# Hyperparameter Tuning in SVM

STSM2634

2025-05-09

In Support Vector Machine (SVM) modelling in R, the following attrributes are provided:

| | |
|---|---|
| model$kernel | # kernel used |
| model$cost | # C parameter |
| model$gamma | # gamma value |
| model$fitted | # predicted values |
| model$SV | # support vectors |
| model$index | # indices of support vectors |
| model$tot.nSV | # total number of support vectors |

You can tune the parameters in the SVM model as follows to get the best model:

```
# Load library
library(tidyverse)
library(e1071)

# Load and split the data
data("mtcars")

set.seed(123)
sample_index <- sample(1:nrow(mtcars), size = 0.7 * nrow(mtcars))
train_data <- mtcars[sample_index, ]
test_data <- mtcars[-sample_index, ]

# Grid search for best epsilon and cost
tune_result <- tune(
  svm,
  mpg ~ .,  # formula
  data = train_data,
  ranges = list(
```

```r
    epsilon = seq(0, 1, 0.1),
    cost = c(0.1, 1, 10, 100)
  ),
  type = "eps-regression",
  kernel = "radial"
)

# Print the best model and parameters
best_model <- tune_result$best.model
summary(best_model)

##
## Call:
## best.tune(METHOD = svm, train.x = mpg ~ ., data = train_data, ranges =
list(epsilon = seq(0,
##     1, 0.1), cost = c(0.1, 1, 10, 100)), type = "eps-regression",
##     kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  radial
##        cost:  1
##       gamma:  0.1
##     epsilon:  0
##
##
## Number of Support Vectors:  22

# Predict on test data
predictions <- predict(best_model, newdata = test_data)

# Evaluate performance
rmse <- sqrt(mean((predictions - test_data$mpg)^2))
cat("RMSE on test data:", round(rmse, 2), "\n")

## RMSE on test data: 2.31

# Check number of support vectors
num_sv <- length(best_model$index)
n_train <- nrow(train_data)
sv_ratio <- num_sv / n_train
cat("Number of support vectors:", num_sv, "\n")

## Number of support vectors: 22

cat("Support vector ratio:", round(sv_ratio, 2), "\n")

## Support vector ratio: 1

cat("Within 20%-80% range:", sv_ratio >= 0.2 & sv_ratio <= 0.8, "\n")
```

```
## Within 20%-80% range: FALSE
```

**Note**: The best model according to your data may not be the universal best model.

Also, you can extract both the training error and feature scaling details from a fitted SVM model.

## Training Error

### A. For Classification

The training error is the proportion of misclassified observations in the training set.

```r
data(iris)

# Fit SVM
model <- svm(Species ~ ., data = iris, kernel = "radial")

# Get predicted classes
pred <- predict(model, iris)

# Calculate training error
training_error <- mean(pred != iris$Species)
print(training_error)

## [1] 0.02666667
```

This gives you the fraction of incorrect predictions, i.e., the training error rate.

### B. For Regression

**Note:** This regression is not linear regression. Rather the regression is done by the support vector machine model, called the support vector regression (SVR).

```r
model_reg <- svm(mpg ~ ., data = mtcars, type = "eps-regression")
pred_reg <- predict(model_reg, mtcars)

# Example: Root Mean Squared Error (RMSE)
rmse <- sqrt(mean((pred_reg - mtcars$mpg)^2))
print(rmse)

## [1] 2.1247
```

## Feature Scaling

By default, svm() automatically scales numeric variables, unless you specify scale = FALSE.

You can retrieve scaling details like this:

```
model$x.scale  # for features
```

```
## $`scaled:center`
## Sepal.Length  Sepal.Width Petal.Length  Petal.Width 
##     5.843333     3.057333     3.758000     1.199333 
## 
## $`scaled:scale`
## Sepal.Length  Sepal.Width Petal.Length  Petal.Width 
##    0.8280661    0.4358663    1.7652982    0.7622377
```

```
model$y.scale  # for response
```

```
## NULL
```