White Paper

Netflix, Dark Knowledge, and Why Simpler Can Be Better

Andrew Lutes
Data Scientist

December 22, 2015

ELDER RESEARCH
DATA SCIENCE · AI · MACHINE LEARNING

Table of Contents

Weary from an all-night coding effort, and rushed by the looming 6:42PM deadline, Lester Mackey searched frantically for the proper prediction file to submit. Lester was a member of "The Ensemble"—a large coalition of data scientists who had joined forces late in the great Netflix challenge, after a grand prize of $1 Million — not to mention substantial prestige in the data science community. After 2 years of effort and frequent lead changes (lately, mostly with rival mash-up team "Belkor's Pragmatic Chaos"), Lester's team looked to have finally achieved its goal….

Rewind about 2 years to October 2006: Netflix unleashed its challenge to crowd source one of its most difficult challenges: making quality movie recommendations to its customer base. The competition goal was to use a customers' previous movie reviews to predict how many stars they would give a particular movie in the future. $1 Million was promised to any team able to beat Netflix's current algorithm by more than 10% in root mean squared error. Within 6 days, Netlflix's in-house algorithm had already been topped and within a few months, significant improvement had already been made. A team of my colleagues at Elder Research reached 3rd place among 20,000 entries before moving on from the competition (due to hedge funds using the leaderboard to try to recruit away talent!). As for "The Ensemble", they would ultimately surpass the 10% improvement mark a mere 9 minutes before the close of the competition, finishing at the top of the accuracy leaderboard. However, by the rules of the Netflix prize, accuracy scores would be rounded off to four decimal places and order of submission would be used as a tie breaker. It just so happened that "Belkor's Pragmatic Chaos" had gained roughly the same score and had edged out "The Ensemble" by 20 minutes, thus claiming the $1M. Belkor had won by a nose; A 2-year race decided by a 20 minute span is akin to a marathon decided by a tenth of a second. If there is a greater saga of competition and drama in data science nerdom then I've never heard it.

Despite the success of the competition for Netflix – both technical and marketing – *they never used the winning algorithm*. The competition was far from a failure; Netflix was able to incorporate a couple of simpler components of the model into production. But why didn't Netflix switch over to an algorithm which had beaten out tens of thousands of smart competitors and shattered their own model's accuracy? Netflix had paid out a million dollars for this product – *why not use it?* It turns out that the winning model, despite its accuracy, was incredibly costly both to train and to implement. The algorithm used a gradient-boosted decision tree to ensemble over 500 unique simpler models. The competition, in an effort to be objective, was awarded by a single measure, but it was—by nature—blind to the host of other factors that contribute to the successful deployment of a model. The algorithm also took so long to develop that the problem faced by Netflix had changed dramatically by the time the competition closed (as addressed more below). By setting such a lofty goal for this metric, Netflix forced competitors to abandon any other considerations as they piled complexity onto their models in the pursuit of this singular target—which was no longer relevant once it was reached.

For data scientists, accuracy is almost always the prime objective. It's how we measure improvement and success. Without it, our models aren't worth using. But, as any good

analyst will tell you, a single number could never tell the whole story. Real life is not a competition, and the comparison between two models is never as simple as a single measure. An analytics practitioner should seek to maximize the return on investment (ROI) he/she provides for a client, not just the accuracy of the model. This ROI is driven primarily by two competing objectives: the additional benefit provided by an analytics solution, and the costs to train, maintain, and deploy a model. In general, the solution to this tradeoff is elegance. In a world of complexity, the simplest workable solution is the one that will likely get your client the most benefit for the least cost.

When to Stop Training

Imagine you have been working on a prediction project for several weeks. Your accuracy has begun to plateau and you must decide how much more to tweak your model to make it better. You consider spending some more time cleaning your input data to eliminate some noise. You could also test different hypotheses, alter your modeling algorithms, or ensemble multiple models to try to squeeze out every ounce of accuracy from a set of data. Every data scientist has been in this position, and it's easy to see how this process could go on *ad infinitum*. The question is when to quit — how good is good enough? A good way to answer this question is to make sure that you have a way to track the efficacy of improvement so that continued investment in an analytics project can be properly valued.

Don't Just Measure Success – Valuate It

Most analytics projects have quantitative accuracy criteria leading their metrics for success. Take a simple binary classification problem: success is often measured by percent correct. It's also important to figure out how much that extra 1% of accuracy in your model will actually be worth to a business. However, the type of error can matter: a false alarm can be a nuisance, say, whereas a false dismissal can be a disaster. It is a big improvement over %correct, to simply associate a cost to each type of error as well as a payoff to each type of correct prediction and calculate an expected payout to any action based on a predictive model. If you multiply this by the number of decisions that you need to make using your model, you have the value of the entire model's accuracy. However, sometimes analytics projects can be exploratory in nature, with more goals more qualitative and strategic than quantitative and tactical. In these cases, it's important not to forget the bottom line. Look at the possible decisions that could be made based on your analysis, and find a way to estimate the potential value in exploring each area.

Compare to a Baseline

You can never really know the value of your model without a basis of comparison. When you are pitching a project or showing the benefit of your model, you should compare your analytic solution to whatever baseline is already in place. However, as you are working on the project, it is better to use a smart baseline that moves with you. As incremental progress is made, continually update your baseline to assess your *marginal* rate of improvement.

If possible, also have an accuracy ceiling. Every system in the real world seems to have a base of unavoidable randomness which science can never break through with

available data, so it's foolish to try. For example, sabermetrics gurus have examined multiple sports to determine what proportion of outcomes were due to talent versus chance. Assuming we could assess which team is better, we would still get some predictions wrong because the better team sometimes loses. This kind of analysis gives modelers a theoretical upper bound on the long-term accuracy of predictions. As we approach this limit, we can stop trying to squeeze out more accuracy because the returns will be limited.

Expect Decreasing Returns

Improvement in modeling usually isn't linear. You can often accomplish 80% of the ideal solution with 20% of the work. While we often want to believe that our next hypothesis will be a game-changing breakthrough, this happens very rarely (which is not to say never!). Once you've reached a steady state, expect that each improvement you make will bring less value than the last.

Cost of Monitoring and Management

Concept Drift and the Period of Impact

No model is eternal because every system will change over time. There are many reasons these changes occur. Sometimes there is a change to the laws or standards that govern the system. Other times, by taking strategic action via the insights from a model, we alter the very system we once understood. If we take advantage of the once-predictable behavior of actors in a system, for example, then those actors will often alter their behavior in response. Take a set of fraudsters who can observe certain types of risky transactions being caught or flagged and quickly move on to different fraud tactics. A third cause of change is a rapidly evolving industry. Look at Netflix: the user base, the movies available, and even the delivery system evolve on a very quick time scale. Any predictive model it develops will need to be altered in some way fairly frequently. The company faces a far different challenge today as it did back in 2009, as a large portion of its business has shifted to a streaming service. This shift dramatically altered the landscape of Netflix's prediction problems. Netflix also faces a continual evolution in user-base and in popular movies, requiring smaller shifts in strategy. In all of these situations, when we consider the value of a model, we need to get a rough idea of how long the model will last and how dramatic any changes will need to be. Important questions to ask at the outset of a project include: *how stable is the system we are trying to measure? How smart are the actors in it? How liable is it to change, and how quickly?*

Maintenance and Agility

The solution to an evolving context may mean training on more recent data, tweaking parameters, or rethinking the foundation of a model. A highly specialized model like the Netflix challenge winner would be incredibly difficult to translate into a new environment and would require a substantial continued investment. Netflix would have to retain staff who could understand and maintain all of these models – a difficult task when you consider that the model was developed as the combination of efforts of several brilliant PhDs, each working for two years at the problem. Someone has to continually monitor the accuracy of a model and understand the inner workings well enough to make

changes when necessary. This process is easiest when a model is light and agile, allowing for quick adaptation and re-training. Don't allow gaps in your capability when the evolution of a system occurs faster than your ability to continually update it with understanding.

Cost of Deployment

Of course, the ROI for an analytics project is only one side of the coin. As we add additional bulk to a model, we don't just require more of our own time; we also can incur costs downstream by making models more difficult to deploy. Because of the effect of prediction costs and maintenance costs, simpler models are often the most effective solution.

Prediction

Complex models can take up a lot of computation power to score, especially lazy algorithms such as KNN where the data *is* the model, or ensembles which must combine several unique predictions. In an environment with high data velocity, these computations can quickly become prohibitive. Netflix, for example, has over 57 Million users and generates about 30 Billion predictions per day to users on a variety of platforms and devices. The cost of implementing the winner of the Netflix challenge would require scoring users through over 500 different models and then combining results to generate over 3 Million predictions every second. Think about the server costs to crunch all that data! Or worse, having users wait several seconds while the engine produced the perfect movie predictions for them. It's unlikely that these costs could be justified by the extra accuracy that this complex model provides over simpler models. It's important at the outset of a project to ask *how often* and *how quickly* models will need to make predictions and weigh the flexibility of these goals against the accuracy of the model.

Interpretation and Incorporation

One should also keep in mind how a model is going to be applied and how complexity will increase that overhead. My last article focused on how understand can hinder acceptance, and how black box models run a greater risk of being cast aside. The output of models will often need to be looped into a business process or communicated to an end user who applies the information the models generate. Model output can be delivered through applications or visualizations which require their own development cycles, as well as user training and documentation. Complex models are more difficult to document and communicate and will likely cost more to deploy. Consider early on how much of the inner workings of a model should be exposed, how detailed a user's understanding of the model needs to be, and how exactly the model will be applied.

Great, so how can I stay simple?

What does all of this mean? Should we declare success every time we hit a reasonable model, abandoning the iterative process of improvement that will drive our models towards greater accuracy? Obviously not. Imagine a model on credit risk which has a multi-year horizon and an extraordinary return on accuracy. In this case, very complicated methods which add a small amount of predictive power could easily be

justified. However, in some situations, complexity must be considered, and we should have some tools to translate complex models into simple ones.

Geoffrey Hinton, data science researcher for Google, recently unveiled a technique he dubbed "dark knowledge." This method consists of training giant, complex, deep learning models, and then using the output of those models as "soft targets" to train simpler models. The "soft targets" identify not only the information we are trying to predict, but also indicate which data points may be difficult to predict and why. For example, when classifying fruit, a soft target could tell you which fruit are apples, and also which of those apples may be more likely confused for oranges or bananas, compared to other apples. In one example, Hinton trained an ensemble of ten neural networks to classify audio samples and achieved 61.1% accuracy on a holdout set. He then trained a single neural network, but instead of using real targets, he trained it on the outputs of the previous ensemble. Using this method, he taught a much lighter model to *behave* like the more complex one and achieved comparable results of 60.8% accuracy. One can apply the same thinking to train simple *classes* of models to behave like more complex ones. For example, one could find that a k-nearest neighbor (KNN) algorithm is very successful in a classification problem, but that KNN generates predictions too slowly for deployment. Instead, one could perhaps train a logistic regression on the outputs of KNN, teaching it to behave like the KNN model, but allowing for quicker prediction generation. This method allows a simple model to more quickly converge towards the best possible model that could be trained rather than towards the absolute truth. Think back to the sports analogy; we want to uncover who the better team is, which isn't necessarily who wins the game. This clever insight may allow data scientists to greatly simplify models while sacrificing little in the way of accuracy, ultimately reducing deployment costs.

Data scientists are exceptional at weeding out noisy trends which won't generalize to future data, but it's time we start broadening our definition of noise. As nice as it feels to be the most accurate scientist on the block, we should measure a model's value relative to a holistic set of trade-offs surrounding deployment. We may be great at choosing the number of layers in a neural net, the number of nodes in a decision tree, or the number of parameters in a logit model to optimize accuracy, but we also must choose parameters with flexibility and deployability in mind. A model should only include a given bit of complexity if it provides a commensurate amount of insight or predictive power. George Box once noted that "all models are wrong, but some are useful." Rather than trying to chase the perfect prediction by building an atom-for-atom replica of the world, we should embrace the inevitability of error. A model should seek to abstract out complexity and focus in on the parts that matter.

About the Author

Andrew Lutes is a data scientist for Elder Research Inc. He enjoys using analytic techniques to find order and meaning within noisy and complex systems. Andrew's technical focuses include causal analysis, experimental design, and predictive analytics. Andrew has applied analytics to support agencies within the

Department of Homeland Security & Department of Labor, to aid corporate investigations, to inform strategic decision making for growing start-ups, and (in his spare time) to assess decision making in sports. Andrew earned a B.S. in Systems Engineering and a B.A. in Economics from the University of Virginia. Currently, he is pursuing an M.S. in Mathematics and Statistics from Georgetown University.

www.elderresearch.com

ELDER RESEARCH

DATA SCIENCE · AI · MACHINE LEARNING