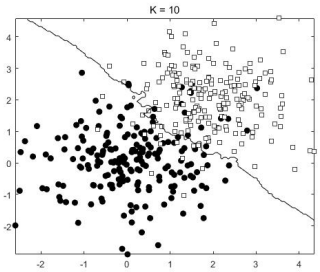

Figure 2: K = 10

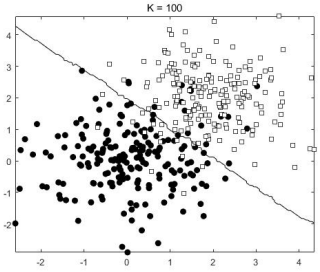

Figure 3: K = 100

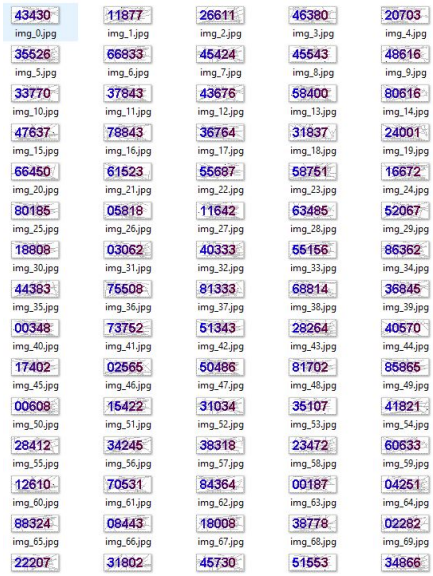

Figure 4: images fetched by python spider

(b) One of methods is that choosing a proper K by Cross-Validation. Compute the validation error on validation set using different values for K. And pick the optimal K with the lowest validation error.

(c) Firstly, I wrote a simple python script to fetch and save check code images from this website automatically.(See Figure 4)

And then, label these images by hand. I use 100 images of them. The raw labels are recored in file $./knn/hack_py/label\_100img.txt$. Each row in this file means the actual codes that corresponding image represents.

Before recognizing a check code image, we should generate a *.mat* file which is used for training in knn. Run *gen_hack_data.m* to generate a .mat file. Finally, we can test the algorithm to recognize a check code image(see *knn_exp.m* Part2 and Figure 5).

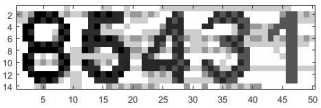

Figure 5: show_image

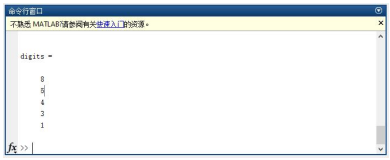

Figure 6: result digits

3 Decision Tree and ID3

The decision tree and respective information gain is illustrated as the graph below:

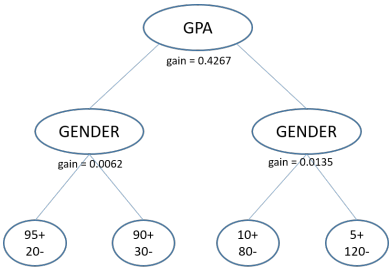

Figure 7: decision tree

4 K-Means Clustering

(a) The visualization of process with smallest SD:

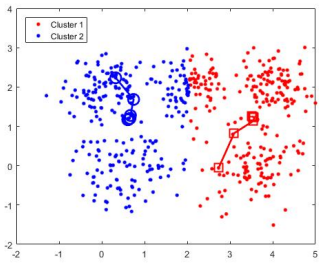

Figure 8: kmeans process with smallest SD

The visualization of process with largest SD:

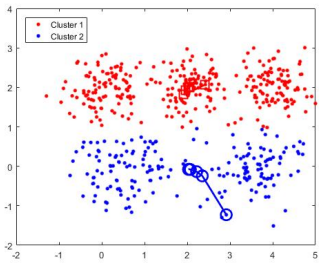

Figure 9: kmeans process with largest SD

(b) We can run it from multiple starting points, and pick the solution with the smallest SD.

(c) The visualization of centroid is illustrated below. And we can find that the cluster center can represent the patterns in dataset.

Figure 10: k=10

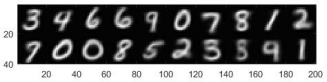

Figure 11: k=50

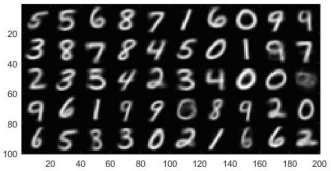

Figure 12: k=50

(d) This is the original image:

Figure 13: original image

And wen can obtain the compressed images after runing *vq.m*:

Figure 14: K = 8

Figure 15: K = 16

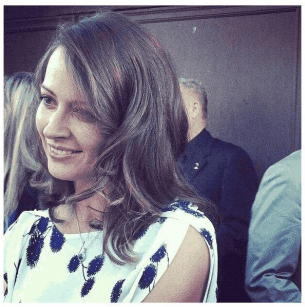

Figure 16: K = 32

We can observe that when we set K to 64, these is no obvious change comparing to original image. When K=64, each pixel can be represented with $log(K) = 8$ bits. So the data is not efficiently compressed actually.