Discussion 11

Greg Kirwin

11/19/2021

Reminders

- Hw4 due November 23 (Next Tuesday)
    - Use caching from last week to your advantage since these models may take time to run
    - Spec 2 seems to be the one that takes a while to run; you can decrease the epochs and loops to make it computationally feasible
- No discussion next week (Thanksgiving break)

Email:

- gjkirwin@wisc.edu

Office Hours:

- Tuesdays and Thursdays at 7pm Central (UTC-6:00) -> Note that the clocks have changed!!
- Fridays at 9am Central
- **Next week (11/23-27) will only have a Tuesday OH**

Today

- Random Forests

Setup

Data come from the mushroom data <!> at UCI. I picked this dataset because it's classification-based. However, it requires a decent amount of data cleaning. If you would like to try using the raw data yourself, I'll include a link in the slides to a GitHub repo where you can do it on your own.

```r
shroom <- fread("mushrooms.csv", header = T)
test <- sample.int(nrow(shroom), size = nrow(shroom) *
    0.2, replace = F)

# will try to predict habitat
ytrain <- as.factor(shroom$habitat[-test])
ytest <- as.factor(shroom$habitat[test])

xtrain <- subset(shroom[-test], select = -c(habitat))
xtest <- subset(shroom[test], select = -c(habitat))

# need to have data as factor, not just char
xtrain <- xtrain %>% mutate_if(is.character, as.factor)
xtest <- xtest %>% mutate_if(is.character, as.factor)
```

some notes

It seems that the `randomForest` function requires some specific data setup to be feasible. It appears we need to have either completely numerical data (as you will for the assignment) or categorical data with particular numbers of classes to use- it showed me warnings when I had less than 5, or gave an error with more than 50.

Estimation

A little example here <!>

```r
forest = randomForest(x = xtrain, y = ytrain,
    xtest = xtest, ytest = ytest, ntree = 1000,
    mtry = 4, importance = T)
forest2 = randomForest(x = xtrain, y = ytrain,
    ntree = 1000, mtry = 4, importance = T)
```

Evaluation

This data is a little different from the airline or simulated data from hw4, because it's all factor. These data are *ordinal*, meaning that the value 0, 1, 2, etc. don't have true meaning. We can't assign "urban" the value 5, for example, because there is no meaningful difference "urban" and "woods" when given a number.

Evaluation contd.

So instead, I come up with a measure of accuracy by just testing to see whether the predicted value is equivalent to the true value, giving it a "1" if correct, and "0" else. The mean of this vector is then my test accuracy. I played around with the `mtry` argument and found that increasing the number (randomly sampled variables at each step) was actually detrimental past about 3 or 4. The "rule of thumb" given in the text says $sqrt(ncol(x))$ is approximately best, which would be between 4 and 5, since I have 22 predictors.

```r
# we can get accuracy by using a 1-0 measure
# of correctness on each row's value
pred <- ifelse(forest$test$predicted == ytest,
    1, 0)
mean(pred)
```

```
## [1] 0.5855911
```

```r
im <- importance(forest)
# MeanDecreaseAccuracy shows decrease in
# accuracy when variable is excluded
im[, 8]
```

```
##         edible        capshape      capsurface        capcolor
##      24.525239       -6.259768       12.979750       17.367477
##        bruises            odor      gillattach       gillspace
##      26.092589       30.042446       14.255796       26.986502
##       gillsize       gillcolor      stalkshape       stalkroot
##      25.233945       19.877669       26.568822       46.427682
```

Evaluation contd.

Here, I show some output on the different classes and overall accuracy of our forest. The caret library is needed to use the function below.

```r
# testing out the caret library
suppressMessages(library(caret))
confusionMatrix(forest$test$predicted, ytest)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction grasses leaves meadows paths urban waste woods
##    grasses     247      0      58    25    22     0    13
##    leaves        0    109       0    98     0     0   100
##    meadows      30      0       1     0     0     0     0
##    paths        11      1       0     9     0     0    13
##    urban        67      0       0     0    50     0     0
##    waste         0      0       0     0     0    38     0
##    woods        84     42       0   109     0     0   497
## 
## Overall Statistics
## 
##                Accuracy : 0.5856
##                  95% CI : (0.5612, 0.6097)
##     No Information Rate : 0.3836
##     P-Value [Acc > NIR] : < 2.2e-16
## 
##                   Kappa : 0.4408
## 
##  Mcnemar's Test P-Value : NA
## 
## Statistics by Class:
## 
```

More Evaluation

Try excluding some variables and see how the overall accuracy changes.

```r
xtrain2 <- subset(xtrain, select = -c(capshape,
    veiltype))
xtest2 <- subset(xtest, select = -c(capshape,
    veiltype))

forest2 = randomForest(x = xtrain2, y = ytrain,
    xtest = xtest2, ytest = ytest, ntree = 2000,
    mtry = 4, importance = T)
confusionMatrix(forest2$test$predicted, ytest)
```

In Hw4

In the homework, the data should be *cardinal* (regular numbers), so you should still be able to calculate MSE the original way, using something like `forest_error = mean((ytest - forest$test$predicted)^2)`.

With factor data, I believe the function is only able to computationally handle around 50 classes, and it gives a warning if you have fewer than 4 or 5 classes. I found that the function is a little touchy for both the predictors and predicted variables.

h2o version

If you're interested in using `h2o` instead, try running the demo for its `randomForest` command to see a quick demo of this package at work. You may need to run it form the Help viewer in RStudio, as it gave me an error for some reason when running it directly.

```
# http://127.0.0.1:35498/help/library/h2o/Demo/h2o.randomFo
# demo(h2o::h2o.randomForest) # may need to
# select it from the help window
```