Statistical Machine Learning (W4400)
Spring 2016
https://courseworks.columbia.edu

Si Kai Lee
sl3950

Homework 2

Due: Tuesday 23 February 2016

Students are encouraged to work together, but homework write-ups must be done individually and must be entirely the author's own work. Homework is due at the **beginning** of the class for which it is due. **Late homework will not be accepted under any circumstances.** To receive full credit, students must thoroughly explain how they arrived at their solutions and include the following information on their homeworks: name, UNI, homework number (e.g., HW03), and class (STAT W4400). All homework must be turned in online through Courseworks in PDF format, have a .pdf extension (not zip or other archive!), and be less than 4MB. If programming is part of the assignment, the code must be turned in in one or more .R files. Homeworks not adhering to these requirements will receive no credit. For your convenience (not required), a tex template for producing nice PDF files can be found on courseworks.

1. **Linear Classification** (20 points)
   Consider a perceptron classifier in $\mathbb{R}^2$, given by a hyperplane with orthogonal vector v_{H} and offset c, and two points $\mathbf{x}_1$ and $\mathbf{x}_2$. Suppose that

$$v_{\mathsf{H}} := \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix} \qquad c := \frac{1}{2\sqrt{2}} \qquad \mathbf{x}_1 := \begin{pmatrix} -3 \\ 0 \end{pmatrix} \qquad \mathbf{x}_2 := \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \end{pmatrix} .$$

   1. (7 points) Compute the classification result for $\mathbf{x}_1$ and $\mathbf{x}_2$.
   2. (7 points) If the classifier is given by the same pair $(v_{\mathsf{H}}, -c)$, but was trained as an SVM with margin 1, do the results change? Please explain your answer.
   3. (6 points) Which cost function does the perceptron cost function approximate, and why do we approximate it?

Solution:
1. $\operatorname{sgn}(\langle x_1, v_H\rangle - c) = \operatorname{sgn}(\frac{-3}{\sqrt{(2)}} - \frac{1}{2\sqrt{(2)}}) = -1$, $\operatorname{sgn}(\langle x_2, v_H\rangle - c) = \operatorname{sgn}(\frac{2}{2\sqrt{(2)}} - \frac{1}{2\sqrt{(2)}}) = 1$
2. Since $\langle x_1, v_H\rangle - c = \frac{-3}{\sqrt{(2)}} - \frac{1}{2\sqrt{(2)}} = \frac{-7}{2\sqrt{(2)}} = -2.82 < -1$, x_1 will be classified as -1. However, $\langle x_2, v_H\rangle - c = \frac{2}{2\sqrt{(2)}} - \frac{1}{2\sqrt{(2)}} = \frac{1}{2\sqrt{(2))}} < 1$, x_2 would fall within the 1 margin and would not be classified.
3. The perceptron cost function approximates the empirical risk which is piece-wise constant function. As such a function is not suitable for numerical optimisation, we approximate the empirical risk with a piece-wise linear function instead to enable gradient descent.

2. **Perceptron** (40 points)
   Our Perceptron implementation will use the homogeneous coordinate representation of vectors,

i.e. vectors $\mathbf{x} \in \mathbb{R}^d$ are represented by $(\mathbf{x}, 1) = (x_1, \ldots, x_d, 1)^t \in \mathbb{R}^{d+1}$. Recall that this representation turns affine hyperplanes

$$\{\mathbf{x} \in \mathbb{R}^d | \langle \mathbf{v}_\mathsf{H}, \mathbf{x} \rangle) - c = 0\}$$

with normal vector $\mathbf{v}_\mathsf{H} = (v_1, \ldots, v_d)^t$ into hyperplanes through the origin of the form

$$\{(\mathbf{x}, 1) \in \mathbb{R}^{d+1} | \, (\mathbf{v}_\mathsf{H}, -c)^t \, (\mathbf{x}, 1) = 0\} \, .$$

To work with a Perceptron, we need a two-class set of linearly separable data. For the purposes of testing the algorithm, we will random-generate this data synthetically. A sample set of n data values in d-dimensional space will be represented as a $n \times (d+1)$-matrix `S`, so each row is the (homogeneous coordinate) representation of a single data point. This matrix contains the data of both classes, so in addition, we need class labels. Class labels are represented as a vector `y` of length n, with entries in $\{-1, +1\}$.

This problem requires linearly separable data. You can download the function `fakedata.R` from the class homepage, which generates such data synthetically: If `z` is a the vector $(\mathbf{v}_\mathsf{H}, -c)$ and `n` the size of the training data set, `fakedata(z, n)` returns the matrix `S` and the class label vector `y`. (Alternatively, you can also write your own implementation of `fakedata`, if you are so inclined—any function that generates linearly separable, Euclidean data is fine.)

Homework questions:

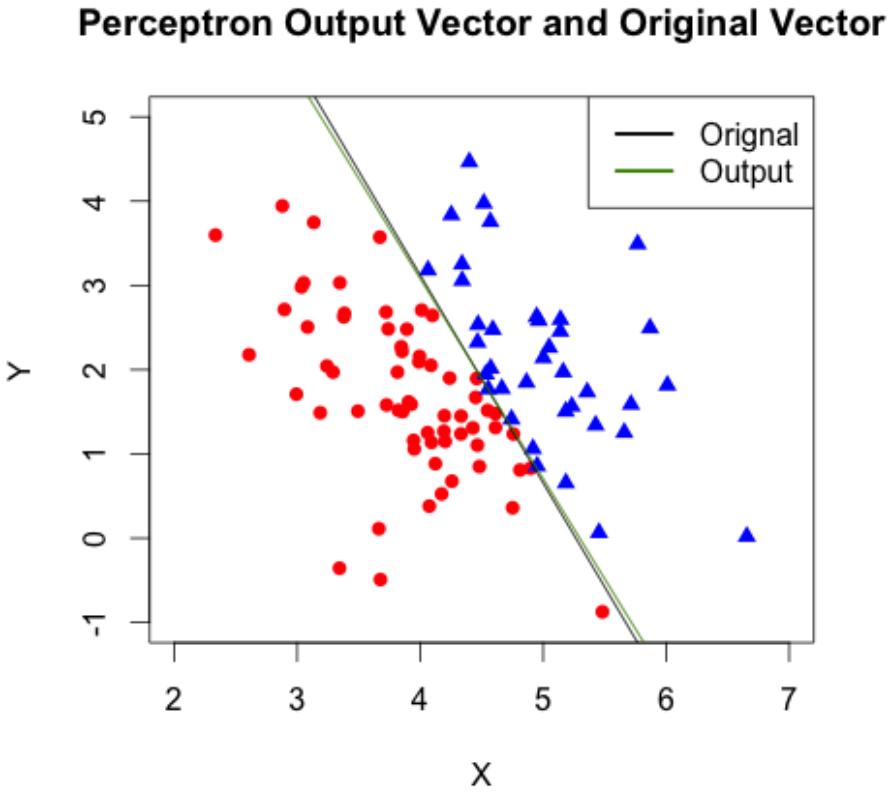
1. (10 points) To evaluate a Perceptron solution (a hyperplane classifier trained by a Perceptron algorithm), write a function `classify(S,z)` (where $z = (v, -c)$ is the vector defining the hyperplane) and return class label vector `y` as described above. `S` is a sample data set and `v` the Perceptron weight vector (which will be returned by the Perceptron training algorithm).
2. (10 points) Implement the Perceptron training algorithm (see slide 38) with $\alpha(k) = \frac{1}{k}$, where k is the number of the current iteration. The training algorithm should be an R function `perceptrain(S,y)` which returns a list containing `z` and `Z_history`, where `z` is again the normal vector $\mathbf{z}_\mathsf{H}$ of the hyperplane. `Z_history` is a matrix containing the history of the normal vector throughout the training run, i.e. a (number of iterations) $\times (d+1)$ matrix, the rows of which are the interim results for `z`.

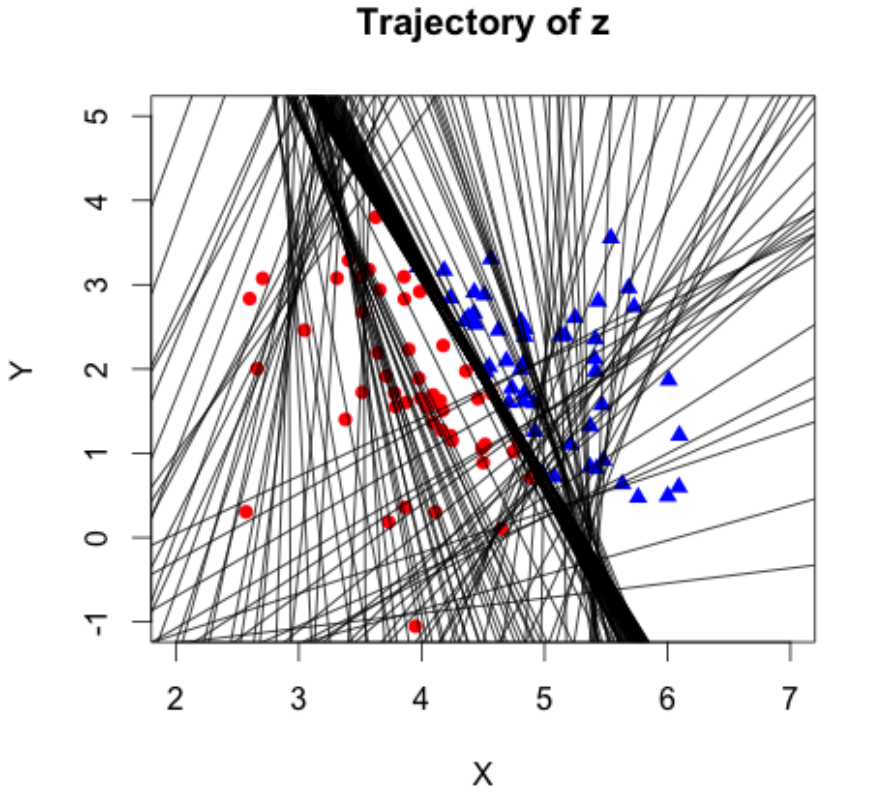
   A recommended sanity check is to train your `perceptron` function on a `fakedata` sample, and to make sure that the classifier returned by your Perceptron algorithm correctly classifies its own training data.
3. (10 points) Generate a new 3D random vector `z`, run `fakedata(z,100)`, and train your Perceptron on this set. Re-run `fakedata` with the same `z` to produce a test set, and check whether it is classified correctly.
4. (10 points) Convert the data and the vectors into their corresponding 2D representation to make it more suitable for plotting.

As a solution, please submit:

- **all** your program code.
- One plot showing the test data set and the classifier hyperplane, and one showing the training data and the trajectory of the algorithm by visualizing `Z_history`.

Solution:


Perceptron Output Vector and Original Vector
Orignal
Output
Y
X
-1
0
1
2
3
4
5
2
3
4
5
6
7

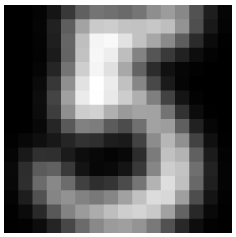

Trajectory of z
Y
X
-1
0
1
2
3
4
5
2
3
4
5
6
7

3. **SVM** (40 points)

In this problem, we will apply a support vector machine to classify hand-written digits. You do not have to implement the SVM algorithm: The R library e1071 provides an implementation, see

`http://cran.r-project.org/web/packages/e1071/index.html`

Download the digit data set from the course website. The zip archive contains two files: Both files are text files. The file `uspsdata.txt` contains a matrix with one data point (= vector of length 256) per row. The 256-vector in each row represents a 16×16 image of a handwritten number. The file `uspscl.txt` contains the corresponding class labels. The data contains two classes—the digits 5 and 6—so the class labels are stored as -1 and $+1$, respectively. The image on the right shows the first row, re-arranged as a 16×16 matrix and plotted as a gray scale image.



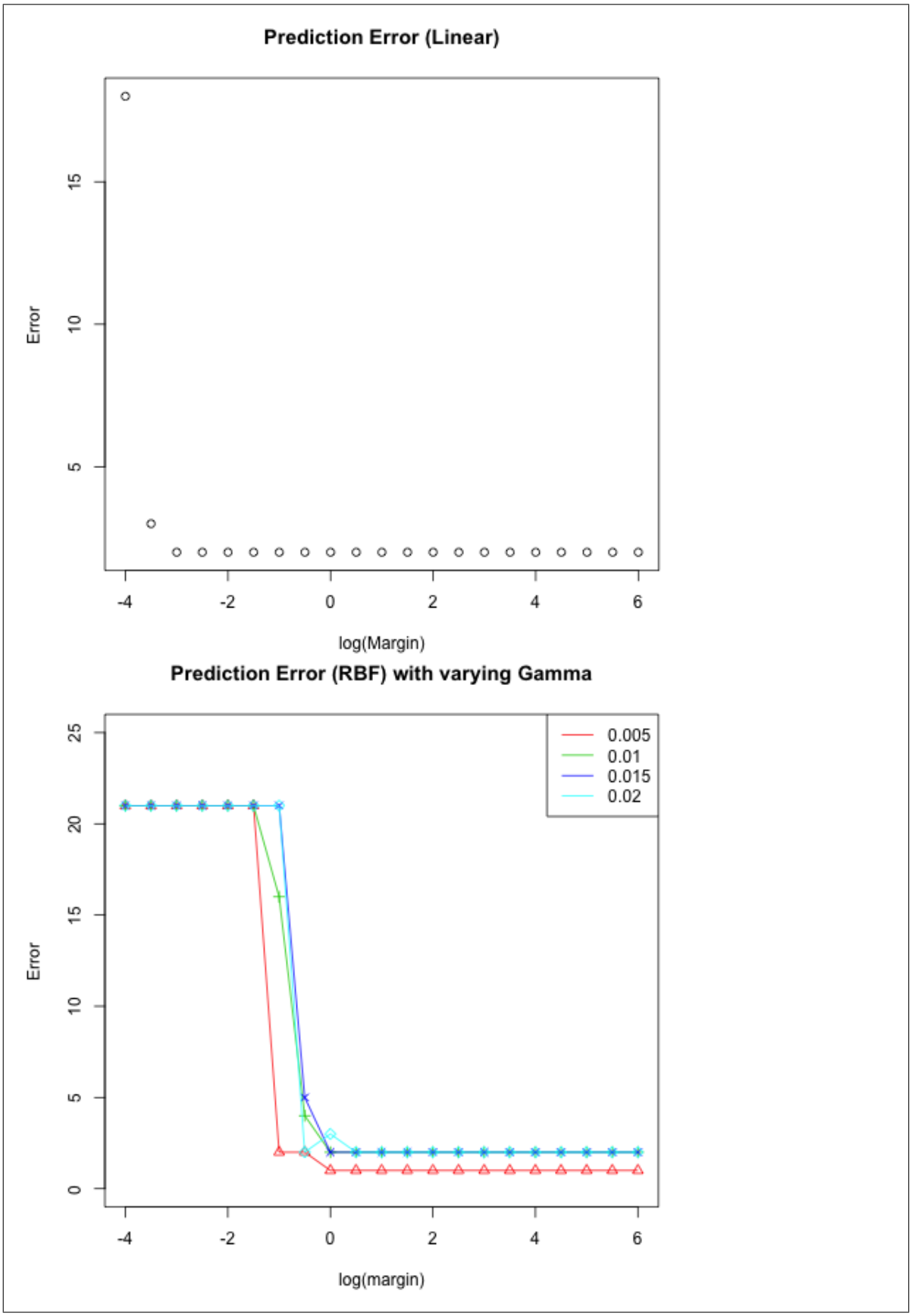
- Randomly select about 20% of the data and set it aside as a test set.
- Train a linear SVM with soft margin. Cross-validate the margin parameter.
- Train an SVM with soft margin and RBF kernel. You will have to cross-validate both the soft-margin parameter and the kernel bandwidth.
- After you have selected parameter values for both algorithms, train each one with the parameter value you have chosen. Then compute the misclassification rate (the proportion of misclassified data points) on the test set.

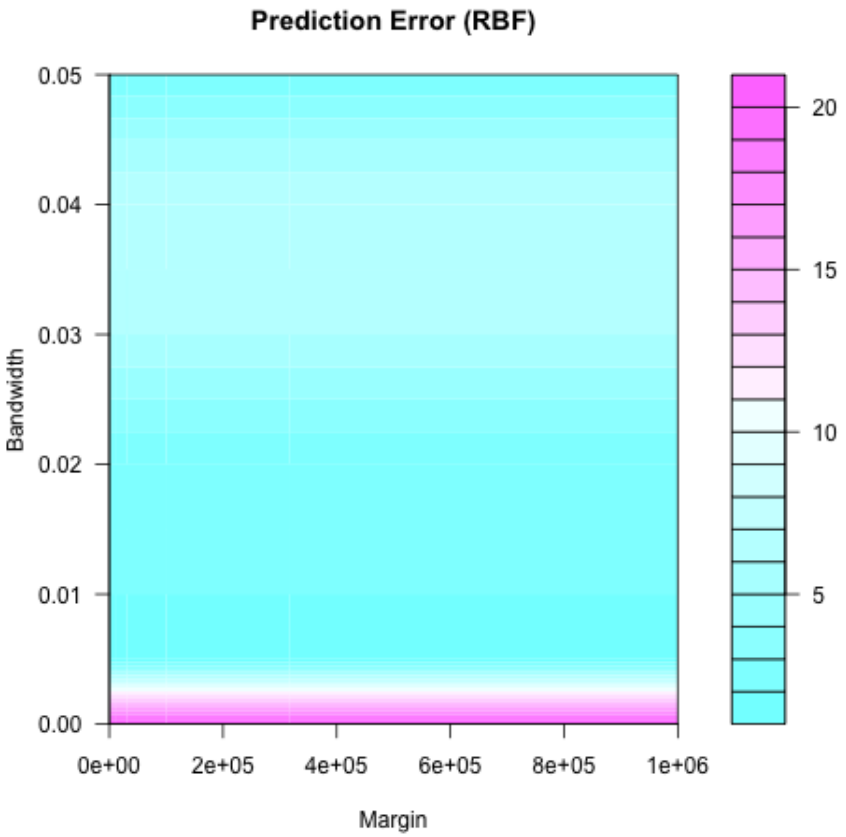
Homework questions:

1. (20 points) Plot the cross-validation estimates of the misclassification rate. Please plot the rate as
   (a) a function of the margin parameter in the linear case.
   (b) a function of the margin parameter and the kernel bandwidth in the non-linear case.
2. (20 points) Report the test set estimates of the misclassification rates for both cases, with the parameter values you have selected, and compare the two results. Is a linear SVM a good choice for this data, or should we use a non-linear one?

Again, in addition to the above, please submit **all** your program code.

Solution:
1a) and 1b) For 1b, the best four combinations of parameters are plotted.


Prediction Error (Linear)
Error
5
10
15
-4
-2
0
2
4
6
log(Margin)
Prediction Error (RBF) with varying Gamma
0.005
0.01
0.015
0.02
Error
0
5
10
15
20
25
-4
-2
0
2
4
6
log(margin)



2

```
                  Test
Linear (0.01)   -1   1
             -1 21   0
              1  2 17

                  Test
RBF(100,0.005)  -1   1
             -1 21   0
              1  2 17
```

Since the result for a linear SVM and a RBF SVM is comparable, we should use a linear SVM as the model is less complex, more interpretable and requires less computation.