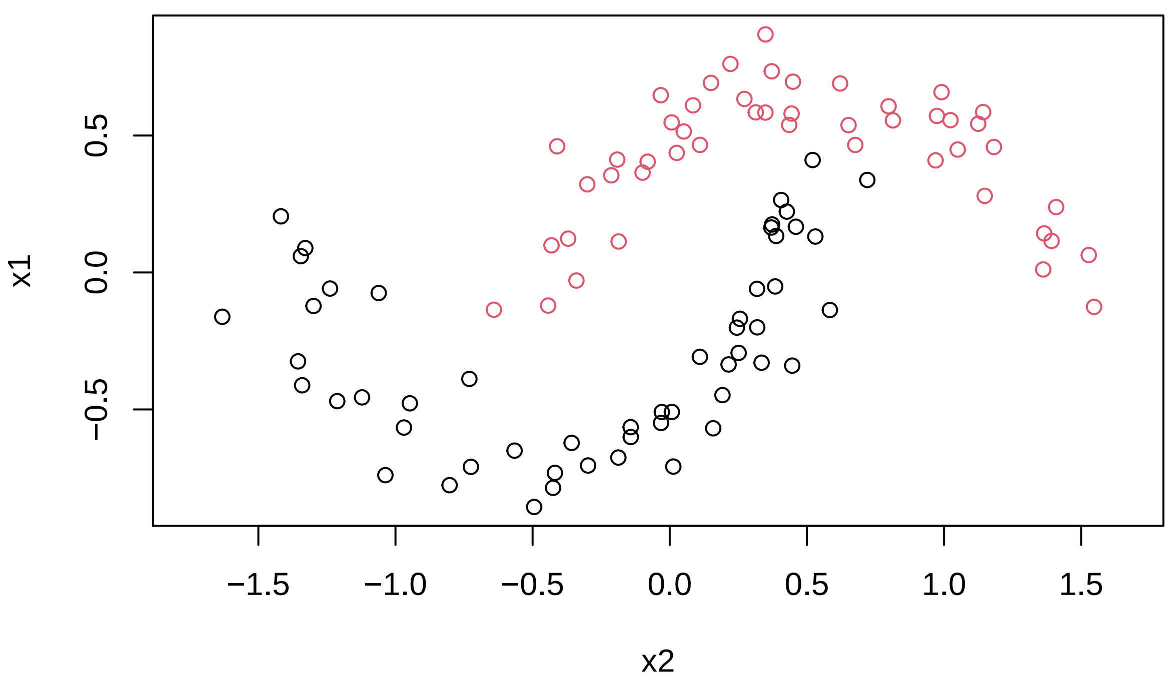
ST340 Lab 7: Support vector machines

2020–21

Load package e1071:

(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)
```

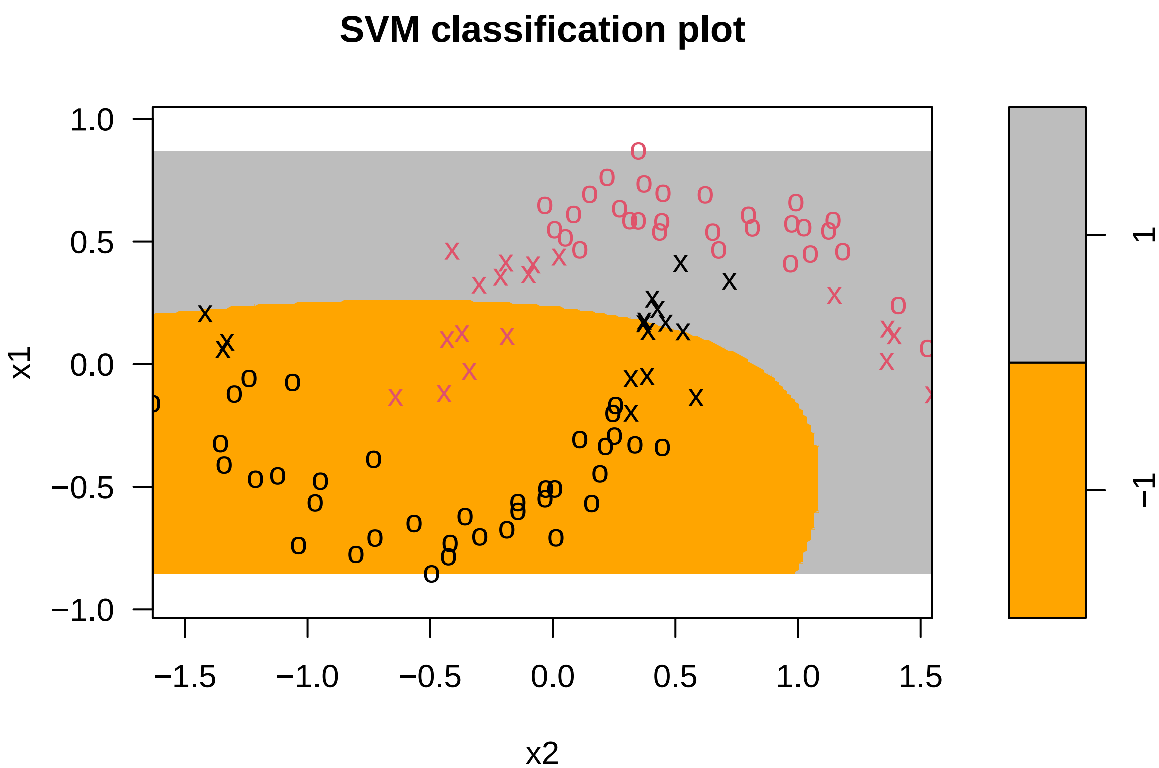



```
d <- data.frame(x1=x1,x2=x2,y=factor(y))
```

```
s = svm(y ~., d, kernel="polynomial", degree=2, coef0 = 1, cost = 1)    # Default cost is 1
paste("accuracy", mean(predict(s,d[,1:2])==y),"   ||   ", "Number of support vectors:", nrow(s$SV))
```

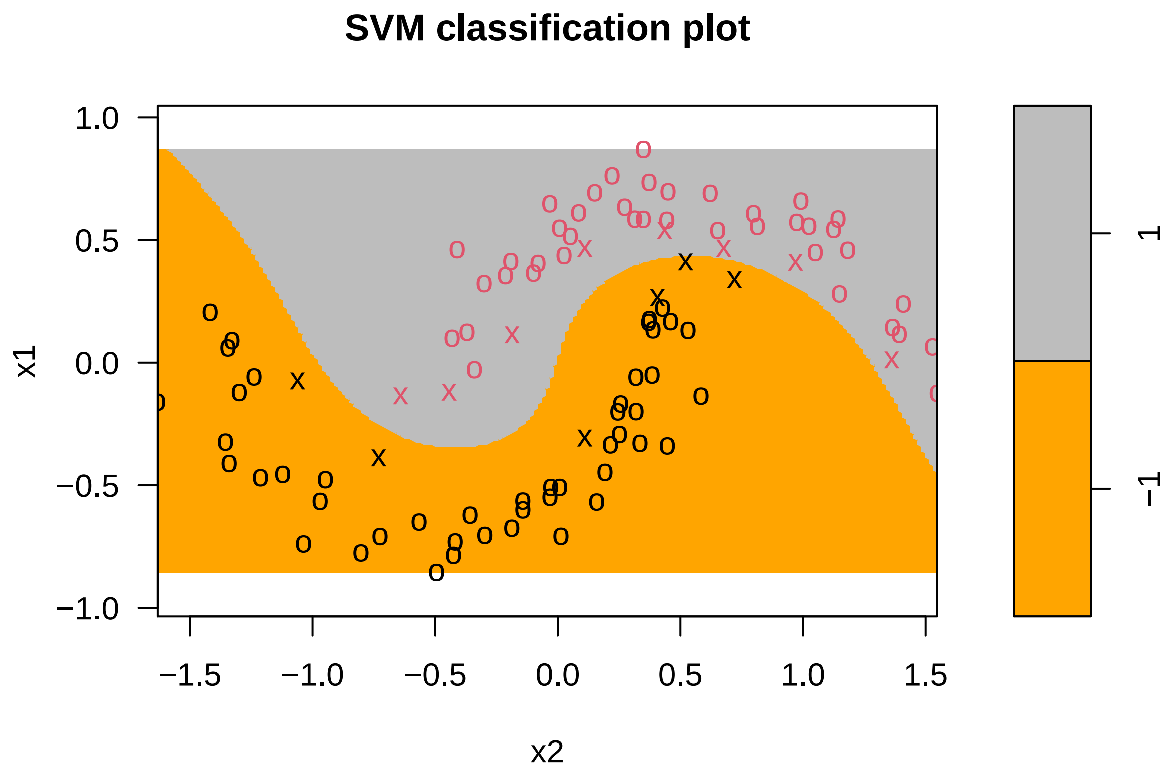
```
## [1] "accuracy 0.87    ||    Number of support vectors: 34"
```

```
plot(s,d, asp=1, grid = 200,col = c("orange","grey74"))
```



(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)
CV_grid = matrix(NA,length(p.range),length(gamma.range))
rownames(CV_grid) = paste("P =",p.range)
colnames(CV_grid) = paste("Gamma =",gamma.range)

for(i in 1:length(p.range)){
  for(j in 1:length(gamma.range)){

  s = svm(y ~., d, kernel="polynomial", degree = p.range[i], gamma = gamma.range[j], coef0 = 1, cross =
  CV_grid[i,j] = s$tot.accuracy

  }
}

# From this LOOCV it seems that setting Gamma = 1 and P = 3 performs the best

s = svm(y ~., d, kernel="polynomial", degree = 3, gamma = 1, coef0 = 1)
plot(s,d, asp=1, grid = 200,col = c("orange","grey74"))
```



(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`.)

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

```
gamma.range <- c(0.001,0.01,0.1,0.5,1,2,5,10)
CV_vector = matrix(NA, nrow = 1, ncol = length(gamma.range))
colnames(CV_vector) = gamma.range


for(j in 1:length(gamma.range)){

  s = svm(y ~., d, kernel="radial", gamma = gamma.range[j], cost = 1, cross = nrow(d))
  CV_vector[1,j] = s$tot.accuracy

}

print(CV_vector)
```

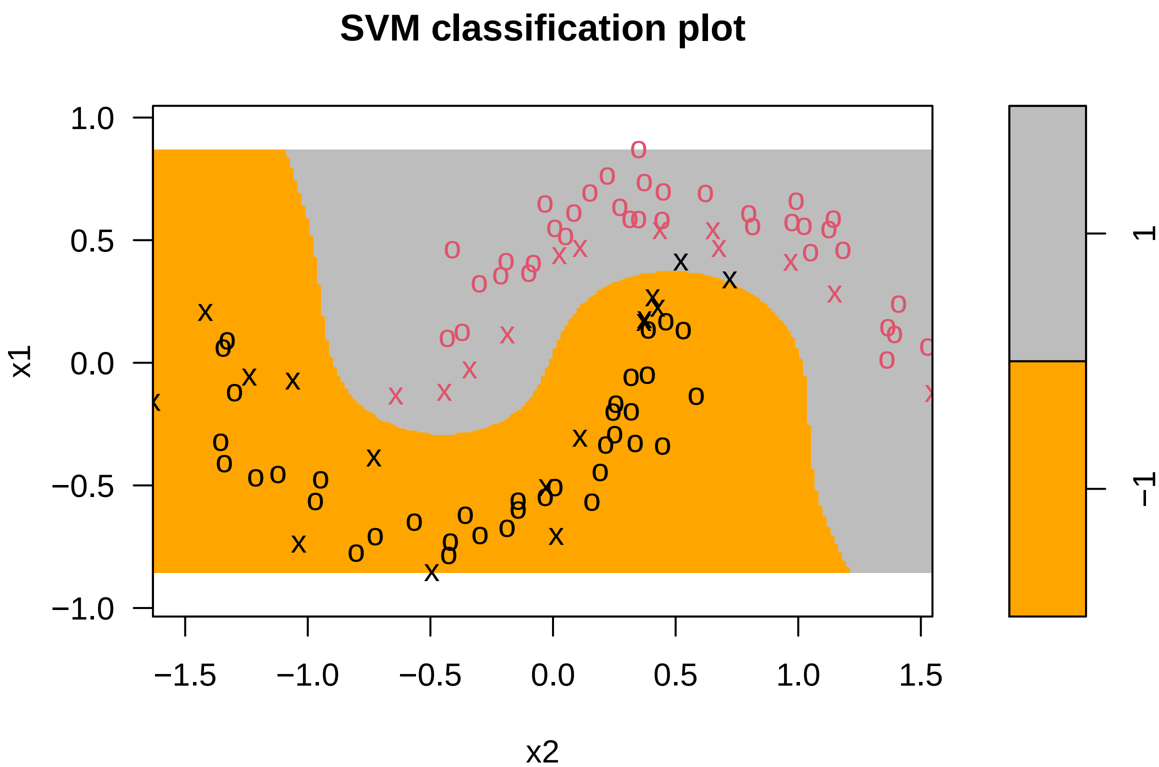
```
##      0.001 0.01 0.1 0.5  1  2  5 10
## [1,]    52   83  85  97 98 98 98 98
```

```
colnames(CV_vector)[which.max(CV_vector)]
```

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

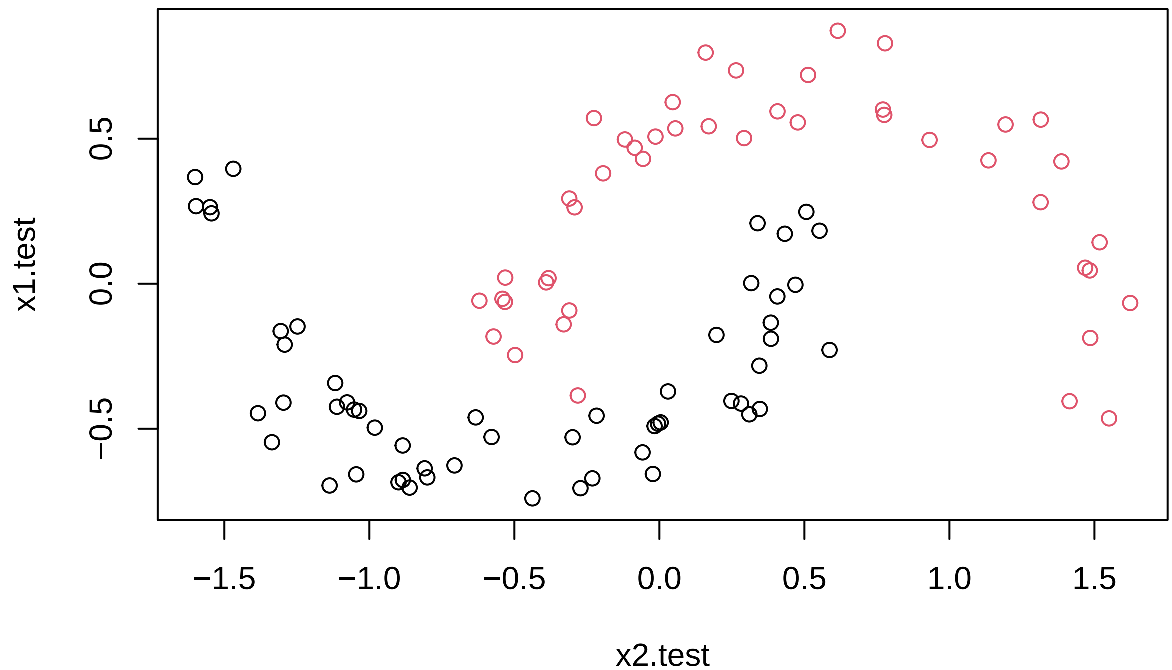
```
# From this LOOCV it seems that setting Gamma = 1 performs the best
# After further investigation, doesnt seem that we can get better prediction accuracy than 98%
```

```
s = svm(y ~., d, kernel="radial", gamma = 1)
plot(s,d, asp=1, grid = 200,col = c("orange","grey74"))
```



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

```
s.poly <- svm(y~.,d,type="C-classification",kernel="polynomial",degree=3,
              gamma=1,coef0=1,cost=1)
s.rbf <- svm(y~.,d,type="C-classification",kernel="radial",gamma=1, cost=1)
set.seed(2)
th=runif(100,-pi,pi)
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)
```

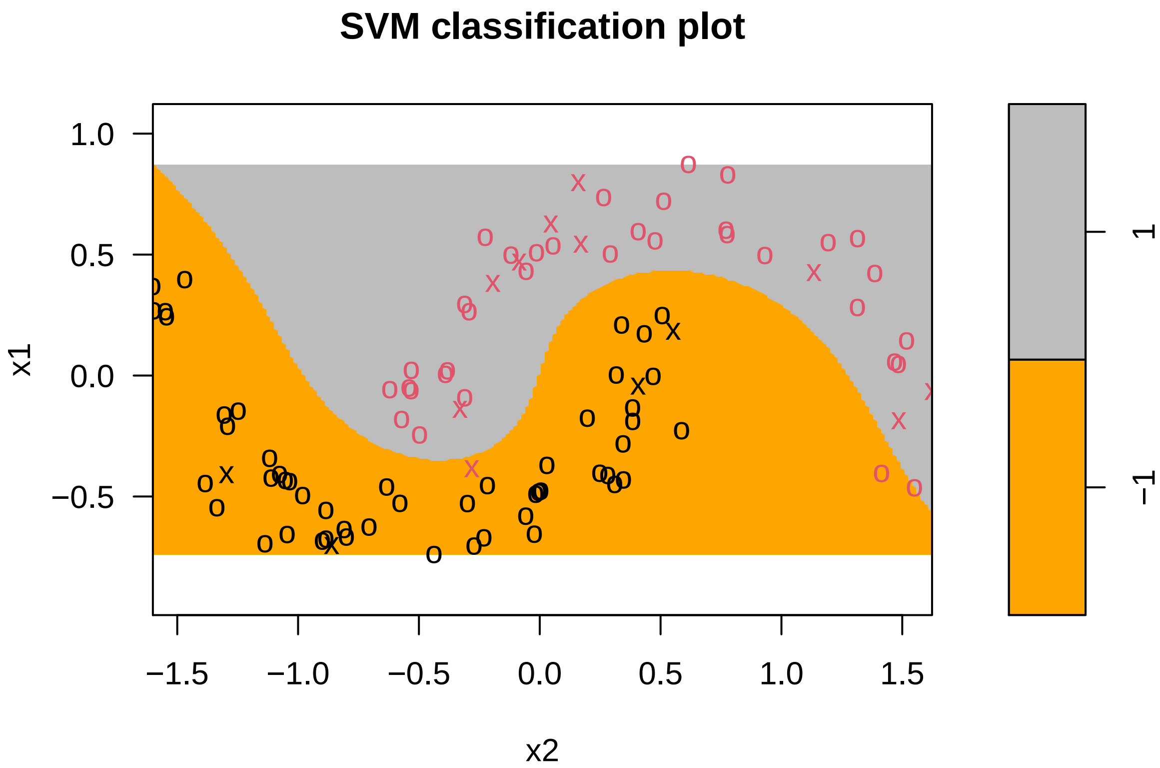
```
## [1] 0.98
```

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

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

Plot of SVM classification with Polynomial Kernel. With degree = 3, gamma = 1

```
plot(s.poly,d.test,grid=200,asp=1,col = c("orange","grey74"), main = "polynomial Kernel\nDegree = 3, Ga
```



Plot of SVM classification with radial basis kernel. With gamma = 1

```
plot(s.rbf,d.test,grid=200,asp=1, col = c("orange","grey74"))
```