Bayesian Neural Networks (BNNs)

Brown-Bag-Seminar 2020, IOS, Konstanz
Daniel Dold, HTWG Konstanz, Institute for Optical Systems

IOS
INSTITUT FÜR OPTISCHE SYSTEME

GEFÖRDERT VOM
Bundesministerium
für Bildung
und Forschung

12.11.2020

Contents

- From MaxLik to MAP to BNN
- Properties of BNNs
- Current Methods
  - MCMC
  - Variational Inference
  - MC-Dropout
  - DeepEnsembles
  - SWAG

Inspired by some papers we discussed in the JC:

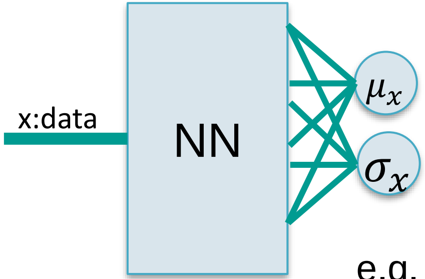
- Wilson, A. G., & Izmailov, P. (2020). Bayesian Deep Learning and a Probabilistic Perspective of Generalization.
- Gal, Y., Islam, R., & Ghahramani, Z. (2017). Deep Bayesian Active Learning with Image Data. 34th International Conference on Machine Learning, ICML 2017
- Farquhar, S., Smith, L., & Gal, Y. (2020). Liberty or Depth: Deep Bayesian Neural Nets Do Not Need Complex Weight Posterior Approximations.
- Betancourt, M. (2017). A Conceptual Introduction to Hamiltonian Monte Carlo.

Maximum likelihood (MaxLik)

$$\hat{\theta}_{ML} = \underset{\theta}{argmax} \prod_i p(y_i|x_i,\theta)$$

$$\hat{\theta}_{ML} = \underset{\theta}{argmin} - \sum_i \log p(y_i|x_i,\theta)$$

$$p(y|x,D) = p(y|x,\hat{\theta}_{ML})$$



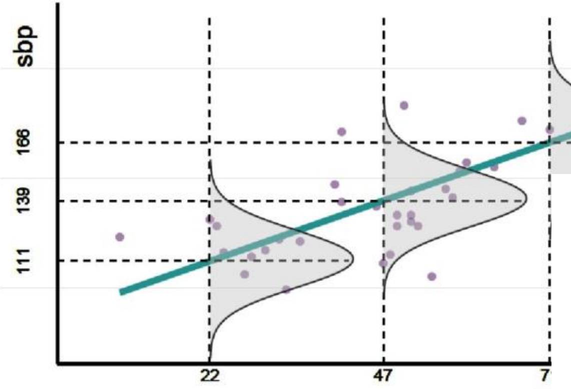
x:data

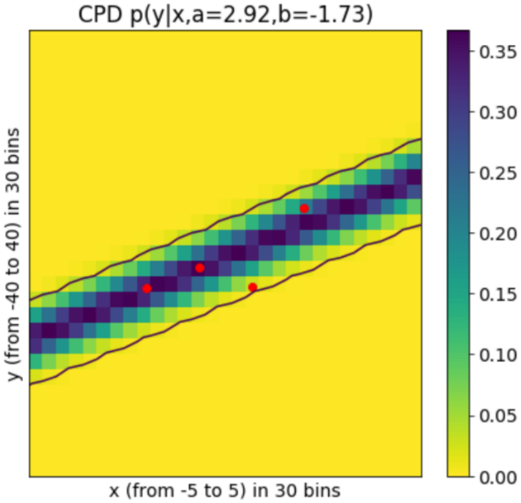
NN

μ_x

σ_x

e.g. $p(y|x,\hat{\theta}_{ML}) \sim N(\mu(x),\sigma(x))$





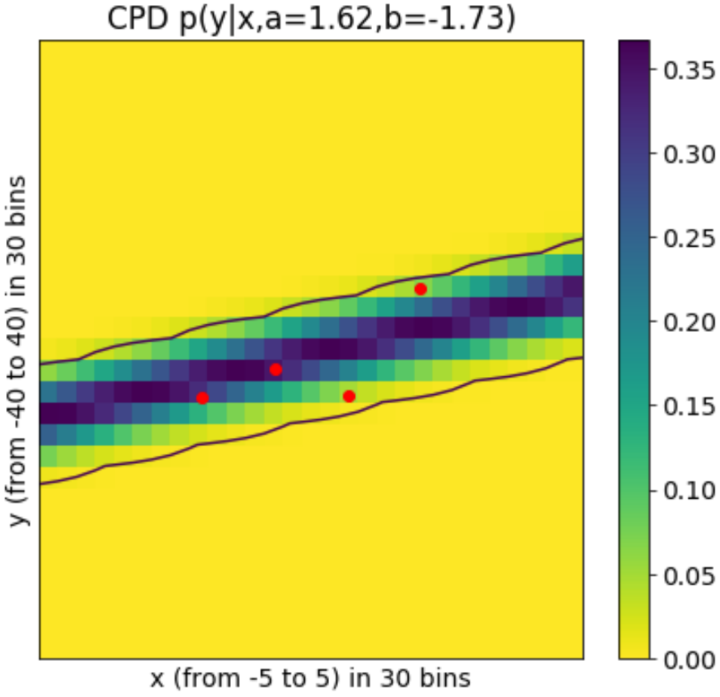
Maximum a-posteriori (MAP)

$$\hat{\theta}_{MAP} = \underset{\theta}{\operatorname{argmax}} \prod_i p(y_i|x_i,\theta)p(\theta)$$

$$\hat{\theta}_{MAP} = \underset{\theta}{\operatorname{argmin}} - \sum_i \left((\log p(y_i|x_i,\theta) + \log p(\theta)\right)$$

$$p(y|x,D) = p(y|x,\hat{\theta}_{MAP})$$

MAP converges towards MLE if we have enough data



Sidebar:

L2 regularisation == gausian prior

$$p(\theta) \sim N(0, \sigma^2)$$

$$p(\theta) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{\theta^2}{2\sigma^2}}$$

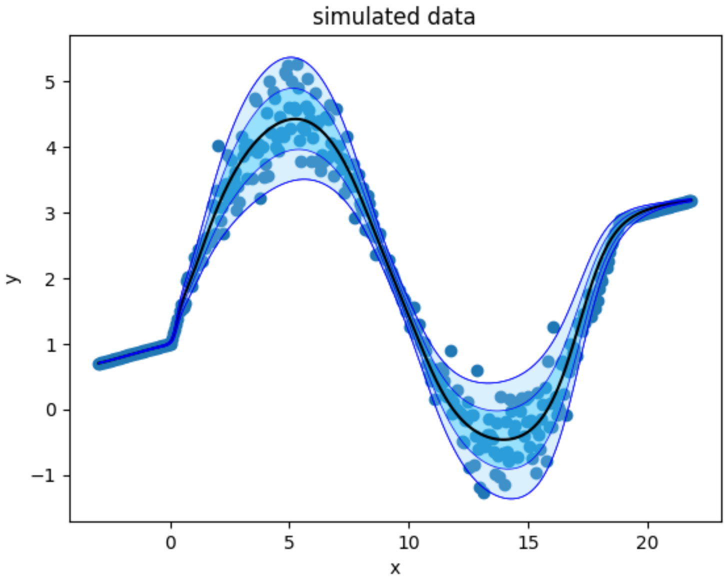
$$\log p(\theta) = -\frac{1}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\boldsymbol{\theta^2}$$

PyTorch implementation of l2 regularisation: $s_{weight\_decay}\theta$

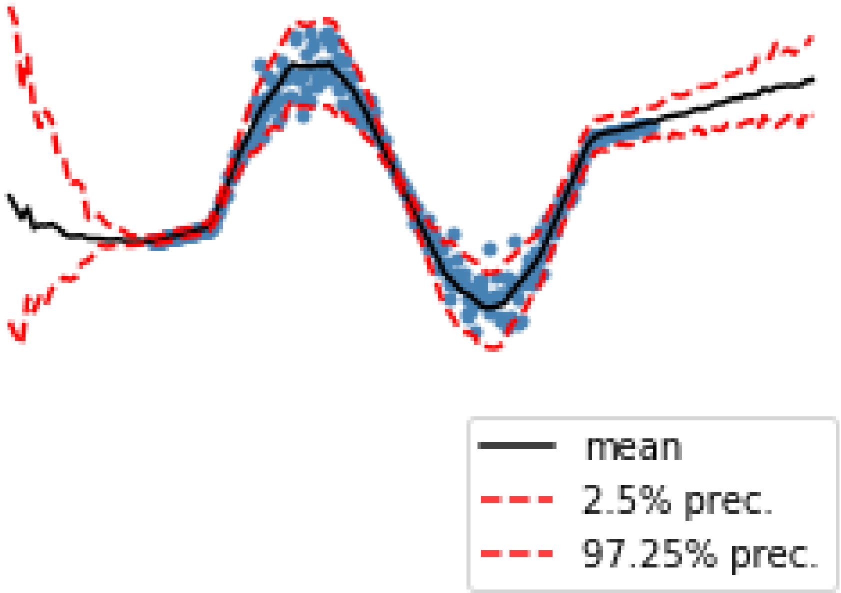
$$\frac{\partial \log p(\theta)}{\partial \theta} = \frac{\theta}{\sigma^2}$$

$\Rightarrow s_{weight\_decay} = \frac{1}{\sigma^2}$ or $s_{weight\_decay} = \frac{1}{\sigma^2 \, \#batch}$

Aleatoric vs epistemic uncertainty



Aleatoric uncertainty.



Aleatoric + epistemic uncertainty.

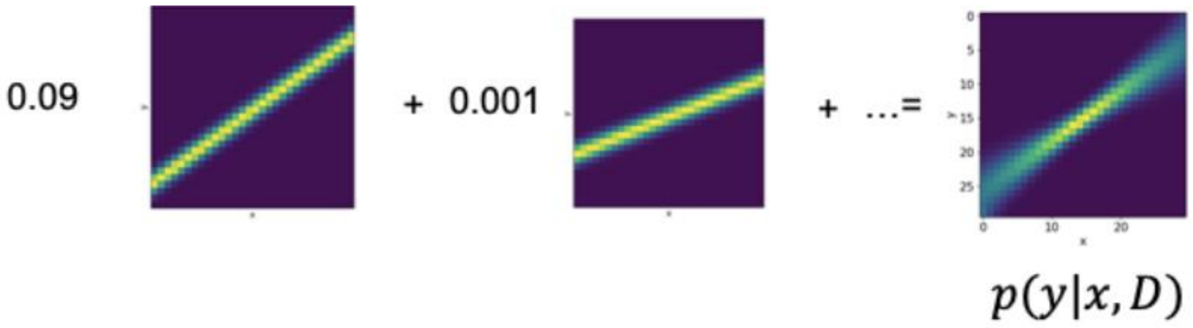
Baysian neural networks (BNNs)

Baysian model averaging (BMA)

$$p(y|x,D) = \int p(y|x,\theta) \cdot p(\theta|D) d\theta$$

Posterior predictive distribution

$$p(\theta|D) = \frac{p(D|\theta)p(\theta)}{\int p(D|\theta)p(\theta)d\theta}$$

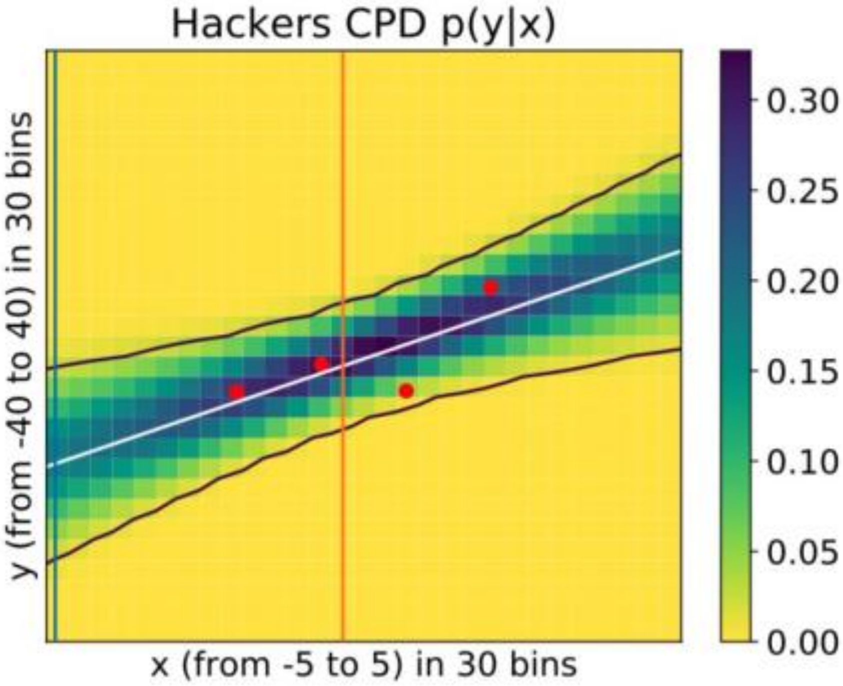


Monte Carlo integration

$$p(y|x,D) = \int p(y|x,\theta) \cdot p(\theta|D) d\theta$$

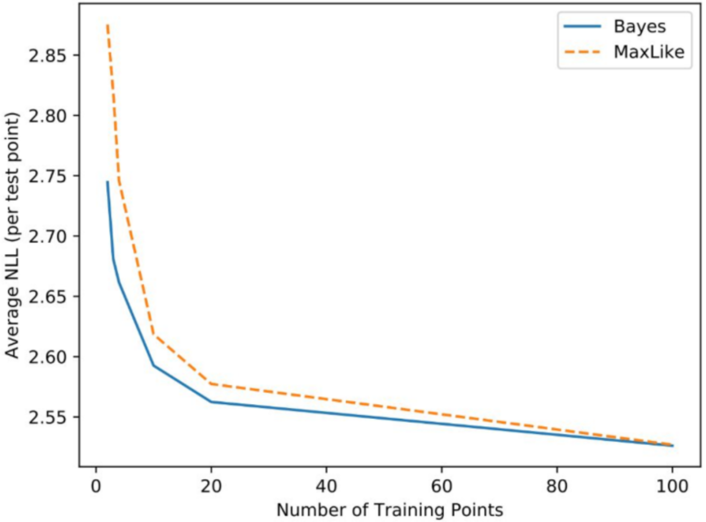
$$= \mathbb{E}_{p(\theta|D)} p(y|x,\theta)$$

$$\mathbb{E}_{p(\theta|D)} p(y|x,\theta) \approx \frac{1}{N} \sum_{\theta_i \sim p(\theta|D)}^{N} p(y|x,\theta_i)$$



BNNs properties

- Improved performance
- BNNs can capture epistemic uncertainty



Prediction performance of ML vs BNNs. The plot visualizes the NLL of a linear model in a regression task.

Sampling approaches

$$p(y|x,D) = \int p(y|x,\theta)\cdot p(\theta|D)d\theta \approx \frac{1}{N}\sum_{\theta_i\sim p(\theta|D)}^{N} p(y|x,\theta_i)$$

$$\boldsymbol{\theta_i\sim p(\theta|D)}$$

- Approximate $\mathrm{p}(\theta|D)$ with samples.
- MCMC Sampling Methods:
  - Gibbs
  - HMC (works in high dimensional space)
  - Metropolis Hastings
  - RWM
  - ...
- Does not require to compute $\int p(D|\theta)p(\theta)d\theta$
- Exact given enough computational time
- Does not work with large parameters

Markov chain Monte Carlo (MCMC)

$$p(y|x, D) = \int p(y|x, \theta) \cdot p(\theta|D) d\theta \approx \frac{1}{N} \sum_{\theta_i \sim p(\theta|D)}^{N} p(y|x, \theta_i)$$

$$\boldsymbol{\theta_i \sim p(\theta|D)}$$

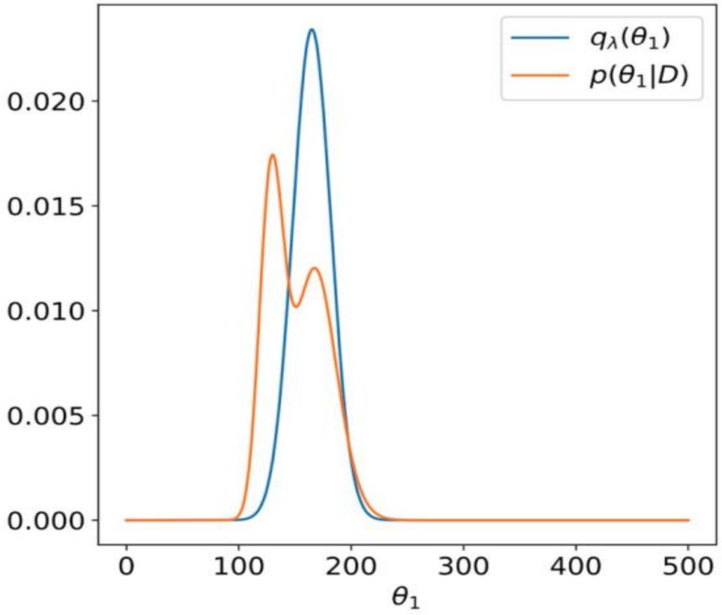
- Markov chain: History doesn‘t matter
- Metropolis Algorithm
  1. Start with an initial state θ_s
  2. Choose a random proposal θ_p (e.g $\theta_p \sim N(\theta_s, 1)$)
  3. Move to new proposeal with probability $min\left(1, \frac{p(D|\theta_p)p(\theta_p)}{p(D|\theta_s)p(\theta_s)}\right)$
  4. Report θ
  5. Repeat step 2-4 until markov chain converge

Variational inference

$$p(y|x,D) = \int p(y|x,\theta) \cdot p(\theta|D) d\theta \approx \int p(y|x,\theta) \cdot \mathrm{q}_{\lambda}(\theta) d\theta$$

$$\boldsymbol{p(\theta|D) \approx q_{\lambda}(\theta)}$$

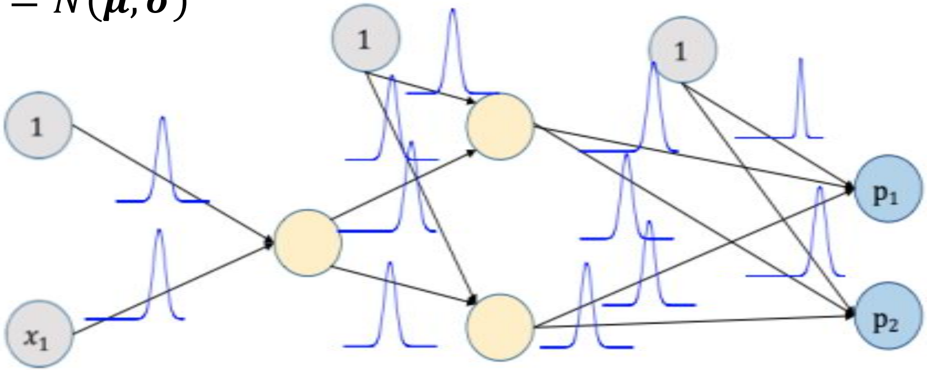
- Aim: approximate a complicated posterior $p(\theta|D)$ with a simpler one $\mathrm{q}_{\lambda}(\theta)$.
- λ defines the variational parameter
- Sampling from variational distribution is then straight forward
- Challenge: Tune λ until variational distribution is as close as possible to the real posterior distribution



Variational inference

$$p(\theta|D) \approx \mathrm{q}_{\lambda}(\theta)$$

- Replace weights with a variational distribution
- E.g. Every weight is Gausian ➔ $\mathrm{q}_{\lambda}(\theta) = N(\boldsymbol{\mu}, \boldsymbol{\sigma})$
- $\lambda = (\boldsymbol{\mu}, \boldsymbol{\sigma})$



Variational inference optimization

Minimize reverse KL-divergence

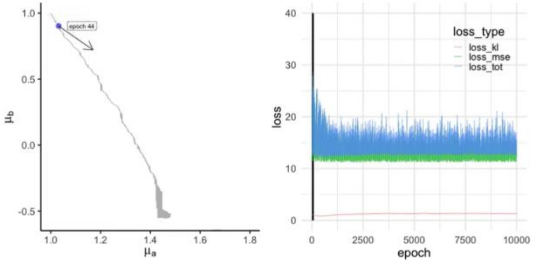
$$KL[q_\lambda(\theta)||p(\theta|D)] = \int q_\lambda(\theta) \log \frac{q_\lambda(\theta)}{p(\theta|D)} d\theta$$

⇩

$$\lambda^* = \operatorname{argmin}\{KL[q_\lambda(\theta)||p(\theta)] - \mathbb{E}_{\theta\sim q_\lambda}[\log p(D|\theta)]\}$$

KL Averaged NLL

- Solve λ^* with SGD
- Approximate $E_{\theta\sim q_\lambda}[\log p(D|\theta)]$ with a single sample each SGD step

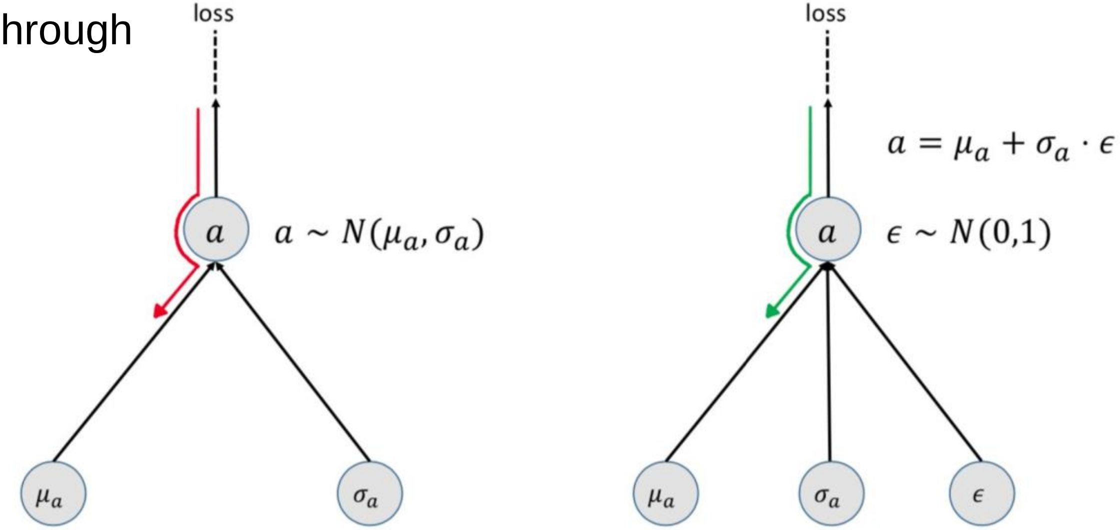


https://www.youtube.com/watch?v=MC_5Ne3Dj6g

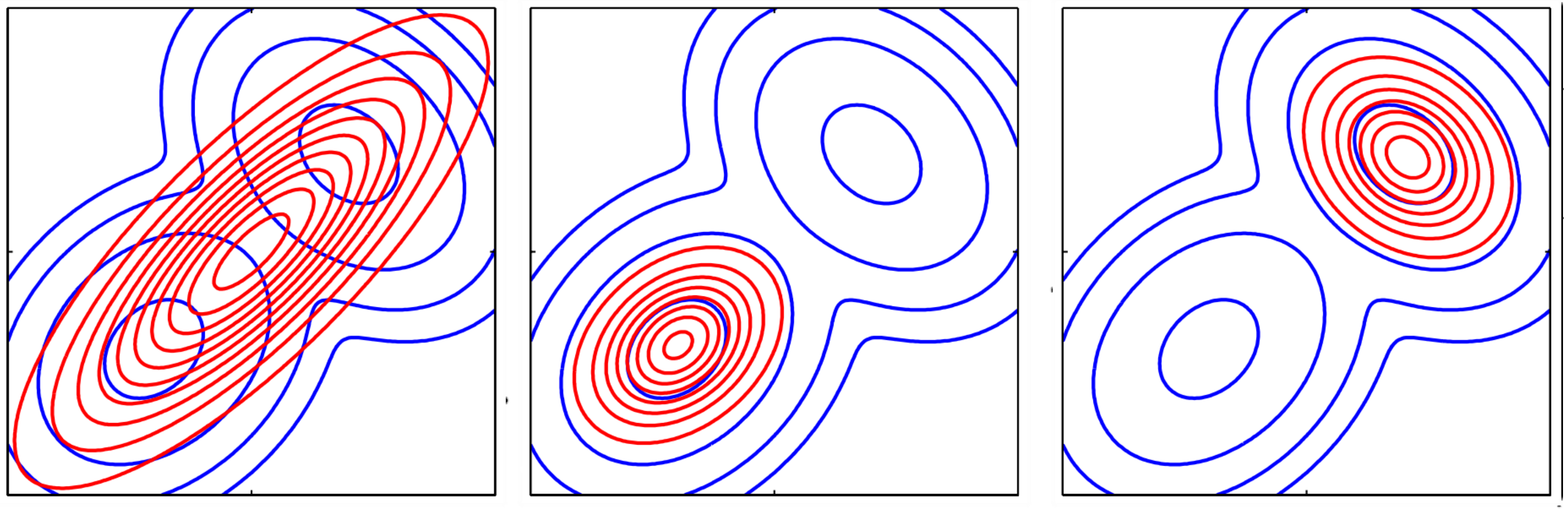
Reparameterization trick
(2013, Kingma and Welling)

How to propagate the gradient through sampling?

$$N(\mu_a, \sigma_a) = \mu_a + \sigma_a \cdot \epsilon,$$
$$\epsilon \sim N(0,1)$$



Difference between forward and backward KL



Blue lines represents the bimodal distribution and the red contours represents the approximated single Gaussian distribution. The left figure shows the result from the forward KL and the other two visualizes the results from the backward KL-divergence. Figure is taken from Bishop (2006).

MC-Dropout

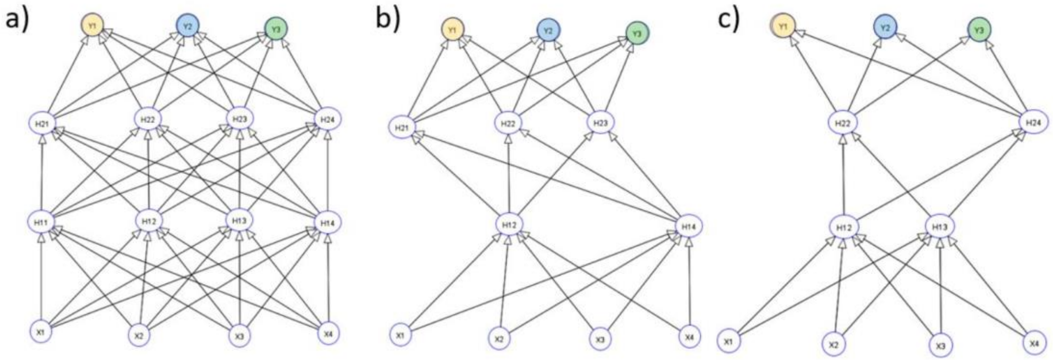
(Gal et al., 2016)

Use Dropout during training and **inference**

$$p(y|x,D) = \int p(y|x,\theta) \cdot p(\theta|D)d\theta$$

$$\approx \int p(y|x,\theta) \cdot q_{\vartheta}^{*}(\theta)d\theta$$

$$\approx \frac{1}{T}\sum_{t=1}^{T} p\left(y|x,\widehat{\theta_t}\right)$$



DeepEnsembles

(Lakshminarayanan et al., 2017)

Kaggle wisdom: several models are better

$$p(y|x,D) = \int p(y|x,\theta) \cdot p(\theta|D) d\theta \approx \frac{1}{M} \sum_{i=1}^{M} p(y|x,\theta_i)$$

- Initialize M models independently
- Train each model independent
- Training with adversarial examples (not necessary)

SWAG

(Maddox et al., 2019)

Extension of Stochastic Weight Averaging (SWA) (Izmailov et al., 2018)

Use Gaussian distribution for weights

Algorithm:

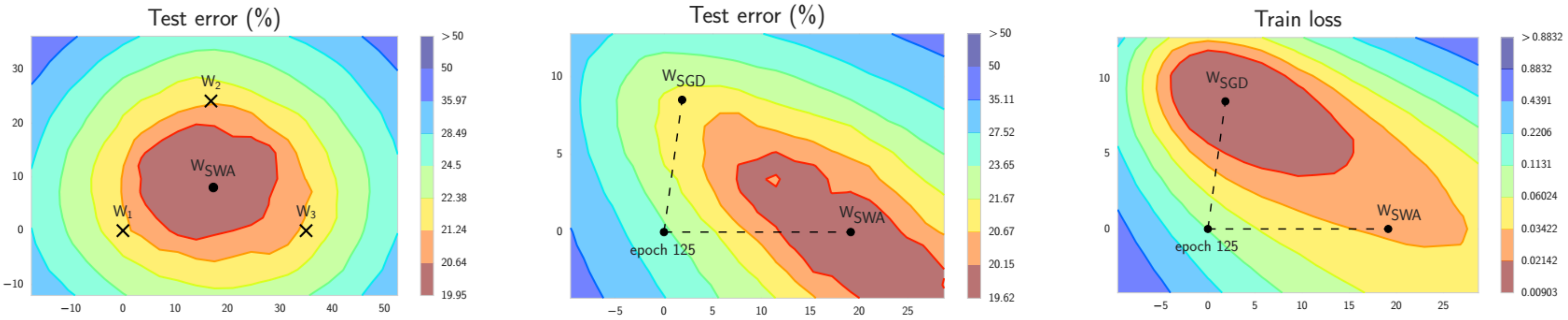
- Pretrain model
- Retrain model and compute statistics after each epoch:
    - SWA ➔ compute $\overline{\theta_{SWA}}$
    - SWAG ➔ compute $\bar{\theta}, \Sigma_{diag}$ or $\bar{\theta}, \Sigma_{low-rank}$

$$p(y|x,D) = \int p(y|x,\theta) \cdot p(\theta|D) d\theta \approx \frac{1}{M} \sum_{i=1}^{M} p(y|x,\theta_i), \qquad \theta_i \sim N(\bar{\theta}, \Sigma)$$

MultiSWAG: Combine SWAG with DeepEnsembles (Wilson et al., (2020))

Property of SWA → SWAG

SWA argue that they generalize better



Cross entropy loss of train and test loss. Figure taken from Izmailov et al., (2018)

Is SWAG Bayesian?

(Maddox et al., 2019)

- Use a regularisation as a prior
- Assumes that SGD steps evoke the posterior distribution!?
    - Discusse in one of the upcoming journal clubs

Evaluate the uncertainty BNNs

$$p(y|x,D) = \int p(y|x,\theta) \cdot p(\theta|D) d\theta \approx \frac{1}{M}\sum_{i=1}^{M} p(y|x,\theta_i)$$

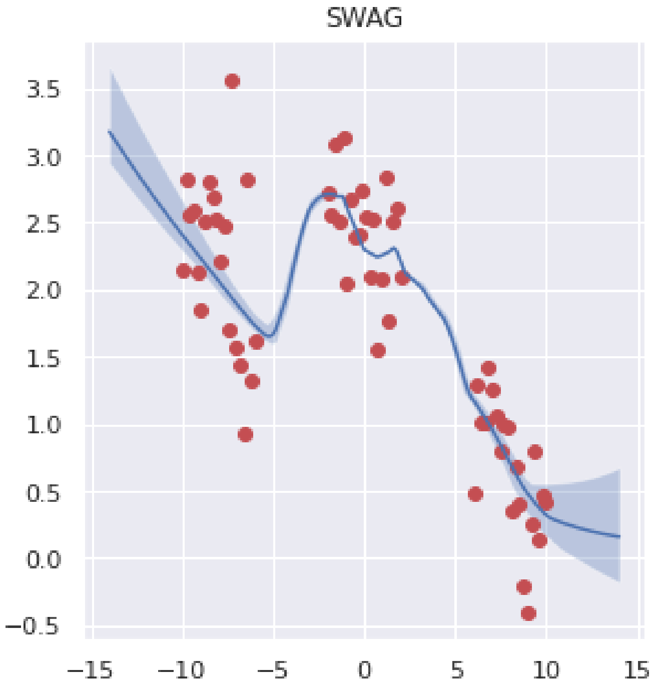
In Regression:

- Variance $\frac{1}{M}\sum_{i=1}^{M} p(y|x,\theta_i)^2 - \left(\frac{1}{M}\sum_{i=1}^{M} p(y|x,\theta_i)\right)^2$
- Quantiles

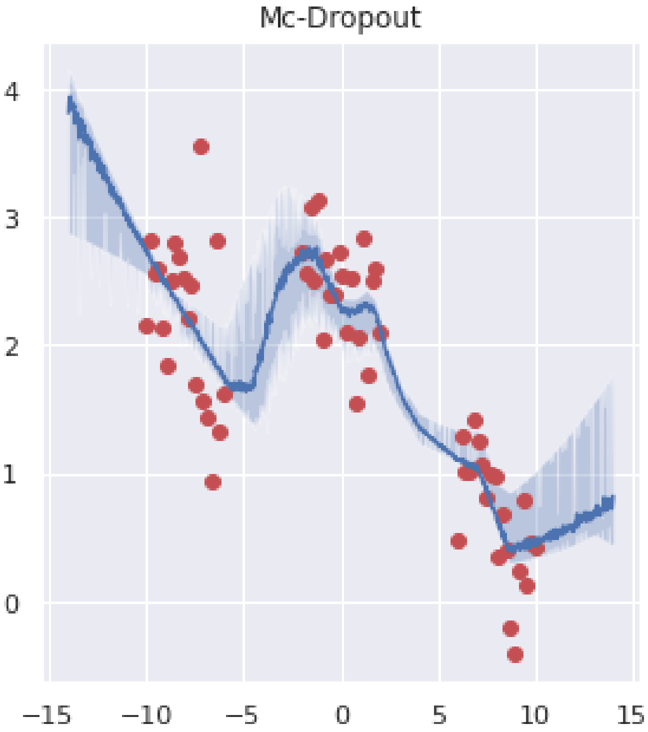
In Classification:

- Entropy $\mathrm{H} = -\sum_c \sum_i p(y|x,\theta_i) \log \sum_i p(y|x,\theta_i)$
- BALD $\mathrm{I}[\mathrm{y},\theta|x,D] = \mathrm{H}[\mathrm{y}|\mathrm{x},\mathrm{D}] - \mathbb{E}_{\theta\sim p(\theta|D)}\left[\mathrm{H}[\mathrm{y},\mathrm{x},\theta]\right]$ (Houlsby et al., 2011)

Regression SWAG vs Mc-Dropout



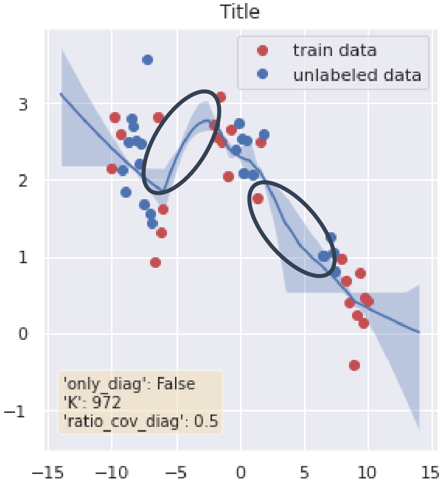
SWAG

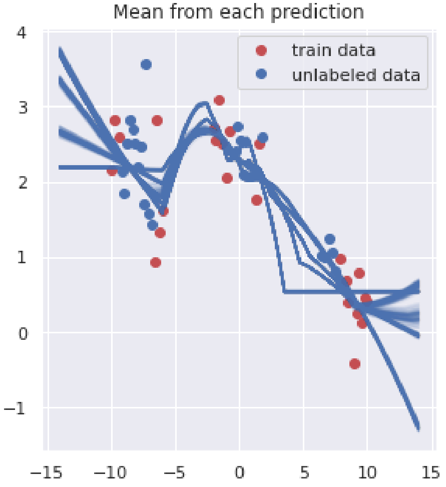


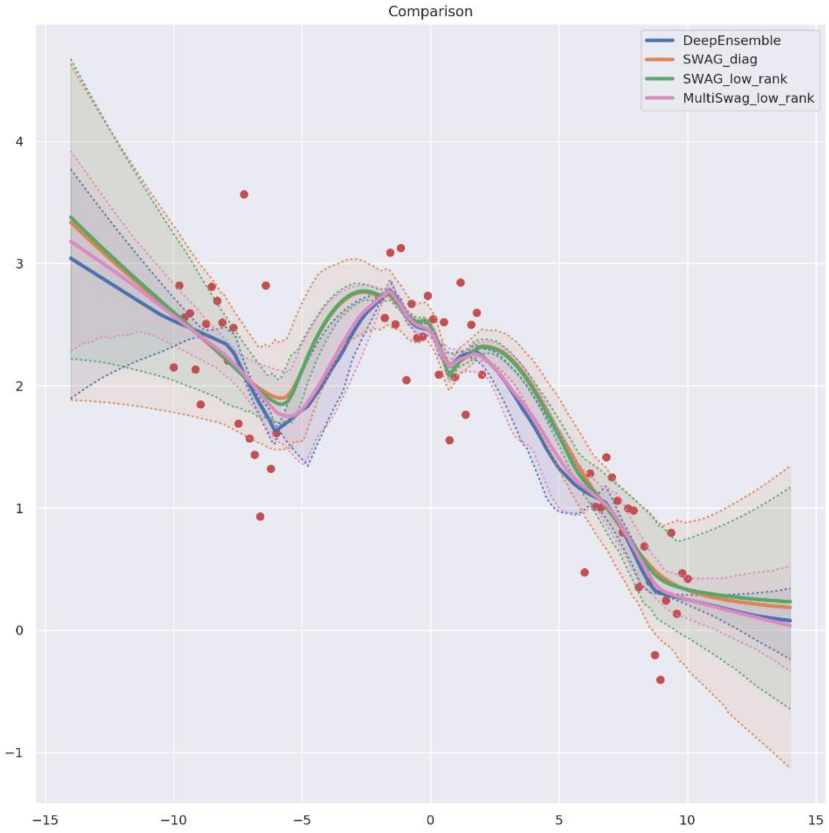
Mc-Dropout

Regression

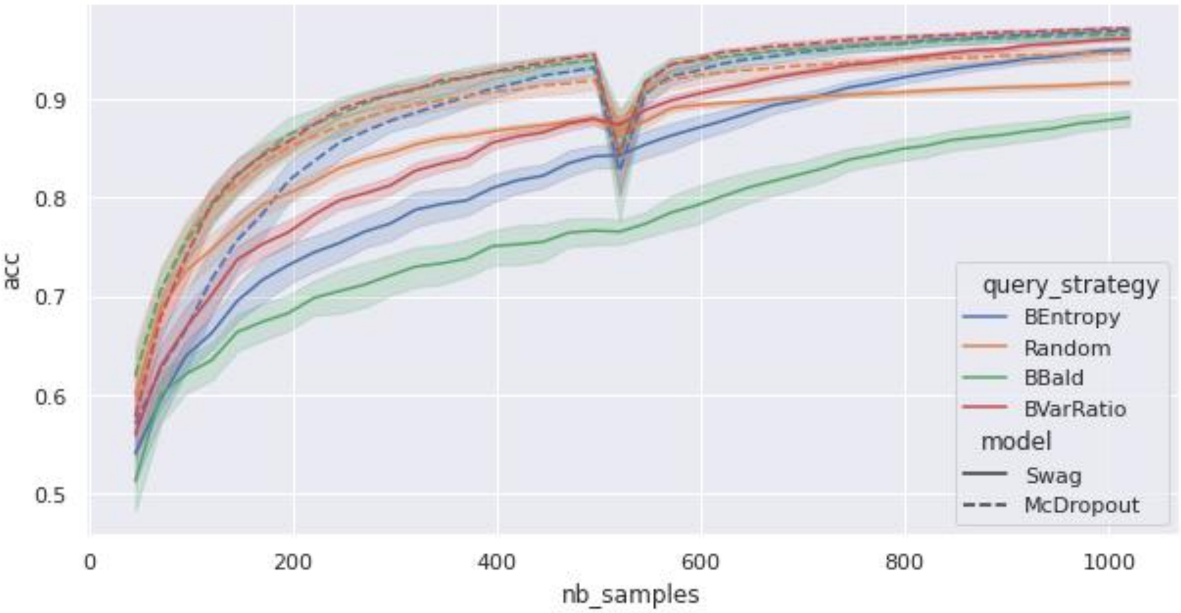
- MultiSWAG.
- Current research question: Which BNNs provide useful in-between uncertainty? (Farquhar et al. 2020)



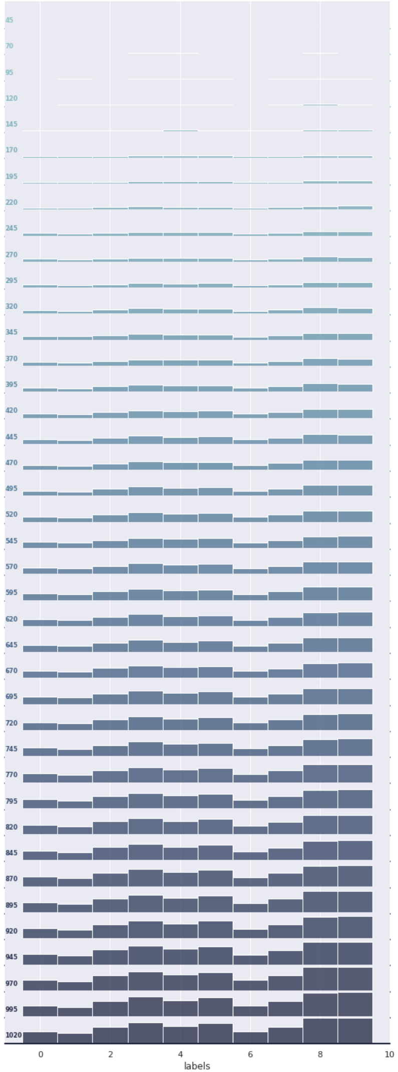




MNIST Classification



AL performance of McDropout with BALD acquisition function



Summarize

- BMA is the core of BNNs $p(y|x, D) = \int p(y|x, \theta) \cdot p(\theta|D) d\theta$
- Model diversity is more important than modelling exact posteriors (Wilson et al., 2020)
- VI stuck in one mode?! (Wilson et al., 2020) (Stefan MA)
- SWAG performance depends heavily on hyper-parameters
- SWAG does not model epistemic uncertainty well

References

- Lakshminarayanan, B., Pritzel, A., & Deepmind, C. B. (2017). Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles.
- Maddox, W., Garipov, T., Izmailov, P., Vetrov, D., & Wilson, A. G. (2019). A Simple Baseline for Bayesian Uncertainty in Deep Learning.
- Izmailov, P., Podoprikhin, D., Garipov, T., Vetrov, D., & Wilson, A. G. (2018). Averaging Weights Leads to Wider Optima and Better Generalization. 34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018, 2, 876–885. http://arxiv.org/abs/1803.05407
- Wilson, A. G., & Izmailov, P. (2020). Bayesian Deep Learning and a Probabilistic Perspective of Generalization. http://arxiv.org/abs/2002.08791
- Gal, Y., & Uk, Z. A. (2016). Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning Zoubin Ghahramani. PMLR. http://yarin.co.
- Gal, Y., Islam, R., & Ghahramani, Z. (2017). Deep Bayesian Active Learning with Image Data. 34th International Conference on Machine Learning, ICML 2017, 3, 1923–1932. http://arxiv.org/abs/1703.02910
- Houlsby, N., Huszár, F., Ghahramani, Z., & Lengyel, M. (2011). Bayesian Active Learning for Classification and Preference Learning.
- Farquhar, S., Smith, L., & Gal, Y. (2020). Liberty or Depth: Deep Bayesian Neural Nets Do Not Need Complex Weight Posterior Approximations.
- Bishop, C. (2016). *Pattern Recognition and Machine Learning.* Springer-Verlag New York
- Murphy, K (2012). *Machine learning : a probabilistic perspective*. Massachusetts Institute of Technology

Thanks

For your attention

Most of the figures are taken from the Book „Duerr, Oliver, Sick, Beate, Murina, Elvis. *Probabilistic Deep Learning: With Python, Keras and TensorFlow Probability.* Manning Publications 2020".