Support Vector Machines

Yifei Sun, Runze Cui

Contents

```r
library(mlbench)
library(ISLR)
library(caret)
library(tidymodels)
library(e1071)
library(kernlab)
library(ggrepel)
```

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

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

set.seed(111111)
data_split <- initial_split(dat, prop = 0.75)

training_data <- training(data_split)
testing_data <- testing(data_split)
```

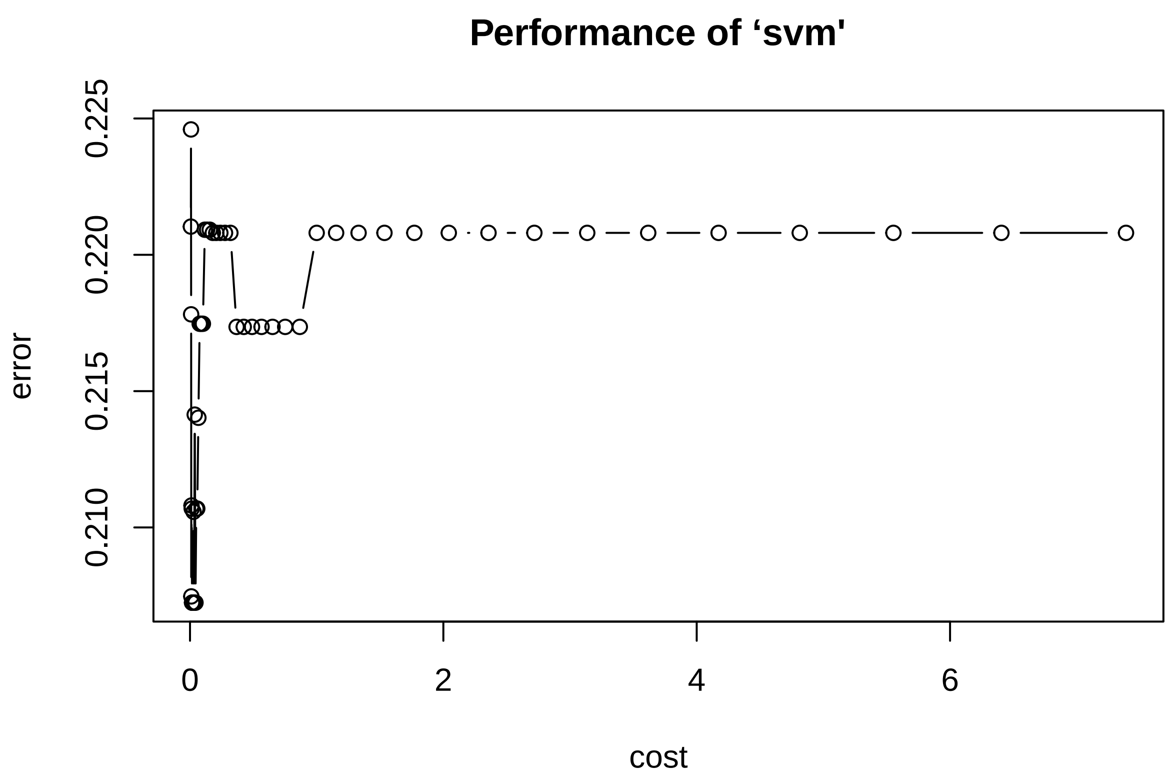
Using e1071

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.

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



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

```
##         cost
## 7 0.01587742
```

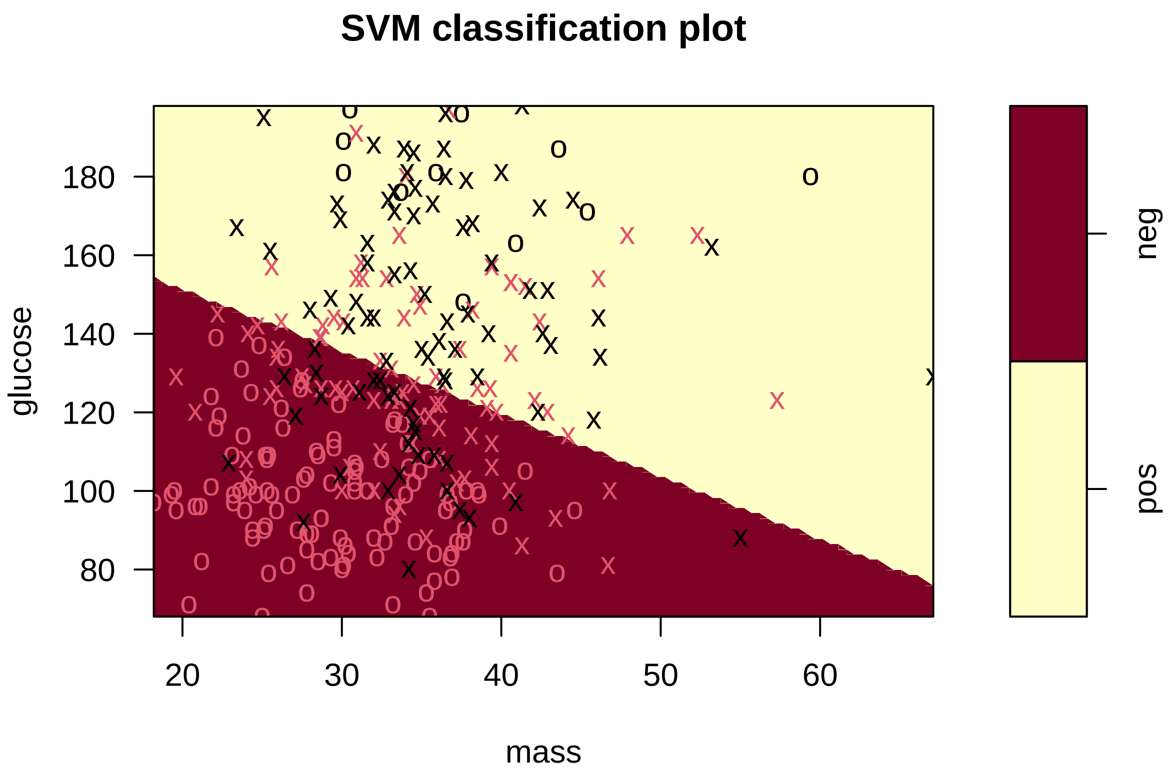
```
best.linear <- linear.tune$best.model
summary(best.linear)
```

```
##
## Call:
## best.svm(x = diabetes ~ ., data = training_data, cost = exp(seq(-5,
##     2, len = 50)), kernel = "linear", scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  0.01587742
##
## Number of Support Vectors:  174
##
##  ( 88 86 )
##
##
## Number of Classes:  2
##
## Levels:
##  pos neg
```

```r
pred.linear <- predict(best.linear, newdata = testing_data)

confusionMatrix(data = pred.linear,
                reference = testing_data$diabetes)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction pos neg
##        pos  17   7
##        neg  16  58
##
##                Accuracy : 0.7653
##                  95% CI : (0.6689, 0.845)
##     No Information Rate : 0.6633
##     P-Value [Acc > NIR] : 0.01894
##
##                   Kappa : 0.4368
##
##  Mcnemar's Test P-Value : 0.09529
##
##             Sensitivity : 0.5152
##             Specificity : 0.8923
##          Pos Pred Value : 0.7083
##          Neg Pred Value : 0.7838
##              Prevalence : 0.3367
##          Detection Rate : 0.1735
##    Detection Prevalence : 0.2449
##       Balanced Accuracy : 0.7037
##
##        'Positive' Class : pos
##
```

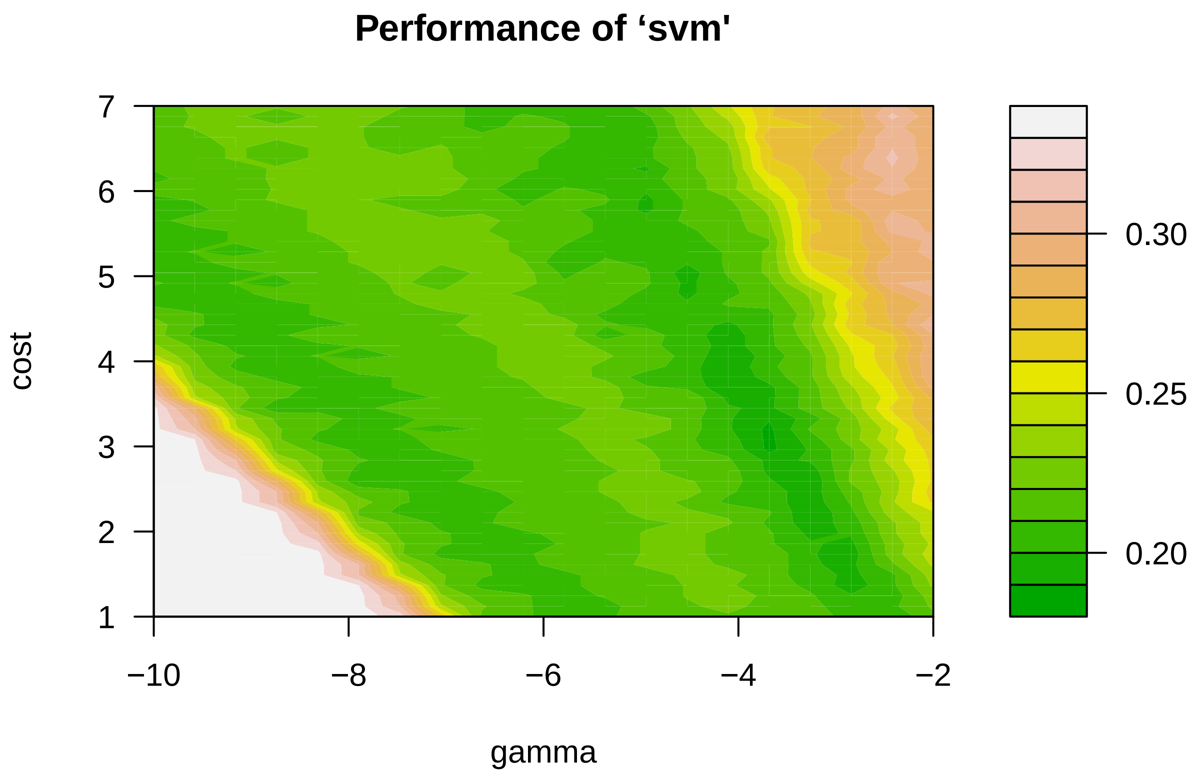
```r
plot(best.linear, training_data,
     glucose ~ mass,
     slice = list(pregnant = 5, triceps = 20,
                  insulin = 20, pressure = 75,
                  pedigree = 1, age = 50),
     grid = 100)
```



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 = training_data,
                        kernel = "radial",
                        cost = exp(seq(1, 7, len = 50)),
                        gamma = exp(seq(-10, -2,len = 20)))

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



```r
# summary(radial.tune)

radial.tune$best.parameters
```

```
##        gamma     cost
## 336 0.025117 19.28222
```

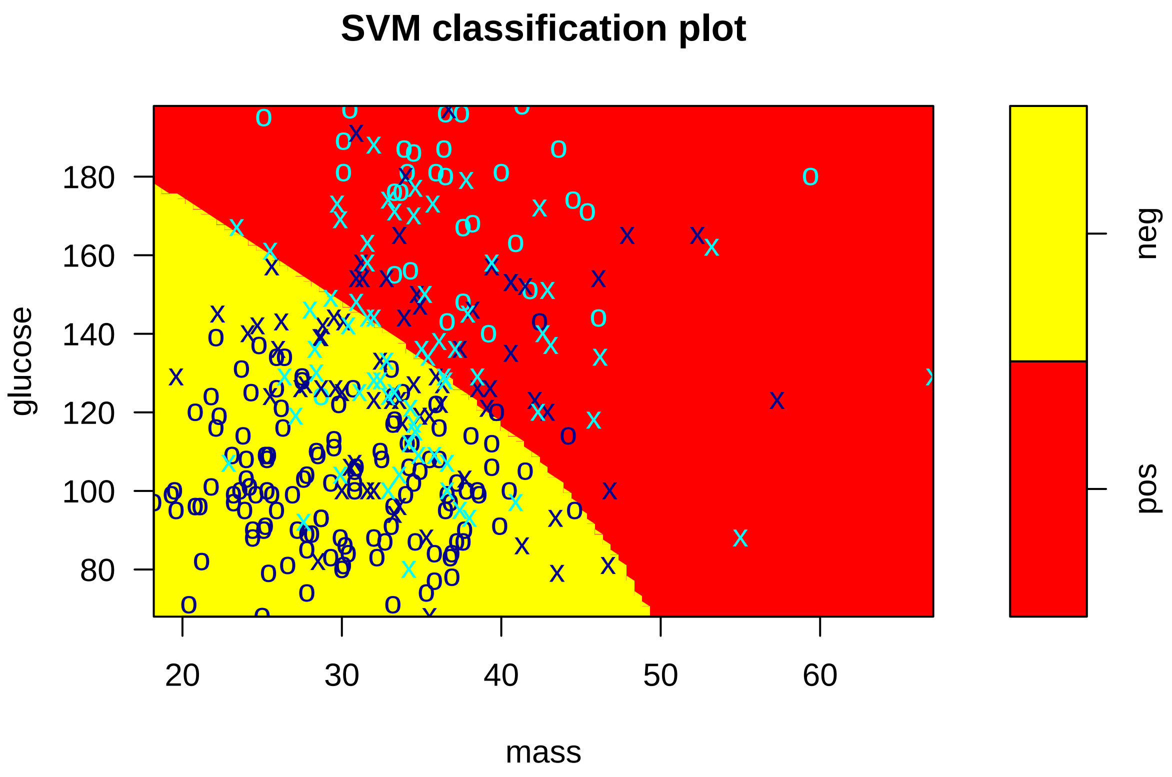
```r
best.radial <- radial.tune$best.model
summary(best.radial)
```

```
##
## Call:
## best.svm(x = diabetes ~ ., data = training_data, gamma = exp(seq(-10,
##     -2, len = 20)), cost = exp(seq(1, 7, len = 50)), kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  19.28222
##
## Number of Support Vectors:  138
##
##  ( 72 66 )
##
##
## Number of Classes:  2
##
## Levels:
##  pos neg
```

```r
pred.radial <- predict(best.radial, newdata = testing_data)

confusionMatrix(data = pred.radial,
                reference = testing_data$diabetes)
```

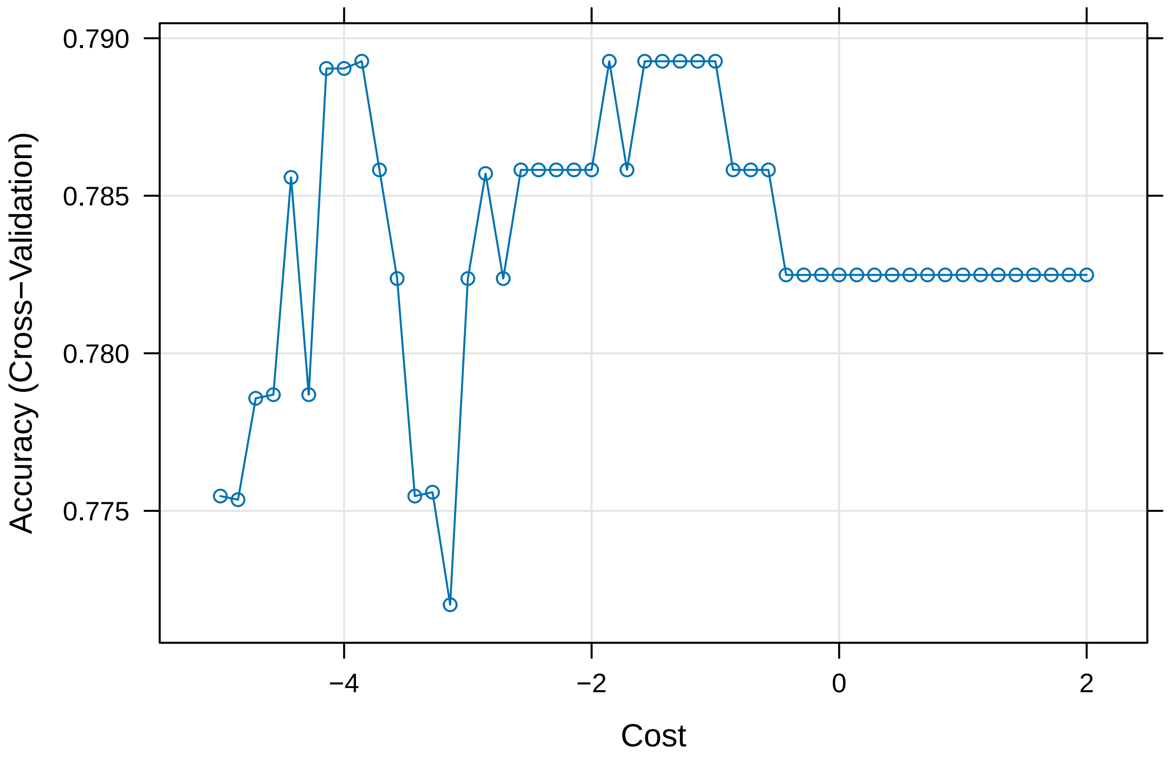
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction pos neg
##        pos  15   8
##        neg  18  57
##
##                Accuracy : 0.7347
##                  95% CI : (0.6359, 0.8188)
##     No Information Rate : 0.6633
##     P-Value [Acc > NIR] : 0.08044
##
##                   Kappa : 0.3582
##
##  Mcnemar's Test P-Value : 0.07756
##
##             Sensitivity : 0.4545
##             Specificity : 0.8769
##          Pos Pred Value : 0.6522
##          Neg Pred Value : 0.7600
##              Prevalence : 0.3367
##          Detection Rate : 0.1531
##    Detection Prevalence : 0.2347
##       Balanced Accuracy : 0.6657
##
##        'Positive' Class : pos
##
```

```r
plot(best.radial, training_data,
     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)
```

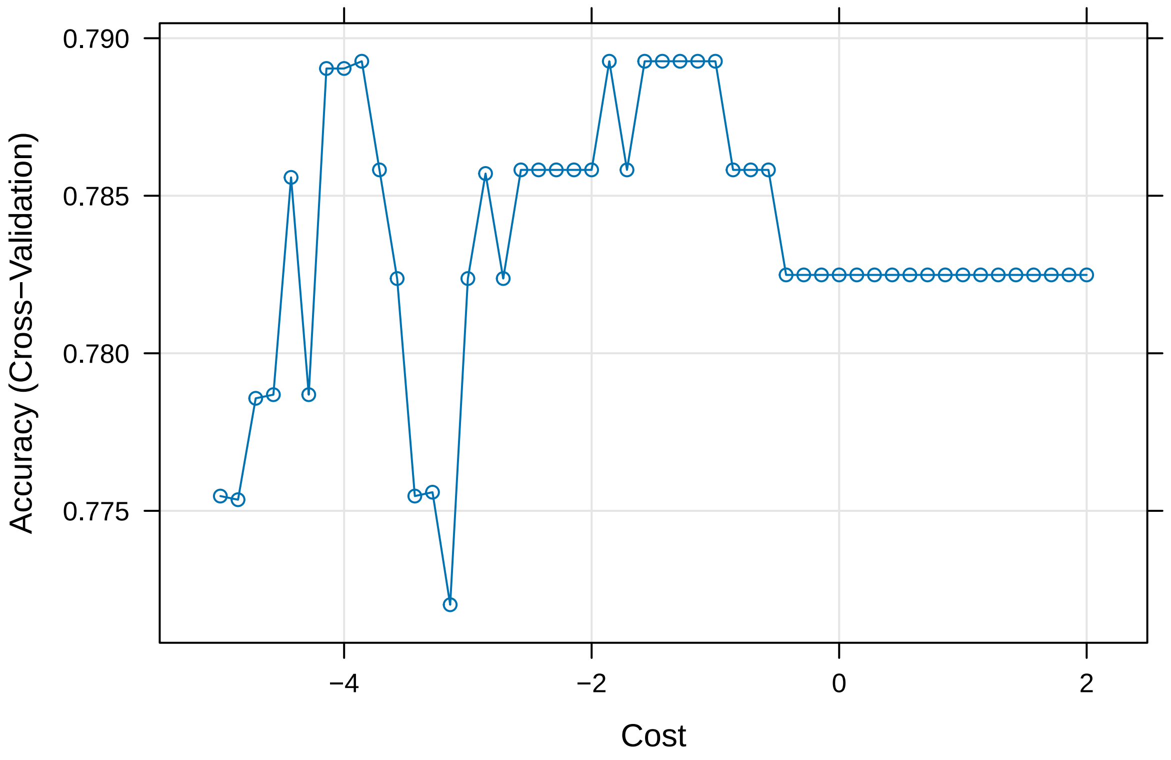


Using caret

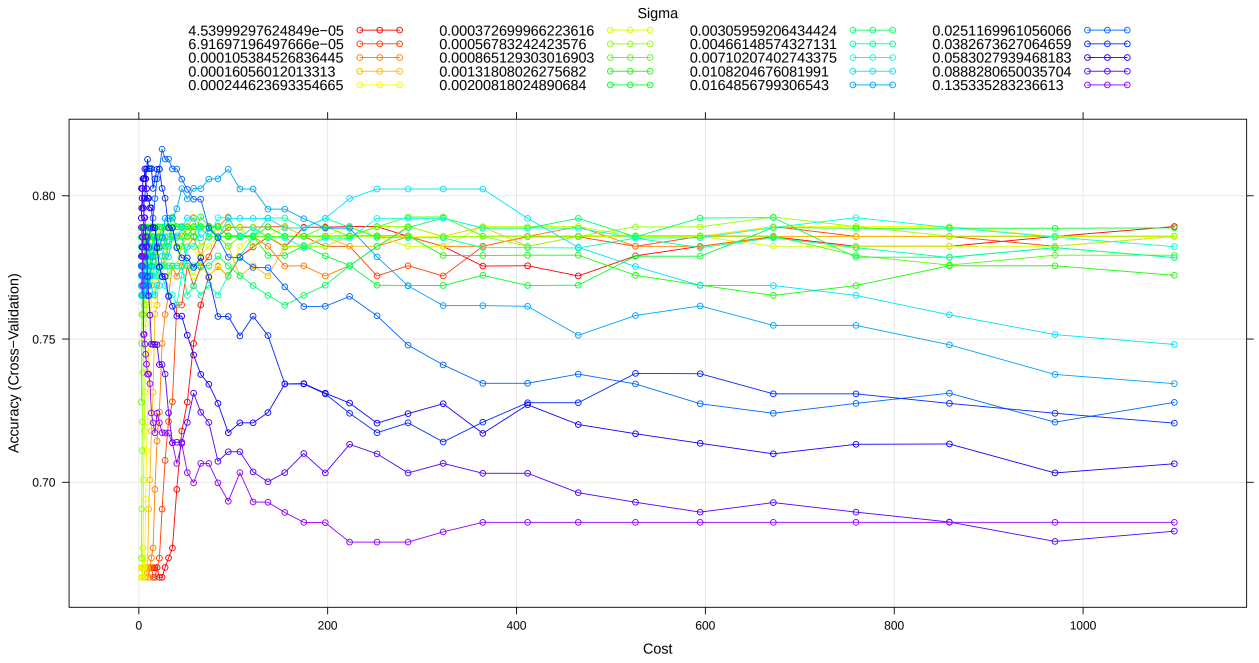
```r
ctrl <- trainControl(method = "cv")

# kernlab
set.seed(1)
svml.fit <- train(diabetes ~ . ,
                  data = training_data,
                  method = "svmLinear",
                  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 = training_data,
                   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, 7, len = 50)),
                         sigma = exp(seq(-10, -2, len = 20)))

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

myCol <- rainbow(25)
myPar <- list(superpose.symbol = list(col = myCol),
              superpose.line = list(col = myCol))

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

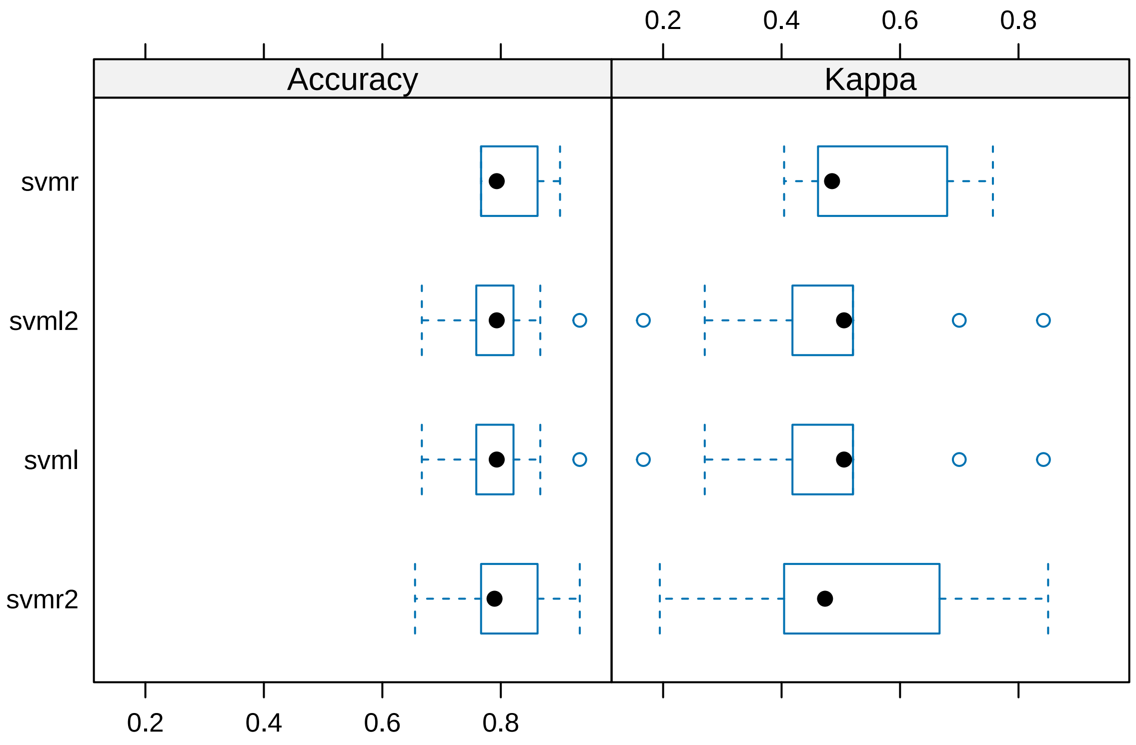


```
# tune over cost and uses a single value of sigma based on kernlab's sigest function
set.seed(1)
svmr.fit2 <- train(diabetes ~ . , data = training_data,
                   method = "svmRadialCost",
                   tuneGrid = data.frame(C = exp(seq(-3, 3, len = 20))),
                   trControl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

# Platt’s probabilistic outputs; use with caution
set.seed(1)
svmr.fit3 <- train(diabetes ~ . , data = training_data,
                   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")
```

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

summary(resamp)
```

```
##
## Call:
## summary.resamples(object = resamp)
##
## Models: svmr, svmr2, svml, svml2
## Number of resamples: 10
##
## Accuracy
##            Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## svmr  0.7666667 0.7714286 0.7931034 0.8162726 0.8620690 0.9000000    0
## svmr2 0.6551724 0.7666667 0.7894089 0.7955829 0.8448276 0.9333333    0
## svml  0.6666667 0.7606322 0.7931034 0.7892693 0.8143473 0.9333333    0
## svml2 0.6666667 0.7606322 0.7931034 0.7892693 0.8143473 0.9333333    0
##
## Kappa
##            Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## svmr  0.4042553 0.4615385 0.4852071 0.5558390 0.6753662 0.7567568    0
## svmr2 0.1944444 0.4185761 0.4733728 0.5122545 0.6298343 0.8500000    0
## svml  0.1666667 0.4218617 0.5054264 0.4880550 0.5202452 0.8421053    0
## svml2 0.1666667 0.4218617 0.5054264 0.4880550 0.5202452 0.8421053    0
```

```
bwplot(resamp)
```



We finally look at the test data performance.

```
pred.svml <- predict(svml.fit, newdata = testing_data)
pred.svmr <- predict(svmr.fit, newdata = testing_data)

confusionMatrix(data = pred.svml,
                reference = testing_data$diabetes)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction pos neg
##        pos  17   7
##        neg  16  58
##
##                Accuracy : 0.7653
##                  95% CI : (0.6689, 0.845)
##     No Information Rate : 0.6633
##     P-Value [Acc > NIR] : 0.01894
##
##                   Kappa : 0.4368
```

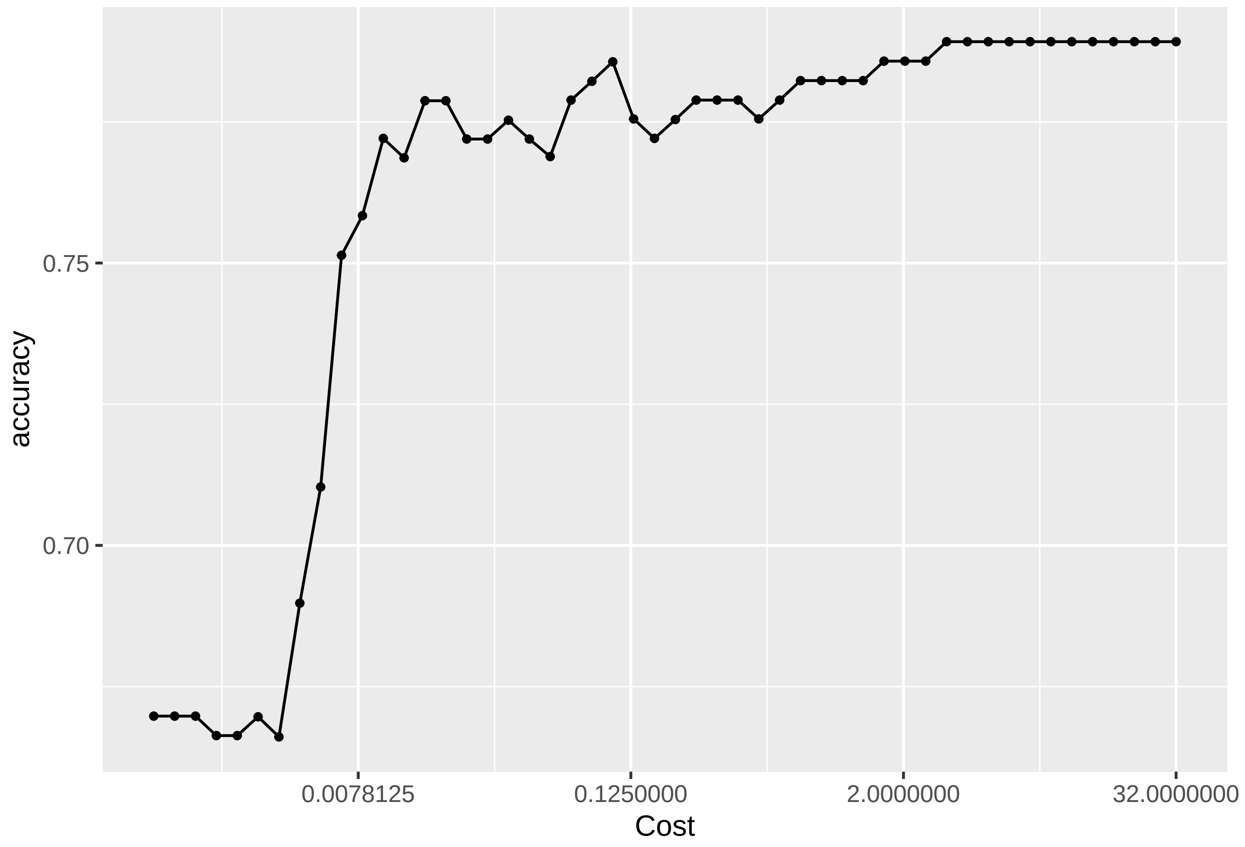
```
##
##  Mcnemar's Test P-Value : 0.09529
##
##             Sensitivity : 0.5152
##             Specificity : 0.8923
##          Pos Pred Value : 0.7083
##          Neg Pred Value : 0.7838
##              Prevalence : 0.3367
##          Detection Rate : 0.1735
##    Detection Prevalence : 0.2449
##       Balanced Accuracy : 0.7037
##
##        'Positive' Class : pos
##
```

```
confusionMatrix(data = pred.svmr,
                reference = testing_data$diabetes)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction pos neg
##        pos  14   8
##        neg  19  57
##
##                Accuracy : 0.7245
##                  95% CI : (0.625, 0.8099)
##     No Information Rate : 0.6633
##     P-Value [Acc > NIR] : 0.11883
##
##                   Kappa : 0.3281
##
##  Mcnemar's Test P-Value : 0.05429
##
##             Sensitivity : 0.4242
##             Specificity : 0.8769
##          Pos Pred Value : 0.6364
##          Neg Pred Value : 0.7500
##              Prevalence : 0.3367
##          Detection Rate : 0.1429
##    Detection Prevalence : 0.2245
##       Balanced Accuracy : 0.6506
##
##        'Positive' Class : pos
##
```

Using **tidymodels**

```
set.seed(1)
cv_folds <- vfold_cv(training_data)

# Model specification
svm_linear_spec <- svm_linear(cost = tune()) %>%
  set_engine("kernlab") %>%
  set_mode("classification")

# svm_linear_spec %>% extract_parameter_dials("cost")

# Create a grid of tuning parameters
svm_linear_grid_set <- parameters(cost(range = c(-10, 5)))
svm_linear_grid <- grid_regular(svm_linear_grid_set, levels = 50)

# Create a workflow
svm_linear_workflow <- workflow() %>%
  add_model(svm_linear_spec) %>%
  add_formula(diabetes ~ .)

# Tune the model
svm_linear_tune <- tune_grid(
  svm_linear_workflow,
  resamples = cv_folds,
  grid = svm_linear_grid)

# Plot the results
autoplot(svm_linear_tune, metric = "accuracy")
```



```
svm_linear_best <- select_best(svm_linear_tune, metric = "accuracy")

final_svm_linear_spec <- svm_linear_spec %>%
  update(cost = svm_linear_best$cost)
```

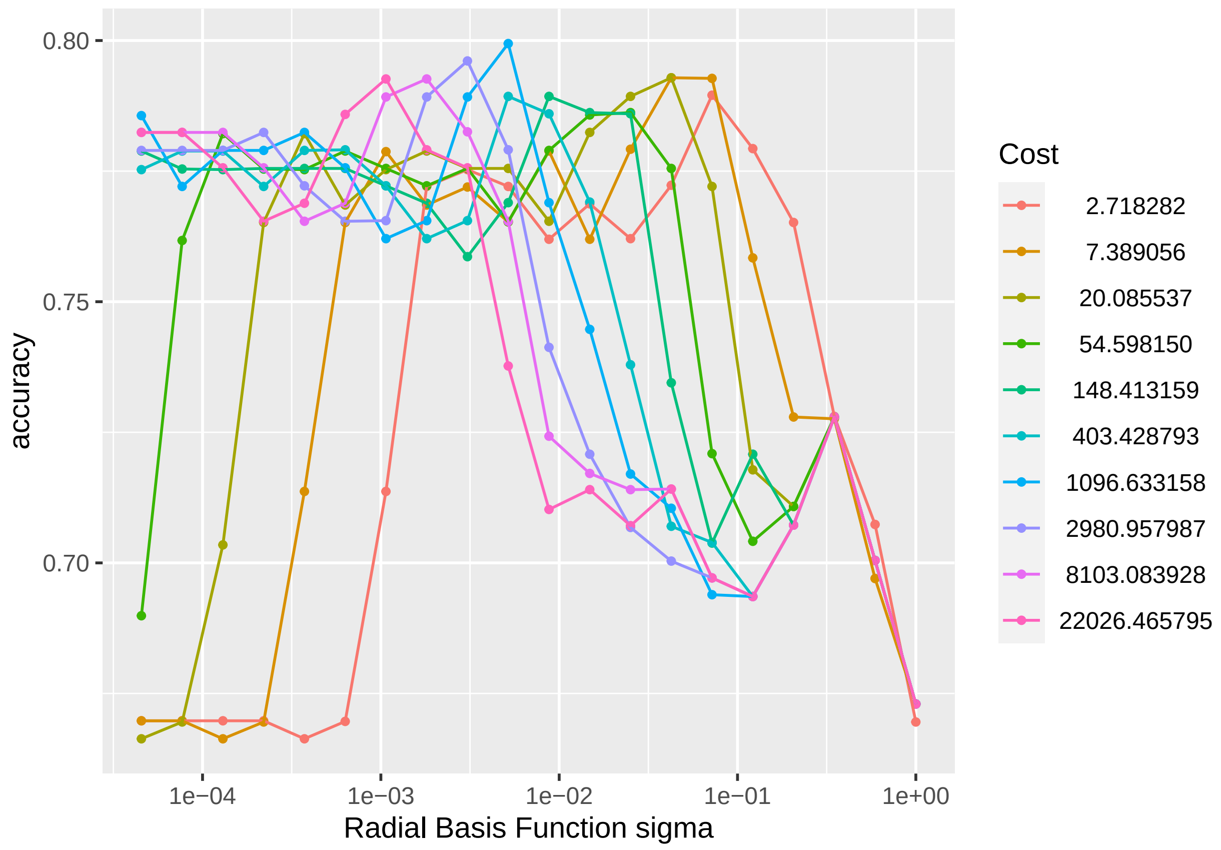
```r
svm_linear_fit <- fit(final_svm_linear_spec, formula = diabetes ~ ., data = training_data)
```

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

```r
svm_linear_model <- extract_fit_engine(svm_linear_fit)

head(predict(svm_linear_fit, new_data = testing_data))
```

```
## # A tibble: 6 x 1
##   .pred_class
##   <fct>
## 1 neg
## 2 pos
## 3 neg
## 4 neg
## 5 pos
## 6 neg
```

```r
# Model specification
svmrbf_spec <- svm_rbf(cost = tune(), rbf_sigma = tune()) %>%
  set_engine("kernlab") %>%
  set_mode("classification")

# svmrbf_spec %>% extract_parameter_dials("cost")
# svmrbf_spec %>% extract_parameter_dials("rbf_sigma")

# Create a grid of tuning parameters
svmrbf_grid_set <- parameters(cost(range = c(1, 10), trans = log_trans()),
                              rbf_sigma(range = c(-10, 0), trans = log_trans()))
svmrbf_grid <- grid_regular(svmrbf_grid_set, levels = c(10, 20))

# Create a workflow
svmrbf_workflow <- workflow() %>%
  add_model(svmrbf_spec) %>%
  add_formula(diabetes ~ .)

# Tune the model
svmrbf_tune <- tune_grid(
  svmrbf_workflow,
  resamples = cv_folds,
  grid = svmrbf_grid)

# Plot the results
autoplot(svmrbf_tune, metric = "accuracy")
```



```
svmrbf_best <- select_best(svmrbf_tune, metric = "accuracy")

final_svmrbf_spec <- svmrbf_spec %>%
  update(cost = svmrbf_best$cost, rbf_sigma = svmrbf_best$rbf_sigma)

svmrbf_fit <- fit(final_svmrbf_spec, formula = diabetes ~ ., data = training_data)

svmrbf_model <- extract_fit_engine(svmrbf_fit)

head(predict(svmrbf_fit, new_data = testing_data))
```

```
## # A tibble: 6 x 1
##   .pred_class
##   <fct>
## 1 neg
## 2 pos
## 3 neg
## 4 neg
## 5 neg
## 6 neg
```

```
model_compare <- workflow_set(preproc = list(diabetes ~ .),
                              models = list(svm_linear = final_svm_linear_spec,
                                            svm_rbf = final_svmrbf_spec)) %>%
  workflow_map(resamples = cv_folds)

model_compare %>% collect_metrics() %>% filter(.metric == "accuracy")
```

```
## # A tibble: 2 x 9
##   wflow_id          .config preproc model .metric .estimator  mean     n std_err
##   <chr>             <chr>   <chr>   <chr> <chr>   <chr>      <dbl> <int>   <dbl>
## 1 formula_svm_line~ Prepro~ formula svm_~ accura~ binary     0.789    10  0.0141
## 2 formula_svm_rbf   Prepro~ formula svm_~ accura~ binary     0.799    10  0.0209
```