Comp9417 Final Exam

# Question 1:

## a)

At first, we assume A, B, C and D are mutually independent.

According to the Bayes Assumption, we have:

From the dataset we know that

And the count vector for positive is and that for negative is.

Then the estimated parameter vectors are:

According to the formula given:

Thus,

## b)

With smooth, the estimated parameter vectors are:

Hence,

## c)

Under the Bernoulli distribution, the estimated parameter vectors are:

And we have the bit vector

Thus,

Therefore,

## d)

With smooth, the estimated parameter vectors are:

And we have the bit vector

Thus,

Therefore,

# Question 2

## a)

According to the question, , thus

Then we have:

and

## b)

While condition satisfied (iteration < 100 in this question):

BEGIN

Compute by

and

Then calculate

Finally update

END

## c)

Figure 2.1 Python code for gradient descent

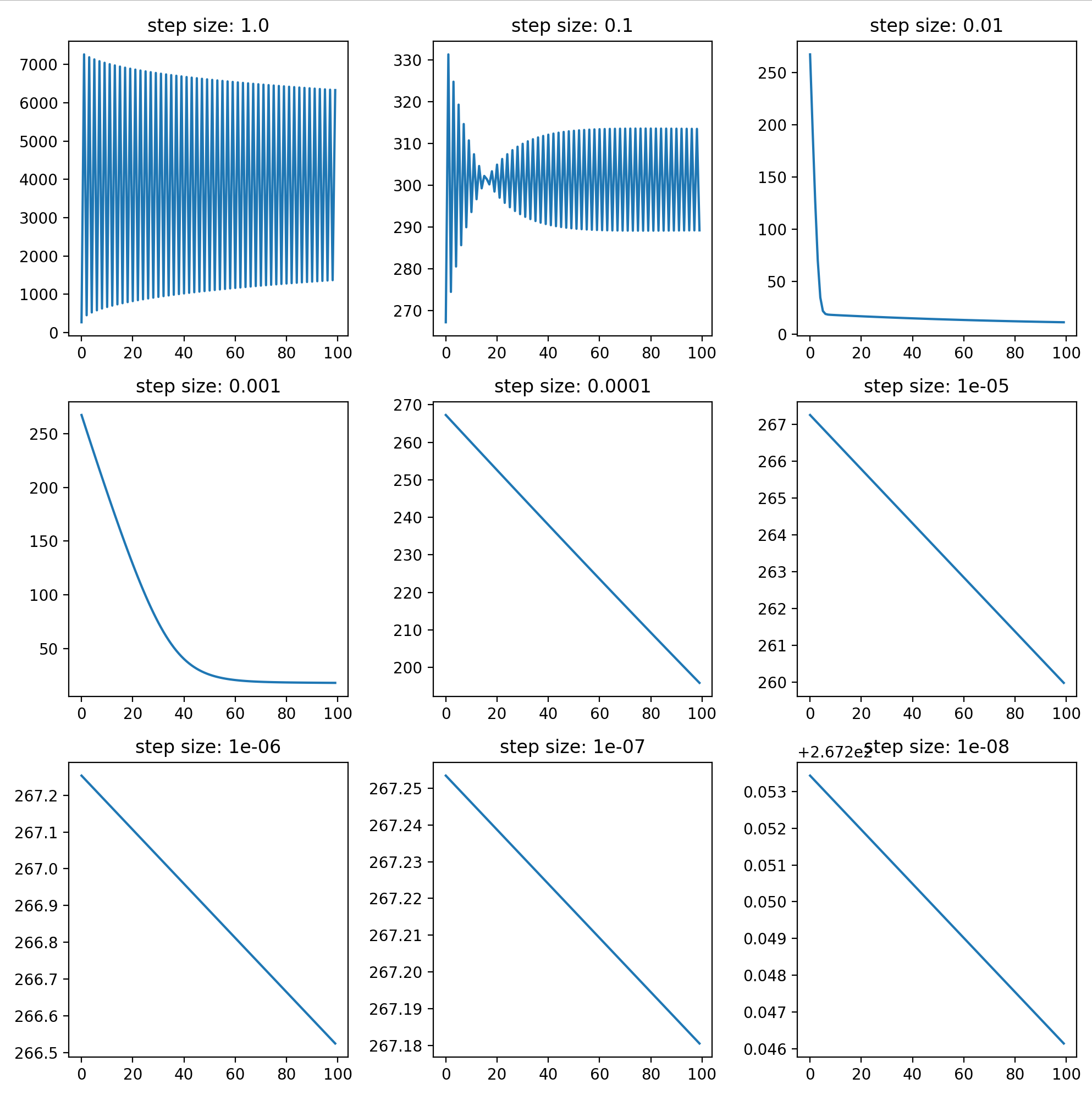


Figure 2.2 Gradient for nine different alphas

## d)

Oscillations appeared for step size of 1.0 and 0.1. And the loss decreased significantly when while the model performs best with . However, when step size became smaller, no obvious change happened to the loss.

The reason for the oscillation is that the learning rate is too high so that the parameters are over adjusted and when it becomes low no significant adjustment could be made. If we change iteration size to a large number (for example 10000) we can observe a significant decrease in losses of a smaller learning rate.

## e)

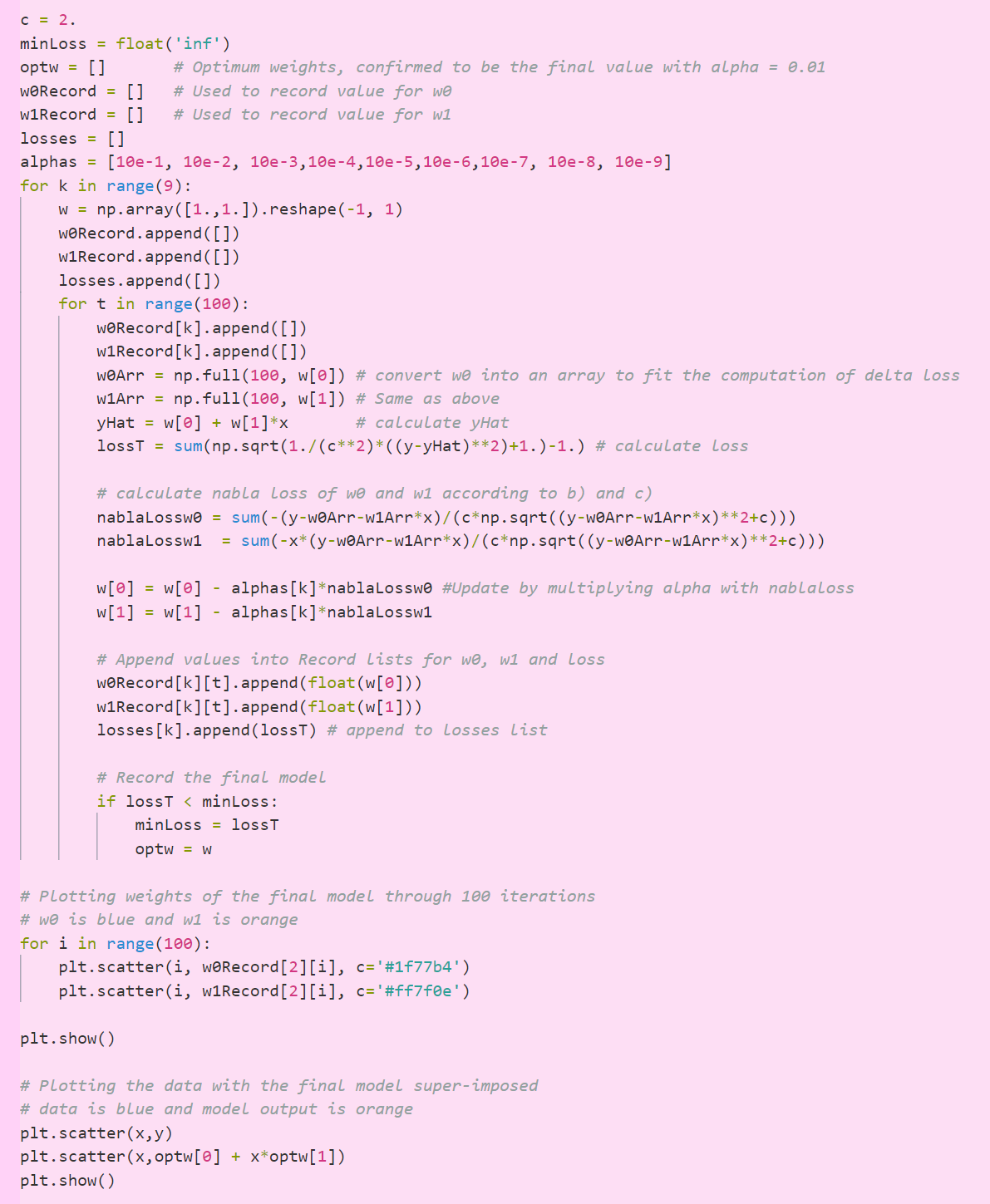
The optimal step-size is 0.01, and the final model is

Figure 2.3 Code with two plots

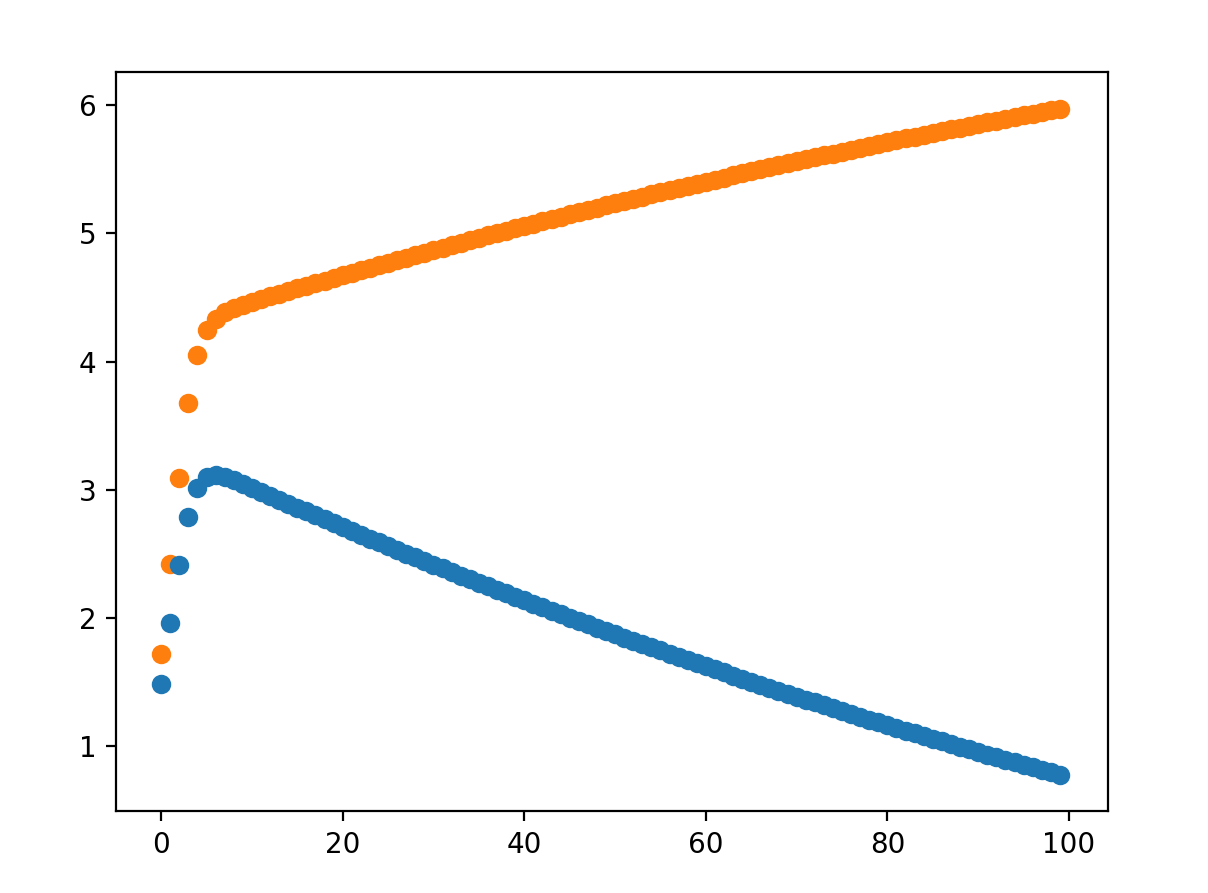
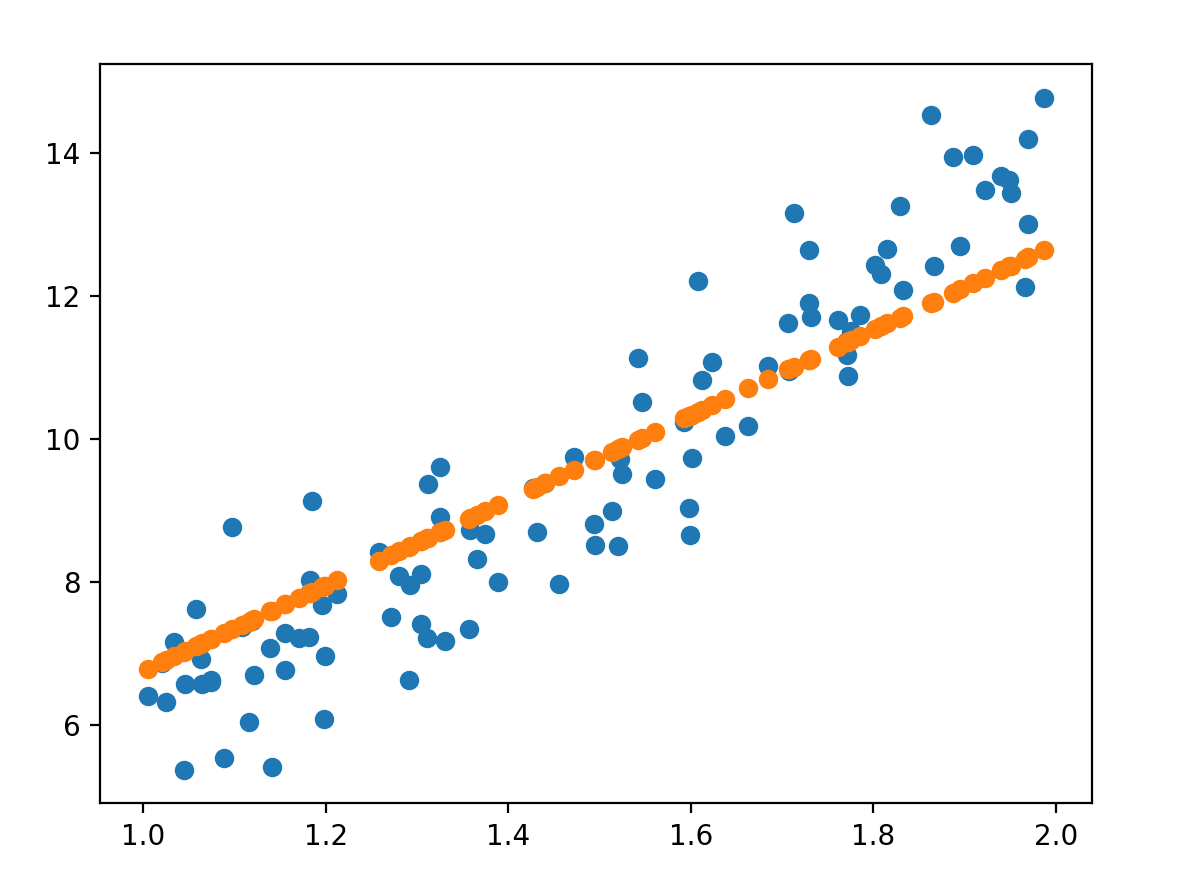
Figure 2.4 Plot of final model for w0 and w1 over 100 iterations (w0: blue, w1: orange).

Figure 2.5 Plot of the data with the final model super-imposed (data: blue, predict: orange).

## f)

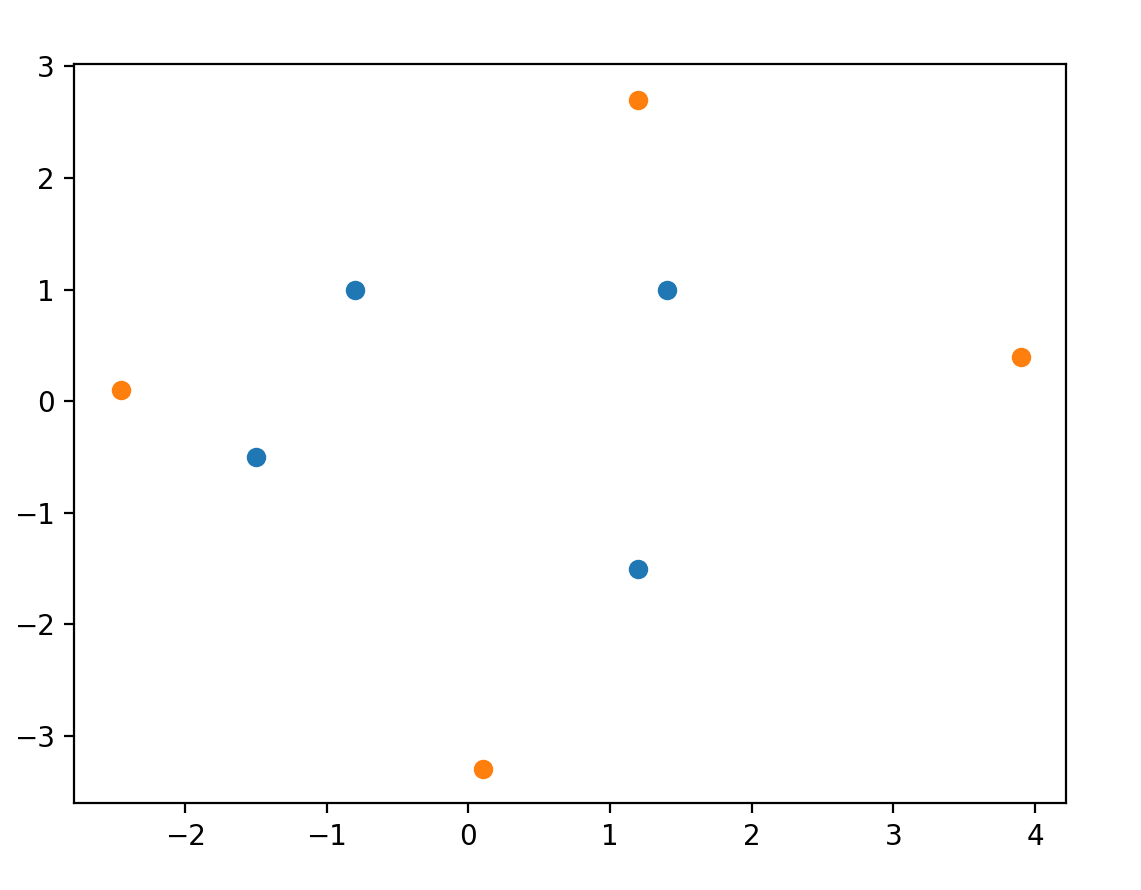
The value for optimal alpha value increases as c increases, vice versa.

A larger value of c is chosen when we want to reduce the influence of the squared error and reduce the numerical value of SE, vice versa.

The final c chosen should generate a more accurate model and should also produce a more intuitive loss value for human brain.

## Question 3

## a)

Figure 3.1 plot of data (blue: data for y=1, orange: data for y=-1)

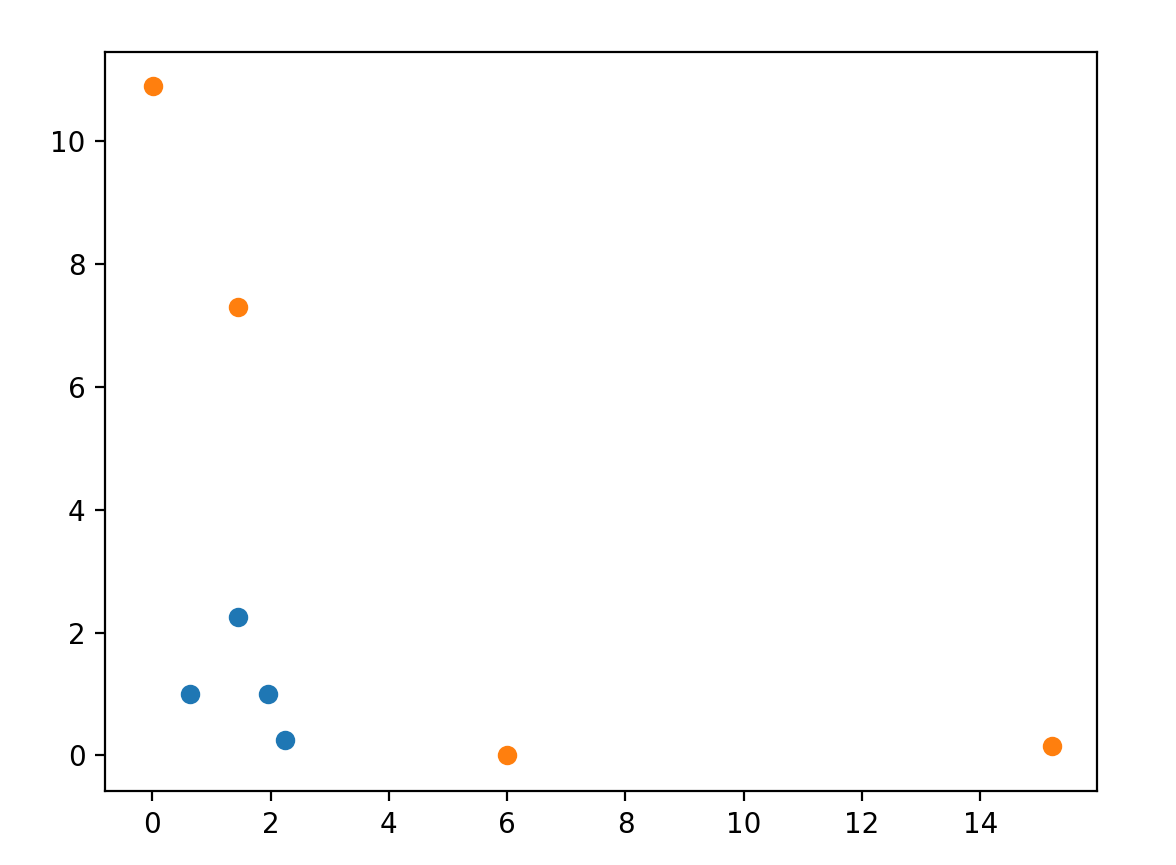
The simplest polynomial kernel is

in which .

So, the feature function could be defined as

## b)

According to a), the feature representation is in .

And the subset chosen is . The plot is shown in the following figure, and the transformed data is linear separable

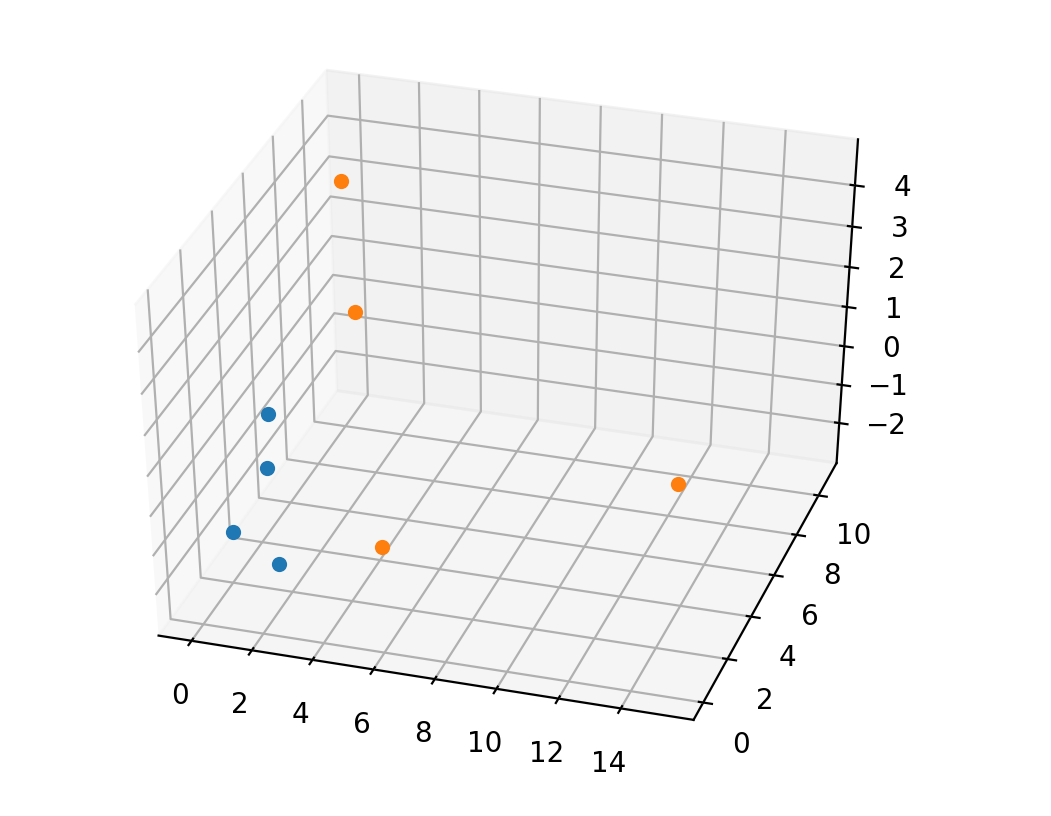
Figure 3.2 plot of the subset of the 3-dimensional vector (blue: data for y=1, orange: data for y=-1)

Figure 3.3 plot of transformed data (blue: data for y=1, orange: data for y=-1)

## c)

The table below is the table outlining all updates of the weight vector with the iteration. Notice that the training process terminates after 72 iterations and the weight vector is

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| iter | w0 | w1 | w2 | w3 |
| 1 | 1 | 1 | 1 | 1 |
| 2 | 0.8 | -2.042 | 0.968 | 0.558765 |
| 3 | 1 | -1.65 | 1.168 | 0.954745 |
| 4 | 0.8 | -1.652 | -1.01 | 1.048083 |
| 5 | 0.8 | -1.652 | -1.01 | 1.048083 |
| 6 | 0.8 | -1.652 | -1.01 | 1.048083 |
| 7 | 1 | -1.202 | -0.96 | 1.260215 |
| 8 | 1.2 | -0.914 | -0.51 | 0.751098 |
| 9 | 1.4 | -0.786 | -0.31 | 0.524824 |
| 10 | 1.4 | -0.786 | -0.31 | 0.524824 |
| 11 | 1.4 | -0.786 | -0.31 | 0.524824 |
| 12 | 1.4 | -0.786 | -0.31 | 0.524824 |
| 13 | 1.2 | -1.074 | -1.768 | -0.39159 |
| 14 | 1.2 | -1.074 | -1.768 | -0.39159 |
| 15 | 1.4 | -0.624 | -1.718 | -0.17945 |
| 16 | 1.6 | -0.336 | -1.268 | -0.68857 |
| 17 | 1.6 | -0.336 | -1.268 | -0.68857 |
| 18 | 1.6 | -0.336 | -1.268 | -0.68857 |
| 19 | 1.8 | 0.056 | -1.068 | -0.29259 |
| 20 | 1.8 | 0.056 | -1.068 | -0.29259 |
| 21 | 1.8 | 0.056 | -1.068 | -0.29259 |
| 22 | 1.6 | -1.1445 | -1.07 | -0.22329 |
| 23 | 1.8 | -0.6945 | -1.02 | -0.01116 |
| 24 | 2 | -0.4065 | -0.57 | -0.52028 |
| 25 | 2 | -0.4065 | -0.57 | -0.52028 |
| 26 | 2 | -0.4065 | -0.57 | -0.52028 |
| 27 | 2.2 | -0.0145 | -0.37 | -0.1243 |
| 28 | 2.2 | -0.0145 | -0.37 | -0.1243 |
| 29 | 2.2 | -0.0145 | -0.37 | -0.1243 |
| 30 | 2 | -1.215 | -0.372 | -0.055 |
| 31 | 2.2 | -0.765 | -0.322 | 0.157129 |
| 32 | 2.4 | -0.477 | 0.128 | -0.35199 |
| 33 | 2.4 | -0.477 | 0.128 | -0.35199 |
| 34 | 2.4 | -0.477 | 0.128 | -0.35199 |
| 35 | 2.4 | -0.477 | 0.128 | -0.35199 |
| 36 | 2.2 | -0.479 | -2.05 | -0.25865 |
| 37 | 2.2 | -0.479 | -2.05 | -0.25865 |
| 38 | 2.2 | -0.479 | -2.05 | -0.25865 |
| 39 | 2.2 | -0.479 | -2.05 | -0.25865 |
| 40 | 2.4 | -0.191 | -1.6 | -0.76777 |
| 41 | 2.4 | -0.191 | -1.6 | -0.76777 |
| 42 | 2.4 | -0.191 | -1.6 | -0.76777 |
| 43 | 2.6 | 0.201 | -1.4 | -0.37179 |
| 44 | 2.6 | 0.201 | -1.4 | -0.37179 |
| 45 | 2.6 | 0.201 | -1.4 | -0.37179 |
| 46 | 2.4 | -0.9995 | -1.402 | -0.30249 |
| 47 | 2.6 | -0.5495 | -1.352 | -0.09036 |
| 48 | 2.8 | -0.2615 | -0.902 | -0.59948 |
| 49 | 2.8 | -0.2615 | -0.902 | -0.59948 |
| 50 | 2.8 | -0.2615 | -0.902 | -0.59948 |
| 51 | 2.8 | -0.2615 | -0.902 | -0.59948 |
| 52 | 2.8 | -0.2615 | -0.902 | -0.59948 |
| 53 | 2.8 | -0.2615 | -0.902 | -0.59948 |
| 54 | 2.6 | -1.462 | -0.904 | -0.53018 |
| 55 | 2.8 | -1.012 | -0.854 | -0.31805 |
| 56 | 2.8 | -1.012 | -0.854 | -0.31805 |
| 57 | 2.8 | -1.012 | -0.854 | -0.31805 |
| 58 | 2.8 | -1.012 | -0.854 | -0.31805 |
| 59 | 3 | -0.62 | -0.654 | 0.077933 |
| 60 | 3 | -0.62 | -0.654 | 0.077933 |
| 61 | 3 | -0.62 | -0.654 | 0.077933 |
| 62 | 3 | -0.62 | -0.654 | 0.077933 |
| 63 | 3 | -0.62 | -0.654 | 0.077933 |
| 64 | 3 | -0.62 | -0.654 | 0.077933 |
| 65 | 3 | -0.62 | -0.654 | 0.077933 |
| 66 | 3 | -0.62 | -0.654 | 0.077933 |
| 67 | 3 | -0.62 | -0.654 | 0.077933 |
| 68 | 3 | -0.62 | -0.654 | 0.077933 |
| 69 | 3 | -0.62 | -0.654 | 0.077933 |
| 70 | 3 | -0.62 | -0.654 | 0.077933 |
| 71 | 3 | -0.62 | -0.654 | 0.077933 |
| 72 | 3 | -0.62 | -0.654 | 0.077933 |

Figure 3.4 table for all updates of weight vector

The following table demonstrates the correcteness:

|  |  |  |
| --- | --- | --- |
|  |  |  |
| (-0.8 1.) | (0.64 1. -1.13137085) | 1.86102915 |
| (3.9 0.4) | (15.21 0.16 2.20617316) | 6.362906843 |
| (1.4 1.) | (1.96 1. 1.97989899) | 1.285098987 |
| (0.1 -3.3) | (0.01 10.89 -0.46669048) | 4.164630476 |
| (1.2 2.7) | (1.44 7.29 4.58205194) | 2.303368058 |
| (-2.45 0.1) | (6.0025 0.01 -0.34648232) | 0.755092323 |
| (-1.5 -0.5) | (2.25 0.25 1.06066017) | 1.524160172 |
| (1.2 -1.5) | (1.44 2.25 -2.54558441) | 0.437315588 |

Figure 3.5 table for correctness

Screen shot for code is in next page.

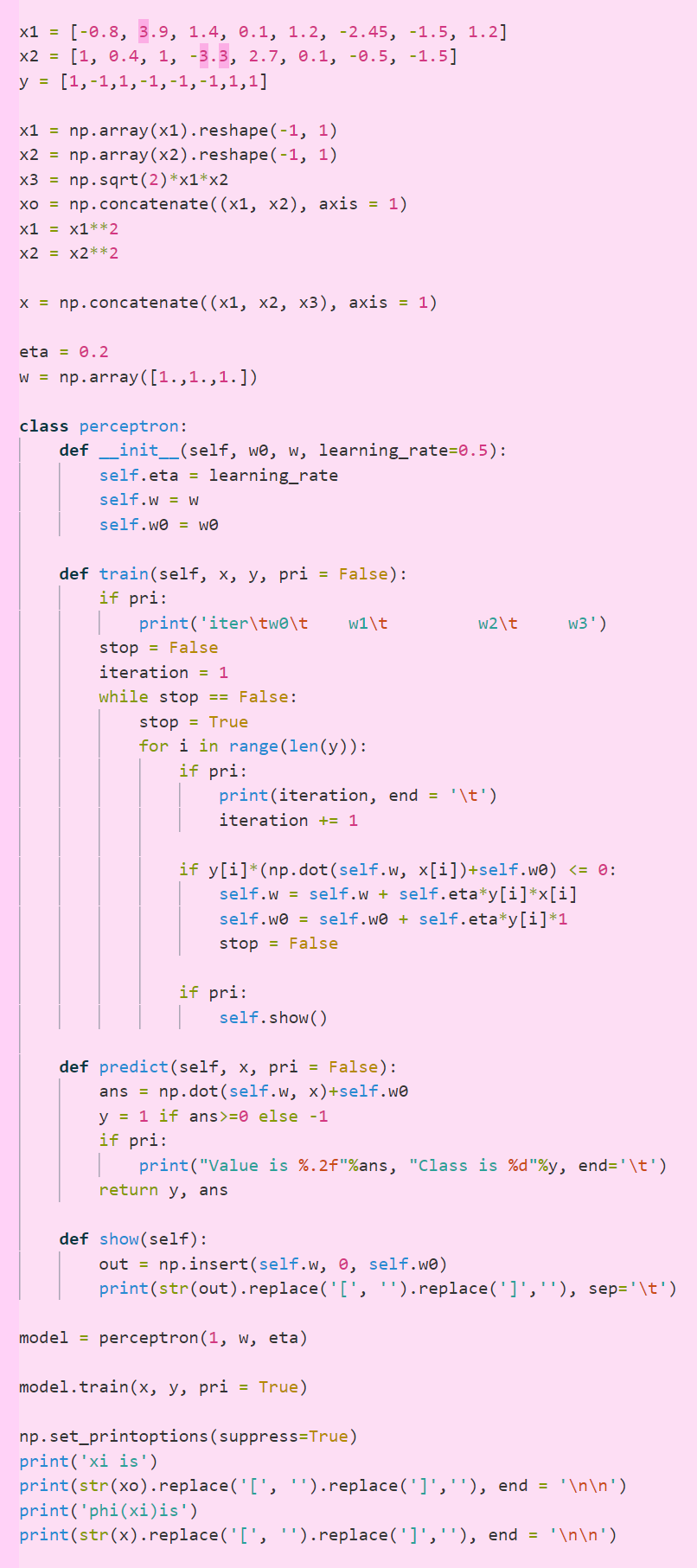


Figure3.6 Code for this question

## d)

From the question, we have:

Therefore,

## e)

The claim is true.

Because the input vectors for positive class and negative class are symmetric along a straight line and there are four points for each class.

From the symmetrical characteristic, we can state that when the model correctly classify data point for the positive class, it means that it can correctly classify data for the negative model. For example, is classified as positive which means that

Then we have

which means that when , we have .

Thus, if is classified as positive, must be classified as negative by the same model.

From the algorithm we can know that after a mistake, the weight would be adjusted and the model would be correctly classified then. And, when a data point is correctly classified, no change would be made.

Therefore, after 4 mistakes, the remaining data would not make mistakes anymore.

## Question 4

## a)

The data is linearly separable

## b)

According to the question, we have:

And we have

Matrix multiplication for a 7-dimensional matrix is impossible for hand writing, so we must ellipses some of the points.

By observation, the line separating could separate the whole dataset appropriately. So, we calculate by those samples.

Thus, the new x and y are:

Then calculate the helper matrices:

Hence,

Then, we have

From, we can derive that

Substitute by we get:

The partial derivatives are:

Solve to get

while

Thus

## c)

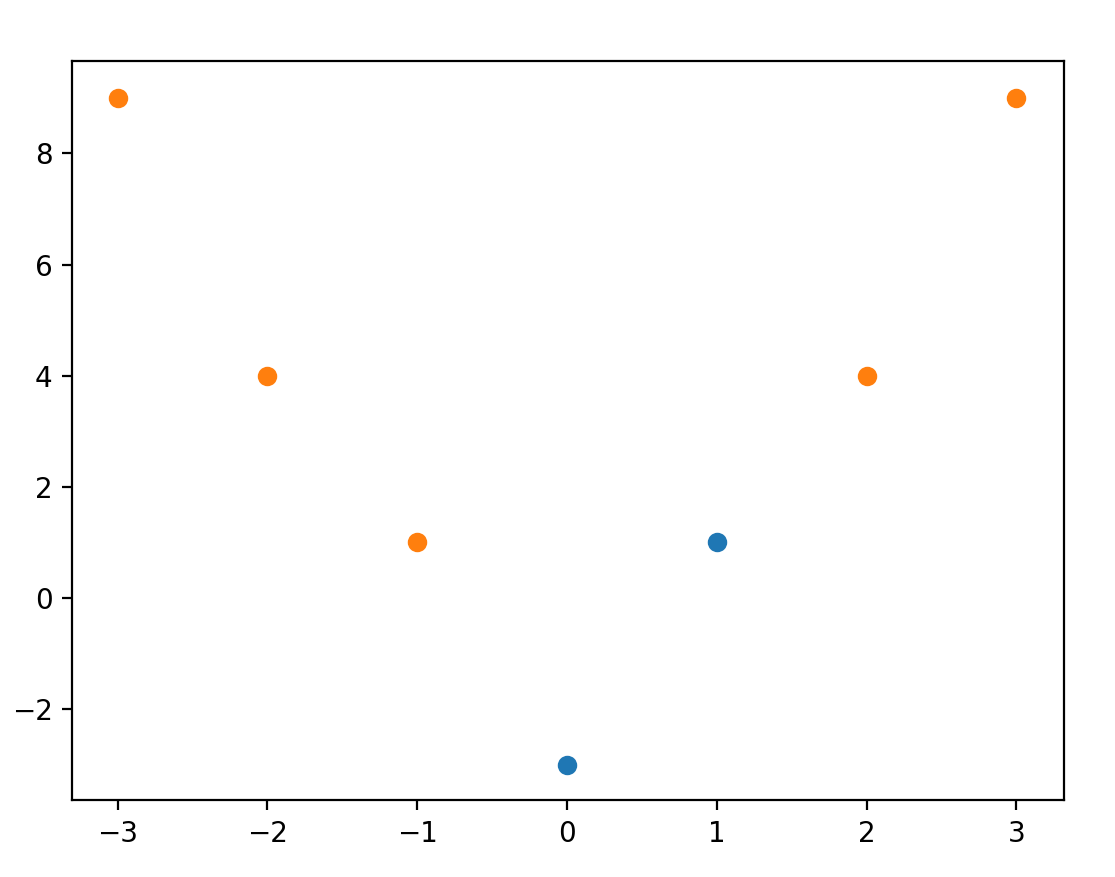
To calculate **w**:

According to b), are support vectors.

Solve t:

The margin is

Use w and -t to predict by

Figure 4.1 superimposed graph (blue for class=1, orange for class=-1)

From the figure above, we can conclude that the model perfectly fit the data.

## d)

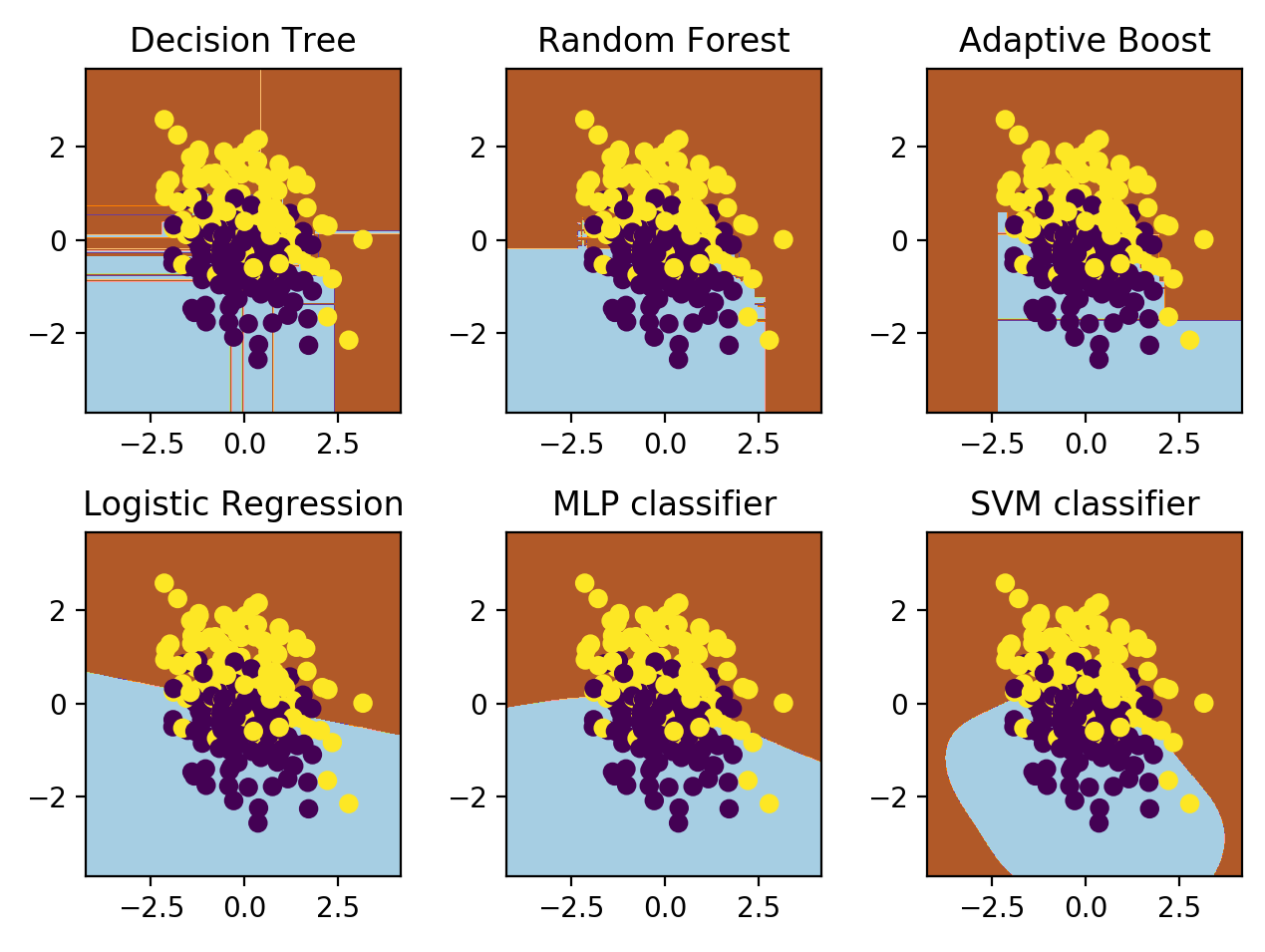
Without kernel, a linear classifier could never represent non-linear functions. But if appropriate kernel applied, the classifier then becomes linear function of polynomials of input data, then the functions may be represented.

## e)

Kernel trick applies the concept of kernel without calculating feature vectors using . In contrast, it calculates the similarity of input data and output data by substitution. The key idea is to use linear kernel calculation to replace high dimension calculations.

# Question 5

## a)

Figure 5.1 Decision boundary for different classifiers

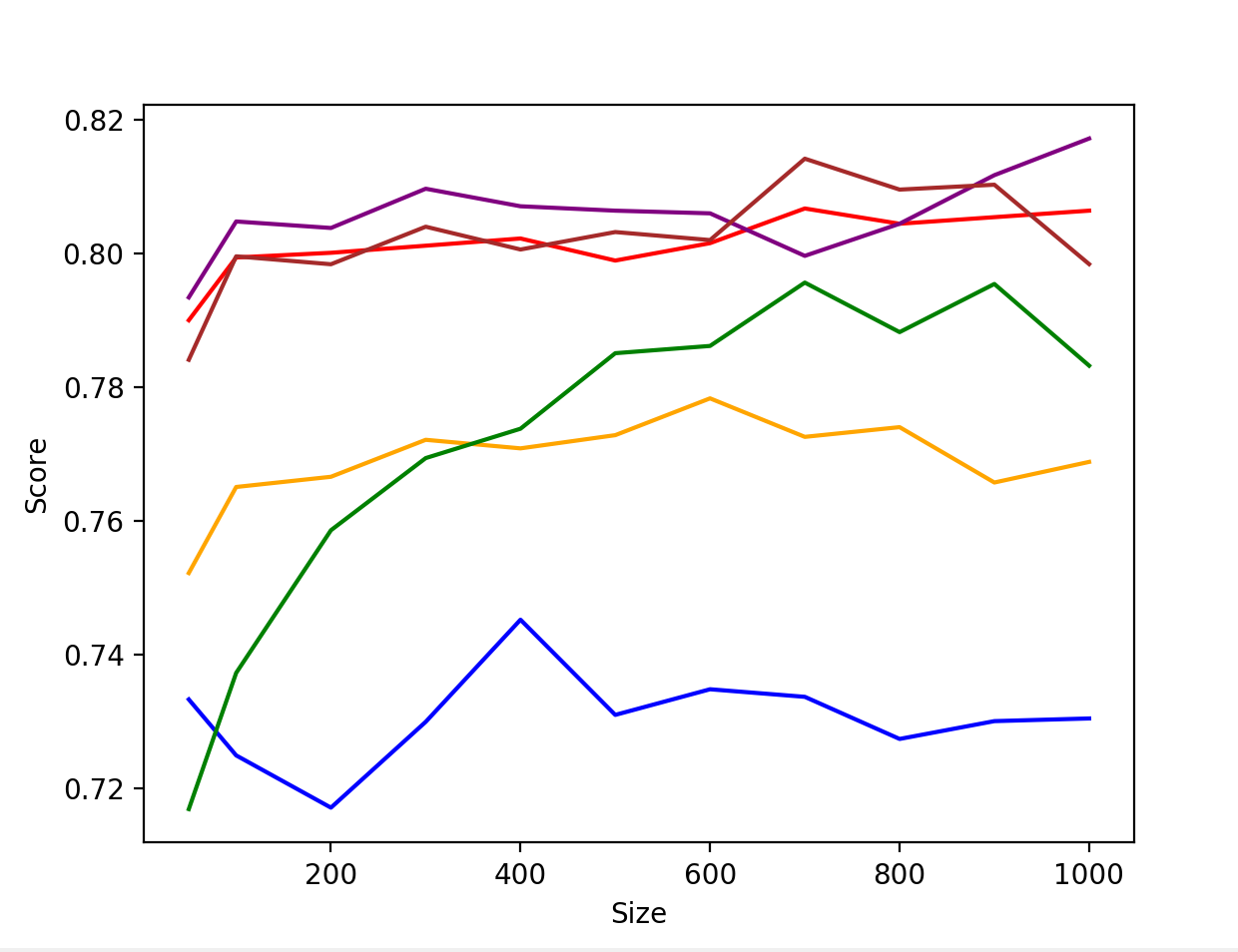
b)

Figure 5.2 Score comparation of different models with different train size

It is obvious that the decision tree has worse performance while Random Forest and AdaBoost models begin with low accuracy but gradually fit the dataset as the train size grows. The top performers are Logic Regression, Neural Network and SVM, and the Neural Network usually performs better than the others. But the difference is not significant among those three models.

## C)

Figure 5.3 Time cost of each model for different train size

It is observed that the time costs of Decision Tree, SVM and Logistic regression are constantly low (around 0) and independent from train size. While time costs of AdaBoost and Random Forest are relative higher with 0.5 and 1.0 respectively. Notice that the cost of AdaBoost is uniform while that of Random Forest increases slightly with size accordingly. However, time used by Neural network grows linearly as the size increases.