Bayesian machine learning

Bayesian deep learning

Julyan Arbel

Statify team, Inria Grenoble Rhône-Alpes & Univ. Grenoble-Alpes, France
✉ julyan.arbel@inria.fr 🖱 www.julyanarbel.com

http://github.com/rbardenet/bml-course

Outline

Outline

Outline

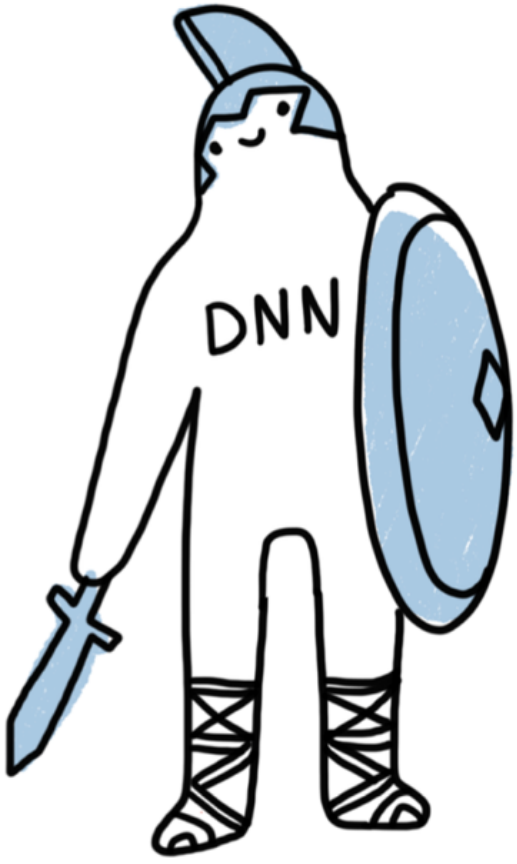
Outline

Outline

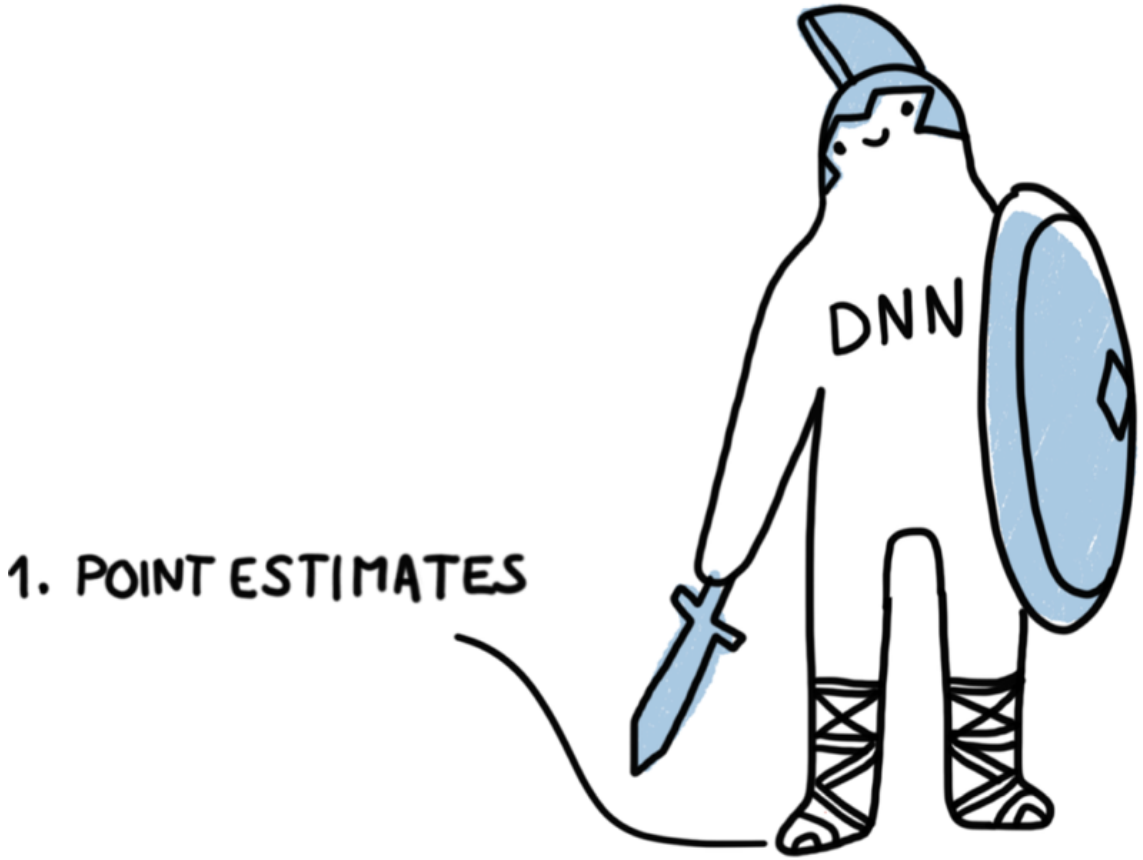
Outline

What comes to your mind when you hear “Bayesian deep learning”?

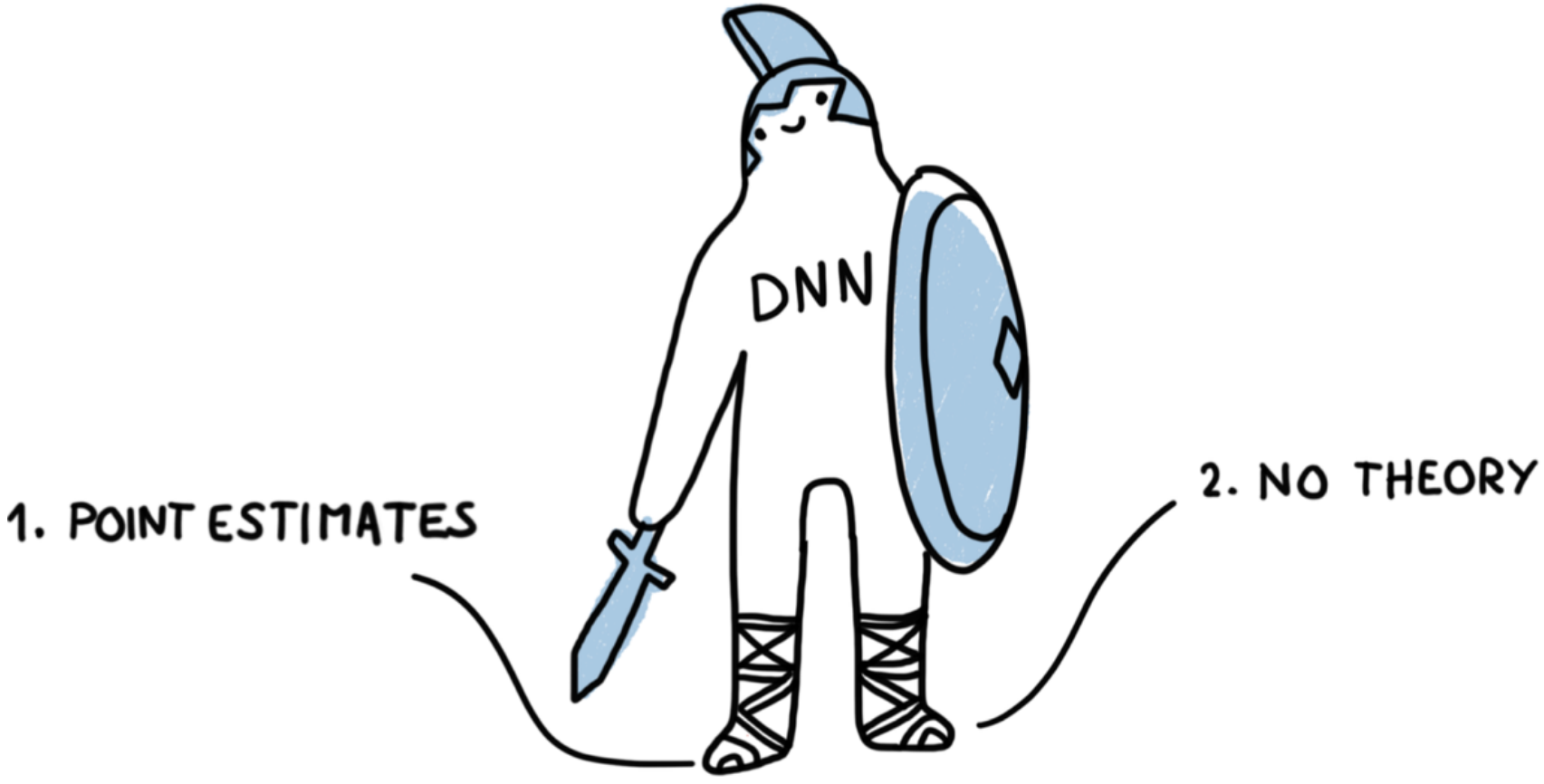
Deep neural networks Achilles heels



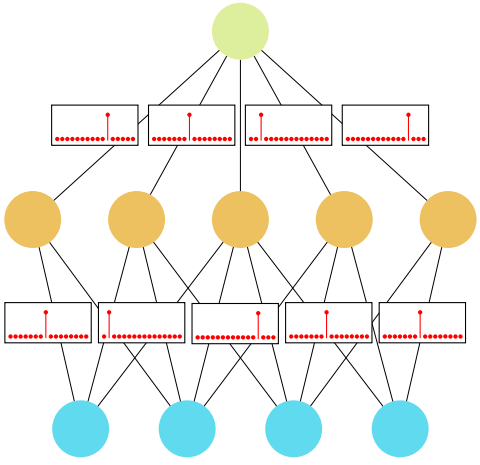
Deep neural networks Achilles heels



Deep neural networks Achilles heels

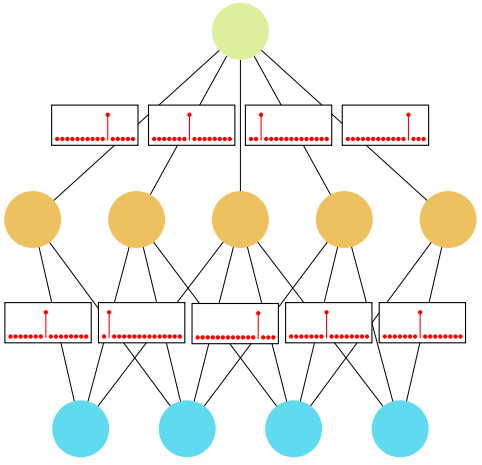


Different flavours of neural networks (Jospin et al., 2020a)

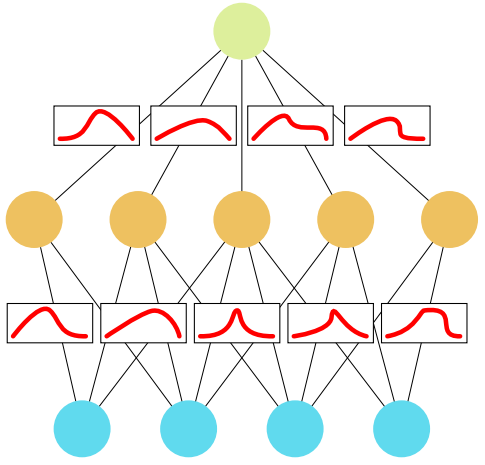


Point estimate NN

Different flavours of neural networks (Jospin et al., 2020a)

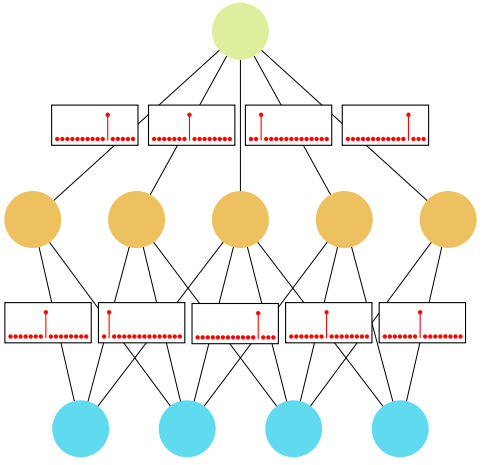


Point estimate NN

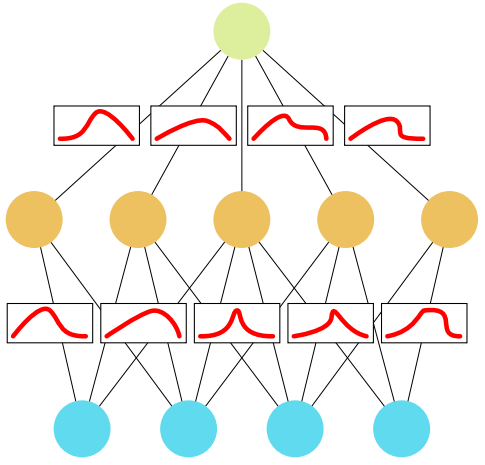


BNN w/ random weights

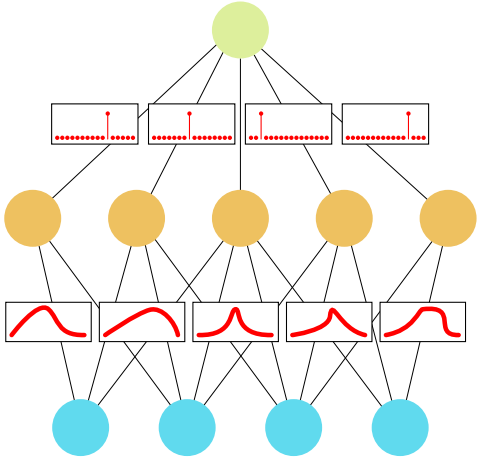
Different flavours of neural networks (Jospin et al., 2020a)



Point estimate NN

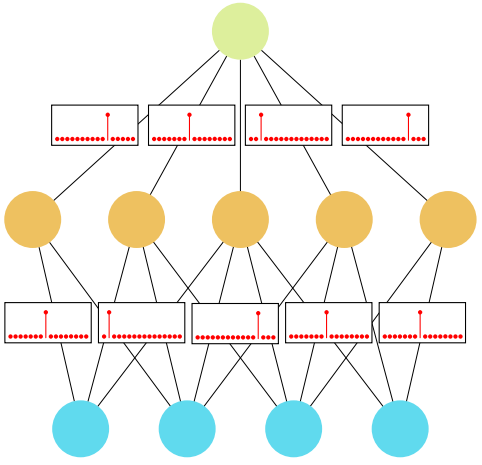


BNN w/ random weights

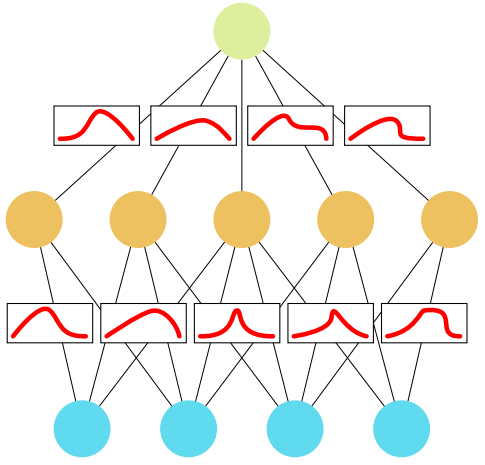


BNN w/ last-layer rd weights

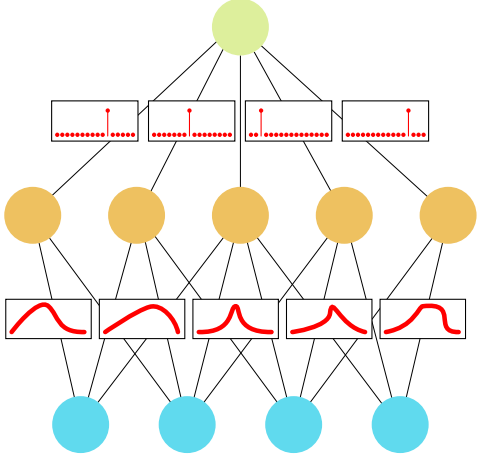
Different flavours of neural networks (Jospin et al., 2020a)



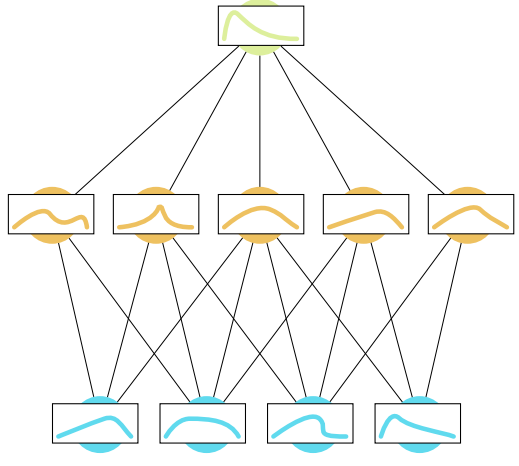
Point estimate NN



BNN w/ random weights



BNN w/ last-layer rd weights



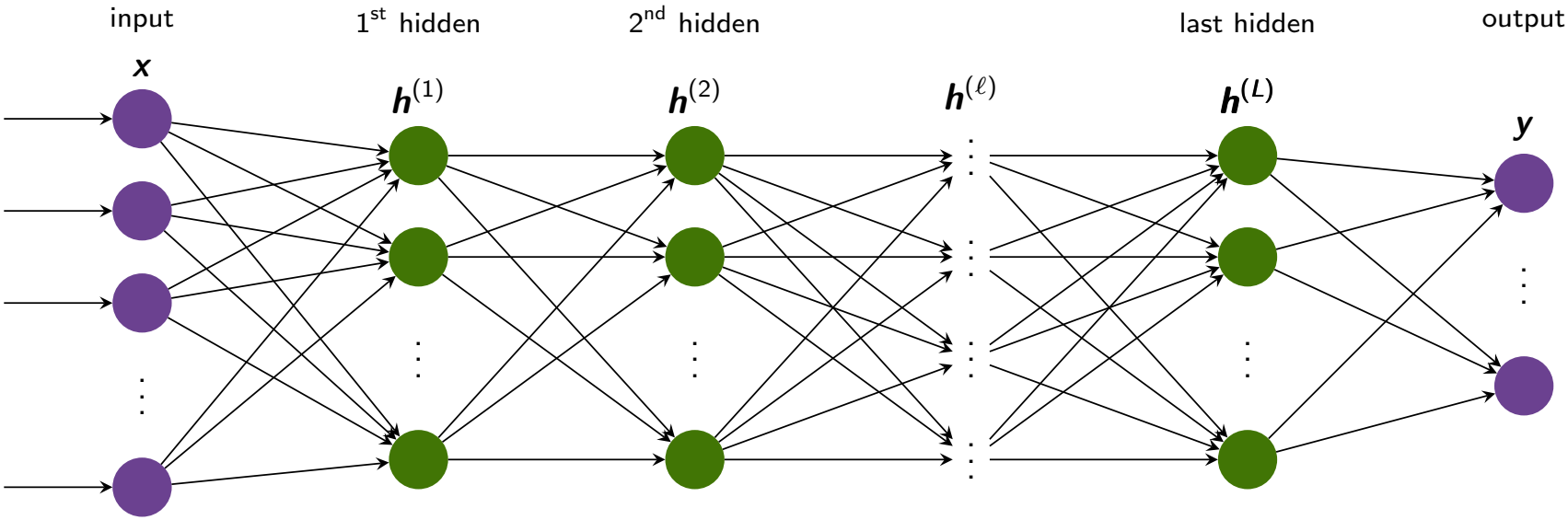
BNN w/ random activations

Useful references

- First substantial reference: Chapter 17 on BNNs by Kevin P. Murphy. *Probabilistic Machine Learning: Advanced Topics*. MIT Press, 2023. URL: http://probml.github.io/book2
- Review papers: Jospin et al. (2020b), Abdar et al. (2021), Goan and Fookes (2020), Fortuin (2022), Ashukha et al. (2020), Band et al. (2021), and Nado et al. (2021)
- Our review paper: Julyan Arbel, Konstantinos Pitas, and Mariia Vladimirova. "A primer on Bayesian neural networks: review and debates". In: *Preprint* (2022)

Outline

Neural networks notations



- pre-nonlinearity $\boldsymbol{g}^{(\ell)} = \boldsymbol{g}^{(\ell)}(\boldsymbol{x})$, post-nonlinearity $\boldsymbol{h}^{(\ell)} = \boldsymbol{h}^{(\ell)}(\boldsymbol{x})$

$$\boldsymbol{g}^{(\ell)}(\boldsymbol{x}) = \boldsymbol{W}^{(\ell)}\boldsymbol{h}^{(\ell-1)}(\boldsymbol{x}), \quad \boldsymbol{h}^{(\ell)}(\boldsymbol{x}) = \phi(\boldsymbol{g}^{(\ell)}(\boldsymbol{x}))$$

- nonlinearity or activation function $\phi : \mathbb{R} \to \mathbb{R}$.
- weight matrix $\boldsymbol{W}^{(\ell)}$ of dimension $H_\ell \times H_{\ell-1}$ including a bias vector

Training

Optimization problem: minimize the loss function

$$\hat{\boldsymbol{w}} = \arg\min_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w}).$$

With gradient-based optimization:

$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta \, \partial_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w}).$$

$\eta > 0$ is a step size, or learning rate. Gradients are computed as products of gradients between each layer from right to left, a procedure called backpropagation (Rumelhart, Hinton, and Williams, 1986).

Gradients are approximated on randomly chosen subsets called batches: stochastic gradient descent, SGD (Robbins and Monro, 1951). See survey of optimization methods bySun et al. (2019).

Architecture choice

- Convolutional neural networks (CNN) are widely used in computer vision.
- Recurrent neural networks (RNN) are advantageous for sequential data, designed to save the output of a layer by adding it back to the input (Hochreiter and Schmidhuber, 1997).
- Residual neural networks (ResNet) have residual blocks which add the output from the previous layer to the layer ahead, so-called skip-connections (He et al., 2016). Allows very deep training.

Expressiveness

Expressiveness describes neural networks' ability to approximate functions (Cybenko, 1989; Funahashi, 1989; Hornik, Stinchcombe, and White, 1989; Barron, 1994).

Universal approximation theorem

Neural networks of one hidden layer and suitable activation function can approximate any continuous function on a compact domain, say $f : [0, 1]^N \to \mathbb{R}$, to any desired accuracy.

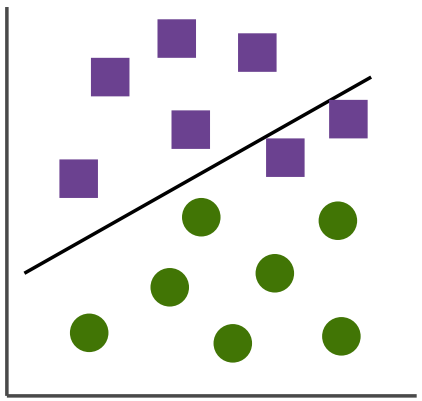
But the size of such networks may be exponential in the input dimension N, which makes them highly prone to overfitting as well as impractical.

Width-depth trade-offs studied by Chatziafratis, Nagarajan, Panageas, and Wang (2020) and Chatziafratis, Nagarajan, and Panageas (2020).

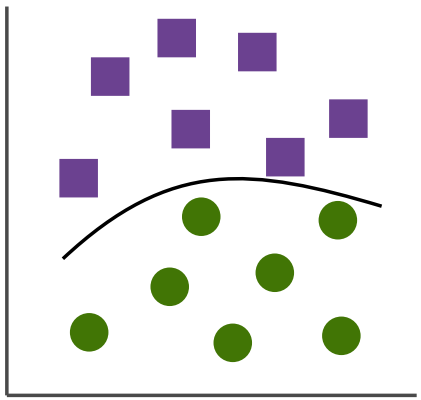
Generalization and overfitting I

Classical regime

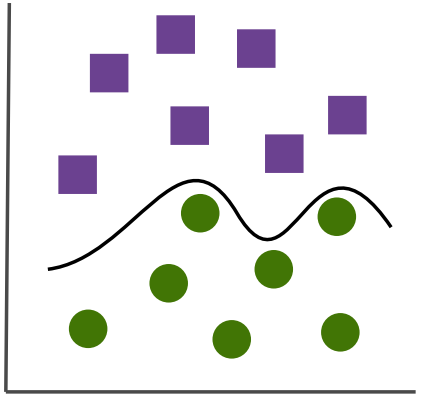
underfitting



optimum



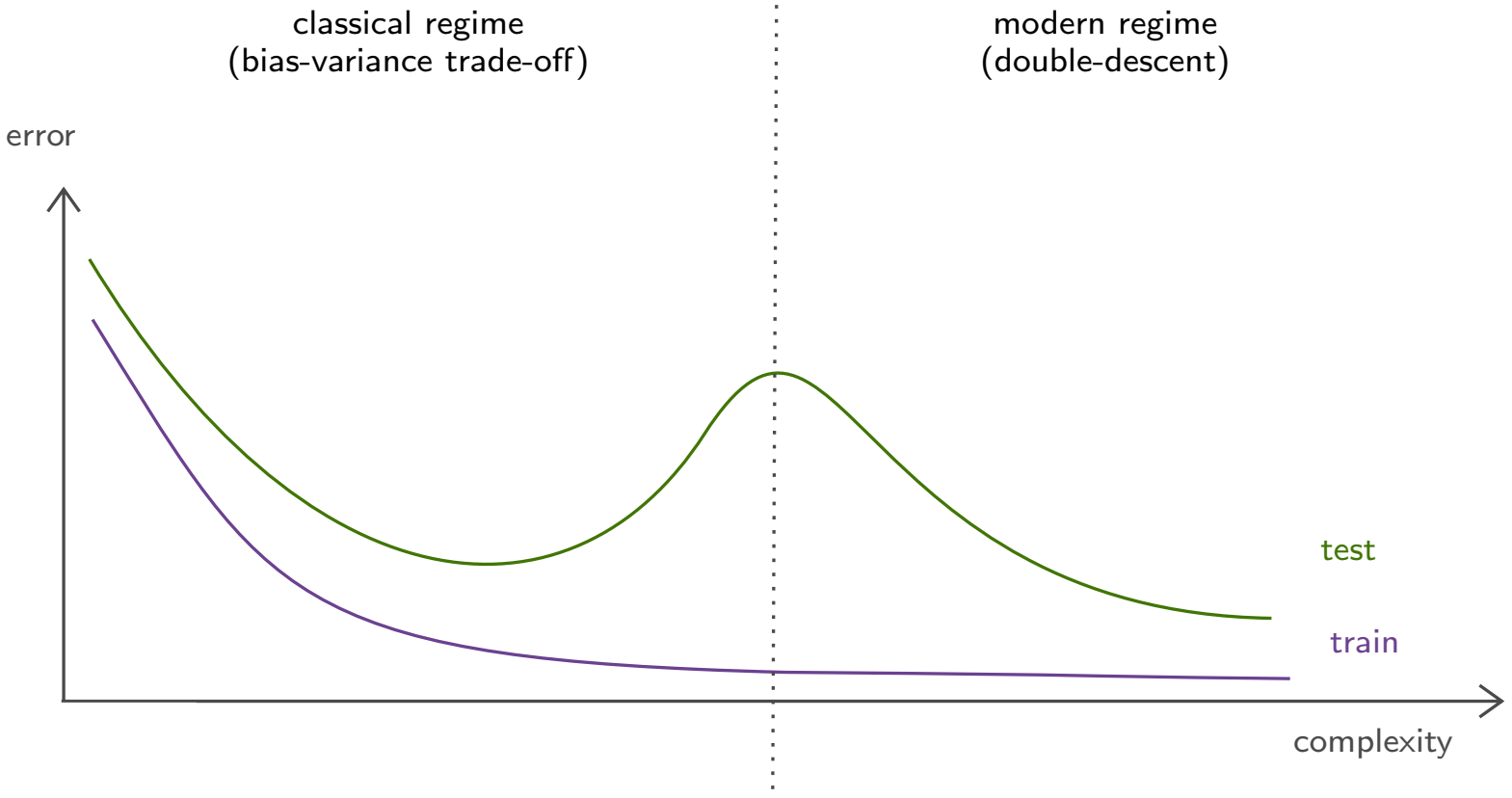
overfitting



Generalization and overfitting II

Modern regime

It was shown recently that when increasing the model size beyond the number of training examples, the model's test error can start decreasing again after reaching the interpolation peak: double-descent (Belkin et al., 2019).



Limitations with point-estimate neural networks

- Inability to distinguish between in-domain and out-of-domain samples (Lee et al., 2018; Mitros and Mac Namee, 2019; Hein, Andriushchenko, and Bitterwolf, 2019; Ashukha et al., 2020), and the sensitivity to domain shifts (Ovadia et al., 2019), which are explained in details later on;
- Inability to provide reliable uncertainty estimates for a deep neural network's decision and frequently occurring overconfident predictions (Minderer et al., 2021);
- Lack of transparency and interpretability of a deep neural network's inference model, which makes it difficult to trust their outcomes;
- Sensitivity to adversarial attacks that make deep neural networks vulnerable for sabotage (Wilson et al., 2016).

Great expectations of Bayesian neural networks

- Uncertainty quantification through the posterior distribution: BNN are shown to be better calibrated than NN
- Distinguishing between the epistemic uncertainty $p(\theta|D)$ and the aleatoric uncertainty $p(y|x,\theta)$: desirable in small dataset settings, providing high epistemic uncertainty for prediction, avoiding overfitting
- Integrating prior knowledge: most regularization methods for NN can be understood as setting a prior
- Interpreting known ML algorithms as approximate Bayesian methods: including regularization, ensembling, constant (learning rate) SGD, etc.

Outline

Outline

Connection prior/initialization

At initialization:

- random weights and biases, e.g., $W_{ij}^{(l)} \sim \mathcal{N}(0, \sigma_w^2)$, $B_i^{(l)} \sim \mathcal{N}(0, \sigma_b^2)$;
- inputs $X^{(1)}$ fixed.

Goal:

- find a criterion that the pre-activations $Z^{(l)}$ should match;
  example: $\mathrm{Var}(Z^{(l)}) = 1$;
- deduce a constraint over the distributions of $W^{(l)}$ and $B^{(l)}$;
  example: provided that $W_{ij}^{(l)} \sim \mathcal{N}(0, \sigma_w^2)$, $B_i^{(l)} \sim \mathcal{N}(0, \sigma_b^2)$, tune σ_w^2 and σ_b^2 accordingly.

Find a good initialization procedure

Naive heuristic. (*Understanding the difficulty of training deep feedforward neural networks*, Glorot and Bengio 2010):

- idea: preserve the variance of the pre-activations $Z^{(l)}$;
- tune σ_w^2 and σ_b^2 s.t.: $\mathrm{Var}(Z^{(l+1)}) = \mathrm{Var}(Z^{(l)})$.

Constraint: the NN is linear ($\phi = \mathrm{Id}$).
Result: $\sigma_b^2 = 0$, $\sigma_w^2 = 1$, $\mathrm{Var}(\frac{1}{\sqrt{n_l}} W_{ij}^l) = 1$.

Remark: this heuristic can be extended to $\phi =$ReLU.
(*Delving deep into rectifiers: Surpassing human-level performance on imagenet classification*, He et al., 2015)
Result: $\sigma_w^2 = 2$.

Outline

Neural-network Gaussian process (NN-GP)

- An MLP with one hidden layer, whose width goes to infinity, and which has a Gaussian prior on all the parameters, converges to a Gaussian process with a well-defined kernel (Neal, 1996).

Proof: H the number of hidden units, ϕ some nonlinear activation function, b biais, weights v and $\boldsymbol{u}$; then unit k can be written

$$f_k(\boldsymbol{x}) = b_k + \sum_{j=1}^{H} v_{jk} h_j(\boldsymbol{x}), \quad h_j(\boldsymbol{x}) = \phi(U_{0j} + \boldsymbol{x}^t \boldsymbol{u}_j)$$

Then

- $\mathbb{E}[f_k(\boldsymbol{x})] =$
- $\mathbb{E}[f_k(\boldsymbol{x}) f_k(\boldsymbol{x}')] = \ldots := \mathcal{K}(\boldsymbol{x}, \boldsymbol{x}')$
- The joint distribution over $\{f_k(\boldsymbol{x_n}), n = 1 : N\}$ converges to a multivariate Gaussian.
- So the MLP converges to a GP with mean 0 and kernel $\mathcal{K}$, called the neural network kernel. It is a non-stationary kernel.

Outline

Neural tangent kernel (NTK)

- The NNGP is obtained under the assumption that weights are random and width goes to infinity.
- Natural question: can we derive a kernel from a DNN while it is being trained?
- The answer is yes (Jacot, Gabriel, and Hongler, 2018). The associated kernel $\mathcal{T}(\boldsymbol{x}, \boldsymbol{x}')$ is called the Neural tangent kernel (NTK)

$$\mathcal{T}(\boldsymbol{x}, \boldsymbol{x}') := \nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}; \boldsymbol{\theta}_\infty) \cdot \nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}'; \boldsymbol{\theta}_\infty)$$

  and is obtained with
  - continuous time gradient descent
  - letting the learning rate η become infinitesimally small
  - letting the widths go to infinity.

Outline

Edge of Chaos framework

See: Poole et al. (2016) and Schoenholz et al. (2017)

Main idea. Let x_a and x_b be two (fixed) inputs:

- variance: $v_a^{(l)} = \mathbb{E}[(Z_{j;a}^{(l)})^2]$;
- correlation: $c_{ab}^{(l)} = \frac{1}{\sqrt{v_a^{(l)} v_b^{(l)}}} \mathbb{E}[Z_{j;a}^{(l)} Z_{j;b}^{(l)}]$;
- goal: preserve the correlations $c_{ab}^{(l)}$ during propagation;
- solution: find a recurrence equation $c_{ab}^{(l+1)} = f(c_{ab}^{(l)})$;
- to do so, we must make an assumption on the distribution of $Z_{j;a}^{(l)}$;
  $\Rightarrow$ Gaussian hypothesis: $Z_{j;a}^{(l)} \sim \mathcal{N}(0, v_a^{(l)})$;
  Central Limit Theorem ($n_l \to \infty$): $Z_{j;a}^{(l+1)} = \frac{1}{\sqrt{n_l}} W_j^{(l)} X_a^{(l)} + B_j^{(l)}$;
- resulting simplified dynamics:

$$v_a^{(l+1)} = \mathcal{V}(v_a^{(l)} | \sigma_w, \sigma_b), \qquad c_{ab}^{(l+1)} = \mathcal{C}(c_{ab}^{(l)}, v_a^{(l)}, v_b^{(l)} | \sigma_w, \sigma_b).$$

Edge of Chaos results (Poole et al., 2016; Schoenholz et al., 2016)

Additional assumptions.

- the sequence $(v_a^{(l)})_l$ tends to a non-zero limit v^*, independent from the starting point $v_a^{(0)}$;
- $(v_a^{(l)})_l$ is assumed to have already converged;
- so: $c_{ab}^{(l+1)} = \mathcal{C}(c_{ab}^{(l)}, v^*, v^* | \sigma_w, \sigma_b) = \mathcal{C}_*(c_{ab}^{(l)} | \sigma_w, \sigma_b)$.

Phases of information propagation:

- *chaotic phase*: $\lim_{l\to\infty} c_{ab}^l = c^* < 1$
  $\Rightarrow$ decorrelate (partially or fully);
- *ordered phase*: $\lim_{l\to\infty} c_{ab}^l = c^* = 1$ with $\mathcal{C}'_*(1) < 1$
  $\Rightarrow$ correlate fully at an exponential rate;
- *edge of chaos*: $\lim_{l\to\infty} c_{ab}^l = c^* = 1$ with $\mathcal{C}'_*(1) = 1$
  $\Rightarrow$ correlate fully at sub-exponential rate.

$\Rightarrow$ tune σ_w and σ_b such that the NN lies in the "edge of chaos" phase.

Edge of Chaos I

Poole et al. (2016) and Schoenholz et al. (2017)

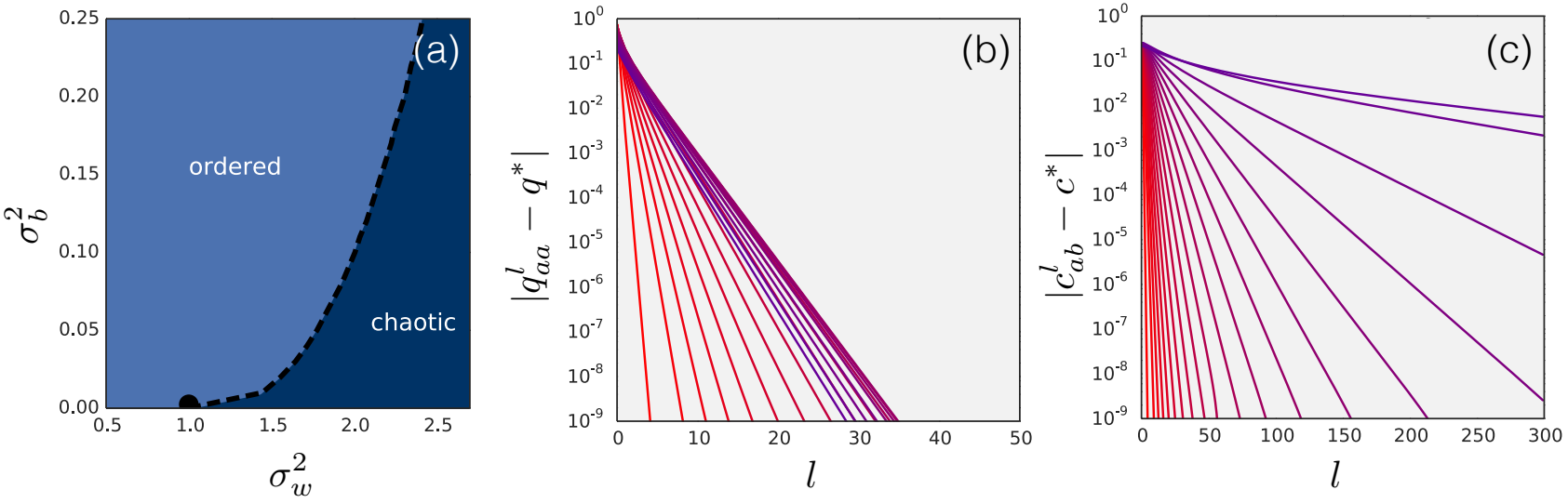


Figure: (a) Edge of chaos diagram showing the boundary between ordered and chaotic phases as a function of σ_w^2 and σ_b^2. (b) The residual $|q^* - q_{aa}^l|$ as a function of depth on a log-scale with $\sigma_b^2 = 0.05$ and σ_w^2 from 0.01 (red) to 1.7 (purple). Clear exponential behavior is observed. (c) The residual $|c^* - c_{ab}^l|$ as a function of depth on a log-scale. Again, the exponential behavior is clear. The same color scheme is used here as in (b).

From Schoenholz et al. (2017)

Outline

Definition (Generalized Weibull-tail on $\mathbb{R}$)

A random variable X is generalized Weibull-tail on $\mathbb{R}$ with tail parameter $\beta > 0$ if both its right and left tails are upper and lower bounded by some Weibull-tail functions with tail parameter β:

$$\begin{aligned} \mathrm{e}^{-x^\beta l_1^r(x)} \leq & \overline{F}_X(x) \leq \mathrm{e}^{-x^\beta l_2^r(x)}, && \text{for } x > 0 \text{ and } x \text{ large enough,} \\ \mathrm{e}^{-|x|^\beta l_1^l(|x|)} \leq & F_X(x) \leq \mathrm{e}^{-|x|^\beta l_2^l(|x|)}, && \text{for } x < 0 \text{ and } -x \text{ large enough,} \end{aligned}$$

where l_1^r, l_2^r, l_1^l and l_2^l are slowly-varying functions. We note $X \sim GWT(\beta)$.

This tail description reveals the difference between hidden units' distributional properties in finite- and infinite-width Bayesian neural networks, since hidden units are generalized Weibull-tail with a tail parameter depending on those of the weights:

Theorem (Vladimirova, Arbel, and Girard, 2021)

Consider a Bayesian neural network with ReLU activation function. Let ℓ-th layer weights be independent symmetric generalized Weibull-tail on $\mathbb{R}$ with tail parameter $\beta_w^{(\ell)}$. Then conditional on the input $\mathbf{x}$, the marginal prior distribution induced by forward propagation on any pre-activation is generalized Weibull-tail on $\mathbb{R}$: for any $1 \leq \ell \leq L$, and for any $1 \leq m \leq H_\ell$,

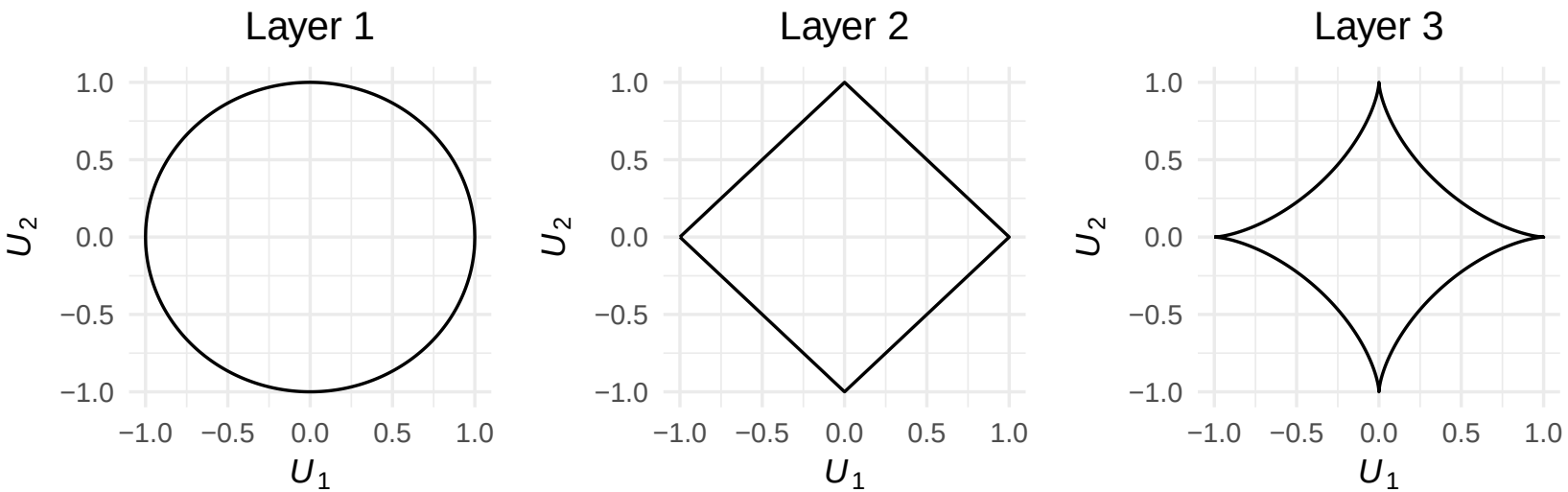
$$g_m^{(\ell)} \sim GWT(\beta^{(\ell)}),$$

with tail parameter $\beta^{(\ell)}$ such that $\frac{1}{\beta^{(\ell)}} = \frac{1}{\beta_w^{(1)}} + \cdots + \frac{1}{\beta_w^{(\ell)}}$, where a GWT distribution is defined in Definition 1.

Note that the most popular case of weight prior, iid Gaussian (Neal, 1996), corresponds to $\text{GWT}_{\mathbb{R}}(2)$ weights. This leads to units of layer ℓ which are $\text{GWT}_{\mathbb{R}}(\frac{2}{\ell})$.

Understanding priors at the unit level

Layer	Penalty on $\boldsymbol{W}$	Approximate penalty on $\boldsymbol{U}$	
1	$\|\boldsymbol{W}^{(1)}\|_2^2$, $\mathcal{L}^2$	$\|\boldsymbol{U}^{(1)}\|_2^2$	$\mathcal{L}^2$ (weight decay)
2	$\|\boldsymbol{W}^{(2)}\|_2^2$, $\mathcal{L}^2$	$\|\boldsymbol{U}^{(2)}\|$	$\mathcal{L}^1$ (Lasso)
ℓ	$\|\boldsymbol{W}^{(\ell)}\|_2^2$, $\mathcal{L}^2$	$\|\boldsymbol{U}^{(\ell)}\|_{2/\ell}^{2/\ell}$	$\mathcal{L}^{2/\ell}$



Outline

Other priors

- Although Gaussian priors are simple and widely used, they are not the only option.
- For some applications, it is useful to use shrinkage or sparsity promoting priors which encourage most of the weights to be small/close to zero. Examples of such priors
  - Laplace prior

$$p_\lambda(x) = \frac{\lambda}{2} e^{-\lambda|x|}.$$

  - spike-and-slab

Outline

Bayesian inference: basic sampling algorithm

Denote data by $D = \{D_x, D_y\}$ and parameters (weights) by $\boldsymbol{\theta}$.

Algorithm 1 Inference procedure for a BNN.

Define $p(\boldsymbol{\theta}|D) = \dfrac{p(D_{\boldsymbol{y}}|D_{\boldsymbol{x}}, \boldsymbol{\theta})p(\boldsymbol{\theta})}{\int_{\boldsymbol{\theta}} p(D_{\boldsymbol{y}}|D_{\boldsymbol{x}}, \boldsymbol{\theta'})p(\boldsymbol{\theta'})d\boldsymbol{\theta'}}$;

for $i = 0$ **to** N **do**

  Draw $\boldsymbol{\theta}_i \sim p(\boldsymbol{\theta}|D)$;

  $\boldsymbol{y}_i = \Phi_{\boldsymbol{\theta}_i}(\boldsymbol{x})$;

end for

return $Y = \{\boldsymbol{y}_i | i \in [0, N)\}$, $\Theta = \{\boldsymbol{\theta}_i | i \in [0, N)\}$;

Laplace approximation

Computes a Gaussian approximation to the posterior centered at the MAP estimate

- It is simple
- But... computing the Hessian is expensive, and may result in a non-positive definite matrix since the log likelihood of deep neural networks is non-convex.
- Gauss–Newton approximation to the Hessian

Generalized Gauss–Newton approximation

$$\boldsymbol{H}^{\mathrm{GGN}} := \sum_{i=1}^{n} \mathcal{J}_{\boldsymbol{w}}(\boldsymbol{x}_i)^{\top} \boldsymbol{\Lambda}(\boldsymbol{y}_i; f_i) \mathcal{J}_{\boldsymbol{w}}(\boldsymbol{x}_i),$$

where $\mathcal{J}_{\boldsymbol{w}}(\boldsymbol{x})$ is the network per-sample Jacobian $[\mathcal{J}_{\boldsymbol{w}}(\boldsymbol{x})]_c = \nabla_{\boldsymbol{w}} f_c(\boldsymbol{x}; \boldsymbol{w}_{\hat{\rho}})$, and $\boldsymbol{\Lambda}(\boldsymbol{y}; f) = -\nabla^2_{ff} \log p(\boldsymbol{y}; f)$ is the per-input noise matrix.

Bayesian neural networks: early algorithms

- Markov chain Monte Carlo (MCMC), Hamiltonian Monte Carlo (HMC). No-U-Turn sampler (NUTS) it most often used in probabilistic programming languages (Stan, PyMC3, Pyro, etc): is improves over classic HMC by allowing hyperparameters to be set automatically instead of manually
- Variational inference (VI): scales better than MCMC algorithms. Idea: find an approximate variational distribution in a variational family that is as close as possible to the exact posterior by minimizing the Kullback–Leibler divergence. Turns sampling into optimization.
- Stochastic variational inference (SVI): scales better than VI, stochastic gradient descent method applied to VI. Gradient of objective is computed only on mini-batches.

Bayesian neural networks: early algorithms

- Markov chain Monte Carlo (MCMC), Hamiltonian Monte Carlo (HMC). No-U-Turn sampler (NUTS) it most often used in probabilistic programming languages (Stan, PyMC3, Pyro, etc): is improves over classic HMC by allowing hyperparameters to be set automatically instead of manually
- Variational inference (VI): scales better than MCMC algorithms. Idea: find an approximate variational distribution in a variational family that is as close as possible to the exact posterior by minimizing the Kullback–Leibler divergence. Turns sampling into optimization.
- Stochastic variational inference (SVI): scales better than VI, stochastic gradient descent method applied to VI. Gradient of objective is computed only on mini-batches.

Bayesian neural networks: early algorithms

- Markov chain Monte Carlo (MCMC), Hamiltonian Monte Carlo (HMC). No-U-Turn sampler (NUTS) it most often used in probabilistic programming languages (Stan, PyMC3, Pyro, etc): is improves over classic HMC by allowing hyperparameters to be set automatically instead of manually
- Variational inference (VI): scales better than MCMC algorithms. Idea: find an approximate variational distribution in a variational family that is as close as possible to the exact posterior by minimizing the Kullback–Leibler divergence. Turns sampling into optimization.
- Stochastic variational inference (SVI): scales better than VI, stochastic gradient descent method applied to VI. Gradient of objective is computed only on mini-batches.

Bayesian neural networks: early algorithms

BUT

Stochasticity in gradient estimation stops backpropagation from functioning

Tricks for Monte Carlo gradient estimation

A number of tricks (see Mohamed et al., 2020)

- Log-derivative trick: score function estimators
- Reparameterisation trick: pathwise derivative estimator
- Measure-valued gradient estimators

BUT

Stochasticity in gradient estimation stops backpropagation from functioning

Tricks for Monte Carlo gradient estimation

A number of tricks (see Mohamed et al., 2020)

- Log-derivative trick: score function estimators
- Reparameterisation trick: pathwise derivative estimator
- Measure-valued gradient estimators

Bayesian neural networks: adapted algorithms

- Bayes-by-backprop (BBB) and probabilistic backpropagation (PBP): implement the reparameterisation trick
- Monte Carlo dropout: turning dropout into an approximate Bayesian algorithm (variational inference)
- Bayes via stochastic gradient descent: includes MCMC algorithms based on the SGD dynamic such as stochastic gradient Langevin dynamic (SGLD) and Variational Inference based on SGD dynamic such as ensembling

Monte Carlo dropout I

Dropout technique reinterpreted as a form of approximate Bayesian variational inference (Kingma, Salimans, and Welling, 2015; Gal and Ghahramani, 2016).

Idea: performing random sampling at test time. Instead of turning off the dropout layers at test time (as is usually done), hidden units are randomly dropped out according to a Bernoulli(p) distribution. Repeating this operation M times provides M versions of the MAP estimate of the network parameters $\boldsymbol{w}^m$, $m = 1, \ldots, M$ (where some units of the MAP are dropped), yielding an approximate posterior predictive in the form of the equal-weight average:

$$p(y|x, \mathcal{D}^n) \approx \frac{1}{M} \sum_{m=1}^{M} p(y|x, \boldsymbol{w}^m).$$

- Monte Carlo dropout captures some uncertainty from out-of-distribution (OOD) inputs
- But... does not provide valid posterior uncertainty
- Folgoc et al. (2021) show that the Monte Carlo dropout posterior predictive assigns zero probability to the true model posterior predictive distribution

Tempered and cold posteriors

Tempered posterior

A tempered posterior distribution with temperature parameter $T > 0$ is defined as

$$p(\boldsymbol{w}|D) \propto \exp(U(\boldsymbol{w})/T)$$

where $U(\boldsymbol{w})$ is the posterior energy function

$$U(\boldsymbol{w}) := \log p(\mathcal{D}|\boldsymbol{w}) + \log p(\boldsymbol{w}),$$

Cold posterior effect

Empirical evidence (Wenzel et al., 2020) that posteriors exponentiated to some power greater than one (or, equivalently, dividing the energy function $U(\boldsymbol{w})$ by some temperature $T < 1$), performs better than an untempered one.

Tempered and cold posteriors

Tempered posterior

A tempered posterior distribution with temperature parameter $T > 0$ is defined as

$$p(\boldsymbol{w}|D) \propto \exp(U(\boldsymbol{w})/T)$$

where $U(\boldsymbol{w})$ is the posterior energy function

$$U(\boldsymbol{w}) := \log p(\mathcal{D}|\boldsymbol{w}) + \log p(\boldsymbol{w}),$$

Cold posterior effect

Empirical evidence (Wenzel et al., 2020) that posteriors exponentiated to some power greater than one (or, equivalently, dividing the energy function $U(\boldsymbol{w})$ by some temperature $T < 1$), performs better than an untempered one.

Outline

Performance metrics

- Predictive performance: ability of the model to give correct answers. Based on metrics, eg: mean square error, risk of 0-1 loss for classification task.
- Model calibration: assessing that the network is neither overconfident nor underconfident about its prediction. Requires using a test set. Eg: expected calibration error (ECE).

References I

[1] Moloud Abdar et al. "A review of uncertainty quantification in deep learning: Techniques, applications and challenges". In: *Information Fusion* (2021).

[2] Julyan Arbel et al. "A primer on Bayesian neural networks: review and debates". In: *Preprint* (2022).

[3] Arsenii Ashukha et al. "Pitfalls of in-domain uncertainty estimation and ensembling in deep learning". In: *International Conference on Learning Representations* (2020).

[4] Neil Band et al. "Benchmarking Bayesian Deep Learning on Diabetic Retinopathy Detection Tasks". In: *NeurIPS Workshop on Distribution Shifts: Connecting Methods and Applications*. 2021.

[5] Andrew R Barron. "Approximation and estimation bounds for artificial neural networks". In: *Machine Learning* 14.1 (1994), pp. 115–133.

[6] Mikhail Belkin et al. "Reconciling modern machine-learning practice and the classical bias–variance trade-off". In: *National Academy of Sciences* 116.32 (2019), pp. 15849–15854.

References II

[7] Vaggos Chatziafratis, Sai Ganesh Nagarajan, Ioannis Panageas, and Xiao Wang. "Depth-width trade-offs for ReLU networks via Sharkovsky's theorem". In: *International Conference on Learning Representations* (2020).

[8] Vaggos Chatziafratis et al. "Better depth-width trade-offs for neural networks through the lens of dynamical systems". In: *International Conference on Machine Learning*. 2020.

[9] George Cybenko. "Approximation by superpositions of a sigmoidal function". In: *Mathematics of Control, Signals and Systems* 2.4 (1989), pp. 303–314.

[10] Loic Le Folgoc et al. "Is MC Dropout Bayesian?" In: *arXiv preprint arXiv:2110.04286* (2021).

[11] Vincent Fortuin. "Priors in Bayesian deep learning: A review". In: *International Statistical Review* (2022).

[12] Ken-Ichi Funahashi. "On the approximate realization of continuous mappings by neural networks". In: *Neural Networks* 2.3 (1989), pp. 183–192.

References III

[13] Yarin Gal and Zoubin Ghahramani. "Dropout as a Bayesian approximation: Representing model uncertainty in deep learning". In: *International Conference on Machine Learning*. 2016.

[14] Ethan Goan and Clinton Fookes. "Bayesian neural networks: An introduction and survey". In: *Case Studies in Applied Bayesian Data Science*. Springer, 2020, pp. 45–87.

[15] Kaiming He et al. "Deep residual learning for image recognition". In: *Computer Vision and Pattern Recognition*. 2016.

[16] Matthias Hein et al. "Why ReLU networks yield high-confidence predictions far away from the training data and how to mitigate the problem". In: *Computer Vision and Pattern Recognition*. 2019.

[17] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural Computation* 9.8 (1997), pp. 1735–1780.

[18] Kurt Hornik et al. "Multilayer feedforward networks are universal approximators". In: *Neural Networks* 2.5 (1989), pp. 359–366.

[19] Arthur Jacot et al. "Neural tangent kernel: Convergence and generalization in neural networks". In: *Advances in neural information processing systems* 31 (2018).

References IV

[20] Laurent Valentin Jospin et al. "Hands-on Bayesian Neural Networks–a Tutorial for Deep Learning Users". In: *arXiv preprint arXiv:2007.06823* (2020).

[21] Laurent Valentin Jospin et al. "Hands-on Bayesian Neural Networks–a Tutorial for Deep Learning Users". In: *arXiv preprint arXiv:2007.06823* (2020).

[22] Diederik P Kingma et al. "Variational dropout and the local reparameterization trick". In: *Advances in Neural Information Processing Systems*. 2015.

[23] Kimin Lee et al. "Training confidence-calibrated classifiers for detecting out-of-distribution samples". In: *International Conference on Learning Representations* (2018).

[24] Matthias Minderer et al. "Revisiting the Calibration of Modern Neural Networks". In: *Advances in Neural Information Processing Systems* (2021).

[25] John Mitros and Brian Mac Namee. "On the validity of Bayesian neural networks for uncertainty estimation". In: *arXiv preprint arXiv:1912.01530* (2019).

References V

[26] Shakir Mohamed et al. "Monte Carlo Gradient Estimation in Machine Learning". In: *J. Mach. Learn. Res.* 21.132 (2020), pp. 1–62.

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

[28] Zachary Nado et al. "Uncertainty Baselines: Benchmarks for uncertainty & robustness in deep learning". In: *arXiv preprint arXiv:2106.04015* (2021).

[29] Radford M Neal. *Bayesian learning for neural networks*. Springer Science & Business Media, 1996.

[30] Yaniv Ovadia et al. "Can you trust your model's uncertainty? Evaluating predictive uncertainty under dataset shift". In: *Advances in Neural Information Processing Systems* (2019).

[31] Ben Poole et al. "Exponential expressivity in deep neural networks through transient chaos". In: *International Conference on Neural Information Processing Systems*. 2016.

[32] Herbert Robbins and Sutton Monro. "A stochastic approximation method". In: *The Annals of Mathematical Statistics* (1951), pp. 400–407.

References VI

[33] David E Rumelhart et al. "Learning representations by back-propagating errors". In: *Nature* 323.6088 (1986), pp. 533–536.

[34] Samuel S Schoenholz et al. "Deep information propagation". In: *International Conference on Learning Representations*. 2017.

[35] Shiliang Sun et al. "A survey of optimization methods from a machine learning perspective". In: *IEEE Transactions on Cybernetics* 50.8 (2019), pp. 3668–3681.

[36] Mariia Vladimirova et al. "Bayesian neural network unit priors and generalized Weibull-tail property". In: *Asian Conference on Machine Learning* (2021). eprint: 2110.02885.

[37] Florian Wenzel et al. "How Good is the Bayes Posterior in Deep Neural Networks Really?" In: *International Conference on Machine Learning*. 2020.

[38] Andrew Gordon Wilson et al. "Deep kernel learning". In: *International Conference on Artificial Intelligence and Statistics*. 2016.