Bayesian Neural Networks

Report

Egor Kuznetsov

Moscow State University

June 12, 2023

Bayesian Neural Networks

Bayesian Neural Networks:

- Variational Inference
- Markov Chain Monte Carlo

The idea of treating model's parameters as a random variables.

Bayesian Neural Networks

$$P(\theta|D) = \frac{P(D_y|D_x,\theta)P(\theta)}{\int_\theta P(D_y|D_x,\theta')P(\theta')d\theta'},$$

$$P(y|x,D) = \int P(y|x,\theta')P(\theta'|D)d\theta'$$

$$\approx \frac{1}{\Theta}\sum_{\theta_i\in\Theta}\Phi_{\theta_i}(x).$$

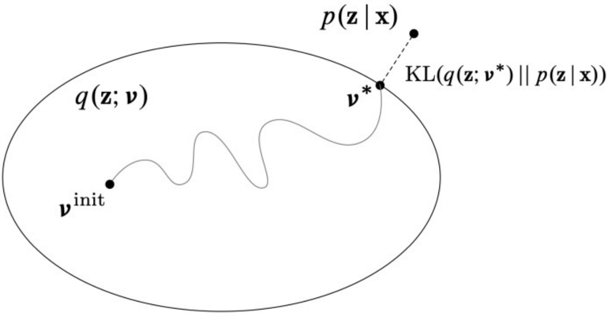
Variational Inference

$$P(\theta|D) \sim q_\phi(\theta),$$

where $q_\phi(\theta)$ is a family of parameterized distributions (e.g. $N(\mu_\phi, \sigma_\phi^2)$).

$$D_{KL}(q_\phi(\theta)||P(\theta|D)) \longrightarrow min \Rightarrow \mathbb{E}_{q_\phi} \log\left(\frac{\hat{P}(D|\theta)P(\theta)}{q_\phi(\theta)}\right) \longrightarrow max_\phi.$$

Variational Inference



Markov Chain Monte Carlo

Approximating $P(\theta|D)$ using predefined Markov Chain $J(\theta_n|\theta_{n-1})$.

$$r = \frac{P(\theta_*|D)}{P(\theta_{n-1}|D)},\ \ p = \min(1, r),\ \ \hat{P}(\theta|D) \propto P(D|\theta)P(\theta)$$

Could be done through No-U-Turn Sampler.[1]
It improves the idea of Hamiltonian Monte Carlo, removing the necessity of choosing step size and amount of "Leapfrog" (Stormer-Verlet integrator) steps.

[1] Hoffman M. D. et al. The No-U-Turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo

Data

MNIST dataset

MLP

Main characteristics:

- Multi-Layer Perceptron
- ReLU
- Batch Normalization

For Variational Inference, ''Adam'' optimizer was used with exponentially changed learning rate.

CNN

Mainly were used while researching Variational Inference. Main characteristics:

- Varying number of convolutional layers and their parameters
- AdaptiveAveragePooling and Flattenning
- ReLU (MaxPool was used in type 4, 5 models), Batch Normalization

CNN types

Table: The first type of CNN model

Layer	Input dim	Output dim	Kern size	Stride	Padding
Conv	1	16	5	1	2
Conv	16	32	3	1	1
Conv	32	64	3	1	1
Conv	64	64	3	2	0
Conv	64	32	3	2	0
Conv	32	16	3	2	0
Linear	16	10	-	-	-

CNN types

Table: The second type of CNN model

Layer	Input dim	Output dim	Kern size	Stride	Padding
Conv	1	16	5	1	0
Conv	16	32	3	1	0
Linear	32	10	-	-	-

CNN types

Table: The third type of CNN model

Layer	Input dim	Output dim	Kern size	Stride	Padding
Conv	1	4	3	1	0
Conv	4	8	3	1	0
Linear	8	10	-	-	-

CNN types

Table: The forth type of CNN model

Layer	Input dim	Output dim	Kern size	Stride	Padding
Conv	1	16	3	1	1
Conv	16	32	3	1	1
Conv	32	64	3	1	1
Linear	64	10	-	-	-

Variational Inference

Table: MLP model experimental results

Hid. layers	Perc. in layer	Prior distribution	Test accuracy
2	300	Laplace(0,1)	0.1021
1	200	Normal(0,1)	0.1042
2	400	Normal(0,1)	0.1094
2	100	Uniform(-1,1)	0.1097
3	400	Uniform(-20,20)	0.1122
1	400	Normal(0,1)	0.2
3	400	Laplace(0,20)	0.21
3	300	Normal(0,20)	0.3024
1	50	Uniform(-1,1)	0.3884
1	200	Uniform(-1,1)	0.3923
1	100	Normal(0,1)	**0.3953**
1	400	Laplace(0,1)	0.4047
1	100	Laplace(0,1)	0.5091
1	130	Uniform(-1,1)	0.5306
1	100	Uniform(-1,1)	**0.5352**
1	200	Laplace(0,1)	**0.5825**

Variational Inference

Table: CNN models experimental results

Type	Prior distribution	Test accuracy
3	Uniform(-1,1)	0.1032
1	Uniform(-20,20)	0.106
2	Uniform(-1,1)	0.1096
2	Normal(0,1)	0.1125
1	Normal(0,10)	0.6593
4	Normal(0,20)	0.6683
1	Normal(0,20)	**0.6812**
1	Laplace(0,20)	**0.6938**

Variational Inference

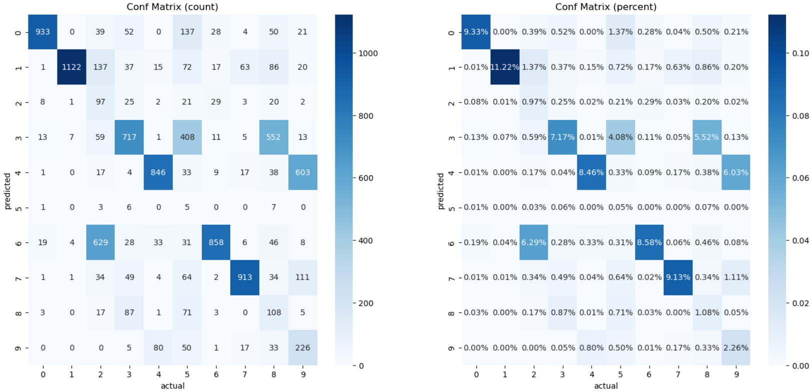


Figure: Confusion matrix. Laplace(0,1) priors, 1 hidden layer, 200 perc. in layer

Variational Inference

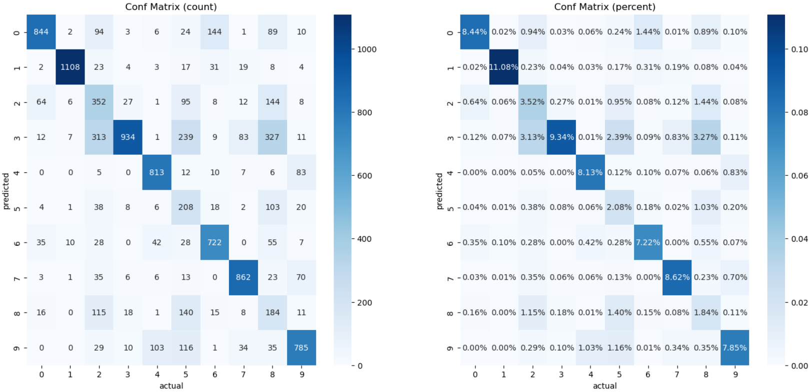


Figure: Confusion matrix. CNN. Normal(0,20) priors, 1-st type of architecture

Markov Chain Monte Carlo

prior dist/model type	1	2	3
Normal(0,10)	0.9758	0.9742	0.9731
Laplace(0,10)	0.9739	0.9511	0.9759
Uniform(-10,10)	0.9461	0.9737	0.9743

Table: Averaged accuracy on validation set for different types of models and weights priors

In the table all models are MLPs, type 1 model contains 1 hidden layer with 200 parameters, type 2 – 1 hidden layer, 400 parameters, type 3 – 2 hidden layers, 200 parameters.

Markov Chain Monte Carlo

To somehow estimate, whether the model had converged or not, for every weight r-hat estimation [2] (so called potential scale reduction) were computed:

$$\sqrt{\hat{R}} = \sqrt{\left(\frac{n-1}{n} + \frac{m+1}{mn}\frac{B}{W}\right)\frac{df}{df-2}},$$

where n is the amount of total parameters, m is the amount of samples, B and W are modelled through F-distribution, df are the degrees of freedom.

If the following estimation is close to 1 it means, that the parameter has converged and will not change drastically with more samples.

[2] Gelman A., Rubin D. B. Inference from iterative simulation using multiple sequences

Markov Chain Monte Carlo

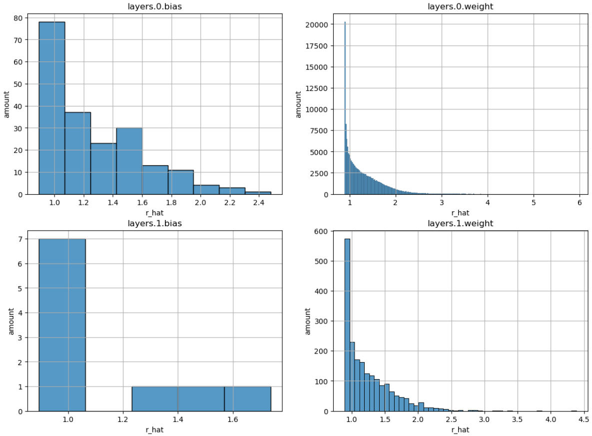


Figure: R-hat estimation for weights in layers. 1-st type of model, Normal(0,10) priors

Markov Chain Monte Carlo

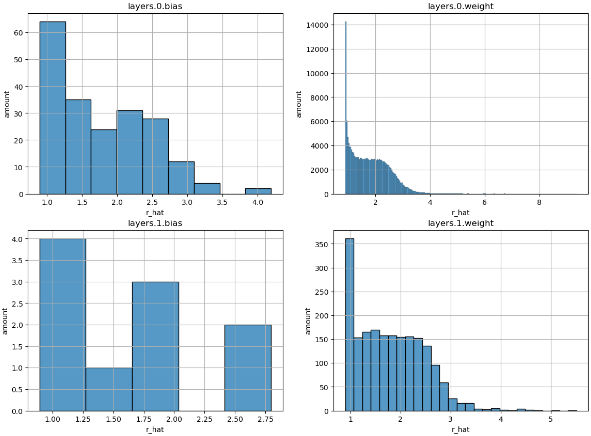


Figure: R-hat estimation for weights in layers. 1-st type of model, Uniform(0,10) priors

Markov Chain Monte Carlo

Robustness test. The noise was sampled from $N(100, 20)$ and applied to the validation sets of models, trained without noise.

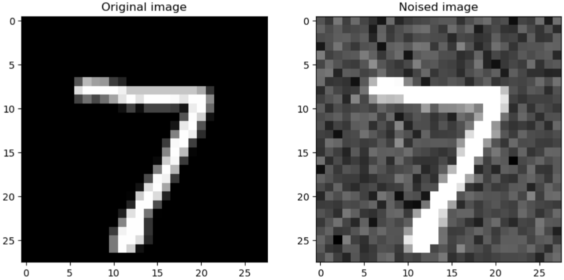


Figure: The number before and after noise addition

Markov Chain Monte Carlo

dist/type	1	2	3
Normal(0,10)	0.9519	0.9519	0.9610
Laplace(0,10)	0.9604	0.9130	0.9609
Uniform(-10,10)	0.9247	0.9603	0.9623

Table: Averaged accuracy for different types of models and weights priors. Noised dataset.

In the table all models are MLPs, type 1 model contains 1 hidden layer with 200 parameters, type 2 – 1 hidden layer, 400 parameters, type 3 – 2 hidden layers, 200 parameters.

Markov Chain Monte Carlo

CNN experiments

Table: Experiments with CNN type 5 model

Normal(0,10)	Laplace(0,10)	Uniform(-10,10)
0.2627	0.3772	0.2842

Type 5 model had two convolutional layers:

$$(k = 3, in\_dim_1 = 1, out\_dim_1 = 4, in\_dim_2 = out\_dim_2 = 4)$$

and one linear output layer $(4, 10)$.

Markov Chain Monte Carlo

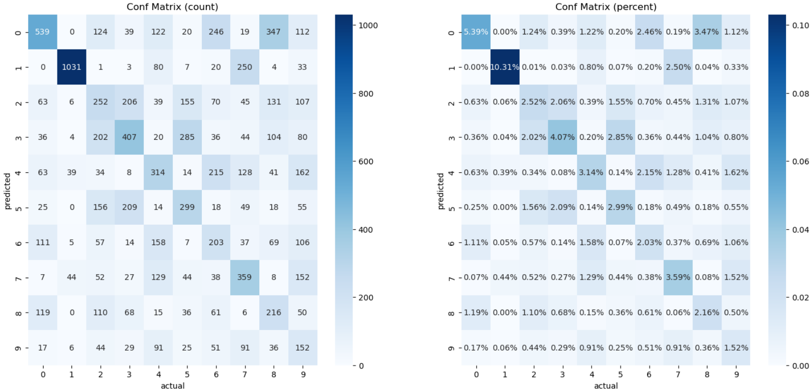


Figure: Confusion matrix. Laplace(0,10) priors, type 5 CNN model

Variational Inference

- There is no exact prior distribution, that is better than another ones.
- It is important to change the characteristics of the prior distributions.
- It is always better to begin the experiments with Bayesian networks from one deterministic model, that have the best results for the task.

Markov Chain Monte Carlo

- Although it's training of MCMC a lot of time, the results on MNIST dataset turned out to be quite good and consistent.
- As in the case with variational inference it is hard to tell, which prior distribution is better.
- To evaluate, whether our model has converged or not it is proposed to use histograms of r-hat estimation for each layer.
- MCMC's trained models showed their robustness on noised validation set.

Code availability

Code is available on GitHub:
https://github.com/MrSkonr/Bayesian-Neural-Net