Ensemble

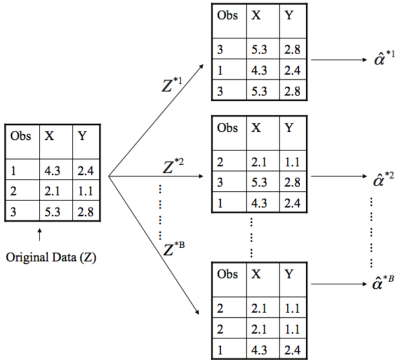
Yang Wang

College of Charleston

- Decision trees are simple and interpretable models for regression and classification.
- However, they are often not competitive with other methods in terms of prediction accuracy
- Bagging (Bootstrap aggregating), random forest, and boosting are good methods for improving the prediction accuracy of trees. They work by growing many trees on the training data and then combining the predictions of the resulting ensemble of trees.
- The latter two methods - random forests and boosting - are among the state-of-the-art methods for supervised learning, However, their results can be difficult to interpret.

Bootstrapping

- Resampling procedure
- Idea: repeatedly draw samples with replacement from a known sample and compute the item of interest on each sample



Obs	X	Y
1	4.3	2.4
2	2.1	1.1
3	5.3	2.8

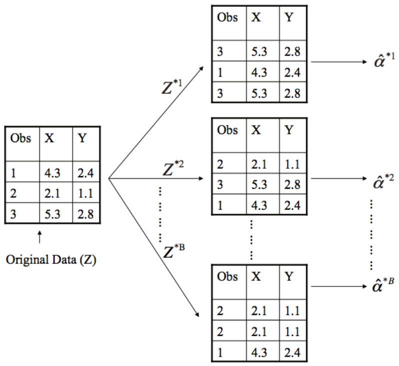
Obs	X	Y
3	5.3	2.8
1	4.3	2.4
3	5.3	2.8

Obs	X	Y
2	2.1	1.1
3	5.3	2.8
1	4.3	2.4

Obs	X	Y
2	2.1	1.1
2	2.1	1.1
1	4.3	2.4

Bootstrapping: Example

- Original dataset has 3 observations
- *B* bootstrap samples (each of size 3) are created by drawing sample with replacement from the original dataset
- Suppose α is a parameter we are interested in (example: mean)
- We will compute an estimate of α *i.e.* $\hat{\alpha}$ for each bootstrap sample
- The *B* estimates of of α obtained from *B* bootstrap samples gives a sampling distribution of $\hat{\alpha}$



Bootstrapping: Example

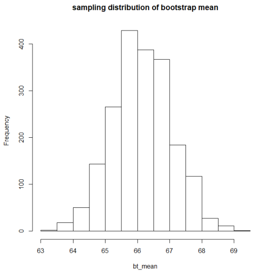
- Suppose we have the following data on the height of 15 individuals
- Data (in inches): 72 70 65 71 72 66 60 61 66 62 67 67 68 67 71
- We want to create a confidence interval for the mean height
- Draw 2000 samples of size 15 with replacement - these are the bootstrap samples

```
> ## original data
> t(height)
  [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14] [,15]
   63   64   67   71   62   71   71   68   68    61    62    62    68    65    69
> ## create 2000 bootstrap samples
> bt_samples <- bootstraps(height, times = 2000)
> ## example of first bootstrap sample
> height[bt_samples$splits[1][[1]]$in_id,]
 [1] 68 62 69 71 62 69 71 61 64 62 71 63 71 65 71
> ## example of second bootstrap sample
> height[bt_samples$splits[2][[1]]$in_id,]
 [1] 68 68 68 67 68 62 62 64 62 71 68 61 62 68 68
```

Bootstrapping: Example

- Compute mean on each bootstrap sample.
- The histogram shows the sampling distribution of mean.
- Find out the 0.025 and 0.975 sample quantiles of the bootstrap means.
- These sample quantiles constructs the 95% confidence interval for mean height.

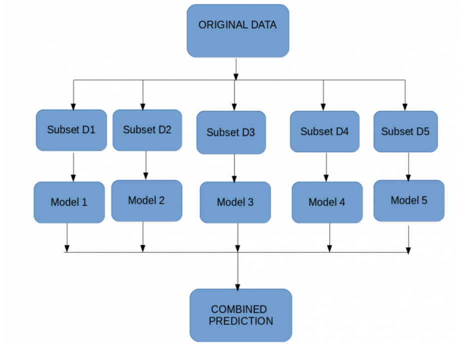
```
## calculate mean on each bootstrap sample
bt_mean <- rep(NA, 2000)
for (i in 1:2000){
  bt_mean[i] <- mean(height[bt_samples$splits[i][[1]]$in_id,])
}
hist(bt_mean, main = "sampling distribution of bootstrap mean")
## 95% confidence interval for mean height using bootstrap method
> quantile(bt_mean, probs = c(0.025, 0.975))
    2.5%    97.5%
64.40000 67.93333
> ## traditional confidence intervals
> c(mean(height[,1]) - qt(.975,14)*sd(height[,1])/sqrt(15),
      mean(height[,1]) + qt(.975,14)*sd(height[,1])/sqrt(15))
[1] 64.12716 68.13951
```



- Advantage: no distribution is assumed

Ensembles

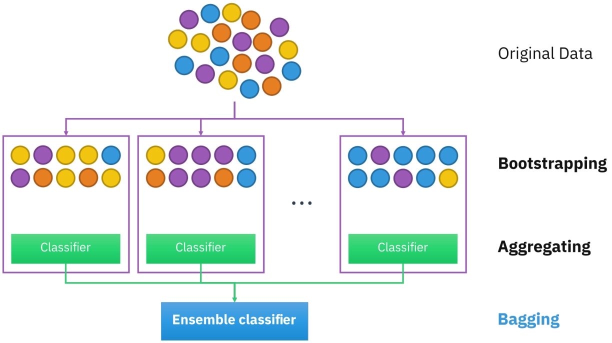
- Goal: predict Y for given X
- May be used for both classification and regression
- Combines the results of multiple models



picture: https://www.analyticsvidhya.com/blog/2018/06/comprehensive-guide-for-ensemble-models/

Bagging (Bootstrap aggregating)

- Generate B bootstrap samples.
- Fit predictive model on each bootstrap sample.
- The final prediction is an average of the predictions obtained from the B models fitted on bootstrap samples.
  - Regression: $\frac{1}{B}\sum_{b=1}^{B} T_b(\boldsymbol{x})$
  - Classification: choose the majority class label among $T_1(\boldsymbol{x}), T_2(\boldsymbol{x}), \ldots, T_B(\boldsymbol{x})$.



Reduce Variance Without Increasing Bias

- Recall: E(X+Y) = E(X) + E(Y)
- Recall: Var(X+Y) = Var(X) + Var(Y) + 2 Cov(X,Y)
- Recall: Var(aX) = a^2 Var(X) (a is a constant)
- Consider identically distributed random variables $T_1, T_2, \ldots T_B$ which are used for estimating a parameter θ.
- The bias of the statistic T_i is given by $bias(\theta) = E(T_i) - \theta$. If $bias(\theta) = 0$.
- T_i is called an unbiased estimator of parameter θ.
- Define $T_{avg} = \frac{1}{B}(T_1 + \cdots + T_B)$.
- $E(T_{avg}) = \frac{1}{B}\sum_{i=1}^{B} E(T_i)$.
- $Var(T_{avg}) = \frac{1}{B^2}(\sum_{i=1}^{B} Var(T_i) + \sum_{\substack{i=1 \\ i \neq j}}^{B} \sum_{j=1}^{B} Cov(T_i, T_j))$.

Reduce Variance Without Increasing Bias

- $E(T_{avg}) = \frac{1}{B}\sum_{i=1}^{B} E(T_i)$.
- $Var(T_{avg}) = \frac{1}{B^2}(\sum_{i=1}^{B} Var(T_i) + \sum_{\substack{i=1 \\ i \neq j}}^{B} \sum_{j=1}^{B} Cov(T_i, T_j))$.
- Suppose $E(T_i) = \mu$, $Var(T_i) = \sigma^2$ and $Cov(T_i, T_j) = \rho\sigma^2$
- ρ is the correlation between T_i and T_j.
- $E(T_{avg}) = \mu$ *i.e* bias remains the same.
- $Var(T_{avg}) = \frac{1}{B^2}(B\sigma^2 + B(B-1)\rho\sigma^2) = \rho\sigma^2 + (1-\rho)\sigma^2/B$.
- For sufficiently large B the second term vanishes.
- Since $\rho < 1$, the variance of $Var(T_{avg}) < \sigma^2$.
- Averaging reduces variance while retaining the bias.
- Idea of ensemble method: build multiple models and use their average for variance reduction.

Bagging: R example

Load the diabetes.csv

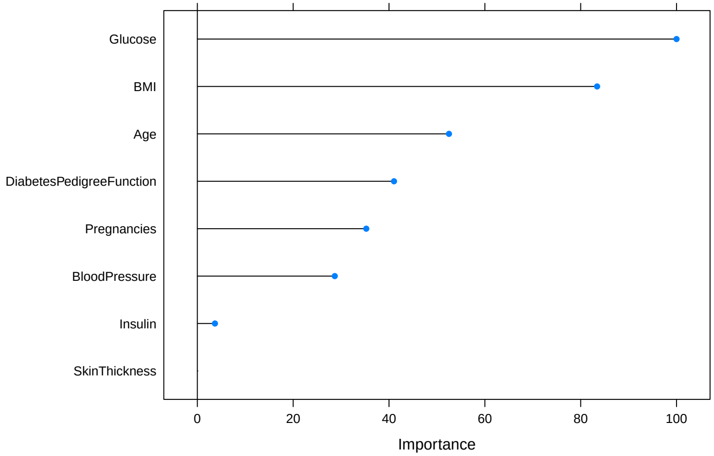
```
library(caret)
set.seed(0)
train_control <- trainControl(method="cv", number=10)
## specify nbagg to control the number of trees. default value is 25
bag <- train(Outcome ~ . , data = training.data, method = "treebag",
               trControl = train_control, nbagg = 50)
> print(bag)
Bagged CART

538 samples
  8 predictor
  2 classes: 'no', 'yes'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 484, 484, 484, 484, 484, 485, ...
Resampling results:

  Accuracy   Kappa
  0.7510482  0.433193

plot(varImp(bag))
> bag$finalModel

Bagging classification trees with 50 bootstrap replications
```

Bagging: R example

```
# get prediction on the test data
pred.test.bag = predict(bag$finalModel, test.data, type = 'class')

# create confusion matrix
> confusionMatrix(pred.test.bag, test.data$Outcome, positive = "yes")
Confusion Matrix and Statistics

          Reference
Prediction  no yes
       no  121  26
       yes  29  54

               Accuracy : 0.7609
                 95% CI : (0.7004, 0.8145)
    No Information Rate : 0.6522
    P-Value [Acc > NIR] : 0.000245

                  Kappa : 0.4775

 Mcnemar's Test P-Value : 0.787406

            Sensitivity : 0.6750
            Specificity : 0.8067
         Pos Pred Value : 0.6506
         Neg Pred Value : 0.8231
             Prevalence : 0.3478
         Detection Rate : 0.2348
   Detection Prevalence : 0.3609
      Balanced Accuracy : 0.7408

       'Positive' Class : yes
```

Bagging: R example

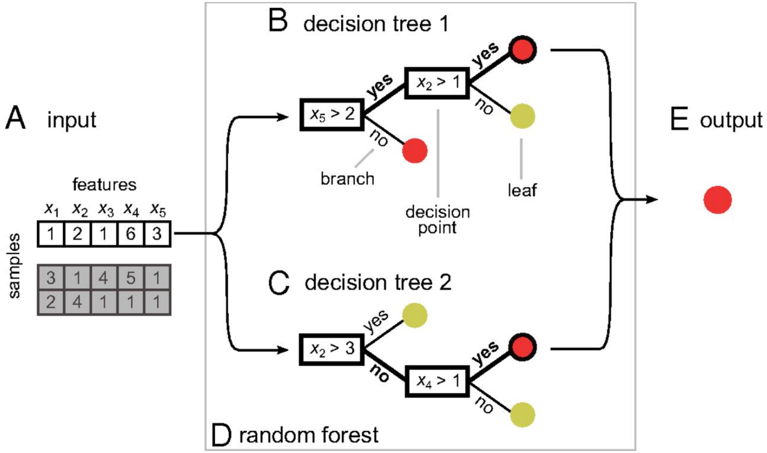
Variable importance plot



Random forest

- Generate B bootstrap samples.
- Fit decision trees on each bootstrap sample - but with a little tweak.
    - Each time a split in a tree is considered, do not use the full set of p predictors.
    - Use a random sample of m predictors (typically $m \approx \sqrt{p}$) .
- The final prediction is an average of the predictions obtained from the B models fitted on bootstrap samples.
    - Regression: $\frac{1}{B}\sum_{b=1}^{B} T_b(\boldsymbol{x})$
    - Classification: choose the majority class label among $T_1(\boldsymbol{x}), T_2(\boldsymbol{x}), \ldots, T_B(\boldsymbol{x})$.

Random forest



picture: https://www.pnas.org/content/115/8/1690

Why does random forest work?

- Random forest often performs better than bagging, although at each split in the tree, the algorithm is not even allowed to consider a majority of the available predictors.
- Suppose that there is one very strong predictor in the data set, along with a number of other moderately strong predictors.
- Then in the collection of bagged trees, most or all of the trees will use this strong predictor in the top split.
- All of the bagged trees will look quite similar to each other and the predictions from the bagged trees will be highly correlated.
- Averaging many highly correlated quantities does not lead to as large of a reduction in variance as averaging many uncorrelated quantities.
- This means that bagging will not lead to a substantial reduction in variance over a single tree in this setting.
- Random forests overcome this problem by forcing each split to consider only a subset of the predictors.
- Therefore, on average $(p - m)/p$ of the splits will not even consider the strong predictor, and so other predictors will have more of a chance.
- We can think of this process as decorrelating the trees, thereby making the average of the resulting trees less variable and hence more reliable.

Random forest: R example

```
library(caret)
set.seed(0)
> modelLookup("rf")
  model parameter                         label forReg forClass probModel
1    rf      mtry #Randomly Selected Predictors   TRUE     TRUE      TRUE

train_control <- trainControl(method="cv", number=10)
rf <- train(Outcome ~ . , data = training.data, method = "rf",
             trControl = train_control, tuneLength = 3)
# metric = "Kappa" may be mentioned if best tree should be selected based on that
## specify tuning parameters using tuneGrid
# rf <- train(Outcome ~ . , data = training.data, method = "rf", ntree = 50,
#                 trControl = train_control, tuneGrid = expand.grid(mtry = c(2,5,8)))
> print(rf)
Random Forest

538 samples
  8 predictor
  2 classes: 'no', 'yes'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 484, 484, 484, 484, 484, 485, ...
Resampling results across tuning parameters:

  mtry  Accuracy   Kappa
  2     0.7492313  0.4221906
  5     0.7603774  0.4519239
  8     0.7492662  0.4330171

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was mtry = 5.
```

Random forest: R example

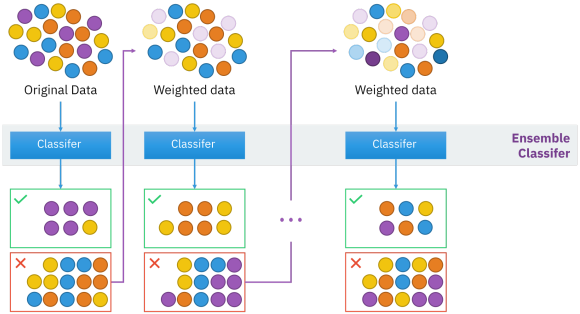
```
> rf$finalModel
Call:
 randomForest(x = x, y = y, mtry = param$mtry)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 5
        OOB estimate of  error rate: 25.84%
Confusion matrix:
     no yes class.error
no  292  58   0.1657143
yes  81 107   0.4308511

# get prediction on the test data
pred.test.rf = predict(rf$finalModel, test.data, type = 'class')
# create confusion matrix
> confusionMatrix(pred.test.rf, test.data$Outcome, positive = "yes")
Confusion Matrix and Statistics
          Reference
Prediction  no yes
       no  125  23
       yes  25  57
               Accuracy : 0.7913
                 95% CI : (0.733, 0.8419)
    No Information Rate : 0.6522
    P-Value [Acc > NIR] : 2.878e-06
                  Kappa : 0.5427
 Mcnemar's Test P-Value : 0.8852
            Sensitivity : 0.7125
            Specificity : 0.8333
         Pos Pred Value : 0.6951
         Neg Pred Value : 0.8446
             Prevalence : 0.3478
```

Boosting

- Recall: Bagging
  - Generate B bootstrap samples
  - Fit model on each bootstrap sample
  - Combine all models to create a simple predictive model
  - Each tree is built on a bootstrap data set, which is independent of the other trees
- Boosting works in a similar way, except that the trees are grown sequentially
- Each tree is grown using information from previously grown trees
- Boosting does not involve bootstrap sampling; instead each tree is fit on a modified version of the original data set

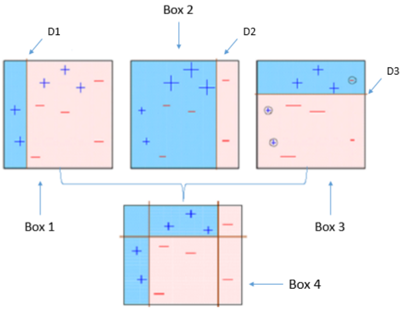
Boosting

- In the beginning the process assigns equal weight to each observations
- After fitting a tree, the performance of the model is evaluated
- The missclassified observations are assigned a higher weight and the correctly classified observations get a lower weight
- A tree is fitted again on the reweighted data
- The process is repeated M times
- The final prediction is a weighted average of the predictions of all fitted models



Adaboost

- Initialize the observation weights $w_i = 1/N, i = 1, 2, ..., N$
- For $m = 1$ to M :
  - Fit a classifier $G_m(x)$ to the training data using w_i
  - Compute $err_m = \frac{\sum_{i=1}^{N} w_i \mathbf{I}(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i}$
  - Compute $\alpha_m = log((1 - err_m)/err_m)$
  - Set $w_i \leftarrow w_i.exp[\alpha_m.\mathbf{I}(y_i \neq G_m(x_i))], i = 1, 2, ..., N$
- Output $G(x) = sign[\sum_{m=1}^{M} \alpha_m G_m(x)]$



Adaboost: R example

```
library(caret)
library(ada)
set.seed(0)
train_control <- trainControl(method="cv", number=10)
ada <- train(Outcome ˜ . , data = training.data, method = "ada",
            trControl = train_control, tuneLength = 3)
> print(ada)
Boosted Classification Trees
538 samples
  8 predictor
  2 classes: 'no', 'yes'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 484, 484, 484, 484, 484, 485, ...
Resampling results across tuning parameters:

  maxdepth  iter  Accuracy   Kappa
  1          50   0.7622292  0.4451545
  1         100   0.7585255  0.4294762
  1         150   0.7658980  0.4523454
  2          50   0.7511530  0.4228118
  2         100   0.7548567  0.4318264
  2         150   0.7660377  0.4614452
  3          50   0.7641160  0.4597316
  3         100   0.7660377  0.4644768
  3         150   0.7678546  0.4727207

Tuning parameter 'nu' was held constant at a value of 0.1
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were iter = 150, maxdepth = 3 and nu = 0.1.he largest value.
The final value used for the model was mtry = 5.
```

Adaboost: R example

```
# get prediction on the test data
pred.test.ada = predict(ada$finalModel, test.data)

# create confusion matrix
> confusionMatrix(pred.test.ada, test.data$Outcome, positive = "yes")
Confusion Matrix and Statistics

          Reference
Prediction  no yes
       no  122  29
       yes  28  51

               Accuracy : 0.7522
                 95% CI : (0.6912, 0.8066)
    No Information Rate : 0.6522
    P-Value [Acc > NIR] : 0.0007075

                  Kappa : 0.4522

 Mcnemar's Test P-Value : 1.0000000

            Sensitivity : 0.6375
            Specificity : 0.8133
         Pos Pred Value : 0.6456
         Neg Pred Value : 0.8079
             Prevalence : 0.3478
         Detection Rate : 0.2217
   Detection Prevalence : 0.3435
      Balanced Accuracy : 0.7254

       'Positive' Class : yes
```

Advantage and disadvantage of ensemble methods

- Advantage:
  - Improved accuracy by combining together hundreds of models into a single prediction.
  - Low variance.
- Shortcoming:
  - Difficult to interpret: when we bag a large number of trees, it is not possible to represent the resulting prediction using a single tree.