Tutorial:
SVM In R

袁会卓

Package: e1071

- Function: svm()
- Usage:
- `svmfit<-svm(x,as.factor(y))`
- `svmfit<-svm(y ~ ., data=dat, kernel='linear', cost=0.1, scale=FALSE)`
- Arguments:
- `formula,data,x,y,scale,kernel,degree,gamma,cost`

Support vector classifier

- Generating the observations:
- `set.seed (1)`
- `x<-matrix(rnorm(20*2), ncol=2)`
- `y<-c(rep(-1,10), rep(1,10))`
- `x[y==1,]=x[y==1,]+1`
- `plot(x, col=(3-y))`
- `dat=data.frame(x=x, y=as.factor(y))`

Support vector classifier

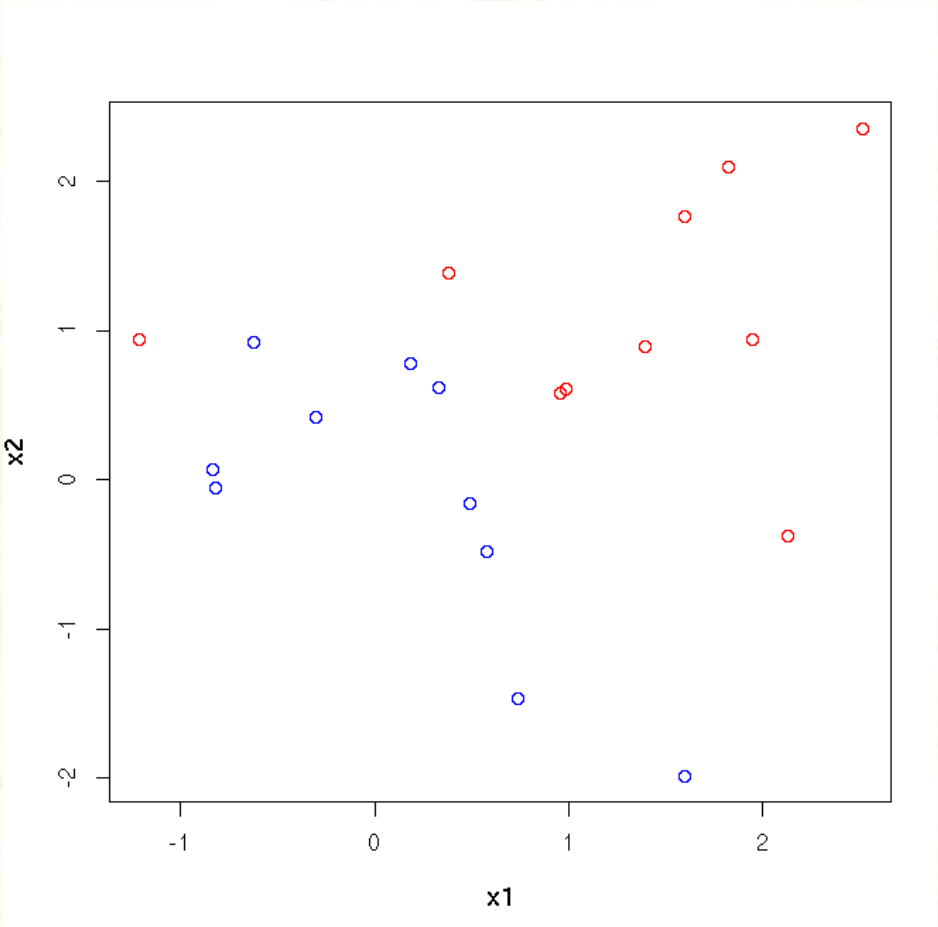
- Generating the observations:
- `set.seed (1)`
- `x<-matrix(rnorm(20*2), ncol=2)`
- `y<-c(rep(-1,10), rep(1,10))`
- `x[y==1,]=x[y==1,]+1`
- `plot(x, col=(3-y))`

```
> x
             [,1]        [,2]
 [1,] -0.6264538  0.91897737
 [2,]  0.1836433  0.78213630
 [3,] -0.8356286  0.07456498
 [4,]  1.5952808 -1.98935170
 [5,]  0.3295078  0.61982575
 [6,] -0.8204684 -0.05612874
 [7,]  0.4874291 -0.15579551
 [8,]  0.7383247 -1.47075238
 [9,]  0.5757814 -0.47815006
[10,] -0.3053884  0.41794156
[11,]  2.5117812  2.35867955
[12,]  1.3898432  0.89721227
[13,]  0.3787594  1.38767161
[14,] -1.2146999  0.94619496
[15,]  2.1249309 -0.37705956
[16,]  0.9550664  0.58500544
[17,]  0.9838097  0.60571005
[18,]  1.9438362  0.94068660
[19,]  1.8212212  2.10002537
[20,]  1.5939013  1.76317575
```

```
> y
 [1] -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  1  1  1  1  1  1  1  1  1  1
```

Support vector classifier

- Not linearly seperable



Support vector classifier

- Fitting the data:
- `dat<-data.frame(x=x, y=as.factor(y))`
- `library('e1071')`

- `svmfit<-svm(y ~ ., data=dat, kernel='linear', cost=10, scale=FALSE)`
- `plot(svmfit,dat)`

```
> attributes(svmfit)
$names
 [1] "call"           "type"           "kernel"         "cost"
 [5] "degree"         "gamma"          "coef0"          "nu"
 [9] "epsilon"        "sparse"         "scaled"         "x.scale"
[13] "y.scale"        "nclasses"       "levels"         "tot.nSV"
[17] "nSV"            "labels"         "SV"             "index"
[21] "rho"            "compprob"       "probA"          "probB"
[25] "sigma"          "coefs"          "na.action"      "fitted"
[29] "decision.values" "terms"

$class
[1] "svm.formula" "svm"
```

Support vector classifier

```
> svmfit$index
[1]  1  2  5  7 14 16 17
> summary(svmfit)

Call:
svm(formula = y ~ ., data = dat, kernel = "linear", cost = 10, scale = FALSE)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  10
      gamma:  0.5

Number of Support Vectors:  7

 ( 4 3 )


Number of Classes:  2

Levels:
 -1 1
```

Support vector classifier

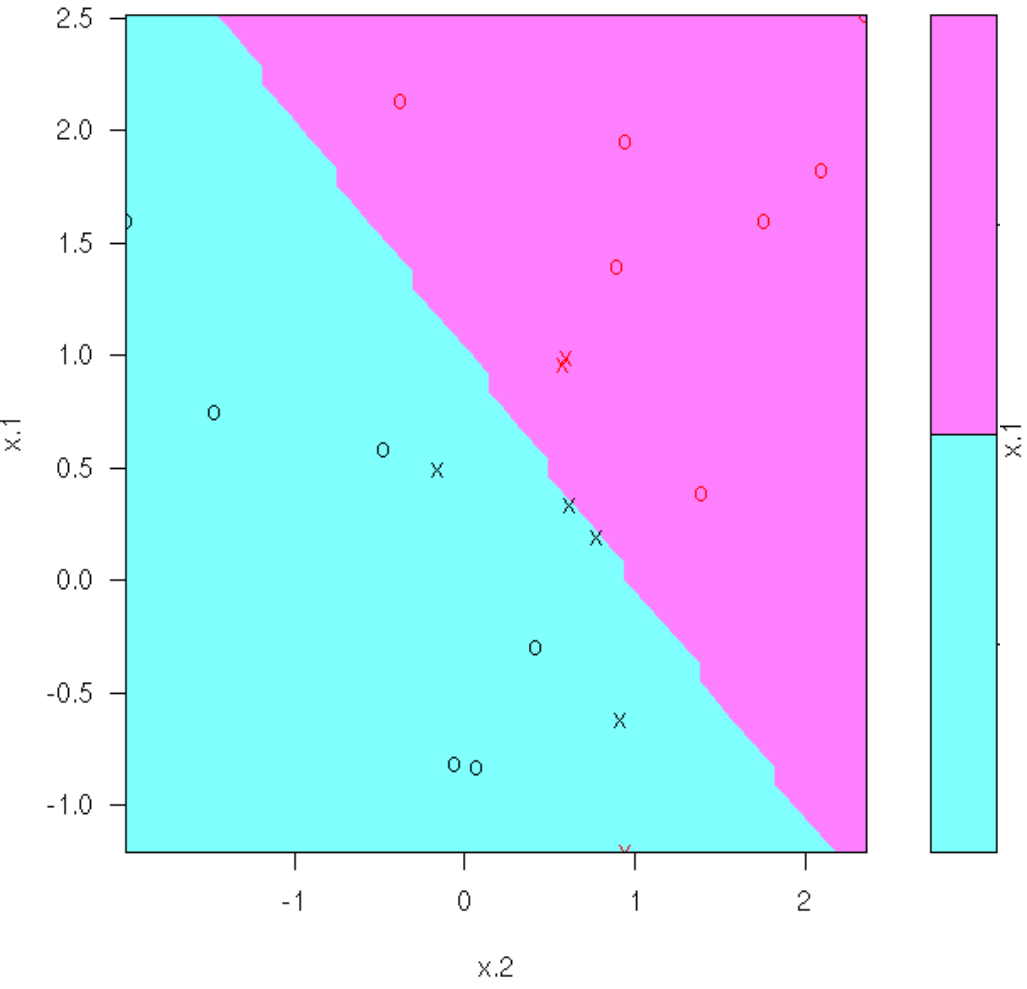
- Cost=0.1:

```
svmfit=svm(y ~ ., data=dat,
kernel='linear', cost=0.1, scale=FALSE)
plot(svmfit,dat)
svmfit$index
summary(svmfit)
```

Support vector classifier

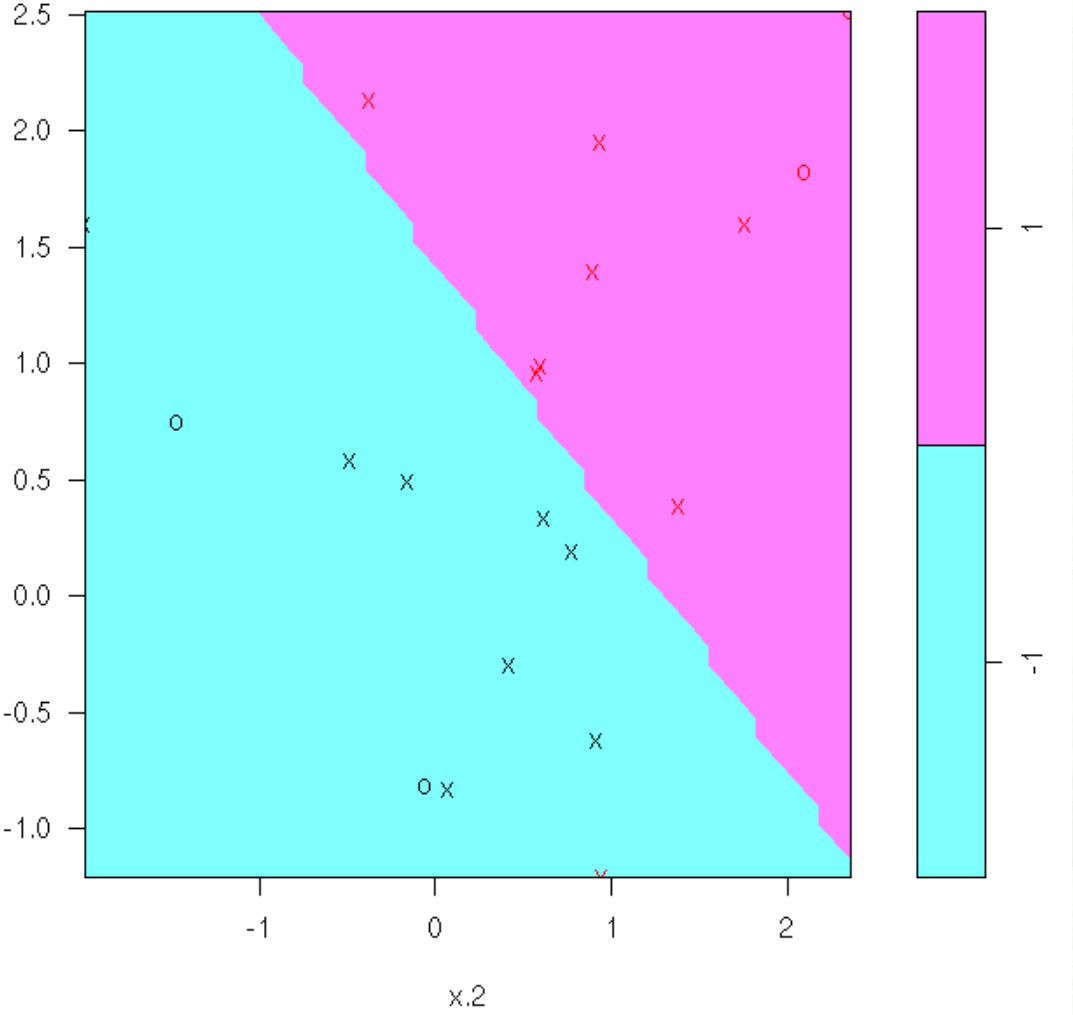
```
      cost:  10
     gamma:  0.5

Number of Support Vectors:  7
```




```
      cost:  0.1
     gamma:  0.5

Number of Support Vectors:  16
```



Support vector classifier

- Finding optimal tuning parameter:
- `tune.out<-tune(svm,y ~ .,data=dat,kernel="linear",ranges=list(cost=c(0.001, 0.01, 0.1, 1,5,10,100)))`
- `bestmod=tune.out$best.model`
- `summary(bestmod)`
- `bestmod$cost`

```
> attributes(tune.out)
$names
[1] "best.parameters"   "best.performance" "method"           "nparcomb"
[5] "train.ind"         "sampling"         "performances"     "best.model"

$class
[1] "tune"
```

Support vector classifier

```
> summary(tune.out)

Parameter tuning of ‘svm’:

- sampling method: 10-fold cross validation

- best parameters:
 cost
  0.1

- best performance: 0.1

- Detailed performance results:
   cost error dispersion
1 1e-03  0.70  0.4216370
2 1e-02  0.70  0.4216370
3 1e-01  0.10  0.2108185
4 1e+00  0.15  0.2415229
5 5e+00  0.15  0.2415229
6 1e+01  0.15  0.2415229
7 1e+02  0.15  0.2415229
```

Support vector classifier

- Construct the test data:
- ```
  set.seed(1)
  ```
- ```
  xtest=matrix(rnorm(20*2), ncol=2)
  ```
- ```
  ytest=sample(c(-1,1), 20, rep=TRUE)
  ```
- ```
  xtest[ytest==1,]=xtest[ytest==1,] + 1
  ```
- ```
  testdat=data.frame(x=xtest,
  y=as.factor(ytest))
  ```

- ```
  ypred=predict(bestmod,testdat)
  ```

- ```
  table(predict=ypred, truth=testdat$y)
  ```
  ```

Support vector classifier

- Construct the test data:
- `set.seed(1)`
- `xtest=matrix(rnorm(20*2), ncol=2)`
- `ytest=sample(c(-1,1), 20, rep=TRUE)`
- `xtest[ytest==1,]=xtest[ytest==1,] + 1`
- `testdat=data.frame(x=xtest, y=as.factor(ytest))`

- `ypred=predict(bestmod,testdat)`

```
        truth
predict -1  1
     -1 10  1
      1  1  8
```

- `table(predict=ypred, truth=testdat$y)`

Support vector classifier

- Construct the test data:
- `set.seed(1)`
- `xtest=matrix(rnorm(20*2), ncol=2)`
- `ytest=sample(c(-1,1), 20, rep=TRUE)`
- `xtest[ytest==1,]=xtest[ytest==`
- `testdat=data.frame(x=xtest, y=as.factor(ytest))`
- Changing parameter:
- `svmfit=svm(y~., data=dat, kernel="linear", cost=.01, scale=FALSE)`
- `ypred=predict(svmfit,testdat)`
- `table(predict=ypred, truth=testdat$y)`

```
         truth
predict -1  1
     -1 10  4
      1  1  5
```

Nonlinear kernel with radial basis

- Kernels available for svm():
- `linear:`
- `Polynomial:`
- `radial basis:`
- `sigmoid:`

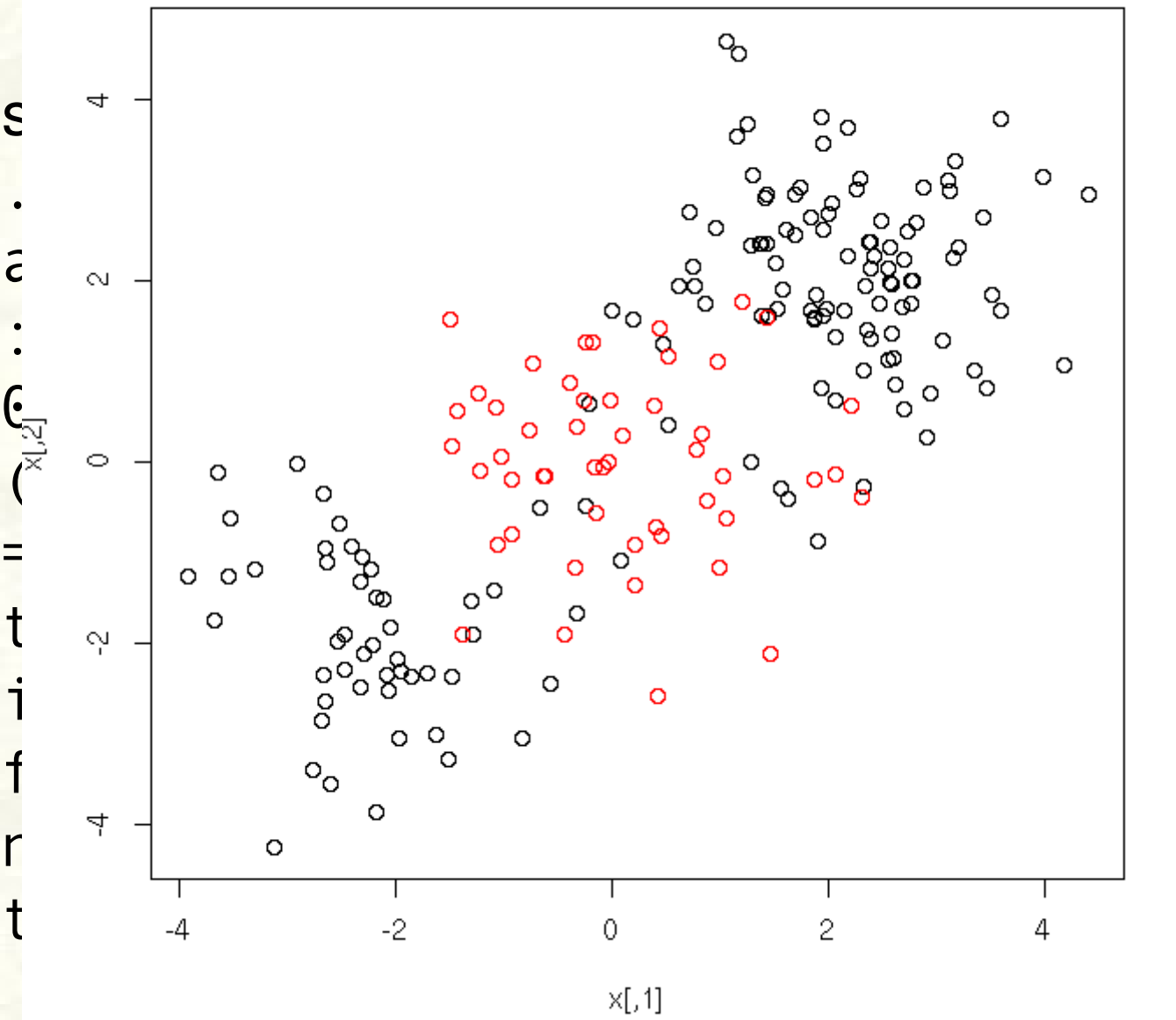
Nonlinear kernel with radial basis

- Construct the test data:

```
set.seed (1)
x=matrix(rnorm(200*2), ncol=2)
x[1:100,]=x[1:100,]+2
x[101:150,]=x[101:150,]-2
y=c(rep(1,150),rep(2,50))
dat=data.frame(x=x,y=as.factor(y))
plot(x, col=y)
train=sample(200,100)
svmfit=svm(y~., data=dat[train,],
kernel="radial", gamma=1, cost =1)
plot(svmfit , dat[train ,])
```

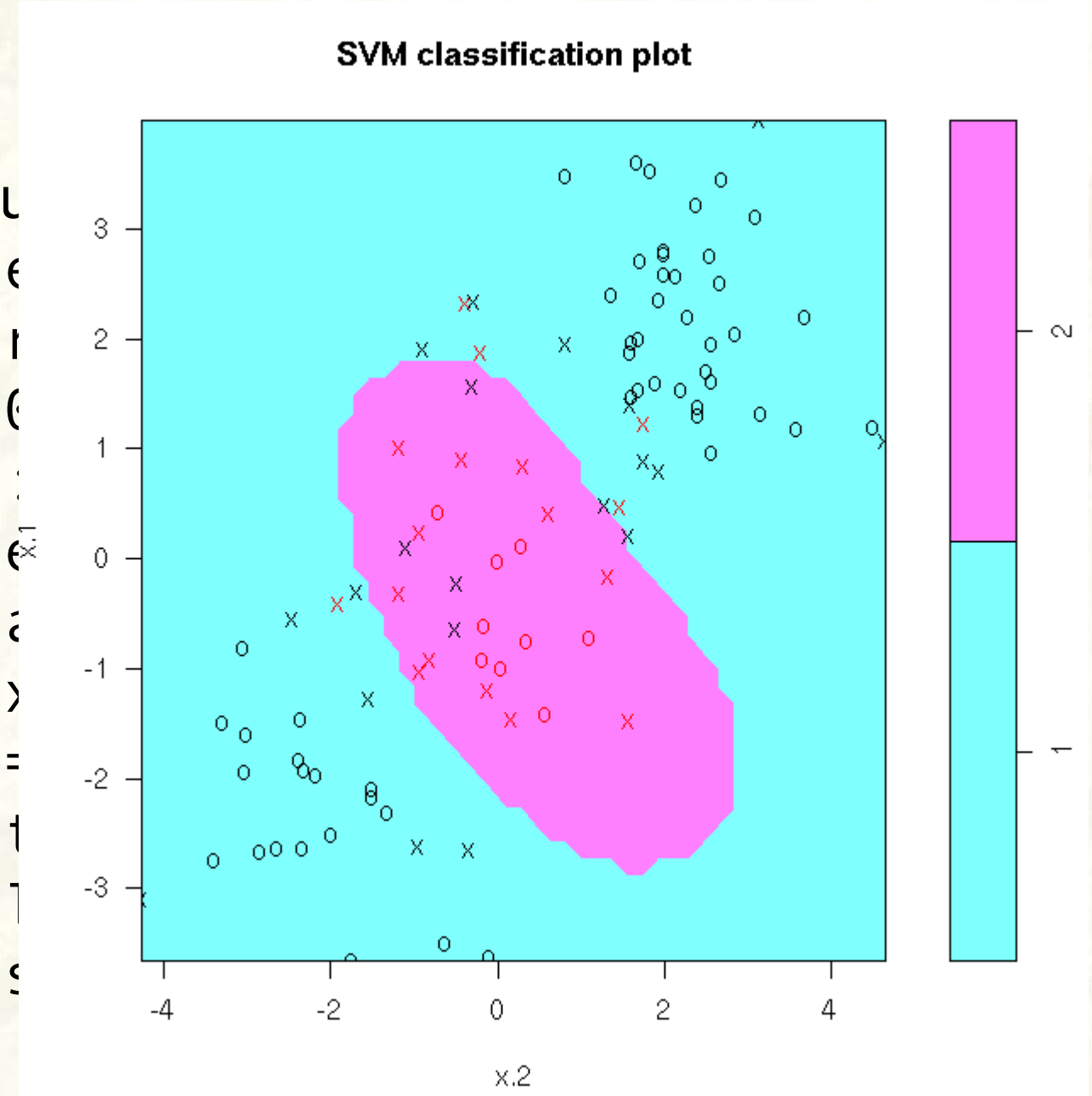
Nonlinear kernel with radial basis

- Cons
- set.
- x=ma
- x[1:
- x[10
- y=c(
- dat=
- plot
- trai
- svmf
  kerr
- plot



Nonlinear kernel with radial basis

- Constru
- set.se
- x=matr
- x[1:10
- x[101:
- y=c(re
- dat=da
- plot(x
- train=
- svmfit
  kernel
- plot(s



Nonlinear kernel with radial basis

```
> summary(svmfit)

Call:
svm(formula = y ~ ., data = dat[train, ], kernel = "radial", gamma = 1,
    cost = 1)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  1
      gamma:  1

Number of Support Vectors:  37

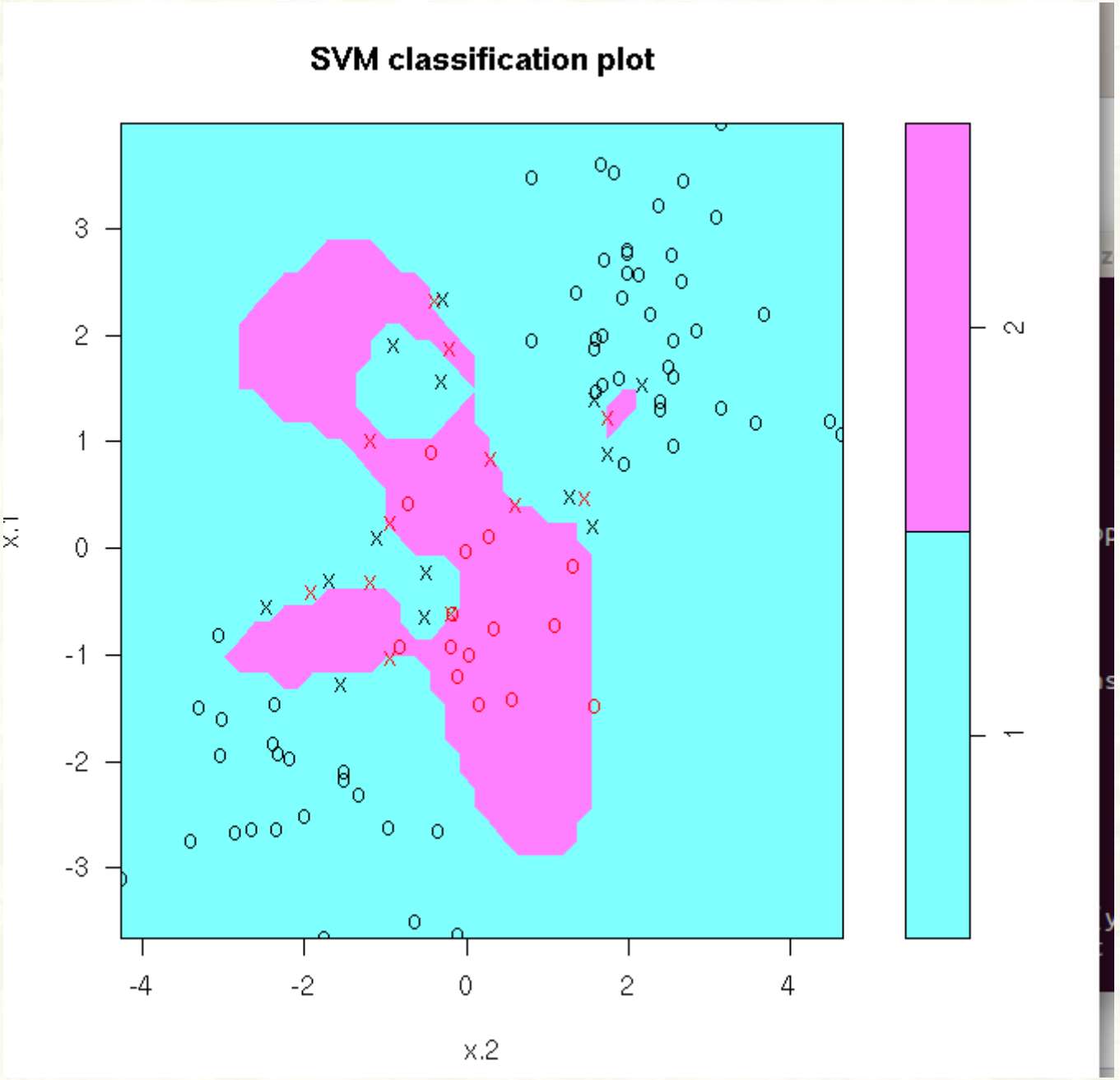
 ( 17 20 )


Number of Classes:  2

Levels:
 1 2
```

Nonlinear kernel with radial basis

- Constr
- set.s
- x=mat
- x[1:1
- x[101
- y=c(r
- dat=d
- plot(
- train
- svmfi
  kerne
- plot(



Nonlinear kernel with radial basis

- Choosing parameter,cost=1,gamma=0.5:
- ```
  tune.out=tune(svm, y~., data=dat[train,],
  kernel="radial",
  ranges=list(cost=c(0.1,1,10,100,1000),
  gamma=c(0.5,1,2,3,4) ))
  ```
- ```
  table(true=dat[-train,"y"],
  pred=predict(tune.out$best.model,
  newx=dat[-train ,]))
  ```

```
     pred
true  1  2
   1 54 23
   2 17  6
```

Nonlinear kernel with radial basis

- Choos
- tune.
  kerne
  range
  gamma
- table
  pred=
  newx=

```
> summary(tune.out)

Parameter tuning of ‘svm’:

- sampling method: 10-fold cross validation

- best parameters:
 cost gamma
    1   0.5

- best performance: 0.13

- Detailed performance results:
   cost gamma error dispersion
1  1e-01   0.5  0.28 0.16865481
2  1e+00   0.5  0.13 0.10593499
3  1e+01   0.5  0.14 0.11737878
4  1e+02   0.5  0.18 0.10327956
5  1e+03   0.5  0.19 0.11005049
6  1e-01   1.0  0.30 0.15634719
7  1e+00   1.0  0.14 0.11737878
8  1e+01   1.0  0.18 0.10327956
9  1e+02   1.0  0.19 0.13703203
```

ROC curves

- Define function to plot ROC curves:
- `rocplot=function(pred, truth, ...){`
- `predob = prediction(pred, truth)`
- `perf = performance(predob, "tpr", "fpr")`
- `plot(perf,...)}`

- `pred:numerical score`
- `truth:true label`

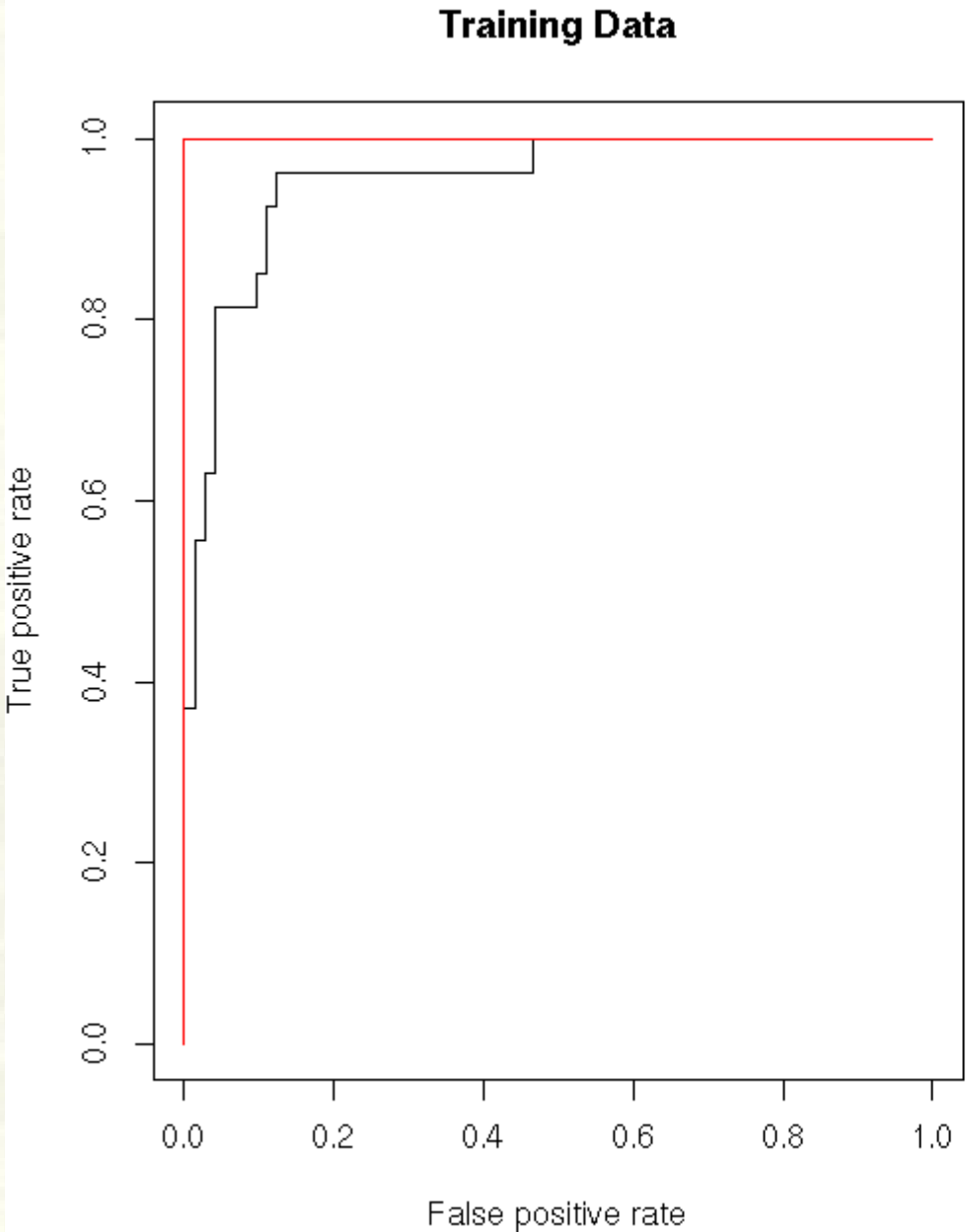
ROC curves

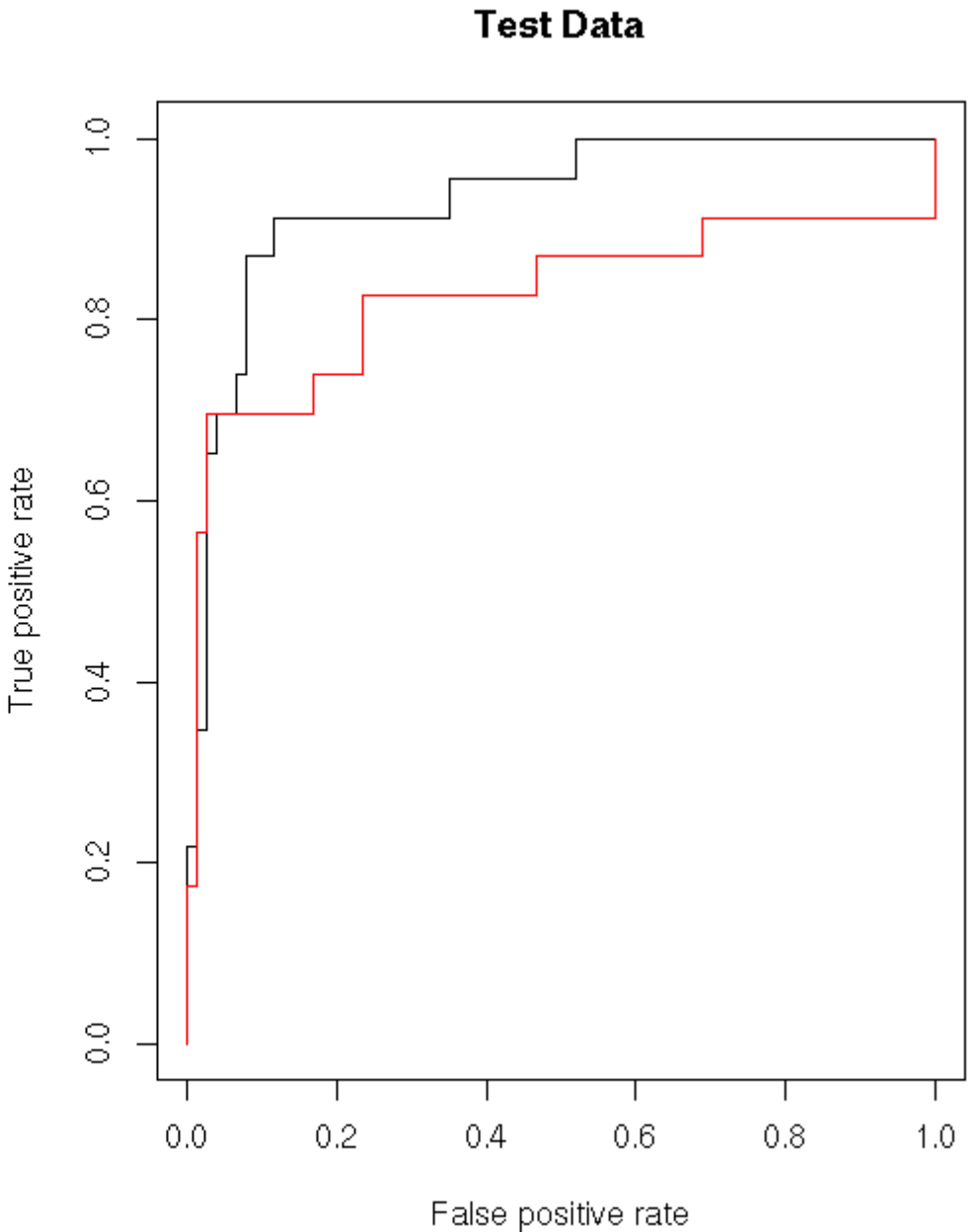
- Plot ROC curves:
- ```
  svmfit.opt=svm(y~., data=dat[train,],
  kernel="radial", gamma=2,
  cost=1,decision.values=T)
  ```

- ```
  fitted=attributes(predict(svmfit.opt,dat[t
  rain,],decision.values=TRUE))
  $decision.values
  ```

- ```
  par(mfrow=c(1,2))
  ```
- ```
  rocplot(fitted,dat[train,"y"],main="Traini
  ng Data")
  ```

ROC curves


Training Data
True positive rate
False positive rate
0.0
0.2
0.4
0.6
0.8
1.0

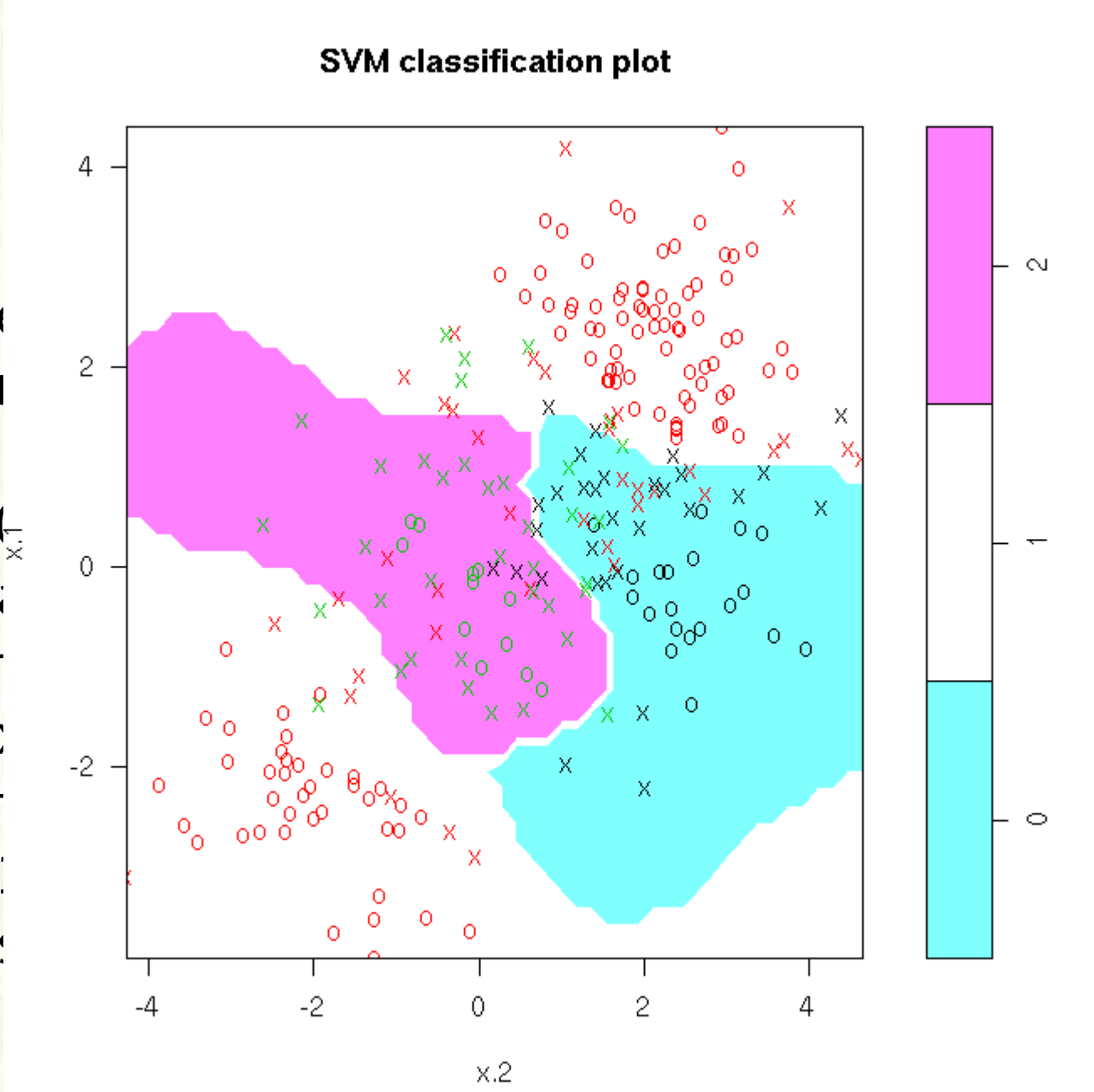

Test Data
True positive rate
False positive rate
0.0
0.2
0.4
0.6
0.8
1.0

SVM with multiple classes

- SVM with multiple classes:
- `set.seed (1)`
- `x=rbind(x, matrix(rnorm(50*2), ncol=2))`
- `y=c(y, rep(0,50))`
- `x[y==0,2]=x[y==0,2]+2`
- `dat=data.frame(x=x, y=as.factor(y))`
- `par(mfrow=c(1,1))`
- `plot(x,col=(y+1))`
- `svmfit=svm(y~., data=dat, kernel="radial", cost=10, gamma=1)`
- `plot(svmfit, dat)`

SVM with multiple classes

- SVM w
- set.s
- x=rbi
- y=c(y
- x[y==
- dat=d
- par(m
- plot(
- svmfi
  cost=
- plot(



Thank you!