Probabilistic Artificial Intelligence
Tutorial 7: Bayesian Deep Learning

Alexander Immer

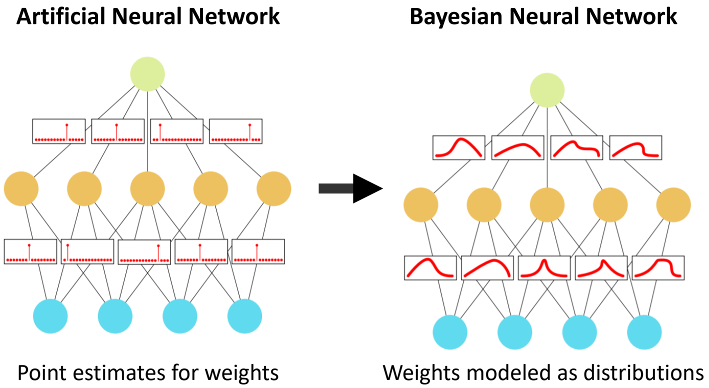
Department of Computer Science
Institute for Machine Learning

November 9, 2023

Outline

Bayesian Neural Networks (BNNs)

Bayesian Neural Network (I)


Artificial Neural Network
Bayesian Neural Network
Point estimates for weights
Weights modeled as distributions

Edited figures from Jospin et al. [2020].

Bayesian Neural Network (II)

Prior: Bayesian neural networks specify a prior $p(\boldsymbol{\theta})$ over weights $\boldsymbol{\theta}$. (e.g., Gaussian prior $p(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}; 0, \sigma^2 I)$)

Likelihood: Likelihood function parameterized by neural network $\boldsymbol{f}$. For example, categorical for classification or Gaussian for regression:

$$p(y_{1:n}|\boldsymbol{x}_{1:n}, \boldsymbol{\theta}) = \prod_{i=1}^{n} p(y_i|\boldsymbol{x}_i, \boldsymbol{\theta}) = \prod_{i=1}^{n} p(y_i|\boldsymbol{f}(\boldsymbol{x}_i; \boldsymbol{\theta}))$$

Posterior:

$$p(\boldsymbol{\theta}|\boldsymbol{x}_{1:n}, y_{1:n}) = \frac{1}{Z} p(\boldsymbol{\theta}) p(y_{1:n}|\boldsymbol{x}_{1:n}, \boldsymbol{\theta})$$

where $Z = p(y_{1:n}|\boldsymbol{x}_{1:n}) = \int p(\boldsymbol{\theta}) p(y_{1:n}|\boldsymbol{x}_{1:n}, \boldsymbol{\theta})\, d\boldsymbol{\theta}$ is the *intractable marginal likelihood* for NNs.

Comparison to Standard Neural Networks

Maximum a posteriori (MAP) is a typical neural network objective.

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} \; \log p(\boldsymbol{\theta}) + \sum_{i=1}^{n} \log p(y_i|\boldsymbol{x}_i, \boldsymbol{\theta})$$

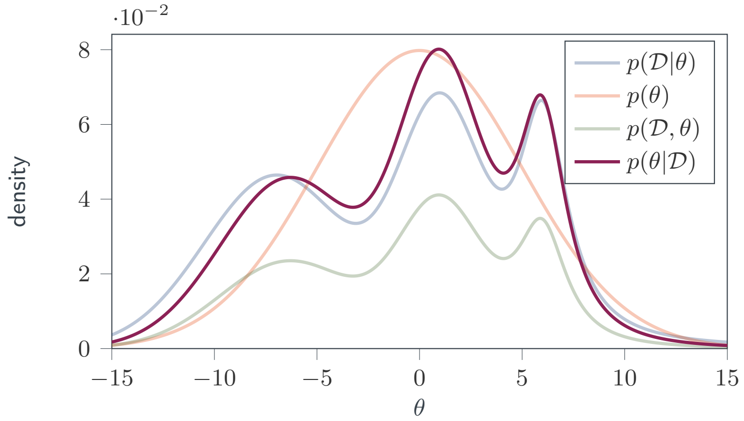
We would optimize it using (stochastic) gradient descent variants.

MAP is only a point estimate, or dirac posterior approximation

$$p(\boldsymbol{\theta}|y_{1:n}, \boldsymbol{x}_{1:n}) \approx \delta(\boldsymbol{\theta} = \hat{\boldsymbol{\theta}})$$

Multimodal Posterior of BNNs

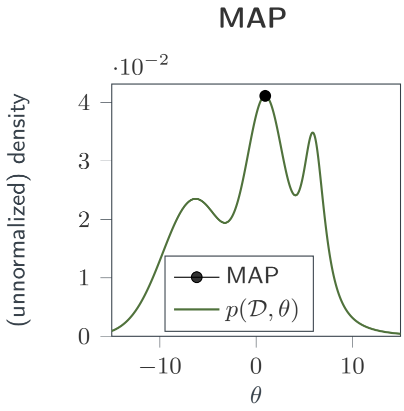
Denote data set by $\mathcal{D}$. Posterior is just normalized joint:

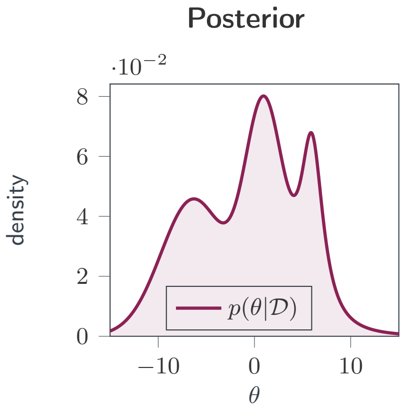


Why do we even care about solving the integral to get a scalar?

Bayesian Inference and Loss Landscapes

Posterior captures the entire loss landscape, MAP a single point.





Inference: Bayesian Posterior Predictive

Posterior predictive for a new input $\boldsymbol{x}^*$ is

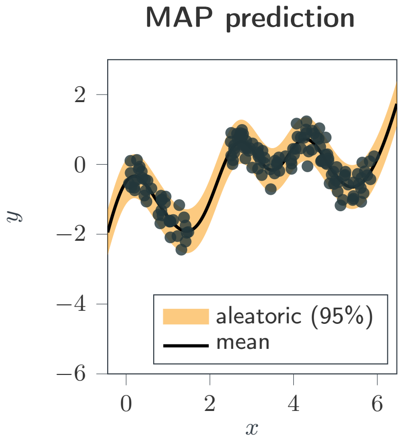
$$p(y^*|\boldsymbol{x}^*, \boldsymbol{x}_{1:n}, y_{1:n}) = \int p(y^*|\boldsymbol{x}^*, \boldsymbol{\theta}) p(\boldsymbol{\theta}|\boldsymbol{x}_{1:n}, y_{1:n}) \, d\boldsymbol{\theta}$$

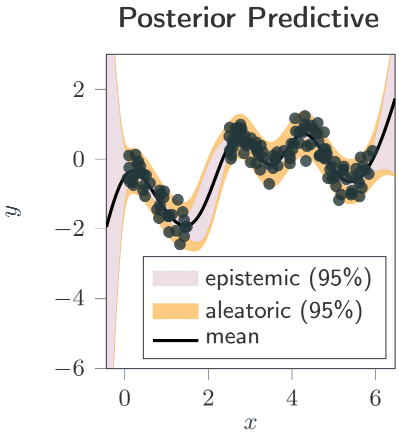
where we marginalized out the posterior.

For the posterior, this forms a *Bayesian model average* of infinitely many models weighted by their posterior probabilities. For the MAP, it is a single point estimate.

Posterior Predictive in Regression

BNN with homoscedastic Gaussian likelihood.





Exam Questions

Which of the following statements about Bayesian Neural Networks (BNNs) are true?

BNNs are neural networks that parameterize a likelihood function and specify a prior distribution over their weights.

If both the likelihood and the prior are Gaussian, MAP inference of the weights of a BNN can be performed in closed form.

Training and evaluating an ordinary neural network with dropout can be interpreted as performing approximate inference on a BNN.

Epistemic uncertainty in BNNs can be reduced by observing more data.

Exam Questions

Which of the following statements about Bayesian Neural Networks (BNNs) are true?

T BNNs are neural networks that parameterize a likelihood function and specify a prior distribution over their weights.

F If both the likelihood and the prior are Gaussian, MAP inference of the weights of a BNN can be performed in closed form.

T Training and evaluating an ordinary neural network with dropout can be interpreted as performing approximate inference on a BNN.

T Epistemic uncertainty in BNNs can be reduced by observing more data.

Intractability of Bayesian Neural Networks

BNNs seem like a good idea, what's holding us back?

Modern neural networks have parameters $\boldsymbol{\theta} \in \mathbb{R}^P$ with P in the order of millions to billions. How do we solve such integrals?

$$p(y_{1:n}|\boldsymbol{x}_{1:n}) = \int p(\boldsymbol{\theta})p(y_{1:n}|\boldsymbol{x}_{1:n}, \boldsymbol{\theta})\, d\boldsymbol{\theta}$$

$\rightarrow$ We need *approximate inference* methods.

Approximate Inference methods for BNNs

Overview of Methods

- Variational Inference
  - Bayes by backprop
  - Variational Gauss-Newton
  - Monte Carlo Dropout
- Laplace approximation
- SWA-Gaussian (SWAG)
- Deep ensembles
- MCMC
  - Stochastic Gradient Langevin Dynamics
  - Hamiltonian Monte Carlo
  - ...

Variational Inference

Variational inference seeks to approximate intractable posterior p by a simple one q that is "as close as possible"

$$p(\boldsymbol{\theta}|\boldsymbol{x}_{1:n}, y_{1:n}) = \frac{1}{Z} p(y_{1:n}, \boldsymbol{\theta}|\boldsymbol{x}_{1:n}) \approx q(\boldsymbol{\theta}; \lambda).$$

The simple distribution q is chosen from a family of distributions and parametrized by λ.

In neural networks, almost always Gaussian $q(\boldsymbol{\theta}; \lambda) = \mathcal{N}(\boldsymbol{\theta}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$.

Variational Inference Recap

We have a variational family $\mathcal{Q}$ that contains distributions $q(\boldsymbol{\theta})$ and want to minimize the KL divergence of q from the posterior $p(\boldsymbol{\theta}|\mathcal{D})$:

$$\begin{aligned}0 &\leq \min_{q\in\mathcal{Q}} D_{\mathrm{KL}}[q(\boldsymbol{\theta})\|p(\boldsymbol{\theta}|\mathcal{D})] = \min_{q\in\mathcal{Q}} \int q(\boldsymbol{\theta}) \log \frac{q(\boldsymbol{\theta})}{p(\boldsymbol{\theta}|\mathcal{D})} \mathrm{d}\boldsymbol{\theta} \\ &= \min_{q\in\mathcal{Q}} \int q(\boldsymbol{\theta}) \log \frac{q(\boldsymbol{\theta})p(\mathcal{D})}{p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})} \mathrm{d}\boldsymbol{\theta} = \min_{q\in\mathcal{Q}} \int q(\boldsymbol{\theta}) \log \frac{q(\boldsymbol{\theta})}{p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})} \mathrm{d}\boldsymbol{\theta} + \log p(\mathcal{D})\end{aligned}$$

We can equivalently maximize the evidence lower bound (ELBO):

$$\begin{aligned}\log p(\mathcal{D}) \geq \max_{q\in\mathcal{Q}} \mathrm{ELBO}(q) &= \max_{q\in\mathcal{Q}} \mathbb{E}_q[\log p(\mathcal{D},\boldsymbol{\theta})] + H[q] \\ &= \max_{q\in\mathcal{Q}} \mathbb{E}_q[\log p(\mathcal{D}|\boldsymbol{\theta})] - D_{\mathrm{KL}}[q(\boldsymbol{\theta})\|p(\boldsymbol{\theta})]\end{aligned}$$

Depending on the choice of prior and q, different forms of the ELBO are used.

The Gaussian Variational Approximation

Using $\mathcal{Q}$ as the set of Gaussian distributions characterized by valid mean and covariance parameters is the most common choice.

$$\begin{aligned}\log p(\mathcal{D}) \geq \max_{(\boldsymbol{\mu},\boldsymbol{\Sigma})} \mathrm{ELBO}(\boldsymbol{\mu},\boldsymbol{\Sigma}) &= \mathbb{E}_{\mathcal{N}(\boldsymbol{\theta};\boldsymbol{\mu},\boldsymbol{\Sigma})}[\log p(\mathcal{D},\boldsymbol{\theta})] + H[\mathcal{N}(\boldsymbol{\mu},\boldsymbol{\Sigma})] \\ &= \mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\mathbf{I})}[\log p(\mathcal{D},\boldsymbol{\mu}+\boldsymbol{\Sigma}^{1/2}\boldsymbol{\epsilon})] + \frac{1}{2}\log|\boldsymbol{\Sigma}| + \frac{D}{2}\log 2\pi e\end{aligned}$$

The last equality is due to the *reparameterization trick*.

Bayes-by-Backprop (Blundell et al. [2015]) uses a diagonal Gaussian approximation and maximizes the ELBO using backprop (autodiff).

Bayes by Backprop Algorithm

Algorithm 1 Bayes by backprop

```
**Input:** $\lambda = \lambda_0 = \{\boldsymbol{\mu}_0, \boldsymbol{\sigma}_0\}$
**for** t = 1,2,... **do**
    Draw $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
    $\boldsymbol{\theta} = t(\epsilon, \lambda) = \boldsymbol{\mu} + \boldsymbol{\sigma} \circ \epsilon$
    $\mathcal{L}(\lambda) = \log q(\boldsymbol{\theta}; \lambda) - \log p(\boldsymbol{\theta}) - \log p(\mathcal{D}|\boldsymbol{\theta})$
    $\nabla_\lambda \mathcal{L} = \mathrm{backprop}_\lambda(\mathcal{L})$
    $\lambda = \lambda - \alpha \nabla_\lambda \mathcal{L}$
**end for**
```

Can minibatch and reduce variance by using multiple ϵ samples.
Due to change in training, can perform worse than standard NNs.

Exam Question

Let $\sigma(x) = \max(0, x)$ denote the ReLU function, and let $w > 0$. Using the reparametrization trick, calculate

$$\nabla_\mu \mathbb{E}_{x\sim\mathcal{N}(\mu,1)} \sigma(wx).$$

We denote the CDF of a normal Gaussian random variable $x \sim \mathcal{N}(0, 1)$ by $F(a) = \mathbb{P}(x \leq a)$.

$$\begin{aligned}
\mathbb{E}_\epsilon \nabla_\mu f(x) &= \mathbb{E}_\epsilon w \sigma^{'}(w(\mu + \epsilon)) \\
&= w \mathbb{E}_\epsilon \mathbf{1}(w(\mu + \epsilon) > 0) \\
&= w \mathbb{E}_\epsilon \mathbf{1}(\mu + \epsilon > 0) \\
&= w \mathbb{P}(\epsilon > -\mu) \\
&= w(1 - \mathbb{P}(\epsilon \leq -\mu)) = w(1 - F(-\mu))
\end{aligned}$$

Laplace Approximation

The Laplace approximation is a Gaussian constructed using a second-order Taylor approximation at a mode of the posterior, $\hat{\boldsymbol{\theta}}$. With Hessian $\mathbf{H}_{\hat{\boldsymbol{\theta}}} = \nabla^2_{\boldsymbol{\theta}} \log p(\mathcal{D}, \boldsymbol{\theta})|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}}$, we have

$$\log p(\mathcal{D}, \boldsymbol{\theta}) \approx \log p(\mathcal{D}, \hat{\boldsymbol{\theta}}) + \tfrac{1}{2}(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}})^{\mathsf{T}} \mathbf{H}_{\hat{\boldsymbol{\theta}}} (\boldsymbol{\theta} - \hat{\boldsymbol{\theta}})$$

Where is the first-order term? We are at the MAP so the gradient is zero! We have a simple Gaussian integral for the marginal likelihood:

$$\begin{aligned} p(\mathcal{D}) &\approx \int \exp\left[\log p(\mathcal{D}, \hat{\boldsymbol{\theta}}) + \tfrac{1}{2}(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}})^{\mathsf{T}} \mathbf{H}_{\hat{\boldsymbol{\theta}}} (\boldsymbol{\theta} - \hat{\boldsymbol{\theta}})\right] \mathrm{d}\boldsymbol{\theta} \\ &= p(\mathcal{D}, \hat{\boldsymbol{\theta}})(2\pi)^{\frac{D}{2}} |-\mathbf{H}_{\hat{\boldsymbol{\theta}}}|^{-\frac{1}{2}}. \end{aligned}$$

$\rightarrow$ The posterior approximation is the Gaussian $\mathcal{N}(\hat{\boldsymbol{\theta}}, \mathbf{H}_{\hat{\boldsymbol{\theta}}}^{-1})$.

Modern Laplace Approximations

The Hessian $\mathbf{H}_{\hat{\boldsymbol{\theta}}}$ is a $P \times P$ matrix that we can neither compute nor store for large P and might be indefinite.

A simple option is to use only the diagonal Hessian, which gives us a diagonal Gaussian posterior approximation like Bayes-by-Backprop.

But we can do much better than that using structured Hessian approximations like KFAC (Martens and Grosse [2015]).

Hessian Approximations

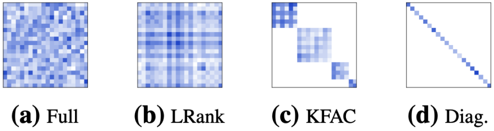


Figure from Daxberger et al. [2021].

Curvature and the Relationship between Laplace and VI

Laplace and Gaussian variational approximations depend on an understanding of the curvature using the Hessian. VI has a more global notion due to the expectation/sampling.

Laplace

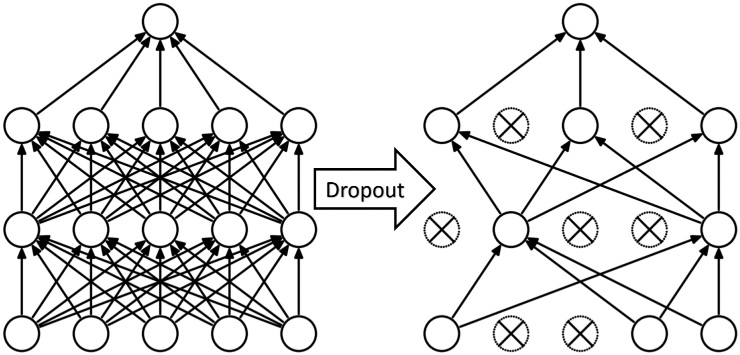
$$0 = \nabla_{\boldsymbol{\theta}} \log p(\mathcal{D}, \boldsymbol{\theta})|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}}$$
$$\boldsymbol{\Sigma}^{-1} = -\nabla^2_{\boldsymbol{\theta}} \log p(\mathcal{D}, \boldsymbol{\theta})|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}}$$

Gaussian VI

$$0 = \mathbb{E}_{q(\boldsymbol{\theta})}[\nabla_{\boldsymbol{\theta}} \log p(\mathcal{D}, \boldsymbol{\theta})]$$
$$\boldsymbol{\Sigma}^{-1} = -\mathbb{E}_{q(\boldsymbol{\theta})}[\nabla^2_{\boldsymbol{\theta}} \log p(\mathcal{D}, \boldsymbol{\theta})]$$

The key difference: Laplace is *post-hoc* and VI requires special optimization. See homework for proof.

Dropout



Edited figure from Srivastava et al. [2014].

Monte Carlo Dropout

Perform dropout not only during training but also during inference.

Can be seen as performing variational inference with a particular family

$$q(\boldsymbol{\theta}|\lambda) = \prod_j q_j(\theta_j|\lambda_j)$$

where $q_j(\theta_j|\lambda_j) = p\delta_0(\theta_j) + (1-p)\delta_{\lambda_j}(\theta_j)$, p is the probability that weight θ_j is set to 0 and λ_j is the value of the weight.

Predictive uncertainty is approximated by averaging across samples

$$p(y^*|\boldsymbol{x}^*, \boldsymbol{x}_{1:n}, y_{1:n}) \approx \frac{1}{m}\sum_{j=1}^{m} p(y^*|\boldsymbol{x}^*, \boldsymbol{\theta}^{(j)})$$

where $\boldsymbol{\theta}^j$ corresponds neural network with certain weights set to 0.

Deep Ensembles

Obtain MAP estimate for several neural networks, each separately trained on a different dataset obtained via bootstrap sampling (uniform sampling with replacement).

Average outputs of different models to approximate predictive. Can also combine this with variational inference, Laplace, and SWAG to get mixture of Gaussian posterior approximations!

Markov Chain Monte Carlo (MCMC)

MCMC methods produce a sequence of iterates (NN) weights $\boldsymbol{\theta}^{(1)}, \dots, \boldsymbol{\theta}^{(T)}$ which by the ergodic theorem allow the estimation of the marginal predictive distribution as

$$p(y^*|\boldsymbol{x}^*, \boldsymbol{x}_{1:n}, y_{1:n}) \approx \frac{1}{m} \sum_{j=1}^{m} p(y^*|\boldsymbol{x}^*, \boldsymbol{\theta}^{(j)}).$$

For modern neural networks, this can become intractable as storing multiple parameters can already be too expensive!

SGLD

SGLD produces samples from Bayesian posterior $\boldsymbol{\theta} \sim \frac{1}{Z} \exp(\log p(\boldsymbol{\theta}) + \sum_{i=1}^{n} \log p(y_i|x_i, \boldsymbol{\theta}))$ as follows

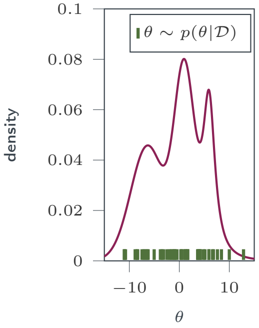
Algorithm 2 Stochastic Gradient Langevin Dynamics

```
Input: $\boldsymbol{\theta} = \boldsymbol{\theta}_0$
for t = 1,2,... do
    $i_i, \ldots, i_m \sim \mathrm{Uniform}(\{1, \ldots, n\})$
    $\epsilon_t \sim \mathcal{N}(0, 2\eta_t I)$
    $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta_t \left(\nabla \log p(\boldsymbol{\theta}_t) + \frac{n}{m} \sum_{j=1}^{m} \nabla \log p(y_{i,j}|\boldsymbol{\theta}_t, \boldsymbol{x}_{i,j})\right) + \epsilon_t$
end for
```

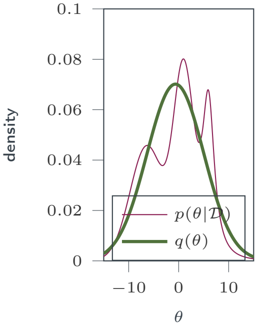
Due to large number of weights, drop samples during "burn-in" period and summarize later samples via subsampling.

Comparison of Approximate Inference Methods

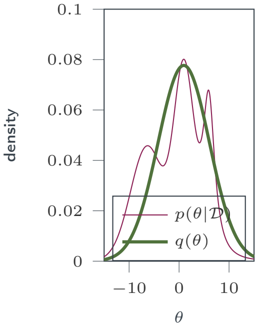
MCMC



Variational



Laplace



Calibration

Calibration

Confidence calibration is the problem of predicting probability estimates representative of the true correctness likelihood.

Recommendation systems: one likes horror movies(60%), comedy(40%), so recommend horror movies only, right?

Ad clicks: users will click on ads A(20%), B(10%), can we trust these numbers for total clicks in reality?

Reliability diagrams

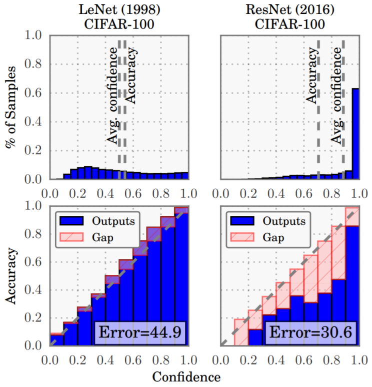
Reliability diagrams plot expected sample accuracy as a function of confidence.

- Group predictions into M interval bins (each of size $1/M$)
- Calculate relative frequency of positive samples for each bin
  (B_m set of samples falling into bin m)

$$\mathrm{freq}(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} \mathbf{1}(\hat{y}_i = 1)$$

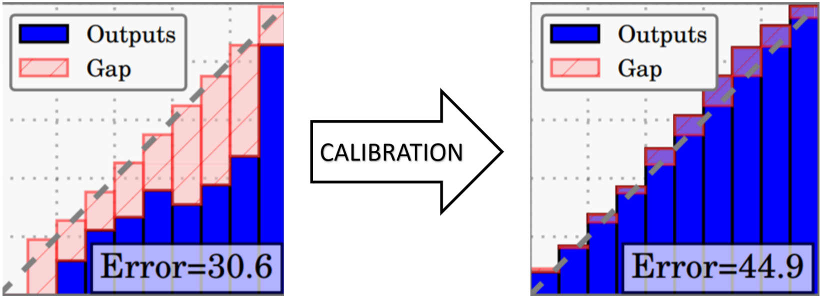
- Calculate average confidence within bin

$$\mathrm{conf}(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} \hat{p}_i$$



Guo et al. [2017]

The Goal of Calibration



Calibration methods (I)

Can empirically improve accuracy of calibration via heuristics:

- Histogram binning: Divide uncalibrated predictions $\hat{p}_i$ into bins $B_1, \ldots, B_M$. Assign calibrated score to each bin $\hat{q}_i = \mathrm{freq}(B_m)$.
- Isotonic regression: Find piecewise constant function f, $\hat{q}_i = f(\hat{p}_i)$, that minimizes the bin-wise squared loss (θ_i: assigned value, a_i: bin boundary):

$$\min_{M,\theta_1,\ldots,\theta_M,a_1,\ldots,a_{M+1}} \sum_{m=1}^{M} \sum_{i=1}^{n} \mathbf{1}(a_m \le \hat{p}_i < a_{m+1})(\theta_m - y_i)^2$$

subject to $\begin{array}{l} 0 = a_1 \le a_2 \le \ldots \le a_{M+1} = 1 \\ \theta_1 \le \theta_2 \le \ldots \le \theta_M \end{array}$.

Calibration methods (II)

- Platt scaling: Learn parameters $a, b \in \mathbb{R}$ that minimize the NLL loss over the validation set when applied to the logits z_i, $\hat{q}_i = \sigma(az_i + b)$.
- Temperature scaling is a special case with $b = 0$ and $a = 1/T$. Higher temperature $\rightarrow$ softer probabilities, lower temperature $\rightarrow$ sharper probabilities.
- Bayesian neural networks often improve the calibration in practice without any heuristics applied.

Two Types of Uncertainty

Epistemic & Aleatoric Uncertainty

<u>Epistemic Uncertainty</u>:
Model uncertainty that corresponds to uncertainty in model parameters and can be explained away given enough data.

<u>Aleatoric Uncertainty</u>:
Inherent noise in observations.

Homoscedastic uncertainty: constant for different inputs.

Heteroscedastic uncertainty: varies with different inputs.

Regression

BNN weights are sampled from a learned posterior distribution $\boldsymbol{\theta} \sim q(\boldsymbol{\theta}|\lambda)$. Predicted mean $\mu\left(\boldsymbol{x}, \boldsymbol{\theta}\right)$ and variance $\sigma\left(\boldsymbol{x}, \boldsymbol{\theta}\right)^2$ describe the output distribution.

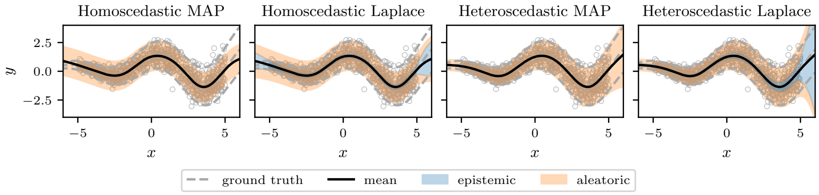
Mean predicted output is obtained as

$$\mathbb{E}\left[y^* | \boldsymbol{x}^*\right] \approx \bar{\mu}\left(\boldsymbol{x}^*\right) := \frac{1}{m} \sum_{j=1}^{m} \mu\left(\boldsymbol{x}^*, \boldsymbol{\theta}^{(j)}\right).$$

$$\begin{aligned}\operatorname{Var}\left[y^* | \boldsymbol{x}^*\right] &= \mathbb{E}[\operatorname{Var}[y^* | \boldsymbol{x}^*, \boldsymbol{\theta}]] + \operatorname{Var}[\mathbb{E}[y^* | \boldsymbol{x}^*, \boldsymbol{\theta}]] \\ &\approx \underbrace{\frac{1}{m} \sum_{j=1}^{m} \sigma^2\left(\boldsymbol{x}^*, \boldsymbol{\theta}^{(j)}\right)}_{\text{aleatoric uncertainty}} + \underbrace{\frac{1}{m} \sum_{j=1}^{m} \left(\mu\left(\boldsymbol{x}^*, \boldsymbol{\theta}^{(j)}\right) - \bar{\mu}\left(\boldsymbol{x}^*\right)\right)^2}_{\text{epistemic uncertainty}}\end{aligned}$$

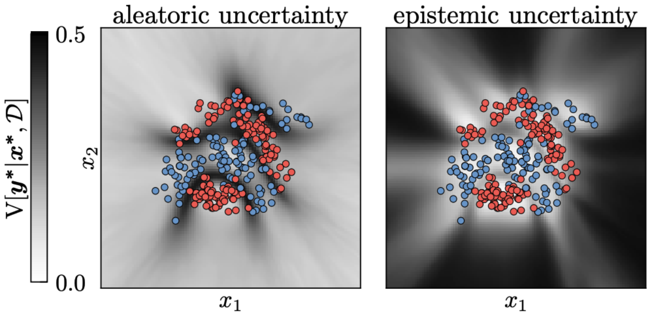
Calibration extends to confidence intervals (Kuleshov et al. [2018]).

Variants of Regression Predictives



Posterior predictive distribution for regression (Immer et al. [2023]).

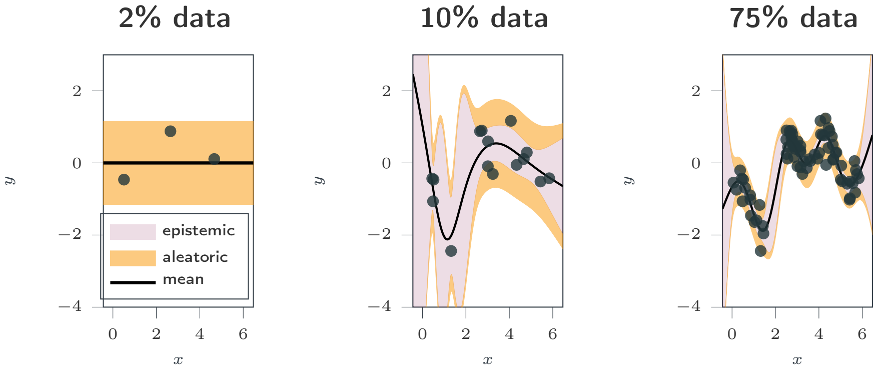
Epistemic and Aleatoric Decomposition in Classification



Decomposition of uncertainties in classification.

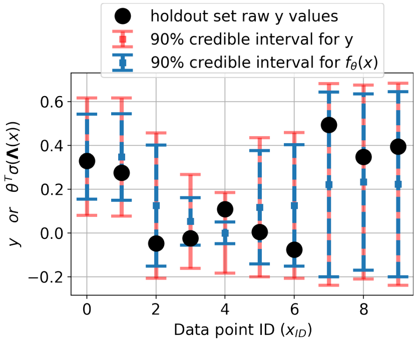
Epistemic and Aleatoric Uncertainties

Uncertainties depend on the data and the model.



A linear model would explain non-linearity with as noise.

Exam Questions



Calibration for regression

Empirical coverage for y...?
Empirical coverage for $f_\theta(x)$...?
any trival solution to increase coverage / calibration?

Summary

Summary

- The posterior of Bayesian neural networks requires to capture the entire loss landscape.
- Bayesian neural networks perform Bayesian model averaging to arrive at predictions that better reflect our believes.
- Approximate inference allow us to estimate the posterior and predictive distribution.
- Calibration can help to align the probability estimates with their true correctness likelihood.
- Bayesian neural networks allow to quantify not only aleatoric but also epistemic uncertainties.

References

C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra. Weight uncertainty in neural network. In *International Conference on Machine Learning*, pages 1613–1622. PMLR, 2015.

E. Daxberger, A. Kristiadi, A. Immer, R. Eschenhagen, M. Bauer, and P. Hennig. Laplace redux-effortless bayesian deep learning. *Advances in Neural Information Processing Systems*, 34: 20089–20103, 2021.

C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger. On calibration of modern neural networks. In *International Conference on Machine Learning*, pages 1321–1330. PMLR, 2017.

A. Immer, E. Palumbo, A. Marx, and J. Vogt. In *Neural Information Processing Systems (NeurIPS)*, 2023.

L. V. Jospin, W. Buntine, F. Boussaid, H. Laga, and M. Bennamoun. Hands-on bayesian neural networks–a tutorial for deep learning users. *arXiv preprint arXiv:2007.06823*, 2020.

V. Kuleshov, N. Fenner, and S. Ermon. Accurate uncertainties for deep learning using calibrated regression. In *International Conference on Machine Learning*, pages 2796–2804. PMLR, 2018.

J. Martens and R. Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *International conference on machine learning*, pages 2408–2417. PMLR, 2015.

N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1): 1929–1958, 2014.