COMP 64101
Reasoning and Learning under Uncertainty

Omar Rivasplata

Department of Computer Science, University of Manchester
Manchester Centre for AI Fundamentals

Lecture 4 - Part 1

Topics: Bayesian Neural Networks

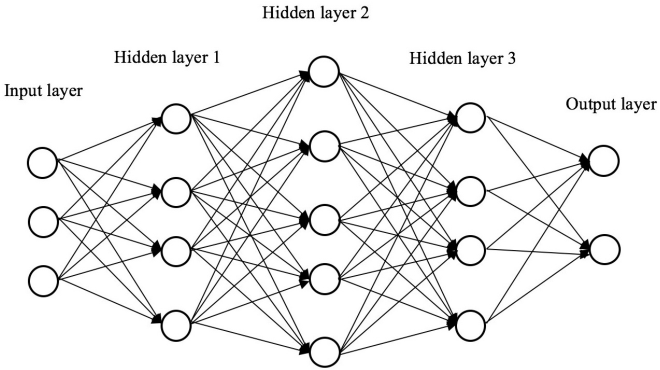
I know it's [Tuesday]. It's a good day for math!

–Inspired by Max Mintz, UPenn.

- Intro (§ 17.1), NNs, BNNs
- Priors for BNNs (§ 17.2)
- Posteriors for BNNs (§ 17.3)
- Generalization in Bayesian deep learning (§ 17.4)
- Online inference (§ 17.5), Hierarchical BNNs (§ 17.6)

Nota bene: Section numbers refer to Murphy (2023).

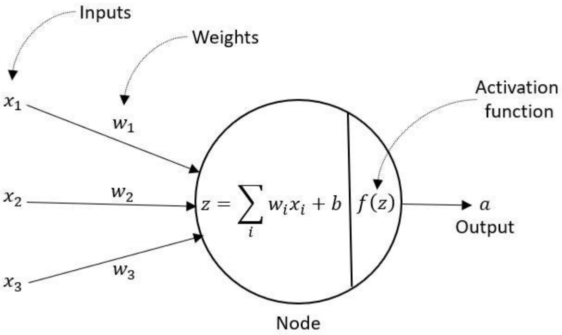
(Pre-) Introduction: Neural Networks



- Neural networks (NNs) are are computational models defined via successive compositions of linear and non-linear functions.
- The classical NN model corresponds to a graph consisting of nodes and a set of edges that link some pairs of nodes.

(Pre-) Introduction: A single neural unit

- The figure shows the computation at a single node of the graph:



- This computational model is called a **perceptron**, and it can be considered to be the basic building block for neural networks.

(Pre-) Introduction: A single neural network layer

- A single 'neural network layer' is a mapping $\mathbb{R}^{d_{\ell-1}} \to \mathbb{R}^{d_\ell}$ formed by
  - **linear operations:** some (affine) linear mapping defined by a matrix $\mathbf{A} \in \mathbb{R}^{d_\ell \times d_{\ell-1}}$ and bias vector $\mathbf{b} \in \mathbb{R}^{d_\ell}$; this mapping therefore takes in an input $\mathbf{x} \in \mathbb{R}^{d_{\ell-1}}$ and produces a vector output $\mathbf{Ax} + \mathbf{b} \in \mathbb{R}^{d_\ell}$.
  - **nonlinear operations:** the previous linear operations are followed by coordinate-wise application of a non-linear function $\sigma(\cdot) : \mathbb{R} \to \mathbb{R}$.

  The resulting output of the layer being a vector with components

$$\sigma((\mathbf{Ax} + \mathbf{b})_1),\ \sigma((\mathbf{Ax} + \mathbf{b})_2),\ \ldots,\ \sigma((\mathbf{Ax} + \mathbf{b})_{d_\ell})$$

- Neural network models are then defined by stacking layers of this kind.
- Notice: $(\mathbf{Ax} + \mathbf{b})_i = \sum_{j=1}^{d_{\ell-1}} A_{i,j} x_j \ + \ b_i$
- Each $\sigma((\mathbf{Ax} + \mathbf{b})_i)$ is a **perceptron**.

(Pre-) Introduction: Fully Connected Feed Forward NN

- Suppose a neural network model is defined by stacking L layers.
- Formally, this is a composition of functions:

$$\mathbb{R}^{d_0} \to \mathbb{R}^{d_1} \to \cdots \to \mathbb{R}^{d_{\ell-1}} \to \mathbb{R}^{d_\ell} \to \cdots \to \mathbb{R}^{d_{L-1}} \to \mathbb{R}^{d_L}$$

  where
  - d_0 is the input dimension (data inputs).
  - d_ℓ is the dimension (number of neurons) for layer ℓ.
  - each layer ℓ has its own matrix $\mathbf{A}^{(\ell)}$, bias $\mathbf{b}^{(\ell)}$, and nonlinearity $\sigma^{(\ell)}$. (It is common to use the same nonlinearity $\sigma^{(\ell)} = \sigma$ for all layers.)
  - layer ℓ takes in $\mathbf{x}^{(\ell-1)} \in \mathbb{R}^{d_{\ell-1}}$ and outputs a vector $\mathbf{x}^{(\ell)} \in \mathbb{R}^{d_\ell}$

- The **trainable parameters** of this model consist of all the entries of all the matrices and bias vectors for all layers. Called **weights**.

N.B.: This model is also called a **multilayer perceptron** (MLP).

(Pre-) Introduction: Classical Neural Network Training

- Input data are feature vectors $\mathbf{x}_1, \ldots, \mathbf{x}_N$
- Supervised learning: need labels $y_1, \ldots, y_N$
- Train NN parameters via Empirical Risk Minimisation (ERM).
- This is a form of maximum likelihood estimation.
- Can use penalisation (regularisation).
- Loss is a weight-dependent and data-dependent function.
- Optimise the loss w.r.t weights using gradient descent methods.
- Stochastic Gradient Descent (SGD): Use random mini-batches.
- **Result:** A fixed setting of the trainable parameters (weights), corresponding to a single function from input features to labels.

17.1 Introduction: Bayesian Neural Networks (BNNs)

- A given neural network architecture can represent many functions, corresponding to different settings of the parameters (weights).
- Many parameter settings can fit the training data well, yet exhibit different properties outside the training data.
- Considering all of these different models together can lead to improved accuracy and uncertainty representation.

- Choices to consider:
  - Prior distribution
  - Likelihood
- Posterior distribution is defined by these choices.

17.2 Priors for BNNs

- In case of interest, a good review:

PRIORS IN BAYESIAN DEEP LEARNING: A REVIEW

Vincent Fortuin
Department of Computer Science
ETH Zürich
Zürich, Switzerland
fortuin@inf.ethz.ch

- Contains a lot more than BNNs
  (e.g. Deep Gaussian Processes, VAEs, neural processes.)
- Section 4 therein is specifically about priors in BNNs.
- And Section 5 therein is about "learning the prior"

17.2.1 Gaussian priors (for MLPs)

- One hidden layer with activation function φ and "linear" output layer:

$$f(\mathbf{x}, \boldsymbol{\theta}) = \mathbf{W}_2 \varphi(\mathbf{W}_1 \mathbf{x} + \mathbf{b}_1) + \mathbf{b}_2$$

- To specify the prior:

$$\mathbf{W}_\ell \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_{\text{weights\_}\ell}), \quad \mathbf{b}_\ell \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_{\text{bias\_}\ell}), \qquad \ell = 1, 2.$$

- Notes:
  - Covariance matrices of the corresponding dimensions.
  - $\mathbf{b}_\ell$ is random vector, so $\boldsymbol{\Sigma}_{\text{bias\_}\ell}$ is the covariance matrix for $\mathbf{b}_\ell$ as usual.
  - $\mathbf{W}_1, \mathbf{W}_2$ are matrices! So then the interpretation of $\boldsymbol{\Sigma}_{\text{weights\_1}}, \boldsymbol{\Sigma}_{\text{weights\_2}}$ is the covariance matrices corresponding to their *flattened* versions.

- Commonly used simplifications:
  - Covariance is diagonal (which corresponds to independent weights!)
  - Covariance is a multiple of the identity (same variance for all weights).

17.2.1 Gaussian priors (for MLPs), cont'd

- L hidden layers with activation function φ and "linear" output layer:

$$f(\mathbf{x}, \boldsymbol{\theta}) = \mathbf{W}_L \varphi(\mathbf{W}_{L-1} \varphi(\cdots \mathbf{W}_2 \varphi(\mathbf{W}_1 \mathbf{x} + \mathbf{b}_1) + \mathbf{b}_2) \cdots + \mathbf{b}_{L-1}) + \mathbf{b}_L$$

- To specify the prior:

$$\mathbf{W}_\ell \sim \mathcal{N}(\mathbf{0}, \alpha_\ell^2 \mathbf{I}), \quad \mathbf{b}_\ell \sim \mathcal{N}(\mathbf{0}, \beta_\ell^2 \mathbf{I}), \qquad \ell = 1, \ldots, L.$$

- Weight initialization:
  - **Xavier Glorot:** $\alpha_\ell^2 = \dfrac{2}{n_{\text{in}} + n_{\text{out}}}$
  - **LeCun:** $\alpha_\ell^2 = \dfrac{1}{n_{\text{in}}}$

  $n_{\text{in}} = d_{\ell-1}$ (fan-in)

  $n_{\text{out}} = d_\ell$ (fan-out)

- **Question:** What parameters does parameter vector $\boldsymbol{\theta}$ represent?

17.2.1 Gaussian priors (for MLPs), cont'd

- The Gaussian prior over weights (and biases), formally speaking, depends on the chosen *mean parameters* and *variance parameters*.
- It is common choice to set the mean to $\mathbf{0}$ as in the previous slide.
- The prior variance parameter for the bias parameters of a given layer is the same as for the weight parameters in the same layer.
- In this case, the Gaussian prior parameters are called hyperparameters.
- Always ask what the word 'parameters' represents in a given context!

17.2.1 Gaussian priors (for MLPs), cont'd

- One hidden layer with activation function φ and "linear" output layer:

$$f(\mathbf{x}, \boldsymbol{\theta}) = \mathbf{W}_2 \varphi(\mathbf{W}_1 \mathbf{x} + \mathbf{b}_1) + \mathbf{b}_2$$

- To specify the prior:

$$\mathbf{W}_\ell \sim \mathcal{N}(\mathbf{0}, \alpha_\ell^2 \mathbf{I}), \quad \mathbf{b}_\ell \sim \mathcal{N}(\mathbf{0}, \beta_\ell^2 \mathbf{I}), \qquad \ell = 1, 2.$$

- Equivalently:

$$\mathbf{W}_\ell \sim \alpha_\ell \mathcal{N}(\mathbf{0}, \mathbf{I}), \quad \mathbf{b}_\ell \sim \beta_\ell \mathcal{N}(\mathbf{0}, \mathbf{I}), \qquad \ell = 1, 2.$$

- Define the random $\boldsymbol{\eta}_\ell \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, and similarly $\boldsymbol{\epsilon}_\ell \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.

- Can write: $\quad f(\mathbf{x}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \alpha_2 \boldsymbol{\eta}_2 \varphi(\alpha_1 \boldsymbol{\eta}_1 \mathbf{x} + \beta_1 \boldsymbol{\epsilon}_1) + \beta_2 \boldsymbol{\epsilon}_2$

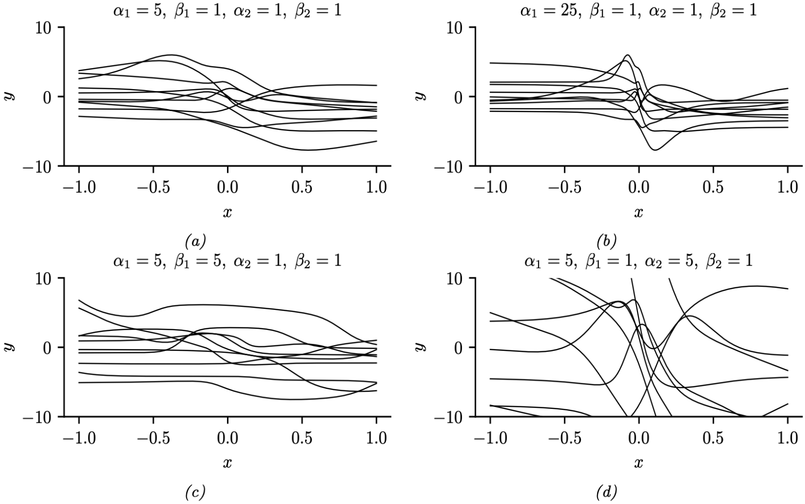


Figure 17.1 in Murphy (2023).

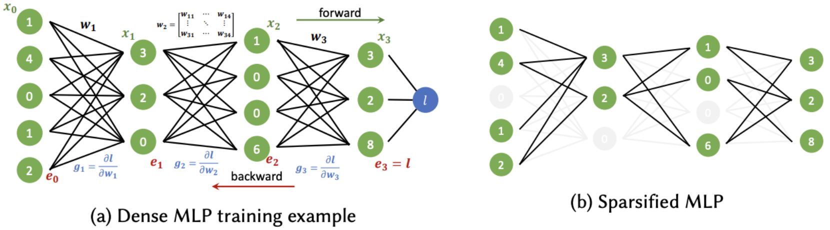


Fig. 2. Training, Inference, and Sparsification examples.

More details about this, and more, in Hoefler et al. (2021).

17.2.3 Learning the prior

- Different priors on parameters correspond to priors on functions.
- Can in principle set hyperparameters (e.g. α, β for Gaussian prior) using grid search to optimise the cross-validation loss.
- A problem: Cross-validation loss is a good estimate of the true loss for stable learning rules. The stability of NNs is an open problem.
- Another problem: Cross-validation can be computationally costly because it scales exponentially with number of hyperparameters.
- Workaround: Optimise the marginal likelihood (a.k.a. evidence)

$$p(\mathcal{D}|\alpha, \beta) = \int p(\mathcal{D}|\mathbf{w}, \alpha, \beta) p(\mathbf{w}|\alpha, \beta) d\mathbf{w}$$

- This is called **empirical Bayes** or **evidence maximisation**.
- **N.B.:** There are other approaches for learning the prior.

17.2.4 Priors in function space

- **To do:** Read about this (Murphy (2023), p. 648).

17.2.5 Architectural priors

- **To do:** Read about this (Murphy (2023), p. 649).
- In particular, read the note about **neural architecture search** being a form of structural prior learning.

17.3 Posteriors for BNNs

- Most important thing to learn:

$$p(\mathbf{w}|\mathcal{D}, \boldsymbol{\theta}) = \frac{p(\mathcal{D}|\mathbf{w}, \boldsymbol{\theta})p(\mathbf{w}|\boldsymbol{\theta})}{p(\mathcal{D}|\boldsymbol{\theta})}$$

- Let's make it easy to remember:

$$\text{posterior} = \frac{\text{likelihood} \times \text{prior}}{\text{evidence}}$$

- Where 'evidence' is the normalization:

$$p(\mathcal{D}|\boldsymbol{\theta}) = \int p(\mathcal{D}|\mathbf{w}, \boldsymbol{\theta})p(\mathbf{w}|\boldsymbol{\theta})d\mathbf{w}$$

17.3 Posteriors for BNNs - case of supervised learning

- Dataset $\mathcal{D}$ is a list of pairs $(\mathbf{x}, y)$, with input features $\mathbf{x}$ and labels y.
- Have discussed the posterior distribution over weights $p(\mathbf{w}|\mathcal{D}, \boldsymbol{\theta})$.
- Interest is on a posterior predictive distribution $p(y|\mathbf{x}, \mathcal{D}, \boldsymbol{\theta})$.

More on Posteriors for BNNs

- 17.3.1 Monte Carlo dropout
  - **To do:** Read about this (Murphy (2023), pp. 649 - 650).
- 17.3.2 Laplace approximation
  - **To do:** Read about this (Murphy (2023), pp. 650 - 651).
- 17.3.3 Variational inference
  - **To do:** Read about this (Murphy (2023), pp. 651 - 652).
- 17.3.4 Expectation propagation
  - **To do:** Read about this (Murphy (2023), p. 652).
- 17.3.5 Last layer methods
  - **To do:** Read about this (Murphy (2023), pp. 652 - 653).

Even more on Posteriors for BNNs

- 17.3.6 SNGP
  - **To do:** Read about this (Murphy (2023), p. 653).
- 17.3.7 MCMC methods
  - **To do:** Read about this (Murphy (2023), pp. 653 - 654).
- 17.3.8 Methods based on the SGD trajectory
  - **To do:** Read about this (Murphy (2023), pp. 654 - 655).
- 17.3.9 Deep ensembles
  - **To do:** Read about this (Murphy (2023), pp. 655 - 659).
- 17.3.10 Approximating the posterior predictive distribution
  - **To do:** Read about this (Murphy (2023), pp. 659 - 662).
- 17.3.11 Tempered and cold posteriors
  - **To do:** Read about this (Murphy (2023), pp. 662 - 663).

Next: Generalisation in deep learning

- Generalisation is about out-of-sample performance.
- Here "sample" can be a training data or testing data.
- I know how my NN did on training data and on testing data.
  Question: How will it do on yet unseen data?
- Need guarantees of predictive performance for unseen data.
- Ideally: guarantees that hold at distribution level.

Different settings require different approaches

- In-distribution generalisation:
  - Training data from distribution P.
  - Testing data from distribution P.
  - All future data to come from distribution P.
  - Did well on training and testing data.
  - Want to ensure doing well on future data.

- Out-of-distribution generalisation
  - Training data from distribution P_0.
  - Testing data from distribution P_0.
  - Future data may come from distribution P_1.
  - Did well on training and testing data.
  - Want to ensure doing well on future data.

17.4 Generalisation in Bayesian deep learning

- Bayesian methods are principled and generally interpretable.
- Arguments that Bayesian methods improve predictive accuracy and generalisation performance.
- However: Bayesian methods can be computationally expensive!

17.4.1 [Flat vs sharp] minima

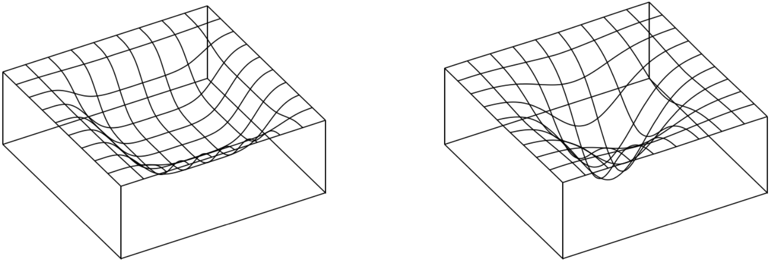


Figure 17.10 in Murphy (2023).

- NN loss surfaces can have many near-zero loss solutions.
- **Mode connectivity:** The observation that any two independently trained SGD solutions connected by a curve (in some subspace of weight space) along which the training loss remains near-zero!

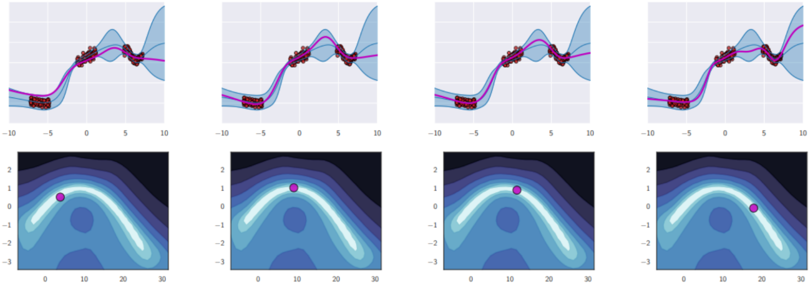


Figure 17.11 in Murphy (2023).

- $N_{\mathrm{eff}}(\mathbf{H}, c) = \displaystyle\sum_{i=1}^{k} \frac{\lambda_i}{\lambda_i + c}$
- Where λ_i are the eigenvalues of the Hessian matrix $\mathbf{H}$ at a local mode.
- $c > 0$ is a regularisation parameter.

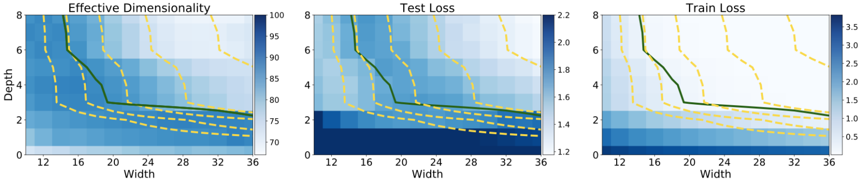


Figure 17.12 in Murphy (2023).

17.4.4 The hypothesis space of DNNs

- Observations of Zhang et al. (2017) showed that CNNs can fit CIFAR-10 images with random labels with zero training error, but can still generalize well on the noise-free test set.
- It was claimed that this observation apparently contradicts the classical understanding of generalization.

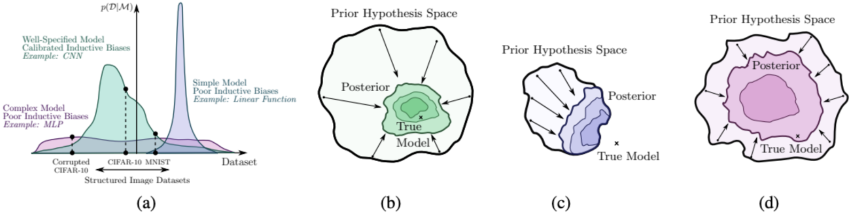


Figure 17.13 in Murphy (2023).

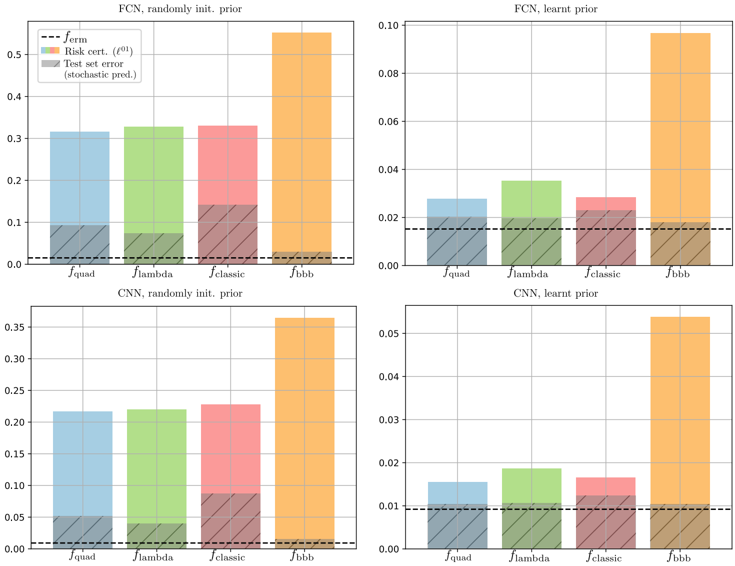
17.4.5 PAC-Bayes

- PAC-Bayes bounds
- A kind of statistical learning bounds

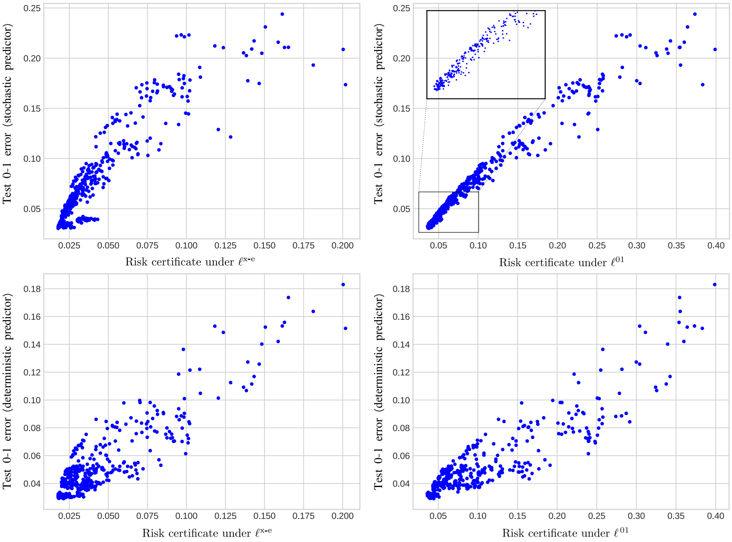
  (a.k.a. generalization bounds, risk bounds)
- Can be used for
  - **Certification:** Evaluating a bound value for a randomised classifier defined by some probability distribution over classifiers.
  - **Optimisation:** Using the bound as learning objective to get a probability distribution over classifiers (e.g. over NN weights).
- See Pérez-Ortiz et al. (2021) for more details on this.
- Ask me about this if you're interested.

  (i.e. don't trust Murphy (2023) on this!)

More on PAC-Bayes: Figure 3 in Pérez-Ortiz et al. (2021)



More on PAC-Bayes: Figure 1 in Pérez-Ortiz et al. (2021)



- Various (B)NNs when presented with the training data ‘blue vs red.’ The green blob is an example of some OOD inputs

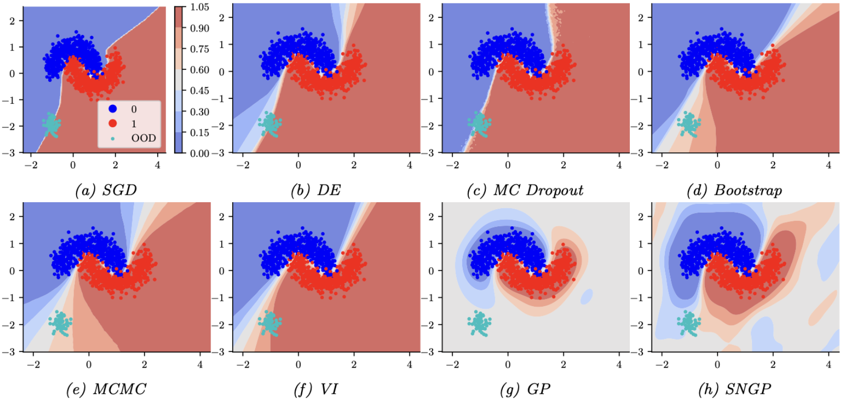


Figure 17.15 in Murphy (2023).

More topics in Chapter 17

- 17.4.7 Model selection for BNNs
  - **To do:** Read about this (Murphy (2023), p. 669).
- 17.5 Online inference
  - **To do:** Read about this (Murphy (2023), pp. 670 - 675).
  - 17.5.1 Sequential Laplace for DNNs
  - 17.5.2 Extended Kalman filtering for DNNs
  - 17.5.3 Assumed density filtering for DNNs
  - 17.5.4 Online variational inference for DNNs
- 17.6 Hierarchical Bayesian neural networks
  - **To do:** Read about this (Murphy (2023), pp. 675 - 678).

17.6.1 Example: multimoons classification

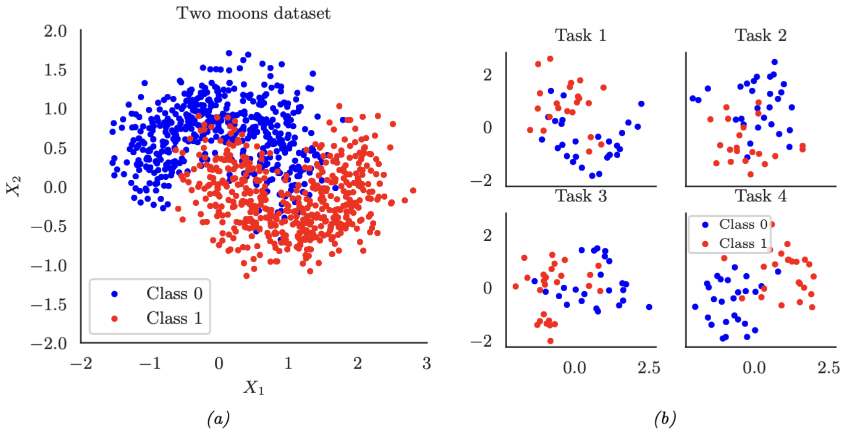


Figure 17.17 in Murphy (2023).

References

Torsten Hoefler, Dan Alistarh, Tal Ben-Nun, Nikoli Dryden, and Alexandra Peste. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *Journal of Machine Learning Research*, 22(241):1–124, 2021.

Kevin P. Murphy. *Probabilistic Machine Learning: Advanced Topics*. MIT Press, 2023. URL `http://probml.github.io/book2`.

María Pérez-Ortiz, Omar Rivasplata, John Shawe-Taylor, and Csaba Szepesvári. Tighter risk certificates for neural networks. *Journal of Machine Learning Research*, 22(227):1–40, 2021.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *ICLR*, 2017. Preprint version arXiv:1611.03530.