# Handwriting Digit Recognition

cs559

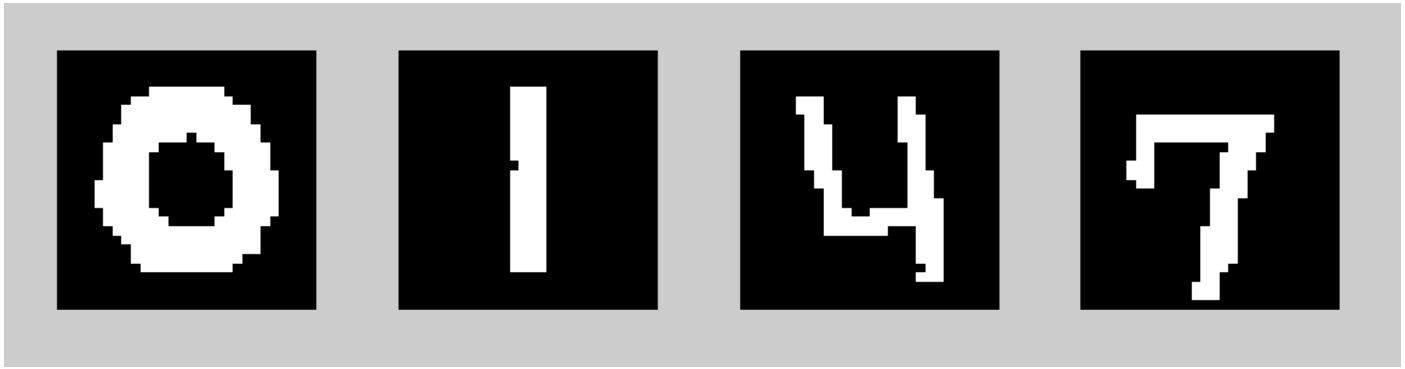
Junsheng Liu

# Introduction

- Handwritten digit recognition is an very important aspect of computer vision, and it is widely used in our world. Now I'll use four kinds of methods to deal with data and two methods(knn, neural network) to realize handwritten digit recognition, get the accuracy of them, and compare them. Actually I have also used some other methods, like NB and MLE, which have an accuracy of less than 0.3, so I mainly talk about knn and neural network with 4 methods of dealing with data. And I will use half of the data as training set, and the other half as testing set.

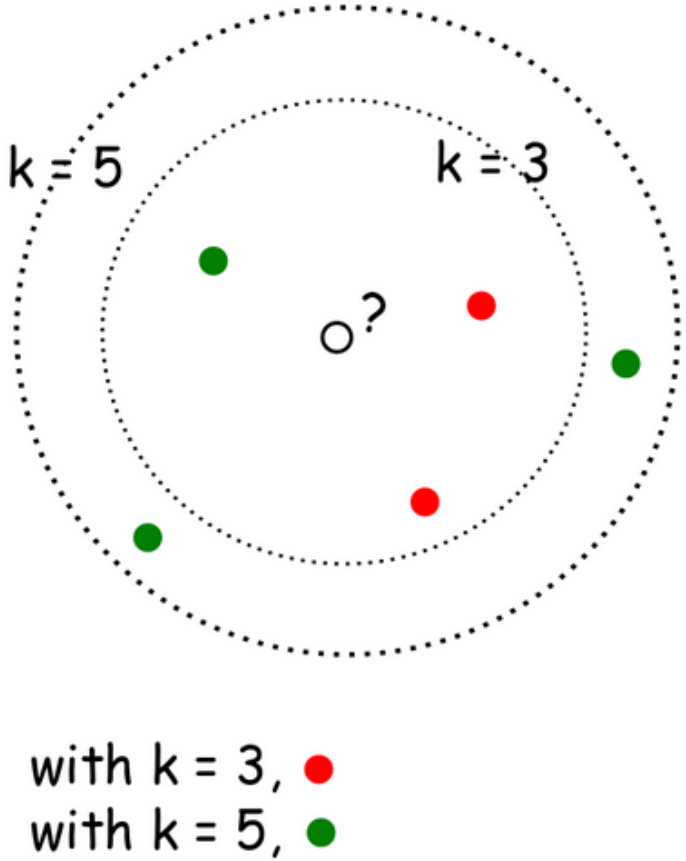
# Introduction

- My data comes from Kaggle with the website https://www.kaggle.com/c/digit-recognizer/data
- There are 42000 samples with each one having 785 attributes. First attribute is label with value between 0 and 9, other 784 attributes are pixels of 28*28 with value between 0 and 255. For convenient, I normalize the value of 784 attributes with 0 and 1. 0 is for black color and 1 is for white color.



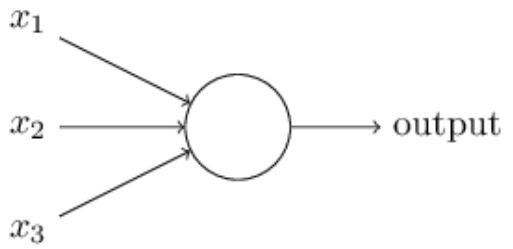
# Introduction

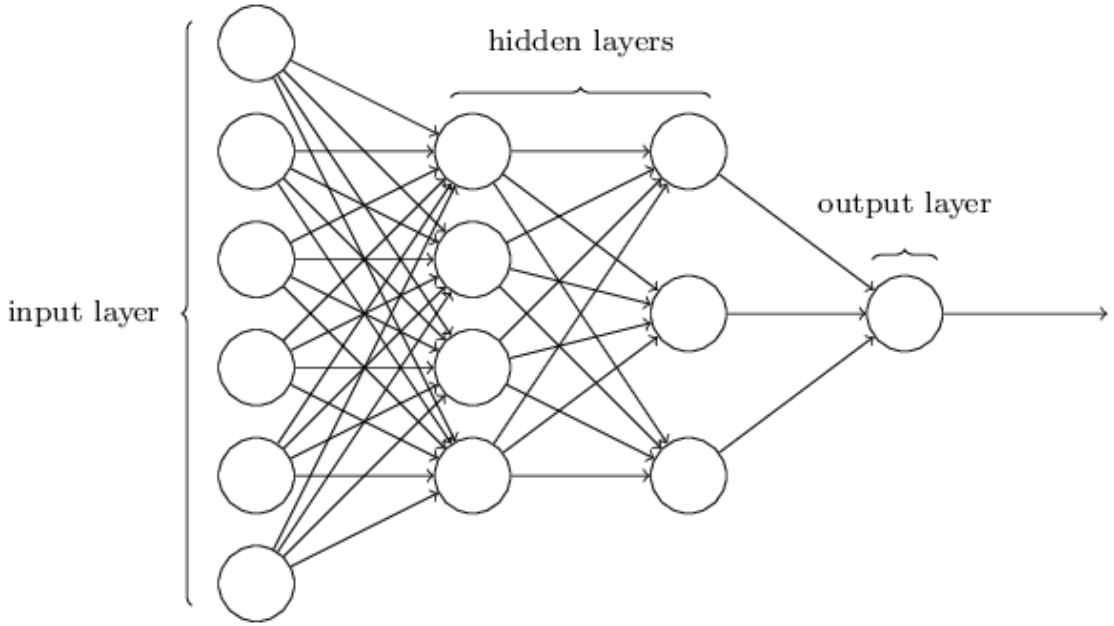
- Knn stands for K-nearest neighbor estimation, which means assign the class of the most common among an object's k nearest neighbors to the object.  This is an easy method and we can use the function knnsearch of matlab to achieve it.



# Intruction

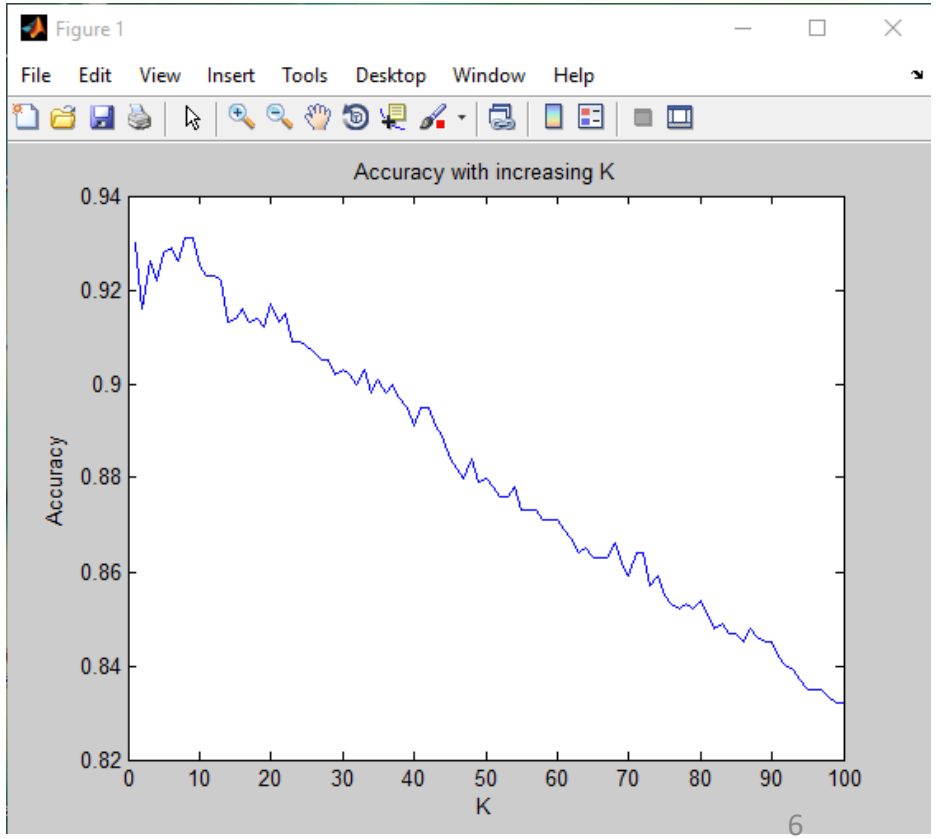
- Neural network is a biologically-inspired method of computer science. It uses a word called perceptron, which works like that using input $x_i$, compare the weighted sum $\sum_i \ w_i x_i$ with threshold value to get the output.
- Neural network has three layers: input layer, hidden layer and output layer. In each layer, there are many perceptrons.



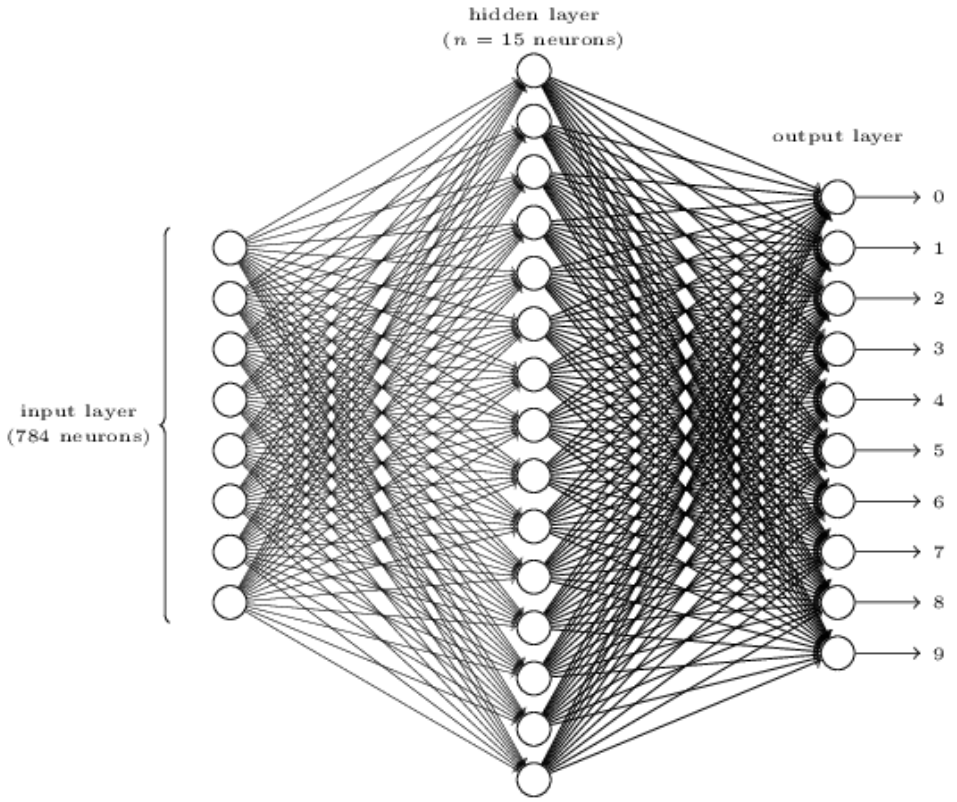
# Experiment(1)

- First, I want to use all the information of the data as an input. Each sample has 785 attributes, although it is a very large number, but we will not lose any information of the sample.
- For knn, I compute the accuracy with increasing k, we can see that almost when k=11, accuracy is the largest, around 0.93, so for next experiments with knn, I would just choose k=11 for convenient .



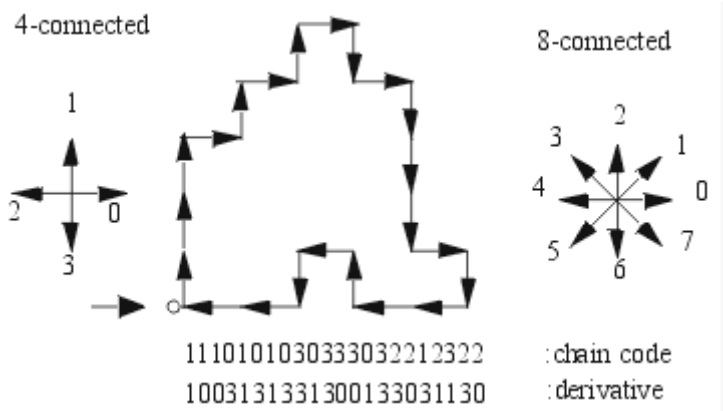
# Experiment(1)

- For neural network, I choose 15 neurons in hidden layer, 784 neurons in input layer, 10 neurons in output layer. Because of 784 attributes, there would be more than 785*15 weights need to be updated for every input, which is a lot of work. Waiting a lot of time with matlab, I get the accuracy of 0.7521, which is not very high, so we can consider experiment(2).



# Experiment(2)

- 784 attributes is a very large number, which would takes a lot of time to do some classifications. We need a different view, can we just use a few features to represent the digit that 784 attributes stand for. I search the Internet, and find a method called freeman code, which is using outline of the digit as the features. I use 8-connected freeman code, which means 8 directions.



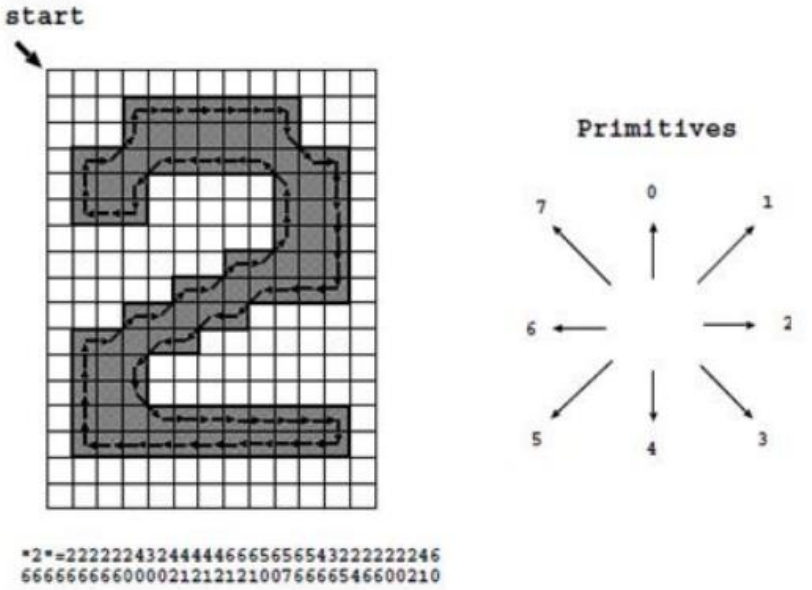
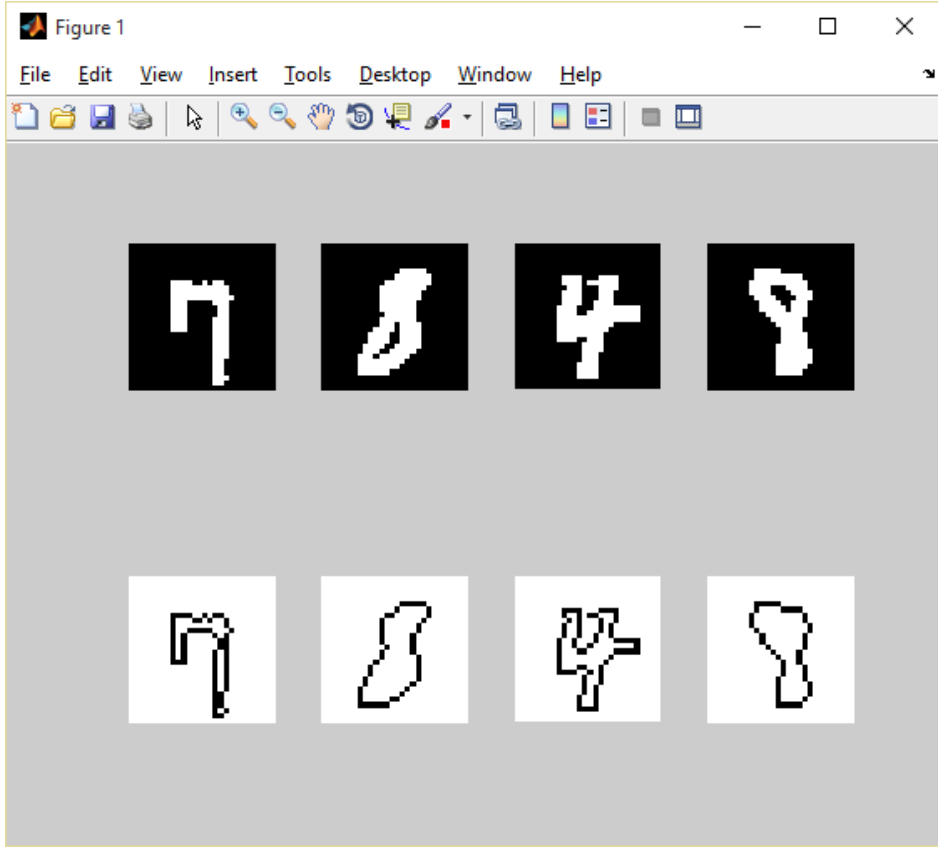




Figure2 Representation of a digit 2 drawn on the left using Freeman code

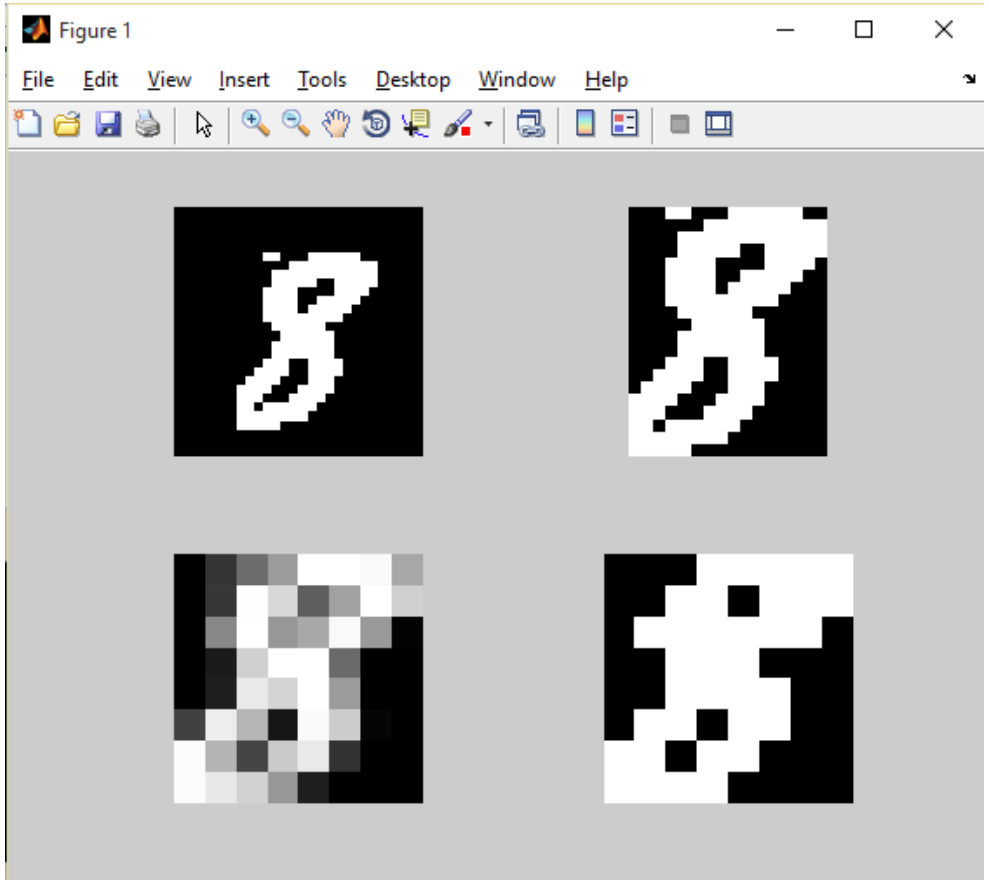
# Experiment(2)

The right picture is use freeman code to get the outlines for each digit. For different size of the digit, the number of directions is different, so I use percentage of the number of a direction in total number of directions as the features, and for the number “6” and ”9”, they have the same directions, so I add the percentages of pixels that fell in the top, bottom, left and right of the image as 4 features. So there are total 12 features. However, the results are not very good. For knn, the accuracy is only 0.5616, and for neural network, 12 input neurons, the accuracy is only 0.4137. It seems that this 12 features is not enough to represent the image.



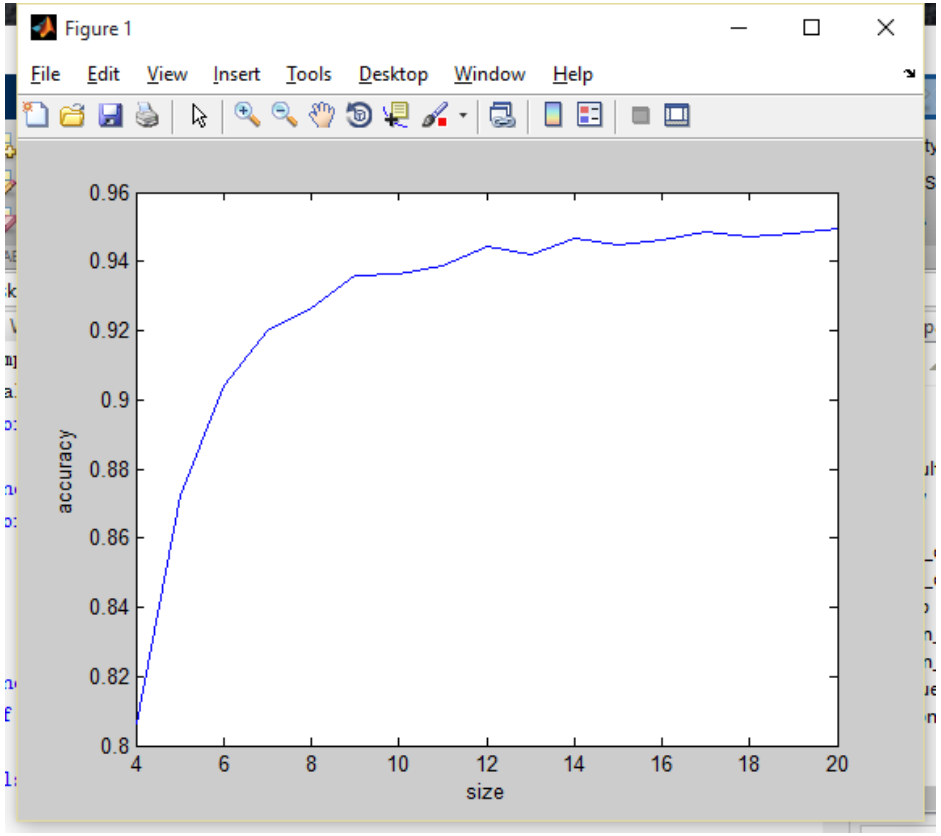
# Experiment(3)

- It seems that we need another different view, let's take a brainstorm. We can cut the image so that the digit is close enough to the boundry. Because of different size of the digit, I use the function imresize to change the size to n*n, and then update the pixel value to 0 and 1. Take right picture for example, I change the size to 8*8, the 4$^{th}$ picture is what we want, which seems not bad.



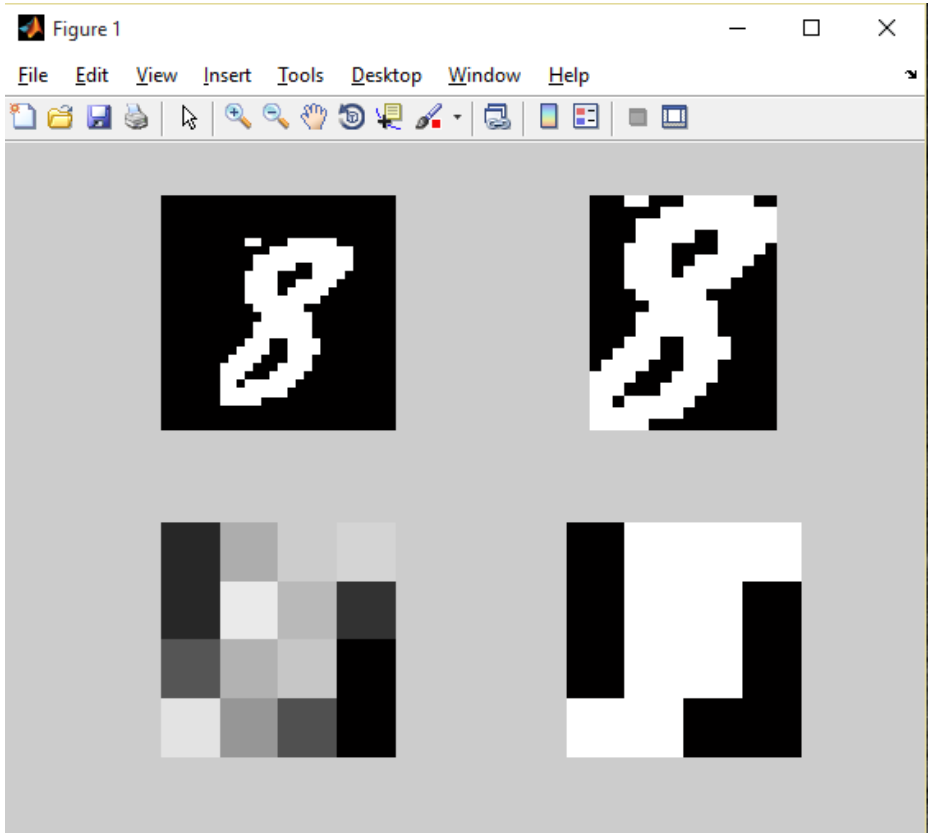
# Experiment(3)

- So which size do we need to change to? With size n*n, there would be $n^2$ attributes. For knn, I plot a picture with change of accuracy according to n. We can see that as long as n is bigger than 5, the accuracy is not smaller than 0.9. So I choose n=6.
- For neural network, there would be 36 input neurons, and the accuracy is 0.8213, which is very good result for neural network.



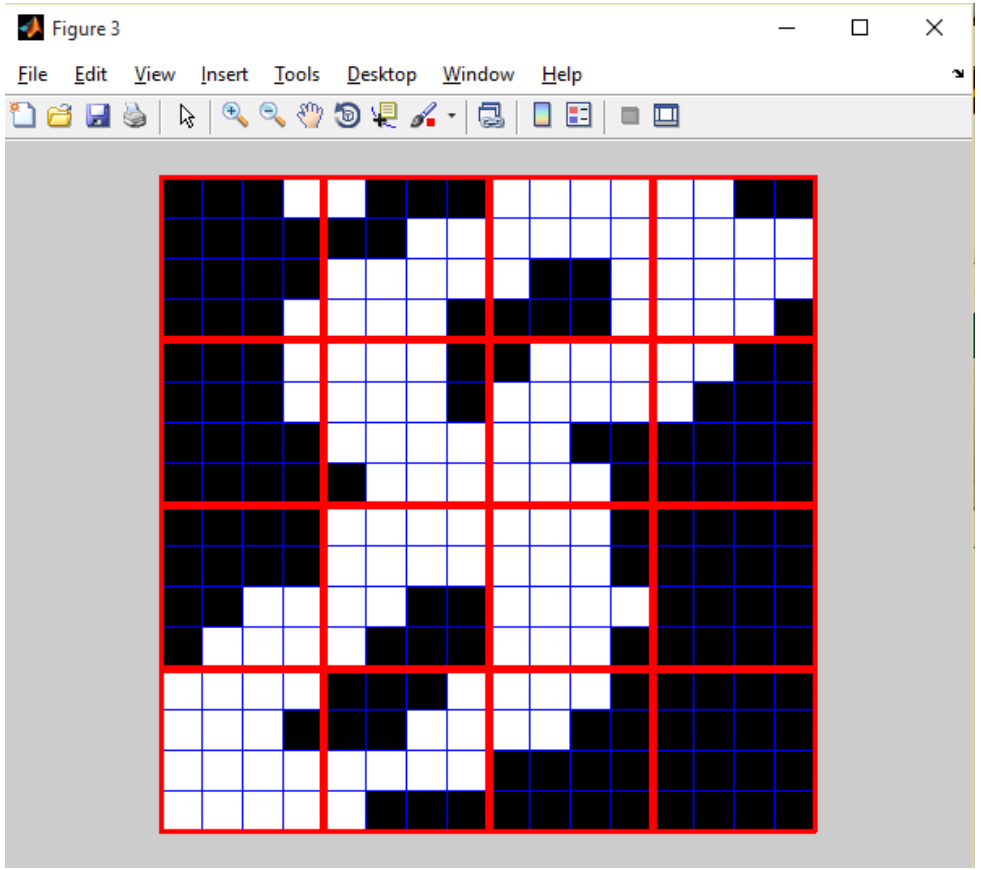
# Experiment(4)

- Let’s take a brainstorm again. 36 attributes is also somewhat too many, but if we change n to 4, like the picture on the right, we can not recognize the fourth image, so the accuracy would be lower. But we can use the idea of percentage in experiment (2) and resizing image in experiment(3).



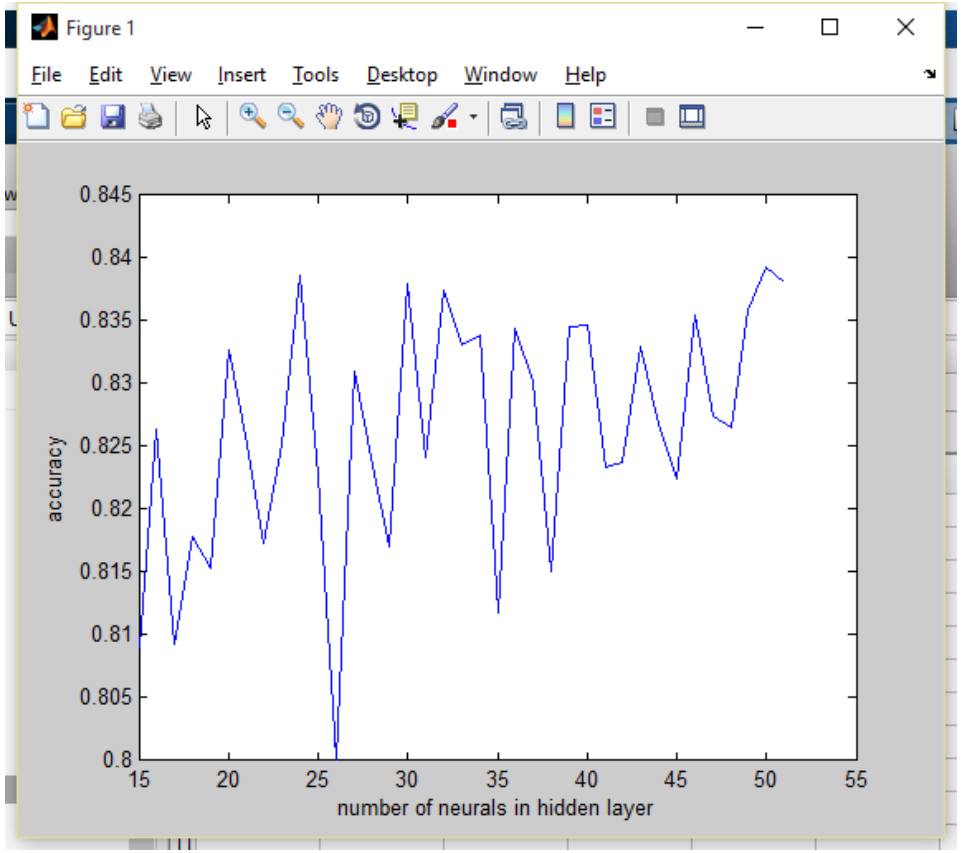
# Experiment(4)

- We can use the method of experiment(3) to get a size 16*16 image, then divide the image into 16 parts with size 4*4 pixels. We can consider the percentage of number of pixels with value 1 in each part as a attribute, so there would be 16 attributes, which is smaller than 36.



# Experiment(4)

- For knn, we have just 16 attributes and get the accuracy of 0.9290, and for neural network, we get the accuracy of 0.8280, which is really amazing. It really takes fewer time than previous methods and has the highest accuracy than previous experiment so far. And also, I can consider the relation between the number of neurons in hidden layer and the accuracy. We can see that 23 number of neurons is enough, and has the accuracy about 84%.



# Conclusion

- From the previous experiments, we can see that the fourth experiment is the best. It has a high accuracy and takes a very few time. Most of all, we get the 16 attributes that can take a good advantage of the image, so we can also use some other methods, like SVM, liblinear, PCA or whatever, by the 16 attributes.
- Also, in my project, the accuracy of knn is always bigger than that of neural network, because my data is very large.