QTM 347 - Problem Set #6

Kevin McAlister

2023-02-16

This is the sixth problem set for QTM 347. This homework will cover exercises related to regression trees.

Please use the intro to RMarkdown posted in the Intro module and my .Rmd file as a guide for writing up your answers. You can use any language you want, but I think that a number of the computational problems are easier in R. Please post any questions about the content of this problem set or RMarkdown questions to the corresponding discussion board.

Your final deliverable should be two files: 1) a .Rmd/.ipynb file and 2) either a rendered HTML file or a PDF. Students can complete this assignment in groups of up to 3. Please identify your collaborators at the top of your document. All students should turn in a copy of the solutions, but your solutions can be identical to those of your collaborators.

This assignment is due by February 24th, 2023 at 11:59 PM EST.

For this problem set, I have given you a subset of the big data set discussed below. First, I've reduced the number of features to 50. Second, I've reduced the overall size of the data set. I've gone ahead and split the data into three sets: `ArticlesTrain.csv` with 5,000 observations, `ArticlesValid.csv` with 2,500 observations, and `ArticlesTest.csv` with 2,500 observations. The validation set should be used to create EPE estimates to tune your models - assess the EPE of models trained using the training data set on the validation set and use this value along with any OOB EPE estimates to make decisions about tuning parameters. The test set should not be used until the last question. For all questions, the outcome variable is `shares` - the natural log of the number of times that an article was shared online.

Data Set Overview

This assignment revolves around a data set of 39,644 articles published by the website mashable.com in 2013 and 2014. Mashable is an international multi-platform media and entertainment company that can be seen as a news aggregator; the site publishes its own articles about current events and topics. In many ways, Mashable was (and still is, but it's under different ownership now) a content aggregator. The goal of Mashable was to provide an aggregate of important daily "news" and to generate ad-revenue via advertising clicks. Therefore, the profitability of the website was directly tied to its ability to attract users to the site. The most common way users were attracted was by shared articles passing through various social media outlets. More shares leads to more ad revenue!

The data set includes article urls, the log number of shares on social media via the site's provided sharing mechanisms, and a collection of 59 covariates related to the length and content of the articles. It is important to note that all 59 of these covariates could potentially be measured *prior to publication*. The reason for this can be seen in the original paper that collected and used this data set - "A Proactive Intelligent Decision Support System for Predicting the Popularity of Online News" by Kelwin Fernandes, Pedro Vinagre, and Paulo Cortez. In this paper, the authors want to demonstrate that predictive models can be used to create

actionable suggestions to increase the number of shares that an article receives. In doing so, they first build a predictive model that is expected to perform well on unseen data (articles that have not yet been written) and use this to create a rules set that provides expected improvements for new articles. The proposed IDSS is operationalized over keywords - summary words that are used to categorize an article. However, it is easy to see how a system of this sort could be used to create content suggestions for the writers, themselves, to write the "clickbaitiest" articles possible.

Many of the covariates presented are processed natural language processing (NLP) metrics that are intended to capture the topic, positive/negative sentiment (polarity), and subjectivity of the article. I'll briefly summarize these concepts here:

1. Topics are presented using LDA dimensions that sum to 1. Latent Dirichlet allocation is a common method of finding common "themes" and word associations across a large corpus of text documents. The authors ran an LDA analysis and settled on 5 "topics" for the entire data set. Each document is assumed to be a mixture of these 5 topics and can be described by a weight that corresponds to how much each document contains each topic. A document with weight 1 on a single topic means that it is wholly decribed by that one topic while weight zero means that there is no part of the document that speaks to the topic. Since this is a mixture, the weights sum to one. There is no guarantee that the LDA topics are coherent, but you can find similarities between articles that are close to each topic. Should you use one of the LDA dimensions in your models (especially your small models), you may want to undertake the task of figuring out what each of them means. **A warning: since the LDA weights for each article sum to 1, placing all 5 into a model that doesn't select away collinear features will run into issues!**

2. Polarity is a measure of the positiveness or negativeness of a document. This is typically computed using a dictionary of positive and negative words and then using associational methods for determining whether words that are not recognized are positive or negative. For this data set, negative polarity means that the document is largely negative while positive polarity means that the document is largely positive. Related measures look for the rate of positive and negative words, title polarity, and conditional polarity for only positive or negative words.

3. Subjectivity is a measure of how "fact-based" an article is. Values of 0 mean that the document is fully fact-based while values of 1 mean that the document is wholly opinion based. The machinery of the subjectivity measure largely mimics that of the polarity measure starting with a dictionary of fact vs. opinion based words and using associations to cover new words.

The rest of the covariates relate to specific features of the articles - keyword usage, data channels, etc. A short description of each covariate can be seen in the `OnlineNewsPopularity.names` file included with this assignment. Further descriptions of the data set can be found in the original paper. The paper is relatively short, so I highly recommend reading it to get a sense of what the authors are doing and how the features are used to produce this automated decision making system.

A final thing to watch out for are two sets of unordered categorical predictors - day of publication and data channel. All of these categorical predictors are already "dummied out", so you don't need to do anything special to make them usable. However, be cautious if building a linear model! The day of the week dummies are also accompanied by `is_weekend` which is just a linear combination of the other predictors. The data channel predictors are binary variables that tell which large topic each article falls under - one of lifestyle, entertainment, business, social media, technology, world news/issues, and viral content. Viral content was dropped from the predictor set. As best I can tell, any article that has a zero on all other data channels falls into the viral content category. You can include this variable in your models if you'd like by adding a column of appropriate indicators.

Question 1 - Implementing Trees (100 pts.)

Part 1 (30 pts.)

Using the training set, implement a series of random forest using all predictors that creates predictions for the log number of shared articles. For each random forest, you should generate 1,000 bootstrap trees. To try to tune the random forest, vary the number of variables to consider in each bootstrap tree (`mtry` in `ranger`).

To tune the random forest parameter, you should compute two metrics of expected prediction error - an OOB metric that will be returned by your RF implementation and a MSE metric for the out-of-sample validation set. At the end, choose M that returns the lowest expected prediction error for the OOB error.

It will take a while to try all values of $M \in (1, 2, ..., 50)$. So, try a targeted optimization strategy:

1. Start by setting $M = (10, 20, 30, 40, 50)$. Pick the one with the lowest EPE, let's call it M_1
2. Now try M set to a sequence of values spaced by 5 from $M_1 - 10$ to $M_1 + 10$. Record the lowest value as M_2. (Note: you can't set M to zero, so adjust as needed).
3. Finally, try M set to a sequence of values spaced by 1 from $M_2 - 5$ to $M_2 + 5$. The minimum EPE M in this instance will be your chosen value.

Note that this strategy doesn't guarantee that you've found the global minimum, but it works more often than not.

Create a plot that shows the OOB error plotted against the number of variable considered for each random forest. Add a line to your plot that does the same thing for the MSE on the validation set. Do the two methods return similar optimal M values?

Part 1 Solution

Part 2 (15 pts)

Using your chosen value for the number of variables to consider in your random forest, train a final random forest **on the combined training and validation sets** that predicts log shares setting the number of variables in each tree to your optimal value.

Compute the importance of the features for your full training data set. Create a graph that demonstrates which variables are most important for the random forest and which ones are least important.

Generate a partial dependence plot for the top two variables. Generate two single-variable plots (and a joint one, if you'd like).

Part 2 Solution

Part 3 (30 pts)

Now, let's train a boosted tree to predict log shares.

There are two parameters we need to tune for this model: the depth of the little trees and the number of little trees to combine. **You need to tune these together.** Find the combination of number of little trees and depth that minimizes EPE. Record these values.

To tune the boosted tree, let's take the approach of setting the depth and shrinkage parameters first and finding M that minimizes the EPE given those values. To ease computational burden, you should use the validation set as a method of computing EPE for this problem and avoid the CV estimator of EPE.

Notes:

A shrinkage parameter, λ, of .05 works quite well for this problem.

Setting the maximum number of little trees at a given depth to 1000 works well for this problem.

You only need to check depths of 1, 2, and 3 for this problem.

For both R and Python implementations of the gradient boosted regression tree, you can monitor the EPE of the tree at each m using a validation set. This is set by the validation fraction - the proportion of the training set to holdout for validation. For this data, you can combine the train and validation sets and set the train fraction to .67.

I don't recommend using bagging here. In `gbm`, this amounts to setting `bag.fraction = 1`. In `sklearn`, this amounts to setting `subsample = 1`.

In Python, the `GradientBoostingRegressor` method has a `max_features` argument. Just leave it at the default. Though, you are more than welcome to try to tune this value - it may improve your boosted tree.

Choosing the best depth and m should leverage the following approach:

1. Train the boosted tree on the training data for at least 1000 trees at shrinkage = .05 with a given depth.
2. Find the minimum EPE implied by the validation set for the particular depth (`min(boost_tree$valid.error)` in `gbm`). Record this value.
3. Choose the depth that minimizes the minimum value. Set this as your optimal depth.
4. Given the optimal depth model, find the corresponding m (`which.min(boost_tree$valid.error)` in `gbm`).

Python's implementation has auto-stopping features. Just let the algorithm auto-stop if you're using this approach.

Part 3 Solution

Part 4 (15 pts)

Retrain your boosted tree on the combined training and validation set **with no holdout observations** setting M and depth to their optimal values.

Compute the importance of the features in your boosted tree. Create a graph that demonstrates which variables are most important and which ones are least important. Are the important variables similar between the random forest and the boosted tree?

Part 4 Solution

Part 5 (10 pts)

Finally, let's see which of the two methods performs better on the true heldout test set. Compute predictions for your optimal random forest and for your optimal boosted tree. Use these predictions to compute an estimate of the EPE. Which method performs best?

Note: there are other ways we can tune a boosted tree. For the midterm, please do not hesitate to try to optimize it!

Part 5 Solution