Consensus Monte Carlo using expectation propagation*

Andrew Gelman† Aki Vehtari‡

28 Dec 2016

The article under discussion considers algorithms for performing inference on large or unwieldy datasets by partitioning the data, analyzing each piece separately, and then putting the inferences together. This is an active area of research in computational statistics (see, for example, Tresp, 2000, Ahn, Korattikara, and Welling, 2012, Gershman, Hoffman, and Blei, 2012, Korattikara, Chen, and Welling, 2013, Hoffman et al., 2013, Wang and Blei, 2013, Scott et al., 2013, Wang and Dunson, 2013, Neiswanger et al., 2013, and Wang, 2014).

Before getting to our own ideas in this area, we would like to review the need for such divide-and-conquer algorithms.

"Big data" is sometimes defined as more than can fit into memory at once, or as any dataset that is too large for us to do what we would like with it. For example, from Wikipedia, *"Big data* is a term for data sets that are so large or complex that traditional data processing applications are inadequate to deal with them."

Steve Scott works at Google and sees really really big data. The problems on which we work are much smaller but the concerns he raises in the paper under discussion are relevant in our work too: for us, a survey with 100,000 respondents counts as "big data" in that the models we'd like to fit can run uncomfortably slowly. For example, it might take a couple hours to run a hierarchical regression predicting survey responses given several factors such as age, sex, ethnicity, education, and state. A few hours doesn't sound so bad, but this inhibits our ability to explore data by trying out and perturbing lots of models.

Why do we need to fit so many models to a single dataset? Because survey adjustment is complicated. As you may have heard, the 2016 U.S. presidential election polls were not far off in aggregate—Hillary Clinton had a small but consistent lead in the polls for months, and she won by three million votes—but they failed in several key states, with the key problem being differential nonresponse: some proportion of Republican voters who were not being reached in surveys (Gelman, 2016). To get the correct inferences from a survey, it is necessary to adjust for as many variables as possible so as to be able to reasonably extrapolate from sample to population. The basic idea is to model the survey response y given some large number of predictors X, then use the model to make inferences about the distribution of y given X in the population, and finally to average this over the joint distribution of X among voters. This process involves two challenging tasks: "multilevel regression" of y on X, and "poststratification," which requires inference about that distribution of X, that is, assumptions about voter turnout. On one hand, both these steps will be most effective when conditioning on more variables; that is, throwing more factors into X: adjusting not just for age, sex, etc., but also for additional demographics such as marital status and income and political variables such as party identification and previous votes. On the other hand, adding complexity makes both modeling steps more difficult: more interactions to consider in the regression and more assumptions to make in the postratification. Hence in practice the need to fit and explore many different models, to assess which factors are essential to include and which can be set aside for present purposes.

*Discussion of "Comparing Consensus Monte Carlo Strategies for Distributed Bayesian Computation," for the *Brazilian Journal of Probability and Statistics*. We thank Dani Gamerman for soliciting this article, Steve Scott for helpful comments, and the Office of Naval Research, National Science Foundation, Institute for Education Sciences, and Moore and Sloan Foundations for partial support of this research.

†Department of Statistics, Columbia University, New York

‡Helsinki Institute of Information Technology, Department of Computer Science, Aalto University, Finland.

All these steps become increasingly important in the real world of nonrepresentative samples, where survey response rates are typically less than 10% (Wang et al., 2015). And adding variables to the right hand side of the regression leads to more complicated models—the more predictors we have for any given dataset, the more serious we must be about regularization. When increasing the width of the data and the complexity of the model, we run into computational constraints, even with data sets of only 10^5 observations.

The above is all background, motivating our interest in methods that allow us to fit big models to big data. As Scott discusses, when you can't work with your whole dataset at once, breaking it into pieces is a natural way to go. He writes of "consensus Monte Carlo" but we prefer the term "divide and conquer algorithms" (see, for example, Wang et al., 2015) because we don't see the Monte Carlo aspect as being crucial to the problem. If, for example, each of the separate analyses were being performed using a deterministic algorithm leading to a Laplace or variational approximation, the same issues would arise, of getting good inferences for the separate problems and of combining them into a single inference capturing as much information as possible from the entire dataset.

We have been thinking about divide-and-conquer, or consensus Monte Carlo, for a long time. And, like Steve Scott, we have found the idea both tantalizing and frustrating. Huang and Gelman (2005) describes our first effort in this area, an adaptation of Gibbs sampling for hierarchical regression, splitting the data into K pieces using cluster sampling so that each separate dataset was smaller in width as well as length by approximately a factor of K. The idea was that time required for each separate computation would be reduced by roughly a factor of K^2 so that even in a purely serial implementation the total computation time, putting together the K separate nodes, would be of order $1/K$ compared to the original computation. In practice, however, the algorithm did not work nearly so well. We got reasonable results in our motivating application, but computation time was reduced by only a factor of 1.1, not the factor of 2 or 8 or 32 that we had hoped. The challenge comes in the implementation.

So we agree with Steve Scott about the importance of these ideas and are also with him on their difficulty. In our discussion we would like elaborate on two points that Scott treats only briefly.

Our first concern is the role of the prior distribution. If data are subdivided, each piece of the likelihood contains less information, which will make priors and regularization even more necessary. We do not think it is a good general solution to divide the prior into K parts, as $p(\theta)^{1/K}$ just won't do the job in many cases. Scott refers the problem of improper posteriors but that's really just the least of our worries. Even if the posterior is proper, two problems can arise if it is too weakly regularized: (1) the separate models, being too loosely constrained by prior information, will be more computationally difficult to fit, and (2) even setting aside computational challenges, each of these inferences will be noisy and can be so far apart that it will be more difficult to combine them. For example, a weak prior combined with complete separation in logistic regression will lead to computational problems. With weak priors the separate posteriors may have thick tails, iterative estimation algorithms can be slow to converge, and the resulting estimates become noisy.

Regularization is an increasingly important aspect of model fitting and we can't ignore it here. But if we were to simply keep the full prior $p(\theta)$ for the analysis of each of the data subsets, then it would be counted K times when these pieces are put together, and we would then need to correct for this overcounting of prior information. This is *not* just a problem with Bayesian inference; it will arise with any regularization procedure.

The second challenge we wish to address is the problem, within each subset analysis, of how to use the information from the other $K-1$ subsets of the data. Scott's example of "non-overlapping beta posteriors" illustrates the problem well. When information is not shared between subsets, it is possible that all the computational resources are used far away from the combined posterior, and then the combination of the posteriors can be difficult. Once we follow Scott and accept the use of

approximations to the separate inferences, we can make use of the other $K-1$ approximations to regularize each local inference without need to fit all the data into memory at same time. The idea is to set up an algorithm using $K+1$ processors: one for each of the K subset analyses and one "consensus" processor which passes the K separate approximations back and forth.

As we discuss in yet another unpublished paper (Gelman et al., 2014), this sort of message-passing algorithm can be framed as an expectation propagation algorithm (Minka, 2001, Heskes et al., 2005). The steps go as follows:

> First, with any data-splitting Bayesian algorithm, split the model into $K+1$ pieces corresponding to the prior distribution and K factors of the likelihood: $p(\theta|y) \propto p(\theta)\prod_{k=1}^{K} p(y_k|\theta)$. Here, each y_k is not a single data point but rather one of the K "shards" of data, in Scott's terminology.
>
> We then aim to construct an approximate posterior, $g(\theta) = p(\theta)\prod_{k=1}^{K} g_k(\theta)$, where the factors $g(\theta)$ come from some parametric family such as a multivariate normal distribution in θ.
>
> At this point, the usual divide-and-conquer algorithm would separately fit each factor in the likelihood, $p(y_k|\theta)$, to a corresponding approximation, $g_k(\theta)$. For example, one might run Hamiltonian Monte Carlo on θ, then fit a parametric approximation (such as a mixture of multivariate normals) to these simulation draws to obtain g_k.
>
> As noted above, though, this sort of direct approximation can be problematic, as the $1/K$ part of the likelihood can contain too little information to effectively constrain the parameters. Indeed, the whole point of "big data" is, presumably, that a subset would not be enough. So here is where we borrow an idea from expectation propagation and compute the *cavity distribution*, $g_{-k}(\theta) = g(\theta)/g_k(\theta)$, which represents an approximation of all the information except that from the k-th shard of data. We combine the cavity distribution with the corresponding factor of the likelihood to obtain the *tilted distribution*, $g_{\backslash k}(\theta) = p(y_k|\theta)g_{-k}(\theta)$, on which we then perform inference to obtain an updated site approximation $g_k^{\text{new}}(\theta)$ such that $g_k^{\text{new}}(\theta)g_{-k}(\theta)$ approximates $g_{\backslash k}(\theta)$. This computation should be no problem: as far as this node is concerned, all difficulty comes from the likelihood shard, $p(y_k|\theta)$, with the cavity distribution serving as a prior.
>
> The algorithm then proceeds iteratively, with the K separate nodes performing the inference steps to obtain updates of $g_k(\theta)$ and passing these to the consensus node, which then in turn updates the cavity distributions and sends them back out. The resulting process performs consensus inference (or consensus Monte Carlo if the updates at the K nodes are performed using a simulation algorithm such as Hamiltonian Monte Carlo implemented in Stan), with the cavity distributions from expectation propagation providing regularization and stability.

Message-passing algorithms do pay a computational price for the scheduling and communication time between processors, but we have found in big data cases that a relatively small number of communication iterations is sufficient. It is not necessary to assume that the tilted distributions are close to normal, but to make the propagation of the information easier, we have used use multivariate normal approximations for the combined posterior of the shared parameters. (In a hierarchical model the local parameters depend only on the local data and there is no need to combine inferences for them.) Assuming the number of shared parameters is not increasing with sample size, the combined posterior of the shared parameters should be asymptotically approximately normal from the Bernstein-von Mises theorem.

As with any other consensus algorithm, practical implementation can still be a challenge. For example, we agree with Scott that it can make sense to use normal mixtures to approximate the

cavity distributions, but in that case a combinatorial explosion arises; for example if each g_k is a mixture of 3 components, then g will have 3^K components, and some sort of further approximation or trimming will be needed. More generally, we echo Scott's interest in continuing research in this area.

References

Ahn, S., Korattikara, A., and Welling, M. (2012). Bayesian posterior sampling via stochastic gradient Fisher scoring. In *Proceedings of the 29th International Conference on Machine Learning*.

Chib, S. (1995). Marginal likelihood from the Gibbs output. *Journal of the American Statistical Association* **90**, 1313–1321.

Gelman, A. (2016). Explanations for that shocking 2% shift. Statistical Modeling, Causal Inference, and Social Science blog, 9 Nov. `http://andrewgelman.com/2016/11/09/explanations-shocking-2-shift/`

Gelman, A., Vehtari, A., Jylanki, P., Robert, C., Chopin, N., and Cunningham, J. P. (2014). Expectation propagation as a way of life. `https://arxiv.org/abs/1412.4869`

Gershman, S., Hoffman, M., and Blei, D. (2012). Nonparametric variational inference. In *Proceedings of the 29th International Conference on Machine Learning*.

Heskes, T., Opper, M., Wiegerinck, W., Winther, O., and Zoeter, O. (2005). Approximate inference techniques with expectation constraints. *Journal of Statistical Mechanics: Theory and Experiment*, P11015.

Hoffman, M., Blei, D. M., Wang, C., and Paisley, J. (2013). Stochastic variational inference. *Journal of Machine Learning Research*, 14(1), 1303-1347.

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

Minka, T. (2001). Expectation propagation for approximate Bayesian inference. In *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*, ed. J. Breese and D. Koller, 362–369.

Neiswanger, W., Wang, C., and Xing, E. (2013). Asymptotically exact, embarrassingly parallel MCMC. `arXiv:1311.4780`.

Scott, S. L. (2017). Comparing consensus Monte Carlo strategies for distributed Bayesian computation. *Brazilian Journal of Probability and Statistics*.

Scott, S. L., Blocker, A. W., Bonassi, F. V., Chipman, H. A., George, E. I., and McCulloch, R. E. (2013). Bayes and big data: The consensus Monte Carlo algorithm. In *Bayes 250*. `http://research.google.com/pubs/pub41849.html`

Stan Development Team (2016). Stan modeling language: User's gude and reference manual, version 2.13.1. `http://mc-stan.org/`

Tresp, V. (2000). A Bayesian committee machine. *Neural Computation* **12**, 2719–2741.

Wang, C., and Blei, D. M. (2013). Variational inference in nonconjugate models. *Journal of Machine Learning Research* **14**, 899–925.

Wang, C., Chen, M. H., Schifano, E., Wu, J., and Yan, J. (2015). Statistical methods and computing for big data. `https://arxiv.org/pdf/1502.07989.pdf`.

Wang, W., Rothschild, D., Goel, S., and Gelman, A. (2015). Forecasting elections with non-representative polls. *International Journal of Forecasting* **31**, 980–991.