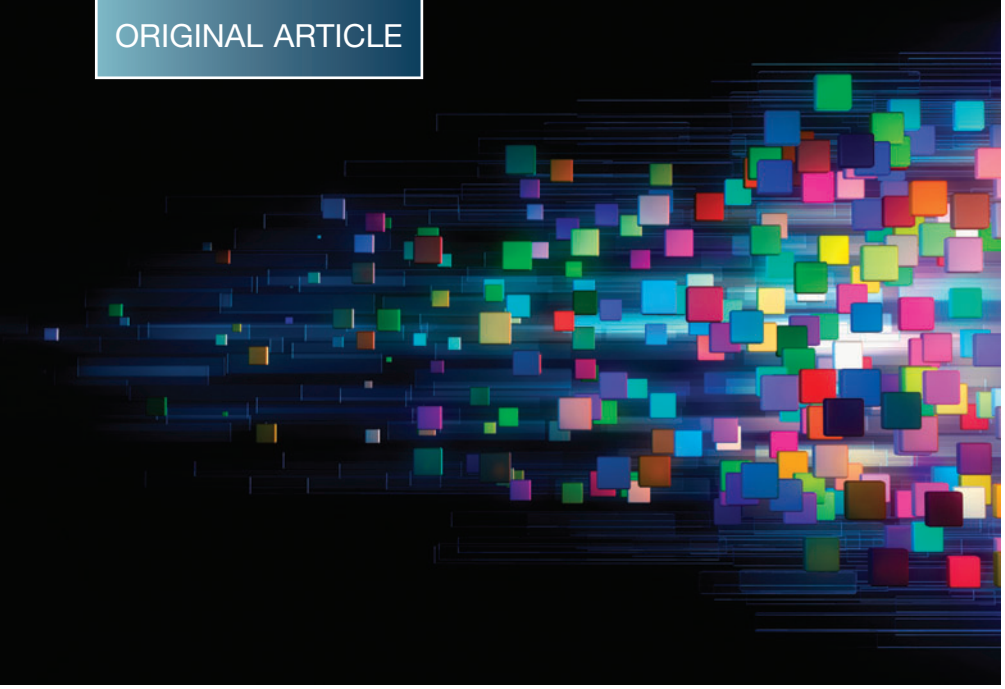
ORIGINAL ARTICLE



BRING THE NOISE:

Embracing Randomness Is the Key to Scaling Up Machine Learning Algorithms

Brian Dalessandro
Data Science, media6degrees (M6D), New York

Despite the progress made in scaling up data processing and storage, major bottlenecks still exist when using big data for statistical inference and predictive modeling. Many classic algorithms used for predictive modeling with big data are constrained by the very math that defines them. These algorithms tend to have complexities that do not scale linearly, so doubling your data often means quadrupling the time it takes to learn models. Many of today's statistical modelers have been trained using the algorithms and software that have largely been optimized for a "small data" paradigm. New tools and new ways of approaching many machine learning problems are now called for in order to scale up industrial-sized applications. Fortunately, new theory, intuition, and tools are emerging to solve these problems. In this article, we introduce the works of Leon Bottou and John Langford (both coincidentally at Microsoft Research), two leading researchers in the field whose research missions have been to scale up machine learning. The work by Bottou, Langford, and their colleagues shows us that by adopting algorithmic strategies that classically would be considered inferior, we can paradoxically both scale up machine learning *and* end up with better predictive performance. From a predictive modeling perspective, more data certainly is better, but only if you can use it. The strategies presented here offer practitioners promising options to making that happen.

Understand Your Error

Bottou's and Langford's[2,4] research efforts are particularly applicable to the types of predictive modeling that are heavily utilized in industrial applications. The branch of machine learning that governs most predictive modeling methods is called supervised learning. In supervised learning, one uses the features (or variables) available to describe an entity as a basis to predict some other event, item, or feature of interest. Some classic and often-used supervised learning methods are linear and logistic regression, decision trees, and support vector machines. All supervised learning methods are characterized, and in some sense defined, by a loss function. The loss function measures how well the specific model predicts the target variable of interest. The most intuitive of such loss functions is accuracy, but others exist, and each one has specific properties that make it more or less suitable for certain problems over others.

If we think of the loss function generally as the total amount of "error" in our predictions, we can actually decompose this error into different components, where each component is driven by certain decisions we make when training the model. One classic decomposition breaks the error down to approximation and estimation error. Estimation error comes from the fact that when building models, we often only have a finite sample of the underlying data at our disposal. If infinite data were available, we may be able to learn exactly how feature inputs should map to the prediction. The difference between the model derived from a finite sample and the model we should expect if the data were infinite is called the estimation error. The approximation error is the difference stemming from how the model *should* look and how we decide it *will* look. To exemplify both, let's assume there exists some process defined by: $Y = 3 * X_1^2 - 2 * X_2^2$. Let's assume the modeler uses a linear regression of the form $Y = \beta_1 * X_1 + \beta_2 * X_2$ to fit this data and has only 100 examples to build the model. The error caused by using linear terms as opposed to quadratic terms would be classified as the approximation error, and the error caused by having only 100 examples is the estimation error.

In several articles, Bottou[1,2] has introduced a quantity called optimization error to the decomposition of a model's overall loss. The intuition behind optimization error is that some algorithms are better than others at finding the minimum of the loss function being optimized, and the error induced by

BIG DATA JUNE 2013 • DOI: 10.1089/big.2013.0010

using an inferior algorithm is the optimization error. An interesting question that follows is: Why choose an algorithm that is expected to perform worse? The answer to this, and a central theme of Bottou's work, is that by deliberately choosing an inferior algorithm we are likely to get better results from a prediction point of view. The reason is that our design choices enable us to trade one type of error for another, and done correctly, we end up better off.

The star algorithm of Bottou's work is called stochastic gradient descent (SGD). SGD comes from a family of optimization algorithms called gradient descent, and the more advanced (and complex) versions of the algorithm are used within the native functions of statistical packages such as R, SAS, and Matlab. As data scales up (particularly with the number of features), packaged statistical learning programs become extraordinarily slow. This puts a natural constraint on the amount of data that can be used, which in turn, increases both approximation and estimation errors. SGD on the other hand is defined in such a way that it is perfectly amenable to larger datasets. It searches for a min/max one data point at a time and takes advantage of data sparsity when it exists. The reduced memory dependence and linear scaling with respect to feature dimensionality makes this much more practical in big data settings. The "stochastic" qualifier comes from the fact that SGD approximates characteristics of the entire dataset one line at a time. The search for the solution is more erratic because of this and is why the algorithm classically has been considered to be inferior. So by choosing SGD, one introduces more optimization error into the model, but using more data reduces both estimation and approximation errors. If the data is big enough, the trade-off is favorable to the modeler.

In Bottou's words:*

> *Stochastic optimization algorithms were invented in the fifties and have been used very successfully in many fields such as adaptive signal processing. However, they were long regarded with suspicion in machine learning because they can be finicky and often perform poorly on small [data] problems. The last five years have been marked by the realization that these algorithms perform very well on large scale problems (which could seem counter-intuitive at first glance).*

When Features Collide

Many breakthroughs happen by embracing what seems counterintuitive at first. Just as Bottou has shown that using an inferior optimization algorithm can actually improve predictive performance, John Langford and colleagues have introduced a dimensionality reduction technique that, despite its simplicity, has proven to be highly effective in big data contexts.[3] The new method, called *feature hashing*, reduces feature dimensionality by randomly assigning each feature in the original space to a new dimension in a lower dimensional space. This is done by simply hashing the ID of the original feature and using the resulting hashed integer as the new feature ID. The works by Langford et al. give clearly written algorithms on how to implement this and under what contexts it should work best.[3] This technique creates a trade-off, because with high probability, multiple features will be hashed into the same new feature ID. Generally, all dimensionality reduction techniques will degrade the data to some degree, but most have been designed to preserve geometric qualities of the data as best as possible. Feature hashing, on the other hand, makes no effort to preserve data quality. In the right context though, the cited work shows that the degradation induced by feature hashing is minimal, and its practical benefits more than outweigh the costs. Specifically, feature hashing scales linearly, uses very simple preprocessing, and preserves data sparsity when it exists. These properties make feature hashing the natural (or only) choice when the data scales to millions or billions of features.

"MANY BREAKTHROUGHS HAPPEN BY EMBRACING WHAT SEEMS COUNTERINTUITIVE AT FIRST."

The power of the feature hashing method is well illustrated in the seminal application given by Langford and colleagues.[4] E-mail spam filtering is a good example of modern machine learning problems: 1) The data is large yet sparse, 2) data is generated in a streaming process, 3) solutions need to scale to real-time deployment, and 4) the solution benefits from having mass customization. In the feature hashing work, words within e-mails are used as features, and mass customization is achieved by creating interaction terms between each user and each word. Such a data design creates a feature dimensionality proportional to the number of words times the total number of users. Even though this data matrix is extremely sparse, the size of it could create major problems in the production system that trains and deploys the model. Feature hashing reduces the dimensionality from about 1 billion to 1 million. This is still large, but not so large that the solution becomes intractable. The trade-off again is to degrade the data slightly (by inducing random collisions of the features) but enable the use of more features (i.e., the user-term interactions). From the point of view of error trade-offs, allowing for a more complex and feature-rich model reduces the approximation error, and this reduction is more than enough to offset the error caused by the collisions.

*Direct correspondence with the author

Toward Scaling Up

Machine learning methods used in modern industrial applications need to be able to scale up in every possible aspect of their use. The three main functions in deployed machine learning applications are data collection/processing, model training, and model execution. Parallelized systems have enabled data processing and retention to happen at unprecedented scale, leaving the bottlenecks in the learning and deployment of predictive models. Fortunately, research groups that bridge the academic and industrial divide have been addressing these challenges directly. The type of research exemplified by Langford, Bottou, and their colleagues offer practical solutions to increasingly common problems in industrial machine learning applications. Their work is more than just mathematical theory; strong empirical results as well as increasing rates of adoption support the proposition that applications of machine learning are indeed scaling up. SGD enables training of prediction models with minimal memory requirements, exploits data sparsity, and even supports a truly streaming approach to training models.

Feature hashing brings many benefits to an industrial production system. The reduction of a very large and potentially shifting feature space to a smaller and fixed feature space enables the use of more complex feature representations while keeping the memory constraints of the working system fixed. Additionally, feature hashing enables a more privacy friendly way to store and use personal data. In both of these methods, the research suggests that what at face value might seem like a shortcut is actually an enhancement as the datasets become bigger and bigger. Nothing of course comes free (or at least without necessary caution). Both of these methods involve embracing a little chaos, and in doing so, one risks having chaos in the results. When using SGD and feature hashing, one ought to have a clear understanding of the underlying algorithms and proceed with careful experimentation and testing before deploying. Embraced methodically and properly though, these methods are likely to become de rigueur tools in the big data machine learning arsenal.

Author Disclosure Statement

No competing financial interests exist.

References

1. Bottou, L., and Bousquet, O. The tradeoffs of large-scale learning. Optimization for Machine Learning 2011; 351.
2. Bottou, L. Large-scale machine learning with stochastic gradient descent. Proceedings of COMPSTAT'2010 Physica-Verlag HD 2010; 177–186.
3. Shi, Q., Petterson, J., Dror, G., et al. Hash kernels for structured data. The Journal of Machine Learning Research 2009; 10:2615–2637.
4. Weinberger, K., Dasgupta, A., Langford, et al. "Feature hashing for large scale multitask learning." Proceedings of the 26th Annual International Conference on Machine Learning. ACM, 2009.

Address correspondence to:

Brian Dalessandro
media6degrees (M6D)
Vice President, Data Science
37 East 18 Street, 9th Floor
New York, NY 10003

E-mail: briand@m6d.com