Lecture 26: XGBoost & Text as Data

Big Data and Machine Learning for Applied Economics
Econ 4676

Ignacio Sarmiento-Barbieri

Universidad de los Andes

November 12, 2020

Agenda

1. Recap: XGBoost
   - XGBoost: Demo
2. Text as Data
3. Review & Next Steps
4. Further Readings

XGBoost: Recap

- Why talk about XGBoost?
  - Among the 29 challenge winning solutions published in 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. (The second most popular method, deep neural nets, was used in 11 solutions)
  - The success of the system was also witnessed in 2015 Data Mining and Knowledge Discovery competition organized by ACM (KDD Cup) , where XGBoost was used by every winning team in the top-10.
  - Historically, XGBoost has performed quite well for structured, tabular data. But, if you are dealing with non-structured data such as images, neural networks are usually a better option (more on this later)

Boosting Trees

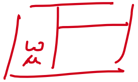
- Learning tree structure is much harder than traditional optimization problem where you can simply take the gradient.
- It is intractable to learn all the trees at once.
- Instead, we use an additive strategy: fix what we have learned, and add one new tree at a time. We write the prediction value at step m as $\hat{y}_i^m$.
- Then we have

$$
\begin{aligned}
\hat{y}_i^0 &= 0 \\
\hat{y}_i^1 &= \hat{y}_i^0 + f_1(x_i) \\
\hat{y}_i^2 &= \hat{y}_i^1 + f_2(x_i) \\
\dots \\
\hat{y}_i^M &= \sum_{m=1}^{M} f_m(x_i) = \hat{y}_i^{m-1} + f_m(x_i)
\end{aligned} \tag{1}
$$

XGBoost is a Boosting Tree

$\hat{y} = f(x)$

$L \to$ árbol



- Which tree do we want at each step?
- Add the one that optimizes our objective.

$$\mathcal{L} = \sum_{i=1}^{N} L(y_i, \hat{y}_i) + \sum_{k=1}^{m} \Omega(f_k) \tag{2}$$

- $L(.)$ is a differentiable convex loss function that measures the difference between the prediction $\hat{y}_i$ and the target y_i.
- The second term $\Omega(f)$ penalizes the complexity of the model, where

Shrinkage $\downarrow$ Ridge $(w - 0)^2$

T # leaves

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda ||\omega||_2 \tag{3}$$

$+ \lambda_2 |w|$

$\to \hat{c}$

$\to <$ paper Manual

XGBoost is a Boosting Tree

$\mathcal{L} = L(y_i, \hat{y}_i)$ < Forma general

MSE

No

- The fuction above cannot be optimized using traditional optimization methods
- Let $\mathcal{L}^m$ be the loss at step m, then $\hat{y}_i^m$ is the prediction of row i at the m iteration
- We need to add f_m to minimize the following objective

$$\mathcal{L}^m = \sum_{i=1}^{N} L\left(y_i - \overbrace{\hat{y}_i^{m-1} + f_m(x_i)}^{\hat{y}_i}\right)^2 + \Omega(f_t) \tag{4}$$

- So in the general case, we take a second order Taylor expansion of the loss function:

$$\mathcal{L}^m = \sum_{i=1}^{N} \left[\underbrace{L(y_i, \hat{y}_i^{(m-1)})}_{\text{Cons}} + r_{im} f_m(x_i) + \frac{1}{2} h_i f_m^2(x_i) \right] + \Omega(f_t) \tag{5}$$

where $r_{im} = \frac{\partial L(y_i, \hat{y}_i^{(m-1)})}{\partial f^{(m-1)}(\mathbf{x}_i)}$ and $h_{im} = \frac{\partial^2 L(y_i, \hat{y}_i^{(m-1)})}{\partial (f^{(m-1)}(\mathbf{x}_i))^2}$

XGBoost is a Boosting Tree

- After we remove all the constants, the specific objective at step m becomes

$$\mathcal{L}^m = \sum_{i=1}^{N} \left[r_{im} f_m(x_i) + \frac{1}{2} h_i f_m^2(x_i) \right] + \Omega(f_t) \tag{6}$$

- This becomes our optimization goal for the new tree. One important advantage of this definition is that the value of the objective function only depends on $\underline{r_{im}}$ and $\underline{h_{im}}$

XGBoost Model Complexity

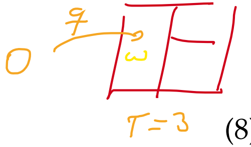
$w_{q(x)} \rightarrow$ vector

$q(x) \rightarrow$ Asigna cada punto a una hoja

- Let's talk about the regularization term
- We need to define the complexity of the tree $\Omega(f)$.
- But us first refine the definition of the tree $f(x)$ as

$$f_t(x) = w_{q(x)}, w \in R^T, q : R^d \to \{1, 2, \cdots, T\}. \tag{7}$$

- Here w is the vector of predictions on leaves, q is a function assigning each data point to the corresponding leaf, and T is the number of leaves.
- In XGBoost, they define the complexity as



$$\Omega(f) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2 \tag{8}$$

$$(w - 0)^2$$

XGBoost Model Complexity

$\boxed{g_i = r_i} \to$ slide 8

- With the aboe notation we can re writ the objective funtion as

$$\mathcal{L}^{(t)} \approx \sum_{i=1}^{n}[g_i \overset{f()}{w_{q(x_i)}} + \frac{1}{2}h_i \overset{f^2}{w^2_{q(x_i)}}] + \gamma T + \frac{1}{2}\lambda\sum_{j=1}^{T} w_j^2 \tag{9}$$

$$= \sum_{j=1}^{T}[(\sum_{i\in I_j} g_i)w_j + \frac{1}{2}(\sum_{i\in I_j} h_i + \lambda)w_j^2] + \gamma T \tag{10}$$

- where $I_j = \{i|q(x_i) = j\}$ set of indicators that assign to the $j-th$ leaf.
- Note the following in the second line, we changed the index of the summation because all the data points on the smae leaf get the same score.
- Defining $\underset{1^{\text{st}}\ \text{deriv}}{G_j} = \sum_{i\in I_j} g_i$ and $\underset{2^{\text{nd}}\ \text{deriv}}{H_j} = \sum_{i\in I_j} h_i$

Quadratic Form

$$\mathcal{L}^{(t)} = \sum_{j=1}^{T}[G_j w_j + \frac{1}{2}(H_j + \lambda)w_j^2] + \gamma T \tag{11}$$

XGBoost Model Complexity

- In the above equation, w_j are independent with respect to each other,
- the form $G_j w_j + \frac{1}{2}(H_j + \lambda) w_j^2$ is quadratic
- then the best w_j for a given structure tree structure $q(x)$ get is:

$$w_j^* = -\frac{G_j}{H_j + \lambda} \tag{12}$$

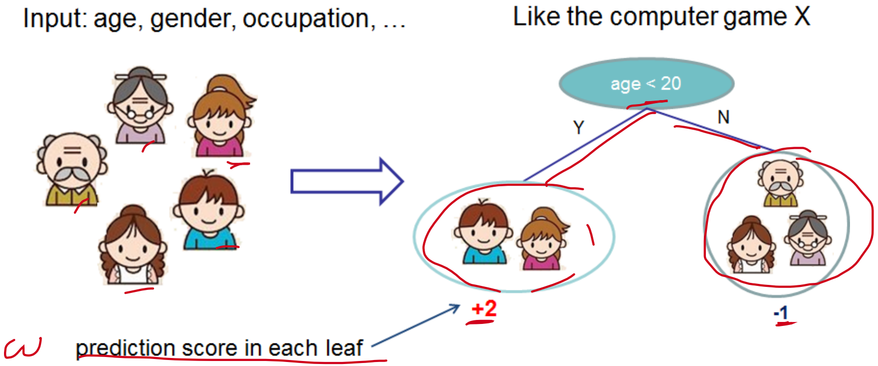
$$\tag{13}$$

- and the best objective reduction we can get

$$\mathcal{L}^* = -\frac{1}{2}\sum_{j=1}^{T}\frac{G_j^2}{H_j + \lambda} + \gamma T \tag{14}$$

$\rightarrow$ meta param

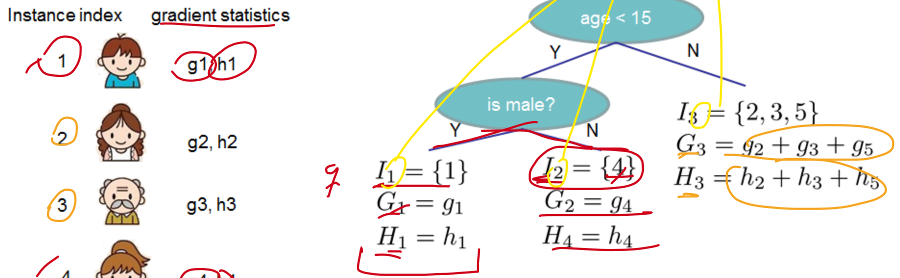
XGBoost Model Complexity: Example


Input: age, gender, occupation, ...
Like the computer game X
age < 20
Y
N
+2
-1
ω
prediction score in each leaf

source:https://xgboost.readthedocs.io/en/latest/tutorials/model.html

XGBoost Model Complexity: Example

Leaves



$$Obj = -\sum_j \frac{G_j^2}{H_j + \lambda} + 3\gamma$$

3

The smaller the score is, the better the structure is

source:https://xgboost.readthedocs.io/en/latest/tutorials/model.html

$L = (y - \hat{y})^2 = \hat{u}^2$

$y = \alpha + \beta x + u$

$L = (y - \alpha + \beta x)^2$

$G = \left[\frac{\partial L}{\partial \beta} = 2(y - \alpha + \beta x)^2 x \;\; \frac{\partial L}{\partial \alpha} \right]$

Learn the tree structure

Now that we have a way to measure how good a tree is, ideally we would enumerate all possible trees and pick the best one. In practice this is intractable, so we will try to optimize one level of the tree at a time. Specifically we try to split a leaf into two leaves, and the score it gains is

L = izq R = derecha

$$\underline{Gain} = \underbrace{\frac{1}{2}\left[\frac{G_L^2}{H_L+\lambda}+\frac{G_R^2}{H_R+\lambda}-\underbrace{\frac{(G_L+G_R)^2}{H_L+H_R+\lambda}}\right]}_{\text{Beneficio de partición}}\underbrace{\boxed{-\gamma}}_{\text{Costo}} \tag{15}$$

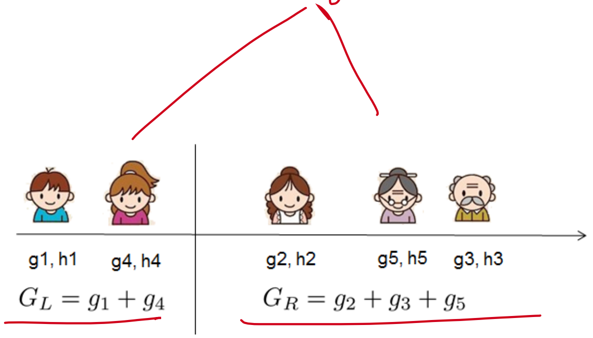
$T =$

Learn the tree structure

- This formula can be decomposed as
  1. the score on the new left leaf
  2. the score on the new right leaf
  3. The score on the original leaf
  4. regularization on the additional leaf.
- We can see an important fact here: if the gain is smaller than γ , we would do better not to add that branch. This is exactly the pruning techniques in tree based models
- For real valued data, we usually want to search for an optimal split. To efficiently do so, we place all the instances in sorted order
- A left to right scan is sufficient to calculate the structure score of all possible split solutions, and we can find the best split efficiently

Learn the tree structure

Age < 15



$$\underline{G_L = g_1 + g_4} \qquad \underline{G_R = g_2 + g_3 + g_5}$$

source:https://xgboost.readthedocs.io/en/latest/tutorials/model.html

XGBoost: Demo

```
#Load the required packages
library("McSpatial") #loads the package
library("dplyr") #for data wrangling
library("caret")

data(matchdata) #loads the data set
set.seed(123) #set the seed for replication purposes
```

```
#linear regression
model_lm<-train(lnprice~.+.:latitude+.:longitude+.:latitude:longitude,
                data = matchdata,
                trControl = trainControl(method = "cv", number = 5),
                method = "lm")
```

XGBoost: Demo

```
model_lm
```

```
## Linear Regression
##
## 3204 samples
##   17 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 2562, 2564, 2561, 2564, 2565
## Resampling results:
##
##   RMSE       Rsquared   MAE
##   0.2058519  0.8472526  0.1442374
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

$\boxed{0,2058}$

XGBoost: Demo

gbm → boosted Trees
xgboost → 'xgboost Library'

- From Caret's manual
- eXtreme Gradient Boosting
  - method = 'xgbTree'
  - Type: Regression, Classification
  - Tuning parameters:
    - nrounds (# Boosting Iterations) M
    - max_depth (Max Tree Depth) — [illegible]
    - eta (Shrinkage)
    - gamma (Minimum Loss Reduction)
    - colsample_bytree (Subsample Ratio of Columns) → Random Forest
    - min_child_weight (Minimum Sum of Instance Weight) ?
    - subsample (Subsample Percentage) → Friedman sample $\frac{1}{2}$ predecir el tiempo $\frac{1}{2}$
  - Required packages: xgboost, plyr
  - A model-specific variable importance metric is available. f/w

$y_{(m-1)} + (\nu) f_m(x_i)$
$\langle 0, 1$

XGBoost: Demo

```r
tune_grid <- expand.grid(
  nrounds = seq(from = 200, to = 1000, by = 50),
  eta = c(0.025, 0.05, 0.1, 0.3),
  max_depth = c(2, 3, 4, 5, 6),
  gamma = 0,
  colsample_bytree = 1,
  min_child_weight = 1,
  subsample = 1
)

tune_control <- caret::trainControl(
  method = "cv", # cross-validation
  number = 3, # with n folds
  ~~#index = createFolds(tr_treated$Id_clean), # fix the folds~~
  verboseIter = FALSE, # no training log
  allowParallel = TRUE # FALSE for reproducible results
)
```

XGBoost: Demo

```
input_x <- select(matchdata, -lnprice)
input_x$year <-as.factor(input_x$year)
input_x$carea <-as.factor(input_x$carea)
require("Matrix")
```

→ features

→ sparcity

```
input_x<- sparse.model.matrix(~.,data=input_x)
input_y <- matchdata$lnprice

model_xgb <- caret::train(
  x = input_x,
  y = input_y,
  trControl = tune_control,
  tuneGrid = tune_grid,
  method = "xgbTree",
  verbose = TRUE
)
```

```
model_xgb$bestTune
```

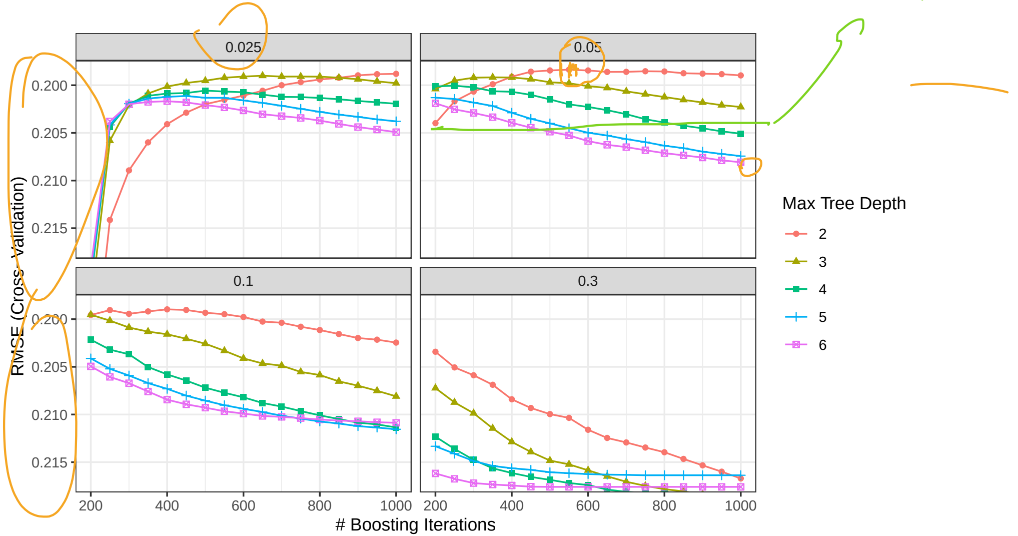
→ RMSE

```
##    nrounds max_depth  eta gamma colsample_bytree min_child_weight subsample
## 93     550         2 0.05     0                1                1         1
```

M

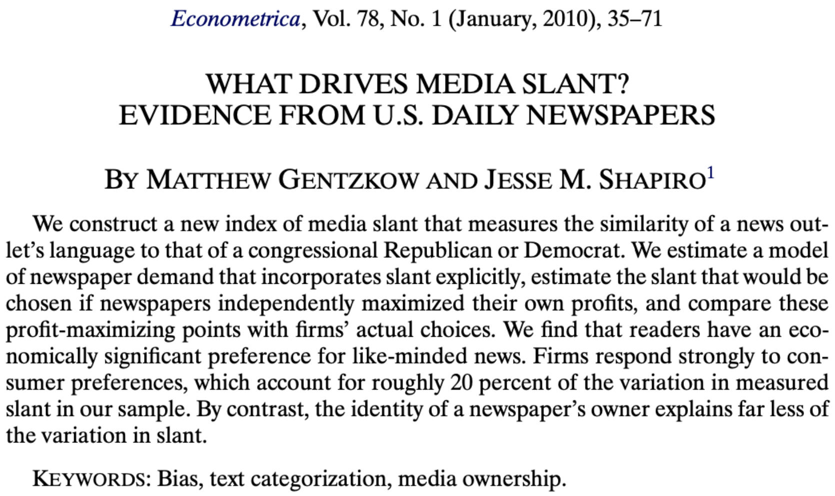
XGBoost: Demo

OLS 0,2053



Text as Data: Motivation

We'll start with a story: Slant in Partisan Speech


Econometrica, Vol. 78, No. 1 (January, 2010), 35–71

WHAT DRIVES MEDIA SLANT?
EVIDENCE FROM U.S. DAILY NEWSPAPERS

BY MATTHEW GENTZKOW AND JESSE M. SHAPIRO[1]

We construct a new index of media slant that measures the similarity of a news outlet's language to that of a congressional Republican or Democrat. We estimate a model of newspaper demand that incorporates slant explicitly, estimate the slant that would be chosen if newspapers independently maximized their own profits, and compare these profit-maximizing points with firms' actual choices. We find that readers have an economically significant preference for like-minded news. Firms respond strongly to consumer preferences, which account for roughly 20 percent of the variation in measured slant in our sample. By contrast, the identity of a newspaper's owner explains far less of the variation in slant.

KEYWORDS: Bias, text categorization, media ownership.

Text as Data: Motivation

Gentzkow and Shapiro: What drives media slant? Evidence from U.S. daily newspapers (*Econometrica*, 2010)

- Build an economic model for newspaper demand that incorporates political partisanship (Republican vs Democrat)
  - What would be independent profit-maximizing "slant"?
  - Compare this to slant estimated from newspaper text.

Congressional Record

United States of America

PROCEEDINGS AND DEBATES OF THE 95th CONGRESS, SECOND SESSION

Vol. 124 WASHINGTON, MONDAY, OCTOBER 2, 1978

TRIBUTE TO THE PHILOMATHEAN SOCIETY

HON. MICHAEL O. MYERS

Text as Data: Motivation

- Jerry Moran, R-KS, says “death tax” relatively often and his district (Kansas 1st) voted 73% for George W. Bush in 2004.

$$\mathbf{X}_{\text{text}} = f(\text{ideology}) \approx g(Y_{Bush})$$

$$\Rightarrow \text{“death tax” is republican}$$

$$\Rightarrow \text{the Wall Street Journal is slanted right.}$$

- William Jefferson, D-LA, says “estate tax” relatively often and his district voted 24% for George W. Bush in 2004.

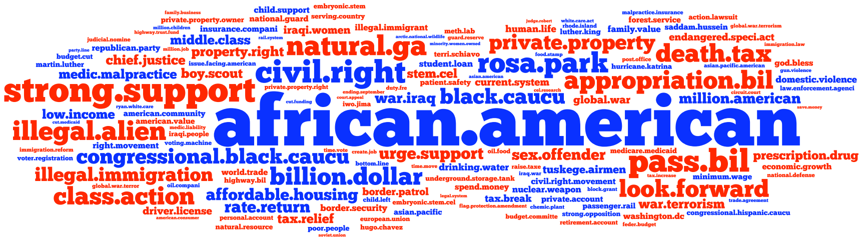
Text as Data: What is slant?

- Text: phrase-counts by speaker in 109th US Congress (05-06)
- Sentiment: two-party constituent vote-share for Bush in 2004.
- Use covariance between phrase frequencies (f_{ij}) and 'Bush' sentiment (y_i) to build an index of partisanship for text.

$$z_i^{slant} = \sum_j \text{cov}(f_j, y) f_{ij}$$

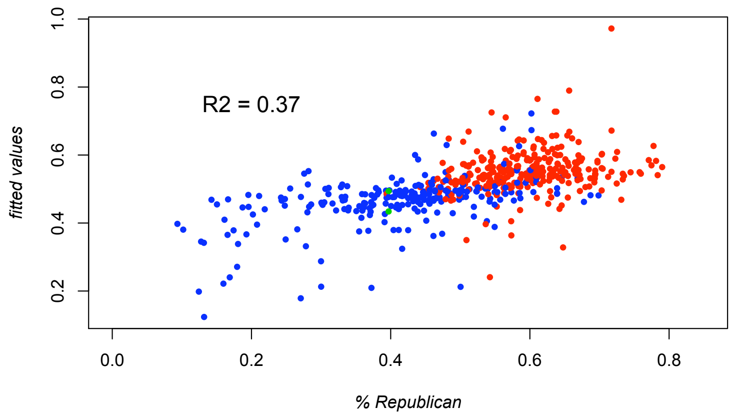
- For example, if phrase j forms a high proportion of what you say, and usage of phrase
- j is correlated with Bush vote-share, then this contributes a positive amount to your slant score.
- This is a type of *marginal regression*.

Text as Data: Wordle



- Colored by sign (positive, negative)
  Size proportional to loading $\text{cov}(f_j, y)$.
- Since y is Republican vote-share,
  - Big positive is a right term and
  - Big negative is a left term.

Slant measure for speakers in the 109th Congress



Democrats get low z_{slant} and Republicans get high z_{slant}.

Do this for newspaper text and you'll get a similar picture

Text as Data: The Big Picture

- **Text is a vast source of data for business**
- It comes connected to interesting "author" variables
  - What you buy, what you watch, your reviews
  - Group membership, who you represent, who you email
  - Market behavior, macro trends, the weather
- Opinion, subjectivity, etc.
- Sentiment is *very* loosely defined: Observables linked to the variables motivating language choice

Text as Data: The Big Picture

- **Text is also super high dimensional**
- And it gets higher dimensional as you observe more speech.
- Analysis of phrase counts is the state of the art (hard to beat).
- For example, occurrences by party for some partisan terms

Congress	State	Party	America	Death Tax	Estate Tax	$\cdots$
63	NM	dem	108	30	140	
		gop	100	220	12	

- Basic units of data
  - doc-term count matrix $\mathbf{X}$ with rows $\mathbf{x}_i$.
  - doc totals $\mathbf{m} = \texttt{rowSums}(\mathbf{X}) = [m_1 \cdots m_n]$.
  - frequency matrix $\mathbf{F} = \mathbf{X}/\mathbf{m}$ with rows $\mathbf{f}_i$.

Review & Next Steps

- XGBOOST
- XGBOOST demo
- Text as data
- Next class: More on text as data
- Questions? Questions about software?

Further Readings

- Chen, T., & Guestrin, C. (2016, August). Xgboost: A scalable tree boosting system. In Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining (pp. 785-794).
- Chen, T., He, T., & Benesty, M. (2018). XGBoost Documentation.
- Friedman, J., Hastie, T., & Tibshirani, R. (2001). The elements of statistical learning (Vol. 1, No. 10). New York: Springer series in statistics.
- Taddy, M. (2019). Business data science: Combining machine learning and economics to optimize, automate, and accelerate business decisions. McGraw Hill Professional.