THE UNIVERSITY *of* EDINBURGH

Lecture 3: Bayesian Neural Networks

Dr Sara Wade
sara.wade@ed.ac.uk

20-23 May, 2024

Outline

1. Introduction

2. Neural Networks

3. Bayesian Neural Networks

Outline

1. Introduction
2. Neural Networks
3. Bayesian Neural Networks

Introduction

- Neural networks have a dominant position in AI and have achieved remarkable success in various domains.
- However, they are hindered by limitations; such as, overconfident prediction, poor uncertainty estimation, lack of interpretability [1].
- Regularization in neural networks is necessary due to their highly overparameterized nature; and thus, can be naturally interpreted as MAP estimates in Bayesian framework.
- Moreover, Bayesian neural networks (BNNs) have emerged as a compelling extension, integrating uncertainty estimation into their predictive capabilities.

[1] Szegedy et al (2013); Nguyen et al (2015); Carbone et al (2021); Roskams-Hieter et al (2023)

Outline

1. Introduction
2. Neural Networks
3. Bayesian Neural Networks

Neural Networks: Introduction

- Originally inspired by the structure and function of neurons in the brain to enable machines to acquire human-like skills.
- The first rose in popularity in the 1980s and resurfaced after 2010, with the advent of modern computing and big data, and increased more recently with large language models.

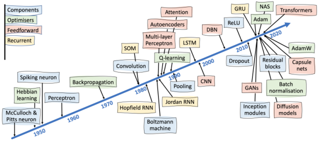


Figure: rough history of neural networks and deep learning from Lones (2024)

From Basis Expansions to Neural Networks

Recall: basis function expansion is a simple way to increase flexibility by replacing $\boldsymbol{x}$ with nonlinear terms $\boldsymbol{\phi}(\boldsymbol{x})$ in a linear model:

$$f(\boldsymbol{x};\boldsymbol{\theta}) = \mathbf{W}\boldsymbol{\phi}(\boldsymbol{x}) + \boldsymbol{b}.$$

$\rightarrow$ linear in $\boldsymbol{\theta} = (\mathbf{W}, \boldsymbol{b})$, but specifying the basis functions is limiting.

This can be extended by endowing the feature extractors with parameters $\boldsymbol{\theta}_2$:

$$f(\boldsymbol{x};\boldsymbol{\theta}) = \mathbf{W}\boldsymbol{\phi}(\boldsymbol{x};\boldsymbol{\theta}_2) + \boldsymbol{b},$$

with $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \boldsymbol{\theta}_2)$ and $\boldsymbol{\theta}_1 = (\mathbf{W}, \boldsymbol{b})$.

Repeating this process recursively creates more complex functions, which is the basic idea behind deep neural networks:

$$f(\boldsymbol{x};\boldsymbol{\theta}) = f_L(f_{L-1}(\cdots f_1(\boldsymbol{x}))),$$

where $f_l(\boldsymbol{x};\boldsymbol{\theta}_l)$ is the function at layer l.

Neural Networks

Neural networks are hierarchical models made of layers: an input layer, one or more hidden layers, and an output layer.

$$\text{input layer } \mathbf{x} \to \underbrace{\mathbf{z}_1 \to \cdots \to \mathbf{z}_L}_{L \text{ hidden layers}} \to \text{output layer } \mathbf{y}.$$

Define the hidden units $\mathbf{z}_l$ at each layer l as a linear transformation of the hidden units at the previous layer passed elementwise through an activation function:

$$\mathbf{z}_l = f_l(\mathbf{z}_{l-1}) = \psi_l(\boldsymbol{b}_l + \mathbf{W}_l \mathbf{z}_{l-1}).$$

Note: in general the activation function may vary across layers.

Neural Networks

- The activation function $\psi : \mathbb{R} \to \mathbb{R}$ is a nonlinear function that transforms the linear combination of the previous layer's units.
- The depth is the number of layers L, and the width of the layer l, denoted D_l, is the number of hidden units for the layer l (with $D_0 =$ the dimension of $\boldsymbol{x}$ and $D_{L+1} =$ the dimension of $\mathbf{y}$).
- The vector of units before applying non-linearity is called the pre-activation: $\mathbf{a}_l = \boldsymbol{b}_l + \mathbf{W}_l \mathbf{z}_{l-1}$, with $\mathbf{z}_l = \psi_l(\mathbf{a}_l)$.
- The parameters of each layer consist of the weight matrix $\mathbf{W}_l \in \mathbb{R}^{D_l \times D_{l-1}}$ and the bias vector $\boldsymbol{b}_l \in \mathbb{R}^{D_l}$.
- Let $\boldsymbol{\theta} = (\mathbf{W}_1, \boldsymbol{b}_1, \ldots, \mathbf{W}_{L+1}, \boldsymbol{b}_{L+1})$. 🤔 What is the dimension of $\boldsymbol{\theta}$?

Model Structure - I

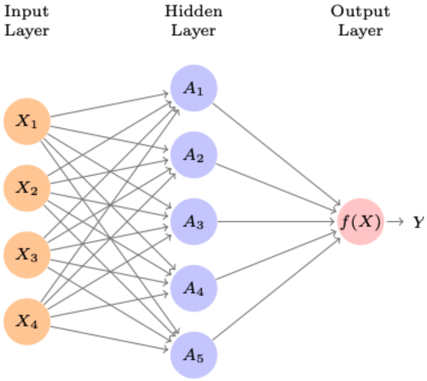


Figure: Simple feed-forward neural network with a single hidden layer. From *Elements of Statistical Learning*.

Model Structure - II

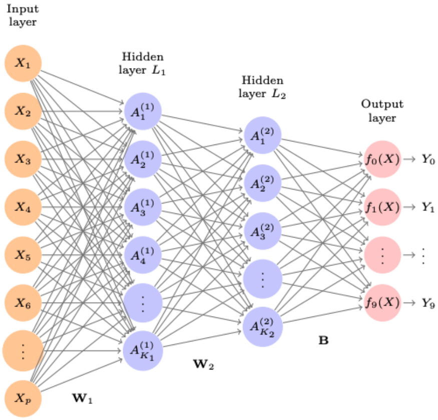


Figure: Simple feed-forward neural network with two hidden layers and multiple outputs. From *Elements of Statistical Learning*.

Activations

We are free to choose any kind of activation function at each layer, but differentiable functions make training easier.

🤔 Why not use a linear activation?

Two choices are the sigmoid and tanh functions. However, they saturate for large inputs leading to vanishing gradients, making model training via gradient descent hard.

The most common non-saturating activation function is the rectified linear unit (ReLU) defined as $\psi(a) = \max(a, 0)$.

Activations

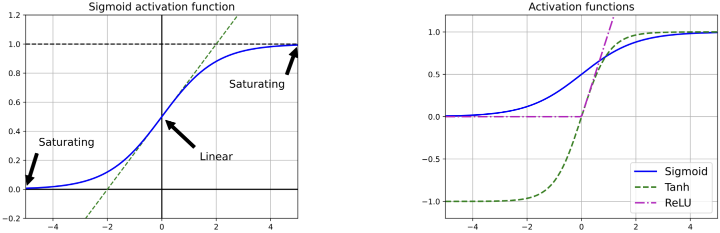


Figure: Some neural network activation functions. From *Probabilistic Machine Learning*.

Training

Given the training data, we seek the parameters which minimize the empirical risk:

$$\widehat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \frac{1}{N} \sum_{n=1}^{N} \mathcal{L}(f(\mathbf{x}_n), \mathbf{y}_n),$$

where $\mathcal{L}$ is a chosen loss (i.e. negative log likelihood), e.g. the squared error loss is common in regression tasks.

The main workhorse in neural network training is gradient-based optimization:

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \eta_t \nabla_{\boldsymbol{\theta}} \mathcal{R},$$

where $\mathcal{R}$ denotes the empirical risk and η_t denotes the step size or learning rate.

Backpropagation

Gradients are computed as products of gradients between each layer from right to left, known as backpropagration, i.e. the chain rule for efficient computation.

Example: consider a single hidden layer and squared error loss, with $D_0 = D_{L+1} = 1$:

$$\mathcal{R} = ||\mathbf{y} - \boldsymbol{b}_2 - \mathbf{W}_2 \psi(\boldsymbol{b}_1 + \mathbf{W}_1 \boldsymbol{x})||^2.$$

$$\begin{aligned}
\frac{\partial \mathcal{R}}{\partial \boldsymbol{\theta}_2} &= \frac{\partial \mathcal{R}}{\partial \mathbf{a}_2} \frac{\partial \mathbf{a}_2}{\partial \boldsymbol{\theta}_2} \\
\frac{\partial \mathcal{R}}{\partial \boldsymbol{\theta}_1} &= \frac{\partial \mathcal{R}}{\partial \mathbf{a}_2} \frac{\partial \mathbf{a}_2}{\partial \mathbf{z}_1} \frac{\partial \mathbf{z}_1}{\partial \mathbf{a}_1} \frac{\partial \mathbf{a}_1}{\partial \boldsymbol{\theta}_1}
\end{aligned}$$

⇒ can be computed recursively![2]

[2] See 13.3 of *Probabilistic Machine Learning*

Identifiability and Issues

Consider the NN:

$$\widehat{y} = b_2 + \mathbf{W}_2 \mathbf{z}_1 = b_2 + \sum_{d=1}^{D_1} w_{2,d} z_{1,d},$$

$$z_{1,d} = \psi(\boldsymbol{b}_{1,d} + \boldsymbol{w}_{1,d}\mathbf{x}) = \psi(b_{1,d} + \sum_{d'=1}^{D_0} w_{1,d,d'} x_{1,d'}).$$

🤔 What are potential sources of non or weak identifiability?

⇒ Leads to complicated, multi-modal objective!

Practical issues: nonconvex objective, and gradient-based optimization suffers from issues due to initialization and vanishing and exploding gradients.[3]

[3] For practical solutions, see 13.3 of *Probabilistic Machine Learning*

Regularization

Regularization is crucial to avoid overfitting as NNs can have millions of parameters.

A common choice is ℓ_2 regularization[4], as in ridge regression, and is called weight decay. In regression with the squared error loss and $D_{L+1} = 1$:

$$\widehat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \, ||\mathbf{y} - f(\mathbf{x}; \boldsymbol{\theta})||^2 + \lambda_w \sum_{l=1}^{L+1} \sum_{d=1}^{D_l} \sum_{d'=1}^{D_{l-1}} w_{l,d,d'}^2 + \lambda_b \sum_{l=1}^{L+1} \sum_{d=1}^{D_l} b_{l,d}^2$$

$\Rightarrow$ This can be naturally interpreted as MAP estimation with Gaussian priors:

$$\boldsymbol{w}_{l,d} \sim \mathrm{N}(0, s_w^2 \mathbf{I}), \quad \boldsymbol{b}_l \sim \mathrm{N}(0, s_b^2 \mathbf{I}).$$

[4] other choices include early stopping, dropout, batch normalization.

Regularization

Sparse NNs encourage sparsity in the weights, allowing model compression, particularly useful for highly-parametrized NNs.

A common choice is ℓ_1 regularization[5], as in lasso regression. In regression with the squared error loss and $D_{L+1} = 1$:

$$\widehat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} ||\mathbf{y} - f(\mathbf{x}; \boldsymbol{\theta})||^2 + \lambda_w \sum_{l=1}^{L+1} \sum_{d=1}^{D_l} \sum_{d'=1}^{D_{l-1}} |w_{l,d,d'}| + \lambda_b \sum_{l=1}^{L+1} \sum_{d=1}^{D_l} b_{l,d}^2$$

⇒ This can be naturally interpreted as MAP estimation with Laplace priors on the weights and Gaussian priors on the bias:

$$\boldsymbol{w}_{l,d,d'} \sim \text{Laplace}(0, s_w), \quad \boldsymbol{b}_l \sim \text{N}(0, s_b^2 \mathbf{I}).$$

[5] see Hoefler et al (2021) for a review

Limitations

- Miscalibrated and/or overconfident predictions (Minderer et al 2021)
- Non-robustness to out-of-distribution samples (Lee at al 2018, ...) and sensitivity to domain shifts (Ovadia et al (2019))
- sensitivity to adversarial attacks (Moosavi-Dezfooli et al (2016), ...)
- poor interpretability (Sundararajan et al 2017)
- poor understanding of generalization (McAllester (1999); Dziugaite and Roy (2017))

Outline

1. Introduction
2. Neural Networks
3. Bayesian Neural Networks

Bayesian Neural Networks

Neural networks are popular due to expressiveness and generalization abilities, while Bayesian inference is heralded for robustness and uncertainty quantification ⇒ BNNs combine these to benefit from the advantages of both.

Enables uncertainty in the predictive distribution:

$$p(\mathbf{y} \mid \mathbf{x}, \mathcal{D}) = \underbrace{\int p(\mathbf{y} \mid \mathbf{x}, \boldsymbol{\theta}) \quad \underbrace{p(\boldsymbol{\theta} \mid \mathcal{D})}_{\text{uncertainty in } \boldsymbol{\theta}} \quad d\boldsymbol{\theta}}_{\text{infinite ensemble of NNs}} \,.$$

However, unique challenges with BNNs, particularly prior specification and inference algorithms.

Prior Specification

Specifying sensible priors is challenging due to:

- high-dimensional parameter and over-parametrization of the model,
- difficulty in understanding how the weights map to the functions.

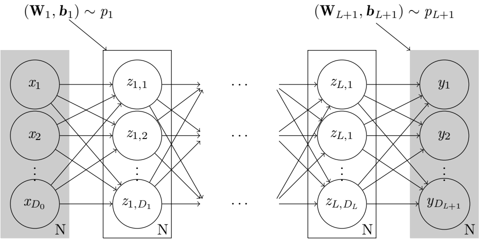


Figure: BNN architecture, where the weights and biases at layer l have some prior p_l.

Prior Specification

Priors[6] employed include:

- Gaussian priors are a common, default choice, partly due to their equivalence to MAP estimation with ℓ_2 regularization.
- Laplace or more generally sparsity promoting priors, for sparse BNNs.
- Predictive complexity priors constructed hierarchically to penalize more complex models (Nalisnick et al (2021)).

[6] see Fortuin et al (2022) for a review

Connections with Gaussian Processes (GPs)

Neal (1996) first connected GPs with BNNs. Let's first briefly introduce GPs.

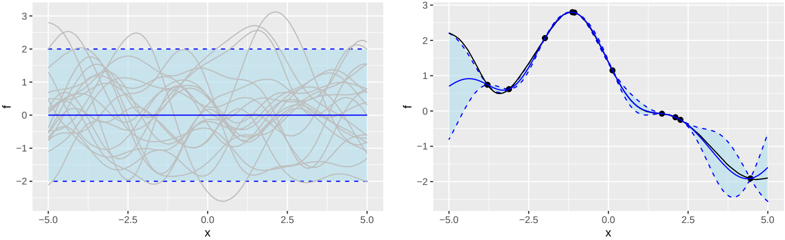
Definition 1

A **Gaussian process** (GP) is an infinite collection of random variables $(f(\boldsymbol{x}) : \boldsymbol{x} \in \mathbb{R}^D)$, where any finite number have Gaussian distribution with consistent parameters.

It is fully specified by:

- mean function $\mu(\boldsymbol{x}) = \mathrm{E}[f(\boldsymbol{x})]$
- covariance function $C_{\boldsymbol{\phi}}(\mathbf{x}, \mathbf{x}') = \mathrm{Cov}(f(\boldsymbol{x}), f(\boldsymbol{x}'))$

Gaussian Processes



- In Bayesian nonparametrics, GPs are used to specify a prior on an unknown function.
- As a prior, GPs satisfy key desiderata: interpretability, large support, tractability.

Connections with Gaussian Processes

Applying the CLT, Neal (1996) showed that a BNN with a single hidden layer and appropriately scaled weight variances converges to a Gaussian process in the infinite-width limit.

This led to the rise of GPs in ML and further extensions

- refining the results for infinite limits and for more general architectures, motivating new covariance functions,
- Gaussian process (GP) inducing priors which tune the prior on the weights by minimizing the divergence between the BNN functional priors and a desired GP.

Variational Inference

Inference in BNNs is challenging, due to them being highly nonlinear functions, leading to high-dimensional, multimodal posteriors.

Variational approximations typically assume a Gaussian variational posterior, combining SVI with the reparametrization trick for Gaussians (Kingma and Welling (2014)).

Often the mean-field assumption is considered (e.g. diagonal covariance); however this is too restrictive for reliable posterior approximation, and more expressive variational posteriors are an active area of research.

Moreover, variational methods suffer from mode collapse, which may also be problematic for BNNs.

Variational Inference

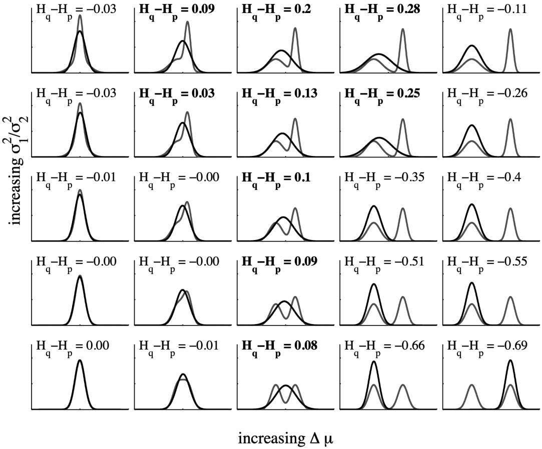


Figure: From Turner & Sahani (2010): illustrating mode collapse when approximating a bimodal posterior with a Gaussian variational posterior.

MCMC

MCMC/HMC and stochastic-gradient-MCMC methods often result in state-of-the-art results, wrt test error (Izmailov et al 2021).

However, this comes significant adidtional computation and storage costs compared to variational inference and other approximate schemes.

Moreover, MCMC also suffers from issues such as multimodality, high curvatures, and saddle points in the posterior. Thus, algorithms that can more effectively search this landscape are also an active area of research.

Summary

- In certain industries, the acceptance of AI algorithms necessitates explanations:
  - medical diagnosis and treatment, where doctors and patients may hesitate to use AI if they cannot explain the rationale behind its outcomes.
  - finance, where biased outputs can cause reputational damage and even legal implications.
- The Bayesian perspective provides a path towards more trustworthy AI, by expressing and quantifying uncertainty in the models; however, this incurs significant computational cost, necessitating improved approximation techniques.

Further Reading

See various references within, including:

- *A primer on Bayesian neural networks: Review and debates*, Arbel et al (2023)
- *Bayesian deep learning in the age of large-scale AI*, Papamerkou et al (2024)
- *Hands-on Bayesian neural networks - A tutorial for deep learning users*, Jospin et al (2022)

ProbAI

Prob_AI

University of BRISTOL

UNIVERSITY OF CAMBRIDGE

THE UNIVERSITY of EDINBURGH

Lancaster University

MANCHESTER 1824
The University of Manchester

WARWICK

UK Research and Innovation

PhD & Postdoc opportunities and events:
https://www.probai.uk