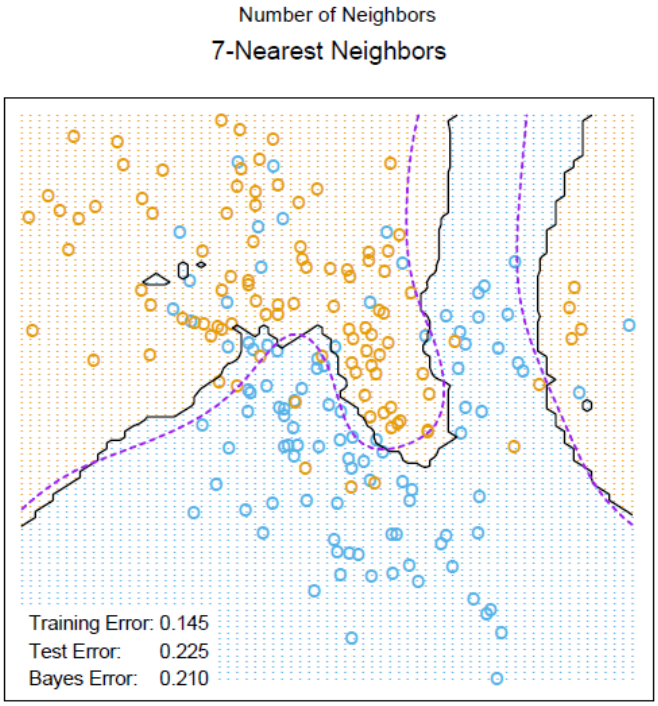
Figure 1



Hastie, et al. Elements of Statistical Learning, Figure 13.4

The inputs are the blue and orange circle point in the figure, the outputs are the blue and orange lattice points and the black contour line. In this part I desire to learn the algorithm of K Nearest Neighbors.

In real world, this can be described as there are two types of plants. We measure two features of them and plot the data in a graph. After that, using the algorithm to classify different types of plants. I download the data from url https://web.stanford.edu/~hastie/ElemStatLearn/data.html, the Mixture Simulation dataset. In the dataset, it has 200 rows and 2 columns in the input x. Label y have 200 rows and 1 column, and the data in it only has 0 and 1. 0 means the point in x is blue and 1 means the point in x is orange.

Dataset:

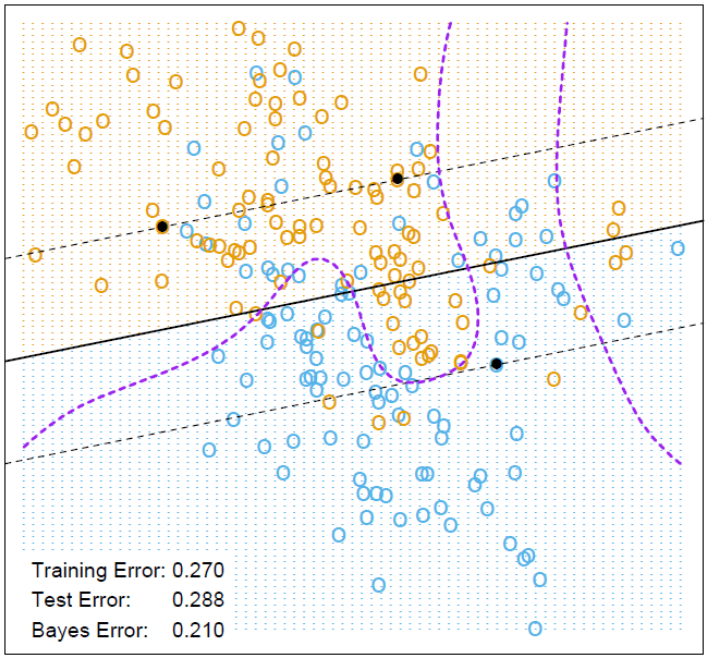
ESL.mixture	list [8]	List of length 8
x	double [200 x 2]	2.5261 0.3670 0.7682 0.6934 -0.0198 2.1965 0.3211 0.0315 0.7175 0.7772 ...
y	double [200]	0 0 0 0 0 0 ...
xnew	double [6831 x 2] (S3: matrix)	-2.6 -2.5 -2.4 -2.3 -2.2 -2.1 -2.0 -2.0 -2.0 -2.0 -2.0 -2.0 ...
prob	double [6831]	3.55e-05 3.05e-05 2.63e-05 2.27e-05 1.96e-05 1.69e-05 ...
marginal	double [6831]	6.65e-15 2.31e-14 7.62e-14 2.39e-13 7.15e-13 2.03e-12 ...
px1	double [69]	-2.6 -2.5 -2.4 -2.3 -2.2 -2.1 ...
px2	double [99]	-2.00 -1.95 -1.90 -1.85 -1.80 -1.75 ...
means	double [20 x 2]	-0.2534 0.2667 2.0965 -0.0613 2.7035 2.3772 1.7415 0.3712 1.2334 -0.2087 ...

Pseudo code

```
Algorithm KNN(A[n], k, x)
    Input:
        A[n] is the feature of n training samples, k is the nearest neighbor number, and
x is the new sample.
    Initialize:
        Make A[1]~A[k] as the initial neighbor of x;
        Calculate the Euclidean distance d(x, A[i]), i =1,2..., k, between the test sample
and x;
        Sort d(x, A[i]) in ascending order
        Calculate the distance D between the farthest sample and x, that is, Max {D (x,
A[i])};
    for (i = k + 1; i < n; i++)
        Calculate the distance d(x, A[i]) between A[i] and x;
        if (d(x, A[i])) < d then replace the farthest sample with A[i];
        Sort d(x, A[i]) in ascending order
        Calculate the distance D between the farthest sample and x, that is, Max {D (x,
A[i])};
    End the for
    Calculate the probability of category of first k samples A[i], i =1,2..., k;
    The class with the highest probability is the class of sample x;
    Output: category of x.
```

I will write the KNN algorithm by myself and input the dataset. After that, I will use some graph library like ggplot to generate the figure. In this part, I will learn the algorithm of KNN, and will try to write the algorithm code by myself.

Figure 2



$C = 10000$

Hastie, et al. Elements of Statistical Learning, Figure 12.2

The inputs are the training dataset, the blue and orange circle point. The outputs are the black boundary. I desire to learn the algorithm of linear support vector machine, and how to use it to classify dataset.

This dataset is as same as the Figure 1. In real world, this can be described as students' grade in two courses, the two grades like two features of a student, and it can plot in a figure. We can classify these students in two groups, and this can also indicate the relation of two courses. In this part, I will try using different method to classify the dataset.

Dataset url: https://web.stanford.edu/~hastie/ElemStatLearn/data.html.

ESL.mixture	list [8]	List of length 8
x	double [200 x 2]	2.5261 0.3670 0.7682 0.6934 -0.0198 2.1965 0.3211 0.0315 0.7175 0.7772 ...
y	double [200]	0 0 0 0 0 0 ...
xnew	double [6831 x 2] (S3: matrix)	-2.6 -2.5 -2.4 -2.3 -2.2 -2.1 -2.0 -2.0 -2.0 -2.0 -2.0 -2.0 ...
prob	double [6831]	3.55e-05 3.05e-05 2.63e-05 2.27e-05 1.96e-05 1.69e-05 ...
marginal	double [6831]	6.65e-15 2.31e-14 7.62e-14 2.39e-13 7.15e-13 2.03e-12 ...
px1	double [69]	-2.6 -2.5 -2.4 -2.3 -2.2 -2.1 ...
px2	double [99]	-2.00 -1.95 -1.90 -1.85 -1.80 -1.75 ...
means	double [20 x 2]	-0.2534 0.2667 2.0965 -0.0613 2.7035 2.3772 1.7415 0.3712 1.2334 -0.2087 ...

Pseudo code
One of the most important algorithms in SVM is the SMO algorithm
SMO algorithm:

Create an alpha vector and initialize it to the 0 vector;
When the number of iterations is less than the maximum number of iterations (outer loop):
For each data vector in the data set (inner loop):
If the data vector can be optimized:
Randomly select another data vector;
Optimize these two vectors simultaneously;
If neither vector can be optimized, exit the inner loop;
If all vectors are not optimized, increase the number of iterations and continue the next loop;

I will implement the liner SVM algorithm in this project, then I will use the code to generate the figure like the Figure 2. In this part, I will learn how to build the liner SVM algorithm.