The Canadian Journal of Statistics

Vol. , 2011, Pages 1

La revue canadienne de statistique

Weighted Bayesian Bootstrap for Scalable Bayes

Blinded-A, Blinded-B and Blinded-C

Key words and phrases: Bayesian; Bootstrap; MCMC; Weighted Bootstrap; ABC; Trend Filtering; Deep Learning; TensorFlow; Regularization.

MSC 2010: Primary 62F40; secondary 6262F15

Abstract: We develop a weighted Bayesian Bootstrap (WBB) for machine learning and statistics. WBB provides uncertainty quantification by sampling from a high dimensional posterior distribution. WBB is computationally fast and scalable using only off-the-shelf optimization software. We provide regularity conditions which apply to a range of machine learning and statistical models. We illustrate our methodology in regularized regression, trend filtering and deep learning. Finally, we conclude with directions for future research.

1. INTRODUCTION

Weighted Bayesian Bootstrap (WBB) is a simulation-based algorithm for assessing uncertainty in machine learning and statistics. Uncertainty quantification (UQ) is an active area of research, particularly in high-dimensional inference problems (e.g., Wang, 2018). Whilst there are computationally fast and scalable algorithms for training models in a wide variety of contexts, uncertainty assessments are still required, as are methods to compute these assessments. Bayesian analysis offers a general solution, but developing computationally fast scalable algorithms for sampling a posterior distribution is a notoriously hard problem. WBB makes a contribution to this literature by showing how off-the-shelf optimization algorithms, such as convex optimization or stochastic gradient descent (SGD), can be adapted to provide uncertainty assessments.

For relatively simple statistical models, the weighted likelihood bootstrap (WLB) method provides approximate posterior sampling through repeated optimization of a randomly weighted likelihood function (Newton & Raftery, 1994). The proposed WBB extends the WLB to a broad class of contemporary statistical models by leveraging advances in optimization methodology. Essentially, the WLB used optimization of certain randomized objective functions to enable approximate marginalization (*i.e.*, integration) required in Bayesian analysis. The same idea – optimize a randomized objective function to achieve posterior sampling – is at the heart of the proposed WBB method, though some changes to the

WLB procedure are required to carry out this program for the models considered. Theoretical support for the WLB approximation is based on connections between posterior variation and curvature of log-likelihood revealed through repeated optimization of randomly weighted likelihoods. By contrast, the proposed WBB calculates a series of randomized posterior modes rather than randomized likelihood maximizers. A key rationale for this proposal is that high dimensional posterior modes are now readily computable, thanks to systems such as `TensorFlow` (Abadi et al., 2015) and `Keras` (Chollet et al., 2015) that deploy stochastic gradient descent (SGD) and convex optimization methods for large-scale problems, such as on neural network architectures used in deep learning (LeCun, Bengio & Hinton, 2015). By linking random weighting with modern-day optimization, we expose a simple scheme for approximate uncertainty quantification in a wide class of statistical models.

Quantifying uncertainty is typically unavailable in a purely regularization optimization method. We contend that UQ is available directly by repeated optimization of randomized objective functions, using the same computational tools that produce the primary estimate, rather than through Markov chain Monte Carlo, variational methods, approximate Bayesian computation, or other techniques. Thus, uncertainty assessments are provided at little extra computational cost over the original training computations. A further benefit is that it is straightforward to add a regularization path across hyper-parameters, which is usually difficult to compute in traditional Bayesian sensitivity analysis. We use predictive cross-validation techniques in this regard.

The rest of the paper is outlined as follows. Section 2 develops our weighted Bayesian Bootstrap (WBB) algorithm. Section 3 provides applications to high dimensional sparse regression, trend filtering and deep learning. WBB can also be applied to Bayesian tree models (Taddy et al., 2015). Finally, Section 4 concludes with directions for future research. Areas for future study include bootstrap filters in state-space models (Gordon, Salmond & Smith, 1993) and comparison with the resampling-sampling perspective to sequential Bayesian inference (Lopes, Polson & Carvalho, 2012), etc.

2. WEIGHTED BAYESIAN BOOTSTRAP

2.1. Set up

We work with a broad class of statistical models for data structures involving outcomes and covariates. Examples considered in Section 3 include high-dimensional regression, trend-filtering, and deep learning. Let $y = (y_1, y_2, \cdots, y_n)$ be an n-vector of outcomes and let θ be a d-dimensional parameter of interest. Covariate data may be organized in an $n \times d$ matrix A whose rows are the design points (or "features") a_i^T where we index observations by i and parameters by j. A large number of estimation/training problems can be expressed in the form

$$\underset{\theta \in \mathcal{R}^d}{\text{minimize}} \quad \mathcal{L}(\theta) := l(y|\theta) + \lambda\phi(\theta)\,, \tag{1}$$

where $l(y|\theta) = \sum_{i=1}^{n} l_i(y_i|\theta)$ is a measure of fit (or "empirical risk function") depending on θ and y and implicitly on A. The penalty function, or regularization term, $\lambda\phi(\theta)$, may encode soft or hard constraints, and is controlled by a hyper-parameter, $\lambda > 0$, whose values index an entire path of solutions. The penalty function effects a favorable bias-variance estimation tradeoff and provides extensive modeling flexibility (Wellner and Zhang, 2012). To accommodate contemporary applications we allow that $\phi(\theta)$ may have points in its domain where it fails to be differentiable (e.g., L^1 norm). If we treat the data, y, as arising from a probabilistic model parameterized by θ, then the likelihood function $p(y|\theta)$ yields the model-associated measure of fit $l(y|\theta) = -\log p(y|\theta)$. The maximimum likelihood estimator (MLE) is $\hat{\theta} := \text{argmax}_\theta\, p(y|\theta)$, though of course this usually differs from the solution to (1): $\theta^* := \arg\min \{l(y|\theta) + \lambda\phi(\theta)\}]$. We recall a key duality between regularization and Bayesian analysis.

2.2. Bayesian Regularization Duality

From the Bayesian perspective, the measure of fit, $l(y|\theta) = -\log p(y|\theta)$, and the penalty function, $\lambda\phi(\theta)$, correspond to the negative logarithms of the likelihood and prior distribution in the model

$$
\begin{gathered}
p(y|\theta) \propto \exp\{-l(y|\theta)\}\ , \quad p(\theta) \propto \exp\{-\lambda\phi(\theta)\} \\
p(\theta|y) \propto \exp\{-(l(y|\theta) + \lambda\phi(\theta))\}.
\end{gathered}
\tag{2}
$$

This posterior $p(\theta|y)$ is often a proper distribution over $\mathcal{R}^d$, even if the prior $p(\theta)$ is not proper. The well-known equivalence between regularization and Bayesian methods is seen, for example, in regression with a Gaussian regession model subject to a penalty such as an L^2-norm (ridge) Gaussian prior or L^1-norm (LASSO) double exponential prior. By this duality, the posterior mode, or maximum a posteriori (MAP) estimate, is θ^*, a solution to (1). See Gribonval and Machart (2013) for a nuanced view of the connection between (1) and (2) in Gaussian regression models.

2.3. Optimization

Advances in optimization methodology provide efficient algorithms to compute $\theta^* = \arg\min \mathcal{L}(\theta)$ for a wide range of loss and penalty functions. Theory is well developed in the case of convex objective functions (e.g., Bertsekas *et al.* 2003; Boyd & Vandenberghe 2004). For example if loss l is convex and differentiable in θ and penalty ϕ is convex, then a necessary and sufficient condition for θ^* to minimize $l(y|\theta) + \lambda\phi(\theta)$ is

$$
0 \in \partial\left\{l(y|\theta^*) + \lambda\phi(\theta^*)\right\} = \nabla l(y|\theta^*) + \lambda\partial\phi(\theta^*) \tag{3}
$$

where ∂ is the subdifferential operator (the set of subgradients of the objective), in this case the sum of a point and a set. Though not a formula for θ^*, such as given by the normal equations in linear regression, (3) usefully guides algorithms that aim to solve θ^*. For example, under separability conditions on the penalty function, coordinate descent algorithms effectively solve for θ^*; see Wright (2015), or Hastie, Tibshirani, and Wainwright (2015, chap. 5) for a statistical perspective. The optimization literature also characterizes θ^* as the fixed point of a proximal operator $\mathrm{prox}_{\gamma\mathcal{L}}(\theta) = \arg\min_z\{\mathcal{L}(z) - \frac{1}{2\gamma}||z-\theta||_2^2\}$, which opens the door to powerful MM algorithms and related schemes; see Lange (2016, chap. 5) and Polson & Scott (2015a, 2015b). Beyond convexity, the guarantees are weaker (e.g., local not global minima) and the algorithms are many (e.g., Nocedal and Wright, 2006). Gradient descent or stochastic gradient descent (SGD) are effective in many cases, owing to parameter dimensionality and structure of the gradients. The appendix develops SGD for one example.

Advances in applied optimization provide effective software tools for data analysis. For example, the R package `glmnet` deploys coordinate descent for loss functions arising from generalized linear models and LASSO or elastic net penalties (Friedman *et al.*. 2010). To solve the generalized LASSO problem, the R package `genlasso` deploys a dual path algorithm (Arnold & Tibshirani, 2014). A variety of general purpose optimization tools for statistics are compiled at the optimization view at CRAN (`http://cran.r-project.org`).

For machine learning, the `TensorFlow` system has greatly simplified gradient descent, SGD, and related algorithms for many applications (Abadi *et al.* 2016).

2.4. WBB Algorithm

We now define the weighted Bayesian Bootstrap (WBB). Recalling the original objective function (1), we form the randomized objective

$$\mathcal{L}_{\mathbf{w}}(\theta) = \left\{ \sum_{i=1}^{n} w_i l_i(y_i|\theta) \right\} + w_0 \lambda \phi(\theta) \tag{4}$$

where entries of $\mathbf{w} = (w_0, w_1, \cdots, w_n)$ are independent and identically distributed (i.i.d.) standard exponentially distributed random weights, generated by the analyst and independently from the data y. Equivalently, $w_i = \log(1/u_i)$ where u_i's are i.i.d. $\text{Uniform}(0, 1)$. Associated with any vector $\mathbf{w}$ is the solution, $\theta^*_{\mathbf{w}} = \arg\min \mathcal{L}_{\mathbf{w}}(\theta)$. Our basic conjecture is that the conditional distribution of $\theta^*_{\mathbf{w}}$ – the distribution induced by $\mathbf{w}$ with the data fixed – approximates the Bayesian posterior (2): for sets B in the parameter space,

$$\Pr\left(\theta^*_{\mathbf{w}} \in B|y\right) \approx \int_B p(\theta|y)\, d\theta. \tag{5}$$

Section 2.5 provides an asymptotic argument in support of (5), and we investigate the approximation numerically in a few examples in Section 3. Assuming this conjecture is true, we have immediately a straightforward, optimization-based algorithm for approximate posterior sampling:

Algorithm 1 Weighted Bayesian Bootstrap

Input:

data: $\mathcal{D} = (y, A)$

model structure: $\mathcal{M} = (\{l_i\}, \lambda, \phi)$

number of draws: T

Output: T parameter samples $\{\theta^{*,t}\}$

Function $\mathbf{WBB}(\mathcal{D}, \mathcal{M}, T)$:

for all $t = 1$ to T **do**

Realize: $(u_0, u_1, \cdots, u_n) \sim_{i.i.d.}$ Uniform$(0, 1)$

Construct: $w_i \leftarrow \log(1/u_i), \quad \forall i, \quad \mathbf{w} = (w_0, w_1, \cdots, w_n)$

Compute: $\theta^{*,t} \leftarrow \arg\min \mathcal{L}_{\mathbf{w}}(\theta)$

end for

When optimization on the original problem (1) is fast and scalable, so too is the WBB. Next we consider the approximation (5) from an asymptotic perspective.

** on lambda ** See Appendix and Polson & Sokolov (2017) for further discussion. To choose the amount of regularization λ, we can use the marginal likelihood $m_\lambda(y)$, estimated by bridge sampling (Gelman & Meng, 1998) or simply using predictive cross-validation. **

2.5. WBB Properties

Conditions under which the target posterior distribution (2) is approximately Gaussian are well established (e.g., Johnson, 1970; Kleijn & van der Vaart, 2012). For example, when data form a random sample from fixed distribution $p(y_i|\theta_0)$ that resides in a sufficiently regular model, and when the prior is smooth and positive around $\theta_0 \in \mathcal{R}^d$, we have

$$\theta|\, y \sim_{\text{approx}} N_d\left(\theta_n^*, J_n^{-1}(\theta_n^*)\right), \tag{6}$$

where, including sample size n as an explicit subscript, we have posterior mode $\theta_n^* = \arg\min \mathcal{L}_n(\theta)$, and where $J_n(\theta_n^*) = nj(\theta_n^*)$ is the Fisher information matrix evaluted at θ_n^*. Here $j(\theta)$ is the information per sample, and $y = (y_1, \cdots, y_n)$ denotes data. Centering on the posterior mode, rather than the MLE, improves accuracy in many cases (?citation?).

As to the WBB distribution, consider a one-term Taylor expansion of $\nabla\mathcal{L}_{\mathbf{w},n}(\theta)$ about the posterior mode θ_n^*, which is allowable for sufficiently smooth loss and penalty terms:

$$\nabla\mathcal{L}_{\mathbf{w},n}(\theta) = \nabla\mathcal{L}_{\mathbf{w},n}(\theta_n^*) + \nabla^2\mathcal{L}_{\mathbf{w},n}(\theta_n^*)(\theta - \theta_n^*) + R_n \tag{7}$$

where R_n is an error term and ∇ and ∇^2 record the gradient vector and matrix of second partial derivatives, respectively of the weighted objective function. Evaluating this expansion at $\theta_{\mathbf{w},n}^* = \arg\min \mathcal{L}_{\mathbf{w},n}(\theta)$ zeros out the left hand side of (7), and leads to:

$$\sqrt{n}\left(\theta^*_{\mathbf{w},n} - \theta^*_n\right) = -\left(\frac{1}{n}\nabla^2 \mathcal{L}_{\mathbf{w},n}(\theta^*_n)\right)^{-1}\left(\frac{1}{\sqrt{n}}\nabla \mathcal{L}_{\mathbf{w},n}(\theta^*_n)\right) + \tilde{R}_n \qquad (8)$$

where $\tilde{R}_n$ is another error term. Following Newton and Raftery (1994), we recognize that the $\mathbf{w}$-induced variation in (8), conditional upon the data, causes the matrix factor to be approximately the inverse information $[J_n(\theta^*_n)]^{-1}$, the score-like second factor to be approximately mean-zero Gaussian with covariance equal to $J_n(\theta^*_n)$, and the error $\tilde{R}_n$ to be negligible. Thus, compared to the target posterior variation (6), we have WBB variation:

$$\theta^*_{\mathbf{w},n}\Big|\, y \sim_{\text{approx}} N_d\left(\theta^*_n, J_n^{-1}(\theta^*_n)\right). \qquad (9)$$

In a relatively narrow asymptotic sense, therefore, the WBB procedure is approximating the target posterior distribution, because both are approximately Gaussian with the same mean and covariance. Details of the asymptotic analysis follow the WLB case presented in Newton (1991) and Newton and Raftery (1994), and differ only slightly in our use of the posterior mode θ^*_n in place of the maximum likelihood estimator $\hat{\theta}_n$, and also in our incorporation of weight w_0 on the penalty term of the objective function. At this level of first-order asymptotic analysis, neither of these features affects the limiting conditional Gaussian distribution of $\theta^*_{\mathbf{w},n}$.

Nick, you wrote $*J_n = nj$ as ‘observed information‘, but if it’s the curvature of the observed log likelihood then I guess it’s not nj. Do you mean Fisher information?

* throughout we use exponential weights...scaling not NB...in fact maybe expo not important (cite Barbe and Bertaile), except through connection to Bayes and the Dirichlet...one curious fact is that the expo gives uniform on the simplex...

We note that rescaling in (4) has no effect on solutions (**??**), and so it is equivalent in the construction to use normalized weights $\tilde{\mathbf{w}}$ that sum to unity. Such $\tilde{\mathbf{w}}$ are uniformly distributed over the $(n+1)$-dimensional unit simplex, and thus correspond to a specific Dirichlet distribution. The use of Exponential/Dirichlet weights is motivated from considerations that are both inferential, related to the original Bayesian bootstrap (Rubin, 1981), and also numerical, owing to benefits of smoothly varying weights. Other weight distributions may also be effective (Barbe and Bertail, 2012).

3. APPLICATIONS

Consider now a number of scenarios to assess when WBB corresponds to a full Bayesian posterior distribution.

3.1. Lasso

First, a simple univariate normal means problem with a lasso prior where

$$y|\theta \sim N(\theta, 1^2), \quad \theta \sim Laplace(0, 1/\lambda)$$

Given the i.i.d. exponential weights w_0 and w_1, the weighted posterior mode $\theta^*_{\mathbf{w}}$ is

$$\theta^*_{\mathbf{w}} = \arg\min_{\theta \in \Theta} \left\{ \frac{w_1}{2}(y-\theta)^2 + \lambda w_0 |\theta| \right\}.$$

This is sufficiently simple for an exact WBB solution in terms of soft thresholding:

$$\theta^*_{\mathbf{w}} = \begin{cases} y - \lambda w_0/w_1 & \text{if } y > \lambda w_0/w_1, \\ y + \lambda w_0/w_1 & \text{if } y < -\lambda w_0/w_1, \\ 0 & \text{if } |y| \leq \lambda w_0/w_1. \end{cases}$$

The WBB mean $E_{\mathbf{w}}(\theta^*_{\mathbf{w}}|y)$ is approximated by the sample mean of $\{\theta^{*,t}_{\mathbf{w}}\}_{t=1}^T$. On the other hand, Mitchell (1994) gives the expression for the posterior mean,

$$\begin{aligned} E(\theta|y) &= \frac{\int_{-\infty}^{\infty} \theta \exp\left\{-(y-\theta)^2/2 - \lambda|\theta|\right\} d\theta}{\int_{-\infty}^{\infty} \exp\left\{-(y-\theta)^2/2 - \lambda|\theta|\right\} d\theta} \\ &= \frac{F(y)}{F(y)+F(-y)}(y+\lambda) + \frac{F(-y)}{F(y)+F(-y)}(y-\lambda) \\ &= y + \frac{F(y)-F(-y)}{F(y)+F(-y)}\lambda \end{aligned}$$

where $F(y) = \exp(y)\Phi(-y-\lambda)$ and $\Phi(\cdot)$ is the c.d.f. of standard normal distribution. We plot the WBB mean versus the exact posterior mean in Figure (1). Interestingly, WBB algorithm gives sparser posterior means.

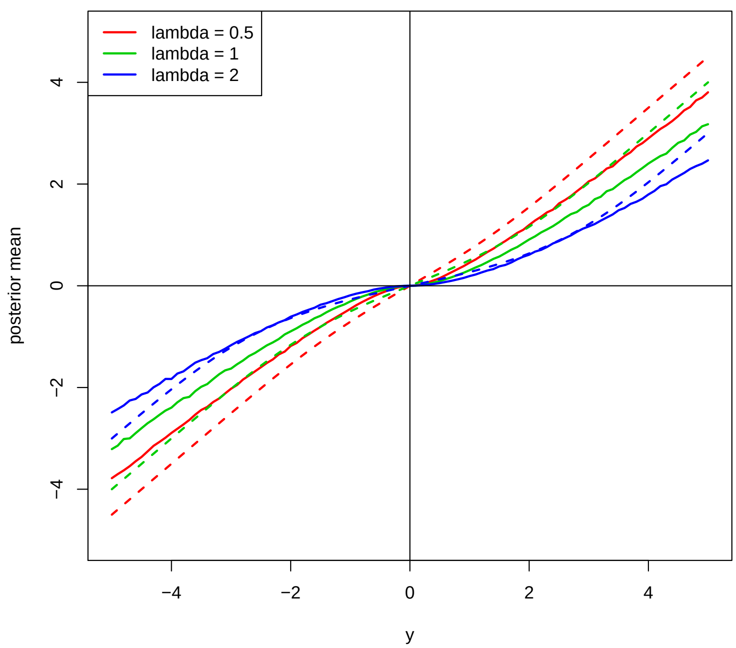

FIGURE 1: Normal means model with lasso prior: WBB mean $E_{\mathbf{w}}(\theta^*_{\mathbf{w}}|y)$ (in solid lines) versus exact posterior mean $E(\theta|y)$ (in dashed lines).

3.2. Diabetes Data

To illustrate our methodology, we use weighted Bayesian Bootstrap (WBB) on the classic diabetes dataset (Efron et al., 2004). The measurements for 442 diabetes patients are obtained ($n = 442$), with 10 baseline variables ($p = 10$), such as age, sex, body mass index, average blood pressure, and six blood serum measurements.

The likelihood function is given by

$$l(y|\beta) = \prod_{i=1}^{n} p(y_i|\beta)$$

where

$$p(y_i|\beta) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left\{-\frac{1}{2\sigma^2}(y_i - x_i'\beta)^2\right\}.$$

We draw 1000 sets of weights $\mathbf{w} = \{w_i\}_{i=1}^{n+1}$ where w_i's are i.i.d. exponentials. For each weight set, the weighted Bayesian estimate $\beta^*_{\mathbf{w}}$ is calculated using Equation (10) via the regularization method in the package `glmnet`.

$$\hat{\beta}_{\mathbf{w}} := \arg\min_{\beta} \sum_{i=1}^{n} w_i(y_i - x_i'\beta)^2 + \lambda w_0 \sum_{j=1}^{p} |\beta_j| \, . \tag{10}$$

The regularization factor λ is chosen by cross-validation with unweighted likelihood. The weighted Bayesian Bootstrap is also performed with fixed prior, namely, w_0 is set to be 1 for all bootstrap samples. (30) analyze the same dataset using the Bayesian Bridge estimator and suggest MCMC sampling from the posterior.

To compare our WBB results we also run the Bayesian bridge estimation. Here the Bayesian setting we use is

$$p(\beta, \sigma^2) = p(\beta|\sigma^2)p(\sigma^2), \text{ where } p(\sigma^2) \propto 1/\sigma^2.$$

The prior on β, with suitable normalization constant C_α, is given by

$$p(\beta) = C_\alpha \exp(-\sum_{j=1}^{p} |\beta_j/\tau|^\alpha).$$

The hyper-parameter is drawn as $\nu = \tau^{-\alpha} \sim \Gamma(2, 2)$, where $\alpha = 1/2$.

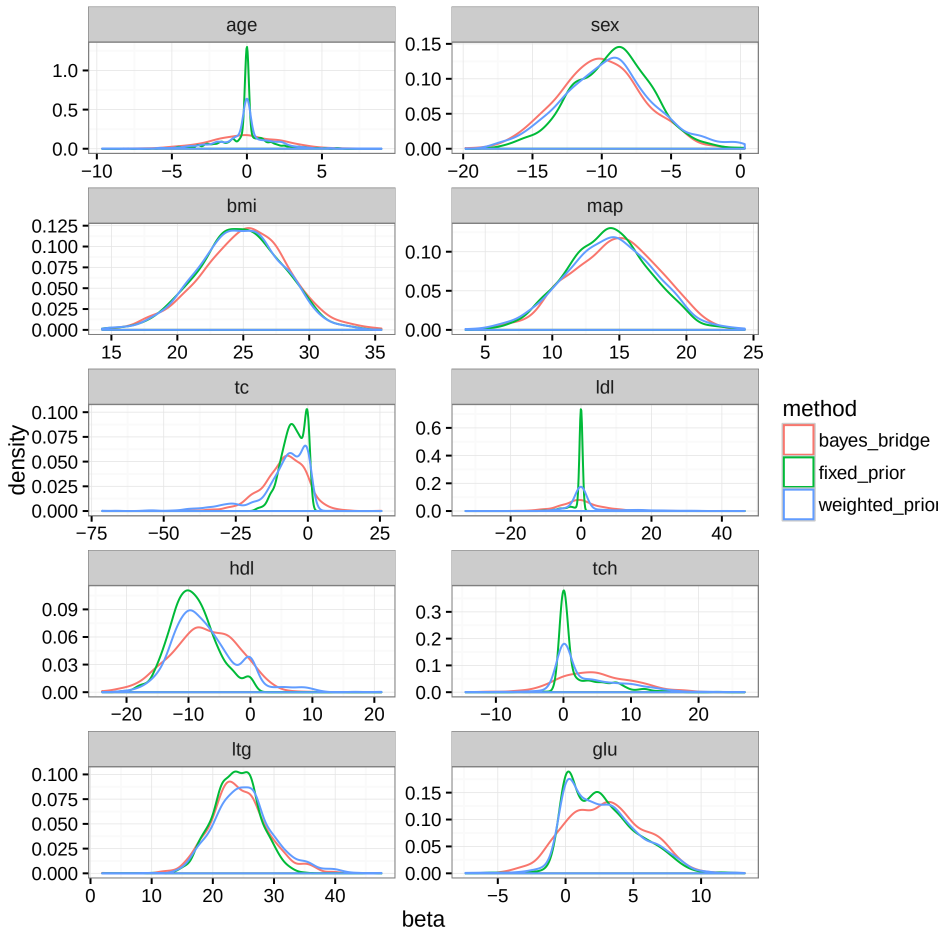

FIGURE 2: Diabetes example: the weighted Bayesian Bootstrap (with fixed prior and weighted prior) and Bayesian Bridge are used to draw from the marginal posteriors for β_j's, j = 1,2,...10.

Figure (2) shows the results of all these three methods (the weighted Bayesian Bootstrap with fixed prior/weighted prior and the Bayesian Bridge). Marginal posteriors for β_j's are presented. One notable feature is that the weighted Bayesian Bootstrap tends to introduce more sparsity than Bayesian Bridge does. For example, the weighted Bayesian Bootstrap posteriors of `age`, `ldl` and `tch` have higher spikes located around 0, compared with the Bayesian Bridge ones. For `tc`, `hdl`, `tch` and `glu`, multi-modes in the marginal posteriors are observed. In general, the posteriors with fixed priors are more concentrated than those with randomly weighted priors. This difference is naturally attributed to variation in the weight assigned to the log-prior penalty term.

3.3. Trend Filtering

The generalized lasso solves the optimization problem:

$$
\begin{aligned}
\beta^* &= \arg\min_{\beta} \ \{l(y|\beta) + \lambda\phi(\beta)\} && (11)\\
&= \arg\min_{\beta} \frac{1}{2}\|y - X\beta\|_2^2 + \lambda\|D\beta\|_1 && (12)
\end{aligned}
$$

where $l(y|\beta) = \frac{1}{2}\|y - X\beta\|_2^2$ is the negative log-likelihood. $D \in \mathcal{R}^{m\times p}$ is a penalty matrix and $\lambda\phi(\beta) = \lambda\|D\beta\|_1$ is the negative log-prior or regularization penalty. There are fast path algorithms for solving this problem (see `genlasso` package).

As a subproblem, polynomial trend filtering (Tibshirani, 2014; Polson & Scott, 2015a) is recently introduced for piece-wise polynomial curve-fitting, where the knots and the parameters are chosen adaptively. Intuitively, the trend-filtering estimator is similar to an adaptive spline model: it penalizes the discrete derivative of order k, resulting in piecewise polynomials of higher degree for larger k.

Specifically, $X = I_p$ in the trend filtering setting and the data $y = (y_1, ..., y_p)$ are assumed to be meaningfully ordered from 1 to p. The penalty matrix is specially designed by the discrete $(k+1)$-th order derivative,

$$D^{(1)} = \begin{bmatrix} -1 & 1 & 0 \dots & 0 & 0 \\ 0 & -1 & 1 \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \\ 0 & 0 & 0 \dots & -1 & 1 \end{bmatrix}_{(p-1)\times p}$$

and $D^{(k+1)} = D^{(1)} D^{(k)}$ for $k = 1, 2, 3....$ For example, the log-prior in linear trend filtering is explicitly written as $\lambda \sum_{i=1}^{p-2} |\beta_{i+2} - 2\beta_{i+1} + \beta_i|$. For a general order $k > 1$,

$$\|D^{(k+1)}\beta\|_1 = \sum_{i=1}^{p-k-1} \Big| \sum_{j=i}^{i+k+1} (-1)^{(j-i)} \binom{k+1}{j-i} \beta_j \Big|.$$

WBB solves the following generalized lasso problem in each draw:

$$\begin{aligned} \beta^*_{\mathbf{w}} &= \arg\min_{\beta} \frac{1}{2} \sum_{i=1}^{p} w_i (y_i - \beta_i)^2 + \lambda w_0 \|D^{(k)}\beta\|_1 \\ &= \arg\min_{\beta} \frac{1}{2} \|Wy - W\beta\|_2^2 + \lambda \|D^{(k)}\beta\|_1 \\ &= W^{-1} \arg\min_{\tilde{\beta}} \frac{1}{2} \|\tilde{y}_{\mathbf{w}} - \tilde{\beta}_{\mathbf{w}}\|_2^2 + \lambda \|\tilde{D}^{(k)}_{\mathbf{w}} \tilde{\beta}_{\mathbf{w}}\|_1 \end{aligned}$$

where

$$W = \mathrm{diag}\left(\sqrt{w_1}/\sqrt{w_0}, \cdots, \sqrt{w_p}/\sqrt{w_0}\right)$$

and

$$\tilde{y}_{\mathbf{w}} = Wy,\ \tilde{\beta}_{\mathbf{w}} = W\beta,\ \tilde{D}^{(k)}_{\mathbf{w}} = D^{(k)} W^{-1}.$$

To illustrate our method, we simulate data y_i from a Fourier series regression

$$y_i = \sin\left(\frac{4\pi}{500} i\right) \exp\left(\frac{3}{500} i\right) + \epsilon_i$$

for $i = 1, 2, \cdots, n = 500$, where $\epsilon_i \sim N(0, 2^2)$ are i.i.d. Gaussian deviates. The cubic trend filtering result is given in Figure (3).

For each i, the WBB gives a group of estimates $\{\beta^*_{\mathbf{w}}(i)\}_{j=1}^T$ where T is the total number of draws. The standard deviation of these weighted solutions consitutes a posterior standard deviation, or essentially a standard error for the estimator $\hat{\beta}_i$.

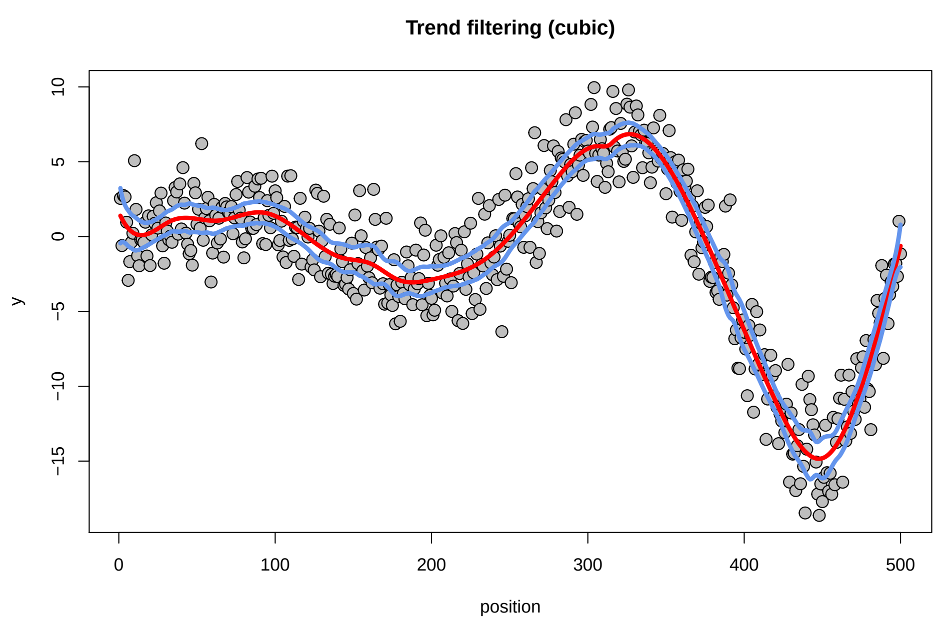

FIGURE 3: Cubic trend filtering: the red line is $\hat{\beta}_i$ for $i = 1, 2, ...500$; the blue line is $\hat{\beta}_i \pm 2 * se$ where the standard errors are computed by WBB. $\lambda = 1000$.

3.4. Deep Learning: MNIST Example

Deep learning is a form of machine learning that uses hierarchical abstract layers of latent variables to perform pattern matching and prediction. Polson & Sokolov (2017) take a Bayesian probabilistic perspective and provide a number of insights into more efficient algorithms for optimization and hyper-parameter tuning. The general goal is to find a predictor of an output y given a high dimensional input x. For a classification problem, $y \in \{1, 2, ..., K\}$ is a discrete variable and can be coded as a K-dimensional 0-1 vector. The model is as follows. Let $z^{(l)}$ denote the l-th layer, and so $x = z^{(0)}$. The final output is the response y, which can be numeric or categorical. A deep prediction rule is then

$$\begin{aligned}
z^{(1)} &= f^{(1)}\left(W^{(0)}x + b^{(0)}\right),\\
z^{(2)} &= f^{(2)}\left(W^{(1)}z^{(1)} + b^{(1)}\right),\\
&\dots\\
z^{(L)} &= f^{(L)}\left(W^{(L-1)}z^{(L-1)} + b^{(L-1)}\right),\\
\hat{y}(x) &= z^{(L)}.
\end{aligned}$$

Here, $W^{(l)}$ are weight matrices, and $b^{(l)}$ are threshold or activation levels. $f^{(l)}$ is the activation function. Probabilistically, the output y in a classification problem is generated by a probability model

$$p(y|x, W, b) \propto \exp\{-l(y|x, W, b)\}$$

where $l(y|x, W, b) = \sum_{i=1}^{n} l_i(y_i|x_i, W, b)$ is the negative cross-entropy,

$$l_i(y_i|x_i, W, b) = l_i(y_i, \hat{y}(x_i)) = \sum_{k=1}^{K} y_{ik} \log \hat{y}_k(x_i)$$

where y_{ik} is 0 or 1 and $K = 10$. Adding the negative log-prior $\lambda\phi(W, b)$, the objective function (negative log-posterior) to be minimized by stochastic gradient descent is

$$\mathcal{L}_\lambda(y, \hat{y}) = \sum_{i=1}^{n} l_i(y_i, \hat{y}(x_i)) + \lambda\phi(W, b).$$

Accordingly, with each draw of weights $\mathbf{w}$, WBB provides the estimates $(W^*_{\mathbf{w}}, b^*_{\mathbf{w}})$ by solving the following optimization problem.

$$(W^*_{\mathbf{w}}, b^*_{\mathbf{w}}) = \arg\min_{W,b} \sum_{i=1}^{n} w_i l_i(y_i|x_i, W, b) + \lambda w_0 \phi(W, b)$$

We take the classic MNIST example (LeCun & Cortes, 2010) to illustrate the application of WBB in deep learning. The MNIST database of handwritten digits, available from Yann LeCun's website, has 60,000 training examples and 10,000 test examples. Here the high-dimensional x is a normalized and centered fixed-size (28×28) image and the output $\hat{y}$ is a 10-dimensional vector, where i-th coordinate corresponds to the probability of that image being the i-th digit.

For simplicity, we build a 2-layer neural network with layer sizes 128 and 64 respectively. Therefore, the dimensions of parameters are

$$W^{(0)} \in \mathcal{R}^{128\times 784},\ b^{(0)} \in \mathcal{R}^{128},$$

$$W^{(1)} \in \mathcal{R}^{64\times 128},\ b^{(1)} \in \mathcal{R}^{64},$$

$$W^{(2)} \in \mathcal{R}^{10\times 64},\ b^{(0)} \in \mathcal{R}^{10}.$$

The activation function $f^{(i)}$ is ReLU, $f(x) = \max\{0, x\}$, and the negative log-prior is specified as

$$\lambda\phi(W, b) = \lambda \sum_{l=0}^{2} \|W^{(l)}\|_2^2$$

where $\lambda = 10^{-4}$.

Figure (4) shows the posterior distribution of the classification accuracy in the test dataset. We see that the test accuracies are centered around 0.75 and the posterior distribution is left-skewed. Furthermore, the accuracy is higher than 0.35 in 99% of the cases. The 95% interval is [0.407, 0.893].

4. DISCUSSION

Weighted Bayesian Bootstrap (WBB) provides a computationally attractive solution to scalable Bayesian inference (Minsker et al., 2014; Welling & Teh, 2011) whilst accounting for parameter uncertainty by drawing samples from a weighted posterior distribution. WBB can also be used in conjunction with proximal methods (Parikh & Boyd, 2013; Polson & Scott, 2015b) to provide sparsity in high

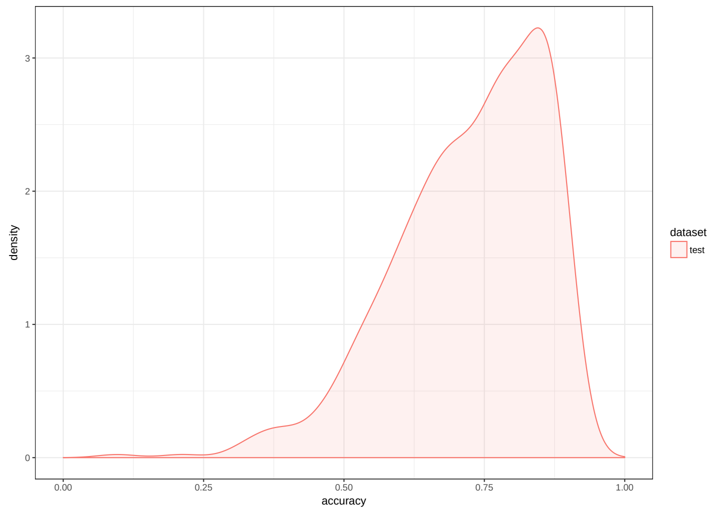

FIGURE 4: Posterior distribution of the classification accuracy. $n = 500, \lambda = 10^{-4}$.

dimensional statistical problems. With a similar ease of computation, WBB provides an alternative to ABC methods (Beaumont, 2009) and Variational Bayes (VB) methods. A fruitful area for future research is the comparison of approximate Bayesian computation with simulated Bayesian Bootstrap inference.

A general class of natural exponential family models can be expressed in terms of the Bregman divergence of the dual of the cumulant transform. Let ϕ be the conjugate Legendre transform of ψ. Hence $\psi(\theta) = \sup_\mu \left(\mu^\top \theta - \phi(\mu)\right)$. Then we can write

$$
\begin{aligned}
p_\psi(y|\theta) &= \exp\left(y^\top \theta - \psi(\theta) - h_\psi(y)\right) \\
&= \exp\left\{\inf_\mu \left((y-\mu)^\top \theta - \phi(\mu)\right) - h_\psi(y)\right\} \\
&= \exp\left(-D_\phi(y, \mu(\theta)) - h_\phi(y)\right)
\end{aligned}
$$

where the infimum is attained at $\mu(\theta) = \phi'(\theta)$ is the mean of the exponential family distribution. We rewrite $h_\psi(y)$ is terms of the correction term and $h_\phi(y)$. Here there is a duality as D_ϕ can be interpreted as a Bregman divergence.

For a wide range of non-smooth objective functions/statistical models, recent regularization methods provide fast, scalable algorithms for calculating estimates of the form (**??**), which can also be viewed as the posterior mode. Therefore as λ varies we obtain a full regularization path as a form of prior sensitivity analysis.

(33) and (29) considered scenarios where posterior modes can be used as posterior means from augmented probability models. Moreover, in their original foundation of the Weighted Likelihood Bootstrap (WLB), (25) introduced the concept of the implicit prior. Clearly this is an avenue for future research.

Extending WBB asymptotics presents some exciting research opportunities. The argument in Section 2.5 relies on smoothness in both the sampling model and prior, and it retains fixed parameter dimension as n increases. Theoretical guarantees remain unavailable for relatively large parameter dimension or for non-smooth penalty functions. Fortunately, groundwork has been done, for example by Van Der Pas *et al.* (2014), Narisety & He (2014) and others on the asymptotic behaviour of the posterior distribution, and by Knight and Fu (2000) and others on sampling theory of optimization-based estimators,

BIBLIOGRAPHY

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Man?e, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Vi?egas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., & Zheng, X. (2015) TensorFlow: Large-scale machine learning on heterogeneous systems. *Software available from tensorflow.org*.

Arnold, T. B., & Tibshirani, R. J. (2014). genlasso: Path algorithm for generalized lasso problems. *R package version 1.3*.

Barbe, P. & Bertail, P. (2012). *The weighted bootstrap* Vol. 98. Springer Science & Business Media.

Beaumont, M. A., Cornuet, J.M., Marin,J.M. & Robert, C. P. (2009). Adaptive approximate bayesian computation. *Biometrika*, 96(4), 983–990.

Bertail, P. & Lo, A. Y. (1991). On Johnson's asymptotic expansion for a posterior distribution. *Centre de Recherche en Economie et Statistique*.

Bertsekas, D.P., Nedi, A. & Ozdaglar, A.E. (2003). *Convex analysis and optimization*. Athena Scientific.

Boyd, S. & Vandenberghe, L. (2004) *Convex optimization*. Cambridge university press.

Chollet, François & others. (2015) Keras *https://keras.io*

Daniels, H. & Young, G. (1991). Saddlepoint approximation for the studentized mean, with an application to the bootstrap. *Biometrika*,78(1), 169–179.

Efron, B. (1981). Nonparametric standard errors and confidence intervals. *Canadian Journal of Statistics*, 9(2), 139–158.

Efron, B. (2012). Bayesian inference and the parametric bootstrap. *The Annals of Applied Statistics*, 6(4), 1971.

Efron, B., Hastie, T., Johnstone, I. & Tibshirani, R. J. (2004). Diabetes dataset. Available from R package *lars*.

Friedman, J., Hastie, T. & Tibshirani, R. J. (2010). Regularization Paths for Generalized Linear Models via Coordinate Descent. *Journal of Statistical Software*, 33(1),1–22.

Fu, W.J., 1998. Penalized regressions: the bridge versus the lasso. *Journal of computational and graphical statistics*, 7(3), pp.397-416.

Gelman, A. & Meng, X.L. (1998). Simulating normalizing constants: From importance sampling to bridge sampling to path sampling. *Statistical Science*, 163–185.

Gordon, N. J., Salmond, D. J. & Smith, A. F. (1993). Novel approach to nonlinear/non- gaussian Bayesian state estimation. *In IEE Proceedings F (Radar and Signal Processing)*, 140(2) 107–113, IET Digital Library.

Gramacy, R. B. & Polson, N. G. (2012). Simulation-based regularized logistic regression. *Bayesian Analysis* , 7(3), 567–590.

Gribonval, R. & Machart, P. (2013). Reconciling ”priors” &”priors” without prejudice?. In *Advances in Neural Information Processing Systems* pp. 2193-2201.

Hans, C. (2009). Bayesian lasso regression. *Biometrika*, 96(4), 835–845.

Hastie, T., Tibshirani, R. & Wainwright, M., 2015. *Statistical learning with sparsity: the lasso and generalizations*. CRC press.

Johnson, R. A. (1970). Asymptotic expansions associated with posterior distributions. *The Annals of Mathematical Statistics*, 41(3), 851–864.

Lange, K. (2016). *MM Optimization Algorithms*. Vol. 147, SIAM.

LeCun, Y., Bengio, Y. & Hinton, G. (2015). Deep learning. *Nature*, 521(7553), 436.

LeCun, Y. & Cortes, C. (2010). MNIST handwritten digit database. Retrieved from *http://yann.lecun.com/exdb/mnist/*

Lopes, H. F., Polson, N. G., & Carvalho, C. M. (2012). Bayesian statistics with a smile: a resampling-sampling perspective. *Brazilian Journal of Probability and Statistics*, 26(4) 358–371.

Minsker, S., Srivastava, S., Lin, L. & Dunson, D. (2014). Scalable and robust bayesian inference via the median posterior. *International Conference on Machine Learning*, 1656–1664.

Mitchell, A. F. (1994). A note on posterior moments for a normal mean with double- exponential prior. *Journal of the Royal Statistical Society. Series B (Methodological)*, 605–610.

Newton, M. A. (1991). The weighted likelihood bootstrap and an algorithm for prepivoting. Ph. D. thesis, Department of Statistics, University of Washington, Seattle.

Newton, M. A. & Raftery,, 1994 A. E. (1994). Approximate Bayesian inference with the weighted likelihood bootstrap. *Journal of the Royal Statistical Society. Series B (Methodological)*, 3–48.

Nocedal, J. & Wright, S.J. (2006). *Numerical optimization*, 2nd edition. Springer series in Operations Research.

Parikh, N. & Boyd, S. (2014). Proximal algorithms. *Foundations and Trends in Optimization*, 1(3), 127–239.

Polson, N. G. & Scott , J. G.(2015). Mixtures, envelopes and hierarchical duality. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 78(4), 701–727.

Polson, N. G., Scott, J. G., & Willard, B. T. (2015). Proximal algorithms in statistics and machine learning. *Statistical Science*, 30(4), 559–581.

Polson, N. G., Scott, J. G., & Windle, J. (2014). The Bayesian bridge. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 76(4), 713–733.

Polson, N. G. & Sokolov, V. (2017). Deep learning: A bayesian perspective. *Bayesian Analysis*,12(4), 1275–1304.

Rubin, D. B. (1981). The Bayesian bootstrap. *The Annals of Statistics*, 9(1), 130–134.

Strawderman, R. L., Wells, M. T., & Schifano, E. D. (2013). Hierarchical Bayes, maxi- mum a posteriori estimators, and minimax concave penalized likelihood estimation. *Electronic Journal of Statistics*, 7, 973–990.

Taddy, M., Chen, C.S., Yu, J. & Wyle M. (2015). Bayesian and empirical Bayesian forests. *arXiv preprint arXiv:1502.02312*.

Tibshirani, R. J. (2014). Adaptive piecewise polynomial estimation via trend filtering. *The Annals of Statistics*, 42(1), 285–323.

Tibshirani, R.J. (2017). Dykstra's Algorithm, ADMM, and Coordinate Descent: Connections, Insights, and Extensions. In *Advances in Neural Information Processing Systems*, pp. 517-528.

Van Der Pas, S., Kleijn, B. & Van Der Vaart, A. (2014). The horseshoe estimator: Posterior concentration around nearly black vectors. *Electronic Journal of Statistics*, 8(2), 2585–2618.

Wang, Y. & Swiler, L. (2018). Special Issue on Uncertainty Quantification in Multiscale System Design and Simulation. *ASCE-ASME Journal of Risk and Uncertainty in Engineering Systems, Part B: Mechanical Engineering*, 4(1), 010301.

Welling, M. & Teh, Y. W. (2011). Bayesian learning via stochastic gradient Langevin dynamics. *In Proceedings of the 28th International Conference on Machine Learning* (ICML- 11), 681–688.

Wellner, J. & Zhang, T. (2012). Introduction to the special issue on sparsity and regularization methods. *Statistical Science*, 27, 447-449.

West, M. (1991). *Modelling with mixtures*. Institute of Statistics and Decision Sciences, Duke University

Wright S.J. (2015). Coordinate descent algorithms. *Mathematical Programming*, 151(1):3-34.

APPENDIX

Stochastic gradient descent (SGD) method or its variation is typically used to find the deep learning model weights by minimizing the penalized loss function, $\sum_{i=1}^{n} w_i l_i(y_i;\theta) + \lambda w_p \phi(\theta)$. The method minimizes the function by taking a negative step along an estimate g^k of the gradient $\nabla \left[\sum_{i=1}^{n} w_i l_i(y_i;\theta^k) + \lambda w_p \phi(\theta^k)\right]$ at iteration k. The approximate gradient is estimated by calculating

$$g^k = \frac{n}{b_k} \sum_{i \in E_k} w_i \nabla l_i(y_i;\theta^k) + \lambda w_p \frac{n}{b_k} \nabla \phi(\theta^k)$$

Where $E_k \subset \{1,\ldots,n\}$ and $b_k = |E_k|$ is the number of elements in E_k. When $b_k > 1$ the algorithm is called batch SGD and simply SGD otherwise. A usual strategy to choose subset E is to go cyclically and pick consecutive elements of $\{1,\ldots,T\}$, $E_{k+1} = [E_k \mod n] + 1$. The approximated direction g^k is calculated using a chain rule (aka back-propagation) for deep learning. It is an unbiased estimator. Thus, at each iteration, the SGD updates the solution

$$\theta^{k+1} = \theta^k - t_k g^k$$

For deep learning applications the step size t_k (a.k.a learning rate) is usually kept constant or some simple step size reduction strategy is used, $t_k = a\exp(-kt)$.

Appropriate learning rates or the hyperparameters of reduction schedule are usually found empirically from numerical experiments and observations of the loss function progression.