Tools for sparse Bayesian deep learning

Yves Atchadé

Boston University

joint work with:
Liwei Wang, Boston University
Introduction

1. deep learning (DL) models have tremendous approximation power. But estimation (training) requires lot of data.
2. In data-poor areas, domain knowledge and sparsity may help.

```
\begin{center}
\begin{tikzpicture}
\begin{axis}[
    width=0.8\textwidth,
    height=0.5\textwidth,
    xlabel={Sparsity [%]},
    ylabel={ImagNet Validation Accuracy [%]},
    xmin=40, xmax=100,
    ymin=45, ymax=75,
    legend pos=north west,
    \addplot+[mark=*,mark options={solid},color=blue] table {data.txt};
    \addplot+[mark=*,mark options={solid},color=red] table {data2.txt};
    \legend{Accuracy, Performance}
\end{axis}
\end{tikzpicture}
\end{center}
```

Fig. 4. Typical test error vs. sparsity showing Occam’s hill (network: ResNet-50 on Top-1 ImageNet).

3. The talk discusses two ideas towards that goal: Cyclical MCMC and asynchronous MCMC.
Cyclical MCMC

Asynchronous MCMC

Experimentation with deep learning models
Cyclical MCMC

1. 'Annealing' / 'tempering'. Let $\mathcal{E} : \mathbb{X} \rightarrow \mathbb{R}$ with minimum set $\mathcal{M}$. Set
   \[ \pi_t(x) \propto \exp(-\beta_t \mathcal{E}(x)), \quad \beta_t > 0. \]

2. As $\beta_t \uparrow \infty$, $\pi_t(\cdot) \approx \pi_\infty(\cdot) = \frac{|\cdot \cap \mathcal{M}|}{|\mathcal{M}|}$. 
1. Combined with the Metropolis algorithm and we get Simulated Annealing (SA)

2. For well-designed nonhom. Markov chain \( \{X_t, \ t \geq 0\} \) with kernels \( \{P_t, \ t \geq 0\} \) with \( \pi_t P_t = \pi_t \), and well-chosen sequence \( \beta_t \),

\[
P(X_t \in \cdot) - \pi_t(\cdot) \approx 0.
\]
Cyclical MCMC

1. It became quickly clear to the MCMC pioneers that the idea behind SA can be used also to sample from a distribution of interest $\pi$ by annealing up to 1.

2. Led to parallel tempering (PT) that targets

$$\bar{\pi}(x_1, \ldots, x_K) \propto \prod_{k=1}^{K} \pi(x_k)^{\beta_k}.$$ 

3. And simulated tempering (ST) that targets

$$\pi(k, x) \propto \exp\left(-\beta_k \mathcal{E}(x)\right) / c_k.$$
Cyclical MCMC

1. Unlike SA which remains a mysterious metaheuristics with some theoretical backing, PT and ST benefits from the rigor of MC theory.

2. However these algorithm come with a higher computational price. Costly to use for DL.

3. With Cyclical MCMC, we go back to the original SA framework.
1. Let $\beta : [0, 1] \to \mathbb{R}$ such that $\beta_0 = \beta_1 = 1$, $\beta_t \downarrow \uparrow$.

2. We extend $t \mapsto \beta_t$ to $\mathbb{R} \to \mathbb{R}$ by periodic extension.

3. Let $\pi(x) \propto e^{-\mathcal{E}(x)}$ a density of interest. For $k \geq 0$, we define

$$
\pi_k(x) \propto \exp\left(-\beta_{(k/L)}\mathcal{E}(x)\right).
$$

4. Cyclical: $\pi_{k+jL} = \pi_k$. 
1. Let $P_k$ be a Markov kernel with invariant distribution $\pi_k$.
2. The Cyclical MCMC sampler is a nonhomog. Markov chain $\{X_k, \ k \geq 0\}$ with sequence of transition kernels $\{P_k, \ k \geq 1\}$.
3. We collect samples at times $jL, \ j = 0, 1, \ldots$. 

Cyclical MCMC
Cyclical MCMC

1. The Cyclical MCMC sampler is a nonhomog. Markov chain $\{X_k, k \geq 0\}$ with sequence of transition kernels $\{P_k, k \geq 1\}$.

2. By periodicity, its can also be viewed as a homogeneous MC $\{X_{jL}, j \geq 0\}$ with transition kernel

$$P_1 \times \cdots \times P_L.$$

3. Intuition: for well-chosen $\beta$, $K$ has very good mixing: for $1 \leq \ell \leq L$:

$$\{P_1 \times \cdots \times P_\ell\} (x, \cdot) - \pi_\ell(\cdot) \approx 0.$$


5. Computationally the algorithm is very efficient.
Cyclical MCMC: illustration

1. 
\[ \pi(x) = \frac{1}{25} \sum_{i=1}^{25} \mathcal{N}(x|\mu_i, \Sigma). \]

2. We compare MaLa, cyclical MaLa, SGLD, cyclical SGLD.

3. 150,000 total iterations split into 300 cycles. Collect samples at end of cycles.

<table>
<thead>
<tr>
<th>sampler</th>
<th>MALA</th>
<th>cMALA</th>
<th>ULA</th>
<th>cULA</th>
</tr>
</thead>
<tbody>
<tr>
<td>SD</td>
<td>29.52 ± 6.89</td>
<td>4.75 ± 0.54</td>
<td>29.11 ± 7.44</td>
<td>4.57 ± 0.34</td>
</tr>
</tbody>
</table>

Table: standard deviation of number of samples in each mode
Cyclical MCMC: illustration

Figure: scatter plots of different method in 25 gaussian mixtures
1. On-going work. The cosine cycles works well. But cycle lengths requires careful tuning.
2. We need more theory.
Cyclical MCMC

Asynchronous MCMC

Experimentation with deep learning models
Asynchronous MCMC for Bayesian sparse deep learning

▶ The Gibbs sampler is a hallmark of MCMC methods.
▶ A density $\pi(x_1, x_2)$ on $X = X_1 \times X_2$.
▶ Let $\pi_1(\cdot|x_2)$ and $\pi_2(\cdot|x_1)$ the two conditional distributions.

Algorithm (Gibbs Sampler)

1. At the $k$-th iteration, given $X^{(k)} = (X_1^{(k)}, X_2^{(k)}) = (x_1, x_2)$.
   1.1 Draw $\bar{X}_1 \sim \pi_1(\cdot|x_2)$, and then draw $\bar{X}_2 \sim \pi_2(\cdot|\bar{X}_1)$.
2. Set $X^{(k+1)} = (\bar{X}_1, \bar{X}_2)$.

▶ Asynchronous Gibbs sampler is a modification of the Gibbs sampler where new random draws are not automatically broadcast.
Asynchronous MCMC was first introduced to the best of my knowledge in the 80’s in the CS community as a way of speeding up simulated annealing.

Asynchronous MCMC has resurfaced recently in machine learning.

Asynchronous MCMC for Bayesian sparse deep learning

- Asynch. Gibbs sampling does not maintain the correct invariant distribution.
- For $a \in [0, 1]$, suppose that $X = \{0, 1\} \times \{0, 1\}$, and

  $\pi(0, 0) = 0$, $\pi(0, 1) = \pi(1, 0) = \frac{1-a}{2}$, $\pi(1, 1) = a$.

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>(1-a)/2</td>
</tr>
<tr>
<td>1</td>
<td>(1-a)/2</td>
<td>a</td>
</tr>
</tbody>
</table>

- If $\tilde{X}^{(k)} = (1, 1)$,

  $$
  \mathbb{P}\left(X^{(k+1)} = (0, 0) | X^{(k)} = (1, 1) \right) = \left(\frac{1-a}{1+a}\right)^2,
  $$

  which will produce a biased sampling asymptotically.
Asynchronous MCMC for Bayesian sparse deep learning

Figure: Gibbs sampler versus asynchronous Gibbs sampler for $a = 0.8$ and $a = 0.1$.

- The bias is essentially
  \[ \| \pi_{1|2}(\cdot|1) - \pi_{1|2}(\cdot|0) \|_{tv} = (1 - a)/(1 + a). \]
- De Sa et al. (2016) formalized this using Dobrushin coefficient.
Suppose we have a log-likelihood function

\[ \ell(\theta) = \ell(\theta, D) = \sum_{i=1}^{n} f_{\theta}(z_i), \; \theta \in \mathbb{R}^p. \]

We use a spike and slab prior for \( \theta \): for \( u > 1 \), \( 0 < \rho_1 < \rho_0 < \infty \):

\[ \delta_j \sim \text{Ber}(p^{-u}), \; \theta_j|\delta \overset{d}{=} \theta_j|\delta_j \overset{\text{ind}}{\sim} \begin{cases} \mathcal{N}(0, \rho_1^{-1}) & \text{if } \delta_j = 1 \\ \mathcal{N}(0, \rho_0^{-1}) & \text{if } \delta_j = 0 \end{cases} \]
Asynchronous MCMC for Bayesian sparse deep learning

The posterior distribution can be written as

\[ \Pi(\delta, \theta|\mathcal{D}) \propto \left( p^u \sqrt{\frac{\rho_1}{\rho_0}} \right)^{-\|\delta\|_0} \exp \left( -\frac{\rho_0}{2} \|\theta - \theta_\delta\|^2_2 - \frac{\rho_1}{2} \|\theta_\delta\|^2_2 + \ell(\theta_\delta) \right) \]

- Asynchronous MCMC algorithm for \( \Pi \):
  1. fix \( \delta \) and update \( \theta \) (using SGLD or standard MCMC update);
  2. fix \( \theta \) and update \( J \) components of \( \delta \) (using asynchronous Gibbs).
Asynchronous MCMC for Bayesian sparse deep learning

Why should asynchronous update work here?

▶ We have

\[ \Pi_j(\delta_j|\delta_{-j}, \theta, \mathcal{D}) \sim \text{Ber}(q_j), \]

where \( q_j \) is driven mainly by \( u \log(p) \) and the log-likelihood ratio

\[ \ell(\theta_{\delta(j,0)}) - \ell(\theta_{\delta(j,1)}) \approx -\theta_j \nabla_j \ell(\theta_{\delta(j,0)}) - \frac{\theta_j^2}{2} \nabla_{jj}^{(2)} \ell(\bar{\theta}). \]

▶ To illustrate, assume logistic regression.

\[
\nabla_j \ell(\theta) = \sum_{i=1}^{n} \left( Y_i - \frac{e^{\langle \theta, x_i \rangle}}{1 + e^{\langle \theta, x_i \rangle}} \right) x_{ij} = \sum_{i=1}^{n} \left( Y_i - \frac{e^{\langle \theta_{*}, x_i \rangle}}{1 + e^{\langle \theta_{*}, x_i \rangle}} \right) x_{ij} \\
- (\theta_j - \theta_{*j}) \sum_{i=1}^{n} D_i(\bar{\theta}) x_{ij}^2 - \sum_{k \neq j} (\theta_k - \theta_{*k}) \sum_{i=1}^{n} D_i(\bar{\theta}) x_{ij} x_{ik}.
\]
Algorithm (Asynch. Sparse SGLD (AS-SGLD))

1. fix $\delta$ and update $\theta$ (using Stochastic Gradient Langevin dynamics – SGLD);

2. Given $\theta$, select $J$ components, for $\vartheta$, and compute $G = \nabla \ell(\theta_{\vartheta})$. Draw independently

$$
\delta_{J_k} \sim \text{Ber}(q_{J_k}), \quad q_{J_k} = \left(1 + \frac{e^{a_0(\theta_{J_k})}}{e^{a_1(\theta_{J_k})} e^{-\theta_{J_k} G_{J_k}}} \right)^{-1}.
$$
Approximate correctness for linear regression

\[ \ell(\theta) = -\frac{1}{2\sigma^2} \| y - X\theta \|_2^2, \quad \theta \in \mathbb{R}^p, \quad \sigma^2 > 0 \quad \text{known.} \quad (1) \]

**Theorem**

*Under classical high dim. lin. regr. assumptions and*

\[ n \gtrsim \max \left( \theta_\star^{-2} (1 + s_\star^3) \log(p), \ J^2 \log(p), \ (\log(p))^3 \right), \]

*and \ u \geq C_2 (1 + s_\star)^2, \quad (2)*

\[
\mathbb{E}_\star \left[ \max_{j: \delta_{\star j} = 1} \left| \mathbb{P}(\delta^{(k)}_j = 1) - \Pi(\delta_j = 1|\mathcal{D}) \right| \right] \\
\leq \left( 1 - \frac{3}{10} \frac{J}{p} \right)^k + \exp \left( -C_3 \theta_\star \sqrt{n} + C_4 J \sqrt{\log(p)} \right) + \frac{10}{p}.
\]

*with probability at least 1 - 10/p (over the data).*
Logistic regression

Figure: Relative error for logistic regression model. Based on 50 replications.
### Logistic regression

<table>
<thead>
<tr>
<th>p/n</th>
<th>Complexity/iteration</th>
<th>1000/500</th>
<th>2000/1000</th>
<th>5000/2500</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exact</td>
<td>$O(nJ|\delta^{(k)}|_0)$</td>
<td>5.25s</td>
<td>35.13s</td>
<td>1360.09s</td>
</tr>
<tr>
<td>Asyn</td>
<td>$O(n(|\delta^{(k)}|_0 + J))$</td>
<td>0.71s</td>
<td>2.19s</td>
<td>99.04s</td>
</tr>
<tr>
<td>SA-SGLD</td>
<td>$O(B(|\delta^{(k)}|_0 + J))$</td>
<td>0.24s</td>
<td>1.44s</td>
<td>30.12s</td>
</tr>
<tr>
<td>Skinny-Gibbs</td>
<td>$O(n(p \lor |\delta^{(k)}|_0^2))$</td>
<td>10.50s</td>
<td>87.27s</td>
<td>1154.40s</td>
</tr>
<tr>
<td>VA</td>
<td>$O(B \cdot J \cdot p)$</td>
<td>4.05s</td>
<td>34.42s</td>
<td>1243.82s</td>
</tr>
</tbody>
</table>

**Table**: Running times to convergence
Cyclical MCMC

Asynchronous MCMC

Experimentation with deep learning models
Experimentation with deep learning models

- Lenet-5 and a baby VGG-16 architectures.
- Using MNIST-FASHION and Cifar-10 datasets.
- The goal is to classify small images.
Experimentation with deep learning models

Accuracy/Sparsity of Lenet-5 on fashion-MNIST

Accuracy/Sparsity of Lenet-5 on cifar10

Accuracy/Sparsity of VGG on cifar10
Experimentation with deep learning models

Sparsity of each layer in Lenet-5(left) and VGG(right)

![Sparsity of each layer in Lenet-5(left) and VGG(right)]
Experimentation with deep learning models

<table>
<thead>
<tr>
<th>Method</th>
<th>Accuracy</th>
<th>Density</th>
</tr>
</thead>
<tbody>
<tr>
<td>SGD with Momentum</td>
<td>0.764</td>
<td>1</td>
</tr>
<tr>
<td>SGLD</td>
<td>0.8029</td>
<td>1</td>
</tr>
<tr>
<td>cSGLD</td>
<td>0.8042</td>
<td>1</td>
</tr>
<tr>
<td>plain SA-SGLD, $u = 50$</td>
<td>0.727</td>
<td>0.0047</td>
</tr>
<tr>
<td>SA-cSGLD, $u = 50$</td>
<td>0.758</td>
<td>0.0065</td>
</tr>
<tr>
<td>SA-SGLD, 10 chains, $u = 50$</td>
<td>0.745</td>
<td>0.0058</td>
</tr>
</tbody>
</table>

Table: VGG-6 with Cifar-10 dataset
Experimentation with deep learning models

We compare

$$\text{Ent} \left( p_W (\cdot | x) \right), \quad \text{and} \quad \text{Ent} \left( \int p_W (\cdot | x) \Pi (dW | D) \right).$$
Concluding thoughts

- We have presented two approximate MCMC ideas that we have found very useful for large scale sparse Bayesian modeling.
- Particularly in low-data and noisy-data settings.
- More theoretical analysis is needed.
- In the context of DL, software and hardware to take advantage of sparsity is also needed.

Thanks!!