STAT230 HW 10
University of California, Berkeley

Thibault Doutre, Student ID 26980469

April 14, 2016

1

```
# Load data ---------------------------------------------

load("HW10.rda")

## shuffle
data = data[sample(nrow(data)),]

# Full Model, R2 ----------------------------------------

## # OLS fit
lm.fit = lm(Y~., data=data)

## # R2
R2 = var(lm.fit$fitted.values)/var(data$Y)
R2
```

```
## [1] 0.5011678
```

```
## # Cross validated R2

R2_cv = function(data, nfold, formula="Y~."){
  # Compute R2 based on cross validated data
  Y_cv = c()
  nrows = nrow(data)
```

```
  for (i in seq(0, nrows-nfold, by=nrows/nfold)){
    print(i+1)
    print(nfold+i)
    test = i:(nfold+i)
    train = -test
    lm.fit = lm(formula, data=data[train,])
    Ytest = predict(lm.fit, data[test,])
    Y_cv = c(Y_cv, Ytest)
  }
  var(Y_cv)/var(data$Y)
}
names(lm.fit)
```

```
##  [1] "coefficients"  "residuals"     "effects"
##  [4] "rank"          "fitted.values" "assign"
##  [7] "qr"            "df.residual"   "xlevels"
## [10] "call"          "terms"         "model"
```

```
nfold = 10
R2_cv10 = R2_cv(data, 10)
```

```
## [1] 1
## [1] 10
## [1] 11
## [1] 20
## [1] 21
## [1] 30
## [1] 31
## [1] 40
## [1] 41
## [1] 50
## [1] 51
## [1] 60
## [1] 61
## [1] 70
## [1] 71
## [1] 80
## [1] 81
```

```
## [1] 90
## [1] 91
## [1] 100

R2_cv10

## [1] 0.591366

summary(lm.fit)

##
## Call:
## lm(formula = Y ~ ., data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.2462 -0.7195  0.0226  0.5911  3.3454
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.44742    0.12700   3.523 0.000713
   ***
## f            0.31565    0.12122   2.604 0.011008
   *
## d            0.45784    0.13563   3.376 0.001145
   **
## t            0.22951    0.11538   1.989 0.050135
   .
## a            0.36626    0.14114   2.595 0.011272
   *
## i            0.07495    0.14521   0.516 0.607209
## n            0.09645    0.12999   0.742 0.460307
## l            0.21065    0.12259   1.718 0.089666
   .
## q            0.07316    0.12947   0.565 0.573640
## e            0.49904    0.13516   3.692 0.000407
   ***
## h            0.33018    0.14225   2.321 0.022860
   *
```

```
## b             0.34986    0.12641   2.768 0.007030
   **
## g             0.22757    0.11859   1.919 0.058596
   .
## c             0.30609    0.13616   2.248 0.027362
   *
## p             0.05387    0.12778   0.422 0.674470
## r             0.19486    0.15407   1.265 0.209677
## o             0.01688    0.11065   0.153 0.879164
## m             0.06620    0.13204   0.501 0.617531
## s             0.11640    0.14126   0.824 0.412407
## j             0.09100    0.11559   0.787 0.433484
## k             0.27364    0.12343   2.217 0.029504
   *
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.109 on 79 degrees of
   freedom
## Multiple R-squared:  0.5012, Adjusted R-squared:
    0.3749
## F-statistic: 3.968 on 20 and 79 DF,  p-value:
   5.525e-06
```

The cross validated R^2 is significantly higher than both the multiple and adjusted R^2. The R^2 computed with the fitted values is approximately equal to the multiple R^2 (precision 10^{-4}).

2

```
# Backward selection ------------------------------------

## # Cross validated MSE

MSE_cv = function(data, nfold, formula="Y~."){
  # Compute R2 based on cross validated data
```

```
  Y_cv = c()
  nrows = nrow(data)
  MSE_test = c()
  MSE_train = c()
  for (i in seq(0, nrows-nfold, by=nrows/nfold)){
    test = (i+1):(nfold+i)
    train = -test
    lm.fit = lm(formula, data=data[train,])
    Ytest = predict(lm.fit, data[test,])
    MSE_test = c(MSE_test, mean((data$Y[test]-Ytest)^2))
    MSE_train = c(MSE_train,
                  mean((data$Y[train]-lm.fit$fitted.values)^2))
  }
  # Training error based on best model
  # Test error based on cross validation (mean)
  return(list(test = mean(MSE_test), train = min(MSE_train)))

}

MSE_cv(data, nfold)
```

```
## $test
## [1] 1.621213
##
## $train
## [1] 0.8486754
```

```
## # Remove less significant feature from lm.fit

backward_lm = function(data){
  # Initialize with OLS
  formula = "Y~."
  lm.fit = lm(formula, data=data)
  # Initialize outputs
  MSE_train = c()
  MSE_test = c()
  next_to_remove = ""
  variables = c()
  while(length(names(lm.fit$model))>1){
```

```
    MSE = MSE_cv(data, 10, formula)
    MSE_train = c(MSE_train, MSE$train)
    MSE_test = c(MSE_test, MSE$test)
    t_values = coef(summary(lm.fit))[, "t value"]
    # Variable with smallest t-value
    next_to_remove = names(which.min(t_values))
    # Store removed variables in the order
    variables = c(variables,next_to_remove)
    # Update formula
    formula = paste(formula,"-",next_to_remove,sep="")
    # Update model using new formula
    lm.fit = update(lm.fit, formula)
  }
  # Intercept only
  MSE = MSE_cv(data, 10, formula)
  MSE_train = c(MSE_train, MSE$train)
  MSE_test = c(MSE_test, MSE$test)
  return(list(variables = variables, MSE_test = MSE_test,
              MSE_train = MSE_train))
}

backward = backward_lm(data)
backward
```

```
## $variables
##  [1] "o" "p" "m" "q" "n" "j" "i" "s" "r" "l" "k"
   "t" "f" "b"
## [15] "a" "c" "g" "e" "h" "d"
##
## $MSE_test
##  [1] 1.621213 1.570532 1.566042 1.497060 1.454698
    1.424109
##  [7] 1.407780 1.391831 1.377380 1.366292 1.381277
    1.409561
## [13] 1.450584 1.497906 1.534049 1.547914 1.643368
    1.698750
## [19] 1.757956 1.851872 1.989268
##
## $MSE_train
```

```
##  [1] 0.8486754 0.8515116 0.8515768 0.8557102
   0.8611087
##  [6] 0.8766084 0.8815017 0.8881369 0.9108102
   0.9304373
## [11] 0.9673466 0.9969965 1.0311388 1.0533816
   1.0777200
## [16] 1.1727420 1.2694991 1.4019090 1.4794819
   1.6056939
## [21] 1.7747474
```

3

```
# Forward selection ----------------------------------------

forward_lm = function(data){
  names = names(data)[-1]
  # Initialize outputs
  MSE_train = c()
  MSE_test = c()
  variables = c()
  # Intercept only
  formula = "Y~1"
  MSE = MSE_cv(data, 10, formula)
  MSE_train = c(MSE_train, MSE$train)
  MSE_test = c(MSE_test, MSE$test)
  while(length(names)>0){
    ## Find best variable to add to the model
    best_new = ""
    best_MSE = Inf
    for (variable in names){
      formula_test = paste(formula,"+",variable,sep="")
      # Update MSE using the cross validated training MSE
      MSE = MSE_cv(data, 10, formula_test)$train
      if (MSE<best_MSE){
        best_MSE = MSE
        best_new = variable
```

```
        }
    }
    # Update names
    names = names[-which(names==best_new)]
    ## Update formula
    formula = paste(formula,"+",best_new,sep="")
    ## Update data
    MSE = MSE_cv(data, 10, formula)
    MSE_train = c(MSE_train, MSE$train)
    MSE_test = c(MSE_test, MSE$test)
    variables = c(variables,best_new)
  }
  return(list(variables = variables, MSE_test = MSE_test,
              MSE_train = MSE_train))
}

forward = forward_lm(data)
forward

## $variables
##  [1] "a" "d" "e" "h" "g" "c" "l" "t" "m" "b" "k"
   "f" "r" "n"
## [15] "s" "j" "q" "o" "i" "p"
##
## $MSE_test
##  [1] 1.989268 1.833424 1.733320 1.661363 1.627916
    1.606918
##  [7] 1.534049 1.522413 1.498745 1.538113 1.519357
    1.487420
## [13] 1.422508 1.430995 1.474486 1.501236 1.513964
    1.564495
## [19] 1.621370 1.616839 1.621213
##
## $MSE_train
##  [1] 1.7747474 1.5050471 1.3857133 1.3150821
   1.2452134
##  [6] 1.1591866 1.0777200 1.0392264 1.0153155
   0.9914651
## [11] 0.9665543 0.9349026 0.9084698 0.8916561
```

```
   0.8791435
## [16] 0.8679010 0.8590601 0.8536233 0.8510003
   0.8487694
## [21] 0.8486754
```

Forward selection and backward selection do not give the same sequence of models here. This is often the case when the number of features is relatively high but the same sequence of models can happen in some cases. Indeed, the significance of one feature can depend on the presence or absence of this feature in the model, especially when the varaibles are correlated.

4

```
# Plot results -----------------------------------------------

ymax = max(max(backward$MSE_test,
               backward$MSE_train,
               forward$MSE_train,
               forward$MSE_test))
ymin = min(min(backward$MSE_test,
               backward$MSE_train,
               forward$MSE_train,
               forward$MSE_test))

## Backward selection

plot(rev(backward$MSE_test), col = "darkblue", type="b",
     xlab = "Number of features",
     ylab = "MSE",
     main = "Forward and Backward selection",
     ylim = c(ymin,ymax))

lines(rev(backward$MSE_train), col = "lightblue", type="b")

abline(v=which.min(rev(backward$MSE_test)),col="darkblue",
       lty=2)
abline(v=which.min(rev(backward$MSE_train)),col="lightblue",
```

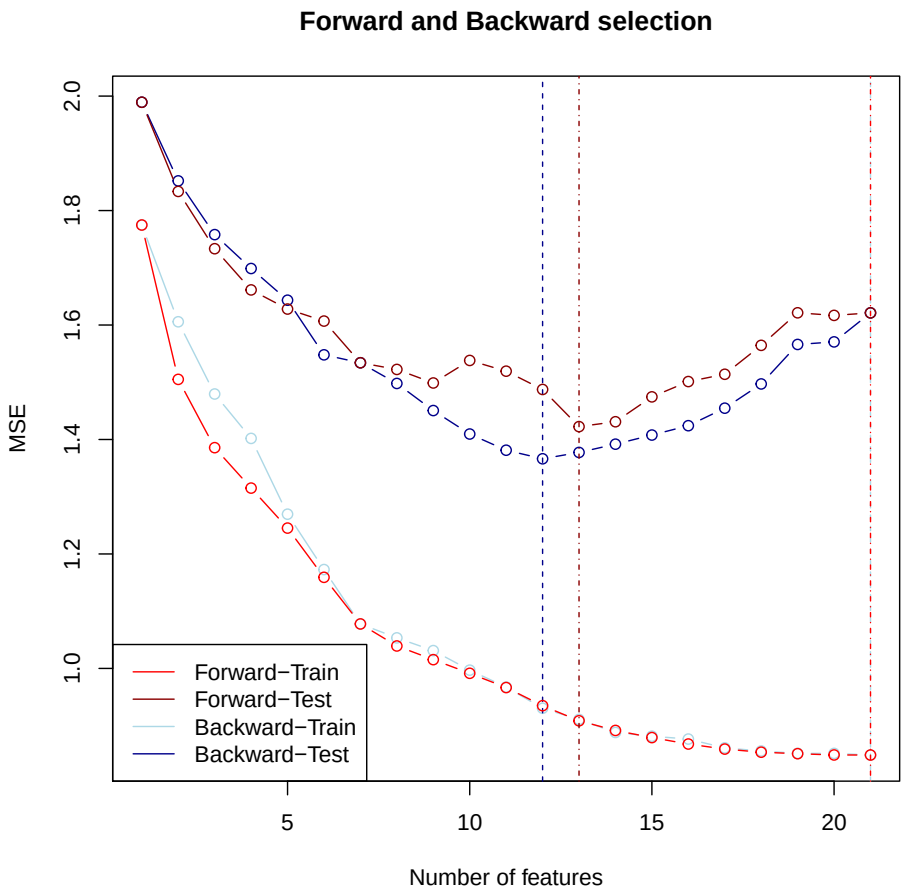
```
       lty=2)


## Forward selection

lines(forward$MSE_train, col = "red1", type="b")
lines(forward$MSE_test, col = "darkred", type="b")

abline(v=which.min(forward$MSE_test),col="darkred",lty=4)
abline(v=which.min(forward$MSE_train),col="red1",lty=4)

## Legend
legend("bottomleft",
       c("Forward-Train","Forward-Test","Backward-Train",
         "Backward-Test"),
       col = c("red1","darkred","lightblue","darkblue"),lwd=1)
```



The train error is decreasing as the number of features grows because we allow more flexibility in the model. As for the test set, there is a minimum corresponding to the optimal trade off between how flexibile the model is and how many noise is added to it with useless features. We talk about overfitting when the model is too flexible.