Introduction

Objective

- Weight uncertainty of deep neural networks (DNNs): posterior inference of weight distributions
- Traditional MCMC was popular in CV a decade ago, including Gibbs sampling, HMC, MH, etc; but NOT scalable
- Scale up with Stochastic Gradient Markov chain Monte Carlo (SG-MCMC)

Main Contributions

- We provide insights on the interpretation of Dropout from the perspective of SG-MCMC, which also allows the use of Batch-Normalization.
- Applications to a wide range of shape classification problems demonstrate the advantages of SG-MCMC over optimization.

Illustrations

Weight Uncertainty in DNNs

- Weight with a fixed value
- Weight with a distribution

Figures: Illustration of Bayesian DNNs with a 2-layer model. All weights in Bayesian DNNs are represented as distributions using SG-MCMC (right figure); rather than having fixed values (left figure), as provided by classical stochastic optimization methods. The SG-MCMC learns correlated uncertainty jointly on all parameters, where (right) associated marginal distributions are depicted.

Basic SG-MCMC algorithm

SGLD: Stochastic Gradient Langevin Dynamics [1]

Algorithm 1: SGLD algorithm

1. Initialize: Random $\theta_0$;
2. for $t = 1, 2, \ldots, T$ do
   1. Estimate gradient from minibatch $S_t$:
   2. $\dot{\theta}_t \leftarrow \nabla \log p(\theta_t) + \frac{N}{|S_t|} \sum_{i \in S_t} \nabla \theta \log p(d_i|\theta_t)$;
3. Parameter update:
   1. $\xi_t \sim \mathcal{N}(0, \epsilon I)$;
   2. $\theta_{t+1} \leftarrow \theta_t + \frac{1}{2} \dot{\theta}_t + \xi_t$;
end

where $\epsilon$ is step-size, and $S_t$ is the mini-batch

More SG-MCMC algorithms

Table: SG-MCMC algorithms and their optimization counterparts. Algorithms in the same row share similar characteristics.

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>SG-MCMC</th>
<th>Optimization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic SGLD</td>
<td>SGLD</td>
<td>SGD</td>
</tr>
<tr>
<td>Preconditioning</td>
<td>pSGLD [2]</td>
<td>RMSprop/Adam</td>
</tr>
<tr>
<td>Momentum</td>
<td>SG-MCMC</td>
<td>momentum SGD</td>
</tr>
<tr>
<td>Thermostat</td>
<td>SGNHT</td>
<td>Santa [3]</td>
</tr>
</tbody>
</table>

Understanding Dropout

On the connection of SGLD and Dropout

For neural networks with the nonlinear function $q(\cdot)$ and consecutive layers $h_{1}$ and $h_{2}$, dropout and dropConnect:

- Dropout: $h_2 = \xi_0 \odot q(\theta h_1)$,
- DropConnect: $h_2 = q(\xi_0 \odot \theta) h_1$,

where the injected noise $\xi_0$ can be binary valued with dropping rate $p$ or its equivalent Gaussian:

- Binary noise: $\xi_0 \sim \text{Ber}(p)$,
- Gaussian noise: $\xi_0 \sim \mathcal{N}(0, \frac{1 - p^2}{p})$.

By combining dropConnect and Gaussian noise:

$h_{t+1} = \xi_t \odot \theta_t - \frac{\gamma}{2} F_t + \theta_t - \frac{\gamma}{2} F_t + \xi_t$

where $\xi_t \sim \mathcal{N}(0, \frac{1 - p^2}{p})$.

The predicted distribution of $\hat{y}$ may be viewed in terms of model averaging across parameters, based on the learned $p(\theta|D)$; this should be contrasted with learning a single point estimate of $\theta$ based on $D$.

Integration of SG-MCMC and Binary Dropout

- Accelerating SG-MCMC using Batch-Normalization

Experiments: Shape Classification

I. Datasets

<table>
<thead>
<tr>
<th>Method</th>
<th>FNN</th>
<th>CNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>2D</td>
<td>MNIST, Animal, 20 Newsgroups</td>
<td>MNIST, Caltech, Cifar10</td>
</tr>
<tr>
<td>3D</td>
<td>Body Shape, Textured Shape</td>
<td>ModelNet</td>
</tr>
</tbody>
</table>

Hand-crafted features as input of FNN, while raw data for CNN

II. Results and Observations

A thorough comparison

Accuracy of FNN on MNIST using a two-layer network (X-X) with ReLU. Please refer to the paper for a lot more results.

Networks: FNN, CNN

Methods: Test Error (%)

- pSGLD + Dropout: 1.36
- SGLD + Dropout: 1.45
- RMSprop + Dropout: 1.35
- SGD + Dropout: 1.51
- pSGLD: 1.45
- SGLD: 1.44
- RMSprop: 1.79
- SGD: 1.72
- RMSprop + BN: 1.65
- SGD + BN: 1.65
- pSGLD + BN: 2.04
- SGLD + BN: 1.32
- RMSprop + D + BN: 1.32
- SGD + D + BN: 1.32
- pSGLD + D + BN: 1.32
- SGLD + D + BN: 1.32

- The testing error for the SG-MCMC methods are consistently lower than their corresponding stochastic optimization counterparts. This indicates that the weight uncertainty learned via SG-MCMC can improve performance.
- Both of Dropout and SG-MCMC slow down learning initially. By integrating SG-MCMC with Dropout, we obtain lower error.

Very Deep Neural Networks

- Dropout: BN: Batch Normalization
- The use of SG-MCMC or Dropout slows down learning initially. This is likely due to the higher uncertainty imposed during learning, resulting in more exploration of the parameter space.
- Increased uncertainty, however, prevents overfitting and eventually results in improved performance.
- Extensive empirical results on adding gradient noise are also shown in [4].

References


Acknowledgements

This research was supported by ARO, DARPA, DOE, NGA ONR and NSF.