Predictive Analytics and Futurism
December 2016
Issue 14

The Random GLM Algorithm: A Better Ensemble?

By Michael Niemerg

The random generalized linear model (RGLM) is a predictive algorithm based upon the idea of putting linear models in an ensemble. It does this by taking some of the features of random forests—randomization, bagging—and applies them, as the name implies, to generalized linear models. This is a seductive premise, but does it make for a competitive algorithm?

Before introducing the RGLM algorithm in more depth, let's talk about its two closely related algorithms: linear models and random forests. This will give us some background to understand both how the RGLM is put together and give us some intuition on how the algorithm might or might not be a good predictor. I will also use random forest as a basis of comparison when I test the model out later on some sample datasets.

For the most part, RGLM is using the linear regression we all know and love (more on how it does this later). It also allows generalized linear models of the logistic, multinomial, and Poisson variety. These allow us to model binary classification, classification, and counts in linear regression form respectively. Linear models have many advantages including ease of interpretability and use, fast training time, and overall versatility. For linear models, variable selection, interactions, and higher-order effects should be considered in the model-building process. Two possible ways to do this could be manually by looking at regression statistics and using good judgement or through stepwise selection procedures that attempt to do so in a more automated fashion through an iterative approach of adding and/or removing variables depending on how they improve a statistical measure. As we will see later, stepwise selection procedures will prove foundational to how RGLM is constructed.

Next up, a quick overview of random forests. Random forests are an ensemble model based upon decision trees. The random forest algorithm involves growing a "forest" of many independent decision trees where each decision tree is based upon bootstrapping (independent sampling with replacement) the dataset being modeled. Additionally, when building the decision trees, each candidate split is based upon a random subset of predictors. Once all the decision trees are created, they are then aggregated via majority voting (classification) or averaged (regression) to get a single prediction.

Random forests have many advantages as an algorithm: they can handle both classification and regression, can be trained rapidly, require modest amounts of model tuning, and do a good job of handling nonlinear interactions. Overall, the combination of these characteristics of random forests make it a powerful algorithm. One of the biggest drawbacks is that while an individual decision tree is easy to interpret, when you aggregate many of them in a random forest you lose that interpretability. However, random forests are still able to give you some insight into their inner workings through variable importance measures.

Now, let's shift focus to RGLM itself. In a sense, RGLM is a cross-breed between GLMs and random forests. Like random forests, the ultimate model is an ensemble. However, it's trying to take the advantages of random forests and apply them using linear models. That is, each base learner that makes up the ensemble in RGLM is a regression, not a decision tree. Like a random forest, however, RGLM still builds its component models from bootstrapping and by using a randomized subset of features in each base learner.

Each one of the base learners is built as follows: 1) A bootstrap sample is selected from the dataset; 2) A randomized set of predictors get selected; 3) The predictors get ranked according to their association with the variable being predicted; 4) The highest ranked predictors become candidates for selection in a regression model; and 5) Stepwise regression (specifically, forward selection) is applied to create a linear model.

For a given model, somewhere between a dozen and several hundred base learners are built. Since each one is built on a different sampling of the data and using a different set of candidate predictors, each one will be unique. Once each of the base models is created, they are then combined into the final ensemble model used for prediction using either averaging for regression or majority voting for classification.

The intuition here is that by randomly sampling both the datasets and the predictors, the ensemble model is more powerful than a single model could be. Ideally, an ensemble allows for a good deal of flexibility in the model it creates, but avoids overfitting since the noise tends to get washed out among the different models. However, RGLM is going to have a challenge in that an ensemble can only be as flexible as its base learner and linear regression base learners can only take this so far.

The reason for this limitation is that the best model involves a trade-off between bias and variability and linear regression is a high bias, low variance procedure whereas ensembles benefit most from base learners that are the opposite. We need low bias base learners in our ensemble so that the resulting model has low bias. We need the "flexibility" that often comes from high variance base learners to ensure that the ensemble is capturing all the signal in the data. Any amount of overfitting caused by high variance within the base learners gets muffled by averaging them, so that the final model will actually have low variance.

Because RGLM has a linear regression base learner, extreme outliers could still dominate even an ensemble while "wiggly-ness" and complex relationships in the data could be hard for RGLM to capture. See, for example, the following charts which show a dataset that is very linear except with a highly nonlinear subregion (Figure 1). A fitted linear regression produced by RGLM (which is virtually identical to the regression model that would be produced with an ordinary GLM) is shown in orange while the random forest is shown in gray (Figure 2). While this example is obviously contrived, it conveys the difference in flexibility between the two models. The random forest is able to identify the anomalous subregion and is able to average out the behavior in that vicinity without much ado while the entire intercept of the linear regression gets thrown off by those points (see how its prediction line hovers above most of the data points). The predictions are simply too high everywhere, except in the anomalous region where the model prediction is too low.

FIGURE 1: TEST DATASET

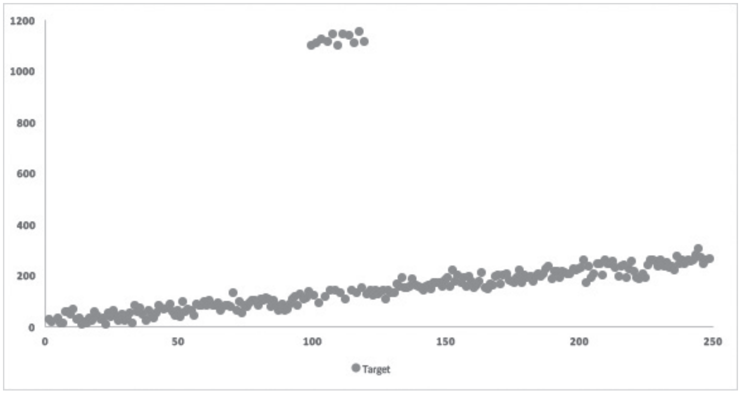
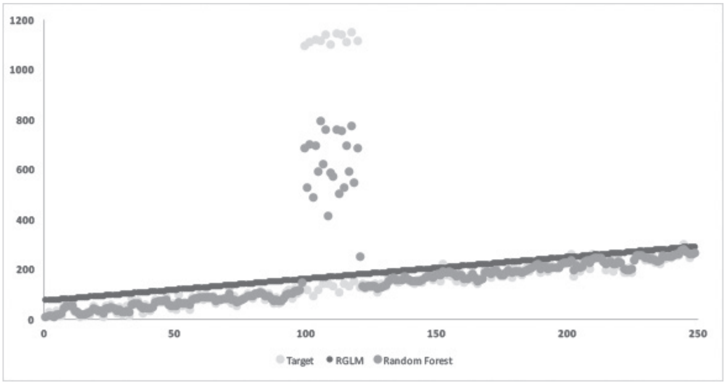


FIGURE 2: TEST DATASET WITH RGLM & RANDOM FOREST PREDICTION



All right, now that we've built some intuition on how RGLM works, let's take it out for a test drive. I chose ten datasets: four suited for logistic regression and six for linear regression.

To summarize the predictive accuracy for logistic regression, I used area under the curve (AUC). AUC is a common measurement when comparing binary classification models. For any model that can return probabilistic output, a receiver operating characteristic (ROC) curve can be constructed. An ROC curve graphs the true positive rate vs. the false positive rate at the possible thresholds for classifying an observation. The AUC is then a measurement of the area under this curve. The higher the AUC the better. To compare predictive accuracy for linear regression datasets, R-squared was used. R-squared is a measure of the strength of linear association between two variables, again the higher the better. I will be calculating these statistics for each dataset on a testing set that was withheld from the model fitting process so that we can get a good sense of model generalizability on new data.

It turns out that random forest outperforms RGLM in the datasets I chose, sometimes by a wide margin. The difference varies

The results, as well as a description of the datasets used, are shown in the following table:

				RANDOM FOREST		RGLM	
Dataset	Model Type	Observations	Predictors	RunTime (seconds)	Evaluation Metric*	RunTime (seconds)	Evaluation Metric*
1	Classification	45000	16	8.15	92.4%	402.22	81.2%
2	Classification	27000	10	2.17	87.1%	142.64	79.6%
3	Classification	27000	100	19.92	88.5%	4463.66	76.2%
4	Classification	800	8	0.05	80.1%	44.98	79.5%
5	Regression	1500	5	0.16	76.9%	5.12	56.9%
6	Regression	9000	12	4.83	99.8%	55.96	93.7%
7	Regression	6000	19	0.64	69.7%	23.18	14.8%
8	Regression	1000	8	0.11	87.4%	5.91	51.0%
9	Regression	10000	4	1.58	99.3%	18.57	100.0%
10	Regression	5000	11	1.61	48.7%	27.39	19.2%

* The evealuation Metric is AUC for classification and R-squared for regression

by dataset with RGLM coming in very competitively for some of the datasets and random forest coming in as an easy winner on others. One RGLM, trained on dataset 9, just barely beat the random forest, and its accuracy rounded up to 100 percent. In terms of computation time, random forest was a clear winner across the board, being an order of magnitude faster to train.

One thing to note is that I tried various configurations of RGLM. The above chart doesn't represent the absolute best I was able to get out of RGLM, but involves a compromise of predictive accuracy and runtime (the parameters are 50 base learners, interactions up to level two, and with each model considering half of its predictors in the base learner as a candidate for forward selection). Some further optimization is able to close some of the gap¬¬¬¬—and RGLM was able to ever so slightly outperform random forest on another dataset with some additional model tuning—but the overall gap remains. I wasn't able to come up with any configuration that made RGLM the clear winner over random forest. Meanwhile, I didn't do any tuning or optimization to the random forest (I simply started with 40 trees and default settings). Furthermore, parameter tuning can only get you so far with RGLM. There aren't that many parameters to tune to begin with. At the end of the day, I think the true drawback is that RGLM is still a linear model which means that it is somewhat limited in its ability to capture highly nonlinear interactions, as mentioned earlier.

To elaborate on that point some more: ultimately, any RGLM ensemble is really just a linear model itself (a combination of linear models is itself a linear model) and so it is inherently limited by all the things that linear models are limited by. In fact, RGLM is probably even more limited to the extent that it is simpler to add higher-order terms and transformations onto a typical linear model than to an RGLM.

One caveat I feel compelled to mention: a truly fair and robust comparison would require a larger sample of datasets. In fact, a much more robust comparison of RGLM to other methods was performed in the paper by Song and Horvath this article was based upon (see the references at the end). In their results, the creators of RGLM were able to get superior performance on RGLM even when comparing it to many of the most common algorithms used today in predictive analytics.

Overall, I haven't seen RGLM used much in practice. Based on Song and Horvath, it seems it can offer superior performance on some datasets, but I'm skeptical of its ability to do so reliably on a wide range of applications. Also, due to its rather lengthy computation time, I'd be hard-pressed to recommend it as an all-purpose algorithm. It's an interesting concept, but I can't help but think it needs some alterations—some way to be a little more like a random forest, some way to alter its base learners to add flexibility, capture nonlinear features, and better benefit from the ensemble approach—before being able to be a top tier contender. ■

Michael Niemerg, FSA, MAAA, is an actuary at Milliman in Chicago. He can be reached at *michael.niemerg@milliman.com.*

REFERENCES:

Hastie, Trevor; Tibshirani, Robert; Friedman, Jerome (2008). The Elements of Statistical Learning (2nd ed.). Springer. ISBN 0-387-95284-5.

Song L, Langfelder P, Horvath S (2013) Random generalized linear model: a highly accurate and interpretable ensemble predictor. BMC Bioinformatics 14:5 PMID: 23323760 DOI: 10.1186/1471-2105-14-5.