DOTE 6635: Artificial Intelligence for Business Research

Machine Learning Basics

Renyu (Philip) Zhang

1

Agenda

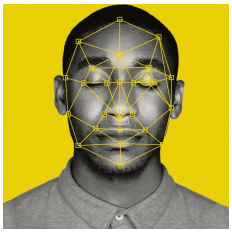
- Supervised learning model evaluation and model selection
- K Nearest Neighbors
- Decision Tree
- Ensemble Methods

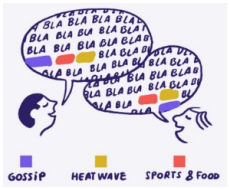
2

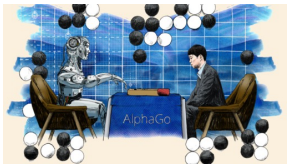
2

Different Types of ML

- Supervised learning
  - Data=(Label, Feature): Given new features, predict the label.
  - E.g., photo recognition.
- Unsupervised learning
  - Data=Feature: Output certain pattern(s) from the data.
  - E.g., topic modeling.
- Reinforcement learning:
  - Data=A series interactions between the agent and the environment
  - Select actions to maximize long-term reward.
  - E.g., AlphaGo.
- Generative AI
  - Use patterns learnt from (un)supervised learning to generate data.
  - E.g., Large language models or stable diffusion







3

3

Supervised Learning

- Model: $\mathrm{Y} = f(X) + \epsilon$
- Data Observations: $(\mathrm{X}, \mathrm{Y})_i, i = 1,2, \ldots, n$
- Objective: To estimate $f(\cdot)$
- Why do we care about estimating $f(\cdot)$?
  - Prediction: Given an unknown X, predict $Y = f(X)$
  - Inference: How is Y changing with X?
- How do we estimate $f(\cdot)$?
  - Identify $f(\cdot)$ in the model class (e.g., linear regression, trees, neural nets, etc.) which minimizes the average error of prediction $f(X)$ compared with the true Y for the data.
  - Error is quantified by a loss function: $\mathrm{L}(Y, X)$

4

4

Supervised Learning in Action

- Given the data observations: $(\mathrm{X}, \mathrm{Y})_i, i = 1,2, \ldots, n$
- Define your model
  - Linear regression: $\mathrm{Y} = a + b \cdot X + \epsilon$
- Define your loss function:
  - Squared error: $(Y - f(X))^2$
- Pick your optimizer
  - OLS estimator
  - Gradient descent
- Run your model on a CPU/GPU Cluster

Different Model Types:

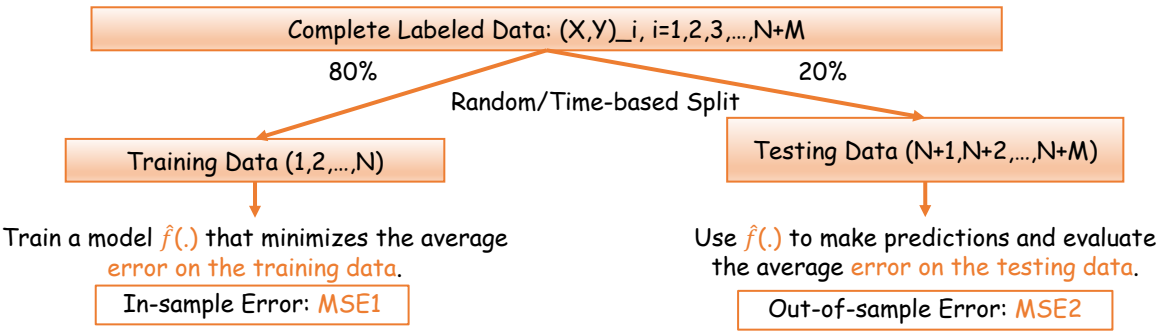
- Parametric model (OLS)
- Nonparametric model (kNN)
- Semiparametric model (Cox Proportional-Hazards Model, DML)

5

5

Supervised Learning Model Evaluation

- Idea: Reserve some data the model has never seen to judge its performance.
- Define the metric: MSE=E[Y-f(X)]^2



- Question: Should we use the MSE1 or MSE2 judge the performance of a model?

6

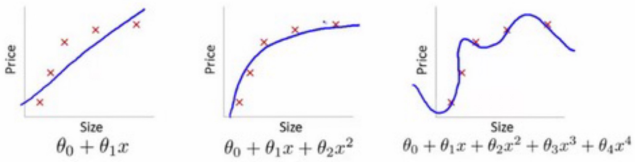
6

Bias-Variance Trade-off

- Assumption: Y = f(X) + epsilon, where epsilon is the zero-mean error.
  - f(.) is unknown and we want to learn from data. $\mathcal{D}$ is the available data set.

$$\underbrace{\mathbb{E}_{\mathcal{D}}[(Y-\hat{f}(X,\mathcal{D}))^2]}_{\text{Error Conditioned on } X} = \underbrace{(\mathbb{E}_{\mathcal{D}}[\hat{f}(X,\mathcal{D})]-f(X))^2}_{\text{Bias}} + \underbrace{\mathbb{E}_{\mathcal{D}}[(\mathbb{E}_{\mathcal{D}}[\hat{f}(X,\mathcal{D})]-\hat{f}(X,\mathcal{D}))^2]}_{\text{Variance}} + \underbrace{\mathbb{V}(\epsilon)}_{\text{Noise}}$$

$$\underbrace{\mathbb{E}_{X,\mathcal{D}}[(Y-\hat{f}(X,\mathcal{D}))^2]}_{\text{MSE}} = \mathbb{E}\Big\{\text{Bias}_{\mathcal{D}}[\hat{f}(X,\mathcal{D})] + \text{Variance}_{\mathcal{D}}[\hat{f}(X,\mathcal{D})]\Big\} + \mathbb{V}(\epsilon),$$



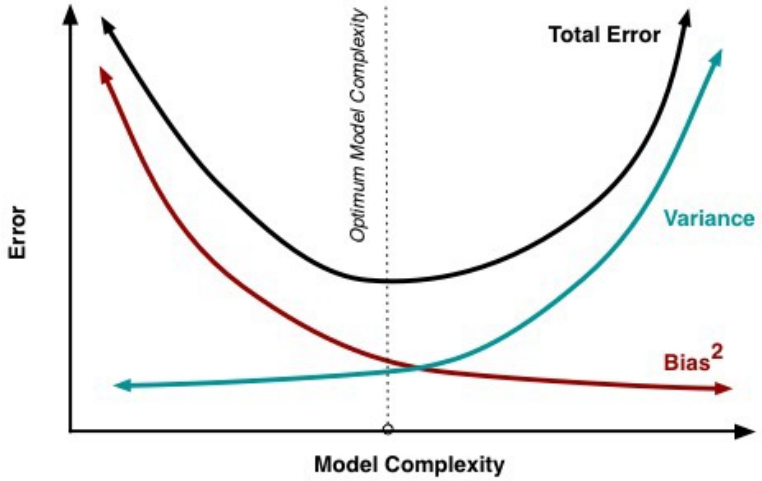
High bias (underfit) | "Just right" | High variance (overfit)

For a proof, see
https://en.wikipedia.org/wiki/Bias%E2%80%93variance_tradeoff

7

7

Bias-Variance Trade-off



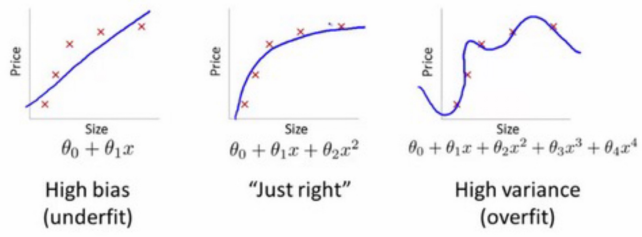
Big Assumption: The complexity of the model does not decrease the quality of model estimation.

8

8

Model Selection

- How do we utilize the bias-variance trade-off idea?
- Consider the problem Y is a polynomial of X.
  - But you do not know the actual degree.
- Hyper-parameter: Degree k.
- Parameter: The coefficients of the polynomial.



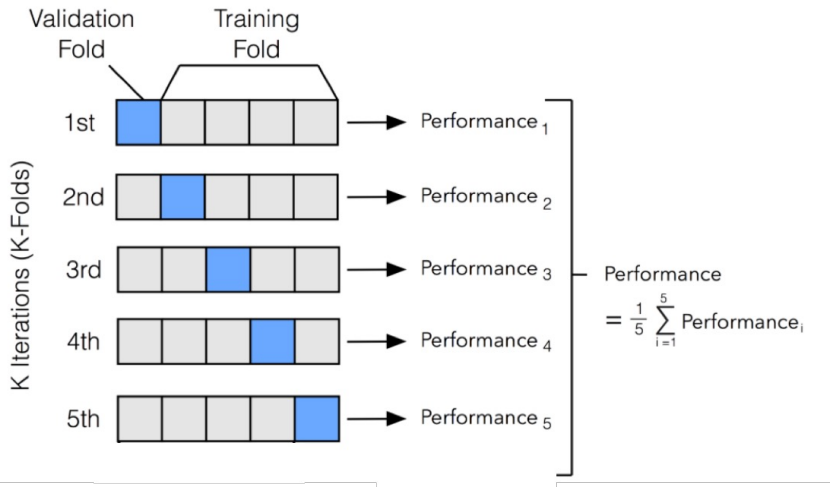
Training Data (1,2,…,N)	Validation Data (N+1,…,N+M)	Testing Data (N+M+1,…,N+M+P)
Find the coefficients	Find k	Evaluate model and report results

- **Question**: Data is valuable. How do we use less data to achieve this?

9

9

Cross Validation



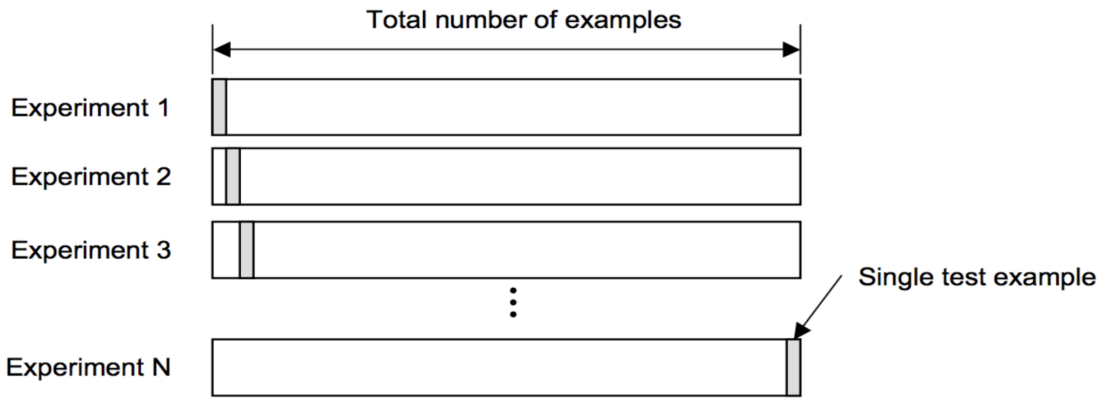
Cross validation combines the training and validation sets.

It provides the same level of statistical power in model selection but requires less amount of data than splitting training and validation sets.

10

10

Leave-one-out Cross Validation



- Leave-one-out is a special case of k-fold cross validation with k = the number of samples.
- Small bias, but computationally very expensive.

11

11

What is the Optimal k?

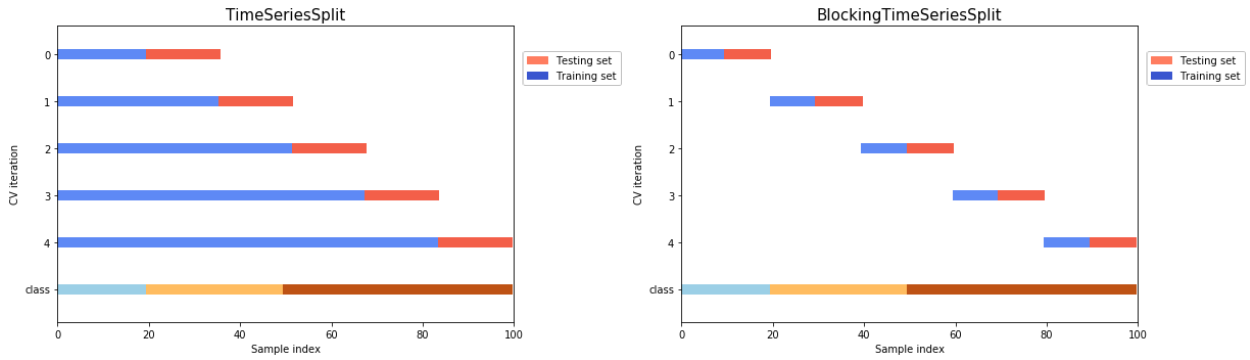
- With a large k:
  - Small bias vs. large computational time
- With a small k:
  - Small computational time vs. large bias
- In practice, the choice of k depends on sample size:
  - For a large data set, 3-fold will be quite accurate.
  - For very sparse data sets, leave-one-out may be a better choice.
- Common practice for k-fold cross validation: k = 5 ~ 10

12

12

Time-series Cross Validation

- Rolling Cross Validation vs. Blocking Cross Validation



13

13

Other Model Selection Methods

- Another way to select the optimal model is to find the mapping between training and testing standard errors and adjust the in-sample error to reflect the out-of-sample errors.
  - Adjusted R-squared for linear regression
  - Akaike Information Criterion
  - Bayesian Information Criterion

Linear Regression:

$$Y_i = \sum_j \alpha_j * X_{ij} + \epsilon_i$$

Residual Squared Error = RSS = $\sum_i (y_i - \sum_j \alpha_j * X_{ij})^2$
Total Squared Error = TSS = $\sum_i (y_i - \bar{y})^2$
R-squared = $1 - \frac{RSS}{TSS}$
Adjusted-R-squared = $1 - \frac{RSS/(n-d-1)}{TSS/(n-1)}$
Mallow's $C_p = \frac{1}{n}(RSS + 2d\sigma^2)$, σ is the estimate of the variance of $epilson$.
AIC = $-2logL + 2 * d$, L is the maximized likelihood.
BIC = $\frac{1}{n}(RSS + log(n)d\sigma^2)$

14

14

Cross Validation vs. Other Methods

- Advantage of cross validation:
  - Asymptotically optimal, producing the unbiased estimate of generalization error.
  - Does not require close-form equation of standard errors.
  - Even for models without likelihood functions (e.g., EM method), it is well-defined.

- Disadvantage of cross validation:
  - Computationally inefficient
  - Evaluates the prediction performance of the model, not the causal estimation accuracy.

- More and more popular in econometrics models:
  - Two-stage estimators such as double machine learning
  - Providing cross validation errors becomes a must for predictive analyses.
  - A way to do model selection in structural estimation as well.

15

15

Agenda

- Supervised learning model evaluation and model selection
- **K Nearest Neighbors**
- Decision Tree
- Ensemble Methods

16

16

Traditional Supervised Learning Models

- Models:
  - K-nearest neighbors
  - Decision tree
  - Ensemble methods
- For each model:
  - Algorithm
  - Implementation
  - Optimization/tuning

17

17

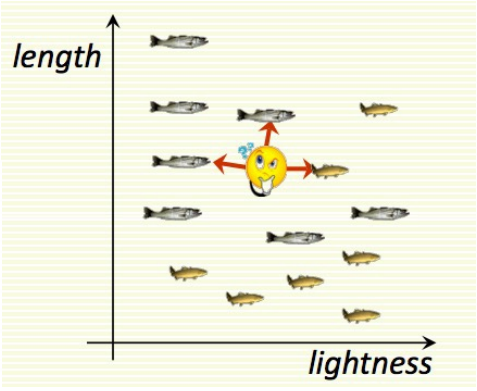
K Nearest Neighbors (k-NN)

- Predict the outcome of an unobserved sample based on its k closest observations in the training set.
- Need to define a proper distance metric (usually L2).
  - Feature normalization as pre-processing
- Formally:

Define N_0 as the set of K nearest neighbors in $(x_1, ..., x_N)$ of x_{N+1}:

$$\text{Classification: } \mathbb{P}(Y = j | X = x_{N+1}) = \frac{1}{k} \sum_{i \in N_0} I(y_i = j)$$

$$\text{Regression: } \mathbb{E}[Y | X = x_{N+1}] = \frac{1}{k} \sum_{i \in N_0} w_i(x_i) y_i,$$



where $w_i(x_i)$ is the weight of x_i. In most applications, the weight is set as $w_i(x_i) = 1$.

- K-NN makes provably accurate predictions when the training set size n is large.

18

18

K Nearest Neighbors

- A larger k means a simpler model, so high bias and low variance.
- Time complexity scales with data size:
  - Training: O(1)
  - Testing/Prediction: O(n*d*k)
  - Nonparametric model
- Curse of dimensionality:
  - The necessary training sample size n scales exponentially with dimension d.
- Addressing these issues: Decision Tree.

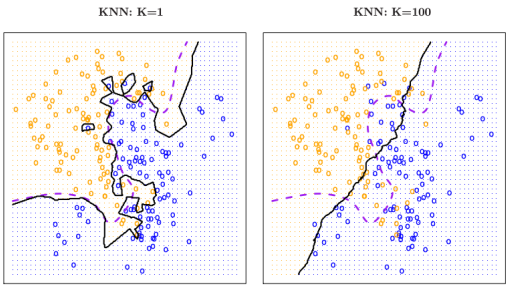


FIGURE 2.16. *A comparison of the KNN decision boundaries (solid black curves) obtained using $K = 1$ and $K = 100$ on the data from Figure 2.13. With $K = 1$, the decision boundary is overly flexible, while with $K = 100$ it is not sufficiently flexible. The Bayes decision boundary is shown as a purple dashed line.*

19

19

K Nearest Neighbors Implementation

- kNN Implementation know-hows:
  - In general, p-norm penalizes one bad dimension much more if p is larger. Choose your distance metric based on the setting you look at.
  - Remember to standardize the features, especially when n is large.
  - If d is large, use some feature selection approach, such as PCA.
  - The hyper-parameter k should be selected based on cross-validation.

JOURNAL OF THE AMERICAN STATISTICAL ASSOCIATION
2022, VOL. 00, NO. 0, 1–11: Theory and Methods
https://doi.org/10.1080/01621459.2022.2115375

Taylor & Francis
Taylor & Francis Group

Check for updates

Optimal Nonparametric Inference with Two-Scale Distributional Nearest Neighbors

Emre Demirkaya[a], Yingying Fan[b], Lan Gao[a,b], Jinchi Lv[b], Patrick Vossler[b], and Jingbo Wang[c]

[a]University of Tennessee Knoxville, Knoxville, TN; [b]University of Southern California, Los Angeles, CA; [c]The Chinese University of Hong Kong, Hong Kong

ABSTRACT
The weighted nearest neighbors (WNN) estimator has been popularly used as a flexible and easy-to-implement nonparametric tool for mean regression estimation. The bagging technique is an elegant way to form WNN estimators with weights automatically generated to the nearest neighbors (Steele 2009; Biau, Cérou, and Guyader 2010); we name the resulting estimator as the distributional nearest neighbors (DNN) for easy reference. Yet, there is a lack of distributional results for such estimator, limiting its application to statistical inference. Moreover, when the mean regression function has higher-order smoothness, DNN does not achieve the optimal nonparametric convergence rate, mainly because of the bias issue. In this work, we provide an in-depth technical analysis of the DNN, based on which we suggest a bias reduction approach for the DNN estimator by linearly combining two DNN estimators with different subsampling scales, resulting in the novel two-scale DNN (TDNN) estimator. The two-scale DNN estimator has an equivalent representation of WNN with weights admitting explicit forms and some being negative. We prove that, thanks to the use of negative weights, the two-scale DNN estimator enjoys the optimal nonparametric rate of convergence in estimating the regression function under the fourth-order smoothness condition. We further go beyond estimation and establish that the DNN and two-scale DNN are both asymptotically normal as the subsampling scales and sample size diverge to infinity. For the practical implementation, we also provide variance estimators and a distribution estimator using the jackknife and bootstrap techniques for the two-scale DNN. These estimators can be exploited for constructing valid confidence intervals for nonparametric inference of the regression function. The theoretical results and appealing finite-sample performance of the suggested two-scale DNN method are illustrated with several simulation examples and a real data application.

ARTICLE HISTORY
Received January 2021
Accepted July 2022

KEYWORDS
Bagging; Bootstrap and jackknife; *k*-nearest neighbors; Nonparametric estimation and inference; Two-scale distributional nearest neighbors; Weighted nearest neighbors

kNN is also an important tool in ML-based causal inference.

20

20

Agenda

- Supervised learning model evaluation and model selection
- K Nearest Neighbors
- **Decision Tree**
- Ensemble Methods

21

Decision Tree

- 40 data points
- Goal: predict MPG
- Need to find: $f : X \rightarrow Y$
- Discrete data (for now)

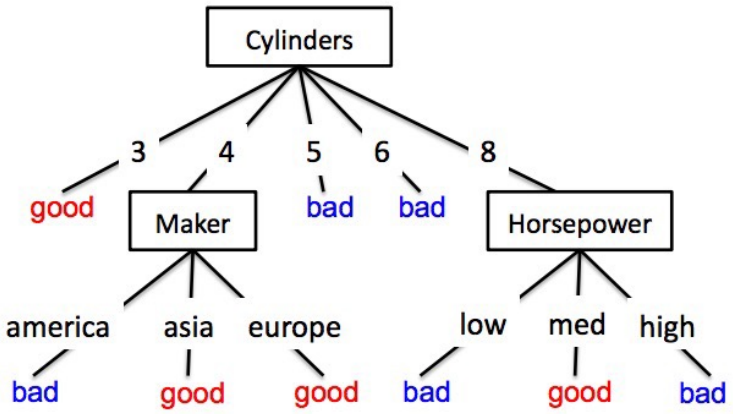
mpg	cylinders	displacement	horsepower	weight	acceleration	modelyear	maker
good	4	low	low	low	high	75to78	asia
bad	6	medium	medium	medium	medium	70to74	america
bad	4	medium	medium	medium	low	75to78	europe
bad	8	high	high	high	low	70to74	america
bad	6	medium	medium	medium	medium	70to74	america
bad	4	low	medium	low	medium	70to74	asia
bad	4	low	medium	low	low	70to74	asia
bad	8	high	high	high	low	75to78	america
:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:
bad	8	high	high	high	low	70to74	america
good	8	high	medium	high	high	79to83	america
bad	8	high	high	high	low	75to78	america
good	4	low	low	low	low	79to83	america
bad	6	medium	medium	medium	high	75to78	america
good	4	medium	low	low	low	79to83	america
good	4	low	low	medium	high	79to83	america
bad	8	high	high	high	low	70to74	america
good	4	low	medium	low	medium	75to78	europe
bad	5	medium	medium	medium	medium	75to78	europe

Y X

22

Decision Tree

- Each internal node is split based on one feature.
- Each leaf node is assigned with one Y.
- Benefits:
  - A smart data structure to scale kNN;
  - Flexible and large space of functions;
  - Human interpretation.

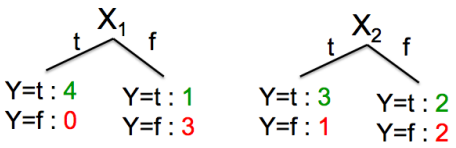


23

23

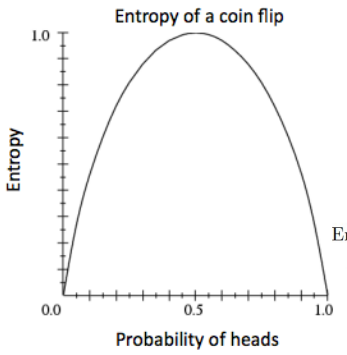
Fitting a Decision Tree

- Given a training set, identifying the best decision tree in NP-complete.



On which feature should we split?

- Heuristics to build a tree:
  1. Start with an empty tree;
  2. Split on the feature that gives the largest reduction in impurity;
  3. Repeat step 2 until a stopping criterion is met.
  4. Assign the major class (classification) or the average outcome to each leaf (regression).



Entropy: $H(Y) = -\sum_{i=1}^{k} \mathbb{P}(Y = y_i) log_2 \mathbb{P}(Y = y_i)$

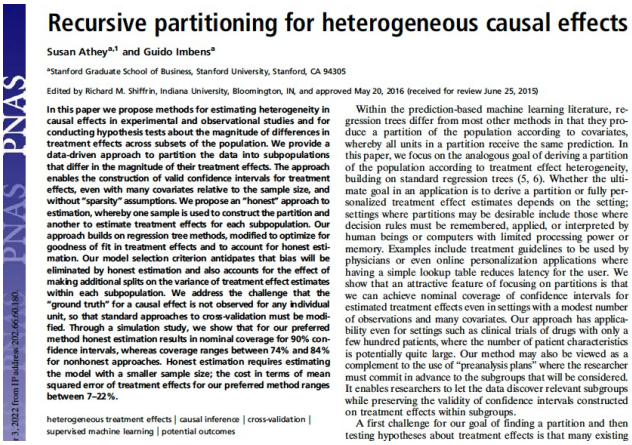
- Measurement of impurity:
  - **Classification:** Gini index; Entropy
  - **Regression:** Squared error

24

24

Decision Tree Implementation

- Decision Tree Implementation know-hows:
  - Decision trees tend to overfit, so fine-tune the hyper-parameters to strike a balance.
  - Use PCA or other feature selection methods to select features before fitting your tree.
  - For classification tree, balance your data before training (down sampling, over sampling, SMOTE, etc.).



PNAS

Recursive partitioning for heterogeneous causal effects

Susan Athey[a,1] and Guido Imbens[a]

[a]Stanford Graduate School of Business, Stanford University, Stanford, CA 94305

Edited by Richard M. Shiffrin, Indiana University, Bloomington, IN, and approved May 20, 2016 (received for review June 25, 2015)

In this paper we propose methods for estimating heterogeneity in causal effects in experimental and observational studies and for conducting hypothesis tests about the magnitude of differences in treatment effects across subsets of the population. We provide a data-driven approach to partition the data into subpopulations that differ in the magnitude of their treatment effects. The approach enables the construction of valid confidence intervals for treatment effects, even with many covariates relative to the sample size, and without "sparsity" assumptions. We propose an "honest" approach to estimation, whereby one sample is used to construct the partition and another to estimate treatment effects for each subpopulation. Our approach builds on regression tree methods, modified to optimize for goodness of fit in treatment effects and to account for honest estimation. Our model selection criterion anticipates that bias will be eliminated by honest estimation and also accounts for the effect of making additional splits on the variance of treatment effect estimates within each subpopulation. We address the challenge that the "ground truth" for a causal effect is not observed for any individual unit, so that standard approaches to cross-validation must be modified. Through a simulation study, we show that for our preferred method honest estimation results in nominal coverage for 90% confidence intervals, whereas coverage ranges between 74% and 84% for nonhonest approaches. Honest estimation requires estimating the model with a smaller sample size; the cost in terms of mean squared error of treatment effects for our preferred method ranges between 7–22%.

heterogeneous treatment effects | causal inference | cross-validation | supervised machine learning | potential outcomes

Within the prediction-based machine learning literature, regression trees differ from most other methods in that they produce a partition of the population according to covariates, whereby all units in a partition receive the same prediction. In this paper, we focus on the analogous goal of deriving a partition of the population according to treatment effect heterogeneity, building on standard regression trees (5, 6). Whether the ultimate goal in an application is to derive a partition or fully personalized treatment effect estimates depends on the setting; settings where partitions may be desirable include those where decision rules must be remembered, applied, or interpreted by human beings or computers with limited processing power or memory. Examples include treatment guidelines to be used by physicians or even online personalization applications where having a simple lookup table reduces latency for the user. We show that an attractive feature of focusing on partitions is that we can achieve nominal coverage of confidence intervals for estimated treatment effects even in settings with a modest number of observations and many covariates. Our approach has applicability even for settings such as clinical trials of drugs with only a few hundred patients, where the number of patient characteristics is potentially quite large. Our method may also be viewed as a complement to the use of "preanalysis plans" where the researcher must commit in advance to the subgroups that will be considered. It enables researchers to let the data discover relevant subgroups while preserving the validity of confidence intervals constructed on treatment effects within subgroups.

A first challenge for our goal of finding a partition and then testing hypotheses about treatment effects is that many existing

Decision tree is also an important tool in ML-based causal inference.

25

25

Agenda

- Supervised learning model evaluation and model selection
- K Nearest Neighbors
- Decision Tree
- **Ensemble Methods**

26

26

Bagging (Bootstrap Aggregation)

- Decision trees tend to overfit. So how should we address this issue?
- Averaging reduces variance (i.e., pooling):

$$Var(\overline{X}) = \frac{Var(X)}{N}$$ (when predictions are **independent**)

- Average models to reduce the variance of predictions.
  - But we only have one training set….. How to construct multiple independent models?
- **Bootstrap**: Given a data set of sample size n, a bootstrap sample is created by sampling n instances uniformly random from the data **with replacement**.
  - Since $(1 - {}^{1}/_{n})^{n} \approx e^{-1} = 0.368$, each sample of the original data has 0.632 probability to be sampled.

27

27

Bootstrapping

- Bootstrapping is a general (and powerful) method of statistical inference to build a distribution for a statistic by resampling from the available data.
- Bootstrapping is very useful when the estimator's variance has no closed-form.
  - Machine/Deep learning models
  - Structural estimations
  - Complex reduced-form clustered standard errors

The Annals of Statistics
1979, Vol. 7, No. 1, 1–26

THE 1977 RIETZ LECTURE

BOOTSTRAP METHODS: ANOTHER LOOK AT THE JACKKNIFE

BY B. EFRON
Stanford University

We discuss the following problem: given a random sample $\mathbf{X} = (X_1, X_2, \ldots, X_n)$ from an unknown probability distribution F, estimate the sampling distribution of some prespecified random variable $R(\mathbf{X}, F)$, on the basis of the observed data $\mathbf{x}$. (Standard jackknife theory gives an approximate mean and variance in the case $R(\mathbf{X}, F) = \theta(\hat{F}) - \theta(F)$, θ some parameter of interest.) A general method, called the "bootstrap," is introduced, and shown to work satisfactorily on a variety of estimation problems. The jackknife is shown to be a linear approximation method for the bootstrap. The exposition proceeds by a series of examples: variance of the sample median, error rates in a linear discriminant analysis, ratio estimation, estimating regression parameters, etc.

28

28

Bootstrapping Variance Estimator

Bootstrap Variance Estimator

1. Draw a bootstrap sample $X_1^*, \ldots, X_n^* \sim P_n$. Compute $\widehat{\theta}_n^* = g(X_1^*, \ldots, X_n^*)$.
2. Repeat the previous step, B times, yielding estimators $\widehat{\theta}_{n,1}^*, \ldots, \widehat{\theta}_{n,B}^*$.
3. Compute:
$$\widehat{s} = \sqrt{\frac{1}{B}\sum_{j=1}^{B}(\widehat{\theta}_{n,j}^* - \overline{\theta})^2}$$
where $\overline{\theta} = \frac{1}{B}\sum_{j=1}^{B}\widehat{\theta}_{n,j}^*$.
4. Output $\widehat{s}$.

29

29

Bootstrapping Confidence Interval Estimator

Bootstrap Confidence Interval

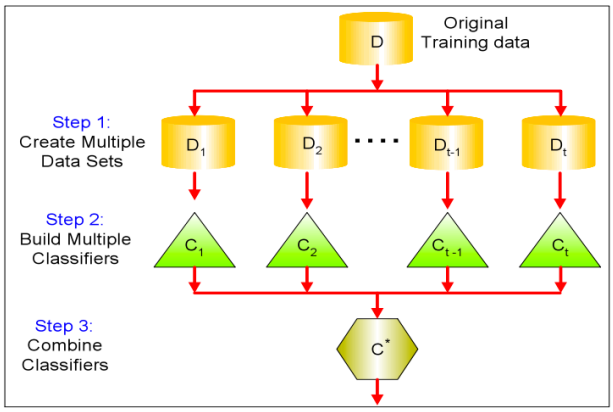
1. Draw a bootstrap sample $X_1^*, \ldots, X_n^* \sim P_n$. Compute $\widehat{\theta}_n^* = g(X_1^*, \ldots, X_n^*)$.
2. Repeat the previous step, B times, yielding estimators $\widehat{\theta}_{n,1}^*, \ldots, \widehat{\theta}_{n,B}^*$.
3. Let
$$\widehat{F}(t) = \frac{1}{B}\sum_{j=1}^{B} I\left(\sqrt{n}(\widehat{\theta}_{n,j}^* - \widehat{\theta}_n) \leq t\right).$$
4. Let
$$C_n = \left[\widehat{\theta}_n - \frac{t_{1-\alpha/2}}{\sqrt{n}},\ \widehat{\theta}_n - \frac{t_{\alpha/2}}{\sqrt{n}}\right]$$
where $t_{\alpha/2} = \widehat{F}^{-1}(\alpha/2)$ and $t_{1-\alpha/2} = \widehat{F}^{-1}(1-\alpha/2)$.
5. Output C_n.

30

30

Bagging Trees

- Fit many large trees to bootstrap-resampled versions of the training data and classify by majority vote (classification) or averaging (regression).



Machine Learning, 24, 123–140 (1996)

Bagging Predictors

LEO BREIMAN leo@stat.berkeley.edu
Statistics Department, University of California, Berkeley, CA 94720

Editor: Ross Quinlan

Abstract. Bagging predictors is a method for generating multiple versions of a predictor and using these to get an aggregated predictor. The aggregation averages over the versions when predicting a numerical outcome and does a plurality vote when predicting a class. The multiple versions are formed by making bootstrap replicates of the learning set and using these as new learning sets. Tests on real and simulated data sets using classification and regression trees and subset selection in linear regression show that bagging can give substantial gains in accuracy. The vital element is the instability of the prediction method. If perturbing the learning set can cause significant changes in the predictor constructed, then bagging can improve accuracy.

Keywords: Aggregation, Bootstrap, Averaging, Combining

Mathematically: $\hat{f}_{\text{bag}}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}^{*b}(x).$

- Feature importance can also be averaged.

31

31

Random Forest

Random forest introduces two sources of randomness.

- Bagging: Each tree is grown upon a bootstrap sample of the training data.
- Random split: For each tree at each node, the best split is chosen from a random sample of m features, instead of all features.

Random forest substantially reduces the model variance/overfitting and generally works well in practice as the benchmark model to start with.

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

To make a prediction at a new point x:

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

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

32

32

Random Forest for Causal Inference

R Package for Generalized Randomized Forests:
https://grf-labs.github.io/grf/

JOURNAL OF THE AMERICAN STATISTICAL ASSOCIATION
2018, VOL. 113, NO. 523, 1228–1242, Theory and Methods
https://doi.org/10.1080/01621459.2017.1319839

Taylor & Francis
Taylor & Francis Group

Check for updates

Estimation and Inference of Heterogeneous Treatment Effects using Random Forests

Stefan Wager and Susan Athey

Stanford University, Stanford, CA

ABSTRACT
Many scientific and engineering challenges—ranging from personalized medicine to customized marketing recommendations—require an understanding of treatment effect heterogeneity. In this article, we develop a nonparametric *causal forest* for estimating heterogeneous treatment effects that extends Breiman's widely used random forest algorithm. In the potential outcomes framework with unconfoundedness, we show that causal forests are pointwise consistent for the true treatment effect and have an asymptotically Gaussian and centered sampling distribution. We also discuss a practical method for constructing asymptotic confidence intervals for the true treatment effect that are centered at the causal forest estimates. Our theoretical results rely on a generic Gaussian theory for a large family of random forest algorithms. To our knowledge, this is the first set of results that allows any type of random forest, including classification and regression forests, to be used for provably valid statistical inference. In experiments, we find causal forests to be substantially more powerful than classical methods based on nearest-neighbor matching, especially in the presence of irrelevant covariates.

ARTICLE HISTORY
Received December 2015
Revised March 2017

KEYWORDS
Adaptive nearest neighbors matching; Asymptotic normality; Potential outcomes; Unconfoundedness

The Annals of Statistics
2019, Vol. 47, No. 2, 1148–1178
https://doi.org/10.1214/18-AOS1709

GENERALIZED RANDOM FORESTS

BY SUSAN ATHEY*, JULIE TIBSHIRANI† AND STEFAN WAGER*

Stanford University and Elasticsearch BV†*

We propose generalized random forests, a method for nonparametric statistical estimation based on random forests (Breiman [*Mach. Learn.* **45** (2001) 5–32]) that can be used to fit any quantity of interest identified as the solution to a set of local moment equations. Following the literature on local maximum likelihood estimation, our method considers a weighted set of nearby training examples; however, instead of using classical kernel weighting functions that are prone to a strong curse of dimensionality, we use an adaptive weighting function derived from a forest designed to express heterogeneity in the specified quantity of interest. We propose a flexible, computationally efficient algorithm for growing generalized random forests, develop a large sample theory for our method showing that our estimates are consistent and asymptotically Gaussian and provide an estimator for their asymptotic variance that enables valid confidence intervals. We use our approach to develop new methods for three statistical tasks: nonparametric quantile regression, conditional average partial effect estimation and heterogeneous treatment effect estimation via instrumental variables. A software implementation, `grf` for `R` and `C++`, is available from CRAN.

33

33

Boosting Tree

- Like bagging, boosting is another general ensemble learning approach.
  - Key idea: To sequentially construct a strong learner by fitting and aggregating a lot of weak learners.
- Boosting tree:
  - Each tree is grown using the information from previously grown trees.
  - Each tree tries to squeeze the errors from the previous trees.
- XGBoost (eXtreme Gradient Boosting Tree)
  - Still the most popular ML model on Kaggle beyond DL.

XGBoost: A Scalable Tree Boosting System

Tianqi Chen
University of Washington
tqchen@cs.washington.edu

Carlos Guestrin
University of Washington
guestrin@cs.washington.edu

ABSTRACT
Tree boosting is a highly effective and widely used machine learning method. In this paper, we describe a scalable end-to-end tree boosting system called XGBoost, which is used widely by data scientists to achieve state-of-the-art results on many machine learning challenges. We propose a novel sparsity-aware algorithm for sparse data and weighted quantile sketch for approximate tree learning. More importantly, we provide insights on cache access patterns, data compression and sharding to build a scalable tree boosting system. By combining these insights, XGBoost scales beyond billions of examples using far fewer resources than existing systems.

Keywords
Large-scale Machine Learning

problems. Besides being used as a stand-alone predictor, it is also incorporated into real-world production pipelines for ad click through rate prediction [15]. Finally, it is the de-facto choice of ensemble method and is used in challenges such as the Netflix prize [3].

In this paper, we describe XGBoost, a scalable machine learning system for tree boosting. The system is available as an open source package[2]. The impact of the system has been widely recognized in a number of machine learning and data mining challenges. Take the challenges hosted by the machine learning competition site Kaggle for example. Among the 29 challenge winning solutions [3] published at Kaggle's blog during 2015, 17 solutions used XGBoost. Among these solutions, eight solely used XGBoost to train the model, while most others combined XGBoost with neural nets in ensembles. For comparison, the second most popular method, deep neural nets, was used in 11 solutions. The success

https://xgboost.readthedocs.io/en/stable/index.html

34

34

Boosting Tree

Real value (label) known from the training data-set

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y_i}^{(t-1)} + f_t(\mathbf{x}_i)) + \Omega(f_t)$$

Can be seen as f(x + Δx) where x = $\hat{y_i}^{(t-1)}$

- Use the prior tree's prediction $\hat{y}_i^{(t-1)}$ to tailor the training data for the next tree.
- Each single tree will be very small to avoid overfitting.
- Observations predicted wrong before will be weighted more in the future.
- The errors will be slowly and gradually squeezed.
- A lot of hyper-parameters can be finetuned.

Algorithm 10.3 *Gradient Tree Boosting Algorithm.*

1. Initialize $f_0(x) = \arg\min_\gamma \sum_{i=1}^{N} L(y_i, \gamma)$.
2. For $m = 1$ to M:
   (a) For $i = 1, 2, \ldots, N$ compute
   $$r_{im} = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f=f_{m-1}}.$$
   (b) Fit a regression tree to the targets r_{im} giving terminal regions R_{jm}, $j = 1, 2, \ldots, J_m$.
   (c) For $j = 1, 2, \ldots, J_m$ compute
   $$\gamma_{jm} = \arg\min_\gamma \sum_{x_i \in R_{jm}} L\left(y_i, f_{m-1}(x_i) + \gamma\right).$$
   (d) Update $f_m(x) = f_{m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$.
3. Output $\hat{f}(x) = f_M(x)$.

35

35

What is the Best Model?

- Suppose you are facing a prediction problem with 10,000+ data points, and each of them has a binary label to be predicted, and 30+ features.
- Which model should you choose?
- Look at similar problems on Kaggle (https://www.kaggle.com/) and use the winning strategies as the starting point and benchmark.
- Rule of thumb:
  - For small tabular data sets (n < 1m) and typical prediction problems, XGBoost is the best.
  - For large data sets (n >> 1m) or NLP/CV problems, you should finetune the corresponding DL methods.

36

36