Distributed Bayesian Learning with Stochastic Natural-gradient Expectation Propagation and the Posterior Server

Yee Whye Teh, Leonard Hasenclever, Thibaut Lienart, Sebastian Vollmer, Stefan Webb
University of Oxford, United Kingdom

Balaji Lakshminarayanan, Charles Blundell
Google DeepMind

January 1, 2016

Abstract

This paper makes two contributions to Bayesian machine learning algorithms. Firstly, we propose stochastic natural gradient expectation propagation (SNEP), a novel alternative to expectation propagation (EP), a popular variational inference algorithm. SNEP is a black box variational algorithm, in that it does not require any simplifying assumptions on the distribution of interest, beyond the existence of some Monte Carlo sampler for estimating the moments of the EP tilted distributions. Further, as opposed to EP which has no guarantee of convergence, SNEP can be shown to be convergent, even when using Monte Carlo moment estimates. Secondly, we propose a novel architecture for distributed Bayesian learning which we call the posterior server. The posterior server allows scalable and robust Bayesian learning in cases where a dataset is stored in a distributed manner across a cluster, with each compute node containing a disjoint subset of data. An independent Markov chain Monte Carlo (MCMC) sampler is run on each compute node, with direct access only to the local data subset, but which targets an approximation to the global posterior distribution given all data across the whole cluster. This is achieved by using a distributed asynchronous implementation of SNEP to pass messages across the cluster. We demonstrate SNEP and the posterior server on distributed Bayesian learning of logistic regression and neural networks.

Keywords: Distributed Bayesian Learning, Expectation Propagation, Stochastic Approximation, Natural Gradient, Markov chain Monte Carlo, Posterior Server, Large Scale Learning, Deep Learning.

1 Introduction

Algorithms and systems for enabling machine learning from large scale datasets are becoming increasingly important in the era of Big Data. This has driven many developments, including various forms of stochastic gradient descent, parallel and distributed learning systems, use of GPUs, sketching, random Fourier features, divide-and-conquer methods, as well as various approximation schemes. These large scale machine learning systems have in turn driven significant advances across many data-oriented sciences and technologies, ranging from the biological sciences, neuroscience, social sciences, signal processing, speech processing, natural language processing, computer vision etc.

In this paper we will consider methods for large scale *Bayesian* machine learning. As opposed to the more common empirical risk minimisation or maximum likelihood approaches, where learning is phrased as finding a set of parameters optimal with respect to a dataset and to a loss or likelihood function, Bayesian machine learning rests upon probabilistic models which capture the dependencies among all observed and unobserved variables, and where learning is phrased as computing the posterior distribution over unobserved variables (including both latent variables and model parameters) given the observed data. The Bayesian framework can more fully capture the uncertainty in learnt parameters and prevent overfitting, so in principle can allow for the use of more complex and larger scale models. However, Bayesian approaches are generally more computationally intensive than optimisation-based ones, and have to date not led to methods which are as scalable.

For complex models, exact computation of the posterior distribution is intractable and approximate schemes such as variational inference (VI) (Wainwright and Jordan, 2008), Markov chain Monte Carlo (MCMC) (Gilks et al., 1996) and sequential Monte Carlo (Doucet et al., 2001) are needed. Scalable methods in both traditions include: stochastic variational inference (Hoffman et al., 2013, Mnih and Gregor, 2014, Rezende et al., 2014) which apply minibatch stochastic gradient descent (Robbins and Monro, 1951) to optimise the variational objective function, stochastic gradient MCMC (Welling and Teh, 2011, Patterson and Teh, 2013, Ding et al., 2014, Teh et al., 2015, Leimkuhler and Shang, 2015, Ma et al., 2015) which uses minibatch stochastic gradients within MCMC, austerity MCMC (Korattikara et al., 2014, Bardenet et al., 2014) which uses data subsampling to reduce computational cost of Metropolis-Hastings acceptance steps, and embarrassingly parallel MCMC (Huang and Gelman, 2005, Scott et al., 2013, Wang and Dunson, 2013, Neiswanger et al., 2014) which distribute data across a cluster, runs independent MCMC samplers on each worker and combines samples across the cluster only at the end to reduce network communication costs. In addition, standard learning schemes have also been successfully deployed in large scale settings, a particularly successful one being expectation propagation (EP) (Minka, 2001) in the TrueSkill XBox player rating and matching system (Herbrich et al., 2007).

Our work builds upon prior work on using EP for performing distributed Bayesian learning (Xu et al., 2014, Gelman et al., 2014). In this framework, a dataset is partitioned into disjoint subsets with each subset stored on a worker node in a cluster. Learning is performed at each worker based on the data subset there using MCMC sampling. As opposed to embarrassingly parallel MCMC methods which only communicate the samples to the master at the end of learning, EP is used to communicate messages (infrequently) across the cluster. These messages coordinate the samplers such that the target distributions of all samplers (which coincidentally are the tilted distributions in EP) on all workers share certain moments, e.g. means and variances, hence the name sampling via moment sharing (SMS) coined by (Xu et al., 2014). At convergence, it can also be shown that the target distributions of the samplers also share moments with the EP approximation to the global posterior distribution given all data, hence the target distributions on the workers can themselves be treated as approximations to the global posterior.

While SMS works well on simpler models like Bayesian logistic regression and spike-and-slab linear regression, we have found that it did not work for more complex, high-dimensional, and non-convex models like Bayesian deep neural networks. This is due to the non-convergence of EP, particularly as the moments of the tilted distributions needed by EP are estimated using MCMC sampling, with estimation noise that further compounds the well-known lack of convergence

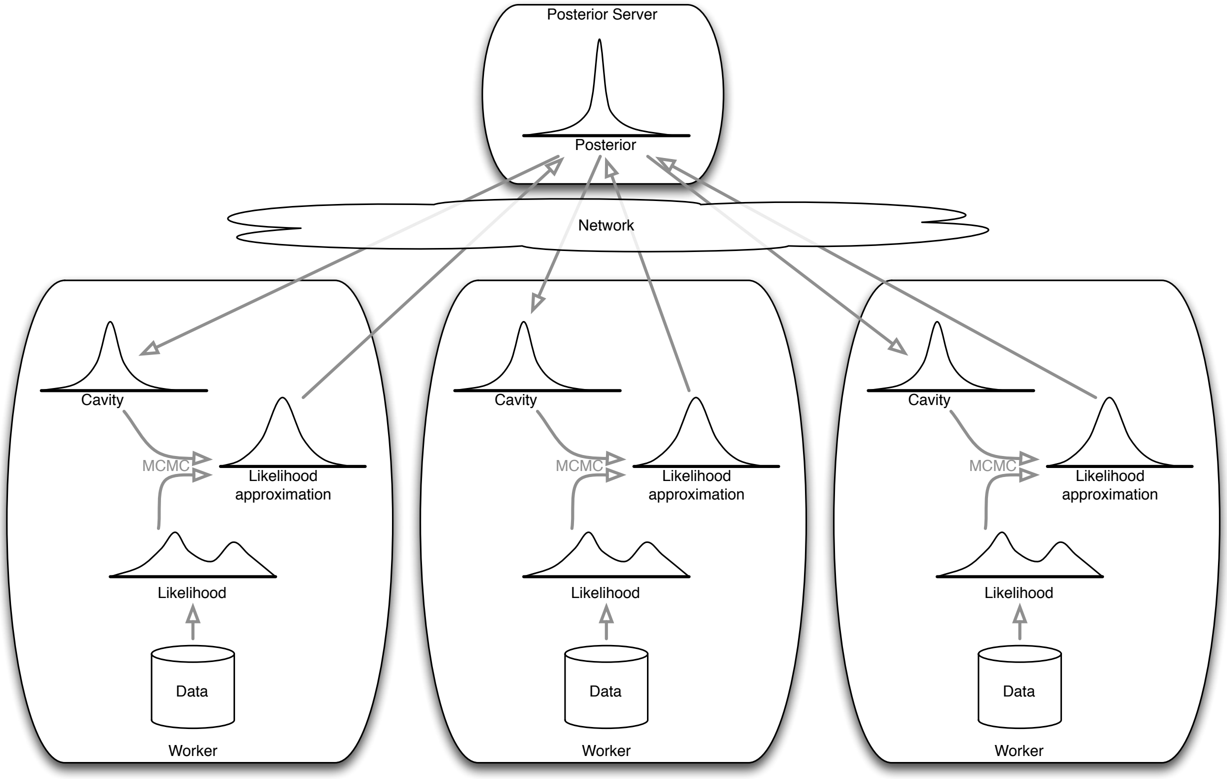

Figure 1: The posterior server.

guarantees for EP, and the fact that extremely long MCMC runs are needed for the samplers to equilibrate due to the complex posterior distribution in these models.

Our first contribution is thus the development of stochastic natural-gradient EP (SNEP), an alternative algorithm to EP which optimises the same EP variational objective function. SNEP is a double-loop algorithm with convergence guarantees. The inner loop is a stochastic natural-gradient descent algorithm which tolerates estimation noise, so that SNEP is convergent even with moments estimated using MCMC samplers. Our derivation of SNEP improves upon the derivation of the convergent EP algorithm of (Heskes and Zoeter, 2002) in that ours works for more general class of models, we make explicit the underlying variational objective function that is being optimised, and ours use a natural-gradient descent algorithm (Amari and Nagaoka, 2001) more tolerant of Monte Carlo noise. SNEP generalises to easily power EP (Minka, 2004).

Building upon the development of SNEP, our second contribution is a distributed Bayesian learning architecture which we call the posterior server. In analogy to the parameter server (Ahmed et al., 2012) which maintains and serves the parameter to a cluster of workers, the posterior server maintains and serves (an approximation to) the posterior distribution. Figure 1 gives a schematic for the steps involved. Each worker has a subset of data, from which we get a likelihood function. It also maintains a tractable approximation of the likelihood and a cavity distribution which is effectively a conditional distribution over the parameters given all data on other workers. An MCMC sampler targets the normalised product of the cavity distribution and the (true) likelihood, and forms a stochastic estimate of the required moments, which is in turn used to update the likelihood approximation using stochastic natural-gradient descent. Each worker communicates with the posterior server asynchronously and in a non-blocking manner, sending the current likelihood approximation and receiving the new cavity distribution. This communication protocol makes more efficient use of computational resources on workers than SMS, which requires either synchronous or blocking asynchronous protocols.

Note that the set-up of the distributed learning problem is that each worker has access to a subset of data, and no worker has access to all data. This situation might occur in situations other than large scale learning. For example, when working with sensitive patient data which cannot be shared directly, we might still want to be able to make use of all available data across multiple sites to improve inference. Typically known as divide-and-conquer or consensus inference (Zhang et al., pted, Zhao et al., 2014, Kleiner et al., 2014, Battey et al., 2015), it is also well-known that this situation is harder than typical distributed learning settings where it is assumed that all data is accessible on all workers (e.g. Dean et al. (2012), Zhang et al. (2015)).

In the following, Section 2 describes our set up of distributed Bayesian learning and reviews the necessary background on exponential families and convex duality. Section 3 formulates EP and power EP within the framework of variational inference, while Section 4 derives SNEP. Section 5 describes the posterior server architecture and Section 6 describes additional techniques we used to make the method work on neural networks. We demonstrate the approach Bayesian logistic regression and Bayesian neural networks in Section 7. Section 8 concludes with a discussion.

2 Problem Set-up and Background

In this section we set-up the problem of distributed Bayesian learning, using the framework of variational inference in exponential families. For an excellent introduction to exponential families and variational inference we refer the interested reader to (Wainwright and Jordan, 2008).

2.1 Exponential Families and Convex Duality

Consider an exponential family described by a d-dimensional sufficient statistics function $s(x)$. A member p_θ of this exponential family is parameterized by a natural parameter $\theta \in \mathbb{R}^d$, and has density (with respect to some base measure, say Lebesgue),

$$p_\theta(x) = \exp\left(\theta^\top s(x) - A(\theta)\right), \tag{1}$$

$$A(\theta) = \log \int \exp\left(\theta^\top s(x)\right) dx. \tag{2}$$

The log partition function $A(\theta)$ is convex and finite on the natural domain of the exponential family,

$$\Theta := \{\theta : A(\theta) < \infty\} \subset \mathbb{R}^d, \tag{3}$$

which is a convex subset of $\mathbb{R}^d$.

Associated with any distribution p and the d-dimensional sufficient statistics function $s(x)$ is a mean parameter,

$$\mu := \mathbb{E}_p[s(x)], \tag{4}$$

where $\mathbb{E}_p$ denotes the expectation operator with respect to p. The set of valid mean parameters $\mathcal{M}$ is a closed convex set, which we refer to as the mean domain,

$$\mathcal{M} = \{\mu : \exists \text{ distribution } p \text{ with } \mu = \mathbb{E}_p[s(x)]\} \subset \mathbb{R}^d \tag{5}$$

Given a natural parameter $\theta \in \Theta$, the exponential family member p_θ is also associated with a mean parameter $\mu = \mathbb{E}_\theta[s(x)]$ (where $\mathbb{E}_\theta$ denotes the expectation with respect to p_θ), which we can write as a function of the natural parameters, $\mu(\theta)$. If the exponential family is minimal[1], then the mapping $\theta \mapsto \nabla A(\theta)$ is one-to-one and onto the interior of $\mathcal{M}$, and maps θ to the mean parameter, $\mu(\theta) = \nabla A(\theta)$. We will assume that our exponential family of interest is minimal.

The convex conjugate of $A(\theta)$ is,

$$A^*(\mu) := \sup_{\theta \in \Theta} \theta^\top \mu - A(\theta). \tag{6}$$

Evaluated at the mean parameter $\mu(\theta)$, the conjugate is the negative entropy of p_θ,

$$A^*(\mu(\theta)) = \mathbb{E}_\theta[\log p_\theta(x)]. \tag{7}$$

Conversely, we have

$$A(\theta) = \sup_{\mu \in \mathcal{M}} \theta^\top \mu - A^*(\mu) \tag{8}$$

and that the natural parameter associated with μ is $\theta = \theta(\mu) = \nabla A^*(\mu)$. It is useful to write down formulae for the KL divergence between two exponential family distributions, parameterized by natural and mean parameter pairs θ, μ and θ', μ' respectively:

$$\begin{aligned}
\mathrm{KL}(p_\theta \| p_{\theta'}) &= \mathbb{E}_\theta[\log p_\theta(x) - \log p_{\theta'}(x)] \\
&= \mathbb{E}_\theta[\theta^\top s(x) - A(\theta) - (\theta')^\top s(x) + A(\theta')] \\
&= \mu^\top(\theta - \theta') - A(\theta) + A(\theta') \\
&= A^*(\mu) + A(\theta') - \mu^\top \theta' \\
&= A^*(\mu) - A^*(\mu') + (\mu' - \mu)^\top \theta'. \qquad (9)
\end{aligned}$$

We will write $\mathrm{KL}(\theta \| \theta'), \mathrm{KL}(\mu \| \theta')$ etc to refer to the same KL divergence between the same two distributions.

As an example, for a diagonal covariance Gaussian of dimension $d/2$, we have

$$\begin{aligned}
p(x) &= \prod_{j=1}^{d/2} \frac{1}{\sqrt{2\pi\sigma_j^2}} \exp\left(-\frac{1}{2\sigma_j^2}(x_j - u_j)^2\right) \\
&= \exp\left(\sum_{j=1}^{d/2} (u_j \sigma_j^{-2})(x_j) + (-\sigma_j^{-2})(\tfrac{1}{2}x_j^2) - \tfrac{1}{2}(u_j^2 \sigma_j^{-2} + \log(2\pi\sigma_j^2))\right) \qquad (10)
\end{aligned}$$

[1] The exponential family is minimal if the d 1-dimensional functions making up the sufficient statistics function $s(x)$ are linearly independent, i.e. $\theta^\top s(x) = 0$ for all x implies $\theta = 0$.

So the sufficient statistics are x_j and $\frac{1}{2}x_j^2$, mean parameters are $\mu_{j1} = u_j$ and $\mu_{j2} = \frac{1}{2}(u_j^2 + \sigma_j^2)$ natural parameters are $\theta_{j1} = u_j\sigma_j^{-2}$ and $\theta_{j2} = -\sigma_j^{-2}$, and

$$A(\theta) = \sum_{j=1}^{d/2} \frac{1}{2}(u_j^2\sigma_j^{-2} + \log(2\pi\sigma_j^2)) \tag{11}$$

$$A^*(\mu) = ?? \tag{12}$$

The conversions between natural and mean parameters are:

$$\begin{aligned} u_j &= \mu_{j1} = -\theta_{j1}\theta_{j2}^{-1} & \theta_{j1} &= \mu_{j1}(2\mu_{j2} - \mu_{j1}^2)^{-1} & \mu_{j1} &= -\theta_{j1}\theta_{j2}^{-1} \\ \sigma_j^2 &= 2(\mu_{j2} - \mu_{j1}^2) = -\theta_{j2}^{-1} & \theta_{j2} &= -(2\mu_{j2} - \mu_{j1}^2)^{-1} & \mu_{j2} &= \frac{1}{2}(\theta_{j1}^2\theta_{j2}^{-2} - \theta_{j2}^{-1}) \end{aligned} \tag{13}$$

We will use a diagonal covariance Gaussian as our exponential family in our experiments, due to the high-dimensionality of the models used.

2.2 Distributed Bayesian Learning

We assume that our model is parameterized by a high-dimensional parameter vector x. Let the prior distribution $p_0(x)$ be a member of a tractable and minimal exponential family distribution, with natural parameter $\theta_0 \in \Theta \subset \mathbb{R}^d$, sufficient statistics function $s(x)$ and log partition function $A(\theta_0)$. Specifically, we will take $p_0(x)$ to be a diagonal covariance Gaussian. We refer to this exponential family as the *base exponential family*.

We assume that our training dataset is spread across a cluster of n compute nodes or workers, with log likelihood $\ell_i(x)$ on compute node $i = 1, \dots, n$. For example, if we let $\{S_i\}$ be a partition of the data indices, each compute node i could store the corresponding subset of the data $D_i = \{y_c\}_{c \in S_i}$, so that the log likelihood $\ell_i(x)$ is a sum over terms, each corresponding to the log density of one data point stored on node i,

$$\ell_i(x) = \sum_{c \in S_i} \log p(y_c|x). \tag{14}$$

The target posterior distribution is then,

$$\tilde{p}(x) := p(x|\{D_i\}_{i=1}^n) \propto p_0(x) \exp\left(\sum_{i=1}^n \ell_i(x)\right). \tag{15}$$

Using neural networks as an example, x corresponds to all learnable weights and biases in a network, $\log p(y_c|x)$ gives the probability of the class of data item c given the corresponding input vector, and the Gaussian prior corresponds to weight decay.

The learning task is then to compute the posterior distribution. For example we may want to draw samples distributed according to $\tilde{p}(x)$, using these to predict on test data by averaging the predictive densities over the samples as a Monte Carlo estimate of the marginal predictive density. Or we may want to estimate the posterior mean or variance of the model parameters x. In the rest of the paper we will aim to obtain posterior samples, means and variances efficiently but approximately.

3 Variational Inference in an Extended Exponential Family

In general, the likelihood functions are intractable and approximations are necessary. In this paper we will formulate the learning task as variational inference in an extended exponential family. In particular, we will consider a class of variational methods known as *power expectation propagation* (power EP) (Minka, 2004).

To start with, we may trivially formulate the target posterior distribution as an *extended exponential family distribution* with sufficient statistics $\tilde{s}(x) := [s(x), \ell_1(x), \dots, \ell_n(x)]$ and natural parameters $\tilde{\theta} := [\theta_0, \mathbf{1}_n]$ where $\mathbf{1}_n$ is a vector of 1's of length n:

$$\tilde{p}(x) = \exp\left(\tilde{\theta}^\top \tilde{s}(x) - \tilde{A}(\tilde{\theta})\right). \tag{16}$$

The extended log partition function $\tilde{A}(\tilde{\theta})$ is (up to a constant) the log marginal probability of the data,

$$\begin{aligned}\tilde{A}(\tilde{\theta}) = \log \int \exp\left(\tilde{\theta}^\top \tilde{s}(x)\right) dx &= \log \int \exp\left(\theta_0^\top s(x) + \sum_{i=1}^n \ell_i(x)\right) dx \\ &= \log \mathbb{E}_{\theta_0}\left[\exp\left(\sum_{i=1}^n \ell_i(x)\right)\right] + A(\theta_0) \\ &= \log p(\{D_i\}_{i=1}^n) + A(\theta_0). \end{aligned} \tag{17}$$

Denoting the convex conjugate by $\tilde{A}^*(\tilde{\mu})$ and the extended mean domain by $\tilde{\mathcal{M}} \subset \mathbb{R}^{d+n}$, the problem of posterior computation can be expressed as the following concave variational maximization problem:

$$\max_{\tilde{\mu} \subset \tilde{\mathcal{M}}} \tilde{\theta}^\top \tilde{\mu} - \tilde{A}^*(\tilde{\mu}). \tag{18}$$

For example, if the prior exponential family is a diagonal covariance Gaussian, then the optimal mean parameter is $\tilde{\mu}^* := [\mu^*, \nu_1^*, \dots \nu_n^*]$, where $\mu^* := \mathbb{E}_{\tilde{\theta}}[s(x)] \in \mathbb{R}^d$ corresponds to the posterior means and variances of the model parameters x, while $\nu_i^* := \mathbb{E}_{\tilde{\theta}}[\ell_i(x)]$ is the posterior expectation of the ith log likelihood $\ell_i(x)$. Hence the extended mean parameters capture the important aspects of the posterior distribution. As expected, this ideal variational problem is intractable and approximations are needed for tractability. The next section describes one such class of approximations.

3.1 Expectation Propagation and Power Expectation Propagation

In this section we will derive a generalization of expectation propagation (EP) (Minka, 2001) called power EP (Minka, 2004). The derivation is a straightforward generalisation of the variational formulation of Wainwright and Jordan (2008) from EP to power EP.

For each worker node i let $\beta_i > 0$ be a given positive real number. Typically, we take $\beta_i = 1$ which corresponds to standard EP, while $\beta_i \to \infty$ corresponds to variational Bayes (Wiegerinck and Heskes, 2003, Minka, 2004). In the formulation of Wainwright and Jordan (2008), EP involves two approximations; both associated with simpler exponential families, which we refer to as *locally extended exponential families* (or sometimes *local exponential families*). For each i, let the ith locally extended exponential family be associated with the sufficient statistics function $s_i(x) := [s(x), \ell_i(x)]$.

Let Θ_i, $\mathcal{M}_i$, A_i, A_i^* be the associated (local) natural domain, mean domain, log partition function and negative entropy respectively. A distribution in this locally extended exponential family with natural parameter $[\theta_i, \eta_i] \in \Theta_i$ has the form

$$p_{\theta,\eta_i} = \exp\left(\theta_i^\top s(x) + \eta_i \ell_i(x) - A_i(\theta, \eta_i)\right), \tag{19}$$

which is a distribution obtained by tilting a tractable distribution with density proportional to $\exp(\theta_i^\top s(x))$ by a single intractable likelihood $\exp(\ell_i(x))$ raised to the power of η_i. This family can be thought of as treating the ith likelihood term exactly, while approximating all other likelihood terms by projecting them onto the tractable base exponential family, the hope being that this family is still tractable while being closer to the true posterior distribution $\tilde{p}(x)$.

For the first approximation, the extended negative entropy $\tilde{A}^*$ is approximated using a tree-like approximation constructed using only the locally extended negative entropies,

$$\tilde{A}^*([\mu, \nu_1, \dots, \nu_n]) \approx A^*(\mu) + \sum_{i=1}^n \beta_i (A_i^*(\mu, \nu_i) - A^*(\mu)). \tag{20}$$

This approximation is related to the Bethe entropy of loopy belief propagation (Yedidia et al., 2001). Secondly, the extended mean domain $\tilde{\mathcal{M}}$ is approximated by a local mean domain,

$$\mathcal{L} := \{[\mu, \nu_1, \dots, \nu_n] : [\mu, \nu_i] \in \mathcal{M}_i \text{ for all } i = 1, \dots, n\}, \tag{21}$$

The local mean domain is an outer bound, $\mathcal{L} \supset \tilde{\mathcal{M}}$. Intuitively, the constraints described by $\tilde{\mathcal{M}}$ are replaced by the weaker constraints described by the local mean domains. We assume that working with the local exponential families in these ways will be more tractable than working with the full extended exponential family.

Let $\mu_0 \in \mathcal{M}$ be a mean parameter in the base exponential family. For the ith local exponential family, we denote a mean parameter by $[\mu_i, \nu_i] \in \mathcal{M}_i$, and require the marginalization constraint $\mu_i = \mu_0$. The power EP variational problem, which is not in general concave, is,

$$\begin{aligned} \max_{\mu_0, [\mu_i, \nu_i]_{i=1}^n} \quad & \theta_0^\top \mu_0 + \sum_{i=1}^n 1 \cdot \nu_i - A^*(\mu_0) - \sum_{i=1}^n \beta_i (A_i^*(\mu_i, \nu_i) - A^*(\mu_i)) \\ \text{subject to} \quad & \mu_0 \in \mathcal{M} \\ & [\mu_i, \nu_i] \in \mathcal{M}_i \quad \text{for } i = 1, \dots, n \\ & \mu_0 = \mu_i \quad \text{for } i = 1, \dots, n \end{aligned} \tag{22}$$

Note in particular that the entropy and mean domain in the original variational problem (18) have been replaced by their respective approximations.

The updates for EP and power EP can be derived as fixed-point equations that solve the variational problem. First we introduce Lagrange multipliers λ_i for the equality constraints $\mu_0 = \mu_i$, so that the above is equivalent to,

$$\begin{aligned} \max_{\mu_0, [\mu_i, \nu_i]_{i=1}^n} \min_{[\lambda_i]_{i=1}^n} \quad & \underbrace{\theta_0^\top \mu_0 - A^*(\mu_0) + \sum_{i=1}^n \left(\nu_i - \lambda_i^\top (\mu_i - \mu_0) - \beta_i (A_i^*(\mu_i, \nu_i) - A^*(\mu_i))\right)}_{=:L(\mu_0, [\mu_i, \nu_i, \lambda_i]_{i=1}^n)} \\ \text{subject to} \quad & \mu_0 \in \mathcal{M} \\ & [\mu_i, \nu_i] \in \mathcal{M}_i \quad \text{for } i = 1, \dots, n \end{aligned} \tag{23}$$

where the domain of λ_i is simply $\mathbb{R}^d$. Let the Lagrangian above be denoted $L(\mu_0, [\mu_i, \nu_i, \lambda_i]_{i=1}^n)$. The Karush-Kuhn-Tucker (KKT) conditions of the above variational problem has to be satisfied at an optimum, and simply involve setting the derivatives with respect to $\mu_0, \mu_i, \nu_i, \lambda_i$ to zero:

$$
\begin{aligned}
\frac{dL}{d\lambda_i} = 0 : && \mu_i &= \mu_0 && \text{(24a)} \\
\frac{dL}{d\mu_0} = 0 : && \theta_0 - \nabla A^*(\mu_0) + \sum_{j=1}^n \lambda_j &= 0 && \\
&& \theta_0 + \sum_{j=1}^n \lambda_j &= \nabla A^*(\mu_0) && \text{(24b)} \\
\frac{dL}{d\mu_i} = 0 : && -\lambda_i - \beta_i \nabla_{\mu_i} A_i^*(\mu_i, \nu_i) + \beta_i \nabla A^*(\mu_i) &= 0 && \\
&& \theta_0 + \sum_{j=1}^n \lambda_j - \beta_i^{-1}\lambda_i &= \nabla_{\mu_i} A_i^*(\mu_i, \nu_i) && \text{(24c)} \\
\frac{dL}{d\nu_i} = 0 : && \beta_i^{-1} &= \nabla_{\nu_i} A_i^*(\mu_i, \nu_i) && \text{(24d)}
\end{aligned}
$$

Equation (24b) shows that an optimal μ_0 has to be the mean parameter corresponding to a (base) exponential family distribution with natural parameter $\theta_0 + \sum_{j=1}^n \lambda_j$. Specifically, the posterior distribution can be approximated as,

$$\tilde{p}(x) \propto p_0(x) \exp\left(\sum_{j=1}^n \ell_j(x)\right) \approx \exp\left(\left(\theta_0 + \sum_{j=1}^n \lambda_j\right)^\top s(x)\right). \tag{25}$$

In other words, we can interpret λ_j as the natural parameter of an exponential family approximation to the likelihood $\exp(\ell_j(x))$.

Further, from (24c) and (24d) above, we see that the optimal $[\mu_i, \nu_i]$ is the mean parameter associated with the local posterior distribution,

$$p_i(x) \propto \exp\left(\left(\theta_0 + \sum_{j=1}^n \lambda_j - \beta_i^{-1}\lambda_i\right)^\top s(x) + \beta_i^{-1}\ell_i(x)\right). \tag{26}$$

For standard EP, where $\beta_i = 1$, we get,

$$p_i(x) \propto \exp\left(\left(\theta_0 + \sum_{j\neq i} \lambda_j\right)^\top s(x) + \ell_i(x)\right). \tag{27}$$

The above local posterior distribution is what is known as the *tilted distribution* in EP, with the term $(\theta_0 + \sum_{j=1}^n \lambda_j - \beta_i^{-1}\lambda_i)^\top s(x)$ corresponding to the *cavity distribution* with (the β_i^{-1}th power of) the exponential family approximation to the ith likelihood removed, and replaced by (the β_i^{-1}th power of) the likelihood factor itself. Each step of EP involves first computing $[\mu_i, \nu_i]$ as the mean parameter of the tilted distribution, then updating λ_i, using (24a) and (24b):

$$\lambda_i^{\text{new}} = \nabla A^*(\mu_i) - \theta_0 - \sum_{j \neq i} \lambda_j. \tag{28}$$

The EP update ensures that the expectation of the sufficient statistics function under the tilted distribution $p_i(x)$ and under its exponential family approximation (with natural parameters $\theta_0 + \sum_{j=1}^n \lambda_j - \beta_i^{-1}\lambda_i + \beta_i^{-1}\lambda_i^{\text{new}}$) match. At convergence, this ensures that the expectations under the tilted distributions and under the approximated posterior distribution (25) match. Note that the (power) EP updates are derived as fixed point equations and have no guarantees of convergence. In practice, if the updates oscillate then damped updates are used instead,

$$\lambda_i^{\text{new}} = \alpha\lambda_i + (1-\alpha)\left(\nabla A^*(\mu_i) - \theta_0 - \sum_{j \neq i} \lambda_j\right). \tag{29}$$

where $\alpha \in [0,1]$ is a damping factor. Damping does not affect the fixed points of EP.

3.2 Computing Mean Parameters

Assuming that the base exponential family is tractable, the above steps of EP involve additions, subtractions, and conversions between mean and natural parameters so are easy to compute. The only difficulty is in the computation of the mean parameters (expectations of the sufficient statistics function) of the tilted distributions, which involve the log likelihoods $\ell_i(x)$. In typical applications of EP to graphical models, these are either obtained analytically or using numerical quadrature.

In more complex and general scenarios, both analytic or quadrature-based methods are ruled out. A successful and common class of methods for calculating expectations under otherwise intractable distributions is Monte Carlo. A number of papers have proposed such an approach, including Barthelmé and Chopin (2011), Heess et al. (2013), Gelman et al. (2014) using importance sampling and Xu et al. (2014), Gelman et al. (2014) using Markov chain Monte Carlo (MCMC). In addition, Heess et al. (2013), Eslami et al. (2014), Jitkrittum et al. (2015) use learning techniques to speed-up the process by directly predicting the natural parameters given properties of the tilted distribution.

We have explored using the sampling via moment sharing (SMS) algorithm (Xu et al., 2014) in the context of distributed Bayesian learning of deep neural networks. Unfortunately, the SMS algorithm did not work even for relatively small neural networks and datasets, partly because of the high dimensionality and partly because the Monte Carlo estimation is inherently stochastic, both of which we found affected the convergence of the EP fixed point equations.

4 Stochastic Natural-gradient Expectation Propagation

In this section we will derive a novel convergent stochastic approximation based alternative to EP, which optimizes the same variational objective but is significantly more tolerant of Monte Carlo noise. Our algorithm is derived using a modified but equivalent variational objective with additional auxiliary variables, and solving the dual problem using a stochastic approximation algorithm (Robbins and Monro, 1951).

4.1 Auxiliary Variational Problem

For each i, we introduce an auxiliary natural parameter vector $\theta'_i \in \Theta$, and introduce a term $-\sum_{i=1}^n \mathrm{KL}(\mu_i \| \theta'_i)$ into the variational objective (23). This results in a lower bound on the original objective and is reminiscent of the EM algorithm (Dempster et al., 1977, Neal and Hinton, 1999) and of typical variational Bayes approximations (Beal, 2003, Wainwright and Jordan, 2008). Plugging the relevant formula for the KL divergence (9) into (23), we have the resulting variational objective

$$\begin{aligned}\max_{\mu_0,[\mu_i,\nu_i,\theta'_i]_{i=1}^n} \quad & \theta_0^\top \mu_0 - A^*(\mu_0) + \sum_{i=1}^n \left(\nu_i - \beta_i \left(A_i^*(\mu_i,\nu_i) - \mu_i^\top \theta'_i + A(\theta'_i)\right)\right) \\ \text{subject to} \quad & \mu_0 \in \mathcal{M} \\ & [\mu_i,\nu_i] \in \mathcal{M}_i \quad \text{for } i = 1,\ldots,n \\ & \theta'_i \in \Theta \quad \text{for } i = 1,\ldots,n \\ & \mu_0 = \mu_i \quad \text{for } i = 1,\ldots,n\end{aligned} \tag{30}$$

Maximizing over θ'_i while keeping the other variables fixed will simply set $\theta'_i = \nabla A^*(\mu_i)$, so that the KL terms vanish and resulting in the original problem (22). Hence the variational problem is equivalent with the same optima as (22).

We consider maximizing over $[\theta'_i]_{i=1}^n$ in an outer loop, and the original parameters $\mu_0, [\mu_i,\nu_i]_{i=1}^n$ in an inner loop. Introducing Lagrange multipliers to enforce the equality constraints again, we have,

$$\begin{aligned}\max_{[\theta'_i]_{i=1}^n} \max_{\mu_0,[\mu_i,\nu_i]_{i=1}^n} \min_{[\lambda_i]_{i=1}^n} & \theta_0^\top \mu_0 - A^*(\mu_0) + \sum_{i=1}^n \left(\nu_i - \lambda_i^\top(\mu_i - \mu_0) - \beta_i \left(A_i^*(\mu_i,\nu_i) - \mu_i^\top \theta'_i + A(\theta'_i)\right)\right) \\ \text{subject to} \quad & \mu_0 \in \mathcal{M} \\ & [\mu_i,\nu_i] \in \mathcal{M}_i \quad \text{for } i = 1,\ldots,n \\ & \theta'_i \in \Theta \quad \text{for } i = 1,\ldots,n\end{aligned} \tag{31}$$

Noticing that the Lagrangian is concave in $\mu_0, [\mu_i,\nu_i]_{i=1}^n$ and that Slater's condition holds, the duality gap is zero and we have

$$\begin{aligned}\max_{[\theta'_i]_{i=1}^n} \min_{[\lambda_i]_{i=1}^n} \max_{\mu_0,[\mu_i,\nu_i]_{i=1}^n} & \theta_0^\top \mu_0 - A^*(\mu_0) + \sum_{i=1}^n \left(\nu_i - \lambda_i^\top(\mu_i - \mu_0) - \beta_i \left(A_i^*(\mu_i,\nu_i) - \mu_i^\top \theta'_i + A(\theta'_i)\right)\right) \\ \text{subject to} \quad & \mu_0 \in \mathcal{M} \\ & [\mu_i,\nu_i] \in \mathcal{M}_i \quad \text{for } i = 1,\ldots,n \\ & \theta'_i \in \Theta \quad \text{for } i = 1,\ldots,n\end{aligned} \tag{32}$$

Maximizing over $\mu_0, [\mu_i,\nu_i]_{i=1}^n$, we have the equivalent dual objective,

$$\begin{aligned}\max_{[\theta'_i]_{i=1}^n} \min_{[\lambda_i]_{i=1}^n} & A\left(\theta_0 + \sum_{i=1}^n \lambda_i\right) + \sum_{i=1}^n \beta_i \left(A_i\left(\theta'_i - \beta_i^{-1}\lambda_i, \beta_i^{-1}\right) - A(\theta'_i)\right) \\ \text{subject to} \quad & \theta'_i \in \Theta \quad \text{for } i = 1,\ldots,n\end{aligned} \tag{33}$$

4.2 Stochastic Natural Gradient Descent

The dual problem can be solved using a double loop algorithm, in which the dual parameters λ_i are minimized by coordinate descent in an inner loop, and the auxiliary variables θ'_i are maximized in an outer loop. It can be shown that this leads to the convergent double loop EP algorithm of (Heskes and Zoeter, 2002). The novelty of our derivation lies in that a single global variational problem is described, rather than a sequence of variational problems each obtained by minorizing around the current parameters and then maximizing, as in the minorization-maximization (MM) algorithm) (Hunter and Lange, 2004) and the CCCP algorithm (Yuille, 2002).

To deal with the stochasticity inherent in using Monte Carlo estimators, we will instead use a stochastic gradient descent algorithm as the inner loop. From (33), the gradient of the dual objective $L([\theta'_j, \lambda_j]_{j=1}^n)$ is,

$$\frac{dL}{d\lambda_i} = \nabla A\left(\theta_0 + \sum_{j=1}^n \lambda_j\right) - \nabla_{\theta_i} A_i\left(\theta'_i - \beta_i^{-1}\lambda_i, \beta_i^{-1}\right) \tag{34}$$

where we used the notation $\nabla_{\theta_i} A_i$ for the partial derivative of $A_i(\cdot, \cdot)$ with respect to its first, d-dimensional, argument.

The gradients above are not covariant and the algorithm is not expected to perform well. A better approach is to use natural gradient descent (Amari and Nagaoka, 2001) or, equivalently, mirror descent (Beck and Teboulle, 2003, Raskutti and Mukherjee, 2015) instead. As noted in the previous section, the Lagrange multiplier λ_i can be interpreted as the natural parameters of the base exponential family approximation to the likelihood $\exp(\ell_i(x))$. Reparameterising using the corresponding mean parameter γ_i instead, with $\lambda_i = \nabla A^*(\gamma_i)$, the gradient is,

$$\begin{aligned}\frac{dL}{d\gamma_i} &= \frac{d\lambda_i}{d\gamma_i}\left(\nabla A\left(\theta_0 + \sum_{j=1}^n \lambda_j\right) - \nabla_{\theta_i} A_i\left(\theta'_i - \beta_i^{-1}\lambda_i, \beta_i^{-1}\right)\right) \\ &= \nabla^2 A^*(\gamma_i)\left(\nabla A\left(\theta_0 + \sum_{j=1}^n \lambda_j\right) - \nabla_{\theta_i} A_i\left(\theta'_i - \beta_i^{-1}\lambda_i, \beta_i^{-1}\right)\right)\end{aligned} \tag{35}$$

The appropriate metric in the mean parameter space is simply $\nabla^2 A^*(\gamma_i)$, so that its inverse cancels the $\nabla^2 A^*(\gamma_i)$ term (Raskutti and Mukherjee, 2015), and the natural gradient update is simply,

$$\gamma_i^{(t+1)} = \gamma_i^{(t)} + \epsilon_t\left(\nabla_{\theta_i} A_i\left(\theta'_i - \beta_i^{-1}\lambda_i^{(t)}, \beta_i^{-1}\right) - \nabla A\left(\theta_0 + \sum_{j=1}^n \lambda_j^{(t)}\right)\right) \tag{36}$$

where the corresponding natural parameter is given as a function of the mean parameter, $\lambda_i^{(t)} := \nabla A^*(\gamma_i^{(t)})$, and ϵ_t is the step size at iteration t. These updates can be performed in series or parallel fashion. In our distributed Bayesian learning setting they are performed in an asynchronous distributed fashion.

Intuitively, the first term of the natural gradient step is the mean parameter of the current local tilted distribution,

$$p_i^{(t)}(x) = \exp\left((\theta'_i - \beta_i^{-1}\lambda_i^{(t)})^\top s(x) + \beta_i^{-1}\ell_i(x) - A_i(\theta'_i - \beta_i^{-1}\lambda_i^{(t)}, \beta_i^{-1})\right), \tag{37}$$

while the second term is the mean parameter of the current exponential family approximation to the global posterior. Their difference gives the update for the mean parameter of the likelihood approximation. The gradient is zero when both terms are equal, precisely the condition from which the EP fixed point equation is derived.

In general, the first term cannot be obtained in closed form, and we instead use a Markov chain Monte Carlo (MCMC) estimate, leading to a stochastic natural-gradient descent algorithm. Specifically, let $\mathcal{K}_i(\cdot \mid x; \theta'_i - \beta_i^{-1}\lambda_i^{(t)}, \beta_i)$ be a Markov chain kernel with previous state x whose invariant distribution is the local tilted distribution (37). Let $x_i^{(t)}$ be the state of the Markov chain at iteration t. The next state $x_i^{(t+1)}$ is obtained by simulating from the Markov chain, using the current values of the parameters,

$$x_i^{(t+1)} \sim \mathcal{K}_i\left(\cdot \mid x_i^{(t)}; \theta'_i - \beta_i^{-1}\lambda_i^{(t)}, \beta_i\right). \tag{38}$$

In summary, the stochastic natural gradient update is,

$$\gamma_i^{(t+1)} = \gamma_i^{(t)} + \epsilon_t \left(s\left(x_i^{(t+1)}\right) - \nabla A\left(\theta_0 + \sum_{j=1}^{n} \lambda_j^{(t)}\right)\right) \tag{39}$$

Technically, the stochastic natural-gradient descent algorithm requires unbiased estimates of gradients, and the mean parameter estimates obtained using MCMC updates are only unbiased if the Markov chain equilibrates in between gradient updates. In practice, we find the following to work well: We initialise the Markov chains by having $x_i^{(1)}$ be distributed according to the initial base exponential family approximation with natural parameters $\theta_0 + \sum_{j=1}^{n} \lambda_j^{(1)}$, so that the Markov chain will in general drift from the exponential family approximation to the tilted distribution, and the gradient (39) obtained even after one step of the Markov chain will in general point in the right direction. This is reminiscent of the intuition behind contrastive divergence (Hinton, 2002) and persistent contrastive divergence (Tieleman, 2008).

Supposing that the inner loop has converged, in the outer loop we can simply set θ'_i to be the natural parameter corresponding to the mean parameter μ_i of the local tilted distribution. Assuming that the inner loop has converged, this and the mean parameters of all other tilted distributions would be equal to μ_0, so that the t'th outer loop update is,

$$\begin{aligned}(\theta'_i)^{(t'+1)} &= \nabla A^*(\mu_i) = \nabla A^*\left(\nabla_{\theta_i} A_i\left((\theta'_i)^{(t')} - \beta_i^{-1}\lambda_i^{(\infty)}, \beta_i^{-1}\right)\right) \\ &= \theta_0 + \sum_{j=1}^{n} \lambda_j^{(\infty)}\end{aligned} \tag{40}$$

where $\lambda_j^{(\infty)}$ is the converged value of the dual parameters in the inner loop. In practice, we simply perform the outer loop update infrequently (and before the inner loop has fully converged). In our experiments we do not see instabilities resulting from this.

In the extreme case where the outer loop update is performed after every inner loop update, we can roll both updates into the following update,

$$\gamma_i^{(t+1)} = \gamma_i^{(t)} + \epsilon_t \left(\nabla_{\theta_i} A_i \left(\theta_0 + \sum_{j=1}^{n} \lambda_j^{(t)} - \beta_i^{-1}\lambda_i^{(t)}, \beta_i^{-1} \right) - \nabla A\left(\theta_0 + \sum_{j=1}^{n} \lambda_j^{(t)}\right)\right) \tag{41}$$

It is interesting to contrast the above with a damped EP update, which in our notation is given by,

$$\lambda_i^{(t+1)} = \lambda_i^{(t)} + \epsilon_t \left(\nabla A^* \left(\nabla_{\theta_i} A_i \left(\theta_0 + \sum_{j=1}^n \lambda_j^{(t)} - \beta_i^{-1} \lambda_i^{(t)}, \beta_i^{-1} \right) \right) - \left(\theta_0 + \sum_{j=1}^n \lambda_j^{(t)} \right) \right) \tag{42}$$

Note that $\nabla A^*(\cdot)$ converts from mean parameters to natural parameters, so that the first term in parentheses can be read as first computing the moments (mean parameters) of the tilted distribution then converting into the corresponding natural parameter. Hence each term in the damped EP update is obtained by converting the corresponding term in (41) from mean to natural parameters, i.e. applying $\nabla A^*(\cdot)$. In other words the update (41) can be thought of as a mean parameter space version of a damped EP update.

4.3 Discussion and Related Works

We refer to the resulting algorithm above as Stochastic Natural-gradient EP, or SNEP in short. While we developed SNEP in the context of distributed Bayesian learning, it is clear that is is generally applicable, since the distribution (15) targeted is simply a product of factors, each of which is approximated by a factor in the base exponential family, precisely the setting of EP. SNEP can be used in place of EP in situations where Monte Carlo moment estimates are used, including graphical models (Heess et al., 2013, Eslami et al., 2014, Jitkrittum et al., 2015, Lienart et al., 2015), hierarchical Bayesian models (Gelman et al., 2014), embarrassingly parallel MCMC sampling (Xu et al., 2014) and approximate Bayesian computation (Barthelmé and Chopin, 2011).

Since Minka (2001), there have been a substantial number of extensions and alternatives to EP proposed. Stochastic EP (Li et al., 2015) and averaged EP (Dehaene and Barthelmé, 2015) assume that all factors can be well approximated by the same exponential family factor. This saves memory storage and was shown to work surprisingly well. It is possible to apply this idea to the SNEP setting as well. Convergent EP (Heskes and Zoeter, 2002) is a double loop convergent EP alternative, but with coordinate descent as its inner loop, rather than stochastic natural-gradient descent in SNEP. This means that convergent EP cannot easily make use of Monte Carlo estimated moments.

The key advantage of SNEP is that because it uses a stochastic natural-gradient descent inner loop, it allows for the use of Monte Carlo estimators for the mean parameters of the tilted distribution. This allows it to be used in black-box settings, where the only requirement is the existence of MCMC samplers targeting the tilted distributions. Black-box methods have recently been developed for variational Bayes (Ranganath et al., 2014) and for power EP (Hernandez-Lobato et al., 2015). In these prior works on black-box variational inference, Naïve Monte Carlo estimators are used, with samples drawn from the approximating distribution. The resulting estimators have high variance, requiring control variates for variance reduction. In contrast SNEP uses MCMC samplers targeting the tilted distribution. It is generally accepted that in high-dimensional settings MCMC samplers often have lower variance than naïve Monte Carlo and as a result work better, with the trade-off that MCMC samplers need to equilibrate. Stochastic natural-gradient descent was also used in stochastic variational inference (Hoffman et al., 2013).

5 The Posterior Server

Our development of SNEP is motivated by the problem of distributed Bayesian learning outlined in Section 2.2, where each log likelihood term $\ell_i(x)$ corresponds to the log probability/density of the data subset on worker node i. Using SNEP, each worker node iteratively learns an exponential family approximation of $\ell_i(x)$, with a master node coordinating the learning across workers. We call the master node the *posterior server*, as it maintains and serves the exponential family approximation of the posterior distribution, obtained by combining the prior with the likelihood approximations at the workers.

In more detail, worker node i maintains the mean and natural parameters γ_i, λ_i of the likelihood approximation and the state of the MCMC sampler x_i. It also maintains the auxiliary parameter θ'_i used in the outer loop. Learning at the worker proceeds by alternating between the MCMC (38) and inner loop updates (39), with periodic outer loop updates (40). These updates require access to the data subset at the node, as well as the cavity distribution, with natural parameters $\theta_{-i} := \theta_0 + \sum_{j \neq i} \lambda_j$, which is obtained by communicating with the posterior server. The posterior server maintains the natural parameter $\theta_{\text{posterior}} := \theta_0 + \sum_{j=1}^{n} \lambda_j$ of the global posterior approximation.

Communication between the worker node and the posterior server involves the worker first sending the posterior server the difference $\Delta_i := \lambda_i^{\text{new}} - \lambda_i^{\text{old}}$ between the current natural parameters λ_i^{new} and the one during the previous communication with the server, λ_i^{old}. The posterior server updates its global posterior approximation via $\theta_{\text{posterior}}^{\text{new}} = \theta_{\text{posterior}}^{\text{old}} + \Delta_i$, and sends the new value $\theta_{\text{posterior}}^{\text{new}}$ back to the worker. The worker in turn uses this to update the cavity, $\theta_{-i}^{\text{new}} = \theta_{\text{posterior}}^{\text{new}} - \lambda_i^{\text{new}}$. Note that communication on the cluster is asynchronous and non-blocking, so that the worker continues its updates in between it sending the message to the posterior server and receiving the return message, so that the λ_i^{new} use above to compute θ_{-i}^{new} should be the natural parameter used previously to compute Δ_i, not the most recent natural parameter.

The pseudocode for the overall algorithm is given in Algorithm 1. Note that all communications are performed asynchronously and in a non-blocking fashion. In particular, Steps 11-17 are performed in a separate coroutine from the main loop (Steps 7-18), and Step 15 can happen several iterations of the main loop after Step 14. This is so that compute nodes can spend most of their time learning (Steps 8-10) and do not have to wait for network communications to complete. We also note that faster compute nodes need not wait for slower ones since they each learn their own separate likelihood approximation parameters. It would be interesting for future research to explore adaptive methods to allow faster compute nodes to increase the data subsets that they learn from, and slower ones to decrease, to balance the learning progress across compute nodes more evenly.

5.1 Discussion

Our naming of the posterior server contrasts with that of the parameter server (Ahmed et al., 2012) which is typically used for maximum likelihood (or minimum empirical risk) estimation of model parameters. Note however that our algorithmic contribution is effectively orthogonal to (Ahmed et al., 2012), who proposed a generic and robust computational architecture for distributed machine learning. We believe it is possible to implement our algorithm using the parameter server software framework.

One of the difficulties of the parameter server architecture is that learning happens at the level of parameters, with a single set of parameters being maintained across the cluster. Since the data subsets on workers are disjoint, the learning on each worker tends to make the local copy of the

Algorithm 1 Posterior Server: Distributed Bayesian Learning via SNEP

```
for each compute node $i = 1, \ldots, n$ asynchronously do
  let $\gamma_i^{(1)}$ be the initial mean parameter of local likelihood approximation.
  let $\lambda_i^{\text{old}} := \lambda_i^{(1)} := \nabla A^*(\gamma_i^{(1)})$ be the initial natural parameter of local likelihood approximation.
  let $\theta_{-i} := \theta_0 + \sum_{j \neq i} \lambda_j^{(1)}$ be the initial natural parameter of cavity distribution
  let $\theta_i' := \theta_{-i} + \lambda_i^{(1)}$ be the initial auxiliary parameter.
  let $x_i^{(1)} \sim p_{\theta_{-i} + \lambda_i^{(1)}}$ be the initial state of MCMC sampler.
  for $t = 1, 2, \ldots$ until convergence do
    update local state via MCMC:
        $x_i^{(t+1)} \sim \mathcal{K}_i \left( \cdot | x_i^{(t)}; \theta_i' - \beta_i^{-1} \lambda_i^{(t)}, \beta_i^{-1} \right)$
    update local likelihood approximation:
        $\gamma_i^{(t+1)} := \gamma_i^{(t)} + \epsilon_t \left( s(x_i^{(t+1)}) - \nabla A \left( \theta_{-i} + \lambda_i^{(t)} \right) \right)$
        $\lambda_i^{(t+1)} := \nabla A^*(\gamma_i^{(t+1)})$
    every $N_{\text{outer}}$ iterations do: update auxiliary parameter:
        $\theta_i' := \theta_{-i} + \lambda_i^{(t)}$
    every $N_{\text{sync}}$ iterations asynchronously do: communicate with posterior server:
      let $\Delta_i := \lambda_i^{(t)} - \lambda_i^{\text{old}}$.
      update $\lambda_i^{\text{old}} := \lambda_i^{(t)}$.
      send $\Delta_i$ to posterior server.
      receive $\theta_{\text{posterior}}$ from posterior server.
      update $\theta_{-i} := \theta_{\text{posterior}} - \lambda_i^{\text{old}}$.
  end for
end for
for the posterior server do
  let $\theta_{\text{posterior}} := \theta_0 + \sum_{j=1}^{n} \lambda_i^{(1)}$ be the initial natural parameter of the posterior approximation.
  maintain a queue of messages from workers.
  for each message $\Delta_i$ received from some worker $i$ do
    update $\theta_{\text{posterior}} := \theta_{\text{posterior}} + \Delta_i$.
    send $\theta_{\text{posterior}}$ to worker $i$.
  end for
end for
```

parameters diverge from those on the parameter server and on other workers. As a result, frequent synchronisation with the parameter server is necessary to prevent stale parameters and gradients. As an example, in the DistBelief method (Dean et al., 2012), experiments were conducted where the communication with the master was performed after every iteration, which can significantly slow down the learning process. On the other hand, one of the interesting aspects of the posterior server is that it lifts learning from the level of parameters to the level of distributions over parameters. As a result each worker can maintain a distinct parameter set in the MCMC sampler and a distinct likelihood approximation (since the likelihoods on different workers are indeed different as they have different data subsets) without requiring frequent communication with the posterior server. The only role of communication here is for the cavity distributions, which can be thought of as way for the system to focus the learning happening on workers on the relevant regions of the parameter space. Empirically, the precise parameterisation of the cavity distribution is not very important. For an extreme example, suppose both the prior and likelihood terms are close to being Gaussians and the tractable family is also Gaussian. Then the likelihood approximation will not in fact depend on the cavity distribution at all, and will converge to the true likelihood on each worker independently. See also Zhang et al. (2015) for a similar idea of allowing each worker a separate parameter vector, using an ADMM-like methodology.

Our application of SNEP to distributed Bayesian learning applies one exponential family approximation per subset of data on each worker node. This contrasts with typical applications of EP and variational inference in general, which applies one approximation per data item. This is made possible due to the black-box flexibility of our approach, since the likelihood associated with a data subset is more complex than for a single data item. In cases where the subset is itself quite large, and the Bernstein-von Mises theorem holds, the likelihood will be close to a Gaussian, so that if we use a (full-covariance) Gaussian as the base exponential family, the approximation will introduce negligible biases. For a recent study of EP in the large data limit, see (Dehaene and Barthelmé, 2015). We can think of our approach as a hybrid which interpolates between a pure variational approach (when $n = N$) and a pure MCMC approach (when $n = 1$), with smaller n corresponding to less bias introduced by approximations but higher variance/computational costs.

6 Additional Techniques for Bayesian Neural Networks

In the next section we will investigate the use of SNEP and the posterior server for distributed Bayesian learning of neural networks. To get the method working well on the notoriously complicated posterior distributions for neural networks, a number of additional techniques are needed, which we describe here. This section may be skipped if the reader is not interested in applications to neural networks.

6.1 Initial Mode Agreement

The learning landscape of neural networks are highly multimodal, with significant non-identifiabilities associated with, e.g. permutations of the units or filters in each layer of the network. If the network parameters are initialised randomly, in a distributed setting the initial learning phase can be very slow because the parameters on different workers are attracted to different modes, and communication across the cluster can take a while for all nodes to agree on a single mode. We address this issue by simply having a single compute node learn for a small number N_{init} of iterations, then using its learnt parameters to initialise all workers.

6.2 Adaptive Stochastic Gradient Langevin Dynamics

Most of the computational costs associated with the algorithm involve the MCMC updates to the state x_i. When the number of data points stored on each compute node is large, standard MCMC updates are infeasible as each update requires computations involving every data point. In our experiments we used the stochastic gradient Langevin dynamics (SGLD) algorithm proposed by Welling and Teh (2011) which scales well to large datasets.

SGLD uses mini-batches of data to provide unbiased estimates of gradients which are used in a time discretized Langevin dynamics simulation whose stationary distribution is the desired tilted distribution (37). The discretization introduces errors which go to zero as the discretization step sizes decreases to zero; see (Teh et al., 2015, Vollmer et al., 2016). Recall that the data points on compute node i is $D_i = \{y_c\}_{c \in S_i}$, and the log likelihood is

$$\ell_i(x_i) = \sum_{c \in S_i} \log p(y_c | x_i). \tag{43}$$

Let $B^{(t)} \subset S_i$ be a mini-batch of data, chosen uniformly at random with fixed size. Each SGLD update is,

$$\begin{aligned} x_i^{(t+1)} &= x_i^{(t)} + \kappa_t (M^{(t)})^{-1} \left(\nabla s(x_i^{(t)})^\top (\theta_i' - \beta_i^{-1} \lambda_i^{(t)}) + \beta_i^{-1} \frac{|S_i|}{|B^{(t)}|} \sum_{c \in B^{(t)}} \nabla \log p(y_c | x_i^{(t)}) \right) + \eta_t, \\ \eta_t &\sim \mathcal{N}(0, 2\kappa_t (M^{(t)})^{-1}). \end{aligned} \tag{44}$$

The term inside the parentheses is an unbiased estimate of the gradient of the log density of the tilted distribution (37), κ_t is the discretization step size, $M^{(t)}$ is (an adaptive) diagonal mass matrix, while η_t is an injected normally-distributed noise, which prevents SGLD from converging to a mode of the distribution and distinguishes it from stochastic gradient descent. See Welling and Teh (2011) for details.

In (44) ($M^{(t)}$) is a mass matrix which effectively controls the length scale of updates to each dimension of x_i. It is well known that in neural networks the length scales of gradients differ significantly across different parameters and adaptation of learning rates specific to each parameter is crucial for successful deployment of stochastic gradient descent learning. We have found that this is the case for SGLD as well, and used an adaptation scheme for the mass matrix reminiscent of Adagrad (Duchi et al., 2011), RMSProp (Tieleman and Hinton, 2012) and Adam (Kingma and Ba, 2015). At iteration t let g_t be the gradient estimate in (44). We use a diagonal mass matrix. Suppose its kth entry at iteration $t-1$ is $M_{kk}^{(t-1)}$, then this is updated at iteration t using:

$$(M_{kk}^{(t)})^2 = (1 - t^{-\frac{1}{2}})(M_{kk}^{(t-1)})^2 + t^{-\frac{1}{2}} g_{tk}^2 \tag{45}$$

In addition, as in RMSProp, we set a minimum value $M_{\min}$ for the diagonal entries of the mass matrix (we used 10^{-5} in our experiments). If instead of $t^{-\frac{1}{2}}$ we used a step size of t^{-1} for the adaptation, the above would simply be an average over the square of all previous gradients, and it would basically be Adagrad (except that the SGLD step sizes κ_t are set separately (and in fact kept constant)). We prefer this adaptation schedule over the one in Adagrad as it decreases more slowly and is less sensitive to the gradients at the initial iterations. We also prefer it over the constant one used in RMSProp as it decreases over time so that the adaptation stabilizes.

6.3 Shifting MCMC States After Communication with Posterior Server

After each communication with the posterior server, the target distribution of the MCMC sampler on the worker, say i, is changed, because of θ_{-i} being updated in Step 16 of Algorithm 1. Assuming that the MCMC sampler was previously converged, it will now not be anymore because of this shift in the target distribution, and a number of burn-in iterations may be needed before the mean parameter estimates can be used for SNEP updates again.

For Gaussian base exponential families, we can shift the MCMC state along with the target distribution when θ_{-i} is updated in the following way. Suppose μ_i^{old}, Σ_i^{old}, μ_i^{new}, Σ_i^{new} are the means and covariances of the approximate Gaussian posterior before and after the update to θ_{-i} (with natural parameters $\lambda_i + \theta_{-i}^{\text{old}}$, $\lambda_i + \theta_{-i}^{\text{new}}$ respectively where λ_i is the current natural parameter of the Gaussian likelihood approximation). Suppose x_i^{old} is the MCMC state before the update. Then we shift the MCMC state as follows:

$$x_i^{\text{new}} = \mu_i^{\text{new}} + (\Sigma_i^{\text{new}})^{\frac{1}{2}} (\Sigma_i^{\text{old}})^{-\frac{1}{2}} (x_i^{\text{old}} - \mu_i^{\text{old}}). \tag{46}$$

The idea is that x_i^{new} should be at the same location relative to the new Gaussian approximation to the posterior as x_i^{old} is relative to the old Gaussian approximation. We have found that no burn-in is needed with this shift in the MCMC state after each communication.

6.4 Averaging across Iterations

To stabilize the learning, we average the posterior estimates across iterations using Polyak averaging (Polyak and Juditsky, 1992). Specifically, we keep a running average of the natural parameter $\theta_{\text{posterior}}$ of the posterior approximation. We find this this improves the stability and quality of the posterior approximation.

7 Experiments

In this section we report on some initial experiments on SNEP and the posterior server. Our experiments were performed where each worker is a separate core on a server. As the MCMC sampler on workers, we chose an adaptive version of stochastic gradient Langevin dynamics (SGLD)(Welling and Teh, 2011) related to Adagrad (Duchi et al., 2011) and Adam (Kingma and Ba, 2015), which is more computationally scalable to larger models and datasets than standard MCMC. See Appendix 6 for details.

7.1 Comparison to SMS on Bayesian Logistic Regression

We start with a comparison of SNEP/posterior server to EP/SMS. Recall that SMS is an algorithm for distributed Bayesian learning whereby each worker has a separate MCMC sampler and coordination across workers is achieved using EP (Xu et al., 2014). It was originally proposed to scale up MCMC methods and as such it assumes that MCMC chains can be run to convergence in between communications with the master. Here we used SGLD in place of the No-U-Turns sampler (NUTS) (Hoffman and Gelman, 2014), with a relatively small number of SGLD iterations per communication. We have found that the moment estimates are quite noisy which adversely affected SMS much more than SNEP. While damping tends to improve SMS performance, it still exhibits more erratic dynamics.

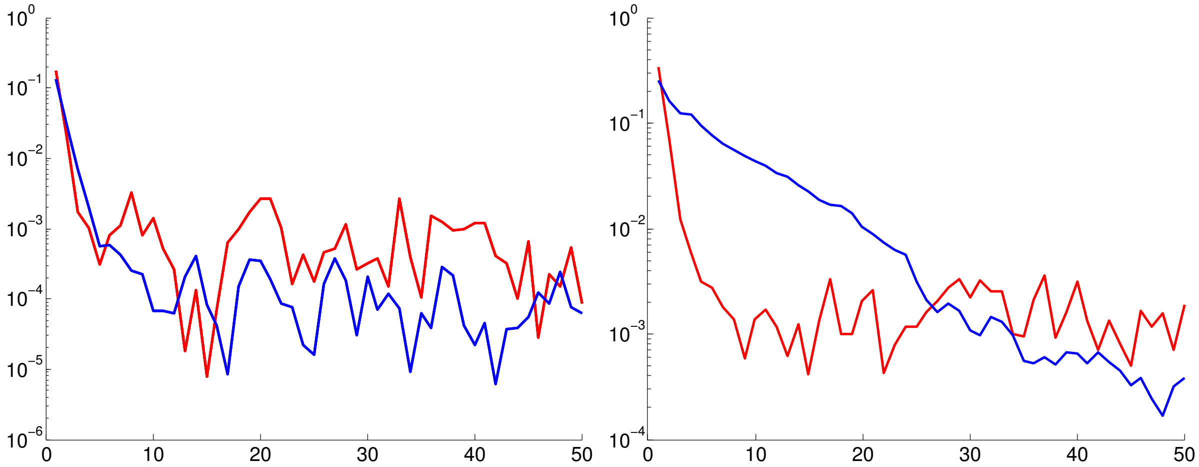

Figure 2: Comparison of MSE between ground truth and approximate posterior mean of SNEP (blue) and SMS (red) as function of iteration on a two-dimensional (left) and a ten-dimensional (right) example.

We illustrate the differing dynamics of SMS and SNEP using Bayesian logistic regression on a simulated dataset with 1000 data items, using the MATLAB codebase and experimental setup (except we use a diagonal covariance prior) of Xu et al. (2014). For both algorithms we use $N_{worker} = 4$ and diagonal covariance Gaussians as the base exponential family. We initialise both algorithms at identical points and use 5000 iterations of SGLD. We set $N_{outer} = N_{sync} = 50$, performing each outer loop iteration in SNEP at the same time as communication with master. We used 50 SGLD iterations for each moment estimation needed for SMS.

First, we present results on a two-dimensional problem. Figure 2 (left) shows the mean squared error (MSE) between the approximate posterior mean and the ground truth. we observe that both SNEP and SMS achieve similar MSE. Next, Figure 3 shows the dynamics of SNEP (left) and SMS (right). The red cross denotes the ground truth posterior mean (as estimated using a long run of NUTS), the blue curve shows the evolution of the estimated posterior mean, with the blue circle being the final approximation. Each black curve shows the evolution of the mean of the Gaussian approximation at one worker, with the black square being the final approximation. While the MSE of SMS and SNEP are comparable, we observe that SNEP factor approximations are more stable compared to SMS factor approximations. Next, we compare the results on a ten-dimensional example. Figure 2 (right) shows the MSE and Figure 4 shows the dynamics of the first two dimensions. We see that while EP displays faster initial convergence, SNEP is more robust and stable, resulting in better approximation of the ground truth posterior mean.

7.2 Bayesian Neural Networks

In this section we report preliminary experimental results applying SNEP and the posterior server to distributed Bayesian learning of neural networks, with an implementation using the Julia technical computing language[2] which will be made publicly available at `http://bigbayes.github.com/PosteriorServer`. We have found that the SMS algorithm exhibited significant instabilities and

[2] `http://julialang.org`.

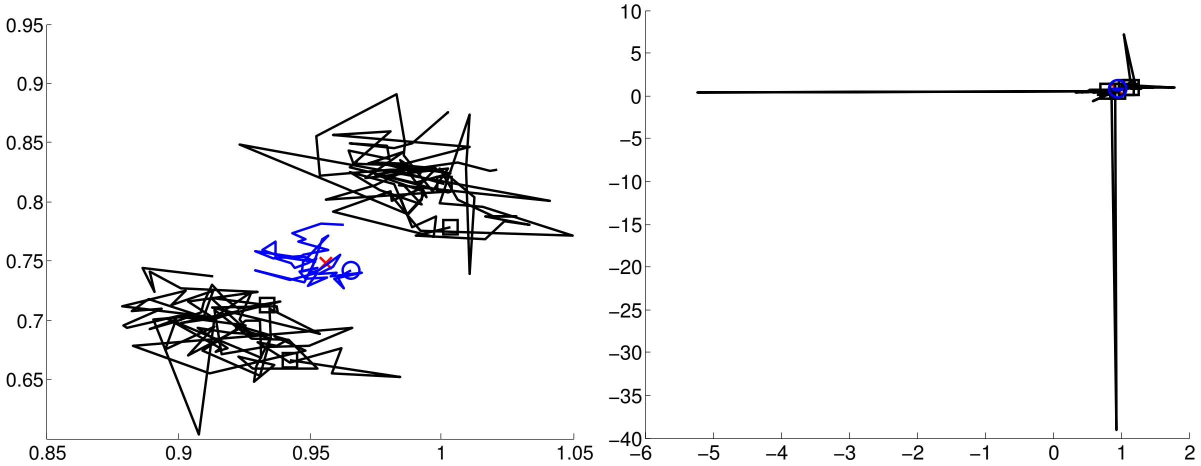

Figure 3: Comparison of dynamics of SNEP (left) and SMS (right) on a two-dimensional example. Green diamond shows prior mean, red cross shows ground truth, blue line show the evolution of the approximate posterior mean, and black lines show the evolution of the mean of the Gaussian likelihood approximations. Blue circle and black squares show the values at the end of the simulation (5000 iterations of SGLD).

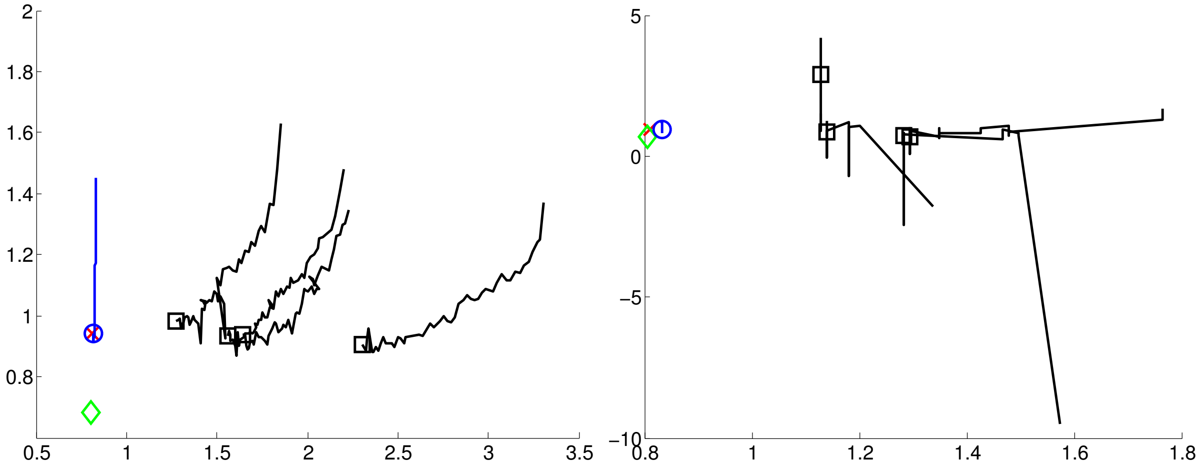

Figure 4: Comparison of dynamics of SNEP (left) and SMS (right) on a ten-dimensional example, where the x and y axes show the first two dimensions only. Green diamond shows prior mean, red cross shows ground truth, blue line show the evolution of the approximate posterior mean, and black lines show the evolution of the mean of the Gaussian likelihood approximations. Blue circle and black squares show the values at the end of the simulation (5000 iterations of SGLD).

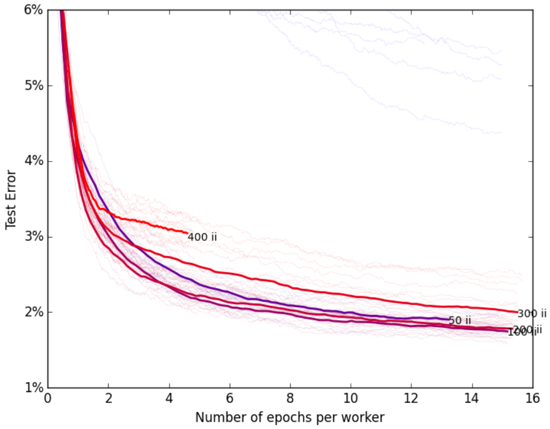

Figure 5: Learning curves as N_{init} is varied, with values in $\{0, 50, 100, 200, 300, 400\}$. Colours correspond to different values, with blue being 0 and red being 400. Each thick curve is obtained by averaging 10 runs (thin lines). An epoch corresponds to 600 SNEP inner loop iterations.

was not suitable for these larger scale problems. Instead our aim here is to explore the behaviour of SNEP when varying various key hyperparameters. We will also compare our distributed learning method to a state-of-the-art stochastic gradient descent (SGD) algorithm with access to the whole dataset on a single computer.

7.2.1 MNIST, Fully Connected Network

In this section we look at training a fully connected neural network on the MNIST dataset[3], which consists of 60000 training images of handwritten digits of size 28 × 28 and 10000 test images, the task being to classify each image into one of ten classes. The network has two hidden layers with 500 and 300 leaky ReLU units respectively and softmax output units.

In the first set of experiments, we varied a number of hyperparameters of the learning regime while keeping the rest kept at default values, to investigate the sensitivity of the learning to these hyperparameters. The default values were chosen by hand in a rough initial round of experimentation as follows: the number of workers is $N_{worker} = 8$, the minibatch size is 100, the number of initial single worker iterations is $N_{init} = 200$, the learning rate for SNEP and step size for SGLD are both 0.01, the number of iterations per communication is $N_{sync} = 10$, the number of inner loop iterations per outer loop update is $N_{outer} = 50$, and each parameter initialised with iid $\mathcal{N}(0, 1)$ draws. Test curves were produced by evaluating networks based on the approximate posterior means.

Figure 5 shows the learning curves as N_{init} is varied. Note that runs with $N_{init} = 0$ performed significantly worse (blue lines are in the top right corner), demonstrating that the initial phase of learning is crucial for workers to agree on a good initial local mode. In practice a small number of initial iterations were sufficient. Larger numbers seem to be slightly detrimental, possibly because the initial iterations have converged to suboptimal local optima. Some runs with $N_{init} = 400$ were

[3] http://yann.lecun.com/exdb/mnist/.

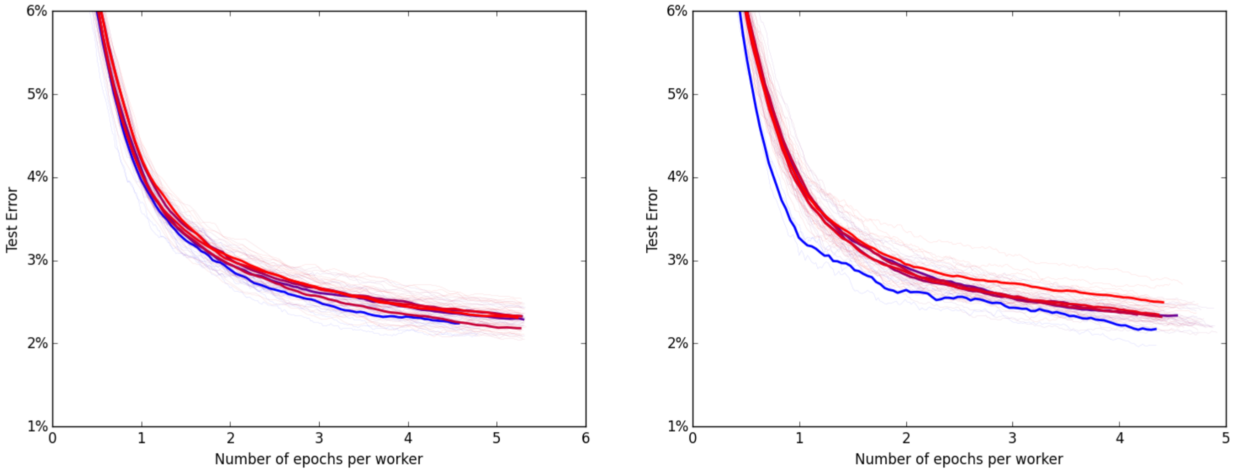

Figure 6: Learning curves as N_{sync} is varied in range $\{10, 20, 30, 40, 50, 60\}$ (left) and N_{outer} is varied in range $\{1, 10, 20, 50, 100, 200\}$. Colours correspond to different values, with blue being lower values and red being higher. Each thick curve is obtained by averaging 10 runs (thin lines). An epoch corresponds to 600 SNEP inner loop iterations.

terminated before completion due to external circumstances.

Figure 6 shows behaviour as N_{sync} and N_{outer} are varied, showing that the method is insensitive to these hyperparameters over reasonably large range of values. Note in particular that infrequent communications with the posterior server (up to once every 60 iterations in the experiment) did not significantly deteriorate the learning process at all. In fact, the related SMS algorithm (Xu et al., 2014) effectively involves communications with the master once every thousands of iterations.

Finally, Figure 7 shows behaviour as N_{worker} is varied. We see that increasing the number of workers improves performance. In particular, distributed learning with 8 to 12 workers performed significantly better than the corresponding method with a single worker. This is possibly due to more effective use of computational resources, or better exploration of the learning landscape due to more varied workers, similar to the effect observed by Zhang et al. (2015). Beyond 12 workers performance decreases, possibly due to smaller number of data items per worker so the variational approximation is more severe.

In Figure 7 we also compared against Adam (Kingma and Ba, 2015), a non-distributed SGD algorithm where a single worker has access to the whole dataset. With the same initialisation as our method, we see that Adam performed significantly worse, although with an alternative initialisation (Glorot and Bengio, 2010) Adam was able to converge faster to a better local mode. We expect that with further exploration of initialisation schemes our method can be further improved. It is worthwhile emphasising that our method works in situations where data is distributed across a cluster with no worker having access to the whole dataset. This is inherently a harder learning problem than one where each worker has access to the whole dataset, so it is not surprising for our method to perform worse, and it is encouraging for our method to achieve close to start-of-the-art performance efficiently.

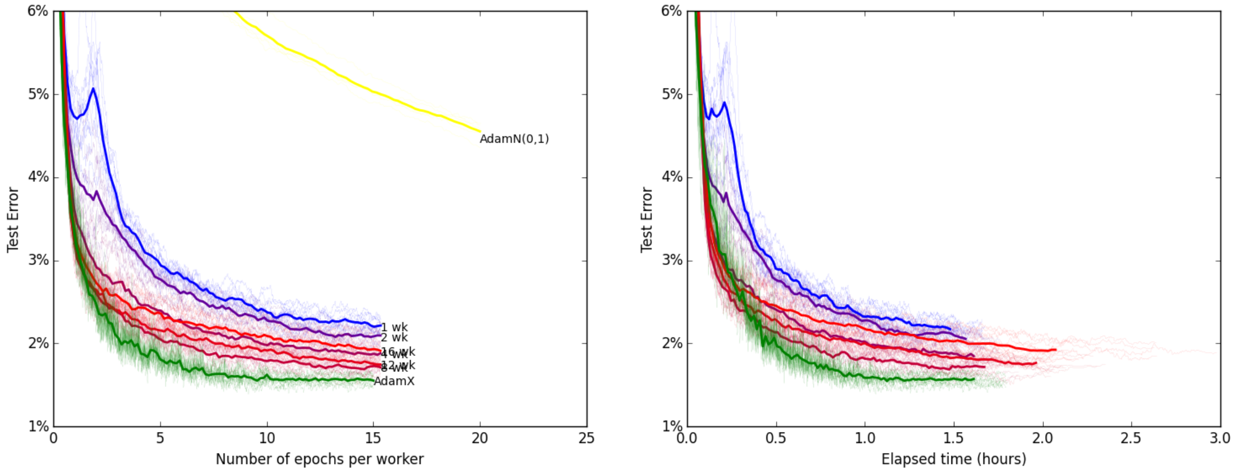

Figure 7: Learning curves as N_{worker} is valued in range $\{1, 2, 4, 8, 12, 16\}$, plotted against number of epochs (left) and wall-clock time (right). Colours correspond to different values, with blue being lower values and red being higher. Each thick curve is obtained by averaging 10 runs (thin lines). An epoch corresponds to 600 SNEP inner loop iterations. As comparison, we also plot learning curves for Adam with two different initialisations: $\mathcal{N}(0, 1)$ matching the initialisation used for our method and Xavier initialisation (Glorot and Bengio, 2010).

7.3 CIFAR-10, Convolutional Network

We also experimented with distributed Bayesian learning of convolutional neural networks, applying these to the CIFAR-10 dataset (Krizhevsky, 2009) which consists of 50000 training instances and 10000 test instances from 10 classes, each instance being a 32x32 colour natural image. The network used is the one described in Alex Krizhesky's CIFAR tutorial[4] and consists of 8 layers: a first convolutional layer followed by max-pooling and local response normalization, a second convolutional layer also followed by max-pooling and local response normalization, and a third convolutional layer followed by a fully connected layer.

For the posterior server, we used the following settings: $N_{worker} = 8$, $N_{sync} = 10$, $N_{outer} = 10$, $N_{init} = 2000$, mini-batch size 100, weights initialised with $\mathcal{N}(0, 0.01)$ draws, and SNEP learning rate and SGLD step sizes of 0.001. Learning curves for 10 runs are shown in Figure 8, along with learning curve for a run of Adam. The learning curves for the distributed phase are based on a running average of the approximated posterior mean, using Polyak averaging (Polyak and Juditsky, 1992) in the natural parameterisation; we found that averaging in the mean parameterisation produced the same results. Averaging produced more stable and slightly better results. The posterior server produced comparable but slightly worse results than Adam, in spite of the distributed data learning problem being harder. We did not investigate improving performance by building in invariances using data perturbations or having multiple learning phases with different learning rates.

We also investigated using the approximate posterior samples obtained at the workers to form predictive probabilities, averaging these over samples and workers. Figure 9 shows a learning curve

[4] `https://code.google.com/p/cuda-convnet/wiki/Methodology`

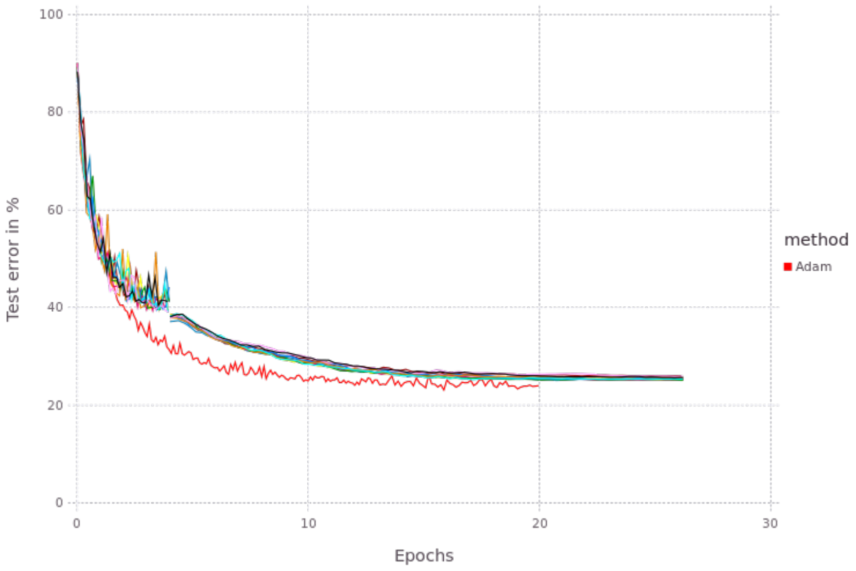

Figure 8: Learning curves for 10 runs of the posterior server (multi-coloured) and a run of Adam (red). The initial one-worker and distributed phases are separated by a gap.

in comparison with using the approximate posterior mean. This improved the test accuracy slightly.

8 Discussion

We have proposed a novel alternative to expectation propagation called stochastic natural-gradient expectation propagation (SNEP). SNEP is demonstrably convergent, even when using Monte Carlo estimates of the moments/mean parameters of tilted distributions. Experimentally, we find that SNEP converges more stably, particularly when Monte Carlo noise is high, although convergence is slower than EP. In future, it would be interesting to develop novel convergent alternatives to EP with faster convergence, and to apply such methods to other settings where Monte Carlo estimates are used within EP. It would also be interesting to investigate relationships of SNEP to other black-box variational algorithms.

Using SNEP, we have proposed the posterior server architecture for distributed Bayesian learning using an asynchronous non-blocking message-passing protocol. The architecture uses a separate MCMC sampler on each worker, and SNEP to coordinate the samplers across the cluster so that the target distributions agree on the moments which characterise the base exponential family. In contrast with typical maximum likelihood parameter server architectures, the posterior server allows each worker to learn separate variational parameters, and as a result requires less frequent synchronisation across the cluster. We believe that this insight can allow for significant advances to distributed learning, although more work is still needed to make this reality.

We applied SNEP and the posterior server to distributed Bayesian learning of both fully-connected and convolutional neural networks, where we showed performance on par with a state-of-the-art non-distributed optimisation algorithm. While our learning setting with disjoint subsets

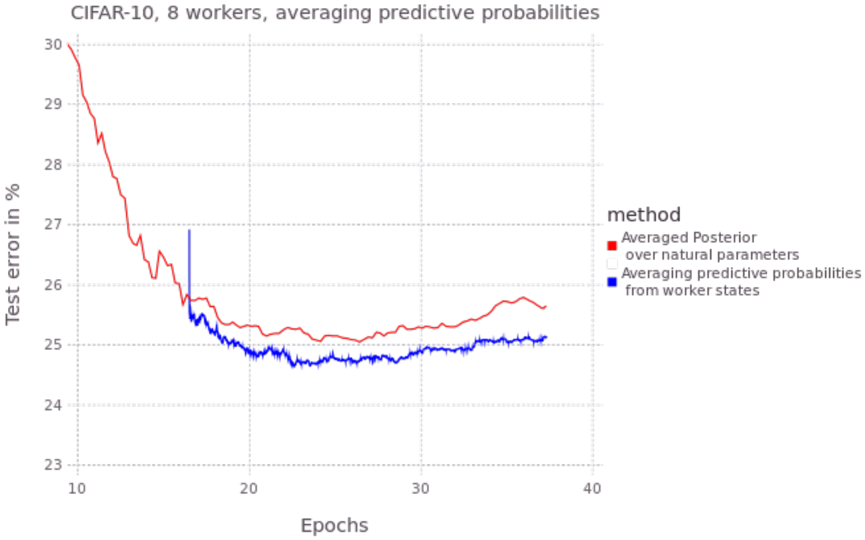

Figure 9: Learning curves for the posterior server, obtained by either a running average of the approximate posterior mean, or by averaging predictive probabilities over samples obtained by worker MCMC samplers. The averaging was started after about 15 epochs.

of data on different workers is harder, so that our current results are encouraging, it is still not satisfying that we did not produce better than state-of-the-art performance when using more computational resources. We believe that further explorations of learning regimes and hyperparameters, as well as of larger datasets, is called for, and will ultimately demonstrate the utility of a distributed Bayesian learning approach.

Acknowledgements

YWT and TL gratefully acknowledges EPSRC for research funding through grant EP/K009362/1, EP/1379622. TL also gratefully acknowledges funding from the Scatcherd European scholarship.

References

Ahmed, A., Aly, M., Gonzalez, J., Narayanamurthy, S., and Smola, A. J. (2012). Scalable inference in latent variable models. In *Proceedings of the ACM International Conference on Web Search and Data Mining (WSDM)*.

Amari, S. and Nagaoka, H. (2001). *Methods of Information Geometry*. American Mathematical Society.

Bardenet, R., Doucet, A., and Holmes, C. (2014). Towards scaling up Markov chain Monte Carlo: an adaptive subsampling approach. In *Proceedings of the International Conference on Machine Learning*.

Barthelmé, S. and Chopin, N. (2011). ABC-EP: Expectation propagation for likelihood-free Bayesian computation. In *Proceedings of the International Conference on Machine Learning.*

Battey, H., Fan, J., Liu, H., Lu, J., and Zhu, Z. (2015). Distributed estimation and inference with statistical guarantees. *ArXiv:1509.05457.*

Beal, M. J. (2003). *Variational Algorithms for Approximate Bayesian Inference.* PhD thesis, Gatsby Computational Neuroscience Unit, University College London.

Beck, A. and Teboulle, M. (2003). Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, 31(3):167–175.

Dean, J., Corrado, G., Monga, R., Chen, K., Devin, M., Mao, M., Senior, A., Tucker, P., Yang, K., Le, Q. V., and Ng, A. Y. (2012). Large scale distributed deep networks. In *Advances in Neural Information Processing Systems*, pages 1223–1231.

Dehaene, G. and Barthelmé, S. (2015). Expectation propagation in the large-data limit. *ArXiv:1503.08060.*

Dempster, A., Laird, N., and Rubin, D. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39:1–38.

Ding, N., Fang, Y., Babbush, R., Chen, C., Skeel, R. D., and Neven, H. (2014). Bayesian sampling using stochastic gradient thermostats. In *Advances in Neural Information Processing Systems.*

Doucet, A., de Freitas, N., and Gordon, N. J. (2001). *Sequential Monte Carlo Methods in Practice.* Statistics for Engineering and Information Science. New York: Springer-Verlag.

Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12.

Eslami, S. A., Tarlow, D., Kohli, P., and Winn, J. (2014). Just-in-time learning for fast and flexible inference. In *Advances in Neural Information Processing Systems.*

Gelman, A., Vehtari, A., Jylänki, P., Robert, C., Chopin, N., and Cunningham, J. P. (2014). Expectation propagation as a way of life. *ArXiv:1412.4869.*

Gilks, W. R., Richardson, S., and Spiegelhalter, D. J. (1996). *Markov Chain Monte Carlo in Practice.* Chapman and Hall.

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the International Conference on Artificial Intelligence and StatisticsInternational Conference on Artificial Intelligence and Statistics.*

Heess, N., Tarlow, D., and Winn, J. (2013). Learning to pass expectation propagation messages. In *Advances In Neural Information Processing Systems.*

Herbrich, R., Minka, T., and Graepel, T. (2007). Trueskill$^{\text{TM}}$: A Bayesian skill rating system. *Advances in Neural Information Processing Systems*, 19:569.

Hernandez-Lobato, J. M., Li, Y., Hernandez-Lobato, D., Bui, T., and Turner, R. E. (2015). Black-box α-divergence minimization. *ArXiv:1511.03243.*

Heskes, T. and Zoeter, O. (2002). Expectation propagation for approximate inference in dynamic Bayesian networks. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, volume 18.

Hinton, G. (2002). Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800.

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

Hoffman, M. D. and Gelman, A. (2014). The no-U-turn sampler: Adaptively setting path lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research*, 15.

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

Hunter, D. R. and Lange, K. (2004). A tutorial on MM algorithms. *The American Statistician.*

Jitkrittum, W., Gretton, A., Heess, N., Eslami, S. M. A., Lakshminarayanan, B., Sejdinovic, D., and Szabó, Z. (2015). Kernel-based just-in-time learning for passing expectation propagation messages. In *Proceedings of the International Conference on Uncertainty in Artificial Intelligence.*

Kingma, D. P. and Ba, J. L. (2015). Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations.*

Kleiner, A., Talwalkar, A., Sarkar, P., and Jordan, M. I. (2014). A scalable bootstrap for massive data. *Journal of the Royal Statistical Society B*, 76.

Korattikara, A., Chen, Y., and Welling, M. (2014). Austerity in MCMC land: Cutting the Metropolis-Hastings budget. In *Proceedings of the International Conference on Machine Learning.*

Krizhevsky, A. (2009). Learning multiple layers of features from tiny images. Technical report, Department of Computer Science, University of Toronto.

Leimkuhler, B. and Shang, X. (2015). Adaptive thermostats for noisy gradient systems. *arXiv:1505.06889.*

Li, Y., Hernandez-Lobato, J. M., and Turner, R. E. (2015). Stochastic expectation propagation. *ArXiv:1506.04132.*

Lienart, T., Teh, Y. W., and Doucet, A. (2015). Expectation particle belief propagation. In *Advances In Neural Information Processing Systems.*

Ma, Y.-A., Chen, T., and Fox, E. B. (2015). A complete recipe for stochastic gradient MCMC. *ArXiv:1506.04696.*

Minka, T. (2004). Power EP. Technical report, Microsoft Research.

Minka, T. P. (2001). *A Family of Algorithms for Approximate Bayesian Inference.* PhD thesis, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science.

Mnih, A. and Gregor, K. (2014). Neural variational inference and learning in belief networks. In *Proceedings of the 31st International Conference on Machine Learning*.

Neal, R. M. and Hinton, G. (1999). A view of the em algorithm that justifies incremental, sparse, and other variants. In *Learning in Graphical Models*, pages 355–368. Kluwer Academic Publishers.

Neiswanger, W., Wang, C., and Xing, E. P. (2014). Asymptotically exact, embarrassingly parallel MCMC. In *Proceedings of the International Conference on Uncertainty in Artificial Intelligence*.

Patterson, S. and Teh, Y. W. (2013). Stochastic gradient Riemannian Langevin dynamics on the probability simplex. In *Advances in Neural Information Processing Systems*.

Polyak, B. T. and Juditsky, A. B. (1992). Acceleration of stochastic approximation by averaging read more: http://epubs.siam.org/doi/abs/10.1137/0330046. *SIAM Journal of Control and Optimization*.

Ranganath, R., Gerrish, S., and Blei, D. (2014). Black box variational inference. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*.

Raskutti, G. and Mukherjee, S. (2015). The information geometry of mirror descent. *IEEE Transactions on Information Theory*, 61:1451–1457.

Rezende, D. J., Mohamed, S., and Wierstra, D. (2014). Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the International Conference on Machine Learning*.

Robbins, H. and Monro, S. (1951). A stochastic approximation method. *Annals of Mathematical Statistics*, 22(3):400–407.

Scott, S. L., Blocker, A. W., Bonassi, F. V., Chipman, H., George, E., and McCulloch, R. (2013). Bayes and big data: the consensus Monte Carlo algorithm. In *Bayes 250*.

Teh, Y. W., Thiéry, A. H., and Vollmer, S. J. (2015). Consistency and fluctuations for stochastic gradient Langevin dynamics. *Journal of Machine Learning Research*.

Tieleman, T. (2008). Training restricted Boltzmann machines using approximations to the likelihood gradient. In *Proceedings of the International Conference on Machine Learning*.

Tieleman, T. and Hinton, G. E. (2012). Lecture 6.5 RMSProp: Divide the gradient by a running average of its recent magnitude. Technical report, Coursera: Neural Networks for Machine Learning.

Vollmer, S. J., Zygalakis, K. C., and Teh, Y. W. (2016). Exploration of the (non-)asymptotic bias and variance of stochastic gradient Langevin dynamics. *Journal of Machine Learning Research*.

Wainwright, M. J. and Jordan, M. I. (2008). Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1-2):1–305.

Wang, X. and Dunson, D. B. (2013). Parallelizing MCMC via Weierstrass sampler.

Welling, M. and Teh, Y. W. (2011). Bayesian learning via stochastic gradient Langevin dynamics. In *Proceedings of the International Conference on Machine Learning*.

Wiegerinck, W. and Heskes, T. (2003). Fractional belief propagation. *Advances in Neural Information Processing Systems.*

Xu, M., Lakshminarayanan, B., Teh, Y. W., Zhu, J., and Zhang, B. (2014). Distributed Bayesian posterior sampling via moment sharing. In *Advances in Neural Information Processing Systems.*

Yedidia, J. S., Freeman, W. T., and Weiss, Y. (2001). Generalized belief propagation. In *Advances in Neural Information Processing Systems*, volume 13, pages 689–695.

Yuille, A. (2002). CCCP algorithms to minimize the Bethe and Kikuchi free energies: Convergent alternatives to belief propagation. *Neural Computation*, 14(7):1691–1722.

Zhang, S., Choromanska, A., and LeCun, Y. (2015). Deep learning with elastic averaging SGD. *Advances in Neural Information Processing Systems.*

Zhang, Y., Duchi, J. C., and Wainwright, M. J. (Accepted). Divide and conquer kernel ridge regression: A distributed algorithm with minimax optimal rates. *Journal of Machine Learning Research.*

Zhao, T., Cheng, G., and Liu, H. (2014). A partially linear framework for massive heterogeneous data. *ArXiv:1410.8570.*