Weekly exercises week 11: Support Vector Machines

Marjolein Fokkema

Support Vector Regression

In this exercise, we analyze the `Boston` dataset, and predict median house values (`medv`) using all other predictors in the dataset.

Load the data as follows:

```r
library("MASS")
data(Boston)
```

Separate into training and test set:

```r
set.seed(42)
test <- sample(1:nrow(Boston), size = 100)
train <- which(!(1:nrow(Boston)) %in% test)
```

Use package `e1071` to fit a Support Vector Regression, use function `tune()` to tune the SVM parameters, use function `svm()` to fit the SVM.

When a numeric response variable is supplied to the `formula` argument of function `svm`, an SVM of type `eps-regression` will be fit. This requires specification of an additional parameter: `epsilon`. This naming might be a bit confusing, because the lecture slides used ϵ to denote slack variables, but here it refers to a different parameter.

The value of ϵ defines a margin of tolerance where no penalty is given to errors. Note that the support vectors are the instances within the margin, where the slack variables are non-zero. The larger ϵ, the larger the errors we admit in our solution. By contrast, if ϵ gets close to 0, every error is penalized and we end up with a very wide margin (i.e., many support vectors, tending to the total number of observations). With a value of zero, all observations will contribute to all predicted values, yielding highest bias and lowest variance.

Tuning SVMs

I find tuning the parameters of SVMs more tricky than for other methods. The optimal value of the cost parameter of SVMs, like the k parameter of kNN, is dependent on the number of observations. The C parameter for most SVM implementations scales approximately linearly with the number of training patterns. This can make it more involved to determine a good range of values to tune the parameter. Where for many parameters (e.g., λ in ridge, lasso, elastic net; the learning rate in boosting, the number of trees in tree ensembles) the optimal value generally will not depend on sample size, so one can often rely on a default range for the tuning grid.

To tune the parameters of SVMs, one can start from an initial grid, check if a more-or-less optimal value has been obtained (i.e., where lower or higher values would yield higher prediction error). Next, adjust the grid if necessary by making it wider and/or more fine grained and re-perform the cross validation routine. In fact, such adjustment to the initial grid may be beneficial for tuning the parameters of other prediction methods, too.

a) SVR with linear kernel

Fit an SVR with linear kernel and optimal values for the cost parameter to the training observations.

First specify an initial grid of possible parameter values, e.g.:

```r
library("e1071")
cost <- c(.001, .01, .1, 1, 10)
epsilon <-  c(.001, .01, .1, 1)
set.seed(42)
tune.out <- tune(svm, medv ~ ., data = Boston[train, ], kernel = "linear",
                 ranges = list(cost = cost, epsilon = epsilon))
```

Inspect the CV result by plotting, and printing the `$performances` slot of the resulting object. Evaluate the CV results: Did you obtain a convex curve? Was the resolution of the grid fine enough?

Note that for tuning grids with multiple parameters, the default plots provided by `plot.tune` are difficult to read well, so one can create a custom plot, e.g.:

```r
library("ggplot2")
perf <- tune.out$performances
perf$cost <- factor(perf$cost)
ggplot(perf) + geom_line(aes(epsilon, error, group=cost, color=cost)) +
  scale_x_log10()
```

```r
tune.out$best.parameters
```

```r
cost <- c(.005, .01, .025, .05, .1, .5, 1, 10)
epsilon <- c(.001, .005, .0075, .01, .025, .05, .1, .5, .75, .9, 1)
tune.out <- tune(svm, medv ~ ., data = Boston[train, ], kernel = "linear",
                 ranges = list(cost = cost, epsilon = epsilon))
tune.out$best.parameters
perf <- tune.out$performances
perf$cost <- factor(perf$cost)
ggplot(perf) + geom_line(aes(epsilon, error, group=cost, color=cost)) +
  scale_x_log10() + scale_y_log10()
```

Fit an SVR with linear kernel and optimal parameter values determined by CV.

Evaluate predictive accuracy in terms of MSE and MAE on the test observations.

b) SVR with radial basis kernel

Find the optimal values of `gamma`, `cost` and `epsilon` for an SVR with radial basis kernel. You can, for example, start your search with an initial grid with values `c(.001, .01, .1,  1)` for each parameter. Inspect the results and make the grid wider or more fine-grained in a next iteration.

Fit a radial basis SVR with the optimal parameter settings, and compare predictive accuracy in terms of MSE and MAE with the SVR model with linear kernel you fitted earlier.