Kernel-Based Learning & Multivariate Modeling

Ricard Meyerhofer Parra

15/10/2019

Problem 1: The SVM for regression in action

Use the ksvm method to perform SVM regression on some data set of your choice (as typical examples, use the Concrete Compressive Strength data set or the Yacht Hydrodynamics data set).

If you wish, you can also use standard multivariate regression methods, like (ridge) regression (lm.ridge in R); you can also use kernel (ridge) regression for a comparison against linear methods.

The SVM regression is found in the ksvm method (be sure to specify eps-svr) for the 'type' parameter). Draw conclusions on your results.

. . . .

In this particular case we are going to use the Concrete Compressive Strenght dataset as suggested by the exercise. This dataset contains 1030 instances with 9 attributes which we can see in the following table.

Variable name	Data Type	Measurement	Description
Cement	quantitative	kg in a m3 mixture	Input Variable
Blast Furnace Slag	quantitative	kg in a m3 mixture	Input Variable
Fly Ash	quantitative	kg in a m3 mixture	Input Variable
Water	quantitative	kg in a m3 mixture	Input Variable
Superplasticizer	quantitative	kg in a m3 mixture	Input Variable
Coarse Aggregate	quantitative	kg in a m3 mixture	Input Variable
Fine Aggregate	quantitative	kg in a m3 mixture	Input Variable
Age	quantitative	Day (1~365)	Input Variable
Concrete compressive strength	quantitative	MPa	Output Variable

All this variables are related to the creation of concrete which is a highly non-linear function of age and ingredients. Therefore, this is a regression problem that we can solve with a SVM. The dataset is complete since it does not contain missing values.

Data preparation

Now we are going to read the data that comes from an excel file so we will have to do some transformations to it:

```r
#Read the data:
concrete <- read_excel("Concrete_Data.xls")
#Renaming:
names(concrete) <- c("cement", "blast_furnace",
                 "fly_ash", "water",
                 "superplasticizer",
                 "coarse_aggregate",
                 "fine_aggregate",
                 "age",
                 "concrete_strength")
str(concrete)
```

We are going to scale the values prior to do anything and we are going to divide our dataset in two sets: Training and Test with a 2/3, 1/3 respectively.

```
concrete_strength <- as.data.frame(lapply(concrete, scale))
#Separating the outcome from the predictors:
concrete_strength <- concrete$concrete_strength
concrete <- select(concrete, -concrete_strength)

idx <- createDataPartition(concrete_strength, p = 2/3, list = FALSE)

train_set <- concrete[idx, ]
train_outcome <- concrete_strength[idx]

test_set <- concrete[-idx, ]
test_outcome <- concrete_strength[-idx]
```

Modeling

For hyper-parameter selection we are going to use cross validation and we are going to try how different models perform one against each other:

Linear Regression

We obtain a 60% of precision in the linear regression which is a linear model so it explains us why it has so few accuracy.

```
#Linear Regression
set.seed(3)
lm_tune <- train(x = train_set,
                 y = train_outcome,
                 method = "lm",
                 trControl = ctrl)

test_result <- data.frame(observations = test_outcome, lm_predictions = predict(lm_tune, test_set))
postResample(pred = test_result$lm_predictions, obs = test_result$observations)
```

Regression trees

We obtain a 51.89% of precission with regression trees.

```
#Rpart
set.seed(3)
tree_tune <- train(x = train_set, y = train_outcome,
                   method = "rpart",
                   trControl = ctrl,
                   tuneLength = 5)
test_result$tree_predictions <- predict(tree_tune, test_set)
postResample(pred = test_result$tree_predictions, obs = test_result$observations)
```

Multivariate adaptive regression spline method

We obtain an 82.28% of precision with the Multivariate adaptive regression spline method.

```
mars_tune <- train(x = train_set, y = train_outcome,
                   method = "earth",
```

```r
                   trControl = ctrl,
                   tuneLength = 20)
test_result$mars_predictions <- predict(mars_tune, test_set)
postResample(pred = test_result$mars_predictions, obs = test_result$observations)
```

Support Vector Machines

We have chosen an epsilon of 1.2 and a C of 20000 after looking which of the values was the one that suited best our training data. We have a 98% of precision in train but only a 76.04% in test. Is it possible that with other parameters we could exceed the MARS performance but is still quite close one to each other.

```r
#SVM
ksvm_tune <- ksvm(train_outcome ~ ., data = train_set, scaled=c(),
                  kernel="rbfdot",
                  epsilon= 1.2,
                  C=20000,
                  prob.model=TRUE)

#to see best hypeparameter:
ksvm_predictions_train <- predict(ksvm_tune, train_set)
postResample(pred = ksvm_predictions_train, obs = train_outcome)


test_result$ksvm_predictions <- predict(ksvm_tune, test_set)
postResample(pred = test_result$ksvm_predictions, obs = test_result$observations)
```