Bagging During Markov Chain Monte Carlo for Smoother Predictions

Herbert K. H. Lee
University of California, Santa Cruz

Abstract: Making good predictions from noisy data is a challenging problem. Methods to improve the robustness of predictions include bagging and Bayesian shrinkage approaches. These methods can be gainfully combined by doing bootstrap resampling during Markov chain Monte Carlo (MCMC). The result is smoother predictions that are less affected by outliers or particularly noisy data points. An example is provided in predicting from a neural network model.

Key Words: Bayesian statistics, bootstrap, neural network, robust inference

1. Introduction

The field of statistics arose because we are constantly faced with noisy data. Most statistical methods do a reasonable job of smoothing out some of the noise, resulting in better predictions. However, the presence of one or more observations farther from the true trend can influence the fit, with severe outliers causing predictions to become inaccurate. There is a trade off between a reduction in the expected bias (inaccuracy) of prediction, and the variance (uncertainty) of those predictions. The goal is to have accurate, low variance predictions, but ultimately one must balance those two competing goals. The balance becomes yet more difficult in the presence of outliers or highly noisy observations. One approach to obtain smoothing and reduced variance is a shrinkage approach, or equivalently, a Bayesian approach. This paper takes that idea one level higher by combining a Bayesian approach with bagging.

Breiman (1996a) introduced bagging (Bootstrap AGGregatING) to reduce prediction variance without increasing the prediction bias. It works by using bootstrap resampling, fitting the model to each sample, and then averaging the predictions to get the bagged prediction. It can be used with any model, but it produces the most

improvement with unstable models, those for which the predictions are sensitive to small changes in the data, such as neural networks, classification and regression trees, and variable selection routines (Breiman, 1996b). These references also provide a theoretical justification for the procedure.

In this paper, bagging is implemented during Markov chain Monte Carlo (MCMC) to improve the stability of the procedure, leading to smoother predictions. The decrease in predictive variance given by bagging allows better predictions than a simple Bayesian shrinkage approach alone. Bagging naturally fits into the MCMC framework, because MCMC predictions are themselves an average across many samples. Going one step further, we do a bagging resample step at each MCMC iteration.

An alternative approach to combining bagging and a Bayesian approach is through the Bayesian bootstrap, which has also been shown to be effective (Clyde and Lee, 2001). This sort of Bayesian bagging can also be implemented losslessly in an online setting (Lee and Clyde, 2004).

The next section describes the elements of the methodology, and the following section puts them together. The subsequent section provides an example where the model being fit is a neural network. Finally, the paper concludes with some additional remarks.

2. Background

The mechanics of bagging are based on the standard bootstrap (Breiman, 1996a). The bootstrap works by resampling the original data $\mathbf{X}=(x_1,\ldots,x_n)$ to produce M

pseudo-datasets ($\mathbf{X}_1,\ldots,\mathbf{X}_M$). Each of these pseudo-datasets is generated by drawing n samples with equal probability *with replacement* from the original dataset. Thus each pseudo-dataset is of the same size as the original, and each element of a pseudo-dataset is a member of the original dataset. However, elements may be repeated, and elements of the original dataset may be omitted from a particular pseudo-dataset. To estimate any functional of interest, one simply evaluates that functional over each of the M pseudo-datasets and averages the result to get the bootstrap estimate. In bagging, the functional of interest is the prediction from a model. For example, to get a prediction for y(x) from a particular model, one would fit that model on each of the M pseudo-datasets, make a prediction at x for each of those fits, and then average those M predictions to get the bagged prediction.

Within the Bayesian framework, inference is typically difficult or impossible to do in closed form, so Markov chain Monte Carlo (MCMC) is usually employed. This technique simulates from the posterior distribution, providing a Monte Carlo sample from the posterior so that quantities of interest, such as predictions, can be estimated. The idea is to create a Markov chain with stationary distribution equal to the posterior distribution. It turns out that such a chain can be created without knowing the normalizing constant of the posterior distribution, which is the typical stumbling block that makes analytical inference impossible. Typically each parameter is cycled through, and is updated conditional on the current values of all of the other parameters. Gibbs steps involve direct sampling from this conditional distribution, and Metropolis-Hastings steps use a form of rejection sampling for the conditional distribution. The set of parameters is updated sequentially, and then the full cycle is repeated in each iteration. For more details, many texts are now available for MCMC, such as (Gelman et al, 2003).

The particular model that we use in this paper is a neural network. A neural network is in the family of infinitely parametric models, and thus a type of non-parametric regression. In practice a finite number of nodes are used, and this represents an approximation to the full nonparametric model. One way to view a neural network is as a location-scale mixture of logistic functions $h(x) = 1/(1+\exp(-g'x))$ where g is a vector of coefficients and x is the input vector of explanatory variables (Lee, 2004). A prediction is then given as $b_0 + \sum b_i h_i(x)$, where the b_i are the mixing (linear combination) coefficients. The set of location-scale logistic functions forms a basis over the space of continuous functions as well as over the space of square-integrable functions. Thus a neural network can approximate arbitrarily closely most standard functions. In the machine learning literature, the logistic basis functions h(x) are typically referred to as hidden nodes. Neural networks make sense in the context of this paper because they are both highly flexible models, as well as "unstable" in the lexicon of Breiman, and thus amenable to more improvement with bagging (Breiman, 1996b). Within the Bayesian paradigm, a neural network can be fit using MCMC. With any of the standard priors, the mixing coefficients and the overall variance can be fit with Gibbs steps, and location-scale parameters inside the logistic basis functions can be fit with Metropolis-Hasting steps. The chain is initialized with arbitrary values; the particular values are unimportant as long as the chain is run long enough to reach equilibrium, a better choice of starting values allows faster convergence. More details are available in Lee (2004) and the references therein.

3. Bagging During MCMC

Combining bagging and MCMC is straightforward, because both are based on sampling. The idea is to resample the dataset, with replacement, after each MCMC iteration (each full cycle through the parameters). Thus at the end, estimates are taken as an average simultaneously over the resampled data and over the samples from the posterior distribution. This joint process is an approximation of the result, since the Markov chain is no longer strictly stationary as the underlying dataset is changing at each iteration. But the overall result is actually more stable than standard MCMC, so it is a very useful approximation. It may make sense to start the bootstrap resampling only after initial convergence of the chain has been achieved, as that is when samples are obtained for the final Monte Carlo estimate.

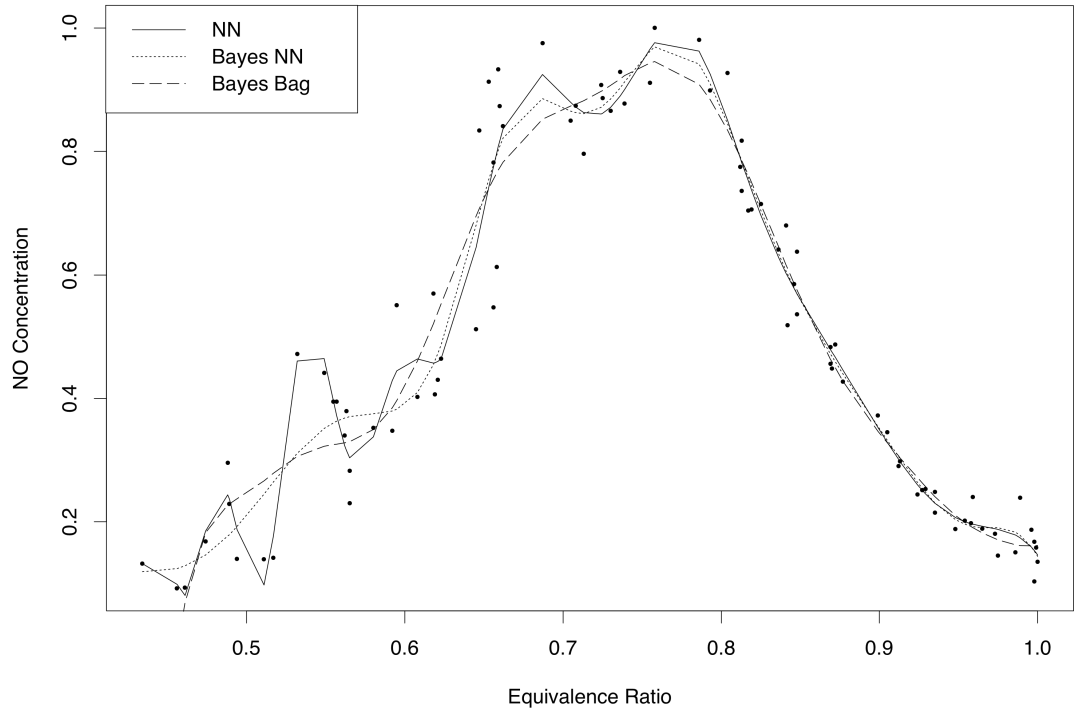
The algorithm can be summarized as:

1. Start MCMC with initial values (the same as with ordinary MCMC)
2. Cycle through all of the parameters, updating each from conditional distributions
3. Repeat step (2) until initial convergence to the stationary distribution
4. Resample the dataset using a bootstrap sample
5. Cycle through all of the parameters, updating each from conditional distributions
6. Repeats steps (4) and (5) until the desired number of Monte Carlo samples are obtained
7. Produce prediction estimates from the Monte Carlo sample

4. Example

The method is now demonstrated on a real example, the ethanol data set of Brinkman (1981). The goal is to understand nitrogen oxide emissions (both NO and NO_2) when ethanol is burned as a fuel in an automobile engine. The explanatory

variable is the equivalence ratio, a measure of the level of ethanol in the fuel, when the engine is operated. The data have been rescaled to the unit square. From the figure below, it can be seen that the data are quite noisy. A 12-node neural network is fit, with the predictions shown with the solid line. The neural network fit results in picking up a lot of local behavior, which may be over-reacting to a few particularly noisy observations, such as around 0.54 and 0.77. Using a Bayesian model results in some shrinkage, which is shown with the dotted line. Finally combining the Bayesian approach with bagging during MCMC is shown with the dashed line. This combination results in the smoothest predictions, which still capture the shape of the curve really well, but don't give as much influence to particularly noisy observations.



5. Conclusions

Bagging and MCMC are natural partners, since both are sampling techniques. They can be easily combined, and the results give more robust predictions, which are less influenced by very noisy or outlying data points. One could consider incorporating the Bayesian bootstrap instead of the standard bootstrap, and then the combined procedure could be seen as estimation of a particular posterior using MCMC and bagging, where the posterior also includes unknown weights on the data points.

References

Breiman, L. (1996a) Bagging Predictors, *Machine Learning*, **26**(2), 123–140.

Breiman, L. (1996b) Heuristics of Instability and Stabilization in Model Selection, *The Annals of Statistics*, **24**, 2350–2383.

Brinkman, N. D., (1981) Ethanol Fuel—A Single-Cylinder Engine Study of Efficiency and Exhaust Emissions, *SAE Transactions*, **90**, 1410-1424.

Clyde, M. A. and Lee, H. K. H. (2001) Bagging and the Bayesian Bootstrap. In *Artificial Intelligence and Statistics 2001*, T. Richardson and T. Jaakkola eds., 169-174.

Gelman, A., Carlin, J. B., Stern, H. S. and Rubin, D. B. (2003) *Bayesian Data Analysis*, 2nd edition. Boca Raton: Chapman & Hall.

Lee, H. K. H. (2004) *Bayesian Nonparametrics via Neural Networks*. Philadelphia: ASA-SIAM.

Lee, H. K. H. and Clyde, M. A. (2004) Lossless Online Bayesian Bagging, *Journal of Machine Learning Research*, **5**, 143-151.