Random Forests and Boosting

Christoph Kern

Mannheim Machine Learning Modules

c.kern@uni-mannheim.de

March 21 and 22, 2018

UNIVERSITY
OF MANNHEIM

Outline

Introduction

Some limitations of (single) trees

- Difficulties in modeling additive structures
- Lack of smoothness of prediction surface
- High variance / **instability** due to hierarchical splitting process

$\rightarrow$ **Ensemble methods**

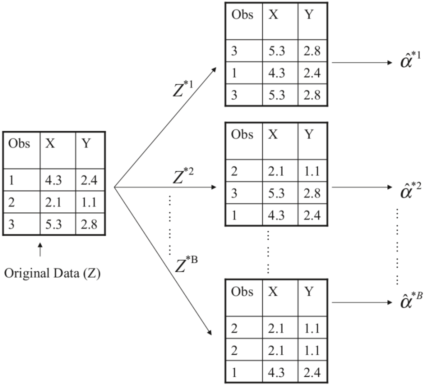
- Address instability via combining multiple prediction models
- Combine diverse models into a more robust ensemble

How to construct ensembles?

- Combine models based on different methods
  - Stacking: Build a meta-model that uses (multiple) predictions as input
- Apply one method with different tuning parameter settings
- Combine models with different features
- Use one method with different subsets of the data
  - **Bagging**: Can be applied to different **base learners** (e.g. CART)

Bagging

Figure: Bootstrap process



James et al. (2013)

Bootstrap: Sampling B samples of size n with replacement from original data set

Applications

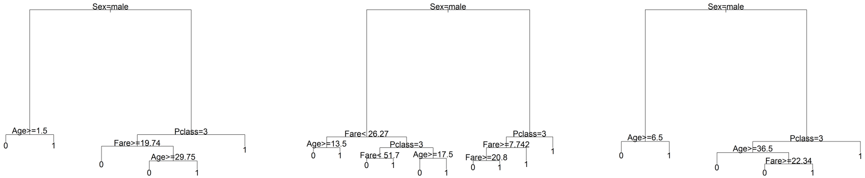
- Estimate the variability of model parameters
    - e.g. standard errors of regression coefficients
- Estimate test error with training data
    - Fit model on bootstrap samples and predict original training set
- Construct an ensemble of learners for prediction
    - **Bagging**: Bootstrap Aggregating
    - Train prediction models on bootstrap samples

Algorithm 1: Bagging Trees

```
Set number of trees B;
Define stopping criteria;
for b = 1 to B do
    draw a bootstrap sample from the training data;
    assign sampled data to root node;
    if stopping criterion is reached then
        end splitting;
    else
        find the optimal split point among the predictor space;
        split node into two subnodes at this split point;
        for each node of the current tree do
            continue tree growing process;
        end
    end
end
```

Figure: Bagging Trees

(a) b = 1 (b) b = 2 (c) b = 3



Random Forests

From Bagging to Random Forests

Variance of an average of B i.i.d. random variables

$$\frac{1}{B}\sigma^2$$

$\rightarrow$ Bagging: Averaging over B trees decreases variance

Variance of an average of B i.d. random variables with $\rho > 0$

$$\rho\sigma^2 + \frac{1-\rho}{B}\sigma^2$$

$\rightarrow$ **Random Forests**: Averaging over B trees with m out of p predictors per split decreases variance and decorrelates trees

Random Forests

From Bagging to Random Forests

Variance of an average of B i.i.d. random variables

$$\frac{1}{B}\sigma^2$$

$\rightarrow$ Bagging: Averaging over B trees decreases variance

Variance of an average of B i.d. random variables with $\rho > 0$

$$\rho\sigma^2 + \frac{1-\rho}{B}\sigma^2$$

$\rightarrow$ **Random Forests**: Averaging over B trees with m out of p predictors per split decreases variance and decorrelates trees

The Random Forest trick

- Randomization with respect to rows *and* columns
- Weaker predictors have more of a chance
- Results in diverse and *decorrelated* trees

Can be taken one step further...

1. Draw a random sample m from the p predictors (w/o Bootstrapping)
2. Draw random split(s) per feature
3. Split node using the best of these random splits

$\rightarrow$ Extremely Randomized Trees (Geurts et al. 2006)

Growing a Forest

Algorithm 2: Grow a Random Forest

```
Set number of trees B;
Set predictor subset size m;
Define stopping criteria;
for b = 1 to B do
    draw a bootstrap sample from the training data;
    assign sampled data to root node;
    if stopping criterion is reached then
        end splitting;
    else
        draw a random sample m from the p predictors;
        find the optimal split point among m;
        split node into two subnodes at this split point;
        for each node of the current tree do
            continue tree growing process;
        end
    end
end
```

A Random Forest

$$\{T_b\}_1^B$$

consists of a set of $b = 1, 2, \ldots, B$ trees which can be used for prediction by...

- Regression
  - ...averaging predictions over all trees
  - $\hat{f}_{rf}^B(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x)$
- Classification
  - ...using most commonly occurring class among all trees
  - $\hat{C}_{rf}^B(x) = \text{majority vote}\{\hat{C}_b(x)\}_1^B$

Observations in each bootstrap sample

$$\begin{aligned} P(\text{obs } i \in \text{ sample } b) &= 1 - \left(1 - \frac{1}{n}\right)^n \\ &\approx 1 - e^{-1} \\ &= 0.632 \end{aligned}$$

Out-of-bag (OOB) error

- Sampling with replacement leads to models based on subsets of the data
- Unused (OOB) observations can be used for test error estimation
  1. Generate predictions for case i using models where i was OOB
  2. Average predictions for i and estimate test error
  3. Compute OOB error over all cases

Tuning Random Forests

- Predictor subset size m out of p
  - Most important tuning parameter in RF
  - Starting value; $m = \sqrt{p}$ (classification), $m = p/3$ (regression)
  - Can be chosen using OOB errors based on different m
- Number of trees
  - sufficiently high (e.g. 500)
- Node size (number of observations in terminal nodes)
  - sufficiently low (e.g. 5)

Interpretation

Interpreting Random Forests

- Inspect each tree of the forest
    - Inefficient for 500+ trees
- Variable importance
    - Summary of "effect size"
- Partial dependence plots
    - Graphical representation of "effect structure"

Variable importance with CART

$$\mathcal{I}_\ell^2(T) = \sum_{t=1}^{J-1} \hat{\imath}_t^2 I(v(t) = \ell)$$

- Sum of squared improvements $\hat{\imath}^2$ over all internal nodes with predictor X_ℓ
  - Regression: Overall reduction in RSS caused by X_ℓ
  - Classification: Overall reduction of impurity caused by X_ℓ

Importance with Random Forests

$$\mathcal{I}_\ell^2 = \frac{1}{M} \sum_{m=1}^{M} \mathcal{I}_\ell^2(T_m)$$

- Average improvement caused by predictor X_ℓ over all trees

Partial dependence plots

$$\tilde{f}(x) = \frac{1}{n}\sum_{i=1}^{n} f(x, x_{iC})$$

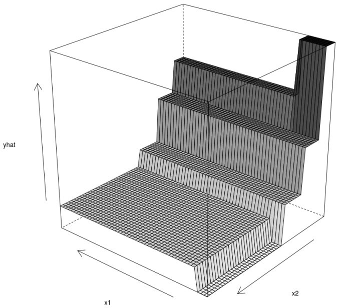
- Goal: Plot results from “black box” learning methods
- Compute $\tilde{f}(x)$ over the range of x while averaging the effects of the remaining predictors x_{iC}
- Generate artificial datasets by fixing x for all cases
  - Regression: Averaging over $f(x, x_{iC})$ for each value of x
  - Classification: Averaging over $\text{logit}(p)$ for each value of x
- Outlook: ICE plots (Goldstein et al. 2014)

Generating PDP's

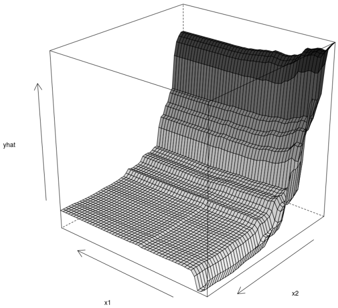
1. Choose a range of values $\{x_{11}, x_{12}, \ldots, x_{1k}\}$ of x_1
2. For each $i \in \{1, 2, \ldots, k\}$
    1. Generate an artificial dataset by fixing x_1 to x_{1i} for all cases
    2. Compute predictions for all cases using the prediction model (e.g. RF)
    3. Average the predictions over all cases
3. Plot the obtained average predictions against x_{1i} for $i = 1, 2, \ldots, k$

Figure: Partial dependence plots

(a) CART



(b) Random Forest



Boosting

Boosting

- Class of ensemble methods which combine **sequential** prediction models
- Adaptive approach with focus on "difficult observations"
- Different flavors exist
  - AdaBoost
  - Gradient Boosting Machines (GBM)
  - ...
- Can be applied to different (weak) base learners
  - Boosting trees
  - ...

AdaBoost

AdaBoost

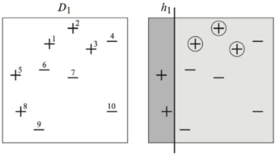
- Algorithm for classification problems ($Y \in \{-1, 1\}$)
- Estimate a sequence of classifiers using reweighted data
- AdaBoost process
  1. Fit classifier $G_m(x)$ to weighted data (intitial weights $w_i = \frac{1}{n}$)
  2. Compute the misclassification rate

$$\text{err}_m = \frac{\sum_{i=1}^{n} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{n} w_i}$$

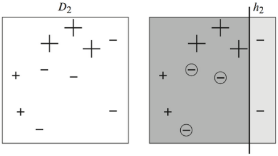
  3. Compute the classifier weight $\alpha_m = \log((1 - \text{err}_m)/\text{err}_m)$
  4. Recalculate weights $w_i = w_i \exp(\alpha_m I(y_i \neq G_m(x_i)))$
- Majority vote classification: $G(x) = \text{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$

Figure: (Ada)Boosting stumps (example)[1]

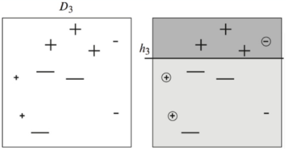
(a) Step 1: $\alpha_1 = 0.42$



(b) Step 2: $\alpha_2 = 0.65$

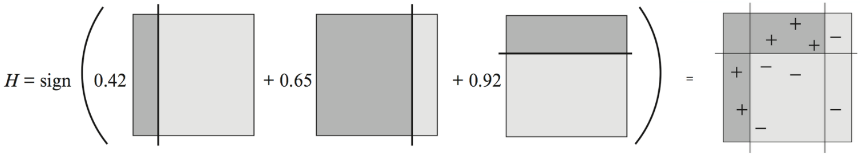


(c) Step 3: $\alpha_3 = 0.92$



[1] http://www.ccs.neu.edu/home/vip/teach/MLcourse/4_boosting/slides/boosting.pdf

Figure: Step 4: Combine models



GBM

Gradient Boosting Machines (GBM)

- General approach to sequential learning
- Applicable with various loss functions
- Boosting trees
  1. Initialize model (with a constant $f_0(x)$)
  2. Compute pseudo-residuals based on current model

$$r_{im} = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f=f_{m-1}}$$

  3. Fit a regression tree to the pseudo-residuals
  4. Compute $\gamma_{jm} = \arg\min_\gamma \sum_{x_i \in R_{jm}} L(y_i, f_{m-1}(x_i) + \gamma)$
  5. Update the current model: $f_m(x) = f_{m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$
- Output $\hat{f}(x) = f_M(x)$

$\rightarrow$ Analogue to steepest descent

Table: GBM components for different loss functions

Setting	Loss function	r_i	$f_0(x)$
Regression	$\frac{1}{2}(y_i - f(x_i))^2$	$y_i - f(x_i)$	mean(y_i)
Regression	$\|y_i - f(x_i)\|$	sign$(y_i - f(x_i))$	median(y_i)
Classification	Deviance	$I(y_i = G_k) - p_k(x_i)$	prior p's

Shrinkage

- Additional tweak in Gradient boosting
- Slow down learning rate to avoid overfitting
- Learning rate is controlled by λ
    - $f_m(x) = f_{m-1}(x) + \lambda \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$

Subsampling

- Optional add-on in Gradient boosting
- Use a random sample (w/o replacement) of pseudo-residuals in each step
- Can be introduced to improve performance and speed
    - "Stochastic gradient boosting"

Algorithm 3: Gradient Boosting for regression

Set number of trees M;
Set interaction depth D;
Set shrinkage parameter λ;
Use $\bar{y}$ as initial prediction;
for $m = 1$ *to* M **do**
    compute residuals based on current predictions;
    assign data to root node, using the residuals as the outcome;
    while *current tree depth* $< D$ **do**
        tree growing process;
    end
    compute the predicted values of the current tree;
    add the shrinked new predictions to the previous predicted values;
end

Table: Gradient Boosting with 5 obs and 2 x's (example)[2]

ID	x_1	x_2	y	$f_0(x)$
1	0	0	1	1.2
2	0	2	3	1.2
3	1	2	2	1.2
4	2	3	0	1.2
5	0	1	0	1.2

[2] Schonlau, M. (2015). Statistical Learning with Boosting. ESRA short course.

Table: Step 1: Split $x_2 > 2.5$

ID	x_1	x_2	y	$f_0(x)$	r_{i1}	γ_{j1}	$f_1(x)$
1	0	0	1	1.2	–0.2	0.3	1.5
2	0	2	3	1.2	1.8	0.3	1.5
3	1	2	2	1.2	0.8	0.3	1.5
4	2	3	0	1.2	–1.2	–1.2	0
5	0	1	0	1.2	–1.2	0.3	1.5

Table: Step 2: Split $x_2 < 1.5$

ID	x_1	x_2	y	$f_0(x)$	$f_1(x)$	r_{i2}	γ_{j2}	$f_2(x)$
1	0	0	1	1.2	1.5	−0.5	−1	0.5
2	0	2	3	1.2	1.5	1.5	0.66	2.166
3	1	2	2	1.2	1.5	0.5	0.66	2.166
4	2	3	0	1.2	0	0	0.66	0.66
5	0	1	0	1.2	1.5	−1.5	−1	0.5

Tuning Gradient Boosting Machines

- Number of trees M
  - Number of "iterations"
  - Overfitting can occur for large M
- Interaction depth D
  - Number of splits for each tree
  - Boosting stumps: $D = 1$
- Shrinkage parameter λ
  - e.g. $\lambda = 0.01$, $\lambda = 0.001$
  - Smaller λ needs larger M
- ...

Summary

- Ensemble methods combine multiple models to stabilize predictions
- RF and Boosting are competitive “general purpose” approaches
- A lot of different flavors exist
- Algorithms typically compared in a large train and tune loop
- Drawbacks: Lower interpretability and higher computational costs

Software Resources

Resources for R

- Standard package to grow RFs: `randomForest`
- Extremely Randomized Trees: `extraTrees`
- Ensembles of Conditional Inference Trees: `cforest`
- Gradient Boosting: `gbm`
- Extreme Gradient Boosting: `xgboost`
- Visualization
  - Partial Dependence Plots: `pdp`
  - Plot model surfaces (also PDPs): `plotmo`

References

Berk, R. A. (2006). An Introduction to Ensemble Methods for Data Analysis. *Sociological Methods & Research, 34*(3), 263–295.

Biau, G., Scornet, E. (2015). *A Random Forest Guided Tour*. arXiv: 1511.05741.

Friedman, J. (2001). Greedy Function Approximation: A Gradient Boosting Machine. *The Annals of Statistics, 29*(5), 1189–1232.

Geurts, P., Ernst, D., Wehenkel, L. (2006). Extremely Randomized Trees. *Machine Learning 63*(1), 3–42.

Goldstein, A., Kapelner, A., Bleich, J., Pitkin, E. (2014). *Peeking Inside the Black Box: Visualizing Statistical Learning with Plots of Individual Conditional Expectation*. arXiv: 1309.6392v2.

James, G., Witten, D., Hastie, T., Tibshirani, R. (2013). *An Introduction to Statistical Learning*. New York, NY: Springer.