Random Forests

September 29, 2019

Motto

The clearest way into the Universe is through a forest wilderness.
John Muir, environmentalist

Boostrap – a revisit

- The bootstrap is, in general, 'creating' new (pseudo) data sets from the existing ones.
- In its original set-up, it is used when it is hard or even impossible to directly compute the standard deviation of an estimate of the quantity of interest.
- **It was not intended** to improve the estimate of a quantity of interest.
- Let us recall 'estimates' of θ based on bootstrap samples

$$\hat{\theta}_1^*, \ldots, \hat{\theta}_B^*$$

- The variability of these estimates as measured by a standard deviation allows to assess the variability of the original estimate $\hat{\theta}$.

Can boostrapp improve estimation?

- Bootstrapping was not intended to improve the estimate of a quantity of interest but one could think that averaging results from the bootstrap sample may reduce variability of the estimate and thus improve estimation.
- For example, one could think that the following estimate could be an improvement due to averaging

$$\hat{\theta}_{bag} = \frac{\hat{\theta}^*_1 + \cdots + \hat{\theta}^*_B}{B}$$

- However, for linear estimation methods, such an estimate will be essentially the same as the one that we started before bootstrapping, i.e.

$$\hat{\theta}_{bag} \approx \hat{\theta}$$

Example – bootstrapping means

- Let us consider $\hat{\theta} = \bar{x}$ as an estimator of the unknown mean μ.
- Consider B bootstrap samples and corresponding means $\bar{x}_1^*, \ldots, \bar{x}_B^*$.
- Then 'bagged' estimator is

$$\bar{x}_{bag} = \frac{\bar{x}_1^* + \cdots + \bar{x}_B^*}{B} = \frac{\frac{\sum_{i=1}^n x_{1,i}^*}{n} + \cdots + \frac{\sum_{i=1}^n x_{B,i}^*}{n}}{B} = \frac{\sum_{i=1}^n \frac{x_{1,i}^* + \cdots + x_{B,i}^*}{B}}{n}$$

- If B is getting large, then each of $\frac{x_{1,i}^* + \cdots + x_{B,i}^*}{B}$ is converging to $\bar{x}$ and thus the bagged estimator is approximately equal to $\bar{x}$ – the original estimator, and thus **no improvement**.

Bagging – boostrapp for highly variable estimates

- If the estimate is non-linear and with high variance, the averaging bootstrap estimates may have sense.
- For example, the decision trees suffer from high variance.
- One can take B bootstrap samples from $(x_1, y_1), \ldots, (x_N, y_N)$ and corresponding bootstrap binary tree predictions $\hat{f}_i^*$, $i = 1, \ldots, B$.
- Each bootstrap tree will typically involve different features than the original, and might have a different number of terminal nodes.
- One can consider bootstrap averages of $\hat{f}_i^*$ from trees predictions at input vector x. The bagged estimate is this average prediction at x from these B trees

$$\hat{f}_{bag}(x) = \frac{\hat{f}_1^*(x) + \cdots + \hat{f}_B^*(x)}{B}$$

Bagging for a classification tree

- Tree produces a classifier $\hat{G}(x)$.
- If $\hat{G}_i^*(x)$'s are bootstrap classifiers, then the bagged classifier $\hat{G}_{bag}(x)$ selects the class with the **most votes** from among $\hat{G}_i^*(x)$'s – **Consensus**
- If the classifier method produces also estimates of **classification probabilities** $\hat{p}_1(x)$ and $\hat{p}_2(x) = 1 - \hat{p}_1(x)$, then the bagged probabilities are obtained as

$$\hat{p}_{1,bag}(x) = \frac{\hat{p}_{1,1}^*(x) + \cdots + \hat{p}_{1,B}^*(x)}{B}$$

- Having the bagged probabilities can also determine an alternative bagged classifier. Namely, the class is chosen that has the **highest bagged probability**.

Example

- A sample of size $N = 30$, with two classes and five features, each having a standard Gaussian distribution with pairwise correlation 0.95.
- The response Y was generated according to

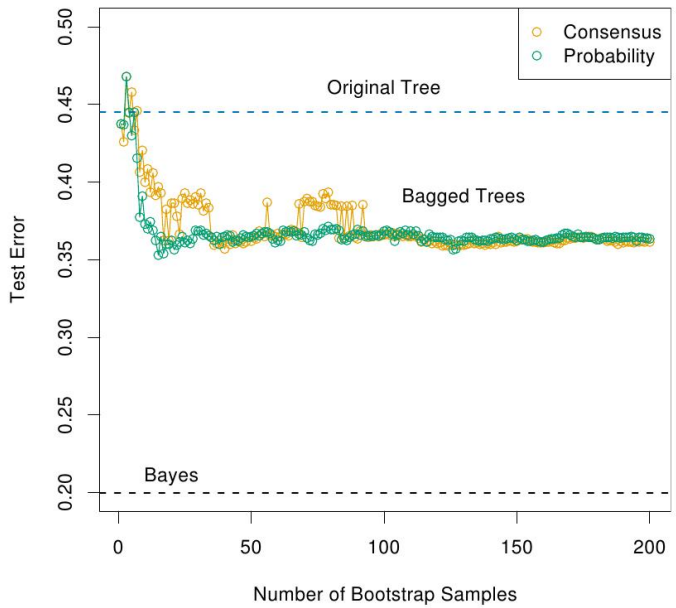
$$P(Y = 1 | x_1 \leq 0.5) = 0.2,$$
$$P(Y = 1 | x_1 > 0.5) = 0.8.$$

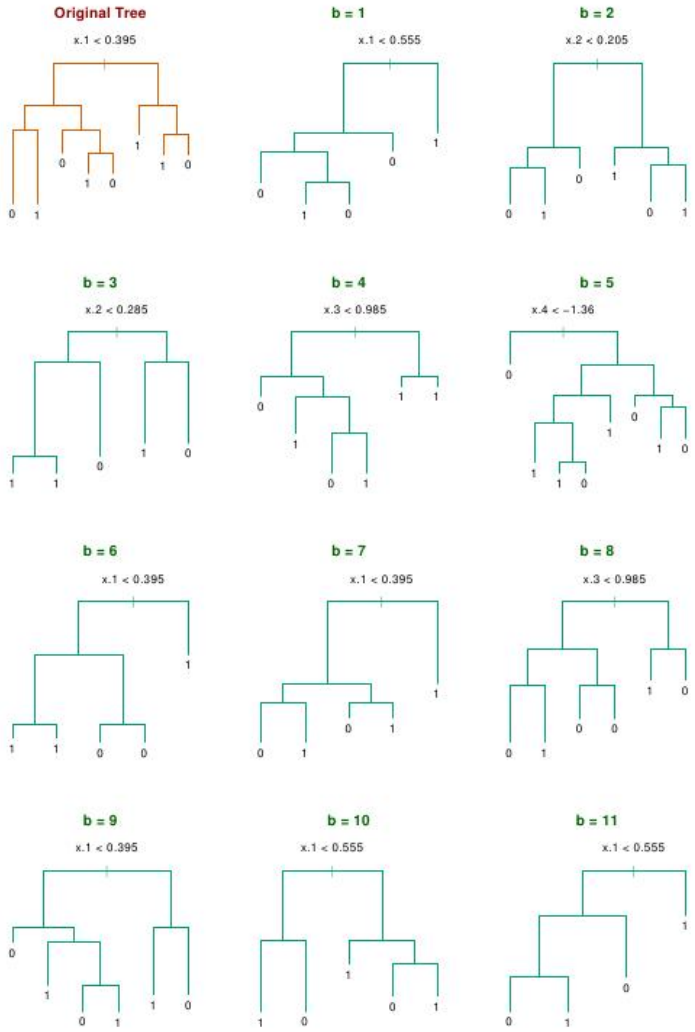
- **What would be the best classifier if you would know how the data were simulated?**
- A test sample of size 2000 was also generated from the same population.
- Fit classification trees to the training sample and to each of 200 bootstrap samples. No pruning was used.

Results

The optimal classifier would have the error rate:

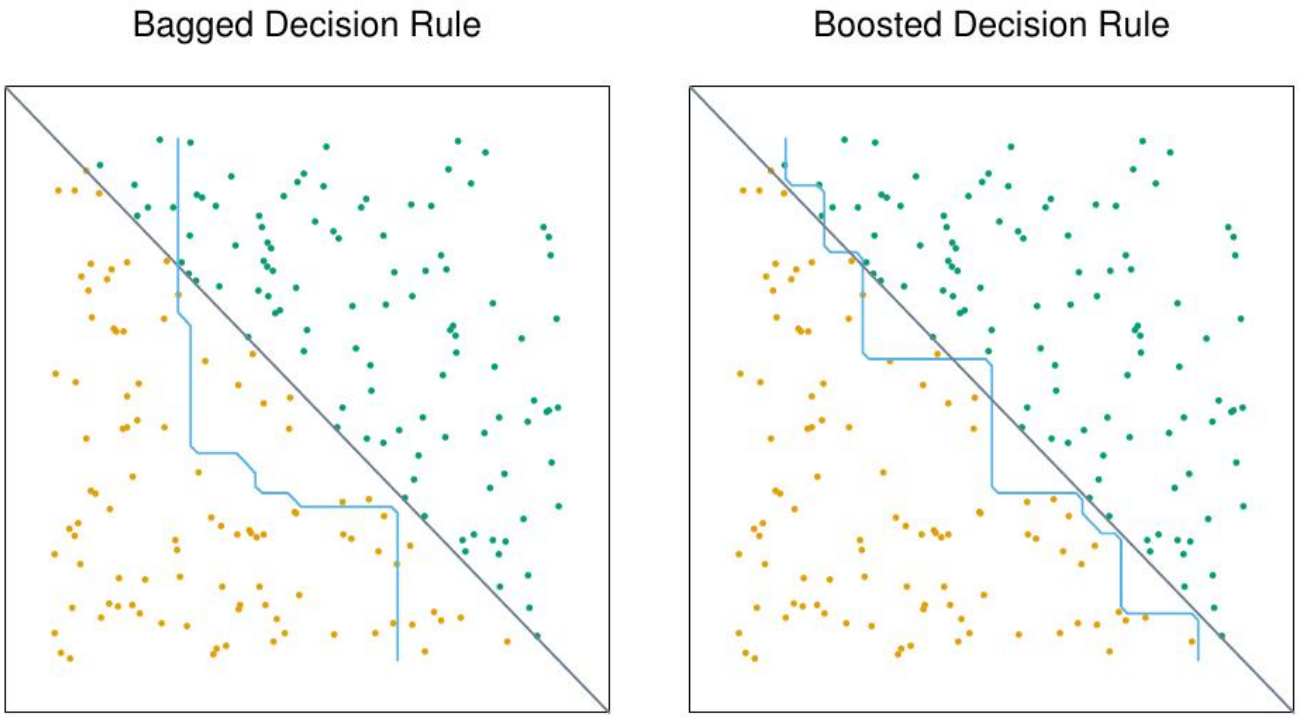
$$P(Y = 1, X_1 < 0.5) + P(Y = 0, X_1 \geq 0.5) =$$
$$P(Y = 1|X_1 < 0.5)P(X_1 < 0.5) + P(Y = 0|X_1 \geq 0.5)P(X_1 \geq 0.5) = 0$$





Not always bagging is good enough

The 100 data points – two features and two classes, separated by the gray linear boundary $x_1 + x_2 = 1$. Classifier $\hat{G}(x)$ a single axis-oriented split, choosing the split along either x_1 or x_2 that produces the largest decrease in training misclassification error.



The decision boundary obtained from bagging the 0-1 decision rule over $B = 50$ bootstrap samples is shown by the blue curve in the left panel. It does a poor job of capturing the true boundary.

Why bagging sometimes is not working?

- Each tree generated in bagging is **identically distributed (id)**, the expectation of an average of B such trees is the same as the expectation of any one of them

$$E(\hat{f}_{bag}(x)) = \frac{E(\hat{f}_1^*(x)) + \cdots + E(\hat{f}_B^*(x))}{B} = E(\hat{f}_1^*(x))$$

- This means the bias of bagged trees with respect to the optimal predictor

$$bias = E(\hat{f}_{bag}(x)) - G(x)$$

  is the same as that of the individual trees.

- The only hope of improvement is through variance reduction. This is in contrast to boosting, where the trees are grown in an adaptive way to remove bias, and hence are not id.

Variance reduction

- It is well known in statistics that the estimation mean square error is made of the two components: the squared bias and the variance of the estimate

$$MSE = bias^2 + variance$$

- An average of B iid random variables has variance σ^2/B.
- If the variables are simply i.d. (identically distributed, but not necessarily independent) with positive pairwise correlation , the variance of the average is

$$\sigma^2(\rho + (1 - \rho^2)/B)$$

- As B increases, the second term disappears, but the first remains, and hence the size of the correlation of pairs of bagged trees limits the benefits of averaging.

Example

- Let $X_1, \ldots, X_N$ be identically distributed normal variables with mean μ and variance σ^2 jointly pairwise correlated with the correlation ρ.
- Consider the sample mean $\bar{X}$. What is the mean and variance of $\bar{X}$?

$$E\bar{X} = \mu, \quad Var\bar{X} = \sigma^2(\rho + (1 - \rho^2)/n)$$

- The idea of bootstrap worked if the original sample is independent identically distributed. However if they are not, the boostrap will reproduce correlation between pairs of the data.
- If each of $X_i = (X_{i1}, \ldots, X_{ip})$ is vector valued and not strongly correlated, then by randomly sampling only some coordinates of X one can reduce correlation between bootstrap samples (specially when the coordinates of X_i are not highly correlated) and thus reducing the variance of the estimate.
- This idea is explored in **random forests**.

Random Forest Algorithm

Here are details of the algorithm

Algorithm 15.1 *Random Forest for Regression or Classification.*

1. For $b = 1$ to B:
   (a) Draw a bootstrap sample $\mathbf{Z}^*$ of size N from the training data.
   (b) Grow a random-forest tree T_b to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size n_{min} is reached.
      i. Select m variables at random from the p variables.
      ii. Pick the best variable/split-point among the m.
      iii. Split the node into two daughter nodes.
2. Output the ensemble of trees $\{T_b\}_1^B$.

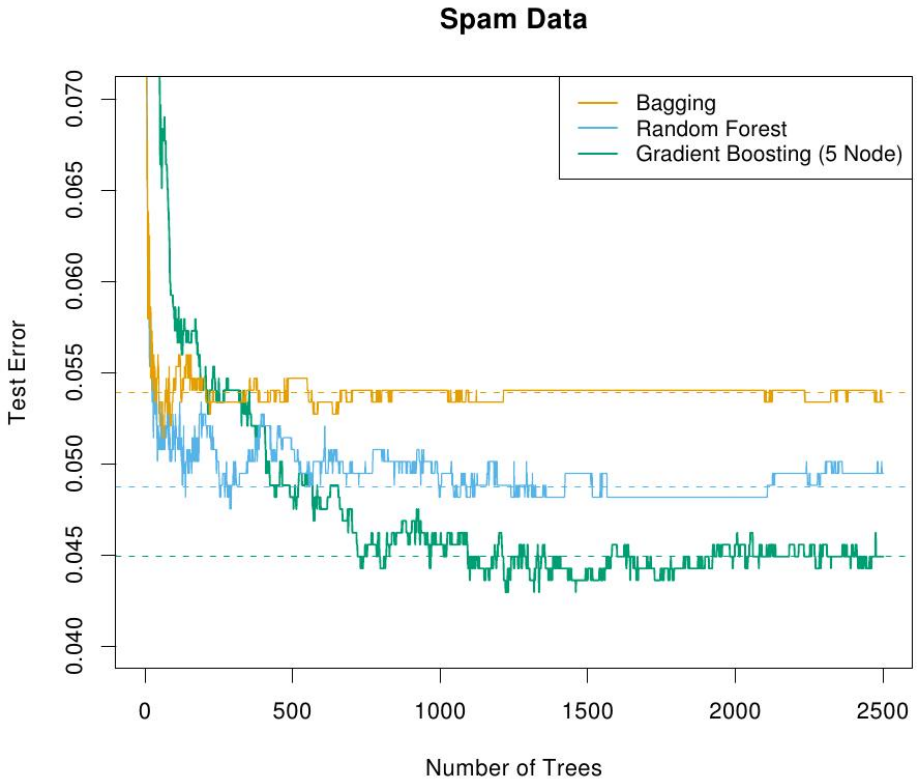
To make a prediction at a new point x:

Regression: $\hat{f}_{\text{rf}}^B(x) = \frac{1}{B}\sum_{b=1}^B T_b(x)$.

Classification: Let $\hat{C}_b(x)$ be the class prediction of the bth random-forest tree. Then $\hat{C}_{\text{rf}}^B(x) = \textit{majority vote}\ \{\hat{C}_b(x)\}_1^B$.

Spam data – comparison

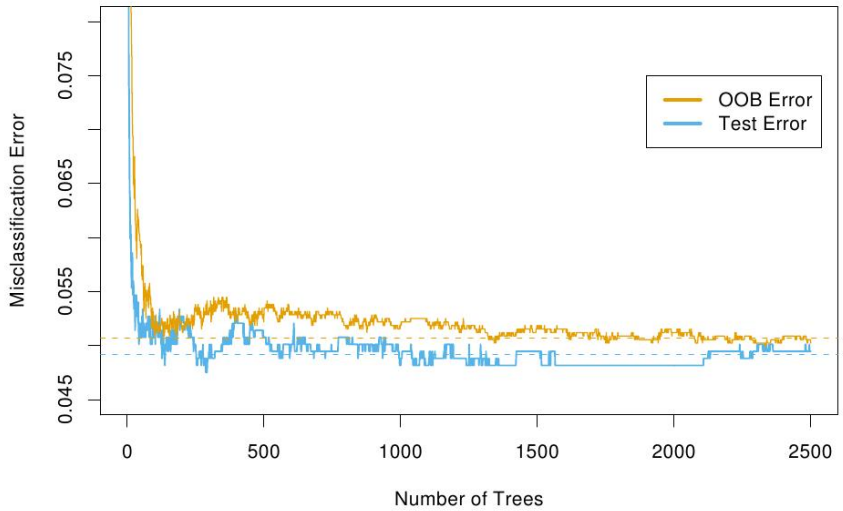
- There is a `randomForest` package in R, maintained by Andy Liaw.
- Random forests do remarkably well, with very little tuning required.
- A random forest classifier achieves 4.88% misclassification error on the spam test data, which compares well with all other methods, and is not significantly worse than gradient boosting at 4.5%. Bagging achieves 5.4% which is significantly worse than either, although still comparable to the additive logistic regression that was clocked at the rate 5.3%.
- In this example the additional randomization helps.



Practical aspects

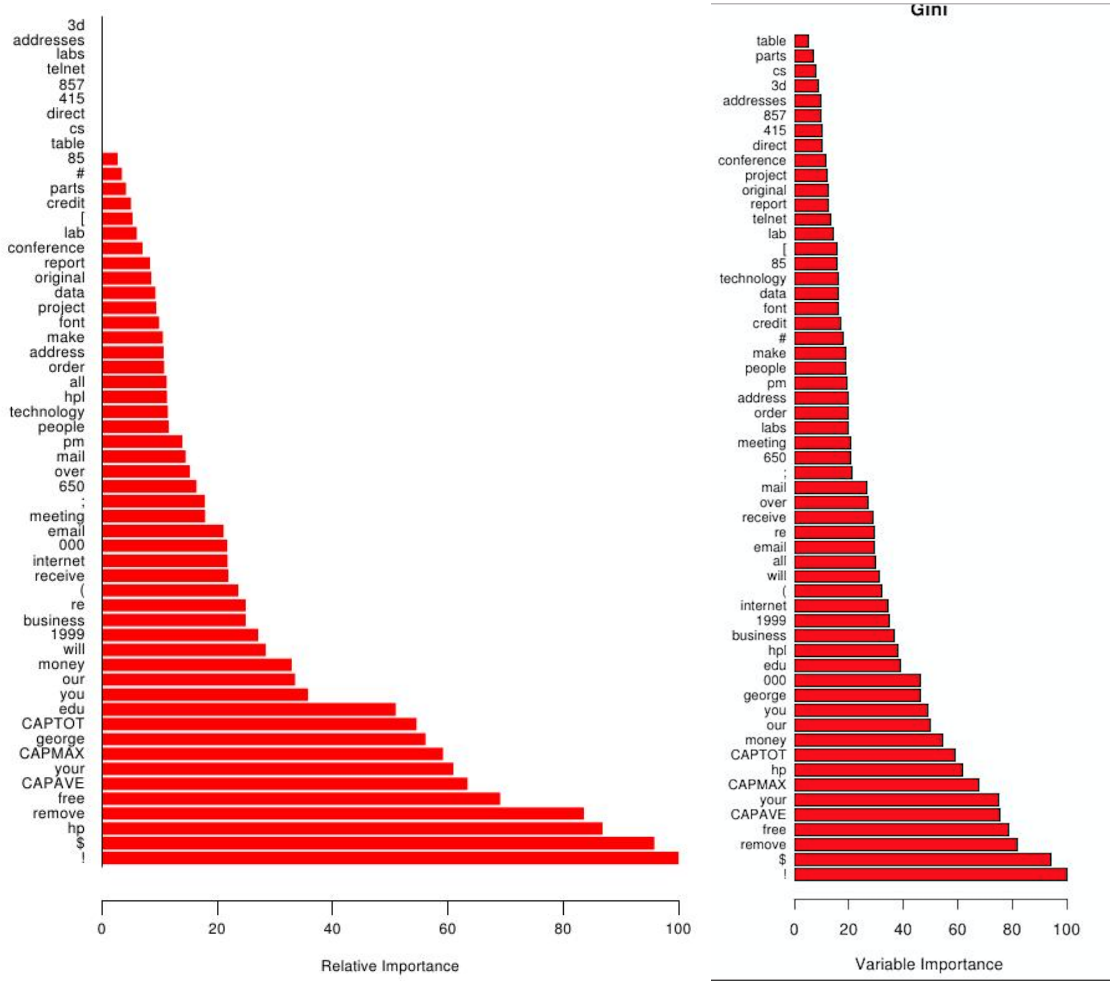
- When used for classification, a random forest obtains a class vote from each tree, and then classifies using majority vote or by averaging probabilities and choosing the class that maximize it.
- When used for regression, the predictions from each tree at a target point x are simply averaged,
- For m the following recommendations were suggested:
  - For classification, the default value for m is $\sqrt{p}$ and the minimum node size is one.
  - For regression, the default value for m is $p/3$ and the minimum node size is five.
- In practice the best values for these parameters will depend on the problem.

Out of bag (OOB) samples – simultaneous cross-validation



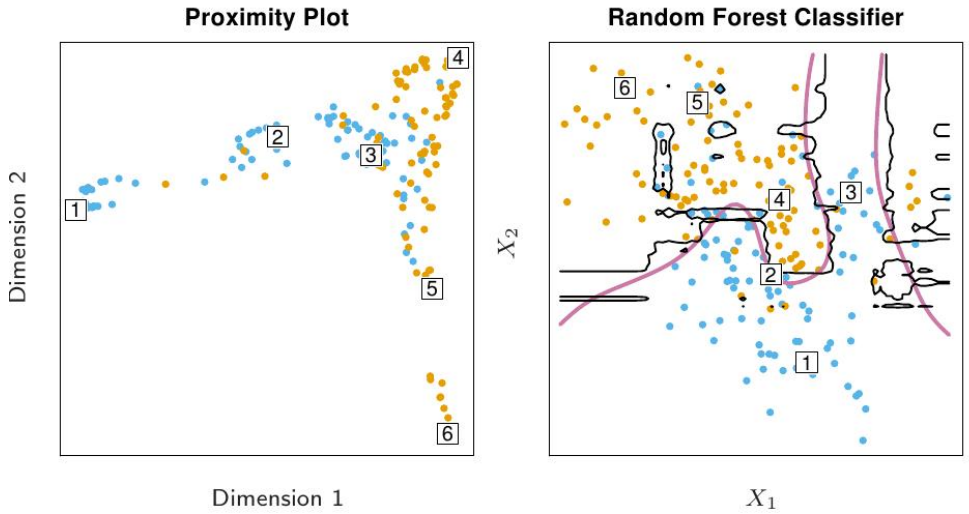
- For each observation x_i, construct its random forest predictor by averaging only those trees that are based on bootstrap samples in which x_i did not appear. For those trees x_i presents itself as 'fresh' observation not used in the predictor.
- Evaluate how many x_i's have been misclassified through the so obtained predictors. This will be the oob misclassification error.
- An oob error estimate is close to that obtained by N -fold cross- validation.
- Hence unlike many other nonlinear estimators, random forests can be fit in one sequence, with **cross-validation being performed along the way**.
- Once the oob error stabilizes, the training can be terminated.
- Figure compares the oob misclassification error for the spam data to the test error. Although 2500 trees are averaged here, it appears from the plot that about 1000 would be sufficient.

Feature importance



- On the left hand side and the right hand side graphs, the importance for boosting tree and the random forest is reported, respectively.

Proximity plots – a visualization technique



- In growing a random forest, an *NN* proximity matrix is accumulated for the training data.
- For every tree, any pair of oob observations sharing a terminal node has their proximity increased by one.
- This proximity matrix is then represented in two dimensions using multidimen- sional scaling.
- The proximity plot gives an indication of which observations are effectively close together in the eyes of the random forest classifier.

Multidimensional Scaling

- A method of representing distances between objects in low dimension
- Let d_{ij}, $i, j = 1, \ldots, N$ be a matrix of distances between objects
- We would like to represent these objects by points x_i, $i = 1, \ldots, N$ in some small dimension k in a way that the Euclidean distances between these points $\|x_i - x_j\|$ approximate d_{ij}.
- We search points x_i's so that the two matrices

$$\mathbf{D} = \left[d_{ij}\right], \quad \mathbf{R} = \left[\|x_i - x_j\|\right]$$

  are in a certain way close.
- For example we can search for x_i's that minimize the sum of squared difference of the entries

$$\sum_{i,j} (d_{ij} - \|x_i - x_j\|)^2$$

- Other measures of closeness of such matrices can be used as well and numerical algorithms are used for construction of x_i's.
- In the previous slide we had the matrix of closeness between points used for construction of the trees. It was represented by points x_i's in two dimensional space ($k = 2$).

Illustrative example: correlated multivariate data

We want to construct the bivariate data that are correlated both between the two variables as well as between samples. The data constitute a $N \times 2$ matrix

$$\mathbf{X} = \begin{bmatrix} X_{11} & X_{12} \\ \vdots & \vdots \\ X_{N1} & X_{N2} \end{bmatrix}$$

We want them to be correlated both between rows and between columns. The correlations for matrices of random variables are often presented through the covariance matrix that is obtained for the vector obtained from the matrix $\mathbf{X}$ by stacking columns one at the top of the other which is denoted by $vec(\mathbf{X})$.

$$vec(\mathbf{X}) = \begin{bmatrix} X_{11} \\ \vdots \\ X_{N1} \\ X_{12} \\ \vdots \\ X_{N2} \end{bmatrix}$$

Let $\mathbf{Z}$, $\mathbf{Z}_N$, $\mathbf{Z}_2$ be $N \times 2$, $N \times 1$, and 1×2 matrices of iid standard normal variables. Let $\mathbf{1}_{.2}$ be two dimensional row of ones and $\mathbf{1}_{N.}$ be N dimensional column of ones. We define our data through

$$\mathbf{X} = \sqrt{1-\rho^2}\sqrt{1-\rho_0^2}\mathbf{Z} + \rho_0 \mathbf{Z}_N \mathbf{1}_{.2} + \rho \mathbf{1}_{N.} \mathbf{Z}_2 .$$

One can see that ρ_0 introduces correlation between columns in $\mathbf{X}$ and ρ between rows.

Correlation matrix

One can verify that the correlation for this matrix variable is

$$Cov(\mathbf{X}) = \begin{bmatrix} \begin{bmatrix} 1 & \rho^2 & \dots & \rho^2 \\ \vdots & \vdots & \vdots & \vdots \\ \rho^2 & \rho^2 & \dots & 1 \end{bmatrix} & \begin{bmatrix} \rho_0^2\sqrt{1-\rho^2} & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & \rho_0^2\sqrt{1-\rho^2} \end{bmatrix} \\ \begin{bmatrix} \rho_0^2\sqrt{1-\rho^2} & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & \rho_0^2\sqrt{1-\rho^2} \end{bmatrix} & \begin{bmatrix} 1 & \rho^2 & \dots & \rho^2 \\ \vdots & \vdots & \vdots & \vdots \\ \rho^2 & \rho^2 & \dots & 1 \end{bmatrix} \end{bmatrix}$$

We see from the off-diagonal blocks that correlation within rows (between columns) is small if correlation between rows is high due to $\sqrt{1-\rho^2}$.

Numerical study – correlated data

```
N=20 #Sample size
d=2   #Dimension of the predictors
rho=0.85  #Correlation 1
rho0=0.2  #Correlation 2
B=10000  #Bootstrap sample size

#Data two dimensional and size N but correlated both within columns
#and within rows

Z=matrix(rnorm(2*N),nrow=N)
ZN=rnorm(N)
Z2=rnorm(2)
X=sqrt(1-rho^2)*sqrt(1-rho0^2)*Z+rho0*ZN%*%t(rep(1,2))+rho*as.matrix(rep(1,N))%*%Z2

 round(X[,1],1)
# -1.2 -1.4 -0.8 -1.1 -2.1 -1.5 -1.8 -2.0 -2.1 -1.1 -0.9
#   -0.9 -1.9 -0.9  0.3 -0.7 -2.2 -1.1 -0.3 -2.1

round(X[,2],1)
#-0.2 -0.7 -0.4  0.3  0.3  1.1  0.1 -0.2  0.3  0.6  0.7 -0.8
#  0.4  0.1  0.3  0.4 -0.2 -0.3  0.0  0.9
```

Numerical study, cont. – estimating mean

```
#Estimate of the common mean (which is zero)

mean(X[,1])+mean(X[,2]) #[1] -1.139279

#Bootstrap estimate

Bmean=vector('numeric',B)
for(i in 1:B)
{
  BN=sample(1:N,size=N, rep=TRUE)
  BX1=X[BN,1]
  BX2=X[BN,2]
  Bmean[i]=mean(BX1)+mean(BX2)
}

mean(Bmean) #[1] -1.140948

#Bootstrapping coordinates as in random forest
Bmean2=vector('numeric',B)
for(i in 1:B)
{
  BN=sample(1:N,size=N, rep=TRUE)
  delta=rbinom(1,1,0.5)
  BX1=delta*X[BN,1]
  BX2=(1-delta)*X[BN,2]
  Bmean2[i]=mean(BX1)+mean(BX2)
}

  mean(Bmean2) #[1] -0.5657908
```

Numerical study, cont. – Monte Carlo study

```
MC=30                                  #Monte Carlo sample size
E1=vector("numeric",MC) #MC-values of the bootstrap estimates
E2=E1                                   #MC-values of the random forest type estimates

for(j in 1:MC) #MC loop
{ Z=matrix(rnorm(2*N),nrow=N)
  ZN=rnorm(N)
  Z2=rnorm(2)
  X=sqrt(1-rho^2)*sqrt(1-rho0^2)*Z+rho0*ZN%*%t(rep(1,2))+rho*as.matrix(rep(1,N))%*%Z2

  for(i in 1:B)      #Bootstrap loop
  { BN=sample(1:N,size=N, rep=TRUE)
    BX1=X[BN,1]
    BX2=X[BN,2]
    Bmean[i]=mean(BX1)+mean(BX2)
  }
  E1[j]=mean(Bmean)

  for(i in 1:B) #Random forest loop
  { BN=sample(1:N,size=N, rep=TRUE)
    delta=rbinom(1,1,0.5)
    BX1=delta*X[BN,1]
    BX2=(1-delta)*X[BN,2]
    Bmean2[i]=mean(BX1)+mean(BX2)
  }
  E2[j]=mean(Bmean2)
}
```

Results of the Monte Carlo study, means and variances of the estimators:

```
mean(E1) #[1]   0.13517
mean(E2) #[1] 0.06707397
var(E1) #[1] 0.8352297
var(E2) #[1] 0.2098515
```