Lecture 6 – Of Trees and Forests

Martin Spindler

2016-03-15

Introduction

- Idea: Partition the feature / covariables space into a set of rectangles and then fit a simple model (constant) in each one. So the estimated function is the average of outcomes falling in this rectangle.
- Recursive binary partitions, i.e. sequentially we choose variable and corresponding split pint which achieve best fit until some stopping criterion is reached.
- Here: Regression, but also used for classification (with different criteria)
- Example [cf blackboard]

Regression Trees

n observations $(x_i, y_i), i = 1, \dots, n, \quad x_i = (x_{i1}, \dots, x_{ip})$
Given a partition into M regions $R_1, \dots, R_M$ with a fitted constant in each region:

$$f(x) = \sum_{m=1}^{M} c_m I(x \in R_m)$$

Minimizing the residual sum of squares (RSS) $\sum_{x_i \in R_m} (y_i - f(x_i))^2$
leads to $\hat{c}_m = ave(y_i | x_i \in R_m)$
Finding the best partition in terms of minimal RSS is generally computational infeasible.
Instead: greedy algorithm

Regression Trees

Algorithm:

Start with all data

1. We consider a splitting variable j and split point s, s.t.

$$\min_{j,s}\left[\min_{c_1}\sum_{x_i\in R_1(j,s)}(y_i-c_1)^2+\min_{c_2}\sum_{x_i\in R_2(j,s)}(y_i-c_2)^2\right]$$

with $R_1(j,s)=\{X|X_j\leq s\}$, $R_2(j,s)=\{X|X_j>s\}$
Determination of the best pair (j,s) is feasible.

2. Repeat 1) on each of the two resulting regions
3. Stop when some criterion is reached

Regression Trees

How large should we grow a tree?
Tree size is a tuning parameter governing the model's complexity.
It should be chosen adaptively from the data.
Strategy 1: Split tree node only, if decrease in rss is sufficiently high (but too short-sighted)
Strategy 2: (preferred)

- Given a large tree, stopping when some minimal node size is reached
- Prune the tree by *cost-complexity pruning*

Regression Trees

Subtree $T \subset T_0$ is any tree that can be obtained by pruning T_0, i.e. collapsing any number of its internal nodes. We denote the terminal nodes by $m = 1, \ldots, M$.
$N_m = |\{x_i \in R_m\}|$
$\hat{c}_m = 1/N_m \sum_{x_i in R_m} y_i$
$Q_m(T) = \sum_{x_i in R_m} (y_i - \hat{c}_m)^2$
Cost-complexity criterion $C_\alpha(T) = \sum_{m=1}^{M} N_m Q_m(T) + \alpha|T|$

Regression Trees

Find $T_\alpha \subset T_0$ to minimize $C_\alpha(T)$
α governs the trade-off between tree-size and its goodness of fit to the data.
$\alpha = 0$ yields full tree.
Choice of α by cross validation.
Final tree $T_{\hat{\alpha}}$

Bagging

- Bagging: Bootstrap aggregation or bagging averages
- Training data $Z = \{(x_1, y_1), \dots, (x_n, y_n)\}$
- Fit a model to Z and obtain $\hat{f}(x)$
- Idea: average the predictions over a collection of bootstrap samples, thereby reducing its variance
-
- $Z^{*b}, b = 1, \dots, B$ bootstrap samples
- Fit model to get $\hat{f}^{*,b}(x)$
- $\hat{f}_{bag}(x) = 1/B \sum_{b=1}^{B} \hat{f}^{*,b}(x)$

Bagging

- $\hat{f}_{bag}(x)$ estimate of the true bagging value $E_{\hat{\mathcal{P}}}\hat{f}^*(x)$
- Well suited for high-variance, low-bias procedures
- Application: regression trees

Random Forests

- Introduced by Breiman (2001)
- Very powerful (good performance) in many applications
- Modified version of bagging
- Idea: Building a large collection of de-correlated trees and then average them (Breiman, 2001)

Random Forests

- Trees: low bias, but very noisy / high variance $\Rightarrow$ goal : reduction of variance
- Trees generated by bagging are identically distributed, but not necessarily independent
- For identical distributed variables with positive pairwise correlation ρ: variance of average $\rho\sigma^2 + (1-\rho)/B\sigma^2$ (ρ correlation of the trees) (for i.i.d. rvs: σ^2/B)
- Application: nonlinear estimators like random trees

Random Forests | Procedure

- Bootstrap samples $1, \ldots, B$
- Build trees and
  "Before each split, select $m \leq p$ of the input variables at random as candidates for splitting" (e.g. $m = \sqrt{p}, m = 1$)
- Aggregation:
  $$\hat{f}_{rf}(x) = \frac{1}{B} \sum_{b=1}^{B} T(x; \Theta_b)$$
  Θ_b: split variables, cut points, terminal node values for b

Ensemble Learning | Introduction

- Idea: To build a prediction model by combing the strengths of a collection of simpler base models.
- Bagging, random forests are ensemble methods for classification, where a committee of trees each cast a vote for predicted class.
- Boosting proposed as a committee method where the committee of weak learners evolves over time with a weighted vote for the members
- Ensemble learning
- Developing a population of base learners from the training data
- Combining them to form the composite predictor

Learning Ensembles

- We consider functions of the form

$$f(x) = \alpha_0 + \sum_{T_k \in \mathcal{T}} \alpha_k T_k(x)$$

  with $\mathcal{T}$ dictionary of basis functions, e.g. trees with $|\mathcal{T}|$ quite large.
- Hybrid approach of Friedman and Popescu (2003)
- A finite dictionary $\mathcal{T}_L = \{T_1(x), \ldots, T_M(x)\}$ of basis functions is induced from the training data.
- A family of functions $f_\lambda(x)$ is built by fitting a lasso path in this dictionary

$$\alpha(\lambda) = argmin_\alpha \sum_{i=1}^{N} L[y_i, \alpha_0 + \sum_{m=1}^{M} \alpha_, T_m(x_i)] + \lambda \sum_{m=1}^{M} |\alpha_m|$$

Ensemble Generating Algorithm

How to choose the set of base functions $b(x;\gamma)$ forming $\mathcal{T}_L$?

- $f_0(x) = \arg\min_c \sum_{i=1}^{N} L(y_i, c)$
- For $m = 1$ to M do
- $\gamma_m = \arg\min_\gamma \sum_{i \in S_m(\eta)} L(y_i, f_{m-1}(x_i) + b(x_i;\gamma))$
- $f_m(x) = f_{m-1}(x) + \nu b(x;\gamma_m)$
- $\mathcal{T}_{ISLE} = \{b(x;\gamma_1), \ldots, b(x;\gamma_M)\}$

$S_m(\eta)$ refers to a subsample of $N\eta$ of the training observations, typically without replacement.
Recommendation: $\eta \leq 1/2$ and $\eta \sim 1/\sqrt{(}N)$, $\nu = 0.1$
Importance sampled learning ensemble (ISLE)

Special cases of the Algorithm

- Bagging: $\eta = 1$, samples with replacement, $\nu = 0$
- Random forest: sampling is similar with more randomness introduced by the selection of the splitting variable.
- Gradient boosting with shrinkage uses $\eta = 1$
- Stochastic gradient boosting: identical