Lab 4: decision trees

PSTAT 131-231

Week 4

Objectives

- Fit classification and regression trees in R
- Implement cost-complexity pruning
- Compare tree-fitting strategies: grow with a constraint, or grow-then-prune?

Throughout the lab you'll work with the `algae` data from the first homework. Initially, you'll fit a regression tree to the `a1` levels; using this as an example, you'll fit a classification tree to predict high and low `a1` levels.

To evaluate the trees, you'll check their prediction error on a 20% holdout subset of the data.

```
# import algae data
algae <- read_csv('data/algae.csv') %>%
  select(-starts_with('a'), a1) %>%
  mutate(season = factor(season),
         size = factor(size),
         speed = factor(speed))

# hold out 20% of data as a test set
set.seed(12521)
algae_part <- resample_partition(algae, c(test = 0.2, train = 0.8))
train <- as_tibble(algae_part$train)
test <- as_tibble(algae_part$test)
```

Regression trees

Here you'll fit a regression tree to predict algae levels. The cost-complexity pruning procedure may seem somewhat involved, so initially we can explore what will happen if we simply grow a small-ish tree by setting n_{min} (the minimum allowed node size) to a large-ish number.

```
# grow a small regression tree
# NOTE: split = 'deviance' uses RSS for regression
nmin <- 60
tree_opts <- tree.control(nobs = nrow(train),
                          minsize = nmin,
                          mindev = exp(-6))
t_small <- tree(a1 ~ ., data = train,
                control = tree_opts, split = 'deviance')
summary(t_small)
```

```
##
## Regression tree:
## tree(formula = a1 ~ ., data = train, control = tree_opts, split = "deviance")
## Variables actually used in tree construction:
## [1] "Cl"   "PO4"  "oPO4" "NH4"
```

```
## Number of terminal nodes:  5
## Residual mean deviance:  212.5 = 30600 / 144
## Distribution of residuals:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -32.230  -7.008  -1.484   0.000   3.992  75.790
```

The nodes can be examined in detail by checking `$frame`:

```
# examine nodes
t_small$frame %>% select(1:4)
```

```
##       var   n       dev      yval
## 1      Cl 149 66235.184 16.757718
## 2  <leaf>  23  8177.152 47.734783
## 3     PO4 126 31959.019 11.103175
## 6  <leaf>  15  5362.209 30.406667
## 7    oPO4 111 20252.117  8.494595
## 14 <leaf>  49 14835.857 14.008163
## 15    NH4  62  2749.445  4.137097
## 30 <leaf>  56  1197.916  3.183929
## 31 <leaf>   6  1025.793 13.033333
```

This includes information about the splitting variable, the node size, impurity, the prediction, and the cutpoints.

Your turn (1) Display the information in `t_small$frame` for the root node. What is the prediction for observations at the root node?

```
# examine root node
root.node <- t_small$frame[1,]
print(root.node)
```

```
##   var   n      dev     yval splits.cutleft splits.cutright
## 1  Cl 149 66235.18 16.75772        <7.2915         >7.2915
```

The prediction for observations at the root node are a y-value of 16.75772.

Your turn (2) Display the predictions for each of the leaf nodes (hint: use `filter()`) in ascending order. Compare these with the quantiles of `a1`. Does there seem to be any correspondence?

```
# display leaf node predictions
t_small$frame %>%
  filter(var == "<leaf>") %>%
  select(yval) %>%
  arrange(yval)
```

```
##        yval
## 1  3.183929
## 2 13.033333
## 3 14.008163
## 4 30.406667
## 5 47.734783
```

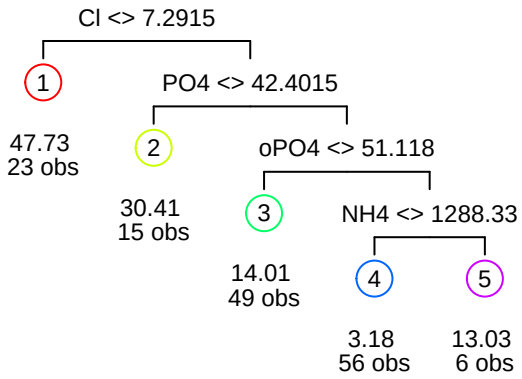
```
# compare with quantiles
quantile(algae$a1, probs = seq(0.3, 0.9, length = 6))
```

```
##    30%    42%    54%    66%    78%    90%
##  1.900  3.600  9.706 16.868 29.832 50.720
```

From what I can see, it does not seem like there is a correspondence between the predictions of the leaf nodes and the quantiles since some of the predictions are close to the quantiles.

The tree can be plotted using `draw.tree`. The function has a few graphical parameters `cex` and `size` that scale the entire figure and the node size, respectively.

```
# plot tree
draw.tree(t_small, cex = 0.75, size = 2.5, digits = 2)
```



The RMSE for this tree on the test data is:

```
# test RMSE
rmse_tsmall <- rmse(t_small, test)
rmse_tsmall
```

```
## [1] 11.94136
```

The idea behind cost complexity pruning is that it can leverage the better fit and increased flexibility of a large tree without overfitting, and it provides a data-driven way to determine the tree size (rather than an artificial stopping rule). We can grow a much larger tree by reducing the minimum node size.

```
# grow a large tree to prune
nmin <- 2
tree_opts <- tree.control(nobs = nrow(train),
                          minsize = nmin,
                          mindev = exp(-8))
t_0 <- tree(a1 ~ ., data = train,
                control = tree_opts, split = 'deviance')
summary(t_0)
```

```
##
## Regression tree:
## tree(formula = a1 ~ ., data = train, control = tree_opts, split = "deviance")
## Number of terminal nodes:  58
## Residual mean deviance:  2.832 = 257.7 / 91
## Distribution of residuals:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   -3.20   -1.06    0.00    0.00    0.80    3.24
```

Your turn (3) Check the training and test RMSE, and plot the tree `t_0`.

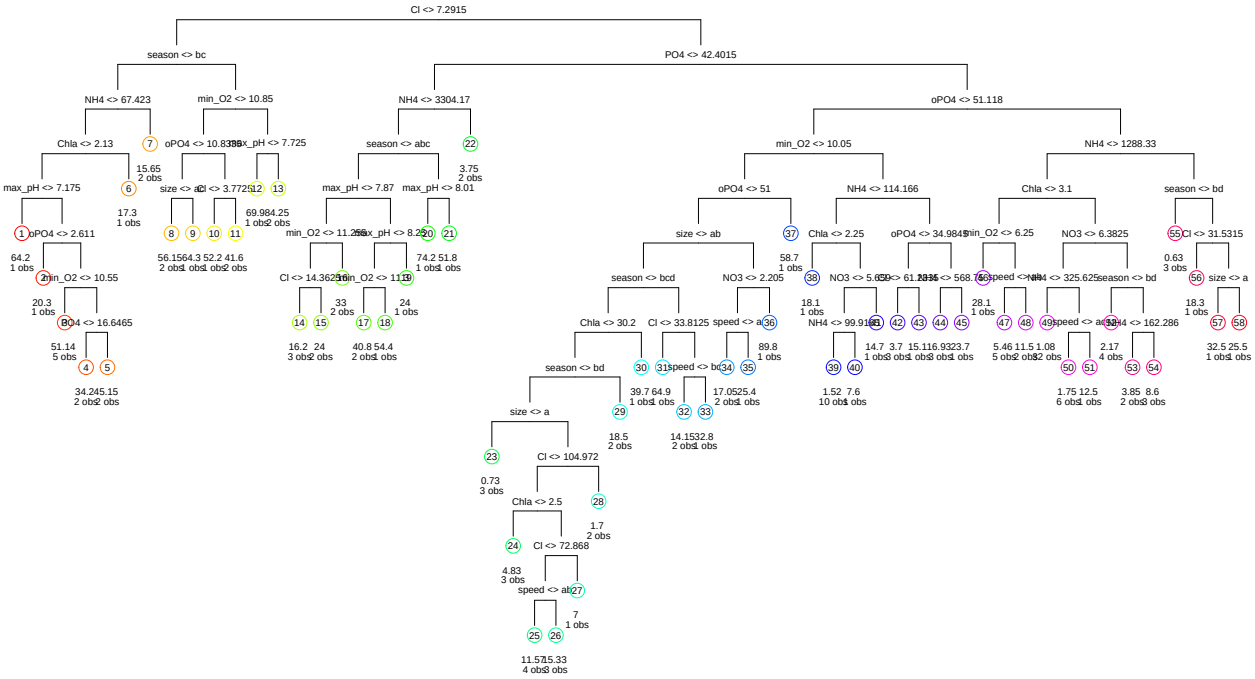
```
# rmses
rmse_train_0 <- rmse(t_0, train)
rmse_test_0 <- rmse(t_0, test)
rmse_train_0
```

```
## [1] 10.68254
```

```
rmse_test_0
```

```
## [1] 19.66142
```

```
# plot tree
draw.tree(t_0, cex = 0.75, size = 2.5, digits = 2)
```



Now cost-complexity pruning can be implemented using `cv.tree`:

```
# cost-complexity pruning
nfolds <- 8
cv_out <- cv.tree(t_0, K = nfolds)
```

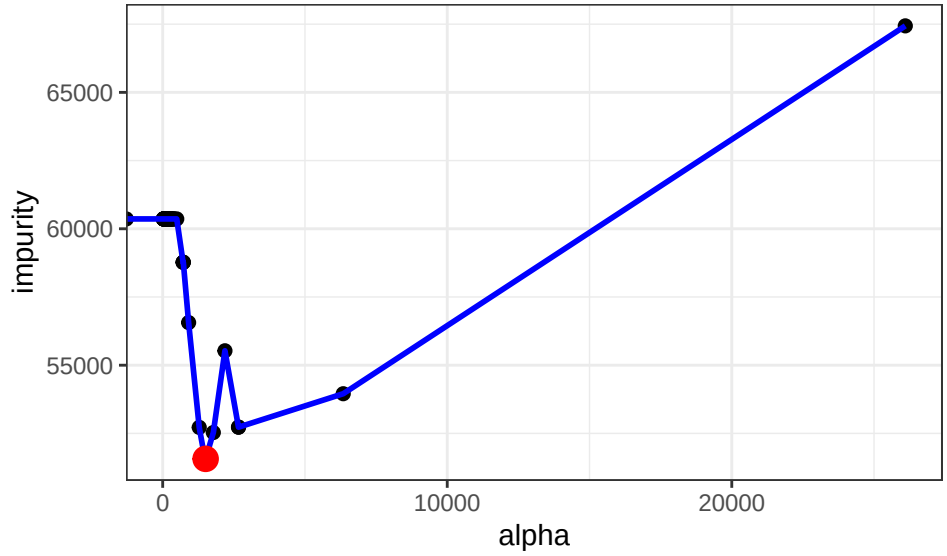
The output is a little unwieldy, so it can be helpful to convert 'by hand' to a tibble. This makes it easier to select the best tuning parameter.

```
# convert to tibble
cv_df <- tibble(alpha = cv_out$k,
                impurity = cv_out$dev,
                size = cv_out$size)

# choose optimal alpha
best_alpha <- slice_min(cv_df, impurity) %>%
  slice_min(size)
```

Your turn (4) Plot the average total tree impurity against the tuning parameter, and describe the trend. Add the best tuning parameter value to the plot as a red point.

```
# plot impurity against tuning parameter
cv_df %>%
ggplot((aes(x = alpha, y = impurity))) +
  geom_point(size = 2) +
  geom_line(color = "blue", size = 1) +
```

```
  geom_point(data = best_alpha, color = "red", size = 4) +
  theme_bw()
```



The final tree can be selected using `prune.tree()`:

```
# select final tree
t_opt <- prune.tree(t_0, k = best_alpha$alpha)
summary(t_opt)
```

```
##
## Regression tree:
## snip.tree(tree = t_0, nodes = c(226L, 25L, 29L, 24L, 15L, 4L,
## 112L, 5L))
## Variables actually used in tree construction:
## [1] "Cl"     "season" "PO4"    "NH4"    "oPO4"   "min_O2" "size"   "NO3"
## Number of terminal nodes:  11
## Residual mean deviance:  117.1 = 16150 / 138
## Distribution of residuals:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -23.610  -4.137  -2.237   0.000   4.563  50.900
```

Your turn (5) Calculate the training and test RMSE for the selected tree `t_opt` and compare this with the training and test RMSE for the small tree. What do you notice? Is there a big improvement in this case?

```
# compare errors with simple tree
rmse_train_opt <- rmse(t_opt, train)
rmse_test_opt <- rmse(t_opt, test)

rmse_train_opt
```

```
## [1] 13.66273
```

```
rmse_test_opt
```

```
## [1] 16.84305
```

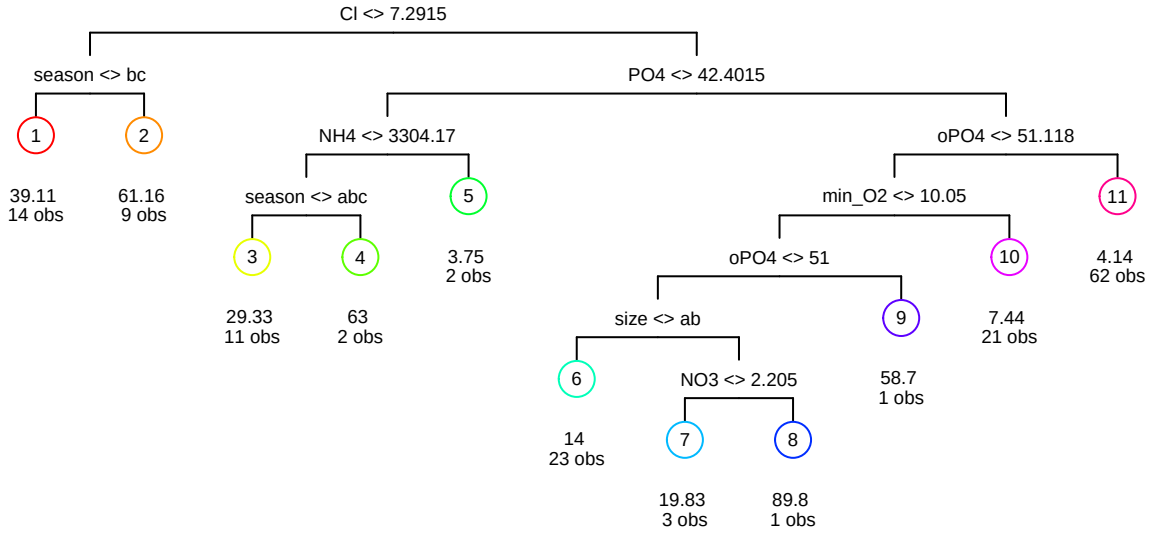
```
rmse_tsmall
```

```
## [1] 11.94136
```

After looking at the different rmse values, I notice that rmse for the training and test data are larger than the rmse for the smaller data. Because of that, I would say that there is not too big of an improvement.

A plot of the tree is shown below:

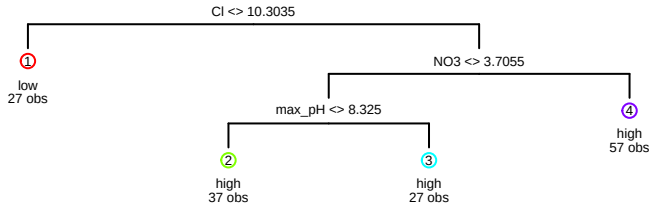
```
# plot
draw.tree(t_opt, cex = 0.6, size = 2.5, digits = 2)
```



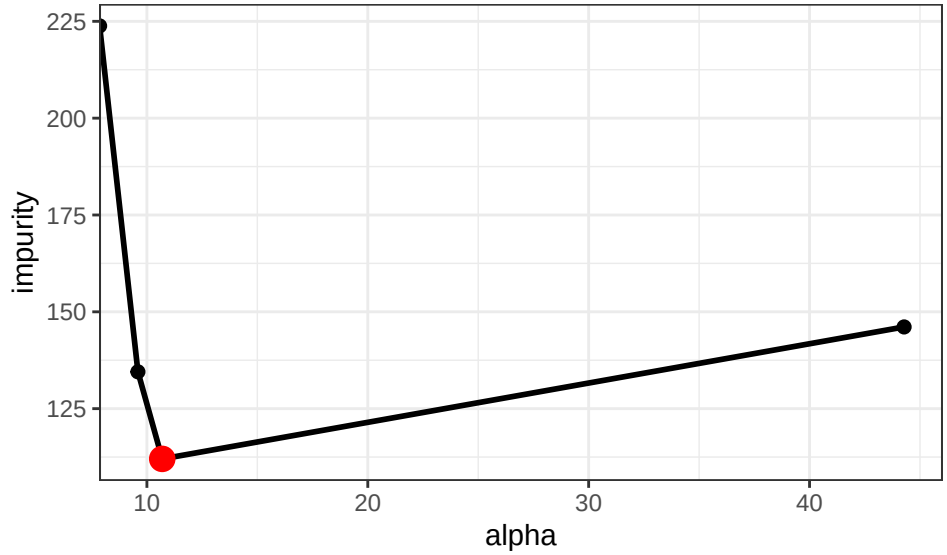
Classification tree

This part is entirely your turn. Let's suppose that an `a1` level is considered high if it exceeds 30. Construct a factor indicating high/low algae levels and fit a classification tree to the new variable using the data in the training partition. Give a plot of the tree and a table of misclassification errors on the test partition.

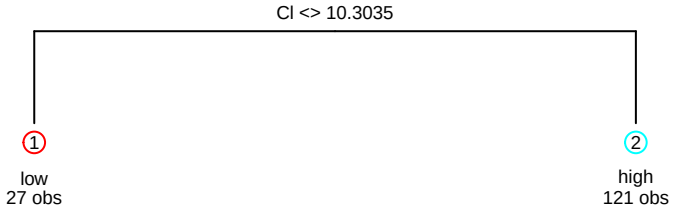
```
# using the same set up as the beginning of the lab
algae_a1 <- algae %>%
  mutate(a1 = factor(a1 > 30, labels = c("high", "low")))

set.seed(12522)
algae_div <- resample_partition(algae_a1, c(test = 0.2, train = 0.8))
train <- as_tibble(algae_div$train)
test <- as_tibble(algae_div$test)

nmin <- 60
tree_opts <- tree.control(nobs = nrow(train),
                          minsize = nmin,
                          mindev = exp(-6))
new_tree <- tree(a1~., data = train, control = tree_opts,split = "deviance")

draw.tree(new_tree, cex = 0.4, digits = 2)
```



```
nfolds <- 8
cv_out <- cv.tree(new_tree, K = nfolds)
cv_df <- tibble(alpha = cv_out$k,
                impurity = cv_out$dev,
                size = cv_out$size)
best_alpha <- slice_min(cv_df, impurity) %>%
  slice_min(size)

cv_df %>%
  ggplot((aes(x = alpha, y = impurity))) +
  geom_point(size = 2) +
  geom_line(color = "black", size = 1) +
  geom_point(data = best_alpha, color = "red", size = 4) +
  theme_bw()
```




```
t_opt <- prune.tree(new_tree, k = best_alpha$alpha)
draw.tree(t_opt, cex = 0.6, digits = 2)
```




```
new_prediction <- predict(t_opt, newdata = test, type = "class")

table(new_prediction, test$a1)
```

```
##
## new_prediction high low
##           high   25   2
##           low     3   9
```