Artificial Intelligence for Business Research @Antai

Machine Learning Basics

Renyu (Philip) Zhang

1

Agenda

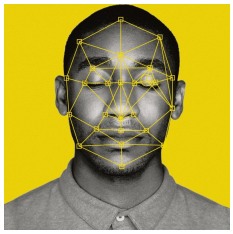
- Supervised learning model evaluation and model selection
- K Nearest Neighbors
- Decision Tree
- Ensemble Methods

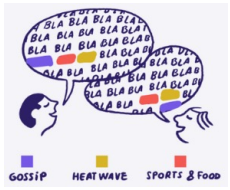
2

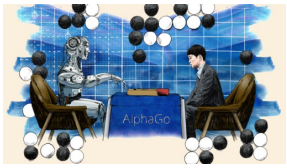
2

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 new 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: $L(Y, f(X))$

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:

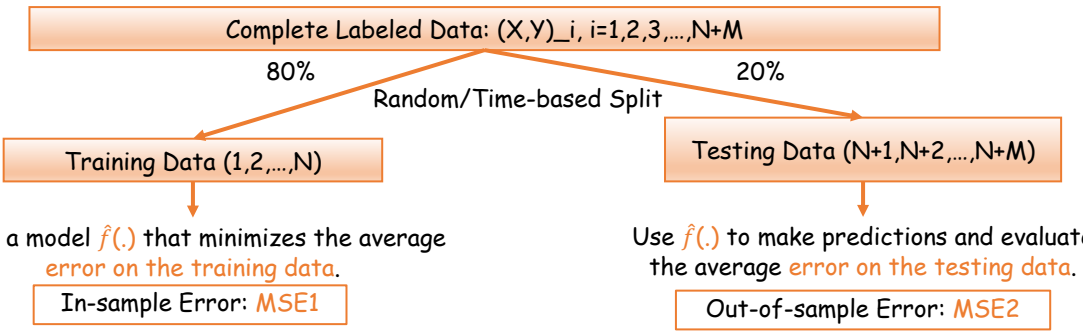
- Parametric model (OLS)
- Nonparametric model (kNN)
- Semiparametric model (Cox Proportional-Hazards Model, DML)

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

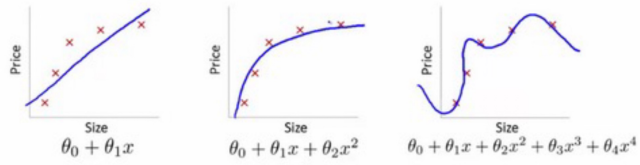
6

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.

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

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



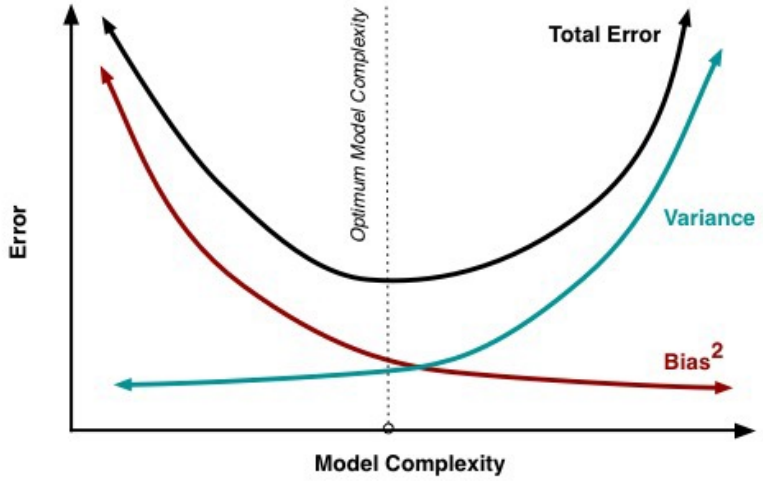
High bias (underfit) | "Just right" | High variance (overfit)

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

7

7

Bias-Variance Trade-off



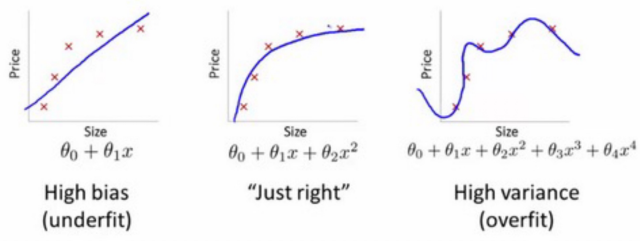
Big Assumption: The complexity of the model does not decrease the quality of model estimation.

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.



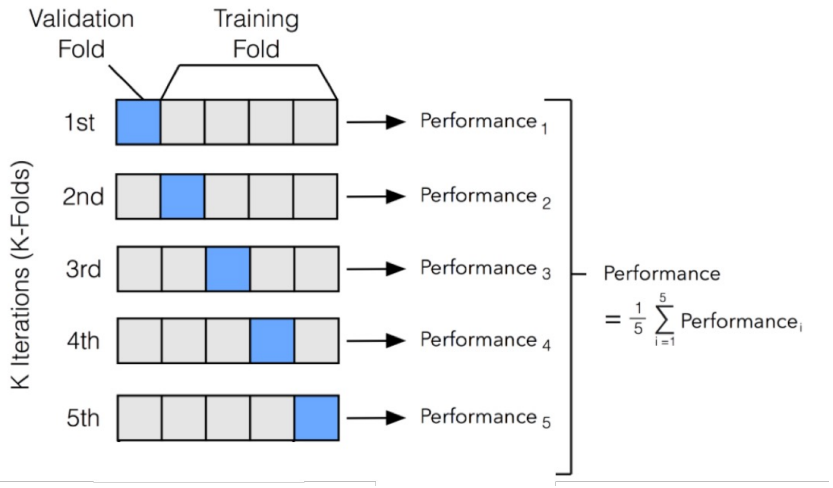
Training Data (1,2,…,N)	Validation Data (N+1,…,N+M)	Testing Data (N+M+1,…,N+M+P)
Find the coefficients	Find k	Evaluate model and report results

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

9

9

Cross Validation



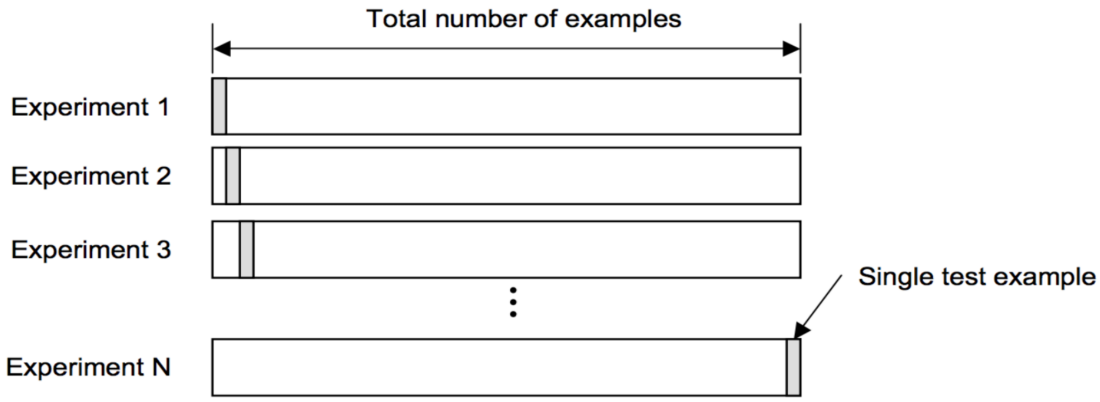
Cross validation combines the training and validation sets.

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?

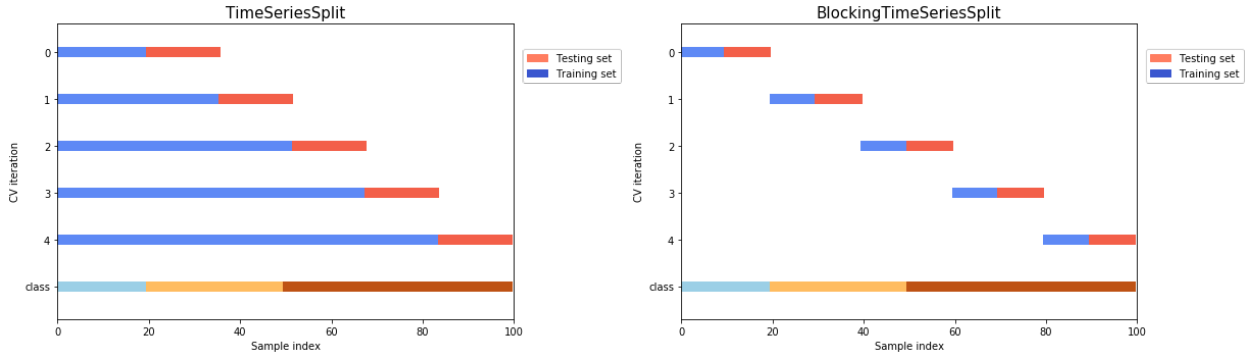
- With a large k:
  - Small bias vs. large computational time
- With a small k:
  - Small computational time vs. large bias
- In practice, the choice of k depends on sample size:
  - For a large data set, 3-fold will be quite accurate.
  - For very sparse data sets, leave-one-out may be a better choice.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

Agenda

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

14

14

Traditional Supervised Learning Models

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

15

15

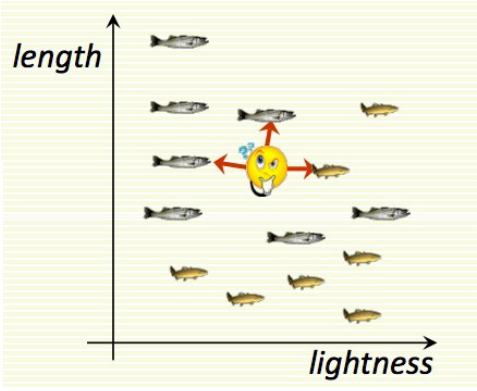
K Nearest Neighbors (k-NN)

- 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.

16

16

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.

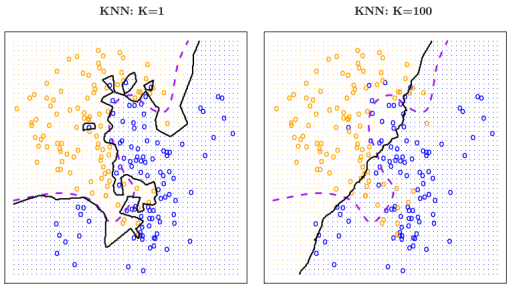


FIGURE 2.16. *A comparison of the KNN decision boundaries (solid black curves) obtained using $K = 1$ and $K = 100$ on the data from Figure 2.13. With $K = 1$, the decision boundary is overly flexible, while with $K = 100$ it is not sufficiently flexible. The Bayes decision boundary is shown as a purple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

17

17

Agenda

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

18

18

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

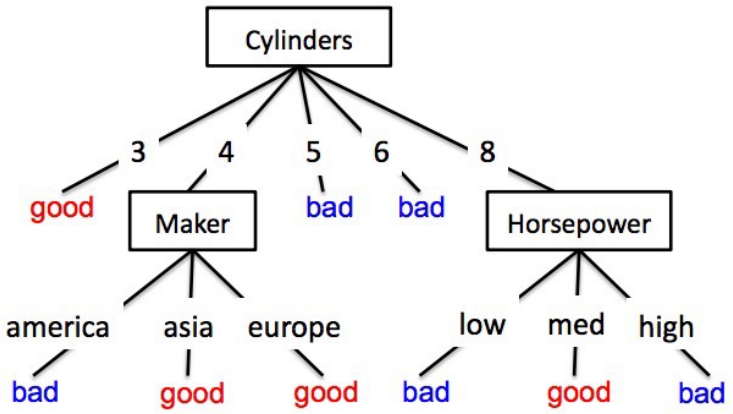
Y X

19

19

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.

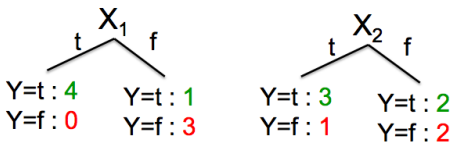


20

20

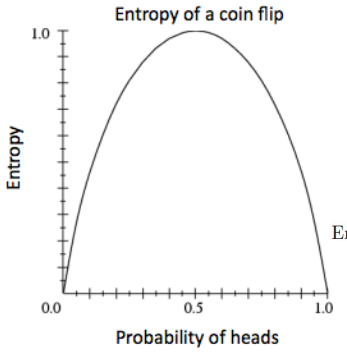
Fitting a Decision Tree

- 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

21

21

Agenda

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

22

22

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.

23

23

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
  - Difficult 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.

24

24

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 $\widehat{s}$.

25

25

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

$$C_n = \left[\widehat{\theta}_n - \frac{t_{1-\alpha/2}}{\sqrt{n}},\ \widehat{\theta}_n - \frac{t_{\alpha/2}}{\sqrt{n}}\right]$$

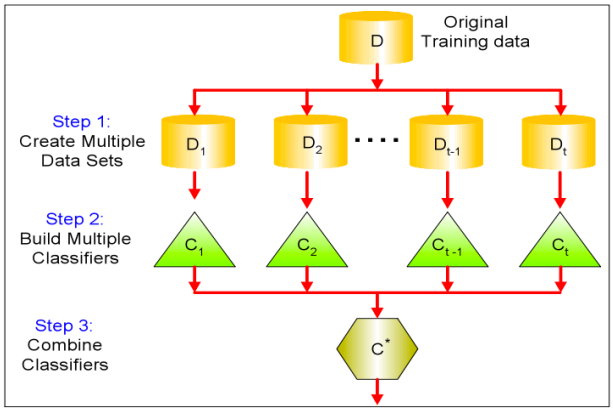
   where $t_{\alpha/2} = \widehat{F}^{-1}(\alpha/2)$ and $t_{1-\alpha/2} = \widehat{F}^{-1}(1-\alpha/2)$.
5. Output C_n.

26

26

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.

27

27

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 $\hat{C}_{\text{rf}}^B(x) = \textit{majority vote}\ \{\hat{C}_b(x)\}_1^B$.

28

28

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,

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

29

29

Boosting Tree

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

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

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 $\hat{f}(x) = f_M(x)$.

30

30

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.

31

31

The Review of Financial Studies

Empirical Asset Pricing via Machine Learning*

Shihao Gu
Booth School of Business, University of Chicago

Bryan Kelly
Yale University, AQR Capital Management, and NBER

Dacheng Xiu
Booth School of Business, University of Chicago

We perform a comparative analysis of machine learning methods for the canonical problem of empirical asset pricing: measuring asset risk premiums. We demonstrate large economic gains to investors using machine learning forecasts, in some cases doubling the performance of leading regression-based strategies from the literature. We identify the best-performing methods (trees and neural networks) and trace their predictive gains to allowing nonlinear predictor interactions missed by other methods. All methods agree on the same set of dominant predictive signals, a set that includes variations on momentum, liquidity, and volatility. (*JEL* C52, C55, C58, G0, G1, G17)

Received September 4, 2018; editorial decision September 22, 2019 by Editor Andrew Karolyi. Authors have furnished an Internet Appendix, which is available on the Oxford University Press Web site next to the link to the final published paper online.

ML for Asset Pricing

- ML methods (DT, RF, XGBT, neural nets, etc.) perform better than traditional regression-based approaches to measure asset risk premiums.
- Using ML-based strategies also obtain larger economic gains than regression-based strategies.
- Capturing nonlinear relationships is a key to the successful predictors (trees and neural nets).

32

32