Modelling and selection

Dominik Klepl

12/4/2019

Now that we kinda understand our data we can start modelling. We'll use two approaches to identidy the best model structure.

1. Forward subset selection to identify the best model by minimizing MSE. We'll split the data into training and testing set using 80:20 ratio.
2. Use AIC OR BIC to identify the best model.

Hopefully both methods will converge on the same model structure.

Of course, we need to load the data first.

```
data = read.csv("data/x_y.csv", header = F)
colnames(data) = c("x", "y")
```

Forward subset selection

First, create matrix X with all possible predictors, i.e. intercept and x^1 to x^5

```
## Is X matrix?

## [1] TRUE

## Show first 5 rows of X

##      intercept          x1          x2            x3           x4
x5
## [1,]         1 -0.18533753 0.034350000 -0.0063663442 1.179923e-03 -
2.186839e-04
## [2,]         1 -1.06786788 1.140341806 -1.2177343852 1.300379e+00 -
1.388633e+00
## [3,]         1  0.10666430 0.011377272  0.0012135487 1.294423e-04
1.380687e-05
## [4,]         1  1.81829611 3.306200727  6.0116519064 1.093096e+01
1.987573e+01
## [5,]         1  0.07756119 0.006015738  0.0004665877 3.618910e-05
2.806869e-06
```

Now we split the data into training (80%) and testing sets

###Forward selection works as follows: fit models using all columns of X separately (fit 6 models) calculate MSE of all models find model with min(MSE) append that model predictor to final model formula and remove that predictor from X

Fit models using selected predictor + all columns of X individually (again 6 models) repeat the other steps

Do this until model has 3 terms

Model fitting function

Because we'll need to fit multiple models, generate predictions and calculate MSE it might be good to have all of this wrapped in one nice function.

```
fit_evaluate = function(X_tr, y_tr=train_y, X_ts, y_ts = test_y){
  #estimate parameters
  theta = solve(crossprod(X_tr), crossprod(X_tr, y_tr))
  #predict test data
  predictions = X_ts %*% theta
  #calculate error
  MSE = mean((y_ts - predictions)^2)

  return(MSE)
}
```

Forward selection

General steps in the algorithm: Run for loop through all columns of X matrix, fitting model and calculating MSE (fitting 6 models) Select model with lowest MSE (on testing data) (hint: function which.min()) Append that predictor to final model and remove it from X (e.g. X[,-selected_predictor]) Repeat...it's quite simple really

```
## Running first round of selection

## Best model with 1 parameter: y ~ x4 + error (MSE= 5.232054 )

##
## Running second round of selection

##
## Best model with 2 parameters: y ~ x4 + x2 + error (MSE= 0.3079839 )

##
## Running third round of selection

##
## Best model with 3 parameters: y ~ x4 + x2 + x1 + error (MSE= 0.007951127 )
```

According to forward selection the best model is: y ~ b1x + b2x^2 + b3x^4

Information criterion selection

Adjust fitting function to return AIC instead of MSE

Also now we use the full dataset as we don't need to compute out-of-sample metrics

```
fit_evaluate = function(X, Y){
  X = as.matrix(X)
  Y = as.matrix(Y)
  #estimate parameters
  theta = solve(crossprod(X), crossprod(X, Y))
  #predict test data
  predictions = X %*% theta
  #calculate error
  residuals = Y - predictions
  sigma_sq = sum(residuals^2)/(nrow(Y) - 1)
  loglik= sum(log(dnorm(data$y, mean = predictions, sd = sqrt(sigma_sq))))

  k = ncol(X)
  AIC = 2*k - 2*loglik

  return(AIC)
}
```

Now we construct a vector with all combinations of [1:3] predictors

```
parameters = 1:6
candidates_1 = t(as.matrix(parameters))
candidates_2 = combn(parameters, m = 2)
candidates_3 = combn(parameters, m = 3)
```

Now we can run a for loop through all allowed parameter combinations

```
## According to AIC the best model is:
##  y ~ x1 + x2 + x4
## (AIC = -432.3259 )
```

Just for fun, let's use BIC as well

```
## According to BIC the best model is:
##  y ~ x1 + x2 + x4
## (BIC = -421.7616 )
```

In order, to keep this notebook short (and easy to navigate) we'll explore the best model in next notebook.