CSE250B Homework 5

Qiao Zhang, A53095965

February 17, 2016

1 Voting perceptrons

1.1

The final decision boundary is not linear.
For plotting the non-linear decision boundary, I choose 10000 points on the 2-D plane and classify them into 2 categories.

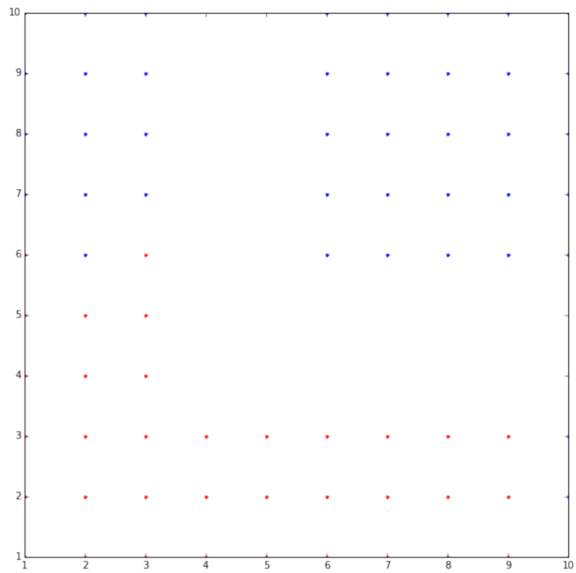

Figure 1: classified points, T = 10

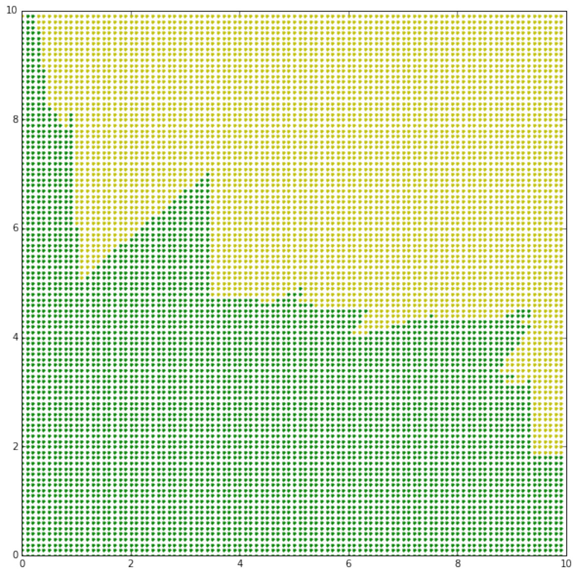

Figure 2: decision boundary, T = 10

1.2 downsample

Pseudo code

l = 1, $c_l = 0$, $w_l = \vec{0}$
repeat T times:
    randomly shuffle (X, Y)
    for every $X^{(i)}$, $Y^{(i)}$:
    if L different values of w are stored:
        combine the former L-1 w into one weighted average result
        combine the former L-1 c into one sum
        keep the last item of w and c
    if $X^{(i)}$, $Y^{(i)}$ is misclassified:
        $w_{l+1} = w_l + Y^{(i)}X^{(i)}$
        $c_{l+1} = 1$
        $l = l + 1$
    else:
        $c_l = c_l + 1$

Performance

Since the length of w in part a is less than 400 and L = 400 here in part b, we can simply compare the result shown in part a and b to assess the downsampling.

From the image, we can find that downsampling does not hamper the performance. Actually it classifies the points on the top left corner better partially as a result of increase in T.

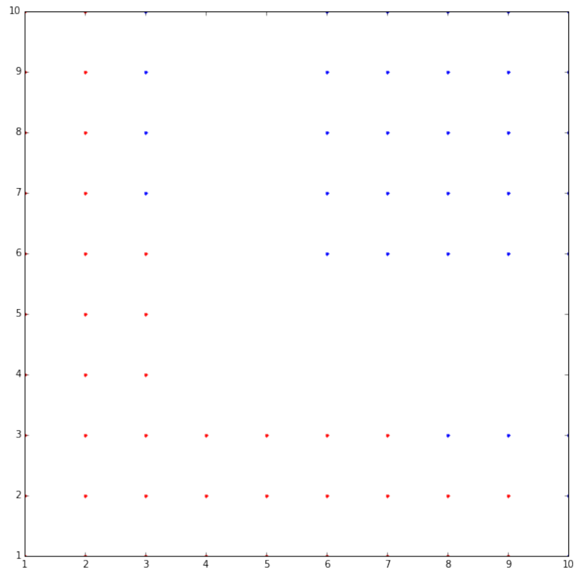

Figure 3: classified points, T = 20, L = 400

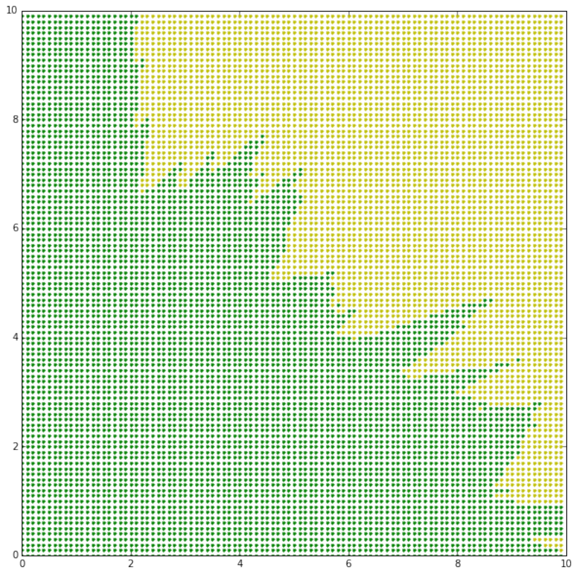

Figure 4: decision boundary, T = 20, L = 400

1.3

Pseudo code

l = 1, $c_l = 0$, $w_l = \vec{0}$
repeat T times:
    randomly shuffle (X, Y)
    for every $X^{(i)}$, $Y^{(i)}$:
    if $X^{(i)}$, $Y^{(i)}$ is misclassified:
        $w_{l+1} = w_l + Y^{(i)}X^{(i)}$
        $c_{l+1} = 1$
        combine the two existing w into one using average perceptron and store at the first place
        combine the two existing c into one using summation and store at the first place
        append the new w_{l+1} to the tail of the list w
        append the new c_{l+1} to the tail of the list c
        $l = l + 1$
    else:
        $c_l = c_l + 1$

Performance

From the result we can see that the average perceptron algorithm helps us obtain a more linear decision boundary.

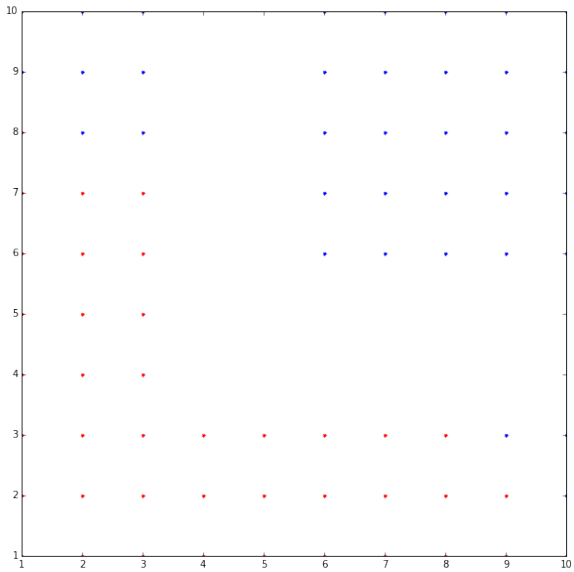

Figure 5: classified points, T = 10

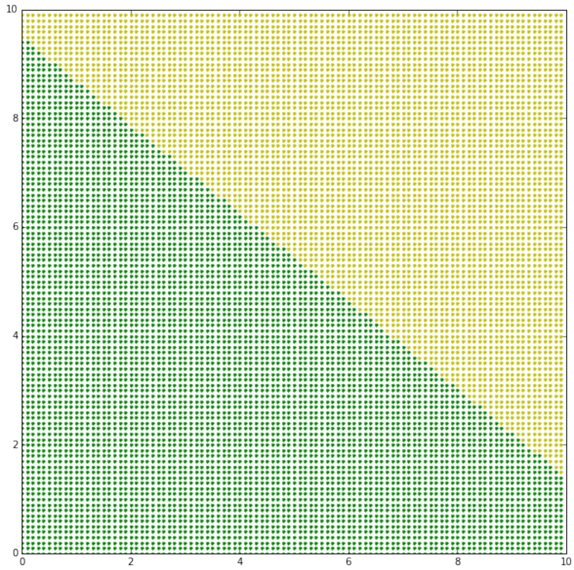

Figure 6: decision boundary, T = 10

2 Kernelized perceptrons

2.1 Quadratic kernel – data1.txt

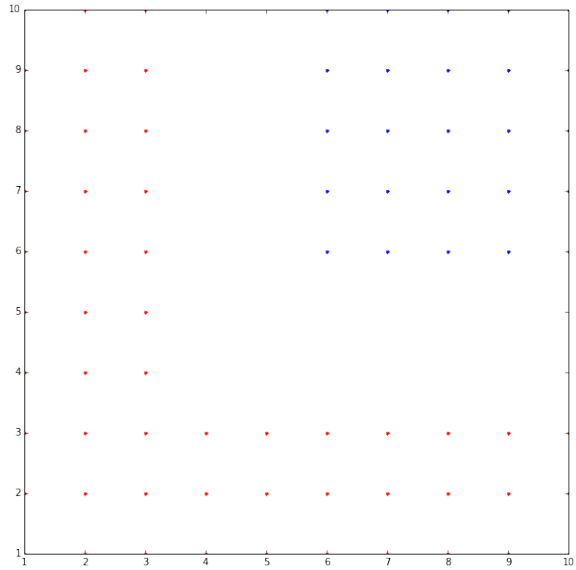

Figure 7: classified samples, T = 1

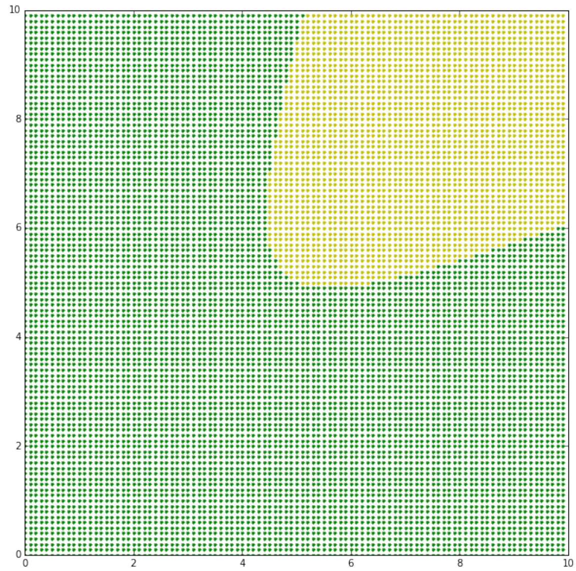

Figure 8: decision boundary, T = 1

2.2 Quadratic kernel – data2.txt

Quadratic kernel is not suitable for this situation.

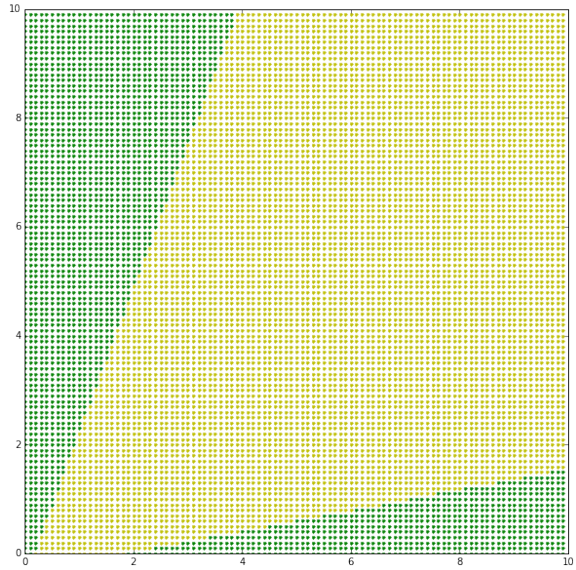

Figure 9: decision boundary, T = 5

2.3 RBF kernel – data1.txt

Generally speaking, RBF kernel performs better than quadratic kernel on both data set 1 and data set 2.

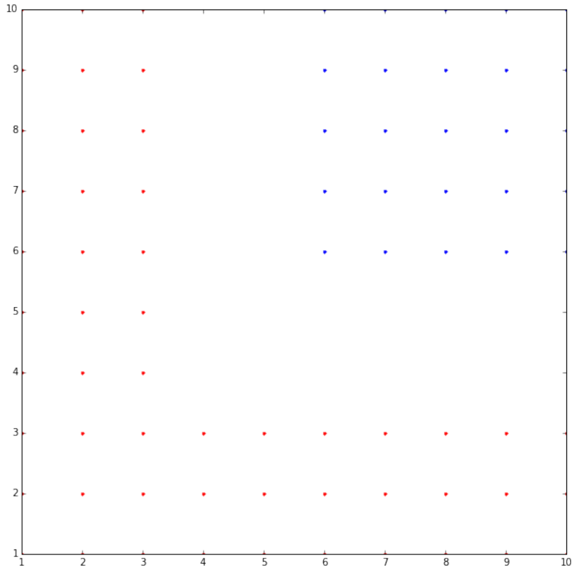

Figure 10: classified samples, T = 1, sigma = 2

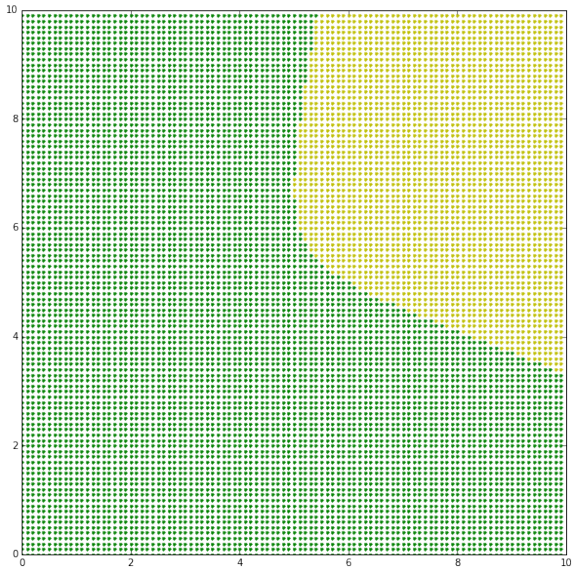

Figure 11: decision boundary, T = 1, sigma = 2

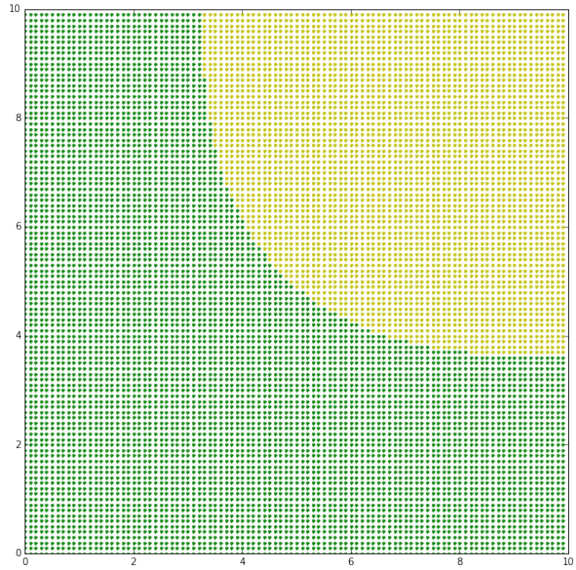

Figure 12: decision boundary, T = 1, sigma = 5

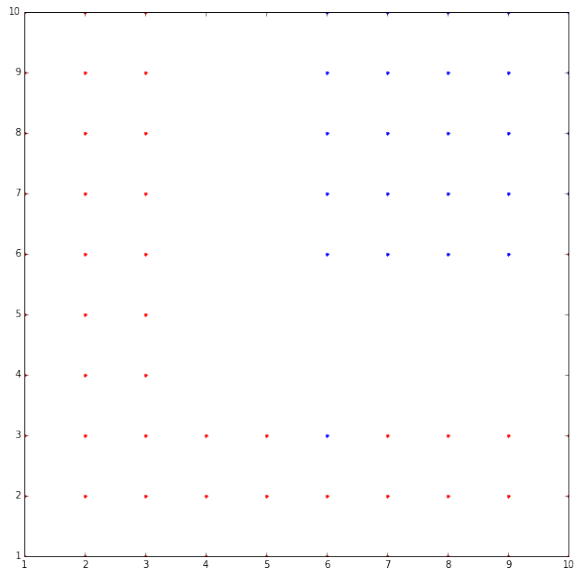

Figure 13: classified samples, T = 1, sigma = 0.2

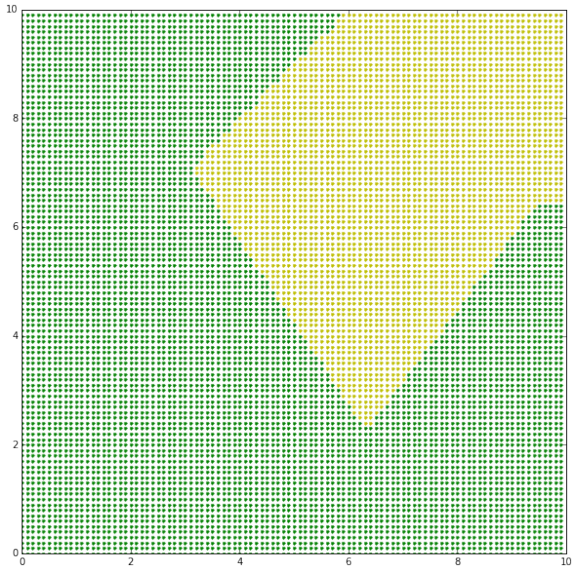

Figure 14: decision boundary, T = 1, sigma = 0.2

2.4 RBF kernel – data2.txt

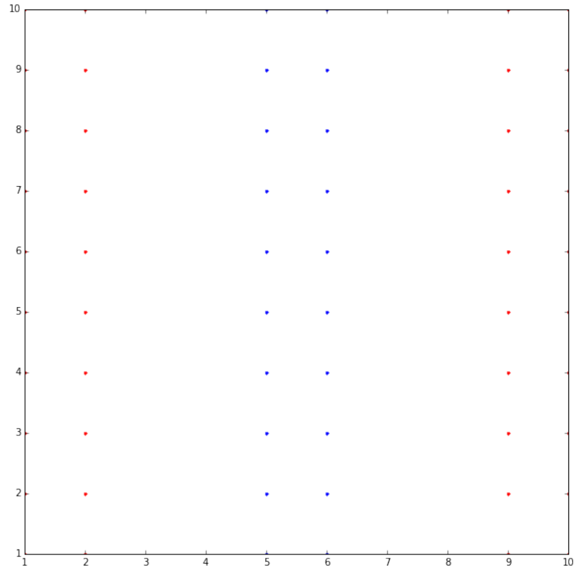

Figure 15: classified samples, T = 1, sigma = 2

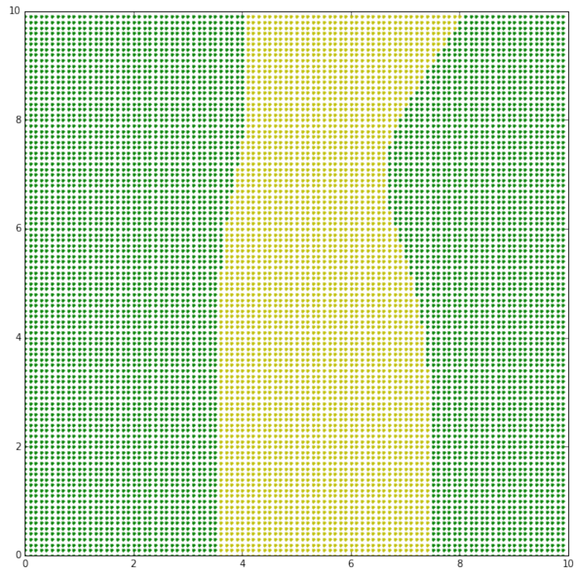

Figure 16: decision boundary, T = 1, sigma = 2

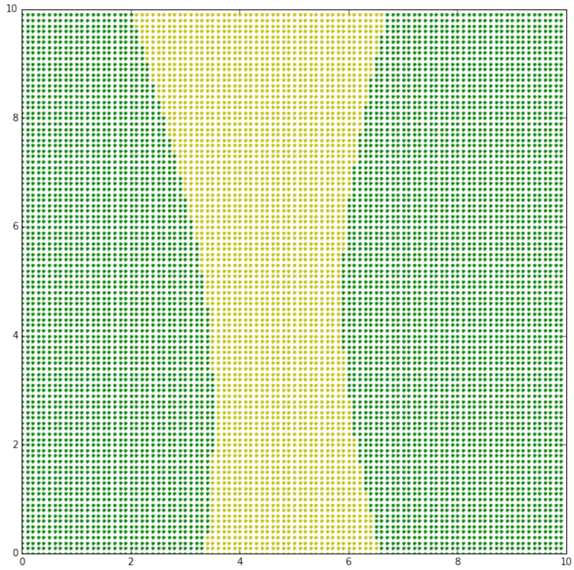

Figure 17: decision boundary, T = 1, sigma = 5

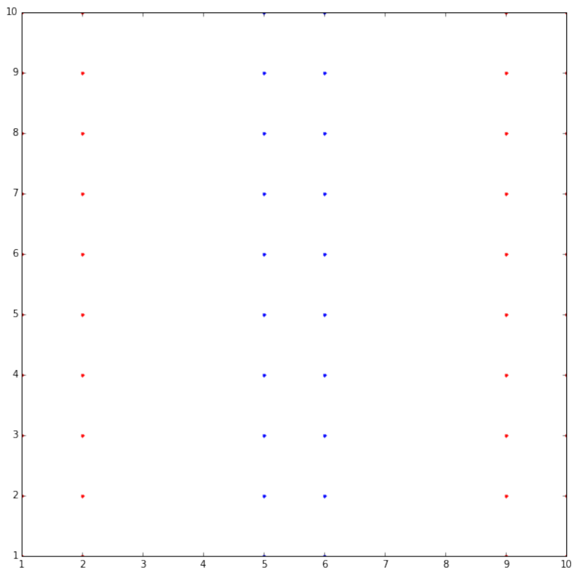

Figure 18: classified samples, T = 1, sigma = 0.2

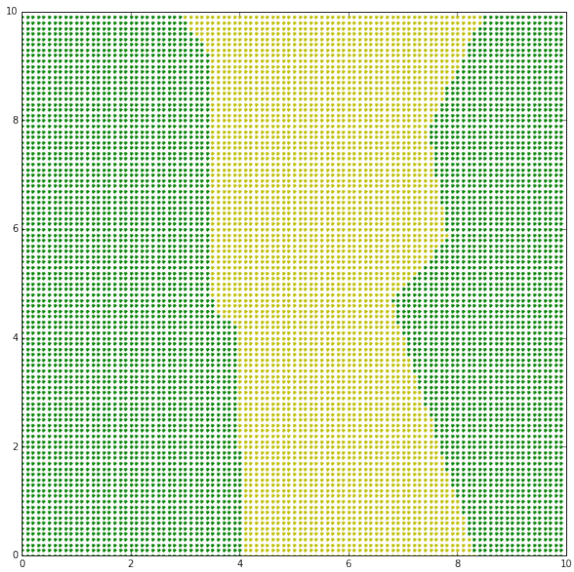

Figure 19: decision boundary, T = 1, sigma = 0.2