

SF2930 Regression Analysis

Alexandre Chotard

Tree-based regression and classification

15 February 2018



Today

Overview

Regression trees

Pruning

Bagging, random forests



Setup

- ▶ Let Y be a real-valued random variable. Given $\mathbf{x} \in \mathbb{X} \subset \mathbb{R}^d$, we want to predict the value of Y .
- ▶ To understand how different values of \mathbf{x} influence Y , we have $N \in \mathbb{N}^*$ examples $(\mathbf{x}_i, y_i)_{i \in [1..N]}$.

Given \mathcal{F} a family of functions and Q a cost function, solve

$$\min_{f \in \mathcal{F}} \sum_{i=1}^N Q(f(\mathbf{x}_i), y_i)$$

- ▶ The variable Y can have discrete or continuous values, corresponding respectively to classification and regression problems.



Setup

- ▶ Let Y be a real-valued random variable. Given $\mathbf{x} \in \mathbb{X} \subset \mathbb{R}^d$, we want to predict the value of Y .
- ▶ To understand how different values of \mathbf{x} influence Y , we have $N \in \mathbb{N}^*$ examples $(\mathbf{x}_i, y_i)_{i \in [1..N]}$.

Given \mathcal{F} a family of functions and Q a cost function, solve

$$\min_{f \in \mathcal{F}} \sum_{i=1}^N Q(f(\mathbf{x}_i), y_i)$$

- ▶ The variable Y can have discrete or continuous values, corresponding respectively to classification and regression problems.



Setup

- ▶ Let Y be a real-valued random variable. Given $\mathbf{x} \in \mathbb{X} \subset \mathbb{R}^d$, we want to predict the value of Y .
- ▶ To understand how different values of \mathbf{x} influence Y , we have $N \in \mathbb{N}^*$ examples $(\mathbf{x}_i, y_i)_{i \in [1..N]}$.

Given \mathcal{F} a family of functions and Q a cost function, solve

$$\min_{f \in \mathcal{F}} \sum_{i=1}^N Q(f(\mathbf{x}_i), y_i)$$

- ▶ The variable Y can have discrete or continuous values, corresponding respectively to classification and regression problems.



Setup

- ▶ Let Y be a real-valued random variable. Given $\mathbf{x} \in \mathbb{X} \subset \mathbb{R}^d$, we want to predict the value of Y .
- ▶ To understand how different values of \mathbf{x} influence Y , we have $N \in \mathbb{N}^*$ examples $(\mathbf{x}_i, y_i)_{i \in [1..N]}$.

Given \mathcal{F} a family of functions and Q a cost function, solve

$$\min_{f \in \mathcal{F}} \sum_{i=1}^N Q(f(\mathbf{x}_i), y_i) + \alpha \text{Complexity}(f)$$

- ▶ The variable Y can have discrete or continuous values, corresponding respectively to classification and regression problems.



Setup

- ▶ Let Y be a real-valued random variable. Given $\mathbf{x} \in \mathbb{X} \subset \mathbb{R}^d$, we want to predict the value of Y .
- ▶ To understand how different values of \mathbf{x} influence Y , we have $N \in \mathbb{N}^*$ examples $(\mathbf{x}_i, y_i)_{i \in [1..N]}$.

Given \mathcal{F} a family of functions and Q a cost function, solve

$$\min_{f \in \mathcal{F}} \sum_{i=1}^N Q(f(\mathbf{x}_i), y_i) + \alpha \text{Complexity}(f)$$

- ▶ The variable Y can have discrete or continuous values, corresponding respectively to classification and regression problems.



Setup

- ▶ We consider here binary decision trees. A decision tree partitions \mathbb{X} into simple regions $(R_j)_{j \in [1..J]}$. For $\mathbf{x} = (x^1, \dots, x^d) \in \mathbb{R}^d$ and j such that $\mathbf{x} \in R_j$, the value predicted by the tree is
 - ▶ the most represented class y among $\{x_i \in R_j\}$ in classification problems,
 - ▶ the average of $\{y_i | x_i \in R_j, i \in [1..M]\}$ in regression problems.



Setup

- ▶ We consider here binary decision trees. A decision tree partitions \mathbb{X} into simple regions $(R_j)_{j \in [1..J]}$. For $\mathbf{x} = (x^1, \dots, x^d) \in \mathbb{R}^d$ and j such that $\mathbf{x} \in R_j$, the value predicted by the tree is
 - ▶ the most represented class y among $\{x_i \in R_j\}$ in classification problems,
 - ▶ the average of $\{y_i | x_i \in R_j, i \in [1..M]\}$ in regression problems.



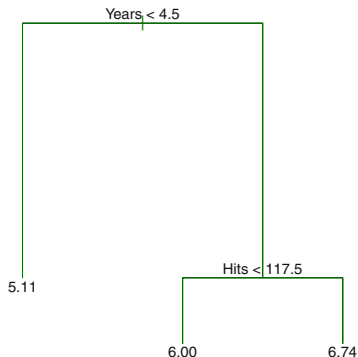
Setup

- ▶ We consider here binary decision trees. A decision tree partitions \mathbb{X} into simple regions $(R_j)_{j \in [1..J]}$. For $\mathbf{x} = (x^1, \dots, x^d) \in \mathbb{R}^d$ and j such that $\mathbf{x} \in R_j$, the value predicted by the tree is
 - ▶ the most represented class y among $\{\mathbf{x}_i \in R_j\}$ in classification problems,
 - ▶ the average of $\{y_i | \mathbf{x}_i \in R_j, i \in [1..N]\}$ in regression problems.

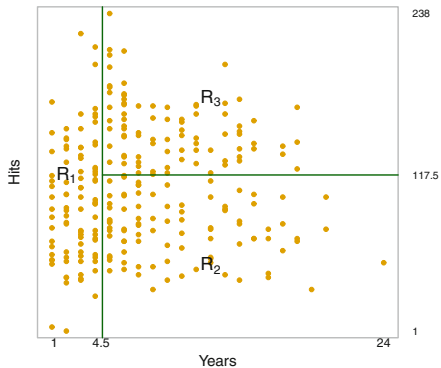


Setup

- ▶ Each node n of the tree is associated to a subset R_n of \mathbb{X} , which is further split into two regions through a test comparing x^i for some $i \in [1..d]$ with a value $c \in \mathbb{R}$. Then for l and r left and right children of n ,
 - ▶ $R_l := \{\mathbf{x} \in R_n \mid x^i < c\}$
 - ▶ $R_r := \{\mathbf{x} \in R_n \mid x^i \geq c\}$

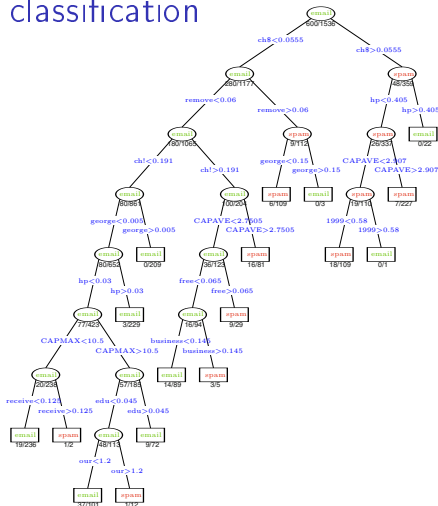


Figur: A regression tree for predicting the log salary (in 1,000\$) of a baseball player, based on the number of years played and the number of hits made in the previous year.



Figur: The three-region partition provided by the regression tree in the previous figure.

Example: spam classification



Figur: Tree for spam classification. Split variables are shown in blue. CAPMAX and CAPAVE = maximal and average length of the longest uninterrupted sequences of capital letters, respectively.

Questions

- ▶ How do we construct such trees?
- ▶ How large should a tree be?
- ▶ What are the pros and cons of tree-based methods?



Today

Overview

Regression trees

Pruning

Bagging, random forests



Regression trees vs. linear regression

- ▶ We generally want to approximate a relation $Y = f(\mathbf{x}) + \varepsilon$.
- ▶ Linear regression involves linear functions

$$f(\mathbf{x}) = \beta_0 + \sum_{j=1}^p \beta_j \mathbf{x}_j.$$

- ▶ We will instead consider

$$f(\mathbf{x}) = \sum_{j=1}^J m_j \mathbb{1}_{R_j}(\mathbf{x}),$$

for a partition $\{R_j\}_{j \in [1..J]}$ of \mathbb{X} .



Regression trees vs. linear regression

- ▶ We generally want to approximate a relation $Y = f(\mathbf{x}) + \varepsilon$.
- ▶ Linear regression involves linear functions

$$f(\mathbf{x}) = \beta_0 + \sum_{j=1}^p \beta_j \mathbf{x}_j.$$

- ▶ We will instead consider

$$f(\mathbf{x}) = \sum_{j=1}^J m_j \mathbb{1}_{R_j}(\mathbf{x}),$$

for a partition $\{R_j\}_{j \in [1..J]}$ of \mathbb{X} .



Regression trees vs. linear regression

- ▶ We generally want to approximate a relation $Y = f(\mathbf{x}) + \varepsilon$.
- ▶ Linear regression involves linear functions

$$f(\mathbf{x}) = \beta_0 + \sum_{j=1}^p \beta_j \mathbf{x}_j.$$

- ▶ We will instead consider

$$f(\mathbf{x}) = \sum_{j=1}^J m_j \mathbb{1}_{R_j}(\mathbf{x}),$$

for a partition $\{R_j\}_{j \in [1..J]}$ of \mathbb{X} .



Building a regression tree: cost function

- ▶ We will construct the tree by minimizing

$$\begin{aligned} \text{RSS}(m_1, \dots, m_J, R_1, \dots, R_J) &= \sum_{i=1}^N (y_i - f(\mathbf{x}_i))^2 \\ &= \sum_{j=1}^J \sum_{i: \mathbf{x}_i \in R_j} (y_i - f(\mathbf{x}_i))^2 = \sum_{j=1}^J \sum_{i: \mathbf{x}_i \in R_j} (y_i - m_j)^2. \end{aligned}$$

- ▶ For each j , the m_j minimizing the sum of squares $\sum_{i: \mathbf{x}_i \in R_j} (y_i - m_j)^2$ is always

$$m_j = \bar{y}_{R_j} = \text{average over all } y_i \text{ such that } \mathbf{x}_i \in R_j.$$

- ▶ To find the minimizing R_j is however computationally infeasible in general.



Building a regression tree: cost function

- ▶ We will construct the tree by minimizing

$$\begin{aligned} \text{RSS}(m_1, \dots, m_J, R_1, \dots, R_J) &= \sum_{i=1}^N (y_i - f(\mathbf{x}_i))^2 \\ &= \sum_{j=1}^J \sum_{i:\mathbf{x}_i \in R_j} (y_i - f(\mathbf{x}_i))^2 = \sum_{j=1}^J \sum_{i:\mathbf{x}_i \in R_j} (y_i - m_j)^2. \end{aligned}$$

- ▶ For each j , the m_j minimizing the sum of squares $\sum_{i:\mathbf{x}_i \in R_j} (y_i - m_j)^2$ is always

$$m_j = \bar{y}_{R_j} = \text{average over all } y_i \text{ such that } \mathbf{x}_i \in R_j.$$

- ▶ To find the minimizing R_j is however computationally infeasible in general.



Building a regression tree: cost function

- ▶ We will construct the tree by minimizing

$$\begin{aligned} \text{RSS}(m_1, \dots, m_J, R_1, \dots, R_J) &= \sum_{i=1}^N (y_i - f(\mathbf{x}_i))^2 \\ &= \sum_{j=1}^J \sum_{i:\mathbf{x}_i \in R_j} (y_i - f(\mathbf{x}_i))^2 = \sum_{j=1}^J \sum_{i:\mathbf{x}_i \in R_j} (y_i - m_j)^2. \end{aligned}$$

- ▶ For each j , the m_j minimizing the sum of squares $\sum_{i:\mathbf{x}_i \in R_j} (y_i - m_j)^2$ is always

$$m_j = \bar{y}_{R_j} = \text{average over all } y_i \text{ such that } \mathbf{x}_i \in R_j.$$

- ▶ To find the minimizing R_j is however computationally infeasible in general.



Building a regression tree: top-down minimization

- ▶ Thus, in order to find regions $\{R_j\}_{j=1}^J$ minimizing

$$\sum_{j=1}^J \underbrace{\sum_{i: x_i \in R_j} (y_i - \bar{y}_{R_j})^2}_{Q_{R_j}}$$

we proceed with a *top-down, greedy* approach.

- ▶ In the first step, when splitting \mathbb{X} into

$$R_1(k, s) = \{x \in \mathbb{X} : x^k < s\} \quad \text{and} \quad R_2(k, s) = \{x \in \mathbb{X} : x^k \geq s\},$$

we minimize

$$Q_{R_1(k,s)} + Q_{R_2(k,s)}$$

w.r.t. k and s .



Building a regression tree: top-down minimization

- ▶ Thus, in order to find regions $\{R_j\}_{j=1}^J$ minimizing

$$\sum_{j=1}^J \underbrace{\sum_{i: \mathbf{x}_i \in R_j} (y_i - \bar{y}_{R_j})^2}_{Q_{R_j}}$$

we proceed with a *top-down, greedy* approach.

- ▶ In the first step, when splitting \mathbb{X} into

$$R_1(k, s) = \{\mathbf{x} \in \mathbb{X} : x^k < s\} \quad \text{and} \quad R_2(k, s) = \{\mathbf{x} \in \mathbb{X} : x^k \geq s\},$$

we minimize

$$Q_{R_1(k,s)} + Q_{R_2(k,s)}$$

w.r.t. k and s .



Building a regression tree: top-down minimization (cont.)

- ▶ For a given k , the optimal s is quite easily found by scanning through the data.
- ▶ Repeating for all $k \in [1..d]$ yields an optimal pair (k, s) .
- ▶ Having found the best split, we partition the data into the two resulting regions and repeat the splitting process on each of these. And so forth.
- ▶ We stop growing the tree when some stopping criterion is reached, e.g., when no region contains more than five training data.



Building a regression tree: top-down minimization (cont.)

- ▶ For a given k , the optimal s is quite easily found by scanning through the data.
- ▶ Repeating for all $k \in [1..d]$ yields an optimal pair (k, s) .
- ▶ Having found the best split, we partition the data into the two resulting regions and repeat the splitting process on each of these. And so forth.
- ▶ We stop growing the tree when some stopping criterion is reached, e.g., when no region contains more than five training data.



Building a regression tree: top-down minimization (cont.)

- ▶ For a given k , the optimal s is quite easily found by scanning through the data.
- ▶ Repeating for all $k \in [1..d]$ yields an optimal pair (k, s) .
- ▶ Having found the best split, we partition the data into the two resulting regions and repeat the splitting process on each of these. And so forth.
- ▶ We stop growing the tree when some stopping criterion is reached, e.g., when no region contains more than five training data.



Bias versus variance

- ▶ The tree size is a tuning parameter governing the model's complexity.
- ▶ A too large tree leads to *overfitting* (high variance on test sets), while a too small tree leads to high bias.
- ▶ The optimal tree size should be determined adaptively from the data.
- ▶ One possibility is to split a node only if the split implies an RSS reduction exceeding a certain threshold; this is however treacherous in top-down minimization, as an apparently meaningless split may imply a good split later on.



Bias versus variance

- ▶ The tree size is a tuning parameter governing the model's complexity.
- ▶ A too large tree leads to *overfitting* (high variance on test sets), while a too small tree leads to high bias.
- ▶ The optimal tree size should be determined adaptively from the data.
- ▶ One possibility is to split a node only if the split implies an RSS reduction exceeding a certain threshold; this is however treacherous in top-down minimization, as an apparently meaningless split may imply a good split later on.



Today

Overview

Regression trees

Pruning

Bagging, random forests



Cost complexity pruning

- ▶ Grow a large tree T_0 while storing the different Q_{R_j} associated with each split.
- ▶ We define a subtree $T \subseteq T_0$ obtained by *pruning* T_0 in a *bottom-up* fashion.
- ▶ Let $|T|$ denote the number of leaves $R_m(T)$ in T .
- ▶ Define, for some $\alpha \geq 0$, the cost function

$$C_\alpha(T) = \sum_{m=1}^{|T|} Q_{R_m(T)} + \alpha|T|,$$

where the $Q_{R_m(T)}$ are available from the first step.

- ▶ The idea is to find a subtree T_α minimizing $C_\alpha(T)$.



Cost complexity pruning

- ▶ Grow a large tree T_0 while storing the different Q_{R_j} associated with each split.
- ▶ We define a subtree $T \subseteq T_0$ obtained by *pruning* T_0 in a *bottom-up* fashion.
- ▶ Let $|T|$ denote the number of leaves $R_m(T)$ in T .
- ▶ Define, for some $\alpha \geq 0$, the cost function

$$C_\alpha(T) = \sum_{m=1}^{|T|} Q_{R_m(T)} + \alpha|T|,$$

where the $Q_{R_m(T)}$ are available from the first step.

- ▶ The idea is to find a subtree T_α minimizing $C_\alpha(T)$.



Cost complexity pruning

- ▶ Grow a large tree T_0 while storing the different Q_{R_j} associated with each split.
- ▶ We define a subtree $T \subseteq T_0$ obtained by *pruning* T_0 in a *bottom-up* fashion.
- ▶ Let $|T|$ denote the number of leaves $R_m(T)$ in T .
- ▶ Define, for some $\alpha \geq 0$, the cost function

$$C_\alpha(T) = \sum_{m=1}^{|T|} Q_{R_m(T)} + \alpha|T|,$$

where the $Q_{R_m(T)}$ are available from the first step.

- ▶ The idea is to find a subtree T_α minimizing $C_\alpha(T)$.



Cost complexity pruning (cont.)

- ▶ Large values of α leads to smaller trees.
- ▶ For $\alpha = 0$, the solution is T_0 .
- ▶ To find T_α , we create a sequence of subtrees by removing successively leaves whose elimination leads to the smallest increase of $\sum_{m=1}^{|\mathcal{T}|} Q_{R_m}(T)$.
- ▶ To find the optimal α , apply K -fold cross validation.



Cost complexity pruning (cont.)

- ▶ Large values of α leads to smaller trees.
- ▶ For $\alpha = 0$, the solution is T_0 .
- ▶ To find T_α , we create a sequence of subtrees by removing successively leaves whose elimination leads to the smallest increase of $\sum_{m=1}^{|\mathcal{T}|} Q_{R_m}(T)$.
- ▶ To find the optimal α , apply K -fold cross validation.



Cost complexity pruning (cont.)

- ▶ Large values of α leads to smaller trees.
- ▶ For $\alpha = 0$, the solution is T_0 .
- ▶ To find T_α , we create a sequence of subtrees by removing successively leaves whose elimination leads to the smallest increase of $\sum_{m=1}^{|T|} Q_{R_m}(T)$.
- ▶ To find the optimal α , apply K -fold cross validation.

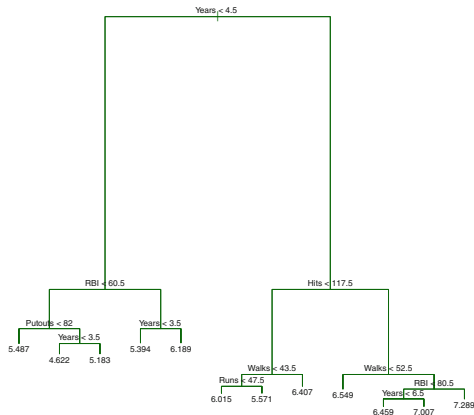


Cost complexity pruning (cont.)

- ▶ Large values of α leads to smaller trees.
- ▶ For $\alpha = 0$, the solution is T_0 .
- ▶ To find T_α , we create a sequence of subtrees by removing successively leaves whose elimination leads to the smallest increase of $\sum_{m=1}^{|T|} Q_{R_m}(T)$.
- ▶ To find the optimal α , apply K -fold cross validation.



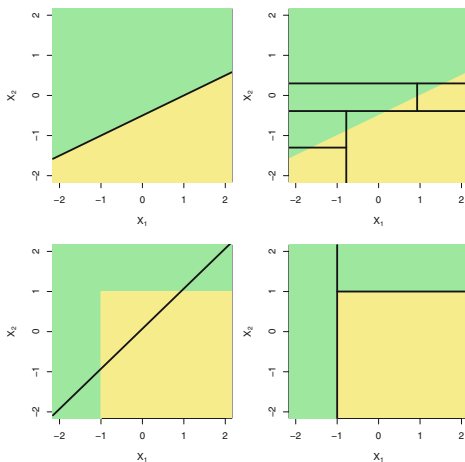
Example: baseball player salaries (cont.)



Figur: Unpruned tree for the log salary (in 1,000\$) of a baseball player.



Trees vs. linear models



Figur: True decision boundary is indicated by the shaded regions. Linear boundary vs. decision tree (boxes).

Node impurity

- ▶ The task of growing a classification tree is very similar to that of growing a regression tree. The only difference is that the RSS is not meaningful any longer.
- ▶ We thus need to find alternative measures $Q_m(T)$ of *node impurity*. For this purpose, define

$$\hat{p}_{j,k} = \frac{1}{|R_j|} \sum_{i: x_i \in R_j} \mathbb{1}_{\{y_i=k\}} \quad \text{and} \quad k(j) = \arg \max_k \hat{p}_{j,k}$$

and let, e.g.,

$$Q_m(T) = \begin{cases} 1 - \hat{p}_{j,k(j)} & \text{misclassification error,} \\ \sum_{k=1}^K \hat{p}_{j,k}(1 - \hat{p}_{j,k}) & \text{Gini index,} \\ -\sum_{k=1}^K \hat{p}_{j,k} \log \hat{p}_{j,k} & \text{cross-entropy.} \end{cases}$$



Node impurity

- ▶ The task of growing a classification tree is very similar to that of growing a regression tree. The only difference is that the RSS is not meaningful any longer.
- ▶ We thus need to find alternative measures $Q_m(T)$ of *node impurity*. For this purpose, define

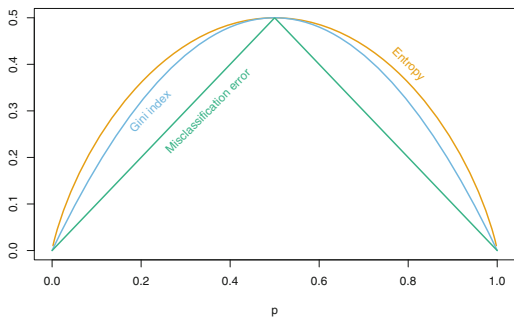
$$\hat{p}_{j,k} = \frac{1}{|R_j|} \sum_{i: x_i \in R_j} \mathbb{1}_{\{y_i=k\}} \quad \text{and} \quad k(j) = \arg \max_k \hat{p}_{j,k}$$

and let, e.g.,

$$Q_m(T) = \begin{cases} 1 - \hat{p}_{j,k(j)} & \text{misclassification error,} \\ \sum_{k=1}^K \hat{p}_{j,k}(1 - \hat{p}_{j,k}) & \text{Gini index,} \\ -\sum_{k=1}^K \hat{p}_{j,k} \log \hat{p}_{j,k} & \text{cross-entropy.} \end{cases}$$



Node impurity measures



Figur: Node impurity measures for two-class classification, as a function of the proportion p in class 2. (Cross-entropy has been scaled.)

Pros and cons with tree-based methods

- ▶ Pros:

- ▶ Easy to explain,
- ▶ able to handle categorical or numerical data,
- ▶ scales well with N

- ▶ Cons:

- ▶ Do not, in their most basic form, have the same level of predictive accuracy as some other regression and classification approaches,
 - ▶ can be very non-robust (i.e., small changes in the data leads to completely different trees).
 - ▶ highly sensitive to data-representation: ill-suited to many problems
- ▶ The predictive accuracy can however be drastically improved using ideas as *bagging* and *random forests* discussed in the following.



Pros and cons with tree-based methods

- ▶ Pros:
 - ▶ Easy to explain,
 - ▶ able to handle categorical or numerical data,
 - ▶ scales well with N
- ▶ Cons:
 - ▶ Do not, in their most basic form, have the same level of predictive accuracy as some other regression and classification approaches,
 - ▶ can be very non-robust (i.e., small changes in the data leads to completely different trees).
 - ▶ highly sensitive to data-representation: ill-suited to many problems
- ▶ The predictive accuracy can however be drastically improved using ideas as *bagging* and *random forests* discussed in the following.



Pros and cons with tree-based methods

- ▶ Pros:
 - ▶ Easy to explain,
 - ▶ able to handle categorical or numerical data,
 - ▶ scales well with N
- ▶ Cons:
 - ▶ Do not, in their most basic form, have the same level of predictive accuracy as some other regression and classification approaches,
 - ▶ can be very non-robust (i.e., small changes in the data leads to completely different trees).
 - ▶ highly sensitive to data-representation: ill-suited to many problems
- ▶ The predictive accuracy can however be drastically improved using ideas as *bagging* and *random forests* discussed in the following.



Today

Overview

Regression trees

Pruning

Bagging, random forests



Prelude: the bootstrap

- ▶ Given an i.i.d. sample $\{\mathbf{x}_i\}_{i=1}^N$ from a probability density p , the *bootstrap algorithm* is based on the approximation

$$\hat{p}(\mathbf{x}) = \frac{1}{N} \sum_{i=1}^N \delta_{\mathbf{x}_i}(\mathbf{x})$$

of p .

- ▶ Given \hat{p} , a new sample set $\{\mathbf{x}_i^*\}_{i=1}^N$ with approximately the same distribution as $\{\mathbf{x}_i\}_{i=1}^N$ can be formed by sampling from \hat{p} , i.e., by drawing repeatedly N times among $\{\mathbf{x}_i\}_{i=1}^N$ with replacement.
- ▶ In machine learning applications, this can be used for creating artificial replicates of the training set.



Prelude: the bootstrap

- ▶ Given an i.i.d. sample $\{\mathbf{x}_i\}_{i=1}^N$ from a probability density p , the *bootstrap algorithm* is based on the approximation

$$\hat{p}(\mathbf{x}) = \frac{1}{N} \sum_{i=1}^N \delta_{\mathbf{x}_i}(\mathbf{x})$$

of p .

- ▶ Given \hat{p} , a new sample set $\{\mathbf{x}_i^*\}_{i=1}^N$ with approximately the same distribution as $\{\mathbf{x}_i\}_{i=1}^N$ can be formed by sampling from \hat{p} , i.e., by drawing repeatedly N times among $\{\mathbf{x}_i\}_{i=1}^N$ with replacement.
- ▶ In machine learning applications, this can be used for creating artificial replicates of the training set.



Prelude: the bootstrap

- ▶ Given an i.i.d. sample $\{\mathbf{x}_i\}_{i=1}^N$ from a probability density p , the *bootstrap algorithm* is based on the approximation

$$\hat{p}(\mathbf{x}) = \frac{1}{N} \sum_{i=1}^N \delta_{\mathbf{x}_i}(\mathbf{x})$$

of p .

- ▶ Given \hat{p} , a new sample set $\{\mathbf{x}_i^*\}_{i=1}^N$ with approximately the same distribution as $\{\mathbf{x}_i\}_{i=1}^N$ can be formed by sampling from \hat{p} , i.e., by drawing repeatedly N times among $\{\mathbf{x}_i\}_{i=1}^N$ with replacement.
- ▶ In machine learning applications, this can be used for creating artificial replicates of the training set.



Bagging

- ▶ In the case of regression, we have approximated the target function using

$$f(\mathbf{x}) = \sum_{j=1}^J \bar{y}_{R_j} \mathbb{1}_{R_j}(\mathbf{x}),$$

where the partition $\{R_j\}_{j=1}^J$ is formed using a decision tree.

- ▶ The decision trees discussed above suffer generally from *high variance*, i.e., building the tree using a different training set may lead to a quite different f .
- ▶ Better would be to use B independent training sets of similar size, each yielding a tree f_b , and use the model

$$f_{\text{avg}}(\mathbf{x}) = \frac{1}{B} \sum_{b=1}^B f_b(\mathbf{x}).$$



Bagging

- ▶ In the case of regression, we have approximated the target function using

$$f(\mathbf{x}) = \sum_{j=1}^J \bar{y}_{R_j} \mathbb{1}_{R_j}(\mathbf{x}),$$

where the partition $\{R_j\}_{j=1}^J$ is formed using a decision tree.

- ▶ The decision trees discussed above suffer generally from *high variance*, i.e., building the tree using a different training set may lead to a quite different f .
- ▶ Better would be to use B independent training sets of similar size, each yielding a tree f_b , and use the model

$$f_{\text{avg}}(\mathbf{x}) = \frac{1}{B} \sum_{b=1}^B f_b(\mathbf{x}).$$



Bagging

- ▶ In the case of regression, we have approximated the target function using

$$f(\mathbf{x}) = \sum_{j=1}^J \bar{y}_{R_j} \mathbb{1}_{R_j}(\mathbf{x}),$$

where the partition $\{R_j\}_{j=1}^J$ is formed using a decision tree.

- ▶ The decision trees discussed above suffer generally from *high variance*, i.e., building the tree using a different training set may lead to a quite different f .
- ▶ Better would be to use B independent training sets of similar size, each yielding a tree f_b , and use the model

$$f_{\text{avg}}(\mathbf{x}) = \frac{1}{B} \sum_{b=1}^B f_b(\mathbf{x}).$$



Bagging (cont.)

- ▶ We use bootstrap sampling for creating artificially B such training sets, each yielding a tree f_b^* , and use the model

$$f_{\text{bag}}(\mathbf{x}) = \frac{1}{B} \sum_{b=1}^B f_b^*(\mathbf{x}).$$

- ▶ The trees f_b^* are grown deep without pruning, implying that each tree has high variance but low bias.
- ▶ In the classification case, we can record the class predicted by each of the B trees, and take the *most popular vote*.



Bagging (cont.)

- ▶ We use bootstrap sampling for creating artificially B such training sets, each yielding a tree f_b^* , and use the model

$$f_{\text{bag}}(\mathbf{x}) = \frac{1}{B} \sum_{b=1}^B f_b^*(\mathbf{x}).$$

- ▶ The trees f_b^* are grown deep without pruning, implying that each tree has high variance but low bias.
- ▶ In the classification case, we can record the class predicted by each of the B trees, and take the *most popular vote*.



Bagging (cont.)

- ▶ We use bootstrap sampling for creating artificially B such training sets, each yielding a tree f_b^* , and use the model

$$f_{\text{bag}}(\mathbf{x}) = \frac{1}{B} \sum_{b=1}^B f_b^*(\mathbf{x}).$$

- ▶ The trees f_b^* are grown deep without pruning, implying that each tree has high variance but low bias.
- ▶ In the classification case, we can record the class predicted by each of the B trees, and take the *most popular vote*.



Bagging (cont.)

- ▶ The number of trees B is not a critical parameter with bagging; using a very large value of B will not lead to overfitting.
- ▶ The reason is that the probability of making an error *converges* as $B \rightarrow \infty$.
- ▶ The probability of making an error depends also on the expected correlation between errors of different trees $f_b(\mathbf{x})$ and $f_{b'}(\mathbf{x})$ when (\mathbf{x}, Y) varies.



Some theory

- ▶ Indeed, in the classification case, let the *margin function* be

$$m(\mathbf{x}, y) = \frac{1}{B} \sum_{b=1}^B \mathbb{1}_{\{f_b(\mathbf{x})=y\}} - \max_{k \neq y} \frac{1}{B} \sum_{b=1}^B \mathbb{1}_{\{f_b(\mathbf{x})=k\}}$$

- ▶ Moreover, define the *generalization error* by

$$\text{PE} = \mathbb{P}(m(\mathbf{x}, Y) < 0).$$

- ▶ **breiman:2001** (**breiman:2001**) shows that PE *converges* as $B \rightarrow \infty$. In addition, the PE can be bounded as

$$\text{PE} \leq (1 - s^2)\rho/s^2,$$

where, roughly, s is the limiting margin function—the *strength*—and ρ describes the expected *correlation* between the trees' errors as (\mathbf{x}, Y) varies.



Some theory

- ▶ Indeed, in the classification case, let the *margin function* be

$$m(\mathbf{x}, y) = \frac{1}{B} \sum_{b=1}^B \mathbb{1}_{\{f_b(\mathbf{x})=y\}} - \max_{k \neq y} \frac{1}{B} \sum_{b=1}^B \mathbb{1}_{\{f_b(\mathbf{x})=k\}}$$

- ▶ Moreover, define the *generalization error* by

$$\text{PE} = \mathbb{P}(m(\mathbf{x}, Y) < 0).$$

- ▶ breiman:2001 (breiman:2001) shows that PE *converges* as $B \rightarrow \infty$. In addition, the PE can be bounded as

$$\text{PE} \leq (1 - s^2)\rho/s^2,$$

where, roughly, s is the limiting margin function—the *strength*—and ρ describes the expected *correlation* between the trees' errors as (\mathbf{x}, Y) varies.



Some theory

- ▶ Indeed, in the classification case, let the *margin function* be

$$m(\mathbf{x}, y) = \frac{1}{B} \sum_{b=1}^B \mathbb{1}_{\{f_b(\mathbf{x})=y\}} - \max_{k \neq y} \frac{1}{B} \sum_{b=1}^B \mathbb{1}_{\{f_b(\mathbf{x})=k\}}$$

- ▶ Moreover, define the *generalization error* by

$$\text{PE} = \mathbb{P}(m(\mathbf{x}, Y) < 0).$$

- ▶ **breiman:2001 (breiman:2001)** shows that PE *converges* as $B \rightarrow \infty$. In addition, the PE can be bounded as

$$\text{PE} \leq (1 - s^2) \rho / s^2,$$

where, roughly, s is the limiting margin function—the *strength*—and ρ describes the expected *correlation* between the trees' errors as (\mathbf{x}, Y) varies.



Random park \Rightarrow random forest

- ▶ In order to *decorrelate* the trees, it is desirable so build them different.
- ▶ Say that there is one strong predictor x^i along with a number of other moderately strong predictors \Rightarrow most or all of the trees will use this strong predictor in the top split \Rightarrow all trees will look quite similar \Rightarrow highly correlated predictions.
- ▶ Thus, at each split, consider only splitting of a *randomly chosen subset* of $m \leq d$ predictors.
- ▶ Therefore, on average $(d - m)/d$ of the splits will not even consider the strong predictor, and so other predictors will have more of a chance.
- ▶ Typically, one uses $m \approx \sqrt{d}$; $m = d$ corresponds to standard bagging.



Random park \Rightarrow random forest

- ▶ In order to *decorrelate* the trees, it is desirable so build them different.
- ▶ Say that there is one strong predictor x^i along with a number of other moderately strong predictors \Rightarrow most or all of the trees will use this strong predictor in the top split \Rightarrow all trees will look quite similar \Rightarrow highly correlated predictions.
- ▶ Thus, at each split, consider only splitting of a *randomly chosen subset* of $m \leq d$ predictors.
- ▶ Therefore, on average $(d - m)/d$ of the splits will not even consider the strong predictor, and so other predictors will have more of a chance.
- ▶ Typically, one uses $m \approx \sqrt{d}$; $m = d$ corresponds to standard bagging.



Random park \Rightarrow random forest

- ▶ In order to *decorrelate* the trees, it is desirable so build them different.
- ▶ Say that there is one strong predictor x^i along with a number of other moderately strong predictors \Rightarrow most or all of the trees will use this strong predictor in the top split \Rightarrow all trees will look quite similar \Rightarrow highly correlated predictions.
- ▶ Thus, at each split, consider only splitting of a *randomly chosen subset* of $m \leq d$ predictors.
- ▶ Therefore, on average $(d - m)/d$ of the splits will not even consider the strong predictor, and so other predictors will have more of a chance.
- ▶ Typically, one uses $m \approx \sqrt{d}$; $m = d$ corresponds to standard bagging.



Random park \Rightarrow random forest

- ▶ In order to *decorrelate* the trees, it is desirable so build them different.
- ▶ Say that there is one strong predictor x^i along with a number of other moderately strong predictors \Rightarrow most or all of the trees will use this strong predictor in the top split \Rightarrow all trees will look quite similar \Rightarrow highly correlated predictions.
- ▶ Thus, at each split, consider only splitting of a *randomly chosen subset* of $m \leq d$ predictors.
- ▶ Therefore, on average $(d - m)/d$ of the splits will not even consider the strong predictor, and so other predictors will have more of a chance.
- ▶ Typically, one uses $m \approx \sqrt{d}$; $m = d$ corresponds to standard bagging.

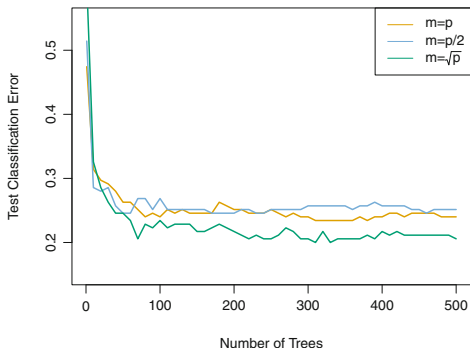


Random park \Rightarrow random forest

- ▶ In order to *decorrelate* the trees, it is desirable so build them different.
- ▶ Say that there is one strong predictor x^i along with a number of other moderately strong predictors \Rightarrow most or all of the trees will use this strong predictor in the top split \Rightarrow all trees will look quite similar \Rightarrow highly correlated predictions.
- ▶ Thus, at each split, consider only splitting of a *randomly chosen subset* of $m \leq d$ predictors.
- ▶ Therefore, on average $(d - m)/d$ of the splits will not even consider the strong predictor, and so other predictors will have more of a chance.
- ▶ Typically, one uses $m \approx \sqrt{d}$; $m = d$ corresponds to standard bagging.



Example: cancer-type prediction



Figur: Here $p = 500$. The test error as a function of the number of trees. Each colored line corresponds to a different value of m . A single classification tree has an error rate of 45.7%.