

CSE 250B: Assignment #5

February 16, 2016

Apoorve Dave

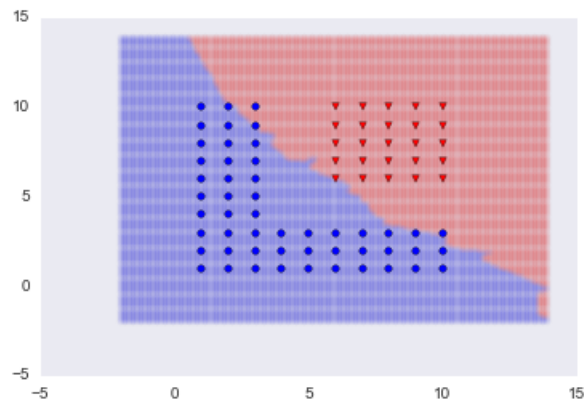
a1dave@ucsd.edu - A53103070

1 Voting Perceptrons

1.1 Voting Perceptron

The decision boundary from voting perceptron is Not Linear. Plot:

Figure 1: Voting Perceptron Decision Boundary(T=10):



1.2 Downsampled Voting Perceptron

1.2.1 Psuedocode

One way to downsample the weights as obtained by normal voting perceptron is to pick the latest L weights, discarding the previous ones. The main idea behind this approach is that in the beginning of learning, the weights of perceptrons will be premature. The earliest of learned weights therefore tend to be of lesser importance compared to the latest learned weights. Therefore, in my algorithm, I have implemented the same voting perceptron as mentioned in the HW but stored them in a FIFO queue, discarding the earliest weights as and when a new weight vector is added to the queue. Prediction is done as before, based on weighted majority vote:

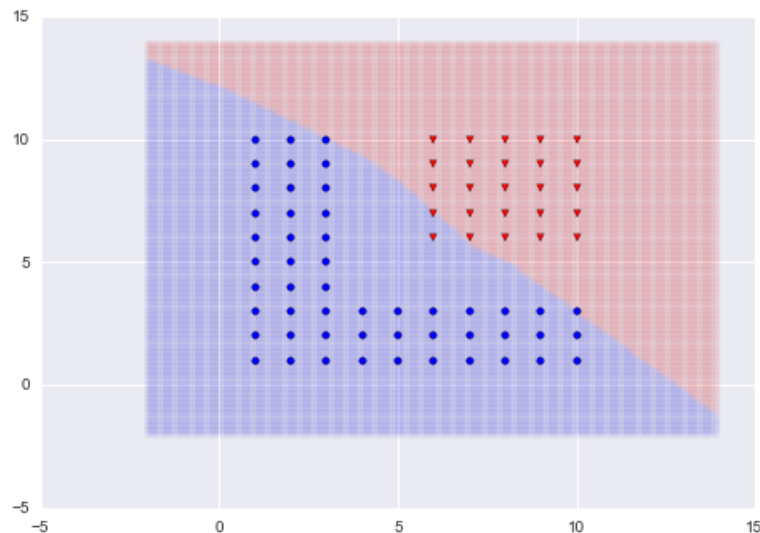
$$\text{sign} \left(\sum_{j=1}^L c_j \text{sign}(w_j \cdot x) \right)$$

Algorithm 1 Psuedocode for downsampled classifiers

```

1: Create queue weightQueue to store the latest  $L$  weight vectors, and a corresponding queue for
   survivalTimes
2: Set maxLength of queues to  $L$ . (These are FIFO. If filled, earliest element pops out)
3: Set initial weight  $w_1 = 0$  to 0 and  $c_1 = 0$ .
4: for  $i = 1$  to  $T$ : do
5:   -Randomly permute the data points
6:   for  $j = 1$  to  $n$ : do
7:      $w = \text{weightQueue.head}$  //latest value
8:     if  $(x(i); y(i))$  is misclassified by  $w$ : then
9:        $w_{\text{new}} = w + y^{(i)} \cdot x^{(i)}$ 
10:      Push  $w_{\text{new}}$  to head of weightQueue, 1 to survivalTimes. Earliest elements will be popped if
      queue is full.
11:     else
12:       Increment count for the head of survivalTimes
13:     end if
14:   end for
15: end for
16: return weightQueue and survivalTimes

```

1.2.2 Decision Boundary with Downsampled weightsFigure 2: Decision Boundary with $T=100$ and $L=20$ **1.3 Averaged Perceptron****1.3.1 Psuedocode:**

The averaged perceptron is a simpler implementation of maintaining survival times as a measure of good weight vectors. In this case, we can keep a running average of all the weight vectors seen till now. If a weight

correctly classifies a point, we add this weight vector to a cumulative weight vector. If it fails to classify the point correctly, we update the weight vector as before (i.e. $w_{l+1} = w_l + y^{(i)} * x^{(i)}$) before adding it to cumulative weight vector. We can then either average over the cumulative weight vector or just use the sum directly to classify unseen data based on

$$\text{sign}(w_{\text{cumulative}} \cdot x) \quad (\text{where } w_{\text{cumulative}} = \sum c_j w_j)$$

Algorithm 2 Pseudocode for the Averaged Perceptron

```

1: Create  $w_{\text{current}} = 0, w_{\text{cumulative}} = w_{\text{current}}$ 
2: for  $i = 1$  to  $T$ : do
3:   -Randomly permute the data points
4:   for  $j = 1$  to  $n$ : do
5:     if  $(x(i); y(i))$  is misclassified: then
6:        $w_{\text{current}} = w_{\text{current}} + y^{(i)} \cdot x^{(i)}$ 
7:     end if
8:      $w_{\text{cumulative}} = w_{\text{cumulative}} + w_{\text{current}}$ 
9:   end for
10: end for
11: return  $w_{\text{cumulative}}$ 
12: P.S. Note here score is maintained as every correct classification updates  $w_{\text{cumulative}}$  by  $w_{\text{current}}$ , otherwise updates  $w_{\text{current}}$  before adding to  $w_{\text{cumulative}}$ .

```

1.3.2 Averaged Perceptron Decision Boundary:

Figure 3: Averaged Perceptron Decision Boundary with $T=10$

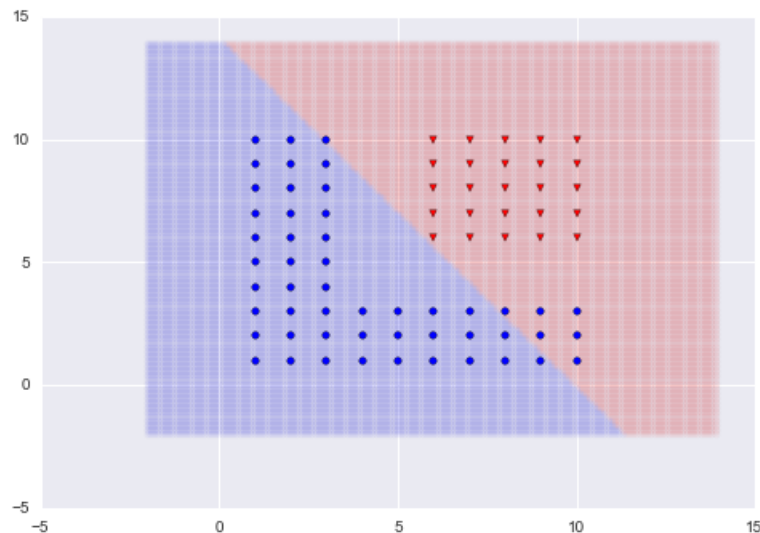
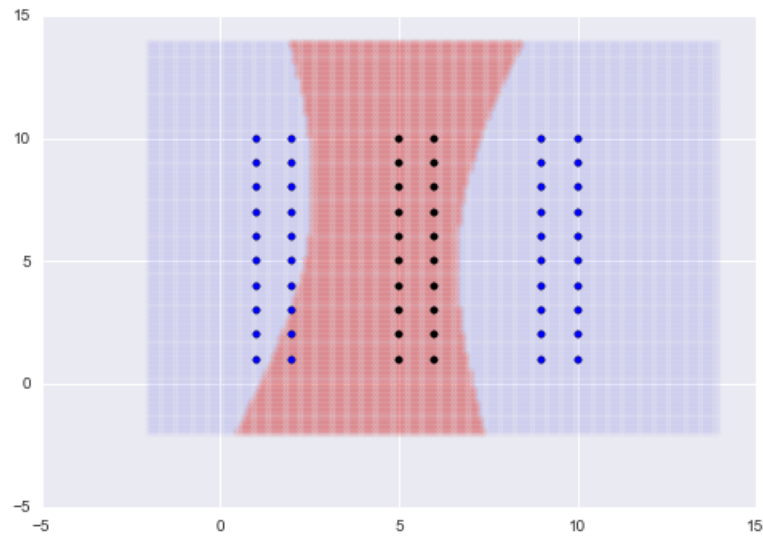


Figure 5: Data Set 2 Decision Boundary with Quadratic Kernel

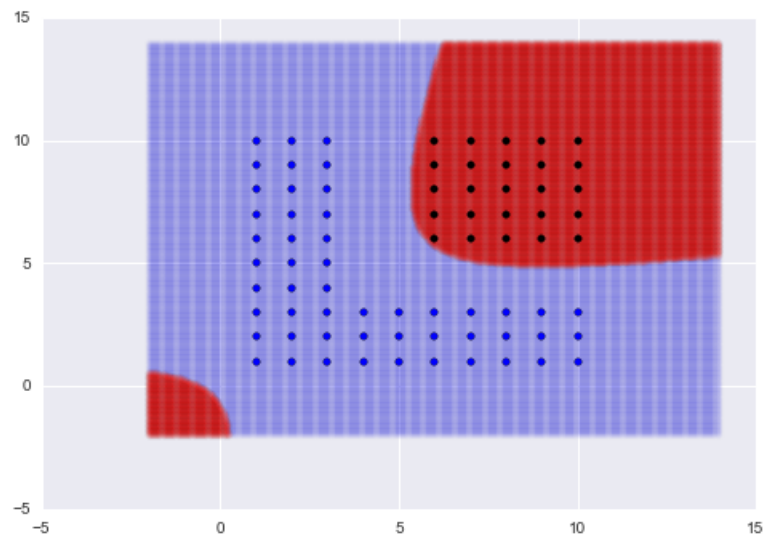


2 Kernelized Perceptrons

2.1 Quadratic Kernel

2.1.1 Data Set 1 Decision Boundary

Figure 4: Data Set 1 Decision Boundary with Quadratic Kernel



2.1.2 Data Set 2 Decision Boundary

2.2 RBF Kernel

2.2.1 Data Set 1 Decision Boundaries

Figure 6: RBF, Data Set 1, $\sigma = 0.05$

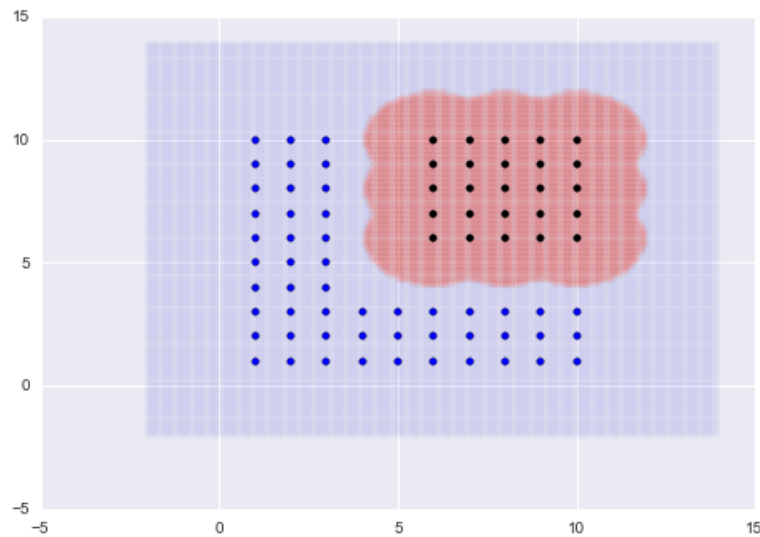


Figure 7: RBF, Data Set 1, $\sigma = 0.08$

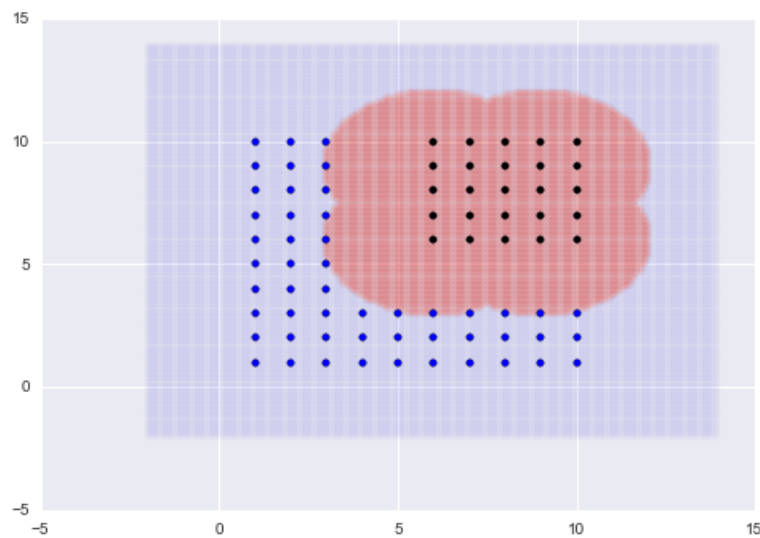
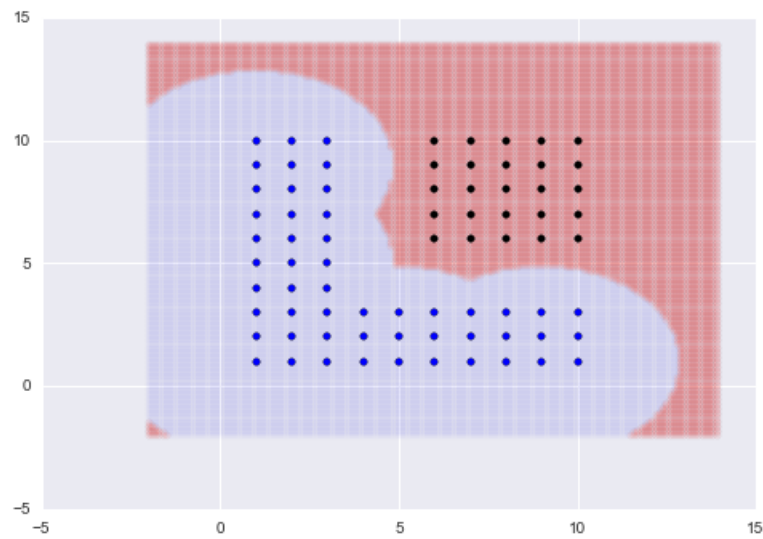


Figure 8: RBF, Data Set 1, $\sigma = 0.1$ 

2.2.2 Data Set 2 Decision Boundaries

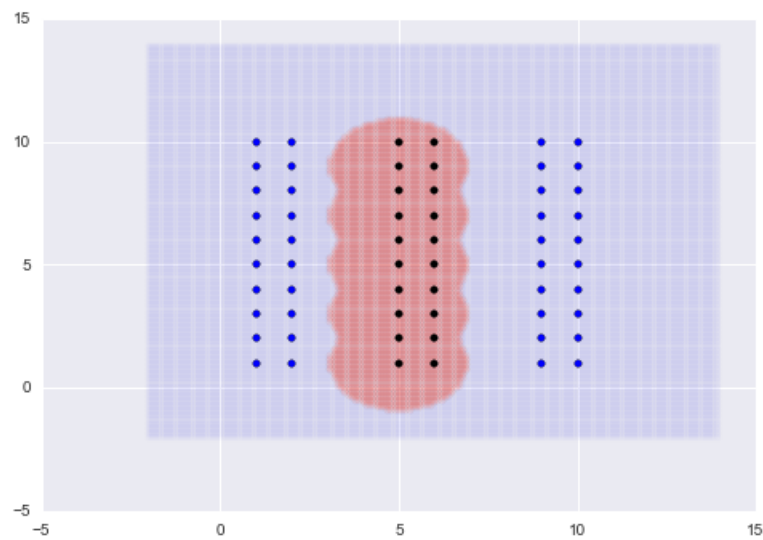
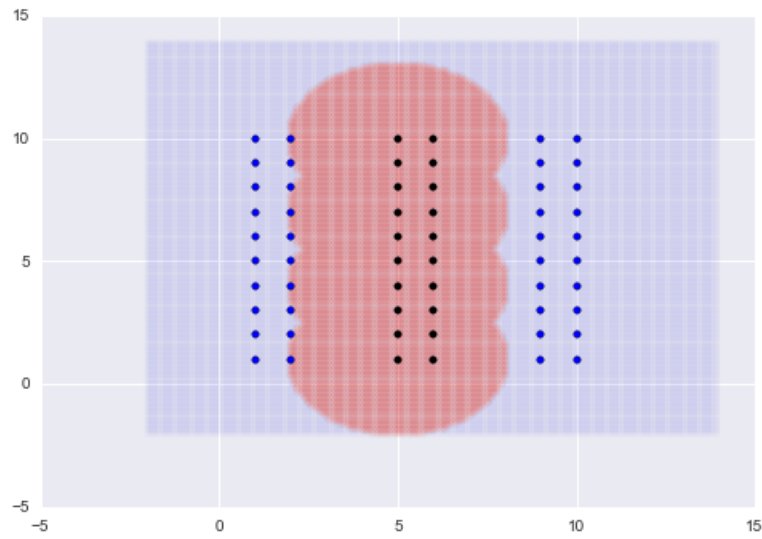
Figure 9: RBF, Data Set 2, $\sigma = 0.05$ 

Figure 10: RBF, Data Set 2, $\sigma = 0.08$ Figure 11: RBF, Data Set 2, $\sigma = 0.1$ 