STAT452/652 Solution to Assignment 1 - Part 1

Due on Oct 16, 2021

1 Concepts

1.1 Question 1

If $X_2 = c$ then the slope of X_1 is $\beta_1 + \beta_3/c$. If $X_1 = d$ then the slope of X_2 is $\beta_2 - \beta_3 d/X_2(1 + X_2)$.

1.2 Question 2

The formula for the regression model being fit is $f(X, Z) = \beta_0 + \beta_1 X + \sum_{q=2}^{Q} \beta_q z_q + \sum_{q=2}^{Q} \beta_{1q} X z_q$. (This is the general expression. Plug in $Q = 2$ when Z is categorical (factor) with two levels.)

2 Applications A

2.1 Question 1

The following table contains minima, maxima and means for our two constructed variables.

```
##      TWcp TWrat
## Min   216  3.03
## Max  1490 40.90
## Mean  757  9.42
```

2.2 Question 2

```
##     t value Pr(>|t|)
## CP    -3.99 0.000123
## Rat    4.01 0.000115
```

Both variables appear to be quite useful for predicting ozone. The cross-product looks a bit better (its p-value is half that of the ratio), but both effects are so strong that it's hard to say one the clear winner.

Slopes for temperature when wind speed is at its minimum and maximum values are given in the following display.

```
### Get coefficients
pars = fit.cp$coef
coef.temp = pars["Temp"]
coef.int = pars["TWcp"]

### Get wind values
wind.min = min(AQ$Wind)
wind.max = max(AQ$Wind)
```

```
### Compute slopes
slope.min = coef.temp + coef.int * wind.min
slope.max = coef.temp + coef.int * wind.max
slopes = c(slope.min, slope.max)
names(slopes) = c("Min", "Max")
print(signif(slopes, 3))
```

```
##    Min    Max
##  3.500 -0.497
```

2.3 Question 3

Validation set MSPEs for our two models are:

```
### Create function to compute MSPEs
get.MSPE = function(Y, Y.hat){
  return(mean((Y - Y.hat)^2))
}
```

```
set.seed(4099183)

n = nrow(AQ) # Sample size
reorder = sample.int(n) # Randomized order

### Identify which observations go in the training set (set == 1) or the
### validation set (set == 2)
prop.train = 0.75 # Proportion of observations to put in the training set
set = ifelse(test = (reorder < prop.train * n),
  yes = 1,
  no = 2)

### Construct training and validation sets
data.train = AQ[set == 1, ]
data.valid = AQ[set == 2, ]
Y.valid = data.valid$Ozone

### Fit both models to the training set
train.cp = lm(Ozone ~ . - TWrat, data = data.train)
train.rat = lm(Ozone ~ . - TWcp, data = data.train)

### Get predictions
pred.cp = predict(train.cp, data.valid)
pred.rat = predict(train.rat, data.valid)

### Get MSPEs
MSPEs = rep(0, times=2)
names(MSPEs) = c("CP", "Ratio")
MSPEs["CP"] = get.MSPE(Y.valid, pred.cp)
MSPEs["Ratio"] = get.MSPE(Y.valid, pred.rat)
print(signif(MSPEs, 3))
```

```
##    CP Ratio
##   199   218
```

We see that the cross-product model performs better on this split than the model with a ratio.

2.4 Question 4

```
### Create function which constructs folds for CV
### n is the number of observations, K is the number of folds
get.folds = function(n, K) {
  ### Get the appropriate number of fold labels
  n.fold = ceiling(n / K) # Number of observations per fold (rounded up)
  fold.ids.raw = rep(1:K, times = n.fold)
  fold.ids = fold.ids.raw[1:n]

  ### Shuffle the fold labels
  folds.rand = fold.ids[sample.int(n)]

  return(folds.rand)
}
```

```
K = 10 #Number of folds
M = 20 # Number of times to repeat CV

all.models = c("Solar.R", "Wind", "Temp", "Full", "Second", "CP", "Ratio")

### Create a 3D array to store CV errors by model, then iteration,
### then fold. Set names of models.
all.CV.MSPEs = array(0, dim = c(length(all.models), M, K))
dimnames(all.CV.MSPEs)[[1]] = all.models


for (j in 1:M) {
  ### Get CV fold labels
  folds = get.folds(n, K)

  ### Perform cross-validation
  for (i in 1:K) {
    ### Get training and validation sets
    data.train = AQ.raw[folds != i,]
    data.valid = AQ.raw[folds == i,]
    Y.valid = data.valid$Ozone

    ##########################
    ### Regression Models ###
    ##########################

    ### Fit models
    fit.solar = lm(Ozone ~ Solar.R, data = data.train)
    fit.wind = lm(Ozone ~ Wind, data = data.train)
    fit.temp = lm(Ozone ~ Temp, data = data.train)
    fit.full = lm(Ozone ~ ., data = data.train)
    fit.2 = lm(Ozone ~ . ^ 2 + I(Solar.R^2) + I(Wind ^ 2) + I(Temp ^ 2), data = data.train)
    fit.cp = lm(Ozone ~ . + Temp:Wind, data = data.train)
    fit.rat = lm(Ozone ~ . + I(Temp/Wind), data = data.train)
```

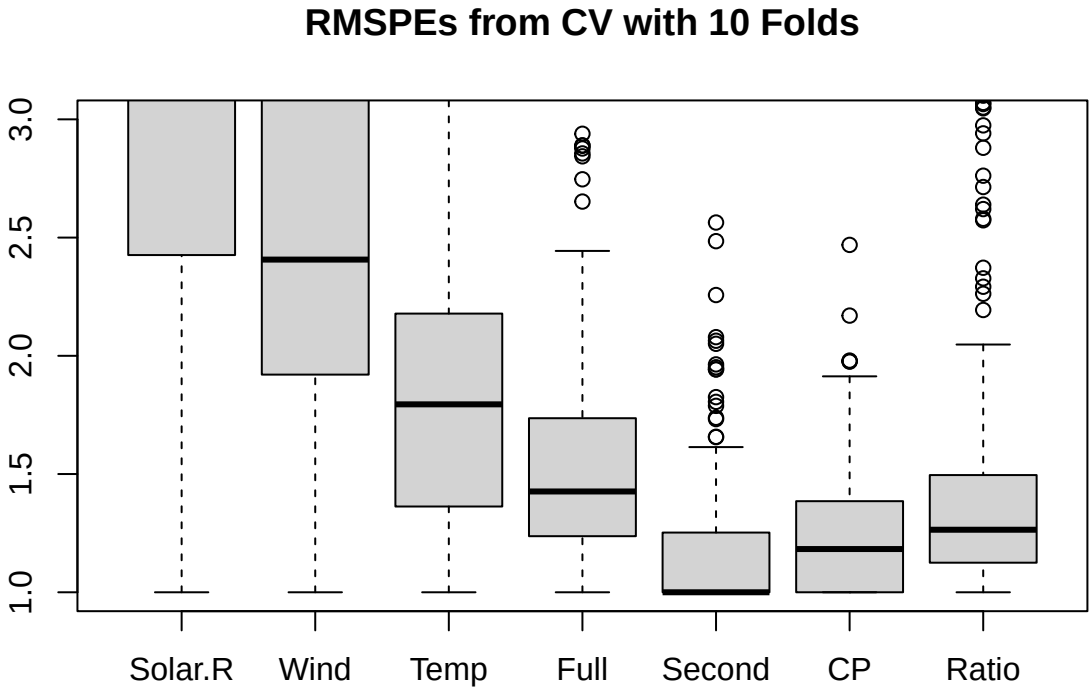
```
    ### Get predictions
    pred.solar = predict(fit.solar, data.valid)
    pred.wind = predict(fit.wind, data.valid)
    pred.temp = predict(fit.temp, data.valid)
    pred.full = predict(fit.full, data.valid)
    pred.2 = predict(fit.2, data.valid)
    pred.cp = predict(fit.cp, data.valid)
    pred.rat = predict(fit.rat, data.valid)

    ### Calculate MSPEs
    ### Be careful with order of indices!!!
    all.CV.MSPEs["Solar.R", j, i] = get.MSPE(Y.valid, pred.solar)
    all.CV.MSPEs["Wind", j, i] = get.MSPE(Y.valid, pred.wind)
    all.CV.MSPEs["Temp", j, i] = get.MSPE(Y.valid, pred.temp)
    all.CV.MSPEs["Full", j, i] = get.MSPE(Y.valid, pred.full)
    all.CV.MSPEs["Second", j, i] = get.MSPE(Y.valid, pred.2)
    all.CV.MSPEs["CP", j, i] = get.MSPE(Y.valid, pred.cp)
    all.CV.MSPEs["Ratio", j, i] = get.MSPE(Y.valid, pred.rat)
  }
}

### Combine all MSPEs for each model into a single list
### Note: as.numeric() converts its input into a 1D vector
data.CV.MSPEs = apply(all.CV.MSPEs, 1, as.numeric)

### Get RMSPEs
data.CV.RMSPEs = apply(data.CV.MSPEs, 1, function(W){
  best = min(W)
  return(W/best)
})
data.CV.RMSPEs = t(data.CV.RMSPEs)

### Make boxplot. Focus on the region near the best models
boxplot(data.CV.RMSPEs, main = paste0("RMSPEs from CV with ", K, " Folds"), ylim = c(1, 3))
```



The new models are competitive, but the best is still the model with all interactions and second-order terms.

3 Applications B

```
### Read-in data
ins = read.csv("Insurance.csv")

### Convert categorical predictors to factors
ins$zone = as.factor(ins$zone)
ins$make = as.factor(ins$make)

### Remove observations with no claims
ins = dplyr::filter(ins, claims > 0)
```

Here is the result of a regression of per on the other variables.

```
fit.ins = lm(per ~ ., data = ins)
coef.table = summary(fit.ins)$coef
print(coef.table)
```

```
##                 Estimate   Std. Error     t value     Pr(>|t|)
## (Intercept)  1.186200e+01 1.321380e-01  89.7697628 0.000000e+00
## km          -3.434143e-01 2.063656e-02 -16.6410669 6.672158e-58
## zone2       -1.375833e-01 9.717096e-02  -1.4158887 1.569832e-01
## zone3       -2.143003e-02 9.753311e-02  -0.2197205 8.261140e-01
## zone4        4.316718e-01 9.691760e-02   4.4540082 8.952730e-06
## zone5       -1.041544e+00 1.043269e-01  -9.9834735 7.112927e-23
## zone6       -4.440303e-01 1.008852e-01  -4.4013426 1.139738e-05
## zone7       -2.862106e+00 1.378180e-01 -20.7672881 6.039755e-86
## bonus        2.300740e-01 1.404526e-02  16.3808983 2.803491e-56
## make2       -1.403389e+00 1.139683e-01 -12.3138518 1.694961e-33
```

```
## make3       -1.709548e+00 1.188711e-01 -14.3815329 1.903245e-44
## make4       -1.834010e+00 1.240120e-01 -14.7889736 9.199222e-47
## make5       -1.316946e+00 1.138403e-01 -11.5683598 6.803423e-30
## make6       -8.253261e-01 1.128673e-01  -7.3123604 3.950590e-13
## make7       -1.716116e+00 1.153488e-01 -14.8776200 2.839815e-47
## make8       -2.069770e+00 1.199166e-01 -17.2600825 7.725085e-62
## make9        1.459262e+00 1.208863e-01  12.0713597 2.642252e-32
## insured     -5.724293e-05 1.150595e-05  -4.9750727 7.151952e-07
## claims       3.029138e-03 3.518818e-04   8.6083950 1.602788e-17
```

```
### Compute fitted values for later
intcpt.baseline = coef.table["(Intercept)", "Estimate"]
intcpt.97 = intcpt.baseline + coef.table["zone7", "Estimate"] +
  coef.table["make9", "Estimate"]
```

The total number of parameters we fit is 19. The regression intercept when make and zone are both at their first levels is 11.9. The intercept when these variables are at their last level is 10.5.