Statistical Machine Learning (STAT W4400)

Homework 2

Linan Qiu
`lq2137`

October 13, 2015

Note: since submission only allows `.pdf` and `.R`, I did not preserve the directory structure of this project. The `.R` files will not run properly due to imports and assumed directory structure. For a functional version, go to https://github.com/linanqiu/stat-w4400-homework/tree/master/hw2/r

1 Linear Classification

1.1 Classification Results

$\mathbf{v}_h$ is a unit vector.

```
vh = matrix(c(1/sqrt(2), 1/sqrt(2)), nrow=2)
c = 1/(2 * sqrt(2))
x1 = matrix(c(-3, 0), nrow=2)
x2 = matrix(c(1/2, 1/2), nrow=2)

> sign(crossprod(x1, vh) - c)
     [,1]
[1,]   -1
> sign(crossprod(x2, vh) - c)
     [,1]
[1,]    1
```

$$r_1 = \text{sgn}(< \mathbf{x}_1, \mathbf{v}_h > -c) = -1$$

$$r_1 = \text{sgn}(< \mathbf{x}_2, \mathbf{v}_h > -c) = 1$$

1.2 SVM with Margin

Concept of margin applies only to training, not classification. Classification works for any linear classifier. For the test point $\mathbf{x}$,

$$r = \operatorname{sgn}(< \mathbf{x}_1, \mathbf{v}_h > -c)$$

still applies. Hence the predicted classes will still be the same.

1.3 Cost Function Approximated by Perceptron Cost Function

It approximates the empirical risk function. Empirical risk function is piece-wise constant, hence would not allow us to gradient descent optimally. The perceptron cost function, by using $\left|\left\langle \mathbf{z}, \begin{pmatrix} 1 \\ \tilde{\mathbf{x}}_i \end{pmatrix} \right\rangle\right|$ instead of just the loss function.

2 Perceptron

2.1 Classify

Included in file `problem2.R`

2.2 Perceptron Training Algorithm

Included in file `problem2.R`

2.3 Train and Test

Included in file `problem2.R`

2.4 2D Representation

Slope can be obtained from $\mathbf{v}_h$ by

$$v_x x + v_y y - c = 0$$

and solving accordingly.

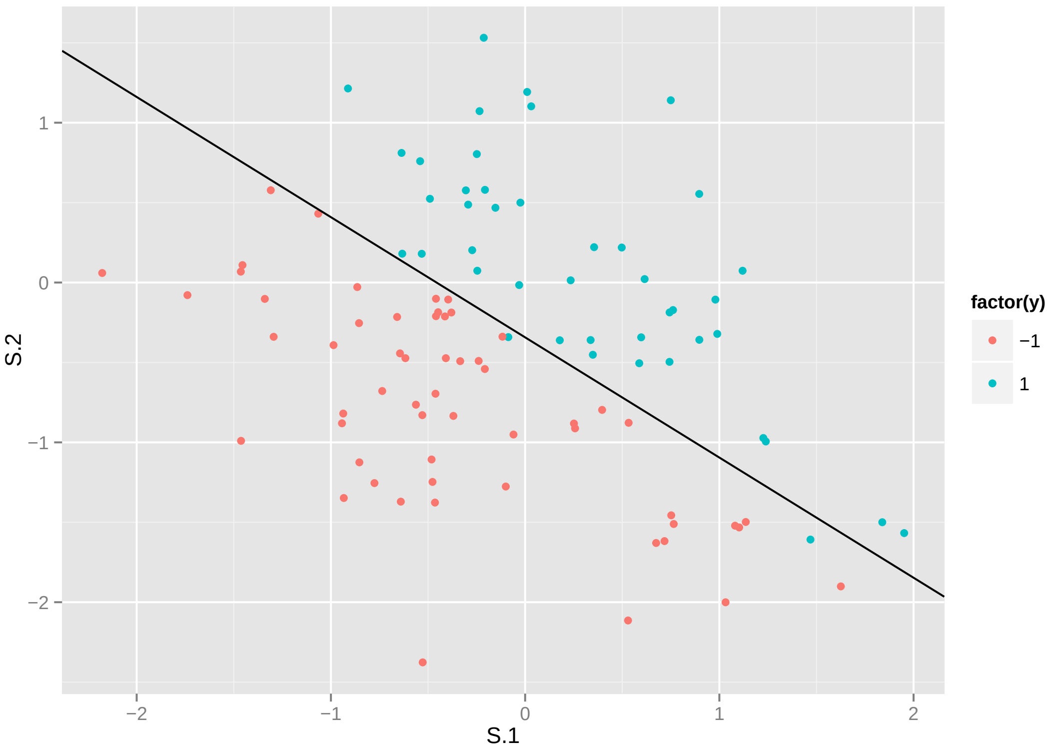

Figure 1: Plot of test data against obtained classifier from `z`.

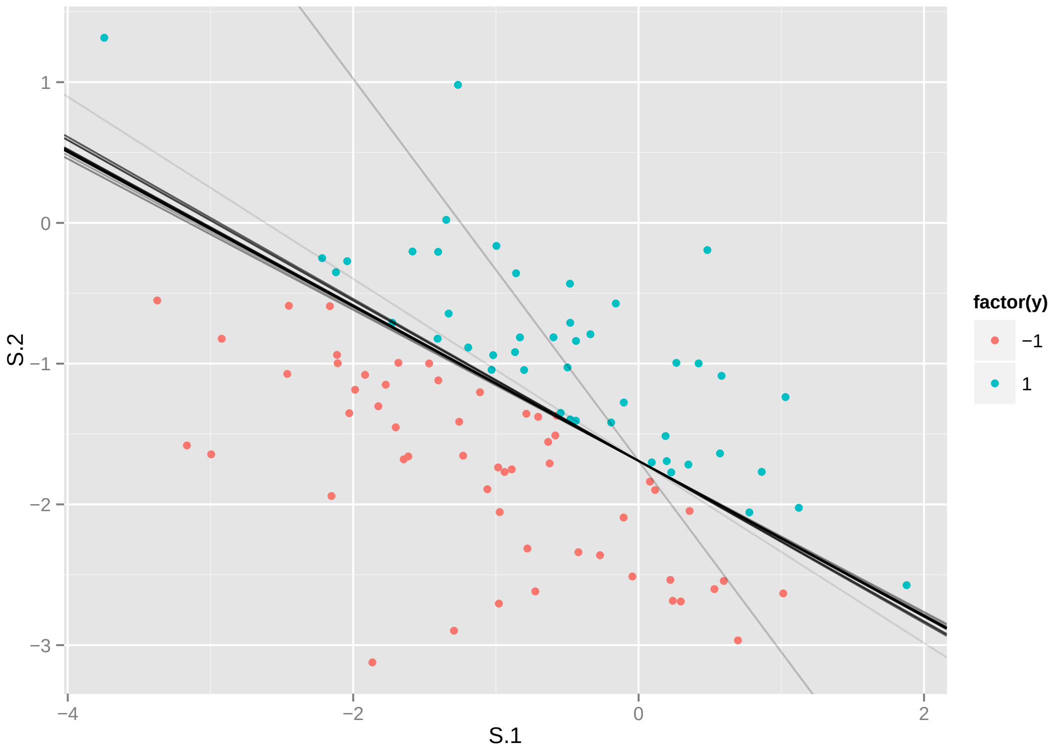

Figure 2: Plot of training data against history of classifiers from `z_history`. Earlier iterations have lower alpha.

3 SVM

3.1 Cross Validation Estimates

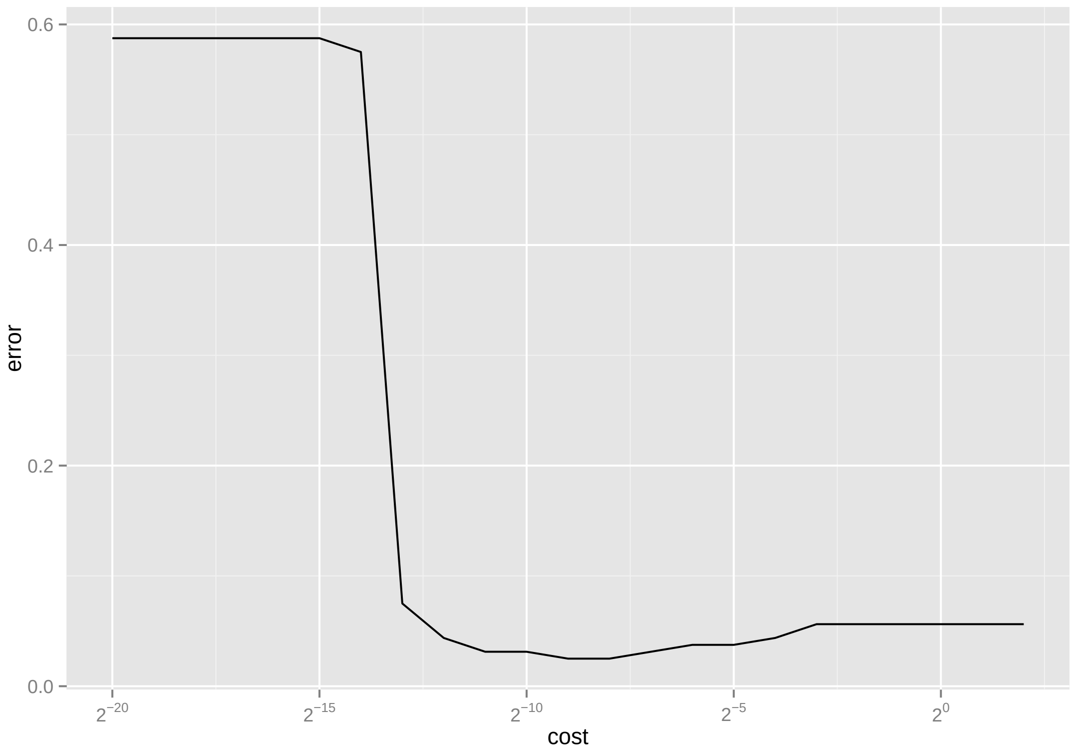

Figure 3: Plot of error against margin parameter (cost) for linear kernel SVM

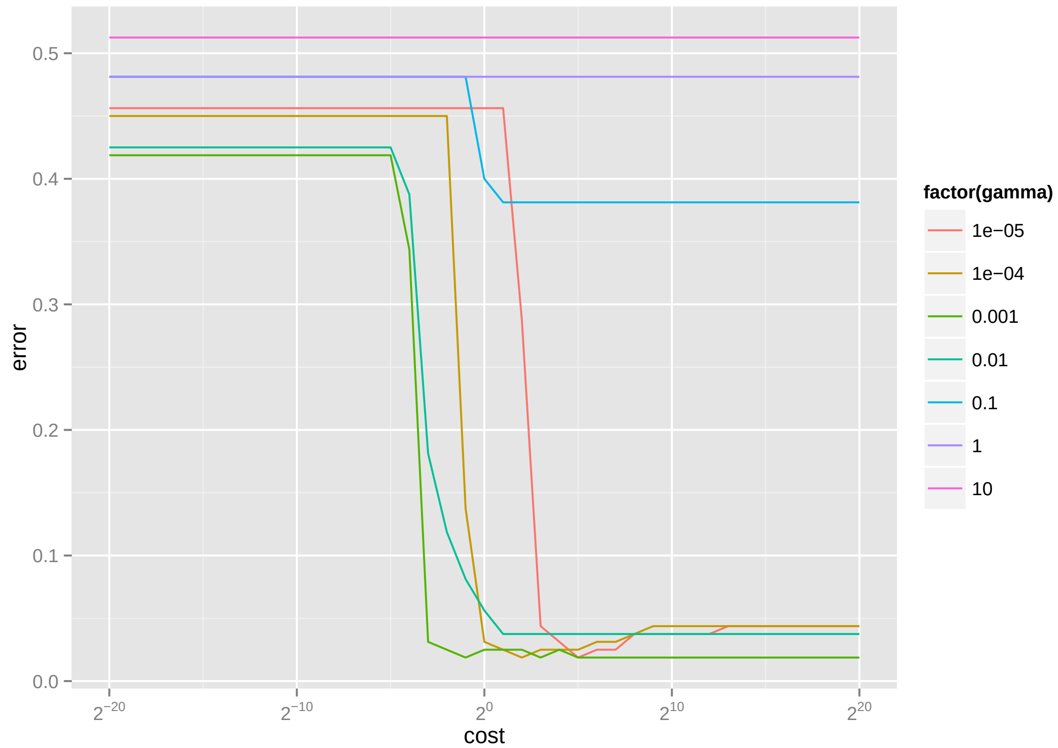

Figure 4: Plot of error against margin parameter (cost) for different kernel bandwidth (gamma) for RBF kernel SVM

3.2 Test

Selected parameter values are:

```
> tuned_linear

Parameter tuning of 'svm':

- sampling method: 10-fold cross validation

- best parameters:
        cost
 0.001953125

- best performance: 0.025

> tuned_rbf

```

```
Parameter tuning of 'svm':

- sampling method: 10-fold cross validation

- best parameters:
 gamma cost
 0.001  0.5

- best performance: 0.01875
```

It seems from the training data tuning that the RBF kernel has a better performance. However, upon testing, this is not immediately clear.

```
> linear_model = svm(labels ~ ., data=data, method="C-classification", kernel="linear", cost=tuned_linear$best.parameter$cost[1])
> linear_predict = predict(linear_model, testset[, -ncol(data)])
> classAgreement(table(pred = linear_predict, true = testset[, ncol(data)]))
$diag
[1] 0.975

$kappa
[1] 0.9497487

$rand
[1] 0.95

$crand
[1] 0.8999645

>
> rbf_model = svm(labels ~ ., data=data, method="C-classification", kernel="radial", cost=tuned_rbf$best.parameter$cost[1], gamma=tuned_rbf$best.parameter$gamma[1])
> rbf_predict = predict(rbf_model, testset[, -ncol(data)])
> classAgreement(table(pred = rbf_predict, true = testset[, ncol(data)]))
$diag
[1] 0.975

$kappa
```

```
[1] 0.9497487

$rand
[1] 0.95

$crand
[1] 0.8999645
```

Both models give the same accuracy rate (hence misclassification rate).