BML lecture #5: Bayesian deep learning

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

Julyan Arbel

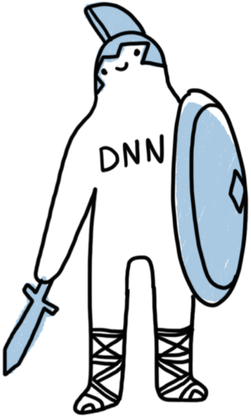
Statify team, Inria Grenoble Rhône-Alpes & Univ. Grenoble-Alpes, France

Outline

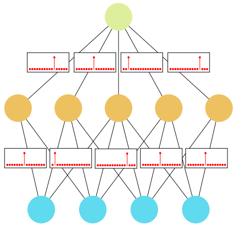
Outline

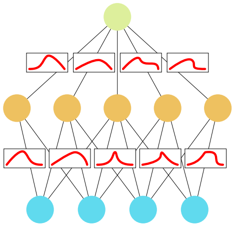
What comes to *your* mind when you hear “Bayesian deep learning”?

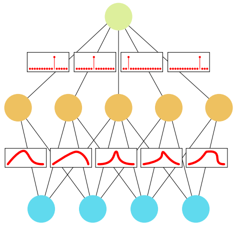
Deep neural networks Achilles heels

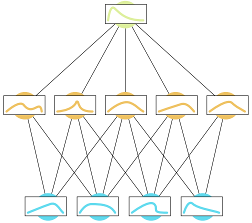


Different flavours of neural networks (Jospin et al., 2020)


Point estimate NN

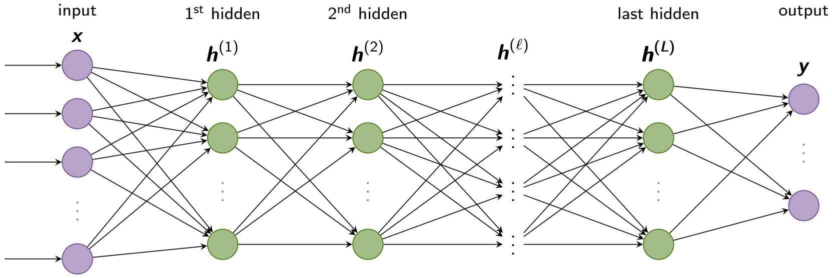

BNN w/ random weights


BNN w/ last-layer rd weights


BNN w/ random activations

Outline

Neural networks notations



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

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

- *nonlinearity* or *activation function* $\varphi : \mathbb{R} \to \mathbb{R}$.
- *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 Sun 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.

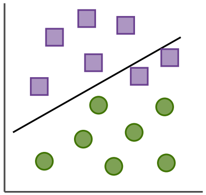
But the size of such networks may be *exponential in the input dimension* N, which makes them highly prone to overfitting as well as impractical.

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

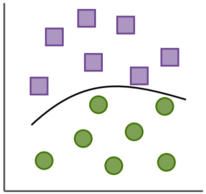
Generalization and overfitting I

Classical regime

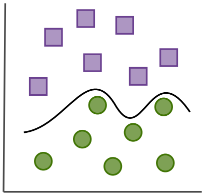
underfitting



optimum



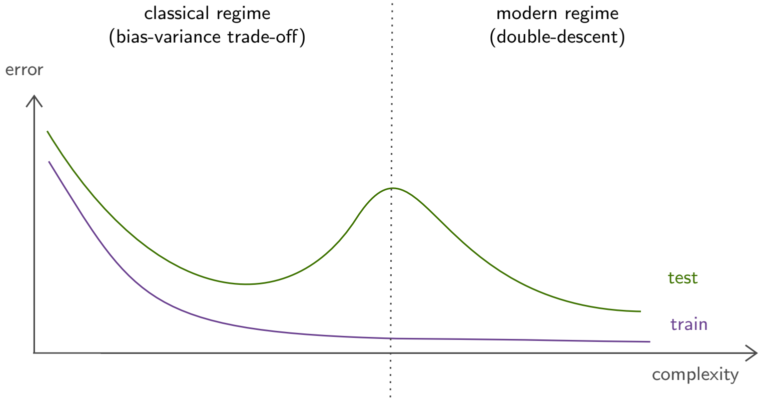
overfitting



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

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 $p(\boldsymbol{\theta}|D) = \frac{p(D_{\boldsymbol{y}}|D_{\boldsymbol{x}}, \boldsymbol{\theta})p(\boldsymbol{\theta})}{\int_{\boldsymbol{\theta}} p(D_{\boldsymbol{y}}|D_{\boldsymbol{x}}, \boldsymbol{\theta'})p(\boldsymbol{\theta'})d\boldsymbol{\theta'}}$;
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 $Y = \{\boldsymbol{y}_i | i \in [0, N)\}, \ \Theta = \{\boldsymbol{\theta}_i | i \in [0, N)\}$;
```

Bayesian neural networks: early algorithms

- Markov chain Monte Carlo (MCMC), Hamiltonian Monte Carlo (HMC). No-U-Turn sampler (NUTS) is 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 *Monte Carlo Gradient Estimation in Machine Learning*, Mohamed et al, JMLR, 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

Outline

Connection to initialization I

In deep learning, initializing neural networks with appropriate weights is crucial to obtaining convergence.

Adequate initialization can help avoid *vanishing* and *exploding gradients*.

Lottery ticket hypothesis: **frankle2019lottery** proposed an iterative algorithm for parameter pruning in neural networks while saving the original initialization of the weights after pruning, also known as the *winning ticket* of the initialization "lottery". Neural networks with such winning tickets could outperform unpruned neural networks.

Typical initialization (Glorot initialization): independently sample each bias $b_i^{(\ell)}$ and each weight $W_{ij}^{(\ell)}$ from zero-mean Gaussian distributions (Glorot and Bengio, 2010):

$$b_i^{(\ell)} \sim \mathcal{N}\left(0, \sigma_b^{(\ell)}\right), \quad W_{ij}^{(\ell)} \sim \mathcal{N}\left(0, \frac{\sigma_w^{(\ell)}}{H_{\ell-1}}\right),$$

Connection to initialization II

for all $i = 1, \ldots, H_\ell$ and $j = 1, \ldots, H_{\ell-1}$, where the normalization of weight variances by $1/H_{\ell-1}$ is conventional to avoid the variance explosion in wide neural networks.

Edge of Chaos I

Poole et al. (2016) and Schoenholz et al. (2017) show that there is a critical line, called Edge of Chaos, separating signal propagation into two regions in $(\sigma_b^{(\ell)}, \sigma_w^{(\ell)})$ initialization plane:

$$w_{ij}^{(\ell)} \sim \mathcal{N}\left(0, \frac{\sigma_w^2}{H_{\ell-1}}\right) \text{ and biases } b_i^{(\ell)} \sim \mathcal{N}(0, \sigma_b^2) \text{ for all } \ell,\ i \text{ and } j.$$

Let

- $\boldsymbol{x}_a$ be an input vector of a data point a.
- $g_{i,a}^{(\ell)}$ be a pre-activation (centered random variable) at layer ℓ given a data point a.
- $q_{aa}^{(\ell)} = \mathbb{E}\left[\left(g_{i,a}^{(\ell)}\right)^2\right]$ the variance of pre-activation at layer ℓ given input a.
- $q_{ab}^{(\ell)} = \mathbb{E}\left[g_{i,a}^{(\ell)} g_{i,b}^{(\ell)}\right]$ the covariance between the pre-activations at layer ℓ given two inputs a and b.

Edge of Chaos II

Two-way recurrence relations:

$$q_{aa}^{(\ell)} = \sigma_w^2 \int \varphi^2\left(u_1^{(\ell-1)}\right) \mathcal{D}g_{i,a} + \sigma_b^2,$$
$$q_{ab}^{(\ell)} = \sigma_w^2 \int \varphi(u_1^{(\ell-1)})\varphi(u_2^{(\ell-1)})\mathcal{D}g_{i,a}\mathcal{D}g_{i,b} + \sigma_b^2,$$

where $u_1^{(\ell-1)} = \sqrt{q_{aa}^{(\ell-1)}}g_{i,a}$,
$u_2^{(\ell-1)} = \sqrt{q_{bb}^{(\ell-1)}}\left(c_{ab}^{(\ell-1)}g_{i,a} + \sqrt{1-(c_{ab}^{(\ell-1)})^2}g_{i,b}\right)$ and
$c_{ab}^{(\ell)} = \frac{q_{ab}^{(\ell)}}{\sqrt{q_{aa}^{(\ell)}}\sqrt{q_{bb}^{(\ell)}}}$. Here, $\mathcal{D}g_{i,a}$ and $\mathcal{D}g_{i,b}$ stand for the distributions of standard Gaussian pre-activations $g_{i,a}$ and $g_{i,b}$.

For any σ_w^2 and σ_b^2, there are limiting points for variance $q^* = \lim_{\ell\to\infty} q_{aa}^{(\ell)}$ and for correlation $c^* = \lim_{\ell\to\infty} c_{ab}^{(\ell)}$. Two regions can be defined depending on the value of c^*: (i) an *ordered* region if $c^* = 1$, as any two inputs a and b, even far from each others, tend to be fully

Edge of Chaos III

correlated in the deep limit $\ell \to \infty$; (ii) a *chaos* region if $c^* < 1$, as any two inputs a and b, even close to each others, tend to decorrelate $\ell \to \infty$.

To study whether the point $c^* = 1$ is *stable*, we need to check the values of the derivative: $\chi_1 = \frac{\partial c_{ab}^{(\ell)}}{\partial c_{ab}^{(\ell-1)}}\Big|_{c_{ab}^{(\ell)}=1}$. There are three cases: (i) *order*, when $\chi_1 < 1$, i.e., the point $c^* = 1$ is stable; (ii) *transition*, when $\chi_1 = 1$; (iii) *chaos*, when $\chi_1 > 1$, i.e., the point $c^* = 1$ is unstable. Therefore, there is a separating line when $c^* = 1$ and $\chi_1 = 1$. By assigning initialization hyperparameters σ_w^2 and σ_b^2 according to the separating line, the information propagates as deep as possible from inputs to outputs, until convergence.

Edge of Chaos IV

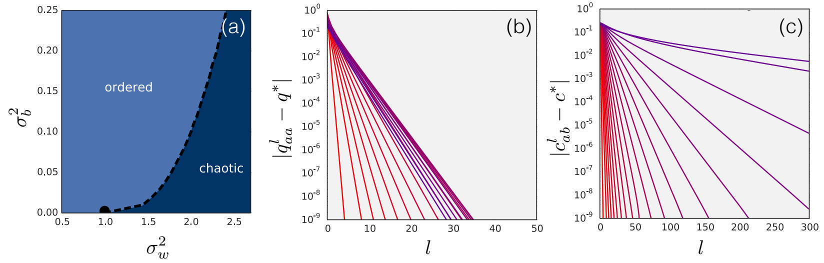


Figure: (a) Edge of chaos diagram showing the boundary between ordered and chaotic phases as a function of σ_w^2 and σ_b^2. (b) The residual $|q^* - q_{aa}^l|$ as a function of depth on a log-scale with $\sigma_b^2 = 0.05$ and σ_w^2 from 0.01 (red) to 1.7 (purple). Clear exponential behavior is observed.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

Adversarial robustness and priors

TBC

Cold posterior effect

TBC

Outline

Outline

Performance metrics

- Predictive performance: ability of the model to give correct answers. Based on metrics (eg mean square error)
- Model calibration: assessing that the network is neither overconfident nor underconfident about its prediction. Requires using a test set.

References I

[1] Arsenii Ashukha, Alexander Lyzhov, Dmitry Molchanov, and Dmitry Vetrov. “Pitfalls of in-domain uncertainty estimation and ensembling in deep learning”. In: *International Conference on Learning Representations* (2020).

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

[3] Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. “Reconciling modern machine-learning practice and the classical bias–variance trade-off”. In: *National Academy of Sciences* 116.32 (2019), pp. 15849–15854.

[4] Vaggos Chatziafratis, Sai Ganesh Nagarajan, and Ioannis Panageas. “Better depth-width trade-offs for neural networks through the lens of dynamical systems”. In: *International Conference on Machine Learning*. 2020.

References II

[5] 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).

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

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

[8] Xavier Glorot and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks". In: *International Conference on Artificial Intelligence and Statistics*. 2010.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition". In: *Computer Vision and Pattern Recognition*. 2016.

References III

[10] Matthias Hein, Maksym Andriushchenko, and Julian Bitterwolf. "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.

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

[12] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. "Multilayer feedforward networks are universal approximators". In: *Neural Networks* 2.5 (1989), pp. 359–366.

[13] Laurent Valentin Jospin, Wray Buntine, Farid Boussaid, Hamid Laga, and Mohammed Bennamoun. "Hands-on Bayesian Neural Networks–a Tutorial for Deep Learning Users". In: *arXiv preprint arXiv:2007.06823* (2020).

[14] Kimin Lee, Honglak Lee, Kibok Lee, and Jinwoo Shin. "Training confidence-calibrated classifiers for detecting out-of-distribution samples". In: *International Conference on Learning Representations* (2018).

References IV

[15] Matthias Minderer, Josip Djolonga, Rob Romijnders, Frances Hubis, Xiaohua Zhai, Neil Houlsby, Dustin Tran, and Mario Lucic. "Revisiting the Calibration of Modern Neural Networks". In: *Advances in Neural Information Processing Systems* (2021).

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

[17] Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, David Sculley, Sebastian Nowozin, Joshua V Dillon, Balaji Lakshminarayanan, and Jasper Snoek. "Can you trust your model's uncertainty? Evaluating predictive uncertainty under dataset shift". In: *Advances in Neural Information Processing Systems* (2019).

[18] Ben Poole, Subhaneil Lahiri, Maithra Raghu, Jascha Sohl-Dickstein, and Surya Ganguli. "Exponential expressivity in deep neural networks through transient chaos". In: *International Conference on Neural Information Processing Systems*. 2016.

References V

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

[20] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. "Learning representations by back-propagating errors". In: *Nature* 323.6088 (1986), pp. 533–536.

[21] Samuel S Schoenholz, Justin Gilmer, Surya Ganguli, and Jascha Sohl-Dickstein. "Deep information propagation". In: *International Conference on Learning Representations*. 2017.

[22] Shiliang Sun, Zehui Cao, Han Zhu, and Jing Zhao. "A survey of optimization methods from a machine learning perspective". In: *IEEE Transactions on Cybernetics* 50.8 (2019), pp. 3668–3681.

[23] Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P Xing. "Deep kernel learning". In: *International Conference on Artificial Intelligence and Statistics*. 2016.