Lecture 13: Ensemble Methods

Applied Multivariate Analysis

Math 570, Fall 2014

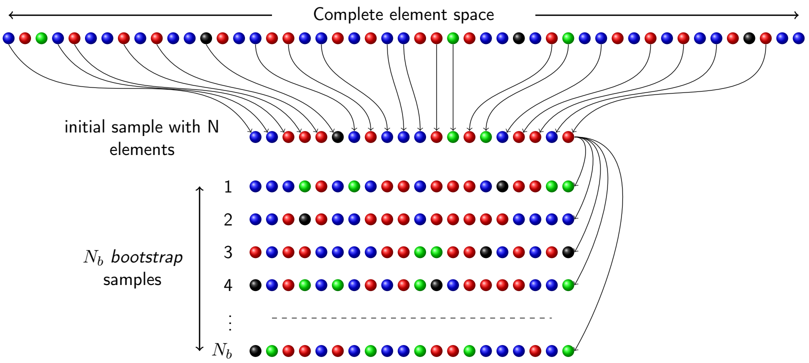
Xingye Qiao
Department of Mathematical Sciences
Binghamton University

E-mail: qiao@math.binghamton.edu

Outline

The next section would be . . . . . .

The next subsection would be ......



Theorem (B. Efron, Ann. Statist. 1979)

When N tend to infinity, the distribution of average values computed from bootstrap samples is equal to the distribution of average values obtained from ALL samples with N elements which can be constructed from the complete space. Thus the width of the distribution gives an evaluation of the sample quality.

Figure: Courtesy of www.texample.net

Bootstrap sample

- Given data $\boldsymbol{X} = \{X_1, X_2, \ldots, X_n\}$, a bootstrap sample is obtained by sampling with replacement n observations from $\{X_1, X_2, \ldots, X_n\}$
- Each observation in the original data may be drawn multiple times or not drawn at all.
- Often one need to know the sampling distribution of a statistic. It can be estimated if we have many $\boldsymbol{X}$ from the true underlying distribution. However, since we do not have the true underlying distribution, we generate many bootstrap samples which are actually from the so-called empirical distribution. Ideally, for large n, the empirical distribution should be close to the true distribution.
- Next, we review some basics of the theoretical background of bootstrap.

Plug-in principal

- $X_1, \ldots, X_n$ follow distribution F.
- We define the empirical distribution function F_n as one which assigns $1/n$ probability mass to any observed value in the current data x_i $(i = 1, \ldots, n)$.
- Many parameters θ of the distribution F can be written as a function of F. Hence we write $\theta = t(F)$
  - For example, if θ is the mean of this distribution, then $\theta = \int xf(x)dx = \int xdF(x)$
  - In a much broader sense, the sampling distribution of a statistic is also a parameter.
- The plug-in principal says we can estimate $\theta = t(F)$ by $\widehat{\theta} = t(F_n)$
  - For example, in the example above, $t(F_n) = \int xdF_n(x) = \sum_{i=1}^{n} \frac{1}{n} x_i = \frac{1}{n} \sum_{i=1}^{n} x_i$; the sample mean of the data, is actually a plug-in estimator.

Another example is the variance of the distribution.

- We write $\sigma^2 = t(F) = \int x^2 dF(x) - (\int x dF(x))^2$
- Replacing F by F_n, we have

$$
\begin{aligned}
t(F_n) &= \int x^2 dF_n(x) - (\int x dF_n(x))^2 \\
&= \sum_{i=1}^{n} x_i^2 \frac{1}{n} - \left(\sum_{i=1}^{n} x_i \frac{1}{n} \right)^2 \\
&= \frac{1}{n} \sum_{i=1}^{n} x_i^2 - (\overline{x})^2 \\
&= \frac{1}{n} \sum_{i=1}^{n} (x_i - \overline{x})^2
\end{aligned}
$$

But this is the common sample variance estimator (up to a multiplicative constant $\frac{n}{n-1}$) in any undergraduate statistical textbook.

Bootstrap estimate for the standard error of an estimator

- For a *general* estimator $\widehat{\theta}$ that is calculated based on sample $\boldsymbol{X}$, we write $\widehat{\theta} = s(\boldsymbol{X})$. Note that it could be a plug-in estimator, but does not have to be.
- We seek to estimate a parameter of the distribution of $\widehat{\theta} = s(\boldsymbol{X})$, especially, the standard error (root of the variance) of $\widehat{\theta}$, i.e., $se(\widehat{\theta})$.
- Note that although $\widehat{\theta} = s(\boldsymbol{X})$ has its own distribution function, it is still a function of $\boldsymbol{X}$; hence $se(\widehat{\theta})$ is still a function of F. To stress this, we denote it as $se_F(\widehat{\theta}) = t(F)$.

Bootstrap estimate for the standard error of an estimator

- A counterpart of $\widehat{\theta}$ is $\widehat{\theta}^*$ defined as $\widehat{\theta}^* = s(\boldsymbol{X}^*)$ where $\boldsymbol{X}^*$ is a bootstrap sample of size n
  - Note that $X_i \sim F$ and $X_i^* \sim F_n$
- According to previous pages, the plug-in estimator for $se_F(\widehat{\theta}) = t(F)$ is $se_{F_n}(\widehat{\theta}^*) := t(F_n)$, that is, to replace the true unknown distribution F (where $\boldsymbol{X}$ comes from) by the empirical distribution F_n (where $\boldsymbol{X}^*$ comes from).
- To implement this, we can hold $\boldsymbol{X}$ fixed and generate a bootstrap sample $\boldsymbol{X}^*$ from $\boldsymbol{X}$, and obtain the point estimator $\widehat{\theta}^*$. Claim: The standard deviation of $\widehat{\theta}^*$ (conditioning on $\boldsymbol{X}$) is the same as the plug-in $se_{F_n}(\widehat{\theta}^*) = t(F_n)$ above. This is perhaps less obvious. See next page.

Side note: more explanation on last claim

- The true variance of $s(\boldsymbol{X})$ is $\int (s(\boldsymbol{x}) - \mathrm{E}_F(s(\boldsymbol{X})))^2 dF(x_1) \dots dF(x_n)$
- The plug-in estimator of the true variance of $s(\boldsymbol{X})$ is

$$\sum_{(x_1^*, x_2^*, \dots, x_n^*)} \left(\frac{1}{n}\right)^n [s(x_1^*, x_2^*, \dots, x_n^*) - \mathrm{E}_{F_n} s(\boldsymbol{X}^*)]^2$$

$$= \sum_{(x_1^*, x_2^*, \dots, x_n^*)} \left(\frac{1}{n}\right)^n [\widehat{\theta}^* - \mathrm{E}_{F_n} \widehat{\theta}^*]^2$$

  Note that the probability mass at $(x_1^*, x_2^*, \dots, x_n^*)$ is $(1/n)^n$
- The last line above is actually the standard deviation of $\widehat{\theta}^*$ (conditioning on $\boldsymbol{X}$)
- Note that the randomness of $\widehat{\theta}^*$ is only coming from how the bootstrap set is sampled, since we have had $\boldsymbol{X}$ and hence F_n fixed.

- Unfortunately, except for the simple case where $\widehat{\theta} = \overline{X}$, the explicit formula for $t(\cdot)$ is difficult to identify.
- Fortunately, we can use computer to approximate $se_{F_n}(\widehat{\theta}^*)$.

Bootstrap Estimate for Standard Error

1. Draw B bootstrap samples $\boldsymbol{X}_1^*, \ldots, \boldsymbol{X}_B^*$, where $X_{b,i} \sim F_n$
2. Calculate $\widehat{\theta}_b^* = s(\boldsymbol{X}_b^*)$ for each bootstrap sample
3. Approximate $se_{F_n}(\widehat{\theta}^*) = \sqrt{\mathrm{Var}_{F_n}\widehat{\theta}^*}$ by sample standard deviation se_B of $\left\{\widehat{\theta}_b^*, b = 1, \ldots, B\right\}$,

$$se_B := \sqrt{\frac{1}{B-1}\sum_{b=1}^{B}\left(\widehat{\theta}_b^* - \overline{\widehat{\theta}^*}\right)^2}$$

- $se_{F_n}(\widehat{\theta}^*)$ is called the *ideal* bootstrap estimate for $se_F(\widehat{\theta})$
- se_B is the actual bootstrap estimate that we use to approximate $se_{F_n}(\widehat{\theta}^*)$
- There are two levels of approximations going on:
  1. $se_{F_n}(\widehat{\theta}^*)$ is a plug-in estimator for $se_F(\widehat{\theta})$.
  2. se_B is a "sample" standard deviation approximating the "population" standard deviation for $\widehat{\theta}^*$ (that is $se_{F_n}(\widehat{\theta}^*)$)
- Here the "population" is the population of $\widehat{\theta}^* = s(\boldsymbol{X}^*)$. Note that even $\boldsymbol{X}$ is given (i.e. given F_n), $\widehat{\theta}^*$ is still random (with up to $\binom{2n-1}{n}$ possible outcomes).
- Each bootstrap point estimate $\widehat{\theta}^*_b = s(\boldsymbol{X}^*_b)$, based on the bootstrap sample $\boldsymbol{X}^*_b$, is an observation of $\widehat{\theta}^*$
- The first approximation error vanishes as $n \to \infty$
- The second approximation error vanishes as $B \to \infty$. Because we can already bootstrap sample many times, usually we can ignore the second approximation error.

- There are $\binom{2n-1}{n}$ distinct bootstrap samples, corresponding to up to $\binom{2n-1}{n}$ distinct possible values of $\widehat{\theta}_b^* = s(\boldsymbol{X}_b^*)$
- Technically, one can calculate

  $$\mathrm{E}_{F_n}(s(\boldsymbol{X}^*)) = \sum_{\ell} w_\ell s(\boldsymbol{X}_\ell^*)$$

  and

  $$se_{F_n}(\widehat{\theta}^*) = se_{F_n}(s(\boldsymbol{X}^*)) = \sqrt{\sum_{\ell} w_\ell \left[s(\boldsymbol{X}_\ell^*) - \mathrm{E}_{F_n}(s(\boldsymbol{X}^*))\right]^2}$$

  where w_ℓ is the probability for the ℓth distinct bootstrap sample. But such calculation is not realistic due to the very large number of $\binom{2n-1}{n}$.

Incidentally, w_ℓ is the probability mass function for a multinomial distribution.

Example: $\widehat{\theta} = \overline{X}$

- $se_F(\overline{X}) = \frac{\sqrt{\sigma^2(X_i)}}{\sqrt{n}}$ (*)
- By plug-in estimation of $\sigma^2(X_i)$ which is $\widehat{\sigma}^2(\{X_i\})$, $se_{F_n}(\overline{X}^*) = \frac{\sqrt{\widehat{\sigma}^2(\{X_i\})}}{\sqrt{n}}$
- If we do generate a new bootstrap sample $\boldsymbol{X}^*$ and recalculate $\overline{X}^*$ based on $\boldsymbol{X}^*$, then the standard deviation of $\overline{X}^*$ (under F_n, i.e., given $\boldsymbol{X}$) would really be $\frac{\sqrt{\widehat{\sigma}^2(\{X_i\})}}{\sqrt{n}}$ (See next page.)
- This is an example where we can directly calculate the plug-in estimate for $se_F(\overline{X})$. Otherwise, neat forms like (*) do not exist; in these cases, we could generate many $\boldsymbol{X}_b^*$'s and approximate the standard deviation of $\overline{X}^*$ by the sample standard deviation of $\overline{X}_b^*$'s.

Side note

$$\mathrm{Var}(\overline{X}^* \mid \boldsymbol{X} = (x_i)) = \frac{\widehat{\sigma}^2(\{x_i\})}{n} \text{ ??}$$

$\overline{X}^* := \frac{1}{n}(x_1 Z_1 + x_2 Z_2 + \cdots + x_n Z_n)$ where $(Z_1, \ldots, Z_n)$ follows multinomial distribution with parameters n and $(\frac{1}{n}, \frac{1}{n}, \ldots, \frac{1}{n})$. Note that $\mathrm{Var}(Z_j) = n\frac{1}{n}(1 - \frac{1}{n}) = 1 - \frac{1}{n}$ and $\mathrm{Cov}(Z_i, Z_j) = -n\frac{1}{n}\frac{1}{n} = -\frac{1}{n}$. Thus treating (x_i) as fixed, we have

$$\begin{aligned}
\mathrm{Var}(\overline{X}^*) &= \mathrm{Var}(\frac{1}{n}\sum_{j=1}^n x_j Z_j) = \mathrm{Cov}(\frac{1}{n}\sum_{i=1}^n x_i Z_i, \frac{1}{n}\sum_{j=1}^n x_j Z_j) \\
&= \sum_{i=1}^n \sum_{j=1}^n \frac{x_i x_j}{n^2}\mathrm{Cov}(Z_i, Z_j) \\
&= \sum_{i=1}^n \frac{x_i^2}{n^2} - \frac{1}{n}\left[\sum_{i=1}^n \sum_{j=1}^n \frac{x_i x_j}{n^2}\right] = \frac{1}{n}\left[\sum_{i=1}^n \frac{x_i^2}{n} - \overline{x}^2\right] = \frac{\widehat{\sigma}^2(\{x_i\})}{n}
\end{aligned}$$

How large should B be?

- The approach of approximating $se_{F_n}(\widehat{\theta}^*)$ by sample standard deviation se_B is essentially to calculate a complicated expectation by averaging Monte Carlo simulations. A canonical example is to approximate $\mathrm{E}X = \int xf(x)dx$ by $\frac{1}{n}\sum_{i=1}^{n} X_i$ where $X_i \sim f(x)$
- This is sometimes called Monte Carlo integration. It is very convenient when the integration involved is very complicated but we have the capacity to generate many observations from the underlying distribution. The foundation of the Monte Carlo integration is law of large number.
- According to Efron and Tibshirani, $B = 25$ is sometimes enough to have a good estimator for $se_{F_n}(\widehat{\theta}^*)$, and rarely is $B > 200$ needed.

Beyond $\overline{X}$ and Beyond Standard Error

- For more complicated $\widehat{\theta}$ than $\overline{X}$, estimating the standard error of $\widehat{\theta}$ boils down to calculating the sample standard deviation of the $\widehat{\theta}^*$ based on B bootstrapped samples.
- More or less like a black box. Need not to know what's going on inside. Bootstrap sampling $\rightarrow$ Calculate $\widehat{\theta}^*$ $\rightarrow$ Calculate sample standard deviation. Done.
- Core idea: the variance of $\widehat{\theta}$ and the variance of $\widehat{\theta}^*$ ought to close (and the larger the n is, the closer they are), supported by the closeness of F_n and F.
- We can do more than just estimating the standard error.

Confidence interval

Traditional method:

- Use $\frac{\widehat{\theta}-\theta}{se(\widehat{\theta})}$ as a pivotal. If it has normal or t distribution, then with probability $(1-\alpha)$,

$$t_{\alpha/2} \leq \frac{\widehat{\theta}-\theta}{se(\widehat{\theta})} \leq t_{1-\alpha/2}$$

- Inverting the pivotal, we have, with probability $(1-\alpha)$

$$\widehat{\theta} - t_{1-\alpha/2} se(\widehat{\theta}) \leq \theta \leq \widehat{\theta} - t_{\alpha/2} se(\widehat{\theta})$$

Note that $t_{\alpha/2}$ is negative.

Bootstrap confidence intervals

It is possible that normal assumptions (or t assumptions) are not true. In these cases, some model-free methods are desired.

- Bootstrap-t interval
- Percentile interval
- Basic bootstrap interval

Following Efron and Tibshiran 1993, *An Introduction to the Bootstrap*.

Percentile-t interval

- Idea: $\frac{\widehat{\theta}-\theta}{se(\widehat{\theta})}$ and $\frac{\widehat{\theta}^*-\widehat{\theta}}{\widehat{se}(\widehat{\theta}^*)}|F_n$ should have very close distribution. Hence, use the quantiles of the latter, and invert the former.
- Let $Z_b = \frac{\widehat{\theta}_b^*-\widehat{\theta}}{\widehat{se}(\widehat{\theta}_b^*)}$, and find the $\alpha/2$ and $1-\alpha/2$ sample percentiles for $\{Z_b, b = 1, \ldots, B\}$, namely, $z_{\alpha/2}$ and $z_{1-\alpha/2}$.

The confidence interval is

$$\widehat{\theta} - z^*_{1-\alpha/2} se(\widehat{\theta}) \leq \theta \leq \widehat{\theta} - z^*_{\alpha/2} se(\widehat{\theta})$$

A simpler interval

- Idea: $\widehat{\theta} - \theta$ and $\widehat{\theta}^* - \widehat{\theta} | F_n$ should have same/close distribution. This is a simplified idea where we choose to let $se(\widehat{\theta})$ be independent of the data in the Percentile t-method.
- Instead of first finding the percentiles of the former and inverting it, we find the sample percentiles of the latter and invert the former.
- We should have with probability $1 - \alpha$

$$\widehat{\theta}^*_{b,\alpha/2} - \widehat{\theta} \leq \widehat{\theta} - \theta \leq \widehat{\theta}^*_{b,1-\alpha/2} - \widehat{\theta}$$

After inverting it, we have

$$2\widehat{\theta} - \widehat{\theta}^*_{b,1-\alpha/2} \leq \theta \leq 2\widehat{\theta} - \widehat{\theta}^*_{b,\alpha/2}$$

Percentile interval

- If the normal assumption is true, then $(\widehat{\theta} - z_{1-\alpha/2} se(\widehat{\theta}), \widehat{\theta} - z_{\alpha/2} se(\widehat{\theta}))$ is the confidence interval for θ
- If we have $\widehat{\theta}^* \sim N(\widehat{\theta}, se(\widehat{\theta})^2)$, then $(\widehat{\theta} - z_{1-\alpha/2} se(\widehat{\theta}), \widehat{\theta} - z_{\alpha/2} se(\widehat{\theta}))$ is also the $(\alpha/2)$ and $(1 - \alpha/2)$ percentiles of $\widehat{\theta}^*$
- Hence we avoid working hard to get $se(\widehat{\theta})$ and all that, and directly estimate them by the sample percentiles of $\left\{\widehat{\theta}^*_b, b = 1, \ldots, B\right\}$

Remarks on Bootstrap

- Poor-men's (but rich enough to afford a computer) statistical inference. Very little effort is needed.
- People who know little about statistics can use it.
- Can fail sometimes,
  1. When $t(F_n)$ is not a good estimate for $t(F)$.
  2. Dimension is high.
  3. Time-series data (not iid.)
- A paradox: the motive to use bootstrap is to deal with complex t, but sometimes if t is too complex then bootstrap would fail as well.

The next subsection would be . . . . . .

Subsampling

- Subsampling can be viewed as a variant of bootstrap.
- Instead of sampling with replacement, we do sampling without replacement to obtain a subset with b observations of the original data. Usually $b \ll n$ such that $b/n \to 0$, and $b \to \infty$ as $n \to \infty$
- Subsampling can provide confidence intervals that work (i.e., do not fail) with much less strong conditions than bootstrap.
- The proof of the consistency for subsampling involves some concentration inequality, but that for boostrap involves very 'high-tech' empirical process theory.

Remark

There are also some catches for subsampling.

- Need to choose b wisely.
- The theoretical guarantee is only asymptotic. Performance in finite sample cases is more sensitive to b
- For both bootstrap and subsampling: too computationally intensive.

The next subsection would be . . . . . .

Bagging

- In addition to statistical inference, bootstrapped samples can also be utilized to improve existing classification or regression methods.
- Bagging = Bootstrap aggregating.
- Proposed by Leo Breiman in 1994 to improve the classification by combining classification results of many randomly generated training sets

Bagging classifier

- Suppose a method provides a prediction $\widehat{f}(\boldsymbol{x})$ at input $\boldsymbol{x}$ based on sample $\boldsymbol{X}$
- The bagging prediction is

$$\widehat{f}_B(\boldsymbol{x}) = \frac{1}{B}\sum_{b=1}^{B}\widehat{f}_b(\boldsymbol{x})$$

where $\widehat{f}_b$ is the prediction based on a bootstrapped sample $\boldsymbol{X}_b^*$

- If we view $\widehat{f}(\boldsymbol{x})$ as an estimator, then we would be interested in $\mathrm{E}_F(\widehat{f}(\boldsymbol{x}))$, which is a function of the distribution F, (like $t(F)$ in previous section)
- In this case, the bagging classifier $\widehat{f}_B(\boldsymbol{x})$ is nothing but an approximation to the ideal bootstrap estimate $\mathrm{E}_{F_n}(\widehat{f}^*(\boldsymbol{x}))$
- Hence the goal of bagging can be viewed as to use a plug-in estimate to approximate $\mathrm{E}(\widehat{f}(\boldsymbol{x}))$
- Neither $\mathrm{E}_F(\widehat{f}(\boldsymbol{x}))$ nor $\mathrm{E}_{F_n}(\widehat{f}(\boldsymbol{x}))$ can be directly calculated. As before, we have $\widehat{f}_B(\boldsymbol{x}) \xrightarrow{p} \mathrm{E}_{F_n}(\widehat{f}^*(\boldsymbol{x}))$ given data (or given F_n) and $\mathrm{E}_{F_n}(\widehat{f}^*(\boldsymbol{x}))$ is supposed to be close to the true mean of $\widehat{f}(\boldsymbol{x})$

Why Bagging Works?

$$\widehat{f}_B(\boldsymbol{x}) \approx E_{F_n}[\widehat{f}(\boldsymbol{x})] \approx E_F[\widehat{f}(\boldsymbol{x})]$$

- A very well known result regarding mean squared error for estimation is that

$$MSE = Variance + Bias^2$$

  In the current setting, this is

$$E_F[\widehat{f}(\boldsymbol{x}) - y]^2 = E_F[\widehat{f}(\boldsymbol{x}) - \mathrm{E}_F(\widehat{f}(\boldsymbol{x}))]^2 + (\mathrm{E}_F(\widehat{f}(\boldsymbol{x})) - y)^2$$

- By using the bagging estimate $\widehat{f}_B(\boldsymbol{x})$ in lieu of $\widehat{f}(\boldsymbol{x})$, we hope to make the variance part $E_F[\widehat{f}(\boldsymbol{x}) - \mathrm{E}_F(\widehat{f}(\boldsymbol{x}))]^2$ almost zero.
- Hence bagging improves MSE by reducing the variance.

Bagging for Classification

Bagging was initially invented to solve classification problem. Two approaches:

- If the basic classifier itself is a plug-in classifier (not the plug-in method that we introduced in bootstrap), which works by first estimating $\eta_j(\boldsymbol{x})$ by $\widehat{\eta}_j(\boldsymbol{x})$ and then use classification rule

$$\widehat{\phi}(\boldsymbol{x}) := \underset{j=1,\dots,K}{\operatorname{argmax}} \widehat{\eta}_j(\boldsymbol{x}),$$

  then we can bagging the bootstrap estimates $\eta_j(\boldsymbol{x})$ to obtain

$$\widehat{\eta}_{j,B}(\boldsymbol{x}) = \frac{1}{B}\sum_{b=1}^{B} \widehat{\eta}_{j,b}(\boldsymbol{x})$$

  then use

$$\widehat{\phi}_B(\boldsymbol{x}) = \underset{j}{\operatorname{argmax}} \widehat{\eta}_{j,B}(\boldsymbol{x})$$

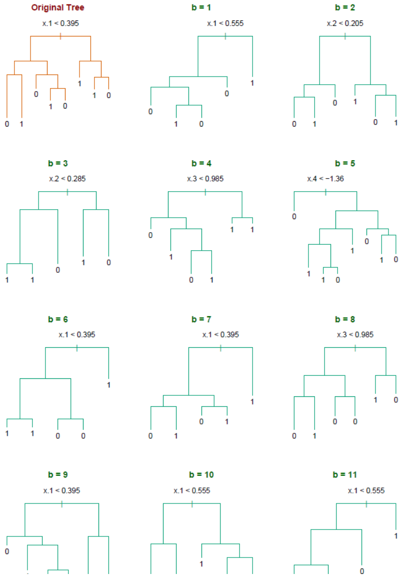
- However, except for a few classifiers such as kNN, most approaches do not work like that. Often, a classifier will report a prediction for $\boldsymbol{x}$ without reporting the estimate for $\eta(\boldsymbol{x})$. For example, SVM, CART, etc.
- In this case, use a majority voting scheme: Let

$$V_{j,B}(\boldsymbol{x}) := \frac{1}{B}\sum_{b=1}^{B} \mathbb{1}\{\widehat{\phi}_b(\boldsymbol{x}) = j\}$$

  then we just use

$$\widehat{\phi}_B(\boldsymbol{x}) := \operatorname*{argmax}_j V_{j,B}(\boldsymbol{x})$$

- In other words, we let the B classifiers to cast votes and choose the class with the most votes.



Original Tree
x.1 < 0.395
b = 1
x.1 < 0.555
b = 2
x.2 < 0.205
b = 3
x.2 < 0.285
b = 4
x.3 < 0.985
b = 5
x.4 < −1.36
b = 6
x.1 < 0.395
b = 7
x.1 < 0.395
b = 8
x.3 < 0.985
b = 9
x.1 < 0.395
b = 10
x.1 < 0.555
b = 11
x.1 < 0.555

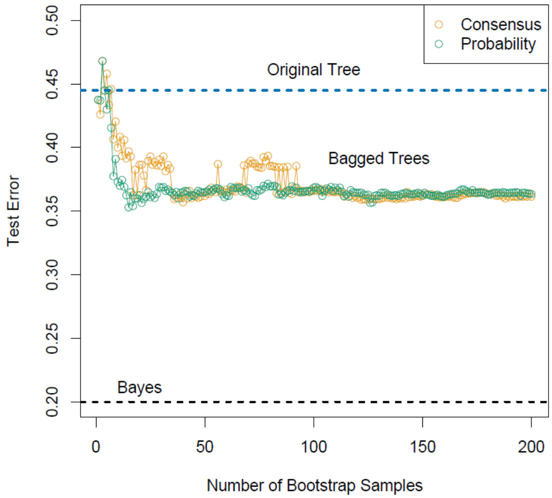


Figure: Consensus – majority voting; Probability – bagging $\eta(\boldsymbol{x})$

- The MSE argument of how bagging works does not apply for classification problem.
- MSE uses squared loss. Classification uses 0-1 loss and in that case the decomposition to variance and bias^2 does not exist.
- "Wisdom of the crowd" argument:

  In binary classification, if there are B independent voters and each would classify $Y = 1$ correctly with probability $1 - e$ and the misclassification rate $e < 0.5$, then $Z = \sum_{b=1}^{B} \widehat{\phi}_b(\boldsymbol{x}) \sim \mathit{Binomial}(B, 1-e)$ and $\frac{1}{B}\sum_{b=1}^{B} \widehat{\phi}_b(\boldsymbol{x}) \xrightarrow{p} 1 - e > 0.5$. Hence, the misclassification rate of the bagged classifier is $P(\frac{1}{B}\sum_{b=1}^{B} \widehat{\phi}_b(\boldsymbol{x}) < 0.5) \to 0$.

A catch: the bootstrap classifiers are not independent at all.

Random forest, however, adds in additional randomness to the bootstrap classifiers which makes them less independent.

Remarks

- Bagging does not improve the estimator if it is only a linear function of the data. It would only have some effect if the estimator is of a nonlinear nature of the data.
- Bagging does not improve an estimator/classifier if it is already very stable; may even worsen it.
- Bagging does not improve a classifier if it is a bad classifier (generalization error > 0.5 in binary classification case; worse than random guessing)
- Bagging in some sense expands the basis of the model space. But examples show that it has not done enough. In contrast, boosting can do a good job in terms of expending model space.

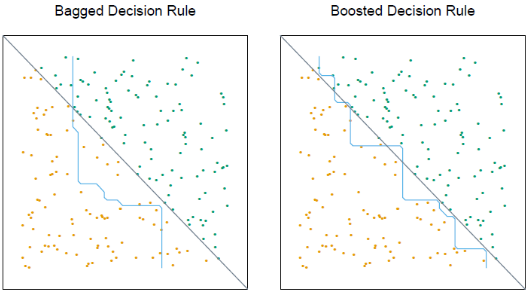


FIGURE 8.12. *Data with two features and two classes, separated by a linear boundary. (Left panel:) Decision boundary estimated from bagging the decision rule from a single split, axis-oriented classifier. (Right panel:) Decision boundary from boosting the decision rule of the same classifier. The test error rates are* 0.166, *and* 0.065, *respectively. Boosting is described in Chapter 10.*

Each basic classifier is just a stump (tree with only one split). The basic classifier is too weak and the model space needs to be expended much aggressively. Next two sections, introduce Boosting and Random Forest respectively.

The next section would be . . . . . .

The next subsection would be . . . . . .

Boosting

- To enhance accuracy of “weak” learner(s)
- Create a strong learner by substantially “boosting” the performance of each single weak learner.
- Focus on binary classification
- Here each weak learner must be slightly better than random guessing (error rate < 0.5).
- First really useful Boosting algorithm: Schapire (1990), Freund (1995), and **Schapire and Freund (1997)**

$\textsc{AdaBoost}$ = Adaptive Boosting.

AdaBoost.M1

We introduce AdaBoost.M1 in the next slide. We follow the introduction in ESL (Chapter 10). The one in Izenman can be specialized to the case we will introduce here.

- Input: $\mathcal{D} := \{(\boldsymbol{x}_i, y_i), i = 1, \ldots, n\}$ where $y_i \in \{+1, -1\}$

Core Idea:

- A weaker learner is applied to an adaptively weighted training sample sequentially.
- All resulting models are weighted in the end to form a final model.

Note that there are two types of weightings: one is for the observations; the other for the learners.

AdaBoost.M1

1. Initialize the weights $w_i = 1/n$ for $i = 1, \ldots, n$
2. For $t = 1, \ldots, T$
   (a) Fit a classifier $\widehat{\phi}_t(\boldsymbol{x})$ to the data with weight vector $\{w_i\}$
   (b) Compute the weighted training data prediction error:
   $$E_t := \sum_{i=1}^{n} w_i \mathbb{1}\{y_i \neq \widehat{\phi}_t(\boldsymbol{x}_i)\}$$
   (c) Compute $\alpha_t := \log\left(\frac{1-E_t}{E_t}\right)$
   (d) Update weights: $w_i \leftarrow \text{normalize}\{w_i \cdot e^{\alpha_t \mathbb{1}\{y_i \neq \widehat{\phi}_t(\boldsymbol{x}_i)\}}\}$
3. Final model:
$$\widehat{\phi}_{AdaBoost}(\boldsymbol{x}) := \text{sign}\left\{\sum_{t=1}^{T} \alpha_t \widehat{\phi}_t(\boldsymbol{x})\right\}$$

- The same learner is applied to an adaptively changing training data repeatedly.
  - An early version of the AdaBoost allowed multiple learning algorithms
- $\alpha_t := \log\left(\frac{1-E_t}{E_t}\right)$ can be viewed as a measure of fit for the model $\widehat{\phi}_t(\cdot)$ to the current weighted data. For a perfectly fit data, $\alpha_t \to \infty$.
  - Justify the fact that α_t is used as the weights for model $\widehat{\phi}_t(\cdot)$ within the final model.
- Those misclassified observations are weighted heavier [by a multiplicative factor of $e^{\alpha_t} = \left(\frac{1-E_t}{E_t}\right)$], i.e., emphasizing more on these data in the next round.
  - If $E_t \approx 1/2$, then the magnitude of the change is milder.
- The normalize$\{c_i\}$ step makes the weights sum up to 1.
- Although the weights for those correctly classified observations are not changed at a first glance, their weights become smaller due to normalization.

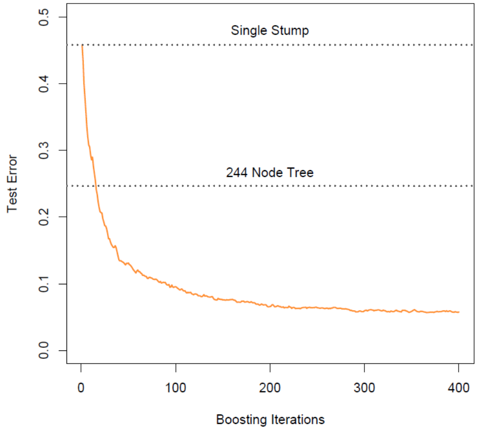


FIGURE 10.2. *Simulated data (10.2): test error rate for boosting with stumps, as a function of the number of iterations. Also shown are the test error rate for a single stump, and a 244-node classification tree.*

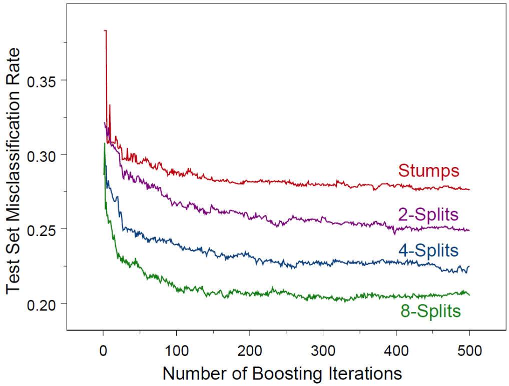


Figure: After 100 iterations, the test set misclassification rates are stablized.

- Empirical evidence has shown that Boosting seems to be resistant to overfit: the testing set misclassification rate will be stable and not go up as iteration grows.
- Some other empirical experiments have shown that the overfitting may appear, if iterations go for a very long time.
- Leo Breiman once speculated that the generalization error of boosting may eventually go to the Bayes error (the theoretically best one can do.)
- Many attempts have been made to try to explain the mysteriously good performance of Boosting.

In the next subsection, we provide a statistical interpretation of boosting. See ESL 10.2 – 10.5 and Izenman 14.3.6 for more details. This view was initially given by Friedman, Hastie and Tibshirani (2000), "Additive logistic regression: a statistical view of boosting," *the Annals of Statistics*

The next subsection would be ......

We first introduce additive model and exponential loss. We then show that AdaBoost is equivalent to fitting an additive model with exponential loss.

- An additive model takes the form

$$f(\boldsymbol{x}) = \sum_{m=1}^{M} \beta_m g(\boldsymbol{x}; \gamma_m)$$

  where β_m is the weight for the mth model and γ_m is the estimated parameters for the mth model.
- One needs to fit β_m and γ_m simultaneously.
- Goal is to minimize $\sum_{i=1}^{n} L(y_i, \sum_{m=1}^{M} \beta_m g(\boldsymbol{x}_i; \gamma_m))$ for an appropriate loss function.
- Such optimization can be quite complicated, even if the loss function has a simple form. One possible way to solve this is to sequentially estimate (β_m, γ_m) for $m = 1, \ldots, M$

Forward stagewise additive model

1. Initialize $f_0(\boldsymbol{x}) = 0$ (no model)
2. For $m = 1$ to M
   (a) Estimate

   $$(\beta_m, \gamma_m) := \operatorname*{argmin}_{\beta,\gamma} \sum_{i=1}^{n} L[y_i, f_{m-1}(\boldsymbol{x}_i) + \beta g(\boldsymbol{x}_i; \gamma)]$$

   (b) Set $f_m(\boldsymbol{x}) = f_{m-1}(\boldsymbol{x}) + \beta_m g(\boldsymbol{x}; \gamma_m)$

- After each iteration the model is enhanced.
- At each iteration, only an optimization involving one pair of (β_m, γ_m) is done.

Exponential loss

- For binary classification problems, we call $u := yf(\boldsymbol{x})$ a functional margin. The larger the margin is, the more confidence we have that the data point $(\boldsymbol{x}, y)$ is correctly classified.
- We consider the exponential loss

$$L(y, f(\boldsymbol{x})) := e^{-yf(\boldsymbol{x})}$$

- The risk function is $\mathrm{E}(e^{-Yf(\boldsymbol{X})})$. In order to find a f that minimizes $\mathrm{E}(e^{-Yf(\boldsymbol{X})})$, we only need to minimize $\mathrm{E}(e^{-Yf(\boldsymbol{X})}|\boldsymbol{X} = \boldsymbol{x})$ for any $\boldsymbol{x}$.
- Write

$$\begin{aligned} \ell(f(\boldsymbol{x})) := & \mathrm{E}(e^{-Yf(\boldsymbol{X})}|\boldsymbol{X} = \boldsymbol{x}) \\ = & e^{-f(\boldsymbol{x})}P(Y = 1|\boldsymbol{X} = \boldsymbol{x}) + e^{f(\boldsymbol{x})}P(Y = -1|\boldsymbol{X} = \boldsymbol{x}) \\ = & e^{-f(\boldsymbol{x})}\eta(\boldsymbol{x}) + e^{f(\boldsymbol{x})}[1 - \eta(\boldsymbol{x})] \end{aligned}$$

Take the derivative of $\ell(f(\boldsymbol{x}))$ with respect to $f(\boldsymbol{x})$, and set it to 0, we have

$$\begin{aligned}
0 = \ell'(f(\boldsymbol{x})) &= -e^{-f(\boldsymbol{x})}\eta(\boldsymbol{x}) + e^{f(\boldsymbol{x})}[1-\eta(\boldsymbol{x})] \\
\Rightarrow e^{-f(\boldsymbol{x})}\eta(\boldsymbol{x}) &= e^{f(\boldsymbol{x})}[1-\eta(\boldsymbol{x})] \\
\Rightarrow e^{2f(\boldsymbol{x})} &= \frac{\eta(\boldsymbol{x})}{1-\eta(\boldsymbol{x})} \\
\Rightarrow 2f(\boldsymbol{x}) &= \log\left(\frac{\eta(\boldsymbol{x})}{1-\eta(\boldsymbol{x})}\right) \\
\Rightarrow f(\boldsymbol{x}) &= \frac{1}{2}\log\left(\frac{\eta(\boldsymbol{x})}{1-\eta(\boldsymbol{x})}\right)
\end{aligned}$$

Recall that this gives the model for logistic regression (when f is a linear function of $\boldsymbol{x}$).

Incidentally, this minimizer of the population risk function under exponential loss coincides with that under the logistic loss function

$$L(y, f(\boldsymbol{x})) := \log(1 + e^{-yf(\boldsymbol{x})})$$

AdaBoost $\approx$ A.M. + Exponential Loss

- We combine the forward stagewise additive modelling with the exponential loss.
- Consider that the basis $g(\cdot)$ in the additive model is each single weak learner $\widehat{\phi}(\cdot)$. What this weak learner really is does not matter.
- In this setting, at each iteration, we seek to find

$$\begin{aligned}\underset{\beta,\gamma}{\operatorname{argmin}} \sum_{i=1}^{n} e^{-y_i\cdot[f_{m-1}(\boldsymbol{x}_i)+\beta\widehat{\phi}(\boldsymbol{x}_i;\gamma)]} &= \sum_{i=1}^{n} e^{-[y_i\cdot f_{m-1}(\boldsymbol{x}_i)+y_i\beta\widehat{\phi}(\boldsymbol{x}_i;\gamma)]} \\ &= \sum_{i=1}^{n} w_i^{(m)} e^{-y_i\beta\widehat{\phi}(\boldsymbol{x}_i;\gamma)}\end{aligned}$$

where $w_i^{(m)} := e^{-y_i\cdot f_{m-1}(\boldsymbol{x}_i)}$

When minimizing $\sum_{i=1}^{n} w_i^{(m)} e^{-y_i \beta \widehat{\phi}(\boldsymbol{x}_i;\gamma)}$, we first fix β:

$$
\begin{aligned}
&\sum_{i=1}^{n} w_i^{(m)} e^{-\beta y_i \widehat{\phi}(\boldsymbol{x}_i;\gamma)} \\
=&\sum_{y_i\widehat{\phi}(\boldsymbol{x}_i;\gamma)=1} w_i^{(m)} e^{-\beta} + \sum_{y_i\widehat{\phi}(\boldsymbol{x}_i;\gamma)=-1} w_i^{(m)} e^{\beta} \\
=&e^{-\beta} \sum_{y_i\widehat{\phi}(\boldsymbol{x}_i;\gamma)=1} w_i^{(m)} + e^{\beta} \sum_{y_i\widehat{\phi}(\boldsymbol{x}_i;\gamma)=-1} w_i^{(m)} \\
=&e^{-\beta} \sum_i w_i^{(m)} \mathbb{1}\{y_i\widehat{\phi}(\boldsymbol{x}_i;\gamma) = 1\} + e^{\beta} \sum_i w_i^{(m)} \mathbb{1}\{y_i\widehat{\phi}(\boldsymbol{x}_i;\gamma) = -1\} \\
=&\left(e^{\beta} - e^{-\beta}\right) \boxed{\textstyle\sum_i w_i^{(m)} \mathbb{1}\{y_i\widehat{\phi}(\boldsymbol{x}_i;\gamma) = -1\}} + e^{-\beta} \sum_i w_i^{(m)}
\end{aligned}
$$

Hence $\widehat{\phi}_m(\boldsymbol{x})$ minimizes $\frac{\sum_i w_i^{(m)} \mathbb{1}\{y_i \neq \widehat{\phi}_m(\boldsymbol{x}_i)\}}{\sum_i w_i^{(m)}}$, the weighted 0-1 loss, i.e. E_m

Once we have $\widehat{\phi}_t(\cdot)$, we find the minimizing β:

$$\ell(\beta) := \left(e^{\beta} - e^{-\beta}\right) \boxed{E_m} + e^{-\beta}$$

$$\ell'(\beta) := \left(e^{\beta} + e^{-\beta}\right) \boxed{E_m} - e^{-\beta} = 0$$

which is minimized at

$$\beta_m = \frac{1}{2} \log \left(\frac{1 - E_m}{E_m} \right)$$

The additive model continues with $f_m(\boldsymbol{x}) = f_{m-1}(\boldsymbol{x}) + \beta_m \widehat{\phi}_m(\boldsymbol{x})$. This means that the weight for the next iteration is $w_i^{(m+1)} = e^{-y_i \cdot f_m(\boldsymbol{x}_i)} = e^{-y_i \cdot [f_{m-1}(\boldsymbol{x}_i) + \beta_m \widehat{\phi}_m(\boldsymbol{x}_i)]} = w_i^{(m)} e^{-y_i \beta_m \widehat{\phi}_m(\boldsymbol{x}_i)}$

Multiplicative factor on the weight updating is

$$e^{-\beta_m y_i \widehat{\phi}_m(\boldsymbol{x}_i)}$$

Recall that $y_i, \widehat{\phi}_m(\boldsymbol{x}_i) \in \{+1, -1\}$. So

$$-y_i \widehat{\phi}_m(\boldsymbol{x}_i) = 2\mathbb{1}\{y_i \neq \widehat{\phi}_m(\boldsymbol{x}_i)\} - 1$$

Hence

$$\begin{aligned} e^{-\beta_m y_i \widehat{\phi}_m(\boldsymbol{x}_i)} &= e^{-\beta_m (2\mathbb{1}\{y_i \neq \widehat{\phi}_m(\boldsymbol{x}_i)\} - 1)} \\ &= e^{-2\beta_m \mathbb{1}\{y_i \neq \widehat{\phi}_m(\boldsymbol{x}_i)\}} e^{-\beta_m} \\ &= e^{-\alpha_m \mathbb{1}\{y_i \neq \widehat{\phi}_m(\boldsymbol{x}_i)\}} e^{-\beta_m} \end{aligned}$$

where $\alpha_m = \log\left(\frac{1-E_m}{E_m}\right)$ was defined in the AdaBoost.M1 algorithm. Note that $e^{-\beta_m}$ will vanish due to normalization.

Short summary,

- The final model is the weighted sum of $\widehat{\phi}_m(\boldsymbol{x})$ with $\beta_m = \frac{1}{2}\alpha_m$ as weights (only half of the weights in AdaBoost.M1)
- The observation weight is updated by $w_i^{(m+1)} = w_i^{(m)} e^{-\alpha_m \mathbb{1}\{y_i \neq \widehat{\phi}_m(\boldsymbol{x}_i)\}}$, the same as in AdaBoost.M1.
- In AdaBoost.M1, the weak learner might have its own loss function. But in principal, it should try to minimize the weighted 0-1 loss.

These suggest that AdaBoost.M1 is **approximately** solving an additive model with weak learners as basis, with the forward stagewise algorithm as the optimization approach, and with the exponential loss.

The reason that it is "approximately" solving, not "exactly" solving, is that the weak learner may not directly minimizes the weighted 0-1 loss.

The next section would be . . . . . .

Bagging in a different view

- In Bagging, we generate bootstrap sample and re-do classification, and aggregate the predictions by majority voting.
- The bootstrap classifiers are not independent and the average effect to reduce variance may not be clear.
- Random Forest: an application of bagging to aggregate bootstrap classification trees, but with some additional randomness due to random subset of variables in tree growing.
- Since trees are known to be noisy (large variance), bagging should bring a lot of benefit to trees.
- But these trees are identically distributed but not independent.

Random Forest

1. For $b = 1$ to B
   (a) Draw a bootstrap sample of size N from $\boldsymbol{X}$, namely $\boldsymbol{X}_b^*$
   (b) Grow a tree T_b based on $\boldsymbol{X}_b^*$ in the common way, except that at each node, find the best split among m random selected subset of all p variables.
   (c) Stop growing when a node has reached to a pre-set minimal node size.
2. Report the forest: $\{T_b\}_1^B$

Regression: $\widehat{f}(\boldsymbol{x}) = \frac{1}{B}\sum_{b=1}^{B} T_b(\boldsymbol{x})$
Classification: $\widehat{\phi}(\boldsymbol{x}) = \text{majority vote}\{T_b(\boldsymbol{x})\}$

Note that the set of m random selected variables may be different at different nodes, even within the same tree.

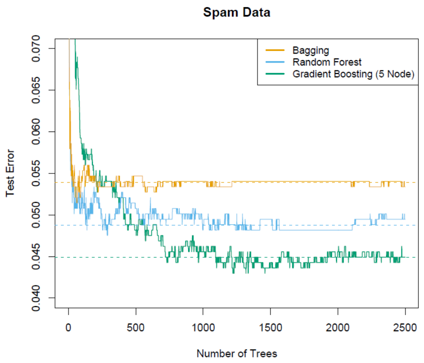


FIGURE 15.1. *Bagging, random forest, and gradient boosting, applied to the spam data. For boosting, 5-node trees were used, and the number of trees were chosen by 10-fold cross-validation (2500 trees). Each "step" in the figure corresponds to a change in a single misclassification (in a test set of 1536).*

Out-of-Bag Sample

- For each bootstrap, there must be a set of observations not selected in the bootstrap sample. – This set of observations is called the Out-of-Bag Sample
- For each observation, there must be a set of trees whose bootstrap samples do not include this observation.

OOB Errors

OOB for the whole forest:

- The OOB Error is defined to be the average of misclassification over all the n observations, where each observation is classified not by the whole forest, but by the sub-forest with those trees whose bootstrap samples do not include this observation.
- Similar to leave-one-out cross validation.
  - 2 observations do not share classifier, compared to 5-fold cv.
  - But, the classifier for each observation is also a random forest
- Unlike the cross validation, there is no additional computational burden. The OOB error is obtained along the way of generating the random forest.

We can also calculate the OOB for a single (bootstrap) tree:

- The misclassification rate of the (bootstrap) tree when it is applied to a OOB sample.

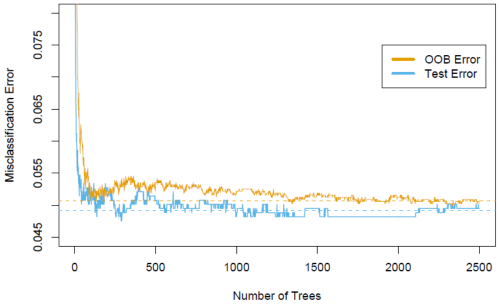


FIGURE 15.4. OOB *error computed on the* `spam` *training data, compared to the test error computed on the test set.*

Variance Importance

- Goal: provide a ranked list of variables representing their relative importance in classification.
- Two versions
  - Permutation based
  - Gini index (or impurity) based

Permutation based variance importance

1. The OOB error for the bth tree in the forest is $E_b(OOB)$
2. For each $j = 1, \ldots, p$, the jth variable in the OOB sample can be randomized (randomly exchange the values on the jth variable). Call this permuted OOB sample OOB_j. A new OOB error for OOB_j can be calculated for the bth tree, denoted as $E_b(OOB_j)$
3. The difference is aggregated over all B trees for each j:

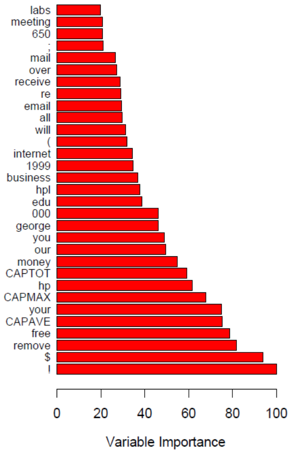
$$VI(j) := \sum_{b=1}^{B} [E_b(OOB_j) - E_b(OOB)]$$

The large VI, the more important the variable is.

- The permutation on variable j works by voiding the effect of variable j, making it usable for the trees.
- Note that it is a little different from removing the variable j from the data and regrowing the tree again!
- Note that the tree is static; we only try a different testing data. Hence there is not much computational burden.
- If a variable j is important, then $E_b(OOB_j) \gg E_b(OOB)$. Otherwise, $E_b(OOB_j) = E_b(OOB)$.

Gini index (or impurity) based variance importance

- Recall that when growing a tree, the split at a variable makes the Gini index on the mother node reduce to the weighted sum of the Gini indices on the two daughter nodes.
- We aggregate such reduction for each variable j over all the B trees
- Large VI $=$ large reduction overall $=$ helped to make many trees reduce impurity $=$ being important.
- It is possible that a variable j does not appear in a single tree; but over the whole forest with many trees, the chance that it is not selected by one tree is very small.



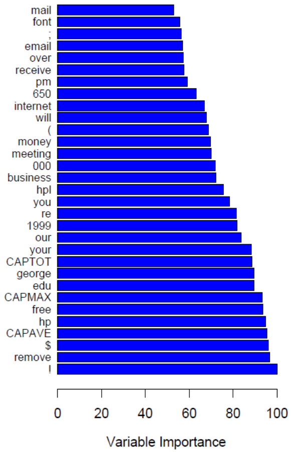


Figure: Left: Gini-based VI. Right: Randomization VI.

Remarks

- Choice of m: classification – $\sqrt{m}$; regression – $p/3$
- Minimal node size: classification – 1; regression – 5
- Random forest is an improved version of bagging trees.
- Friedman and Hall showed that subsampling (without replacement) with $N/2$ is approximately equivalent to bootstrap, while smaller fraction of N may reduce the variance even further.
- In R, package `randomForest`.