CSC522- PROJECT REPORT

Blue Book for Bull Dozers

APRIL 25, 2013

ROOPAK VENKATAKRISHNAN
rvenkat7@ncsu.edu

Problem Statement:

The project involved participating in a competition on Kaggle. The aim was to use various techniques to predict the price of heavy duty equipment in auctions based on its usage, configuration, equipment type, previous sale information about the equipment. The competition calculates scores for your predictions based on RMSLE (Root Mean Square Logarithmic Error). The data, validation and test sets are provided by Kaggle.

By the end of this project I had hoped to gain more perspective into different kinds of regression techniques, their advantages and disadvantages, and which ones might be better than others.

Contents

Problem Statement: 1
Literature Review 3
GBM (Gradient Boosting Machine) 3
Project Report 5
Data Pre-Processing 5
Training / Model Construction 8
Conclusion 12
References 12

Literature Review

When trying to predict prices for an object we are dealing with a continuous scale of possible values. Regression would be an ideal way to predict values. While working with R and trying to predict prices of the equipment I came across several packages which use a variety of techniques for this purpose. One of the successful techniques was a method called '**gbm** (**Gradient Boosting Machine**)'

Let us first try and work on understanding the different terminologies as a whole before jumping into how gbm works.

Regression

Regression is a technique used to find the the relation among different attributes/variables etc. It is usually used for prediction and forecasting. It is heavily used in the field of Machine Learning. Most of the time it is used to discover causal relationships. A point to note however is that just because there exists a correlation between 2 variables does not mean that one causes another.

Boosting

Boosting is a technique used in machine learning to improve supervised learning. The idea in boosting is to work with multiple weak learning algorithms and combine their output to come up with a final strong learning algorithm. Hypothesis boosting or "boosting" is an algorithm which can achieve this.

GBM (Gradient Boosting Machine)

As the name suggests this package works by applying various boosting techniques. The documentation also suggests that it uses models which are common in statistical modelling but aren't used too much for just boosting. The idea is to boost the given input data set in such a way so that accounting is done for the randomization errors that one may come across.

GBM works by constructing a number of trees. The number of trees that are constructed is one of the parameters which can be used to tune this model. The trees are selected and grown in such a way that it randomly picks attributes and grows each of the trees based on the attributes that it picked. It then works by trying improve these trees by the training data and this process is repeated till a set of trees are grown.

One of the other parameters that is used to work with gbm is known as interaction depth. It is basically what level of interaction between attributes should be accommodated. In a usual regression model we only consider interaction between attributes at one level. However gbm goes one level further and does this at more than one level.

One should also remember that interaction depth does not necessarily specify that there are exactly n interactions, but rather up to 'n' interactions

Another attribute that gbm uses is known as shrinkage. This particular parameter is used to specify at what rate gbm learns. Usually it is seen that lower the rate, better the model that is created. But we are capped because the lower the rate is the longer it takes for the model to be constructed. so we have to achieve a balance between time taken to construct a model and the accuracy of the model itself. However this doesn't mean that as shrinkage tends to 0 that the model becomes 100% accurate. It just means that upto a certain point reducing shrinkage will result in longer times but better accuracy.

According to the paper on gbm we can say that gbm, uses a version of AdaBoost that uses exponential loss for pruning the tree. According to the paper gbm works by fitting a parameterized function to a residual model. It tries to work by reducing the mean square error.

Simply put growing many trees based on a randomly selected set of attributes out of the given data set is how the trees are started to be grown. It can be noted from the above fact that since it grows more than one tree that gbm is an ensemble technique. An ensemble technique is one where more than one model is constructed and these are combined to get one final model.

It is mentioned on a couple of articles that gbm is partly based off of AdaBoost . AdaBoost is adaptive boosting. This technique involves an exponential loss function. Further it groups importance among "base learners". The base learner T_k is defined based on the learners T_1 to T_{k-1}. This protects the model from randomization errors.

One other parameter which gbm works with is "minimum observations in a node" , which they call as n.minobsinnode. The idea of this parameter is to define the actual no of parameters in the leaf nodes of the tree. By varying this parameter one can manipulate how the various attributes interact with each other. (Note that interaction depth, is used to define the interaction between variables as well)

If the data can support weights, gbm accepts this as one parameter to try tune the way the trees grow in.

Thus gbm is an ensemble technique which gives the user a great amount of control over how he/she would like to grow the trees, and hence the model. I think that an optimum solution when using gbm would be to have a large enough number of trees and interaction depth, while having a small shrinkage to improve the overall accuracy.

Project Report

During the course of this project I had discussions to analyze the work I was doing. Most of my discussions and analysis techniques were with Jessica Schumaker (jlschuma@ncsu.edu)

When trying to analyze data there are some steps which we must follow in order to improve our results. The steps I followed when trying to work with the data given by Kaggle are

- Data Pre-processing
    - The amount of data handed to us by Kaggle was huge. Not all of this was “clean”. Data which we process needs to be devoid of missing values, noisy unwanted data etc. So cleaning the given data had to be done before it could be processed
    - Given that the data set given to us was huge and also given the fact that I had limited compute power I selected and utilized some of the data and not all of it. The process of selection of which data to use is also a part of data- processing.
- Training
    - The next phase involved training/creating a model which utilized the information we processed. The model which we finally came up with could be used to make predictions on the Test set given by Kaggle.
- Validation & Testing
    - This particular competition had 2 phases. The first one where we used the training data and came up with models which we validated against the validation data provided by Kaggle, and the second phase which involved using both the training and validation data to improve our models and test against the Test set.

Data Pre-Processing

The original set of data provided by Kaggle for training had about 400,000 thousand records with each record having about 53 attributes. However when going through this data one could easily notice that many of the 53 attributes were missing for a majority of the data. Apart from this we also had a second file which described more information about the equipment based on machine ID and model ID.

Dealing with Missing Values

Deciding what to do with missing values plays a very important role in data mining, and can significantly affect the output. In some cases if the number of records with missing values is very low we could simply discard those records to ensure that they don't affect our classification/prediction adversely. However in this case the number of records with missing data was very high. Literally 90% or higher of the dataset had at least one or more of the attributes missing a value. In this case discarding records would leave us with no data to train on.

The next idea would be for us to obtain this missing data. In this case we could either go back and find the actual missing data, or we could guess what it might have been and fill it in. Going

back to find the actual value of this data would be infeasible in this case, given the amount of data to be obtained, and that the sources would be unknown and also the fact that it would cost a lot of money.

We decided to move on to filling in the missing values with the value and then proceeding. There are a few common techniques used to fill in missing values. Some of them are as follows

- Find the mean/average value for this attribute incase it is numeric and fill that in for all missing values. In case of levels fill in the most common level.
- Find a record with the most similar values for other attributes and copy the value for the missing record.
- The most commonly used technique however is to use a 0 or -1 for all the missing attributes.

One set of processed data given to us by Andrew followed the third technique of filling all missing values with a 0. It was very evident with the given dataset that the technique of finding the records with the most similar set of attributes and using that value would not work. This was because most the records had too many missing values and also due to the size of the data set.

The method I proceeded with involved replacing all the missing values with 0. This gave us a data set we could work with.

Working with attributes having text values

When working on regression techniques to predict values, what was noticed is that most these training models require only numeric data and handle text based values too well. Thus all the attributes which had text values in the form of “levels” were converted to numbers. This would not cause any issue as the same procedure was done on the testing set as well.

For example a field having attributes of low, medium and high could be replaced by 1, 2 and 3. This way we can convert all text labels to numeric values which will help the models deal with the data better.

Another idea we obtained was from Andrew’s processed training data set. The date was split into date, month and year. We followed this technique when dealing with dates. My original idea was to convert the date into a timestamp, but on retrospect this might be a better idea for the simple reason that though a time stamp would still give information about when each equipment was sold with respect to another, it would lose the trends such as “towards the end of the year”, or “towards the end of the month” etc.

Mapping Machine Appendix with Training Data

One other thing done during data processing was mapping the machine appendix file to the training data. This can be done by mapping the two columns machineID and ModelID on both the files. This would give us a few more attributes to work with. While doing so We also ensured that the Machine Appendix file was processed for missing attributes and text data.

Selecting Data for processing

In my case this was repeated at different times. At first during data processing, fields which were just a description of another field were ignored and discarded. For example having ProductGroupDesc would not make any sense when also having ProductGroup. This was easily seen as both the attributes always had the same values for different records. i.e when ProductGroup had a value of “TTT” then ProductGroupDesc was always “Track Type Tractors”.

So all such similar fields etc were eliminated. This left with a data set for processing with the same number of rows as the original and the number of attributes to about 60.

Processing the testing data

The actions done on the data set used for training were repeated exactly on the data set to be tested. This way we made sure label values did not change on the data we were going to predict on. For example if we train our model with the labels in such a way that 1 is low, and 3 is high for a particular attribute in our training set and when we process our test set we set the labels such that 1 is for high and 3 is for low, the predictions made by our model will be terribly wrong.

So to overcome this problem we should take care to process the test data the exact same way that we process our training data. This is true especially when replacing text labels with numeric values. This can be done by storing each text label that we replace by a number in a file, say called mappings.

Another method which can be followed if we have the test data with us when processing the training data is to stack the test data and training data together and the process them. This way we know that the test set has been processed the same way as the training set.

External Data

When talking to the TA(Andrew)(during the last week of classes) we had a very interesting discussion, which got me looking up an avenue I did not explore earlier completely/to a full degree in the competition. External data. The TA mentioned obtaining information about construction budgets. Then pushing each forward by a couple of months and using this data to predict prices. As most heavy duty equipment would be used for this purpose, it can be very easily understood that it would be very useful in predicting prices. Andrew mentioned that it helped gain a significant improvement in his score.

Reading more about this, and based on simple logic I would say that this is an avenue I would have like to have explored.

Training / Model Construction

The next step involves training and creating a data model which we can use to make predictions on the test/valid data set.

Limitations

While working on this project we faced a couple of issues.

One of the major issues I faced while trying to build a model was the lack of proper computing power. Given the size of the size of the data set, running training on such huge models seems to require a lot of time as well as a *huge* amount of computing power. One should further note that since running R in parallel is not exactly a very elegant solution we notice that R seems to work only serially. Further when I say “compute power”, it is noticeable that a large amount of memory (RAM) is what is required.

Another issue I faced was that since the time line of the competition was structured in such a way that the test data was released only toward the very end of the competition and also that we did not obtain scores for our predictions on the test set till the 20th of April, I used to train my model on Train.csv and test on the validation data set Valid.csv for most part of the project. Toward the end of the project I retrained on the both the train and validation data set together and submitted to kaggle. I received a RMSLE score of about 0.28.

Training Models and Techniques

Considering that we are trying to predict price which is a varying continuous quantity we would be using regression techniques. The models that I used to create models are rpart, gbm, randomForest etc.

The software I used are R (for creating the models) and excel while working with the data.

caret

During one of the earlier assignments in this course we were introduced to caret. A package which among other things also helps to perform cross validation for the operation passed to it. When we are given a dataset and we want to create a model to predict for future values, we need a method to validate our model and see how well it performs. and based on this tune it and improve it. Obviously the amount of training data given to us for any real world problem is finite, and in such a case cross validation is a very useful based on which we can tune our model.

For this project I used cross validation to tune my model and I used caret to perform the cross validation.

gbm

One of the Packages available in R is gbm. This can be used to build many short trees and use these various trees to predict the value as we require. gbm has many parameters that can be tuned when building a model. The parameters that I thought would be good to work with are

- n.trees:
  The number of trees or the number of iterations to which the data must be fit.
- interaction.depth:
  The depth of interactions between the attributes. Interaction depth is related to the relation between the attributes, and thereby the number of leaves in the tree being grown. Note that if it can related to the number of leaves it can also be related to the actual depth of the tree.
- Shrinkage:
  The shrinkage is also known as learning rate. From various manuals online which describe the gbm method it is said that the lower the value of shrinkage the better. For our dataset it would be good to work with a shrinkage value of about .01 or .001. However we should also note that the lower this value is the longer it takes to construct a model.

Using these parameters with caret I tried to train a model. The below table shows some of the values I used for the various parameters and the value I obtained for RMSLE using cross validation

n.trees	interaction.depth	shrinkage	RMSLE(CV)	Comments
100	7	.1	0.317	
100	10	.1	0.281	
150	10	.1	0.268	0.277 on Kaggle Leaderboard
500	10	.1	0.254	used only a portion of the dataset

It was interesting to note that as the number of trees was increased the time required for the model to be constructed increased exponentially. This can be understood as an increase in the number of trees will require that many more iterations to be run. Further since 5 fold CV is being done the time increase is seen for each of the folds.

From the trend it was visible that an increase in the number of trees would greatly improve the RMSLE. However on my computer running gbm with say 500 trees seemed to make it freeze and

crash. So based on discussions with TA's and a few other people taking the course I decided to use only a portion of the training set and give it a shot. I took about 100,000 rows as opposed to the entire data set. In this case I was able to obtain an RMSLE of 0.254.

It is very evident that given the size of the dataset with an increase in the number of trees we can better results. Though I didn't run a case to see how shrinkage would affect model, after reading about how the gbm model works and how to choose a shrinkage value I am pretty sure that the results would improve with a lower shrinkage value.

rpart

One of the other techniques I used when trying to create a model is known as rpart. It is very interesting to note that among the various models I used this was one of the much faster methods in terms of construction of model.

rpart stands for recursive partitioning and regression trees. A regression tree is a technique where a decision tree is constructed to predict values for a given set of data. The more common techniques see construction of trees for classification purposes as opposed to regression. Consider a case where linear regression is used to predict values. The number of attributes which depend on each other can increase. As the dependency between attributes increases simple regression becomes harder use. This is where a regression tree comes of use.

The parameter I tuned for improving my results with rpart was cp. Where cp stands for complexity parameter. This parameter simply means that rpart will prune out any branch of the tree that doesn't improve the fit by at least cp. Now i started with a cp of 0.01. I noticed that as I reduced cp my CV score on RMSLE improved. Finally with one of the scores I submitted to the kaggle leaderboard.

The results I obtained are as follows.

cp	CV	Comment
0.01	0.449	
0.001	0.359	
0.0001	0.299	Kaggle Leaderboard 0.301
0.00001	0.259	

The results can easily be understood. when cp is 0.01 the technique prunes of a split which doesn't improve the fit by at least 0.01. This means that any fit which doesn't cause a 10% improvement is forgotten and discarded. This means we are losing quite a bit of splits because

we expect quite a bit of improvement. We reduce this value from 10% to 1% in the next stage and we see that the CV score improves. We reduce the cp value from 1% to .1% and finally to .01%.

One should note that if we keep reducing cp thinking that we would get a very great result, it might not necessarily be the case. This would be because we over fit for the training data.

randomForest

One of the other techniques I used to analyze the data was randomForest. This technique seemed to be one of the base techniques used for these kind of prediction and model creation. In fact for this competition the basemark they had put up used randomForest.

Though I am sure that this would have easily given me very good scores if I had used an increased number of trees, I faced quite some issues with this model. This model works by building a huge number of big trees. Now this would require a lot of memory. With the different systems I was able to use I kept facing a problem where my system would crash/freeze.

I was able to to however get access to a VM for a day where I ran RandomForest with 200 trees.

I got a cross validation score of about 0.22 and on testing the Valid.csv file I got an RMSLE of 0.23.

This was one of the best results I was able to achieve. I tried running randomForest on the train and valid file together to predict on the test.csv file. After 2 failed attempts on my system, due to limited time I have set this aside for now.

randomForest works pretty well given that it constructs many trees by picking different attributes for each true to work with.

I would expect that with a higher number of trees I would have got a much better CV score and hence more accurate predictions with randomForest. This is easily verified by also showing that the benchmark for the competition used a 500 tree construction of randomForest.

glm

glm is a straight out linear regression model. Though this is what we are trying to achieve it has no tuning nor any optimization model to improve itself. Thus it is just run straight off on the test data.

When running this on the training set I got a cross validation score of 0.46. When testing against the file train.csv t got an RMSLE of about 0.473.

The most probable reason that this has very bad score when compared to some of the other models is the fact that it probably does not take into account the interactions between attributes in depth.
It considers this only at the first level and no further interactions between attributes which is probably not the case.

Conclusion

After trying the above techniques for data analysis and prediction, there are some things I have learnt, and some things I think that I can safely conclude.

First, data processing is very necessary step without which processing of data would be very difficult. What should be kept in mind here is the fact that if this step is not performed as best as can be the following steps will have lower quality of results.

Of the techniques I tried using, I think that gbm and randomForest would have given me the best results. I am pretty sure that increasing the number of trees in both methods would have improved the results to a great extent. Though on a quick thought it may be thought that randomForest could be better, one cannot immediately conclude so. Based on the various models I built and scores I was able to get, I feel that both gbm and randomForest have their own advantages and disadvantages.

I achieved a leaderboard position of 80 in the competition and used gbm for this purpose. A system I was recently able to get some access to is currently trying to run randomForest with parameter set to achieve a very good result. But based on the scores I have been receiving while using the various models I think I can confidently conclude that both gbm and randomForest are pretty effective techniques in this case for data analysis and prediction

Based on some of the analysis done for this data set I can also comment that R, though a very easy language to work with, does not scale really well. It is noticeable that it does not support parallel processing in an elegant manner and the methods that do exist to achieve this are just “hacks” to achieve this.

References

1 G. Ridgeway. Generalized boosted models: A guide to the gbm package. [https://gradientboostedmodels.googlecode.com/git/gbm/inst/doc/gbm.pdf].Last accessed on April 25.
2 J.H. Friedman (2002). “Stochastic Gradient Boosting,” Computational Statistics and Data Analysis 38(4):367-378.
3 http://cran.r-project.org/web/packages/gbm/index.html . Last visited on April 25
4 http://cran.r-project.org/web/packages/randomForest/index.html . Last visited on April 25