Support Vector Machines

Yifei Sun

Contents

```
library(mlbench)
library(ISLR)
library(caret)
library(e1071)
library(kernlab)
```

We use the Pima Indians Diabetes Database for illustration. The outcome is a binary variable `diabetes`.

```
data(PimaIndiansDiabetes2)
dat <- na.omit(PimaIndiansDiabetes2)
dat$diabetes <- factor(dat$diabetes, c("pos", "neg"))

set.seed(2022)
rowTrain <- createDataPartition(y = dat$diabetes,
                                p = 0.75,
                                list = FALSE)
```

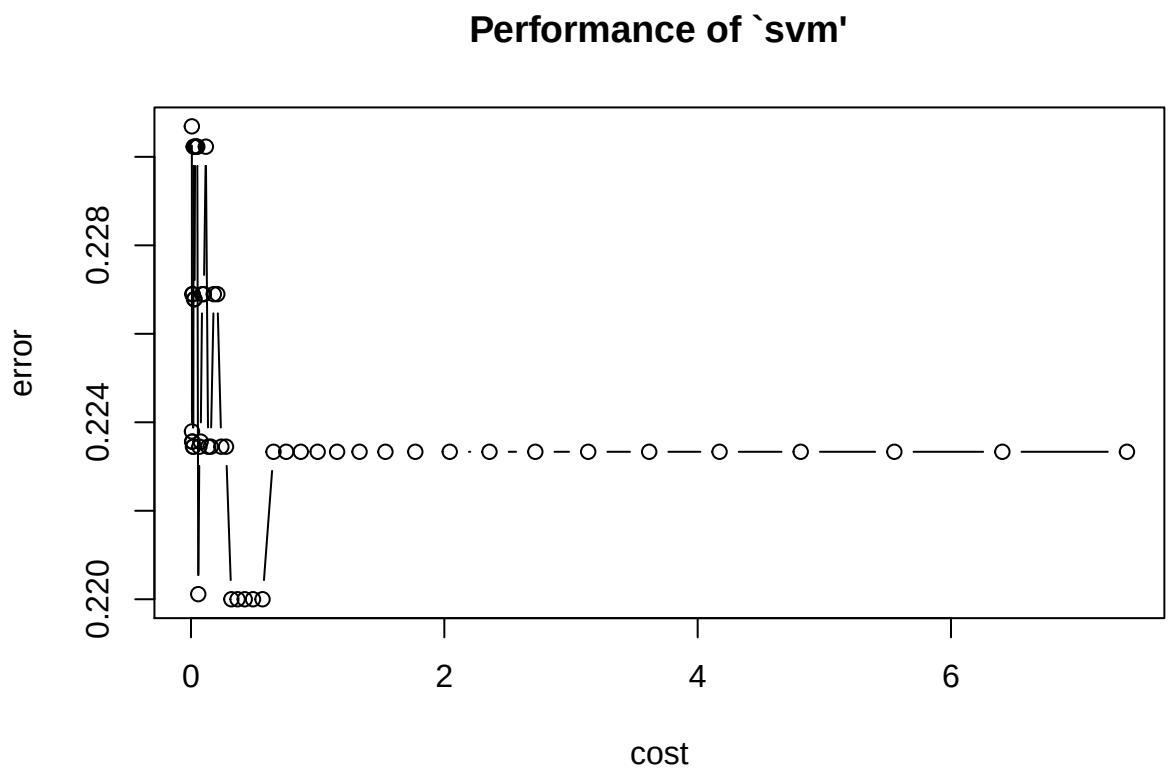
Using e1071

Check https://cran.r-project.org/web/packages/e1071/vignettes/svmdoc.pdf for more details.

Linear boundary

Most real data sets will not be fully separable by a linear boundary. Support vector classifiers with a tuning parameter `cost`, which quantifies the penalty associated with having an observation on the wrong side of the classification boundary, can be used to build a linear boundary.

```
set.seed(1)
linear.tune <- tune.svm(diabetes ~ . ,
                        data = dat[rowTrain,],
                        kernel = "linear",
                        cost = exp(seq(-5,2,len=50)),
                        scale = TRUE)
plot(linear.tune)
```




```
# summary(linear.tune)
linear.tune$best.parameters
```

```
##         cost
## 28 0.3189066
```

```
best.linear <- linear.tune$best.model
summary(best.linear)
```

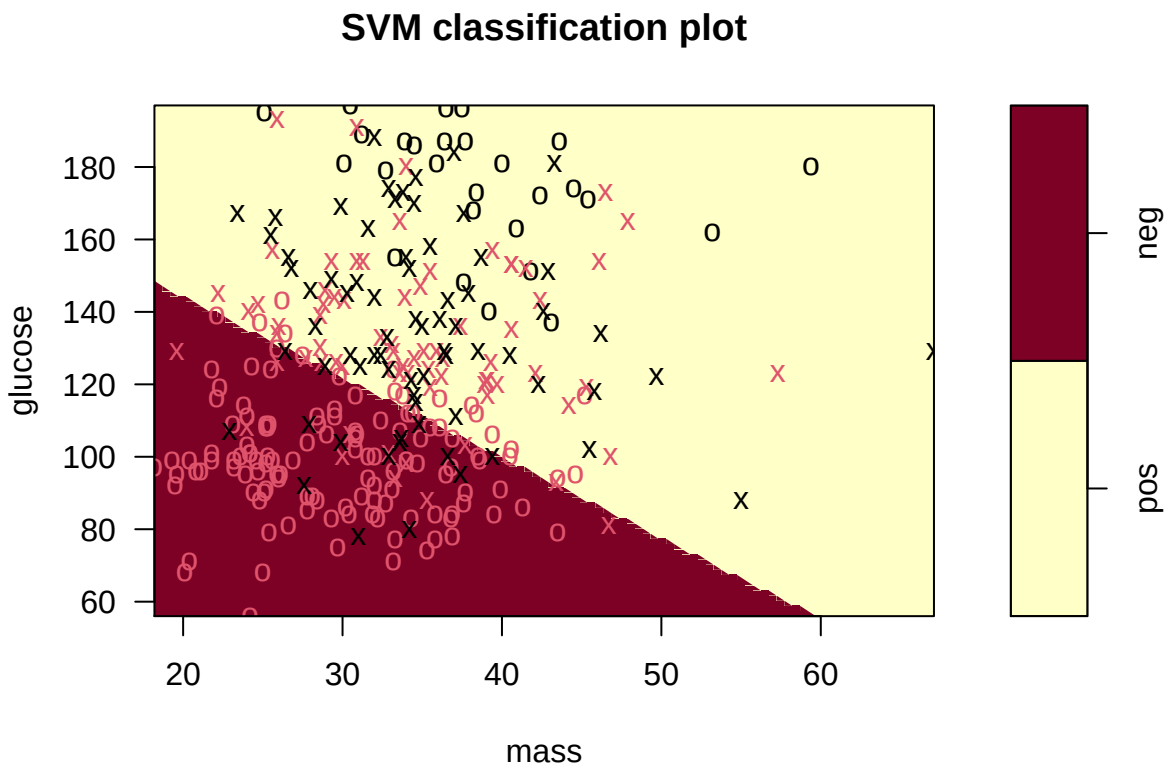
```
##
## Call:
## best.svm(x = diabetes ~ ., data = dat[rowTrain, ], cost = exp(seq(-5,
##     2, len = 50)), kernel = "linear", scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  0.3189066
##
## Number of Support Vectors:  143
##
##  ( 72 71 )
##
##
```

```
## Number of Classes:  2
##
## Levels:
##  pos neg
```

```
pred.linear <- predict(best.linear, newdata = dat[-rowTrain,])

confusionMatrix(data = pred.linear,
                reference = dat$diabetes[-rowTrain])
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction pos neg
##        pos  16   6
##        neg  16  59
##
##                Accuracy : 0.7732
##                  95% CI : (0.677, 0.8521)
##     No Information Rate : 0.6701
##     P-Value [Acc > NIR] : 0.01783
##
##                   Kappa : 0.4428
##
##  Mcnemar's Test P-Value : 0.05501
##
##             Sensitivity : 0.5000
##             Specificity : 0.9077
##          Pos Pred Value : 0.7273
##          Neg Pred Value : 0.7867
##              Prevalence : 0.3299
##          Detection Rate : 0.1649
##    Detection Prevalence : 0.2268
##       Balanced Accuracy : 0.7038
##
##        'Positive' Class : pos
##
```

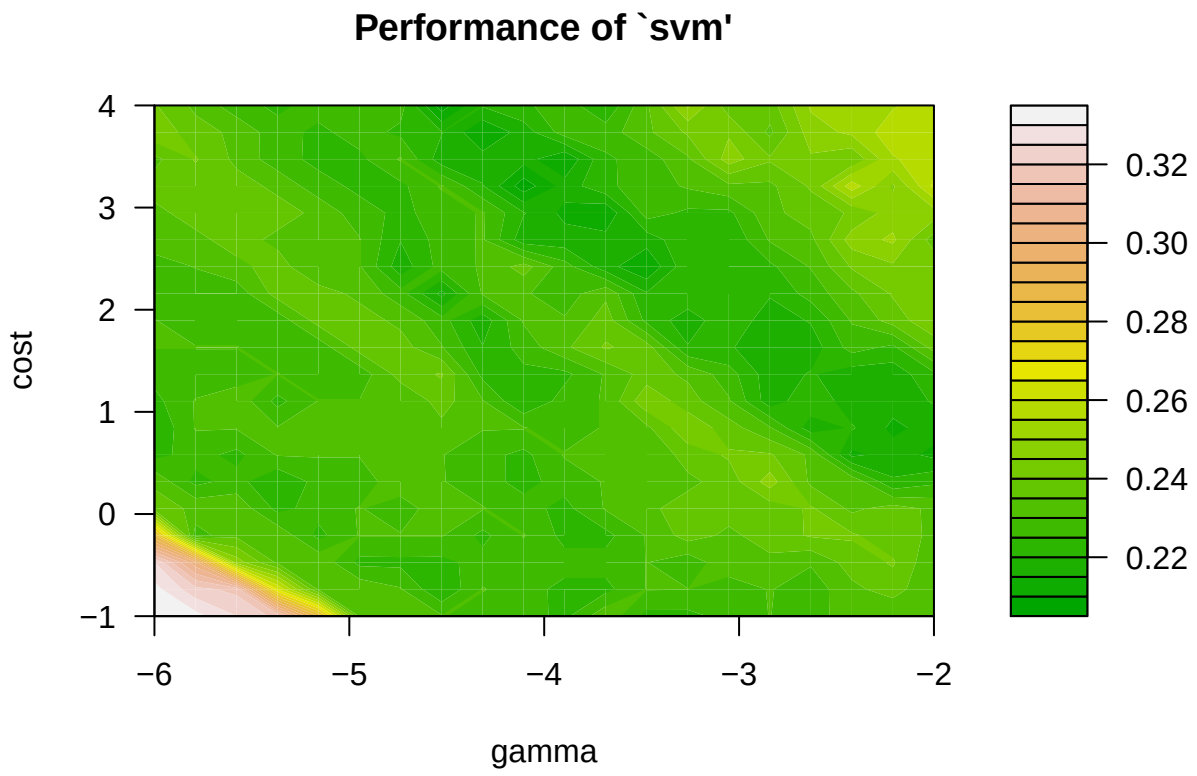
```
plot(best.linear, dat[rowTrain,],
     glucose ~ mass,
     slice = list(pregnant = 5, triceps = 20,
                  insulin = 20, pressure = 75,
                  pedigree = 1, age = 50),
     grid = 100)
```



Radial kernel

Support vector machines can construct classification boundaries that are nonlinear in shape. We use the radial kernel.

```
set.seed(1)
radial.tune <- tune.svm(diabetes ~ . ,
                        data = dat[rowTrain,],
                        kernel = "radial",
                        cost = exp(seq(-1,4,len=20)),
                        gamma = exp(seq(-6,-2,len=20)))

plot(radial.tune, transform.y = log, transform.x = log,
     color.palette = terrain.colors)
```



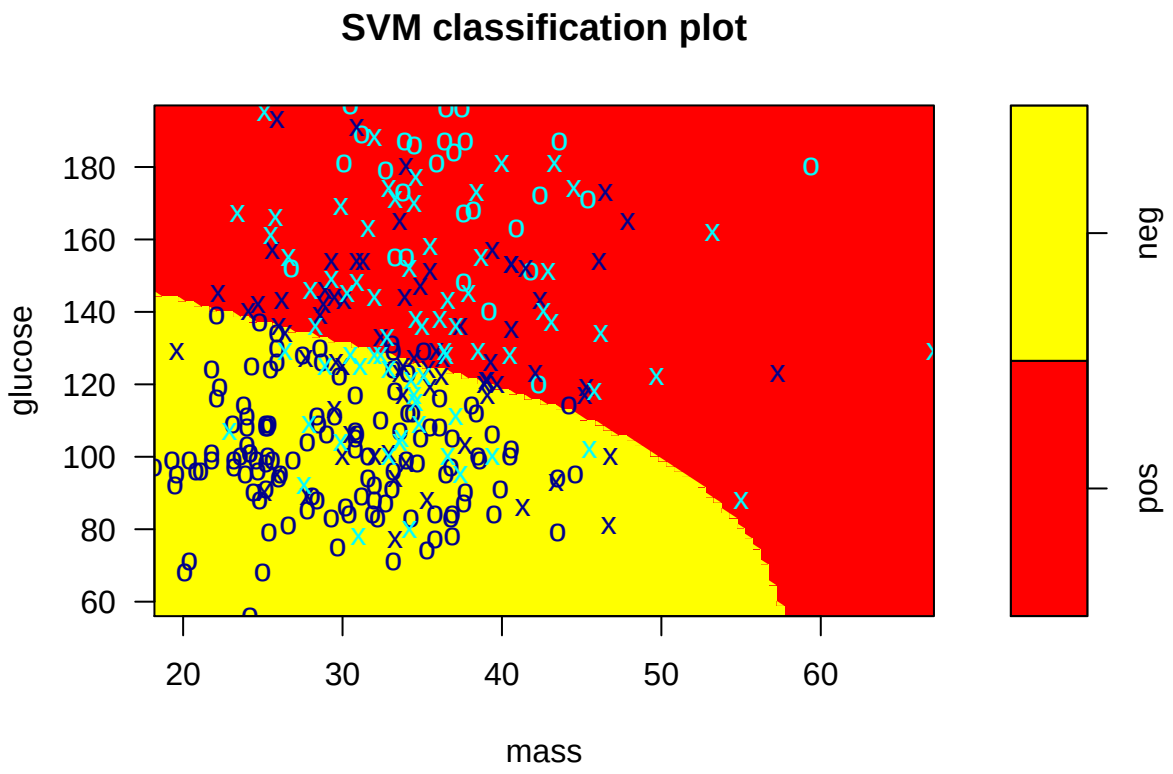

```
# summary(radial.tune)

best.radial <- radial.tune$best.model
summary(best.radial)
```

```
##
## Call:
## best.svm(x = diabetes ~ ., data = dat[rowTrain, ], gamma = exp(seq(-6,
##     -2, len = 20)), cost = exp(seq(-1, 4, len = 20)), kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  24.79213
##
## Number of Support Vectors:  144
##
##  ( 73 71 )
##
##
## Number of Classes:  2
##
## Levels:
##  pos neg
```

```
pred.radial <- predict(best.radial, newdata = dat[-rowTrain,])

confusionMatrix(data = pred.radial,
                reference = dat$diabetes[-rowTrain])
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction pos neg
##        pos  15   5
##        neg  17  60
##
##                Accuracy : 0.7732
##                  95% CI : (0.677, 0.8521)
##     No Information Rate : 0.6701
##     P-Value [Acc > NIR] : 0.01783
##
##                   Kappa : 0.433
##
##  Mcnemar's Test P-Value : 0.01902
##
##             Sensitivity : 0.4688
##             Specificity : 0.9231
##          Pos Pred Value : 0.7500
##          Neg Pred Value : 0.7792
##              Prevalence : 0.3299
##          Detection Rate : 0.1546
##    Detection Prevalence : 0.2062
##       Balanced Accuracy : 0.6959
##
##        'Positive' Class : pos
##
```

```
plot(best.radial, dat[rowTrain,],
     glucose ~ mass,
     slice = list(pregnant = 5, triceps = 20,
                  insulin = 20, pressure = 75,
                  pedigree = 1, age = 50),
     grid = 100,
     symbolPalette = c("cyan","darkblue"),
     color.palette = heat.colors)
```



Using `kernlab`

Check https://cran.r-project.org/web/packages/kernlab/vignettes/kernlab.pdf for more details.

```
x_train <- as.matrix(dat[rowTrain, 1:8])
x_test <- as.matrix(dat[-rowTrain, 1:8])

linear <- ksvm(x = x_train,
               y = dat$diabetes[rowTrain],
               type = "C-svc",
               kernel = "vanilladot",
               C = 1,
               scaled = TRUE)
```

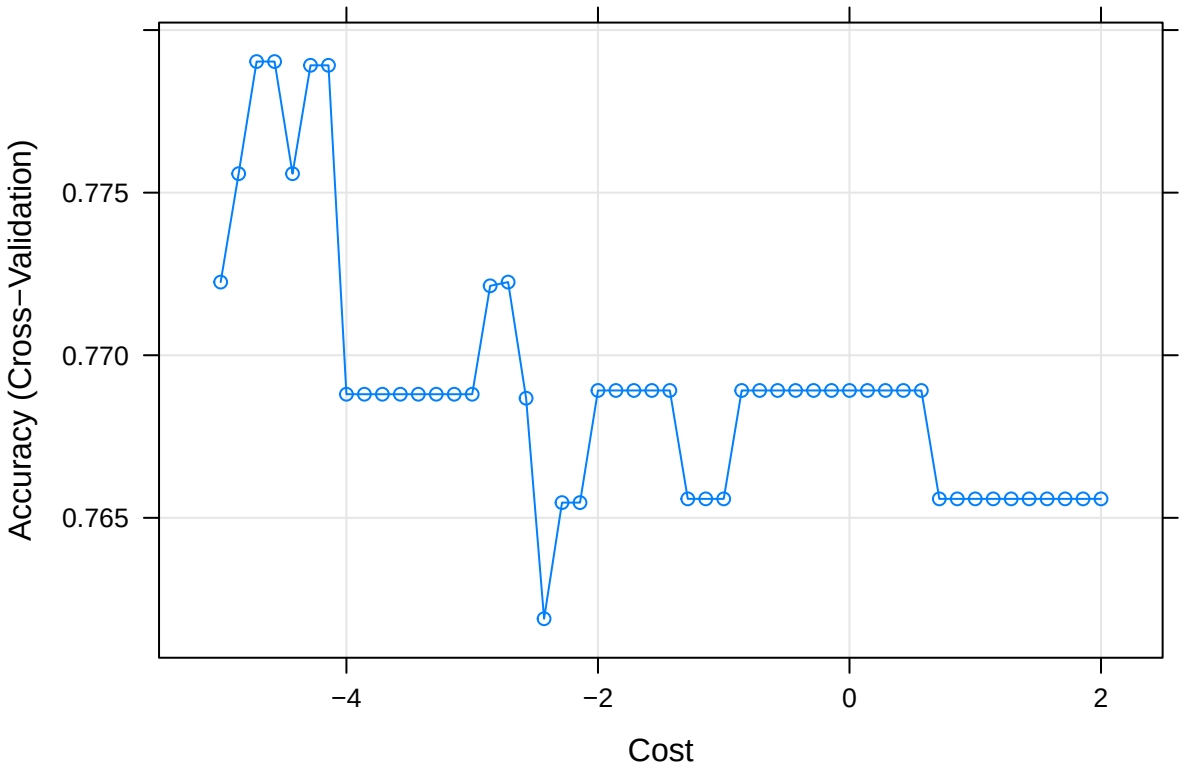
```
##  Setting default kernel parameters
```

```
pred.linear2 <- predict(linear, newdata = x_test)


# "?dots" for definition of kernel functions

set.seed(1)
rbf <- ksvm(x = x_train,
            y = dat$diabetes[rowTrain],
```

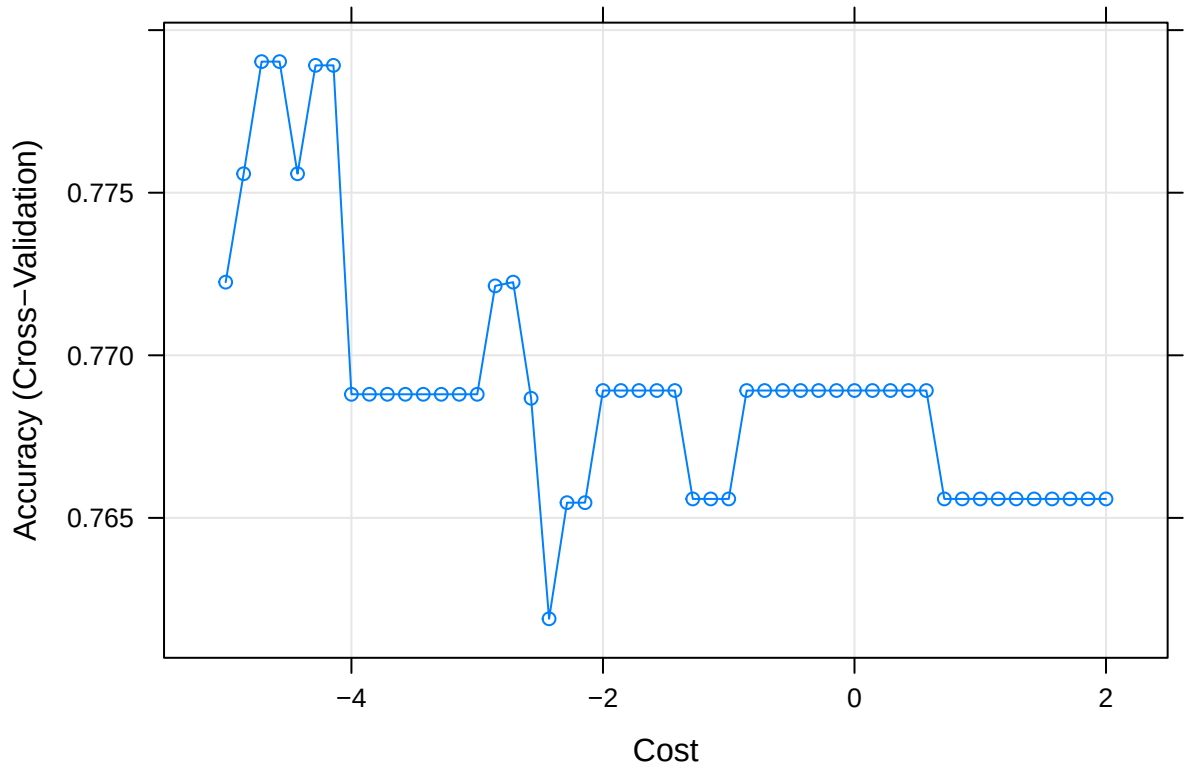
```
            type = "C-svc",
            kernel = "rbfdot" ,
            kpar = "automatic",
            C = 1)

pred.rbf <- predict(rbf, newdata = x_test)
```

Using caret

```
ctrl <- trainControl(method = "cv")

# kernlab
set.seed(1)
svml.fit <- train(diabetes ~ . ,
                  data = dat[rowTrain,],
                  method = "svmLinear",
                  # preProcess = c("center", "scale"),
                  tuneGrid = data.frame(C = exp(seq(-5,2,len=50))),
                  trControl = ctrl)

plot(svml.fit, highlight = TRUE, xTrans = log)
```

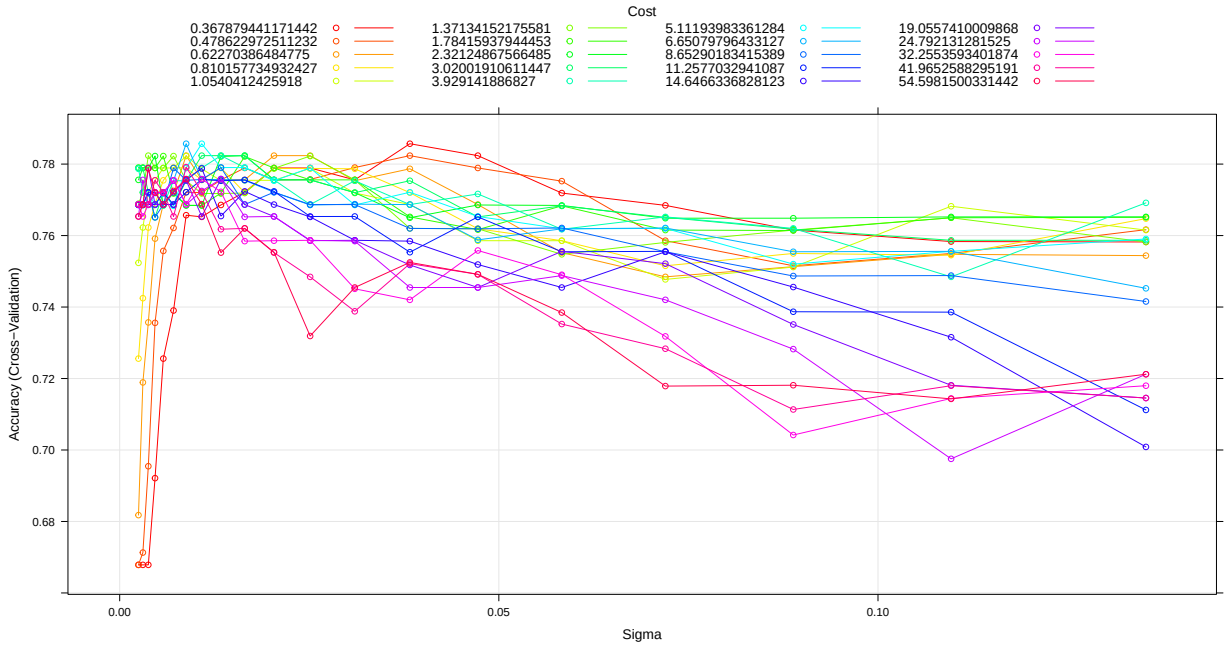


```
# e1071
set.seed(1)
svml.fit2 <- train(diabetes ~ . ,
                   data = dat[rowTrain,],
                   method = "svmLinear2",
                   tuneGrid = data.frame(cost = exp(seq(-5,2,len=50))),
                   trControl = ctrl)

plot(svml.fit2, highlight = TRUE, xTrans = log)
```



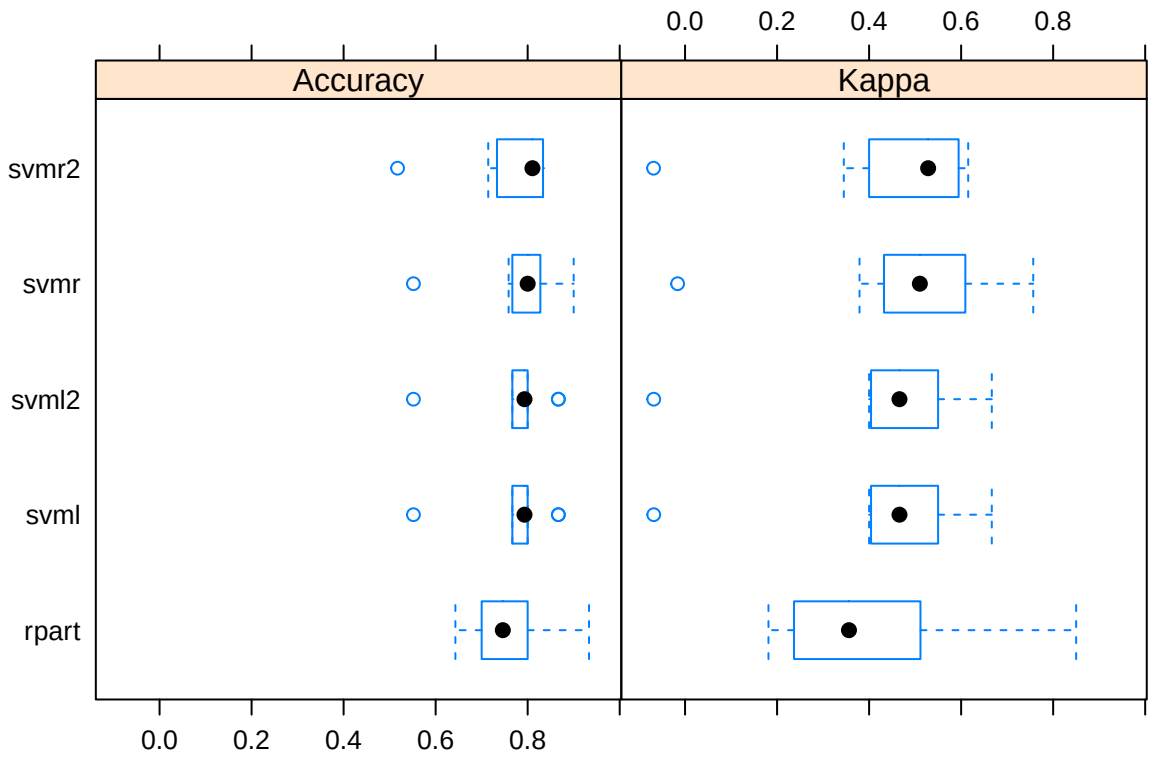

```
svmr.grid <- expand.grid(C = exp(seq(-1,4,len=20)),
                         sigma = exp(seq(-6,-2,len=20)))

# tunes over both cost and sigma
set.seed(1)
svmr.fit <- train(diabetes ~ . , dat,
                  subset = rowTrain,
                  method = "svmRadialSigma",
                  tuneGrid = svmr.grid,
                  trControl = ctrl)

myCol<- rainbow(20)
myPar <- list(superpose.symbol = list(col = myCol),
              superpose.line = list(col = myCol))
```

```
plot(svmr.fit, highlight = TRUE, par.settings = myPar)
```




```
# tune over cost and uses a single value of sigma based on kernlab's sigest function
set.seed(1)
svmr.fit2 <- train(diabetes ~ . , dat,
                   subset = rowTrain,
                   method = "svmRadialCost",
                   tuneGrid = data.frame(C = exp(seq(-3,3,len=20))),
                   trControl = ctrl)

# Platt's probabilistic outputs; use with caution
set.seed(1)
svmr.fit3 <- train(diabetes ~ . , dat,
                   subset = rowTrain,
                   method = "svmRadialCost",
                   tuneGrid = data.frame(C = exp(seq(-3,3,len=20))),
                   trControl = ctrl,
                   prob.model = TRUE)
# predict(svmr.fit3, newdata = x_test, type = "prob")

set.seed(1)
rpart.fit <- train(diabetes ~ . , dat,
                   subset = rowTrain,
                   method = "rpart",
                   tuneLength = 50,
                   trControl = ctrl)
```

```
resamp <- resamples(list(svmr = svmr.fit, svmr2 = svmr.fit2,
                         svml = svml.fit, svml2 = svml.fit2,
                         rpart = rpart.fit))
bwplot(resamp)
```



We finally look at the test data performance.

```
pred.svml <- predict(svml.fit, newdata = dat[-rowTrain,])
pred.svmr <- predict(svmr.fit, newdata = dat[-rowTrain,])

confusionMatrix(data = pred.svml,
                reference = dat$diabetes[-rowTrain])
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction pos neg
##        pos  14   4
##        neg  18  61
##
##                Accuracy : 0.7732
##                  95% CI : (0.677, 0.8521)
##     No Information Rate : 0.6701
##     P-Value [Acc > NIR] : 0.017826
##
##                   Kappa : 0.4229
##
##  Mcnemar's Test P-Value : 0.005578
##
##             Sensitivity : 0.4375
##             Specificity : 0.9385
```

```
##          Pos Pred Value : 0.7778
##          Neg Pred Value : 0.7722
##              Prevalence : 0.3299
##          Detection Rate : 0.1443
##    Detection Prevalence : 0.1856
##       Balanced Accuracy : 0.6880
##
##        'Positive' Class : pos
##
```

```
confusionMatrix(data = pred.svmr,
                reference = dat$diabetes[-rowTrain])
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction pos neg
##        pos  17   7
##        neg  15  58
##
##                Accuracy : 0.7732
##                  95% CI : (0.677, 0.8521)
##     No Information Rate : 0.6701
##     P-Value [Acc > NIR] : 0.01783
##
##                   Kappa : 0.4523
##
##  Mcnemar's Test P-Value : 0.13559
##
##             Sensitivity : 0.5312
##             Specificity : 0.8923
##          Pos Pred Value : 0.7083
##          Neg Pred Value : 0.7945
##              Prevalence : 0.3299
##          Detection Rate : 0.1753
##    Detection Prevalence : 0.2474
##       Balanced Accuracy : 0.7118
##
##        'Positive' Class : pos
##
```