Bayesian neural learning via Langevin dynamics for chaotic time series prediction

Dr. Rohitash Chandra, Lamiae Azizi, and Sally Cripps

1. Centre for Translational Data Science
http://sydney.edu.au/data-science
2. Discipline of Business Analytics
School of Business

THE UNIVERSITY OF SYDNEY

International Conference on Neural Information Processing
Ramada Pearl Hotel, Guangzhou, China
www.iconip2017.org

Overview

Background

- Despite the successes, there are some challenges in training neural networks with backpropagation.
- Firstly, with large data sets, finding the optimal values of hyper-parameters (eg. learning and momentum rate) and weights take a large amount of time.
- Secondly, through canonical backpropagation, we can only obtain point estimates of the weights in the network. As a result, these networks make predictions that do not account for uncertainty in the parameters.

Background and Motivation

- Bayesian methods naturally account for uncertainty in parameter estimates and can propagate this uncertainty into predictions.
- Bayesian neural networks (BNNs) use Markov Chain Monti-Carlo (MCMC) methods such as those based on e.g the Laplace approximation, Hamiltonian Monte Carlo, expectation propagation and variational inference.

Background and Motivation

- Langevin Dynamics refer to a class of MCMC algorithms that incorporate gradients with Gaussian noise in parameter updates. In the case of neural networks, the parameter updates refer to the weights of the network.
- We provide the synergy of MCMC random-walk algorithm and gradient descent via backpropagation neural network. This is referred as Langevin dynamics MCMC for training neural networks.
- We employ six benchmark chaotic time series problems to demonstrate the effectiveness of the proposed method. Furthermore, comparison is done with standalone gradient descent and the MCMC random-walk algorithm.

Overview

Let y_t denote a univariate time series modelled by:

$$y_t = f(\mathbf{x}_t) + \epsilon_t, \text{ for } t = 1, 2, \ldots, n \tag{1}$$

where $f(\mathbf{x}_t) = E(y_t|\mathbf{x}_t)$, is an unknown function, $\mathbf{x}_t = (y_{t-1}, \ldots, y_{t-D})$ is a vector of lagged values of y_t, and ϵ_t is the noise with $\epsilon_t \sim \mathcal{N}(0, \tau^2)\ \forall t$.

Embedding time series

In order to use neural networks for time series prediction, the original dataset is constructed into a state-space vector through Taken's theorem which is governed by the embedding dimension (D) and time-lag (T).
Define

$$\mathcal{A}_{D,T} = \{t; t > D, \quad \text{mod}\ (t - (D+1), T) = 0\} \tag{2}$$

Let $\mathbf{y}_{\mathcal{A}_{\mathcal{D},\mathcal{T}}}$ to be the collection of y_t's for which $t \in \mathcal{A}_{\mathcal{D},\mathcal{T}}$, then, $\forall t \in A_{D,T}$, we compute the $f(\mathbf{x}_t)$ by a feedforward neural network with one hidden layer defined by the function

Priors

To conduct a Bayesian analysis, we need to specify prior distributions for the elements of θ which we choose to be

$$\begin{aligned} v_h &\sim \mathcal{N}(0, \sigma^2) \text{ for } h = 1, \ldots, H, \\ \delta_0 &\sim N(0, \sigma^2) \\ \delta_h &\sim N(0, \sigma^2) \\ w_{dj} &\sim \mathcal{N}(0, \sigma^2) \text{ for } h = 1, \ldots, H \text{ and } d = 1, \ldots, D, \\ \tau^2 &\sim \mathcal{IG}(\nu_1, \nu_2) \end{aligned} \tag{3}$$

where H is the number of hidden neurons. In general the log posterior is

$$\log\left(p(\boldsymbol{\theta}|\mathbf{y})\right) = \log\left(p(\boldsymbol{\theta})\right) + \log\left(p(\mathbf{y}|\boldsymbol{\theta})\right)$$

Log likelihood

In our particular model the log likelihood is

$$\log\left(p(\mathbf{y}_{\mathcal{A}_{\mathbf{D},\mathbf{T}}}|\boldsymbol{\theta})\right) = -\frac{n-1}{2}\log(\tau^2) - \frac{1}{2\tau^2}\sum_{t\in\mathcal{A}_{\mathcal{D},\mathcal{T}}}(y_t - E(y_t|\mathbf{x}_t))^2 \tag{4}$$

where $E(y_t|\mathbf{x}_t)$ is given below:

$$f(\mathbf{x}_t) = g\left(\delta_o + \sum_{h=1}^{H} v_j g\left(\delta_h + \sum_{d=1}^{D} w_{dh} y_{t-d}\right)\right) \tag{5}$$

LD-MCMC Algorithm

Alg. 1 Langevin Dynamics for neural networks

Data: Univariate time series $\mathbf{y}$
Result: Posterior of weights and biases $p(\boldsymbol{\theta}|\mathbf{y})$
Step 1: State-space reconstruction $\mathbf{y}_{\mathcal{A}_{D,T}}$ by Equation 2
Step 2: Define feedforward network as given in Equation 3
Step 3: Define $\boldsymbol{\theta}$ as the set of all weights and biases
Step 4: Set parameters σ^2 ,ν_1, ν_2 for prior given in Equation 6

for *each* k *until max-samples* **do**
  1.Compute gradient $\Delta\boldsymbol{\theta}^{[k]}$given by Equation 8
  2. Draw $\boldsymbol{\eta}$ from $\mathcal{N}(0, \Sigma_\eta)$
  3. Propose $\boldsymbol{\theta}^* = \theta^{[k]} + \Delta\boldsymbol{\theta}^{[k]} + \boldsymbol{\eta}$
  4. Draw from uniform distribution $u \sim \mathcal{U}[0, 1]$
  5. Obtain acceptance probability α given by Equation 9
  if $u < \alpha$ **then**
    $\boldsymbol{\theta}^{[k+1]} = \boldsymbol{\theta}^*$
  end
  else
    $\boldsymbol{\theta}^{[k+1]} = \boldsymbol{\theta}^{[k]}$
  end
end

Design of Experiments

- Experimental evaluation of LD-MCMC algorithm (Algorithm 1) for six benchmark one-step-ahead chaotic time series prediction problems. The comparison is done with MCMC random-walk algorithm and gradient descent (backpropagation).
- The benchmark problems employed are Mackey-Glass, Lorenz, Sunspot, and Laser, Henon, and Rossler time series. Takens' embedding theorem is applied with selected values as follows. $D = 4$ and $T = 2$ is used for reconstruction of the respective time series into state-space vector.
- All the problems used first 1000 data points of the time series from which 60 % was used for training and 40 % for testing the method.

Design of Experiments

- Note that (*) represents early convergence or termination criteria which is 2000 epochs for GD*. On the other hand, GD convergence is defined by 80 000 iterations (epochs) in order to compare with MCMC random-walk algorithm which draws 80 000 samples.
- LD-MCMC draws 40 000 samples and each of them are refined by GD for 1 iteration which makes the computation time similar in terms of number of fitness evaluation given by the loss function.
- Note that GD method employed 30 independence experimental runs, while MCMC and LD-MCMC results are for a single experimental run. The mean and standard deviation (std) of the results are given.

Results

Table: Results for the respective methods

Problem	Method	Train (mean)	Train (std)	Test (mean)	Test(std)	% Accepted
Lazer	GD*	0.02191	0.00129	0.02675	0.00085	-
	GD	0.01035	0.00162	0.01732	0.00142	-
	MCMC	0.02549	0.00837	0.02643	0.00718	9.70
	LD-MCMC	0.01658	0.00211	0.02280	0.00351	57.86
Sunspot	GD*	0.02117	0.00160	0.02359	0.00209	-
	GD	0.00775	0.00026	0.00975	0.00031	-
	MCMC	0.01466	0.00174	0.01402	0.00178	4.55
	LD-MCMC	0.01155	0.00167	0.01090	0.00146	45.89
Mackey	GD*	0.00590	0.00026	0.00669	0.00029	-
	GD	0.00286	0.00025	0.0033	0.00026	-
	MCMC	0.00511	0.00058	0.00520	0.00058	1.74
	LD-MCMC	0.00615	0.00091	0.00627	0.00091	30.35

Results

Table: Results for the respective methods

Problem	Method	Train (mean)	Train (std)	Test (mean)	Test(std)	% Accepted
Lorenz	GD*	0.01570	0.00056	0.01608	0.00052	-
	GD	0.00400	0.00024	0.00460	0.00026	-
	MCMC	0.00813	0.00174	0.00713	0.00150	2.74
	LD-MCMC	0.00890	0.00211	0.00821	0.00207	26.95
Rossler	GD*	0.01570	0.00056	0.01608	0.00052	-
	GD	0.00281	0.00029	0.00462	0.00045	-
	MCMC	0.01371	0.00291	0.01355	0.00297	3.69
	LD-MCMC	0.00722	0.00121	0.00692	0.00120	35.53
Henon	GD*	0.01366	0.00033	0.01778	0.00025	-
	GD	0.00555	0.00029	0.00604	0.00025	-
	MCMC	0.03256	0.03920	0.03127	0.03850	8.71
	LD-MCMC	0.00948	0.00112	0.00912	0.00114	27.17

Results

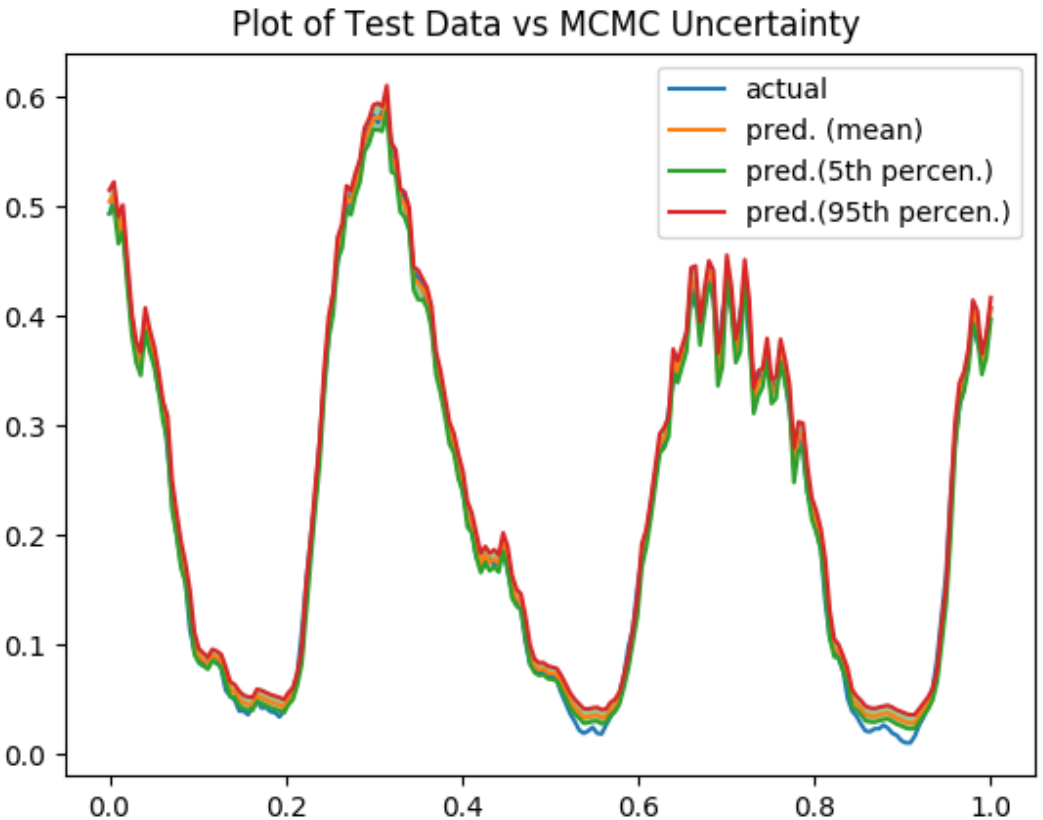

Figure: Results for Sunspot time series using LD-MCMC algorithm. Note that the x-axis represents the time in year while y-axis gives the Sunspot index.

Results

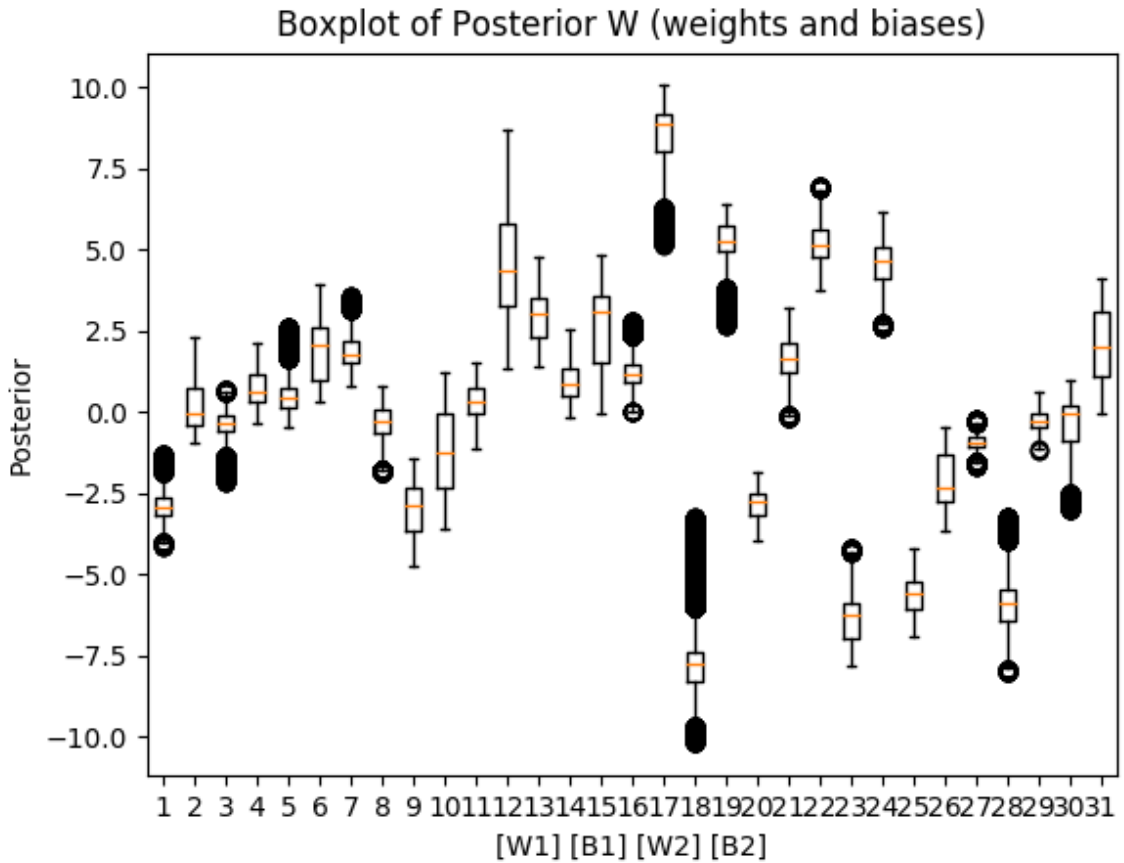

Figure: Posterior of weights and biases. W1 refers to the set of weights from input-hidden layer, W2 refers to the set of weights from hidden-output layer, B1 refers to the set of biases of the hidden layer and B2 refers to the bias of the output layer.

Discussion

- The results in general have shown that LD-MCMC further improves the performance of MCMC random-walk for training FNNs for majority of the problems.
- This improvement has been through the incorporation of gradients in weight updates via Langevin dynamics.
- This has been beneficial for improving the proposals drawn by MCMC random-walk algorithm. In the case for larger datasets, stochastic gradient version of Langevin dynamics can be implemented.

Conclusions and Future Work

- We applied Langevin dynamics for Bayesian learning in neural networks that featured synergy of MCMC with backpropagation. This provides further advantage to conventional neural network training algorithms through uncertainty quantification in predictions.
- A limitation is that the proposed method cannot get the same performance when compared to gradient decent alone when given a large number of training epochs that match with number of samples used by the MCMC algorithms.
- The incorporation of Langevin dynamics with hyper-parameters such as momentum and learning rate could be further explored. In addition, uncertainty quantification could also be provided in terms of number of hidden neurons and layers of the network.

Thank You

More information: https://rohitash-chandra.github.io