Brazilian Journal of Probability and Statistics
2017, Vol. 31, No. 4, 697–700
https://doi.org/10.1214/17-BJPS368
Main article: https://doi.org/10.1214/17-BJPS365

Rejoinder

Steven L. Scott
Google

I thank Gelman and Vehtari (GV) and Draper and Terenin (DT) for a thoughtful discussion. Both pairs of authors have worked with big data "in the trenches" for a long time, and bring valued expertise and perspective.

Both GV and DT observe that big data is only really necessary when fitting complex models, which typically means models involving many parameters. I agree, and obliquely made the same point in the article by focusing on what happens to methods of improving on consensus Monte Carlo (CMC) as the parameter dimension p increases. Unfortunately for the methods investigated in the article, it seems the curse of dimensionality bites hard, with averaging the only method that is unaffected by dimension.

The good news on dimensionality is that in many applications from the tech industry, much of the "bigness" comes from sparse data, such as factors with vast numbers of levels. For example, one element in your model might be a dummy variable indicating a URL or web service that referred a visitor to your page. There are effectively infinitely many potential levels for such a factor, but for any particular visitor all but one are irrelevant. Modeling such a factor using random effects can turn a problem where the parameter dimension p is infinite into one where p is reasonably small. This blurs the line between two cells in DT's table, allowing sharding methods to leach into the "big n, big p" cell. Admittedly, replacing parameters with random effects is a trick that won't work in all problems, so perhaps DT were correct in applying the "model specific" label to that cell.

On the slippery question of what "big data" actually means, both GV and DT adopt the view that "big data" implies a data set which is too large for standard methods to work comfortably. I'd like to point out that this perspective, which is shared by many statisticians, can confuse engineers who view data as either "big" or not. Imagine two analysts working on the same data set. One computes an average, while the other fits a complex Bayesian model using MCMC. Does it make sense to say that one analysts has a "big data problem" while the other does not? There are obviously viewpoints from which the answer is "yes," but it is equally obvious that the difference is not the data, but the algorithms used to process it. One analyst has an algorithm that scales better than the other. Classical MCMC methods scale poorly because they assume you are able to loop over the data at will.

Received June 2017; accepted June 2017.

Recent improvements make better use of processing power, either through mathematical methods like subsampling or the continuous MCMC methods mentioned by DT, or through hardware methods like GPU's. Doing so brings Bayesian inference to many problems for which it was previously impractical. Even so, memory and disk limits will eventually be reached. When they are, cluster or cloud based solutions are essentially the only game in town.

GV correctly call out the fact that consensus Monte Carlo can lead to difficulties with certain priors, particularly those used for regularization. Given the importance of sparsity-inducing priors in high dimensional problems this is a potentially serious drawback. I agree that their preferred method of expectation propagation (EP) handles priors elegantly, and the modest number of communications involved with EP don't impose a significant computational burden. My worry with EP is that it produces an approximate model which is the endpoint of the analysis. The flavor of the approach is similar to the "variational Bayes" methods currently in vogue among the machine learning community. These methods are an acceptable replacement for Monte Carlo estimates as long as the approximating distribution $g(\cdot)$ can be made arbitrarily close to the actual posterior. Nonparametric models (e.g., mixtures, or deep networks) might be able to get close enough to be useful approximants, but the "mean field" methods that currently dominate machine learning are inappropriate for a full Bayesian analysis, where coherency arguments demand that the posterior distribution be treated as a joint distribution.

While I share GV's view that mixtures can (in principle) be used to approximate posteriors, I differ with GV about whether they should be the final deliverable. This argument is the mixture-model equivalent to the different roles that normality plays in Scott et al. (2016) vs. Huang and Gelman (2005). Huang and Gelman (2005) explicitly construct a normal approximation to the posterior by computing the mean and variance of the posterior draws on each worker and then combining them using Rubin's rules. Scott et al. (2016) apply an averaging procedure that is known to work for normal data to a series of non-normal examples. Normality is a sufficient, but not necessary, assumption for averaging to produce satisfactory results. The averaging procedure can capture features (like skewness) that explicit normal approximations destroy. I had hoped to demonstrate a similar effect for the the local averaging procedure in the current article, but the mixtures in question proved too expensive to fit beyond 10–20 dimensions. I retain some optimism for the idea, but I have to conclude that my current implementation is impractical. I suspect that fitting the high-dimensional mixtures needed for EP will be similarly challenging. If our ability to fit high dimensional mixtures were suddenly improved, it would benefit both methods.

One way to unify variational methods (and EP) with the more traditional Bayesian Monte Carlo framework is to use the approximate posterior distribution as a proposal distribution in a Monte Carlo algorithm, such as SIR. If the approximation is nearly exact then most of the resampling weights will be uniform. If Gaussian mixtures are used as the basis for approximation, then simulating from

the approximation is trivial. The simulations can be made in batches, and the likelihood evaluations from the various worker machines needed for resampling can be computed in parallel. Most Bayesian problems require one or more margins of the full posterior distribution. Unless the form of the variational approximation is chosen to make the desired integrals easy, some kind of Monte Carlo will be necessary to compute the desired posterior marginals anyway, so using variational approximations as proposal distributions seems like a natural step.

As a minor issue, GV raise the question of whether Monte Carlo is a vital part of the algorithm, and thus whether we need the name "consensus Monte Carlo." Because of its role in integration, Monte Carlo is central to Bayesian inference, and I think there is a difference between Bayesian and frequentist approaches to big data problems. The frequentist is free to propose an arbitrary estimate. Averaging sharded point estimates can be defined as the estimate, at which point the problem becomes one of understanding the bias and variance properties of the estimate (Zhang, Duchi and Wainwright, 2012). Bayesians have a "right answer" provided by Bayes' rule. If we produce something other than draws from the posterior distribution, then it is unclear what our analysis represents.

Switching focus to DT's discussion, I agree with the implication from their table that reasonable strategies for Bayesian computation depend on available hardware in addition to the size of the problem. DT are correct to point out the universal access to massive computing resources for tiny amounts of money. However the \$1/hour approximate rate they mention is off by two orders of magnitude. The cheapest cloud machines can be had for closer to \$0.01/hour. You can hire 1000 machines at a lower hourly rate than an undergraduate student assistant! Your 1000 machine ensemble comes not just with processors, but with multiple terabytes of (distributed) RAM, and many times that amount of disk. Using those resources effectively is the challenge of the day.

I share DT's desire for a user-friendly computing language that makes efficient cloud computing possible, but until we know what such a language needs to accomplish statistics researchers will most likely be forced to compute in languages favored by computer scientists, such as C++, java, or python. Computer scientists are not sitting still, of course, and there are opportunities to capitalize on their recent advances. Google's Go language was designed for parallel data processing, while the TensorFlow framework was designed for parallel model fitting. TensorFlow also makes DT's table outdated, as new hardware (tensor processing units, or TPU's) has been developed that are even more powerful than GPU's (Wikipedia, 2017). TPU's currently must be utilized through the TensorFlow framework, which is the current state of the art for deep learning, but which may or may not be a good fit for Bayesian computation.

DT ask about the relationship between CMC and the other methods in their table. The nice thing about CMC is that it is agnostic about the algorithm used on each worker machine to generate the worker-level Monte Carlo draws.

As a final note, both GV an DT observe that an analysis of a model's mathematical structure can lead to a more efficient Monte Carlo exploration than a black box MCMC algorithm. This is true for single machine algorithms, and will doubtless be proven true for distributed systems as well. See Bumbuca, Misra and Rossi (2017) for a recent example involving hierarchical models.

References

Bumbuca, F., Misra, S. and Rossi, P. E. (2017). Distributed Markov chain Monte Carlo for Bayesian hierarchical models. Technical report, available at https://ssrn.com/abstract=2964646.

Huang, Z. and Gelman, A. (2005). Sampling for Bayesian computation with large datasets. Technical report, Columbia University Department of Statistics.

Scott, S. L., Blocker, A. W., Bonassi, F. V., Chipman, H. A., George, E. I. and McCulloch, R. E. (2016). Bayes and big data: The consensus Monte Carlo algorithm. *International Journal of Management Science and Engineering Management* **11**, 78–88.

Wikipedia (2017). https://en.wikipedia.org/wiki/Tensor_processing_unit.

Zhang, Y., Duchi, J. C. and Wainwright, M. J. (2012). Communication-efficient algorithms for statistical optimization. In *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*, 6792.

Google
1600 Amphitheatre Parkway
Mountain View, California 94043
USA
E-mail: stevescott@google.com