Inducing Uncorrelated Component Models; Random Forests

Introduction

Running Data Set Example

Boston housing prices; predicting median value.

```
library(readr)
library(dplyr)
library(ggplot2)
library(gridExtra)
library(purrr)
library(glmnet)
library(caret)
library(rpart)

# read in data
Boston <- read_csv("http://www.evanlray.com/data/mass/Boston.csv")

# Initial train/test split ("estimation"/test) and cross-validation folds
set.seed(63770)
tt_inds <- caret::createDataPartition(Boston$medv, p = 0.8)
train_set <- Boston %>% slice(tt_inds[[1]])
test_set <- Boston %>% slice(-tt_inds[[1]])

# Function to calculate error rate
calc_rmse <- function(observed, predicted) {
  sqrt(mean((observed - predicted)^2))
}
```

Motivation

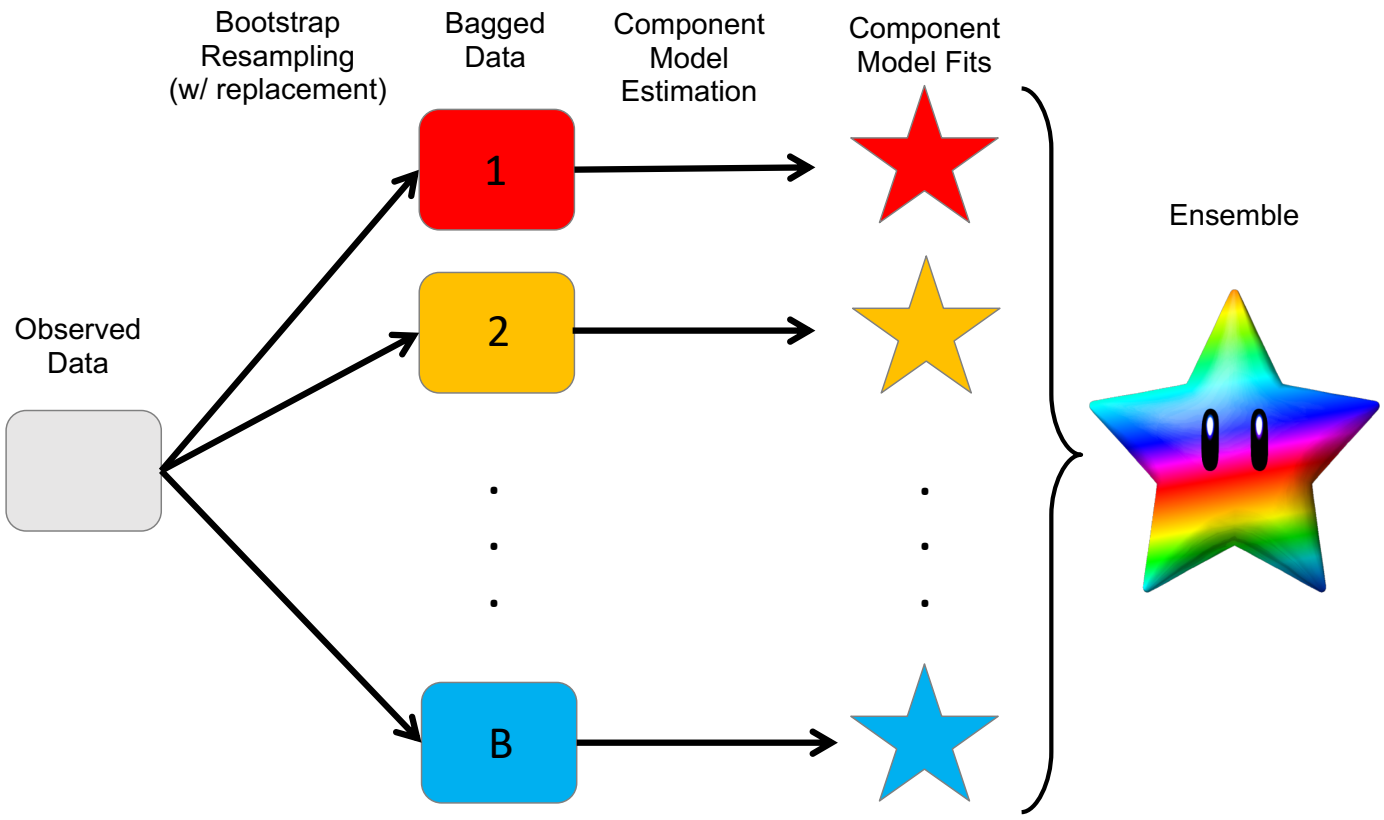
- Ensembles can help reduce variance of predictions
- The improvement over stage 1/component models is largest if the predictions from those component models are uncorrelated.

Two common strategies for getting uncorrelated predictions

- Train different instances of a model on different sets of *rows* of the data set (bagging)
- Train different instances of a model using different *columns* of the data set (or at least, not using all columns in the same way)

Strategy 1: Bagging

Bagging stands for **b**ootstrap **ag**gregation:



Algorithm:

1. For $b = 1, \ldots, B$ (where B is the number of component models, often in the range of 500 or 1000):
   a. Draw a bootstrap sample (i.e., a sample of n rows/observations, drawn with replacement) from the original data set.
   b. Fit the model to the bootstrap sample from step a.
2. Ensemble prediction combines predictions for the B models obtained in step 1 (most commonly, simple average or majority vote).

Example:

```
# I would never really do this like this, code just for illustration of the idea!
fit_one_tree_and_predict_test <- function(b) {
  n <- nrow(train_set)

  bootstrap_resampled_train <- train_set %>%
    dplyr::sample_n(size = n, replace = TRUE)

  tree_fit <- train(medv ~ .,
    data = bootstrap_resampled_train,
    method = "rpart")

  test_predictions <- predict(tree_fit, newdata = test_set)

  return(
    data.frame(
      b = b,
      test_index = seq_len(nrow(test_set)),
      test_prediction = test_predictions
  ))
```

```
}

tree_predictions <- map_dfr(1:500, fit_one_tree_and_predict_test)
```

```
tree_predictions_rmses <- tree_predictions %>%
  group_by(b) %>%
  summarize(
    rmse = calc_rmse(test_set$medv, test_prediction)
  )
head(tree_predictions_rmses)
```

```
## # A tibble: 6 x 2
##       b  rmse
##   <int> <dbl>
## 1     1  6.15
## 2     2  5.70
## 3     3  5.08
## 4     4  5.12
## 5     5  4.84
## 6     6  4.32
```

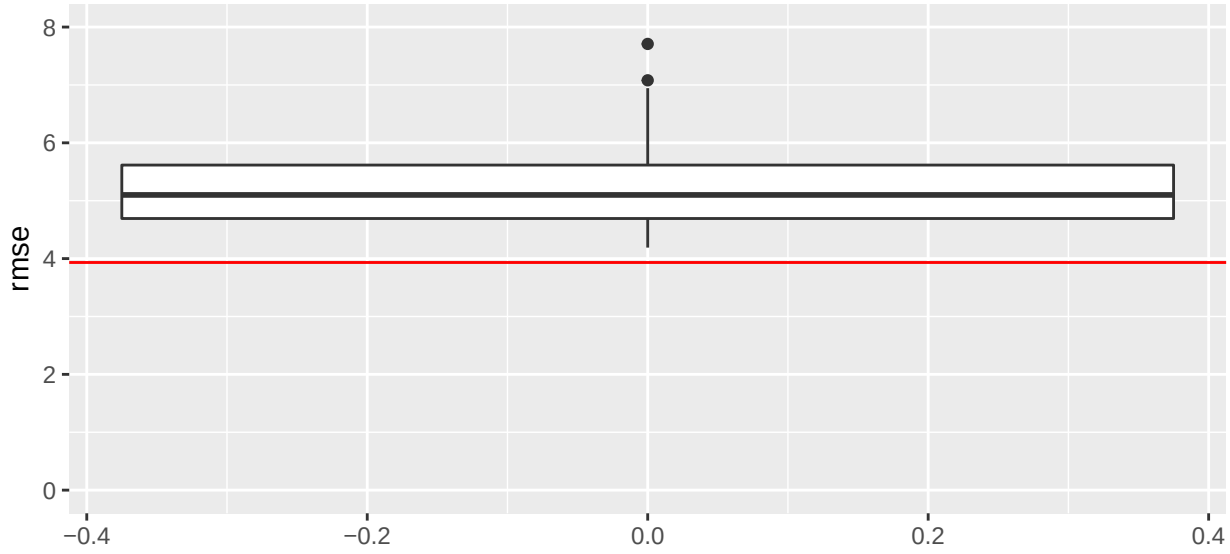
```
dim(tree_predictions_rmses)
```

```
## [1] 500   2
```

```
bagged_ensemble_predictions <- tree_predictions %>%
  group_by(test_index) %>%
  summarize(
    test_prediction = mean(test_prediction)
  )

bagged_ensemble_rmse <- calc_rmse(test_set$medv, bagged_ensemble_predictions$test_prediction)
bagged_ensemble_rmse
```

```
## [1] 3.932775
```

```
ggplot(data = tree_predictions_rmses, mapping = aes(y = rmse)) +
  geom_boxplot() +
  geom_hline(yintercept = bagged_ensemble_rmse, color = "red") +
  ylim(c(0, 8))
```



Strategy 2: Feature Subsets

Similar to above, but different subsets of the features (explanatory variables) are considered for each model, or at different stages within estimation for each model.

- We could divide the explanatory variables into different groups, and train different models on different subsets of the available explanatory variables.
  - Only effective if there are lots of explanatory variables available.

Random Forests

Random forests combine the two strategies above, where the base classifier is a regression or classification tree.

- Bagging is used: each tree in the forest is estimated using a bootstrap sample, drawn with replacement from the original data.
- When growing trees, for each possible split, consider only the splits that could be done using a small fraction of the available explanatory variables, randomly selected.

Tuning parameter (available via `caret` for the implementation in the `randomForest` package for R):

- `mtry`: How many explanatory variables are available for each split?

A smaller value of `mtry` results in:

- more bias in the resulting tree fits (less of a chance to find the best relationships between X and Y)
- less correlation among the resulting tree fits (less of a chance they will use the same splits)
  - therefore, less variance in the final random forest ensemble

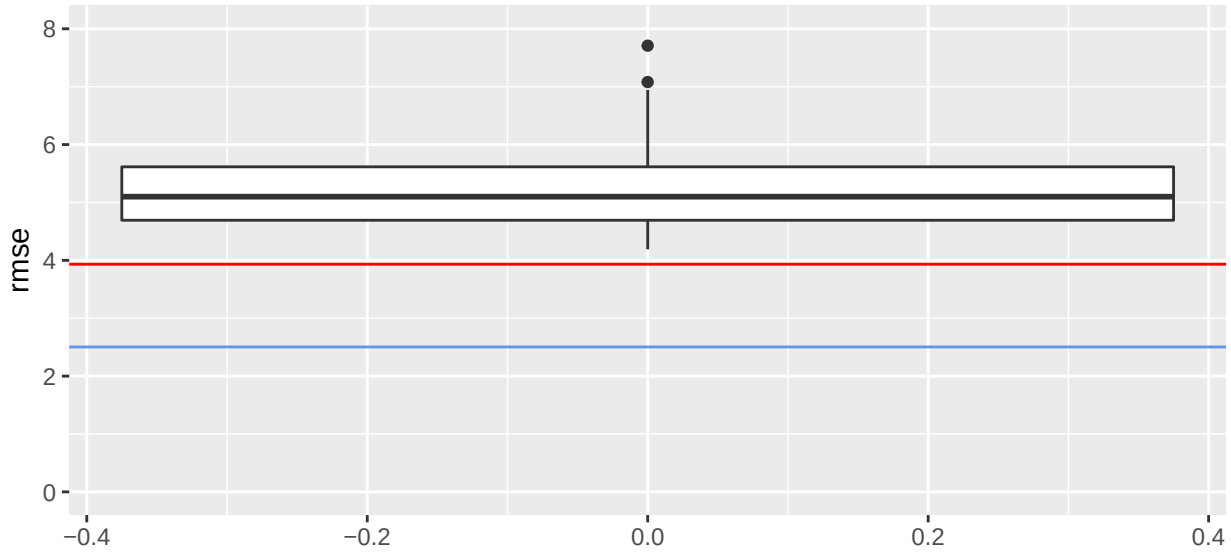
```
rf_fit <- train(
  form = medv ~ .,
  data = train_set,
  method = "rf",
  trControl = trainControl(method = "oob",
    returnResamp = "all",
    savePredictions = TRUE),
  tuneLength = 10
)

rf_fit$results
```

```
##        RMSE  Rsquared mtry
## 1  3.744216 0.8394811    2
## 2  3.549995 0.8557021    3
## 3  3.414874 0.8664777    4
## 4  3.354666 0.8711445    5
## 5  3.310877 0.8744865    6
## 6  3.325431 0.8733806    8
## 7  3.360038 0.8707315    9
## 8  3.362859 0.8705143   10
## 9  3.320884 0.8737266   11
## 10 3.371764 0.8698277   13
```

```
rf_rmse <- calc_rmse(test_set$medv, predict(rf_fit, newdata = test_set))
rf_rmse
```

```
## [1] 2.503623
```

```
ggplot(data = tree_predictions_rmses, mapping = aes(y = rmse)) +
  geom_boxplot() +
  geom_hline(yintercept = bagged_ensemble_rmse, color = "red") +
  geom_hline(yintercept = rf_rmse, color = "cornflowerblue") +
  ylim(c(0, 8))
```



Estimating Test Set Performance with Bagging

- Each tree was trained using data in one of the bags (sampled with replacement from the original data).
- For each tree, there will be some observations that were *not* used in estimation of that tree.

Out of Bag (OOB) Procedure for estimating test set error:

- For each training set observation $i = 1, \ldots, n$:
  - There will be about $B/3$ bootstrap samples that did not include observation i.
  - For each tree b where observation i was **not** in the bag used for estimating that tree, get the predicted value $\hat{y}_i^b$.
  - Compute $\hat{y}_i^{OOB}$ as the average of those predictions (for regression) or the majority vote (for classification).
- Use the out of bag predictions $\hat{y}_1^{OOB}, \ldots, \hat{y}_n^{OOB}$ to estimate test set performance:
  - Estimate test set MSE: $\frac{1}{n}\sum_i (y_i - \hat{y}_i^{OOB})^2$
  - Estimate test set classification error rate: $\frac{1}{n}\sum_i 1(y_i \neq \hat{y}_i^{OOB})$