Learning Weight Uncertainty with Stochastic Gradient MCMC for Shape Classification

Chunyuan Li, Andrew Stevens, Changyou Chen, Yunchen Pu, Zhe Gan, Lawrence Carin

Duke University, Durham NC 27708, USA

Introduction

Objective

1. Weight uncertainty of deep neural networks (DNNs): posterior inference of weight distributions
2. Bring MCMC back to the community of computer vision to tackle "big visual/geometric data"
   - Traditional MCMC: was popular in CV a decade ago, inclduing Gibbs sampling, HMC, MH, etc; but NOT scalable
   - Scale up with Stochastic Gradient Markov chain Monte Carlo (SG-MCMC)

Main Contributions

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

Illustrations

Weight Uncertainty in DNNs

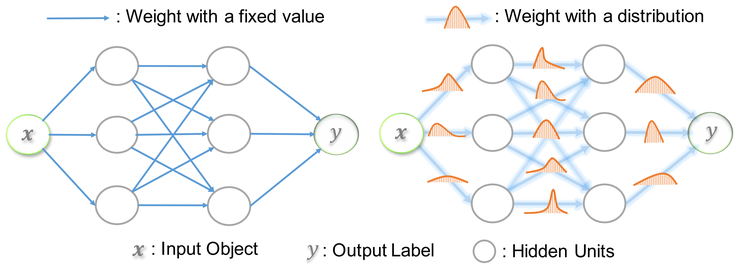

Figure: Illustration of Bayesian DNNs with a 2-layer model. All weights in Bayesian DNNs are represented as distributions using SG-MCMC (right figure); rather than having fixed values (left figure), as provided by classical stochastic optimization methods. The SG-MCMC learns correlated uncertainty jointly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

- Given data $\mathcal{D} = \{\boldsymbol{d}_i\}_{i=1}^N$, $\boldsymbol{d}_i$ is *i.i.d.*; model parameters $\boldsymbol{\theta}$

$$\underbrace{p(\boldsymbol{\theta}|\mathcal{D})}_{\text{Posterior}} \propto \underbrace{p(\boldsymbol{\theta})}_{\text{Prior}} \underbrace{\Pi_{i=1}^N p(\boldsymbol{d}_i|\boldsymbol{\theta})}_{\text{Likelihood}}$$

  For DNNs, $\boldsymbol{d}_i \triangleq (\boldsymbol{x}_i, y_i)$: input $\boldsymbol{x}_i \in \mathbb{R}^D$ and output $y_i \in \mathcal{Y}$.
- Bayesian predictive estimate, for testing input $\tilde{\boldsymbol{x}}$

$$p(\tilde{y}|\tilde{\boldsymbol{x}}, \mathcal{D}) = \mathbb{E}_{p(\boldsymbol{\theta}|\mathcal{D})}[p(\tilde{y}|\tilde{\boldsymbol{x}}, \boldsymbol{\theta})] \approx \tfrac{1}{T}\Sigma_{t=1}^T p(\tilde{y}|\tilde{\boldsymbol{x}}, \boldsymbol{\theta}_t) \quad (1)$$

- In optimization, $\boldsymbol{\theta}_{MAP} = \operatorname{argmax} \log p(\boldsymbol{\theta}|\mathcal{D})$.
  The MAP approximates this expectation as

$$p(\tilde{y}|\tilde{\boldsymbol{x}}, \mathcal{D}) \approx p(\tilde{y}|\tilde{\boldsymbol{x}}, \boldsymbol{\theta}_{\text{MAP}}) \quad (2)$$

Parameter uncertainty is ignored.

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

Algorithms

Basic SG-MCMC algorithm

SGLD: Stochastic Gradient Langevin Dynamics [1]

Algorithm 1: SGLD algorithm

Initialize:Random $\boldsymbol{\theta}_1$;
for $t = 1, 2, \ldots, T$ **do**
  %Estimate gradient from minibatch $\mathcal{S}_t$
  $\tilde{\boldsymbol{g}}_t \leftarrow \nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}_t) + \frac{N}{|\mathcal{S}_t|} \sum_{i \in \mathcal{S}_t} \nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{d}_i|\boldsymbol{\theta}_t)$;
  %Parameter update
  $\boldsymbol{\xi}_t \sim \mathcal{N}(0, \epsilon\mathbf{I})$;
  $\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t + \frac{\epsilon}{2}\tilde{\boldsymbol{g}}_t + \boldsymbol{\xi}_t$;
end

where ϵ is step-size, and $\mathcal{S}_t$ is the mini-batch

More SG-MCMC algorithms

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

Algorithms	SG-MCMC	Optimization
Basic	SGLD	SGD
Preconditioning	pSGLD [2]	RMSprop/Adagrad/Adam
Momentum	SGHMC	momentum SGD
Thermostat	SGNHT	Santa [3]

Understanding Dropout

On the connection of SGLD and Dropout

For neural networks with the nonlinear function $q(\cdot)$ and consecutive layers $\boldsymbol{h}_1$ and $\boldsymbol{h}_2$, dropout and dropConnect:

$$\text{Dropout:} \quad \boldsymbol{h}_2 = \boldsymbol{\xi}_0 \odot q(\boldsymbol{\theta}\boldsymbol{h}_1),$$
$$\text{DropConnect:} \quad \boldsymbol{h}_2 = q((\boldsymbol{\xi}_0 \odot \boldsymbol{\theta})\boldsymbol{h}_1),$$

where the injected noise $\boldsymbol{\xi}_0$ can be binary valued with dropping rate p or its equivalent Gaussian :

$$\text{Binary noise:} \quad \boldsymbol{\xi}_0 \sim \text{Ber}(p),$$
$$\text{Gaussian noise:} \quad \boldsymbol{\xi}_0 \sim \mathcal{N}(1, \frac{p}{1-p}).$$

By combining dropConnect and Gaussian noise

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\xi}_0 \odot \boldsymbol{\theta}_t - \frac{\eta}{2}\tilde{\boldsymbol{f}}_t = \boldsymbol{\theta}_t - \frac{\eta}{2}\tilde{\boldsymbol{f}}_t + \boldsymbol{\xi}_0' \,, \quad (3)$$

where $\boldsymbol{\xi}_0' \sim \mathcal{N}\left(0, \frac{p}{(1-p)}\text{diag}(\boldsymbol{\theta}_t^2)\right)$. Dropout/dropConnect and SGLD share the same form of update rule, with the only difference being that the level of injected noise is different.

- Integration of SG-MCMC and Binary Dropout
- Accelerating SG-MCMC using Batch-Normalization

Experiments: Shape Classification

I. Datasets

	FNN	CNN
2D	`MNIST, Animal, 20 Newsgroups`	`MNIST, Caltech, Cifar10`
3D	`Body Shape, Textured Shape`	`ModelNet`

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 the paper for a lot more results.

Networks	400-400	800-800	1200-1200
Methods	**Test Error (%)**		
pSGLD + Dropout	1.36	1.26	**1.15**
SGLD + Dropout	1.45	**1.25**	1.18
RMSprop + Dropout	1.35	1.28	1.24
SGD + Dropout	1.51	1.33	1.36
pSGLD	1.45	1.32	1.24
SGLD	1.64	1.41	1.40
RMSprop	1.79	1.43	1.39
SGD	1.72	1.47	1.47
RMSspectral	1.65	1.56	1.46
BPB, Gaussian	1.82	1.99	2.04
BPB, Scale mixture	**1.32**	1.34	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 show their ability to regularize learning. By integrating SG-MCMC with Dropout, we obtain lower error.

Very Deep Neural Networks

- D: 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.
- Exensive empirical results on adding gradient noise are also shown in [4].

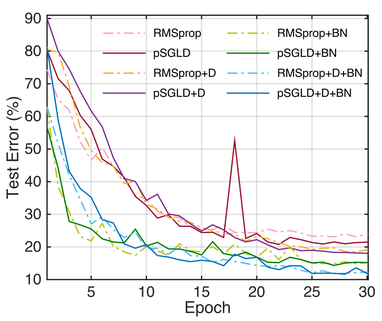

Figure: Learning curves on `Cifar10`

References

[1] Welling et al. *Bayesian Learning via Stochastic Gradient Langevin Dynamics*, ICML 2011
[2] Li et al, *Preconditioned Stochastic Gradient Langevin Dynamics for Deep Neural Networks*, AAAI 2016
[3] Chen et al, *Bridging the Gap between Stochastic Gradient MCMC and Stochastic Optimization*, AISTATS 2016
[4] Neelakantan et al, *Adding Gradient Noise Improves Learning for Very Deep Networks*, ICLR workshop 2016

Acknowledgements

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