Gradient Boosting Machines

Amy Ruskin
MUMT 621

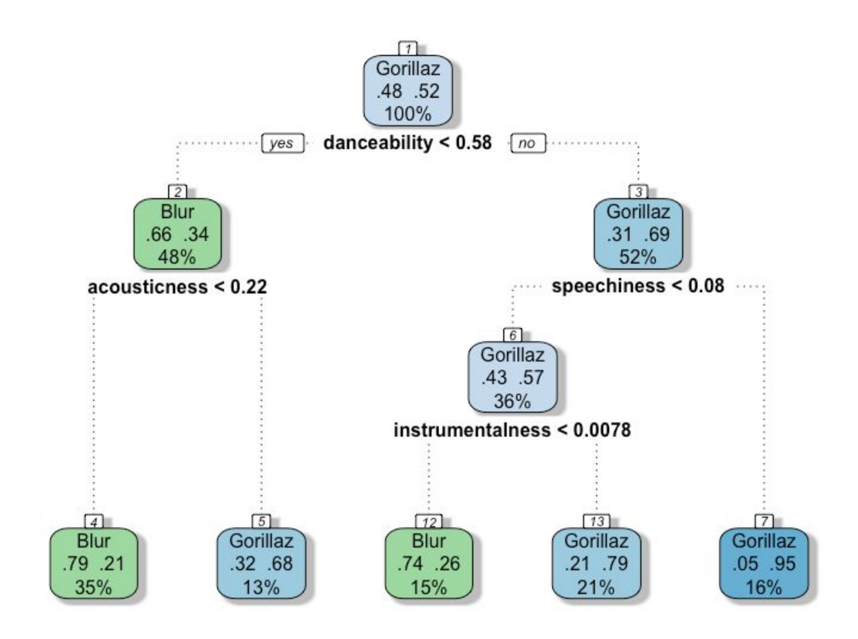
A note on terminology

- **Gradient boosting machine (GBM)** (Friedman 2001)
- Gradient boosting decision trees (GBDT) (Ke et al. 2017; Dorogush et al. 2017; Bai and Fan 2017)
- Gradient tree boosting (Chen and Guestrin 2016)
- Gradient boosted regression tree (Chen and Guestrin 2016)
- Gradient-boosted trees (GBTs)[1]

[1] https://spark.apache.org/docs/latest/mllib-ensembles.html#gradient-boosted-trees-gbts

Decision trees

- Partition feature space into rectangles
- Observations in same partition assigned same value
- At each level, find variable and split point to minimize splitting criterion across left and right children
- Examples of splitting criteria
    - **Regression**: sum of squared residuals
    - **Classification:** measures of node impurity like misclassification rate, Gini index, cross-entropy/deviance



A simple example of a decision tree for artist classification (using Spotify audio features)

(Hastie et al. 2009)

Pros & cons of decision trees

Advantages

- Fast to construct
- Easy to interpret models
- Mix numeric and categorical predictors
- Handling of missing values
- Invariant under strictly monotone transformations of predictors
    - Not sensitive to long-tailed distributions and outliers in predictors
- Feature selection performed as part of fitting procedure

Disadvantages

- Instability—highly susceptible to small changes in training data
- Lack of smoothness in prediction surface
- Difficulty capturing additive structure
- Inaccurate compared to other methods

(Hastie et al. 2009; Friedman 2001)

TABLE 10.1. *Some characteristics of different learning methods. Key:* ▲ *= good,* ◆ *=fair, and* ▼ *=poor.*

Characteristic	Neural Nets	SVM	Trees	MARS	k-NN, Kernels
Natural handling of data of "mixed" type	▼	▼	▲	▲	▼
Handling of missing values	▼	▼	▲	▲	▲
Robustness to outliers in input space	▼	▼	▲	▼	▲
Insensitive to monotone transformations of inputs	▼	▼	▲	▼	▼
Computational scalability (large N)	▼	▼	▲	▲	▼
Ability to deal with irrelevant inputs	▼	▼	▲	▲	▼
Ability to extract linear combinations of features	▲	▲	▼	▼	◆
Interpretability	▼	▼	◆	▲	▼
Predictive power	▲	▲	▼	◆	▲

Boosting can improve **accuracy** of trees at the cost of **interpretability** and **speed**

(Hastie et al. 2009, p. 351)

What is boosting?

- Combination of “weak” models → powerful “committee” (Hastie et al. 2009)
- Iteratively fit models to compensate for shortcomings in previously trained models (Li 2016)

Gradient tree boosting algorithm

L(*y*, *f*(*x*)): some differentiable loss function

M: total # of trees in model

N: total # of observations in training data

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

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

   (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

   $$\boxed{\gamma_{jm} = \arg\min_\gamma \sum_{x_i \in R_{jm}} L\left(y_i, f_{m-1}(x_i) + \gamma\right).}$$

   (d) Update $f_m(x) = f_{m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$.
3. Output $\hat{f}(x) = f_M(x)$.

Pseudo-residual (for observation *i*, based on predictions from trees 1 to *m*-1)

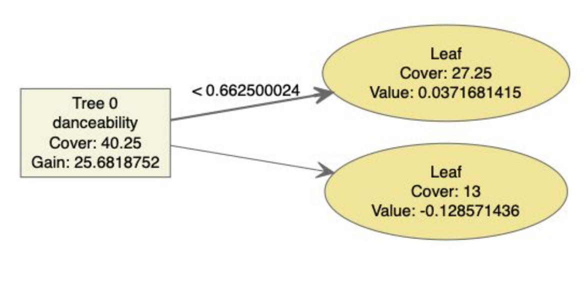
Value assigned at leaf *j* in tree *m*

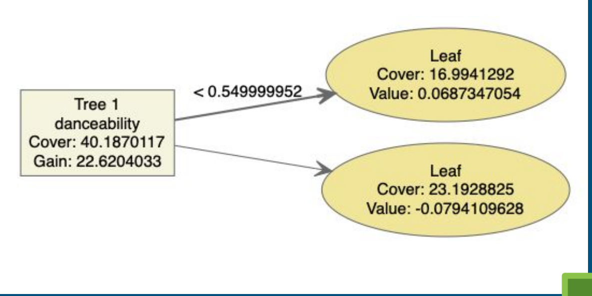
(Hastie et al. 2009, p. 361)

GBM parameters

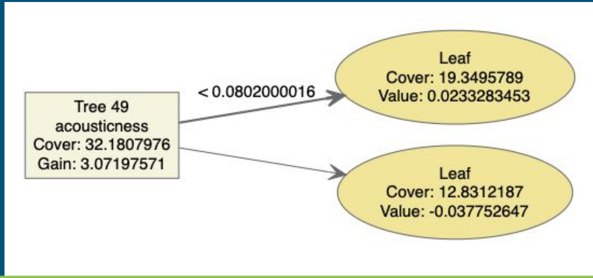
- Number of trees in model
- Shrinkage
  - Scale each update by “learning rate” parameter
  - Lower learning rate → more trees needed to converge
- Size of constituent trees
  - Constrain maximum depth, minimum number of observations per leaf
- Penalization
- Bagging/subsampling

Example: Making predictions with a GBM





…



		Predicted P(Artist = Blur)			
Artist	**Song**	**1 Tree**	**2 Trees**	**…**	**50 Trees**
Blur	“Tracy Jacks”	0.509	0.489		0.619
Blur	“Beetlebum”	0.509	0.526		0.759
Gorillaz	“DARE”	0.468	0.448		0.223
Gorillaz	“Feel Good Inc.”	0.468	0.448		0.135

Tree 0:
Danceability: $0.636 < 0.663$
$P(\text{Blur}) = \text{logit}^{-1}(0.037) = 0.509$

Tree 1:
Danceability: $0.636 > 0.550$
$P(\text{Blur}) = \text{logit}^{-1}(0.037 - 0.079) = 0.489$
…
Tree 49:
Acousticness: $0.098 > 0.080$
$P(\text{Blur}) = \text{logit}^{-1}(0.037 - 0.079 + \ldots - 0.038) = 0.619$

Pros & cons of GBMs

Advantages

- Over single decision tree:
  - Improved accuracy
  - Prediction surface still not smooth, but finer granularity to piecewise approximations
  - Averaging over many small trees → greater stability
- Maintains many desirable properties of decision trees
- Can give quick indication of dataset's potential predictability

Disadvantages

- Slower to compute than single decision tree
- Difficult to interpret model
  - Relative importance of input variables
  - Partial dependence plots

(Hastie et al. 2009; Friedman 2001)

Implementations

- Hot right now:
  - XGBoost (https://github.com/dmlc/xgboost)
    - Developed at University of Washington, first release in 2014
    - Official support in: Python, R, JVM, Ruby, Julia, C
  - LightGBM (https://github.com/microsoft/LightGBM)
    - Developed by Microsoft Research, first release in 2017
    - Official support in: Python, R, C
  - CatBoost (https://github.com/catboost)
    - Developed at Yandex, first release in 2017
    - Official support in: Python, R (apply already trained models in Java, C/C++)
  - Learning tasks: regression, classification (binary and multiclass), ranking
- In machine learning libraries: h2o, scikit-learn, Spark MLlib, and more

Boosted tree methods for MIR

- “Aggregate Features and AdaBoost for Music Classification” (Bergstra et al. 2006)
  - Winning method for genre classification in MIREX 2005 contest, runner-up for artist recognition
  - AdaBoost classifier with small decision trees (or stumps) used as weak classifiers
- “Incorporating Field-aware Deep Embedding Networks and Gradient Boosting Decision Trees for Music Recommendation” (Bai and Fan 2017)
  - Winning entry to WSDM-KKBOX Music Recommendation Challenge using ensemble of Field-aware Deep Embedding Networks and GBMs (LightGBM)
- “Evaluating Music Mastering Quality Using Machine Learning” (Shtern et al. 2018)
  - Used CatBoost implementation of gradient boosting
- “Detecting Music Genre Using Extreme Gradient Boosting” (Murauer and Specht 2018)
  - Submission to crowdAI music genre classification challenge in WebConference 2018
  - Best results from XGBoost classifier trained on numeric features extracted from mp3 files with Essentia

References

- Bai, Bing, and Yushun Fan. 2017. “Incorporating Field-aware Deep Embedding Networks and Gradient Boosting Decision Trees for Music Recommendation.” In *The 11th ACM International Conference on Web Search and Data Mining (WSDM)*.
- Bergstra, James, Norman Casagrande, Dumitru Erhan, Douglas Eck, and Balázs Kégl. 2006. “Aggregate Features and AdaBoost for Music Classification.” *Machine Learning*, 65: 473–484. https://doi.org/10.1007/s10994-006-9019-7
- Chen, Tianqi, and Carlos Guestrin. 2016. “XGBoost: A Scalable Tree Boosting System.” In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785–794. https://doi.org/10.1145/2939672.2939785
- Dorogush, Anna Veronika, Vasily Ershov, and Andrey Gulin. 2018. “CatBoost: Gradient Boosting with Categorical Features Support.” *arXiv preprint arXiv:1810.11363*
- Friedman, Jerome H. 2001. “Greedy Function Approximation: A Gradient Boosting Machine.” *The Annals of Statistics* 29 (5): 1189–1232.

References

- Hastie, Trevor, Robert Tibshirani, and Jerome Friedman. 2009. *The Elements of Statistical Learning* (2nd ed). New York: Springer.
- Ke, Guolin, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, Tie-Yan Liu. 2017. “LightGBM: A Highly Efficient Gradient Boosting Decision Tree.” *Advances in Neural Information Processing Systems* 30 (NIPS 2017): 3149–3157.
- Li, Cheng. 2016. “A Gentle Introduction to Gradient Boosting.” https://www.chengli.io/tutorials/gradient_boosting.pdf (accessed 17 February 2020)
- Murauer, Benjamin, and Günther Specht. 2018. “Detecting Music Genre Using Extreme Gradient Boosting.” In *Companion Proceedings of the The Web Conference 2018*, 1923–27. https://doi.org/10.1145/3184558.3191822
- Shtern, Mark, Pedro Casas, and Vassilios Tzerpos. 2018. “Evaluating Music Mastering Quality Using Machine Learning.” In *Proceedings of the 28th Annual International Conference on Computer Science and Software Engineering*, 126–35.