Hamiltonian Monte Carlo sampled Bayesian Neural Network, for Classification and Regression

Mathilda Nguyen

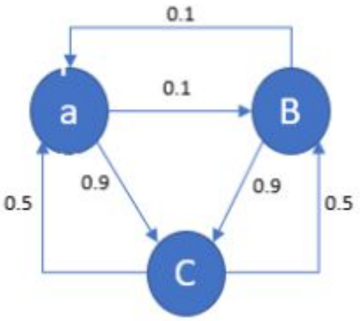
CS 615 - Deep Learning - Summer 2023

•••

MCMC

$$\mu \equiv \mathbb{E}_p[\phi(x)] = \int \phi(x)p(x)dx$$

$$\hat{\mu} \equiv \frac{1}{n}\sum_{i=1}^{n}\phi(x_i)$$



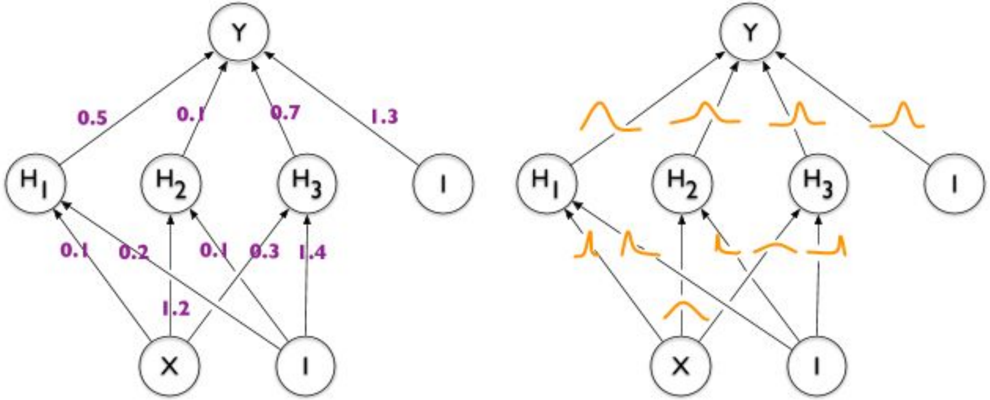
$$P = \begin{bmatrix} 0 & 0.1 & 0.9 \\ 0.1 & 0.0 & 0.9 \\ 0.5 & 0.5 & 0.0 \end{bmatrix}$$

Figure 1: Visualized markov chain and it's transition matrix P

Some amazing MCMC cartoon:

https://chi-feng.github.io/mcmc-demo/app.html#HamiltonianMC,donut

BNN



BNN posterior

$$p(w|\mathcal{D}) \propto p(\mathcal{D}|w)p(w)$$

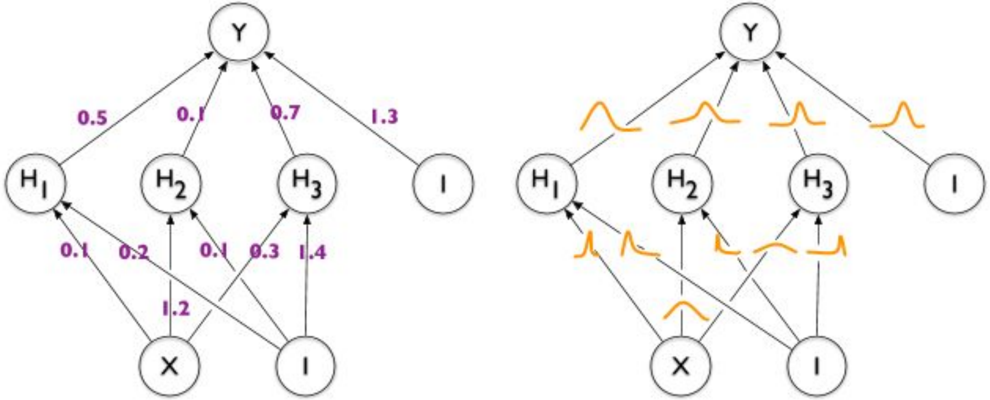
Prediction on data x

- Intractable!
- Can be approximated either by
  (1) Variational Inference (popular)
  (2) MCMC method (less popular)
  -> Hamiltonian Monte Carlo (HMC)

$$p(y|x,\mathcal{D}) = \int_w p(y|x,w)p(w|\mathcal{D})dw$$

$$\frac{1}{M}\sum_{i=1}^{M} p(y|x,w_i), \text{ where } w_i \sim p(w|\mathcal{D})$$

BNN



BNN posterior → $p(w|\mathcal{D}) \propto p(\mathcal{D}|w)p(w)$

Prediction on data x → $p(y|x,\mathcal{D}) = \int_w p(y|x,w)p(w|\mathcal{D})dw$

- Intractable!
- Can be approximated either by
  (1) Variational Inference (popular)
  (2) MCMC method (less popular) → $\frac{1}{M}\sum_{i=1}^{M} p(y|x,w_i)$, where $w_i \sim p(w|\mathcal{D})$
  -> Hamiltonian Monte Carlo (HMC)

The problem I'm tackling

Previously: HMC-BNN for galaxy morphology.

Now: HMC-BNN for

(1) Classification MNIST 100

(2) Regression Boston housing dataset

Traditional vs BNN

TRADITIONAL NEURAL NETWORK

Architecture: MLP

Training: Forward
Calculate Loss
Backward

Objective: Minimize Loss

Prediction: x -> Forward

BAYESIAN NEURAL NETWORK

Architecture: MLP
(and other probability functions for: prior, likelihood, posterior, kinetics, potential, grad potential)

Training: Forward
Run HMC, obtain new weights
Adjust weights

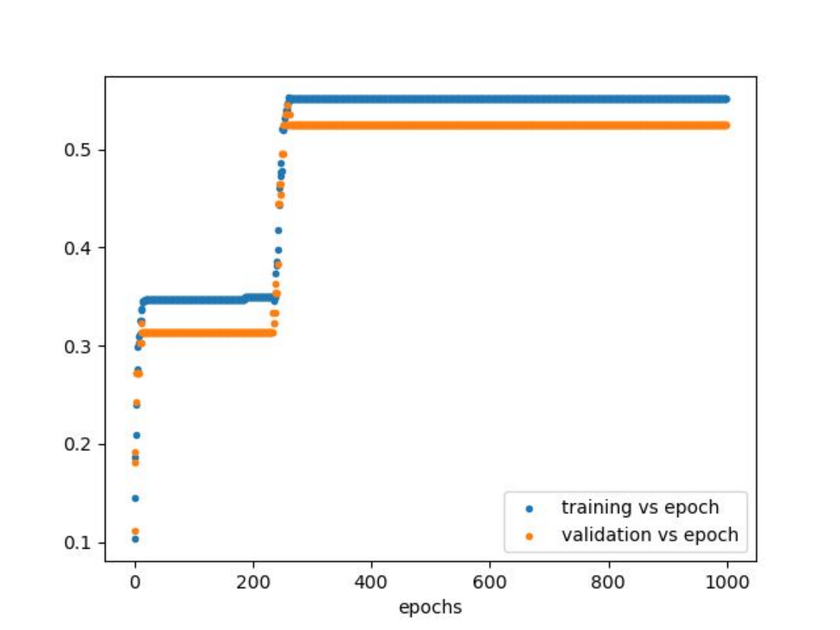
Objective: Maximizing log probability of p(D|w)

Prediction: sample from p(w|D), calculate integral p(y|w,b,D)

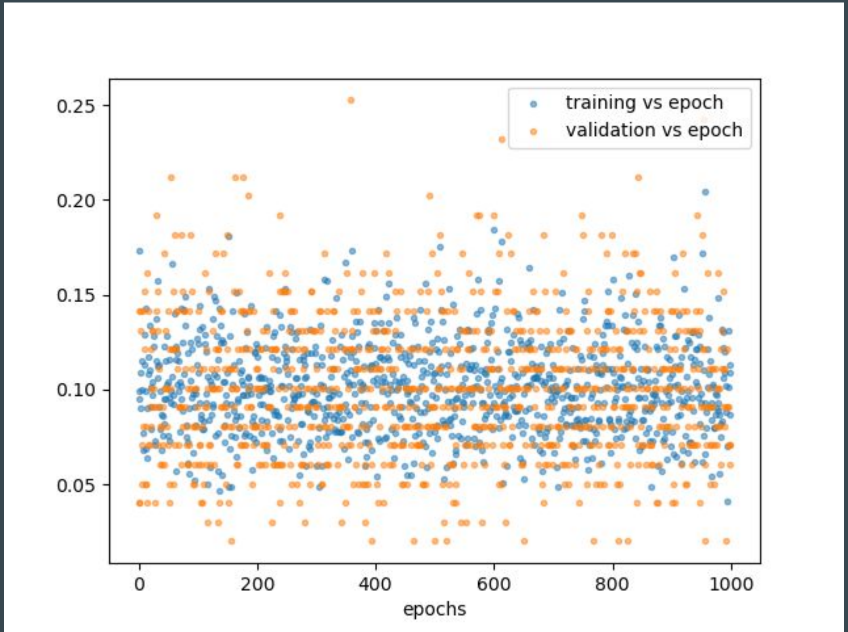
Classification on MNIST - devastating first step

```
[mathilda@mathilda-timstaller ~d/Class/615/project_helper]$ python -m pdb mlp.py
> /home/mathilda/Documents/Class/615/project_helper/mlp.py(1)<module>()                in make_vjp
-> import layers
(Pdb)
(Pdb) c                                                                                 , in trace
Y shape: (999, 10)
X shape: (999, 784)
validate Y shape: (99, 10)
validate X shape: (99, 784)
  0%|                                                            | 0/1000 [00:00<?, ?it/s]NN
 14%|                                                     | 14/100 [00:46<05:09,  3.56s/it]
 15%|                                                     | 15/1
 16%|                                                     | 16/1", line 834, in dispatch_nen_architecture
 17%|                                                     | 17/1
 18%|                                                     | 18/1
 19%|                                                     | 19/1                            .py", line 58, in arr
 20%|                                                     | 20/1", line 901, in dispatch_on
 21%|                                                     | 21/1
 22%|                                                     | 22/1                            , in f_wrapped
 23%|                                                     | 23/1", line 819, in _dispatch
100%|                                                    | 100/100 [05:58<00:00,  3.58s/it]
acceptance_rate= 0.0                                       | 100/100 [05:58<00:00,  3.56s/it].py", line 77, in arr
  0%|                                                 | 1/1000 [05:58<99:25:48, 358.31s/it]
  3%|                                                            | 3/100 [00:11<05:57,  3.69s/it]
```

Classification on MNIST - pretrain

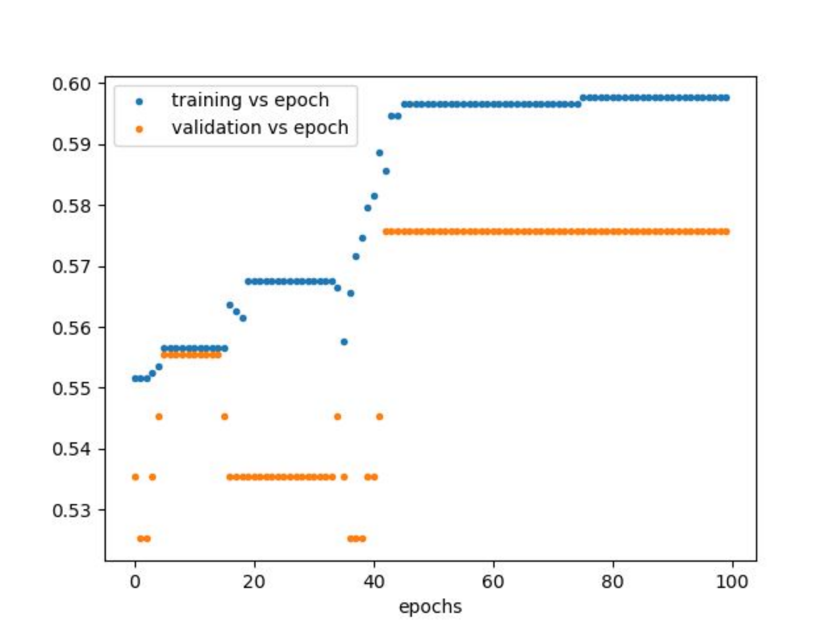


Traditional (few minutes)

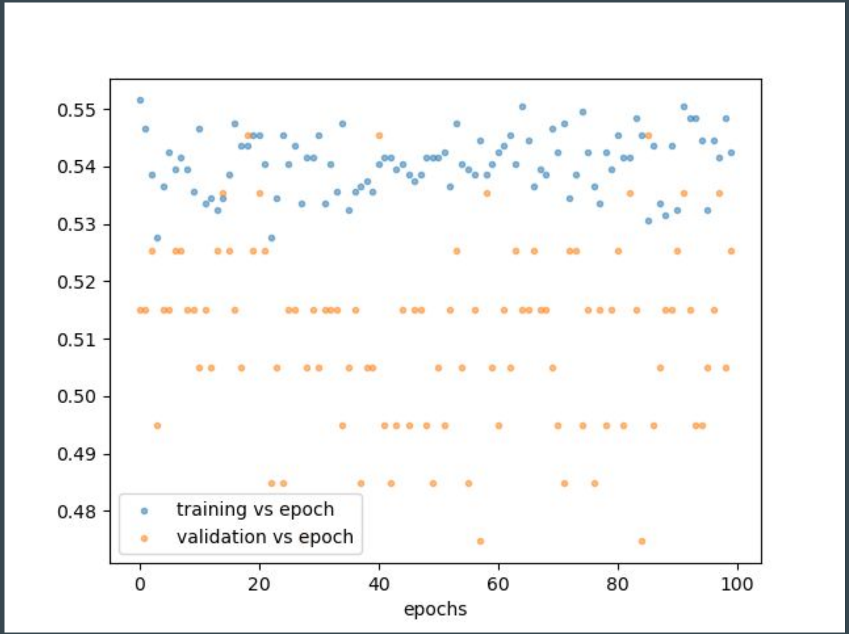


BNN (2 hours)

Classification on MNIST - after pretrain

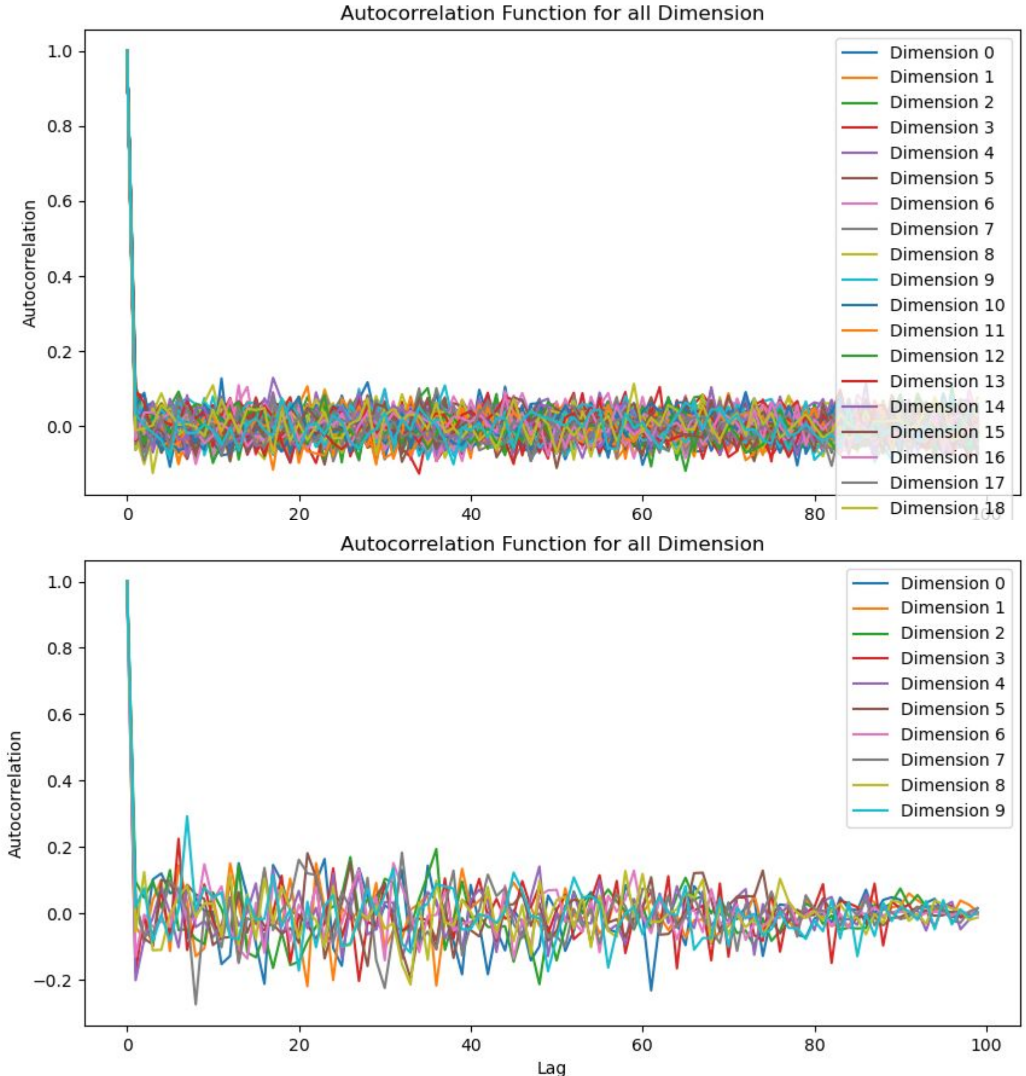


Traditional

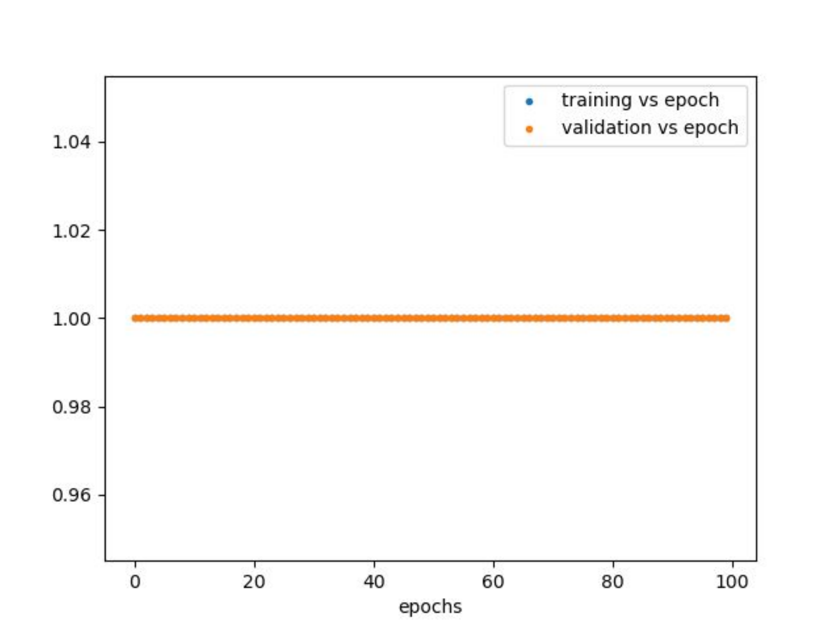


BNN

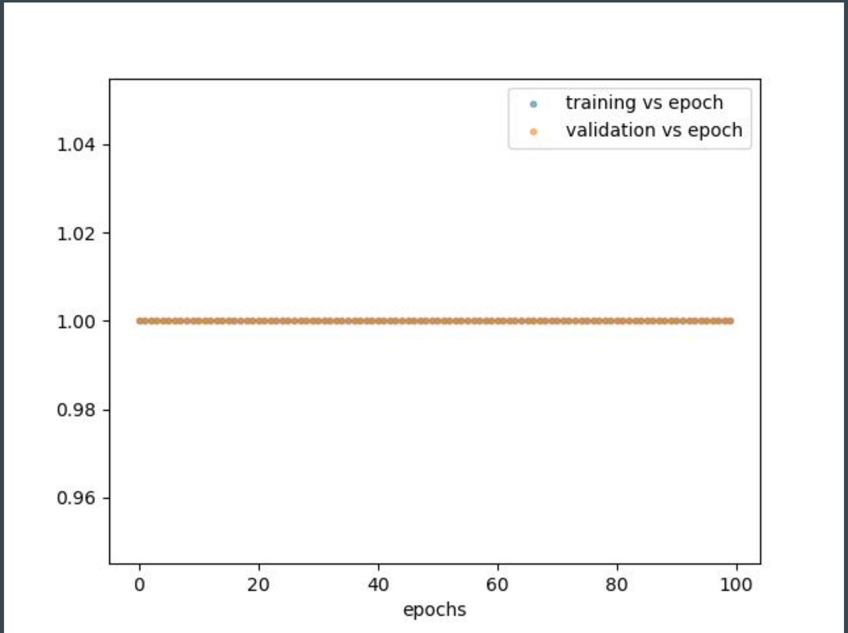
Classification on MNIST - HMC diagnostics



Regression on Boston housing

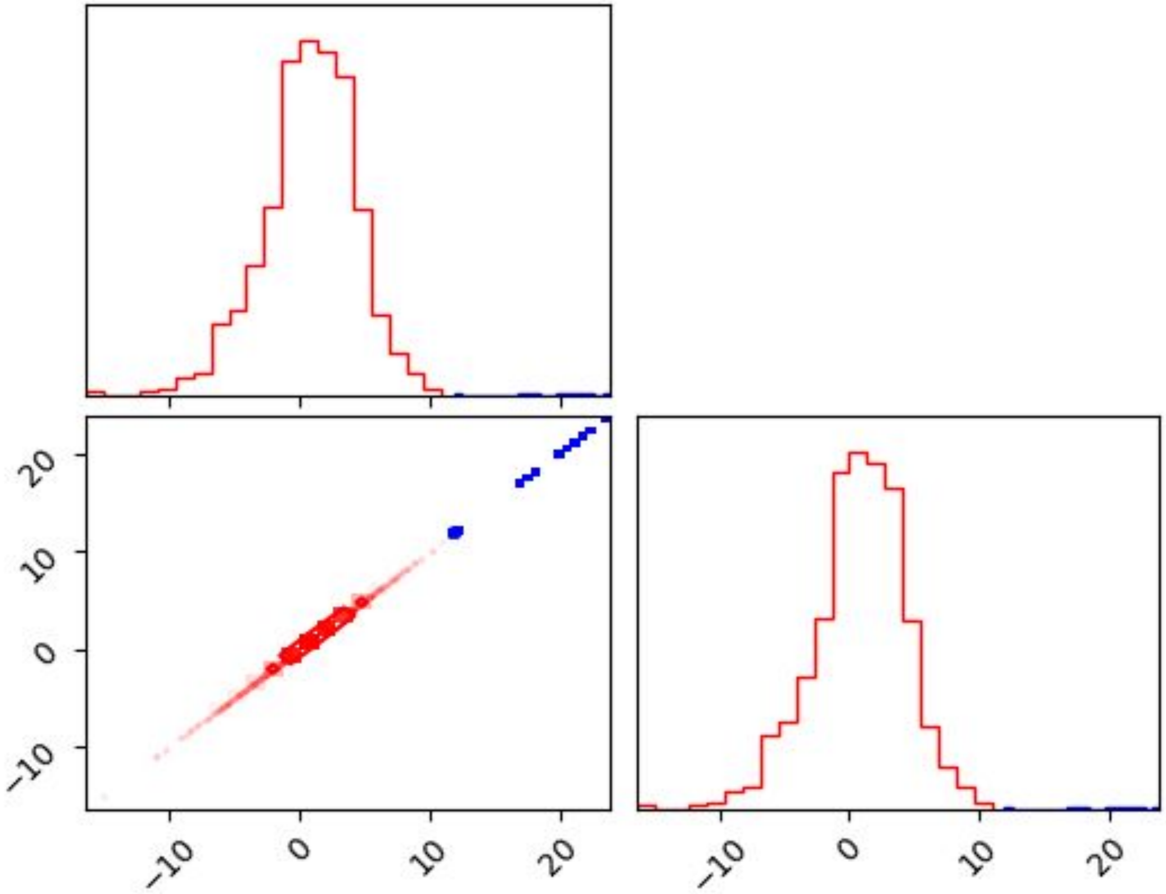


Traditional (few minutes). Tons of nan

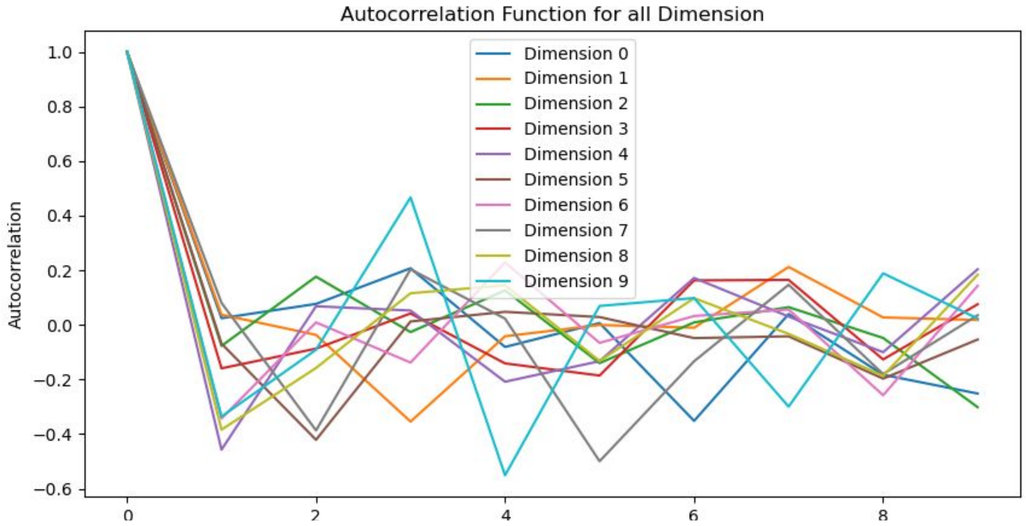


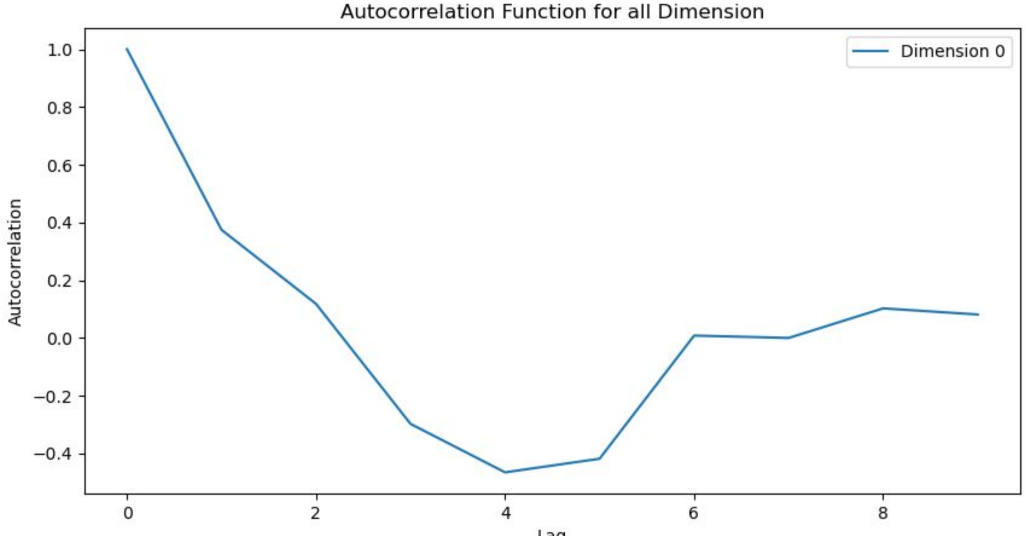
BNN. Looks weird... but there are no nans

But I actually got a plot



Regression on Boston housing - HMC diagnostics





Takeaways

TECHNICAL TAKEAWAYS

(1) Good parameters make a difference between 99 hours and 2 hours
(2) HMC-BNN really does not scale well. Maybe look into VI-BNN (the popular one) instead
(3) I vastly overestimate my ability to understand this thing. However, when you got it it turns out that changing traditional NN to a BNN is not hard!

PERSONAL TAKEAWAYS

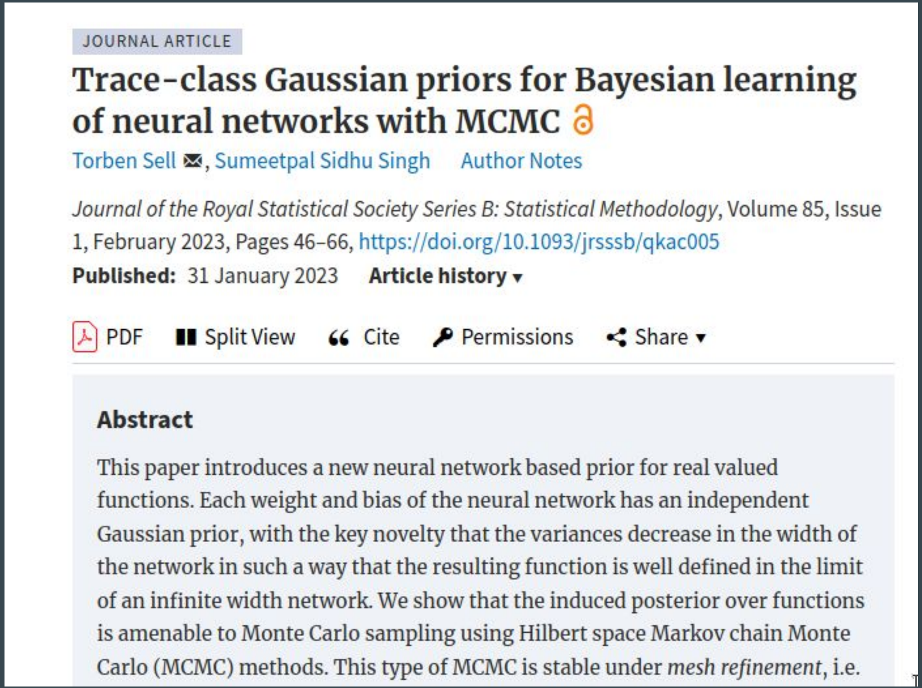
(1) I enjoyed it
(2) Although my result doesn't speak for it BNN is, according to a lot of people with more technical background than me, pretty good/better than traditional. Should be used more

Future direction

THIS PROJECT

(1) Maybe there's a logical error?
(2) Maybe not enough / too much data?
(3) Maybe how I interpreted the samples were wrong?
(4) Wider priors?

EXTENSION OF THIS PROJECT



JOURNAL ARTICLE

Trace-class Gaussian priors for Bayesian learning of neural networks with MCMC

Torben Sell, Sumeetpal Sidhu Singh Author Notes

Journal of the Royal Statistical Society Series B: Statistical Methodology, Volume 85, Issue 1, February 2023, Pages 46–66, https://doi.org/10.1093/jrsssb/qkac005

Published: 31 January 2023 **Article history** ▾

PDF Split View Cite Permissions Share ▾

Abstract

This paper introduces a new neural network based prior for real valued functions. Each weight and bias of the neural network has an independent Gaussian prior, with the key novelty that the variances decrease in the width of the network in such a way that the resulting function is well defined in the limit of an infinite width network. We show that the induced posterior over functions is amenable to Monte Carlo sampling using Hilbert space Markov chain Monte Carlo (MCMC) methods. This type of MCMC is stable under *mesh refinement*, i.e.

Thank you