Solutions for ST340 Lab 7

2019–20

Load package e1071:

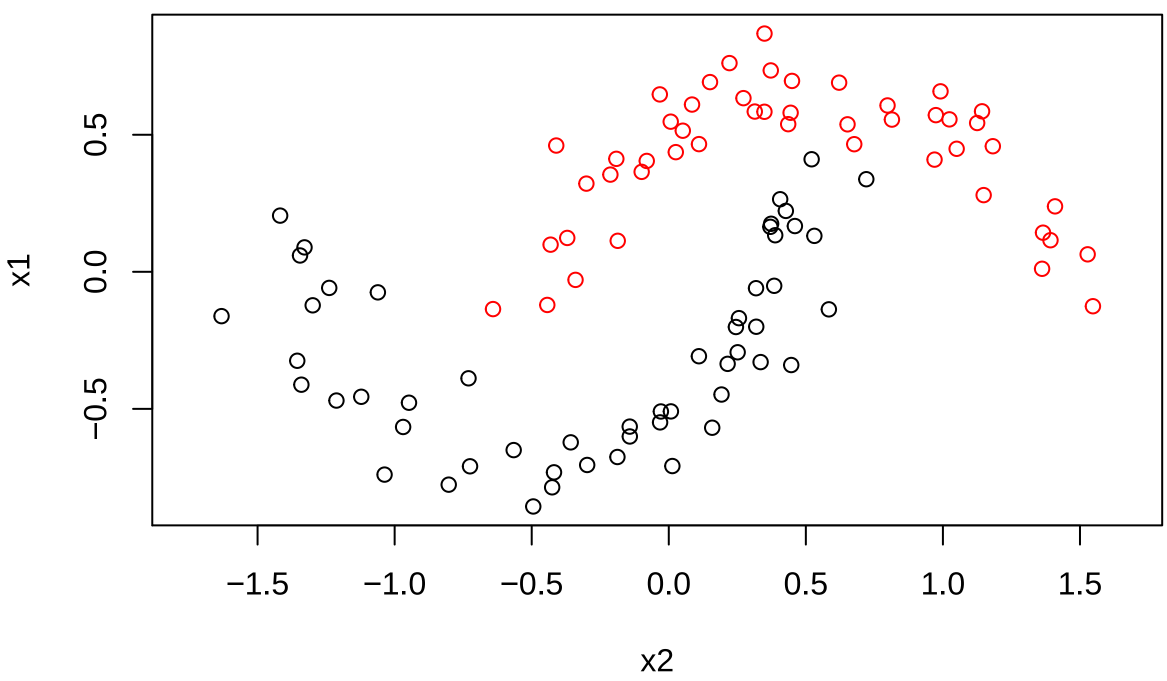
```
if (!require("e1071")){
  install.packages("e1071")
  library("e1071")
}
```

```
## Loading required package: e1071
```

```
## Warning: package 'e1071' was built under R version 3.5.2
```

(a) Run the following code, and then construct and plot an SVM with a quadratic kernel. What is its test set accuracy? (The code to answer this is on the lecture slides).

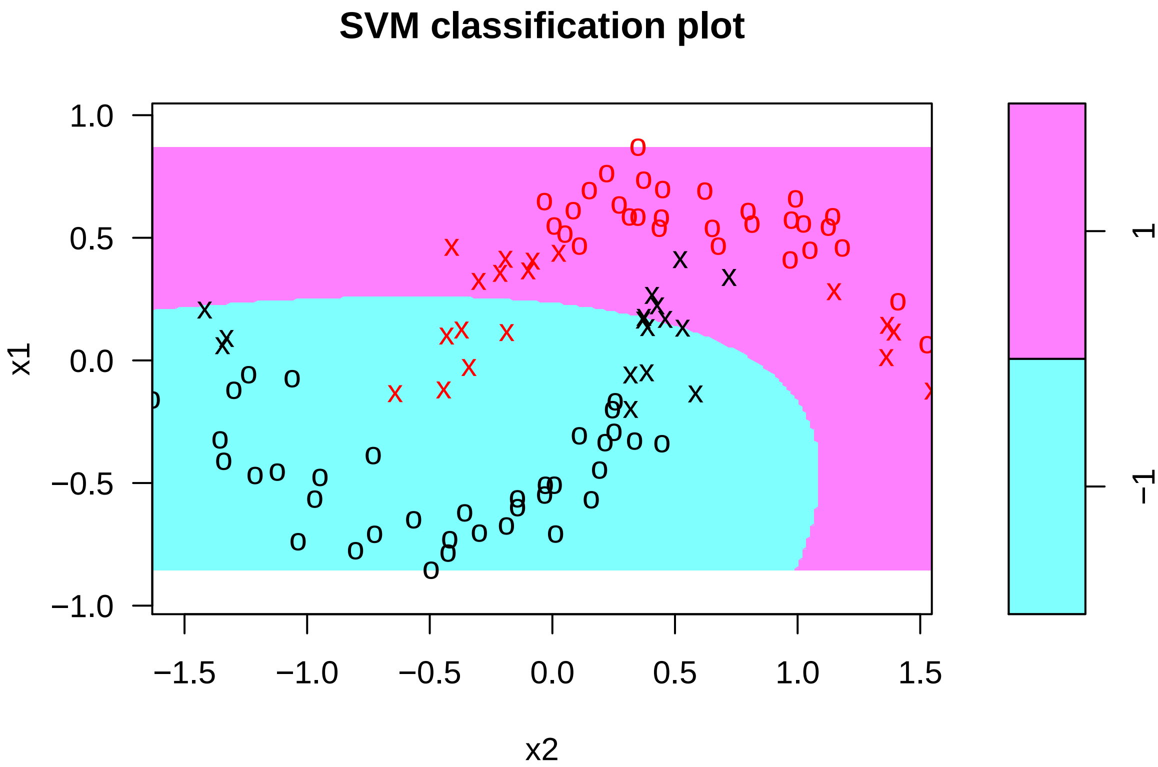
```
set.seed(1)
th=runif(100,-pi,pi)
y=sign(th)
x1=sin(th)-y/3+rnorm(100,sd=0.1)
x2=cos(th)+y/2+rnorm(100,sd=0.1)
plot(x2,x1,asp=1,col=(y+3)/2)
```




```
d <- data.frame(x1=x1,x2=x2,y=factor(y))
s <- svm(y~.,d,type="C-classification",kernel="polynomial",degree=2,gamma=0.5,coef0=1,cost=1)
mean(predict(s,d)==y)
```

```
## [1] 0.87
```

```
plot(s,d,grid=200,asp=1)
```



(b) Use leave-one-out-cross-validation to compare polynomial kernels. Write some code to perform a grid search over possible choices of parameters. For simplicity, fix $c = 1$ (`coef0 = 1`) and $C = 1$ (`cost=1`) and search over the remaining parameters γ and p and to select the model which is optimal in terms of cross-validation error. A suggested grid is:

```
p.range <- 1:10
gamma.range <- c(0.001,0.01,0.1,0.5,1,2,5,10)
```

(Hint: You do not need to write a function to compute the cross-validation error: look at the `cross` flag in `?svm`. You can extract the accuracies of the cross-validation sets from an svm object `s` using `s$accuracies` and `s$tot.accuracy`.)

```
library(knitr) # For the kable function, which outputs nice tables in markdown
```

```
## Warning: package 'knitr' was built under R version 3.5.2
```

```
poly.svm.grid.search <- function(p.range, gamma.range, d,y) {
    accuracy.grid <- matrix(nrow=length(gamma.range),ncol = length(p.range),
                            dimnames=list(gamma.range,p.range))

    for (i in 1:length(gamma.range)) {
        for (j in 1:length(p.range)) {
        s <- svm(y~.,d,type="C-classification",kernel="polynomial",gamma=gamma.range[i],
                 cost=1, coef0=1, degree=p.range[j],cross=length(y))
        accuracy.grid[i,j] <- s$tot.accuracy
      }
    }
    return(accuracy.grid)
}
accuracy.grid <- poly.svm.grid.search(p.range,gamma.range,d,y)
kable(accuracy.grid,format = 'markdown')
```

	1	2	3	4	5	6	7	8	9	10
0.001	52	52	52	64	80	82	82	83	83	83
0.01	83	82	85	85	85	85	85	85	85	85
0.1	85	85	85	86	91	93	95	96	99	98
0.5	84	85	99	98	97	99	97	97	98	98
1	84	84	100	99	97	97	97	97	97	96
2	84	88	99	96	97	95	95	93	94	92
5	84	87	97	95	96	94	95	92	94	92
10	84	87	98	94	94	92	94	92	93	91

```r
poly.index <- which(accuracy.grid == max(accuracy.grid), arr.ind = TRUE)
poly.parameters <- c(gamma.range[poly.index[1]],p.range[poly.index[2]])
print(poly.parameters)
```

```
## [1] 1 3
```

(c) Repeat part (b) to choose a value of γ for the radial basis function.

```r
radial.svm.grid.search <-
  function(gamma.range, d,y) {
    accuracy.grid <- matrix(nrow=length(gamma.range),ncol = 1,dimnames=list(gamma.range))

    for (i in 1:length(gamma.range)) {
      s <- svm(y~.,d,type="C-classification",kernel="radial",gamma=gamma.range[i],cost=1, coef0=1,
               cross=length(y))
      accuracy.grid[i] <- s$tot.accuracy
    }
    return(accuracy.grid)
  }
accuracy.grid <- radial.svm.grid.search(gamma.range,d,y)
kable(accuracy.grid,format = 'markdown',,col.names = '')
```

0.001	52
0.01	83
0.1	85
0.5	97
1	98
2	98
5	98
10	98

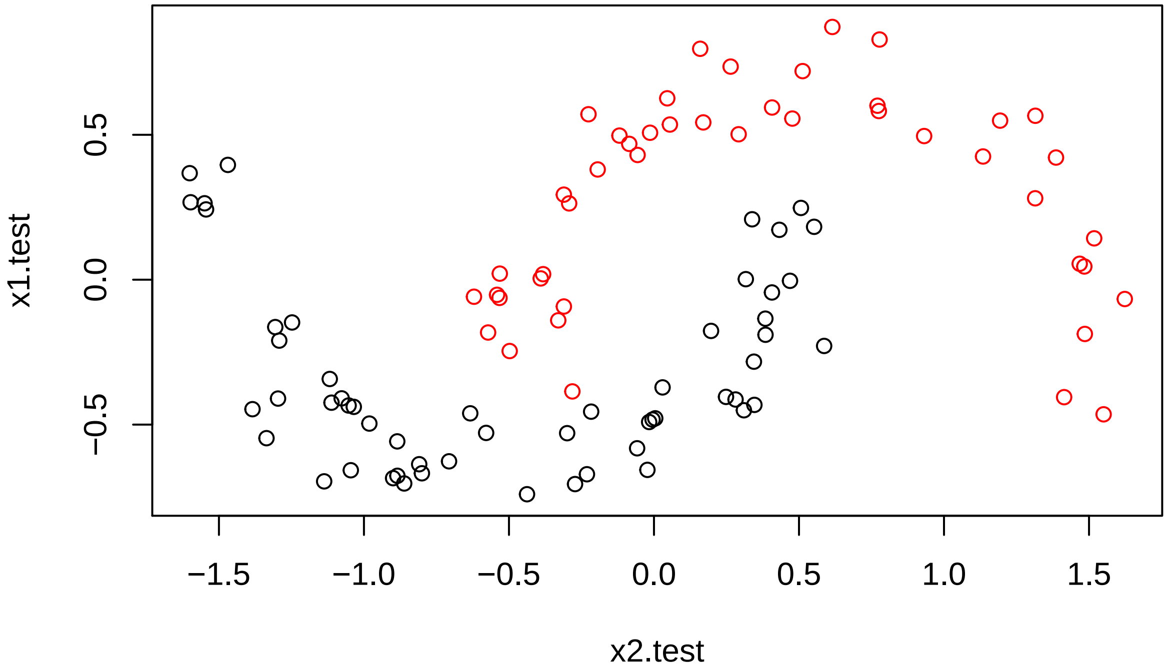
```r
rbf.parameter <- gamma.range[which.max(accuracy.grid)]
print(rbf.parameter)
```

```
## [1] 1
```

(d) Compare the performance of your chosen polynomial kernel and RBF kernel by simulating an independent test dataset and evaluating their accuracies.

```r
s.poly <- svm(y~.,d,type="C-classification",kernel="polynomial",degree=poly.parameters[2],
              gamma=poly.parameters[1],coef0=1,cost=1)
s.rbf <- svm(y~.,d,type="C-classification",kernel="radial",gamma=rbf.parameter,cost=1)
set.seed(2)
th=runif(100,-pi,pi)
```

```r
y.test=sign(th)
x1.test=sin(th)-y.test/3+rnorm(100,sd=0.1)
x2.test=cos(th)+y.test/2+rnorm(100,sd=0.1)
plot(x2.test,x1.test,asp=1,col=(y.test+3)/2)
```



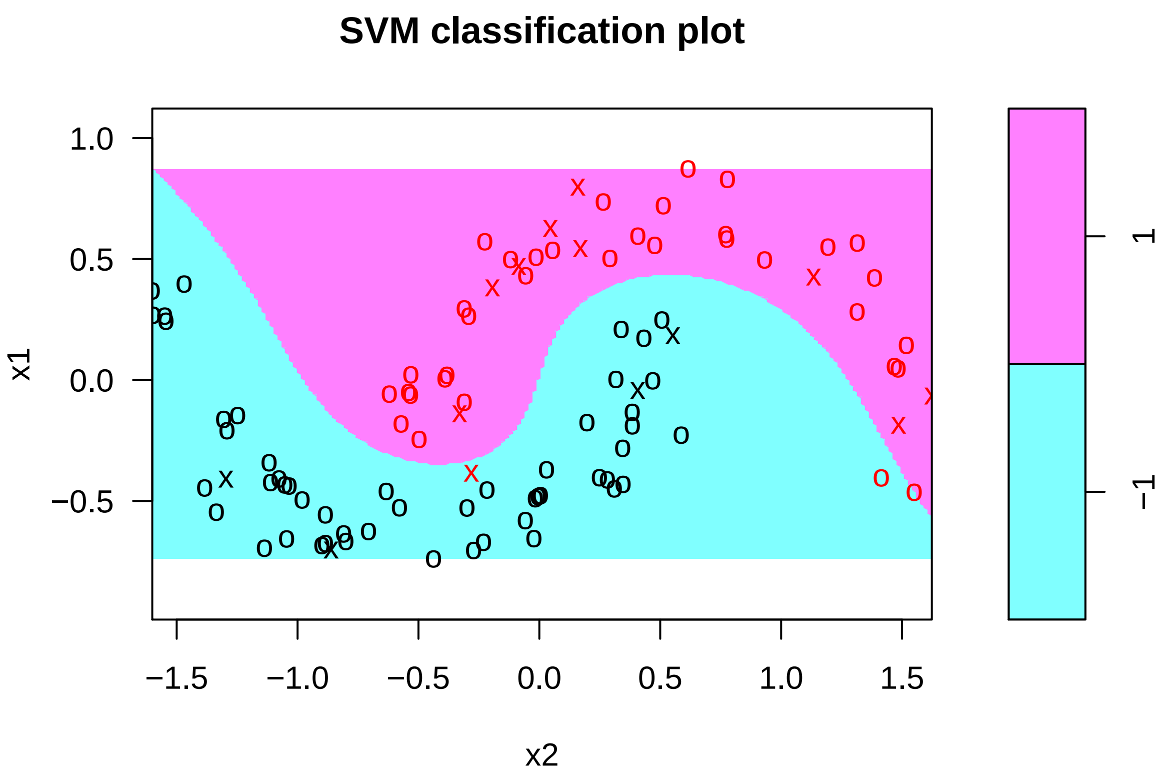

```r
d.test <- data.frame(x1=x1.test,x2=x2.test,y=factor(y.test))

mean(predict(s.poly,d.test)==y.test)
```

```
## [1] 0.98
```

```r
mean(predict(s.rbf,d.test)==y.test)
```

```
## [1] 0.99
```

```r
plot(s.poly,d.test,grid=200,asp=1)
```




```
plot(s.rbf,d.test,grid=200,asp=1)
```