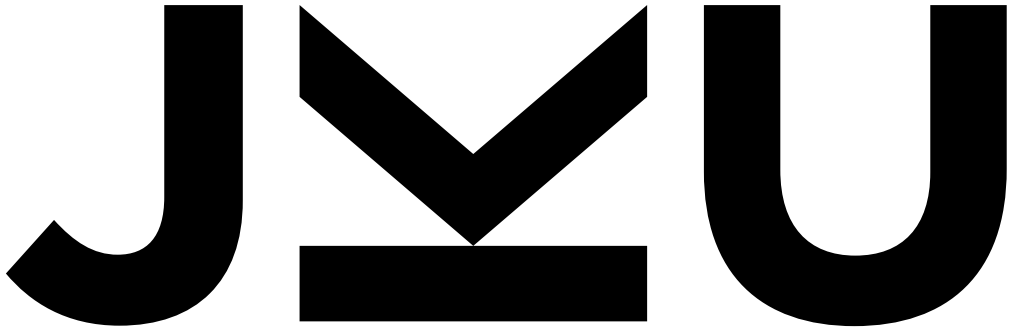
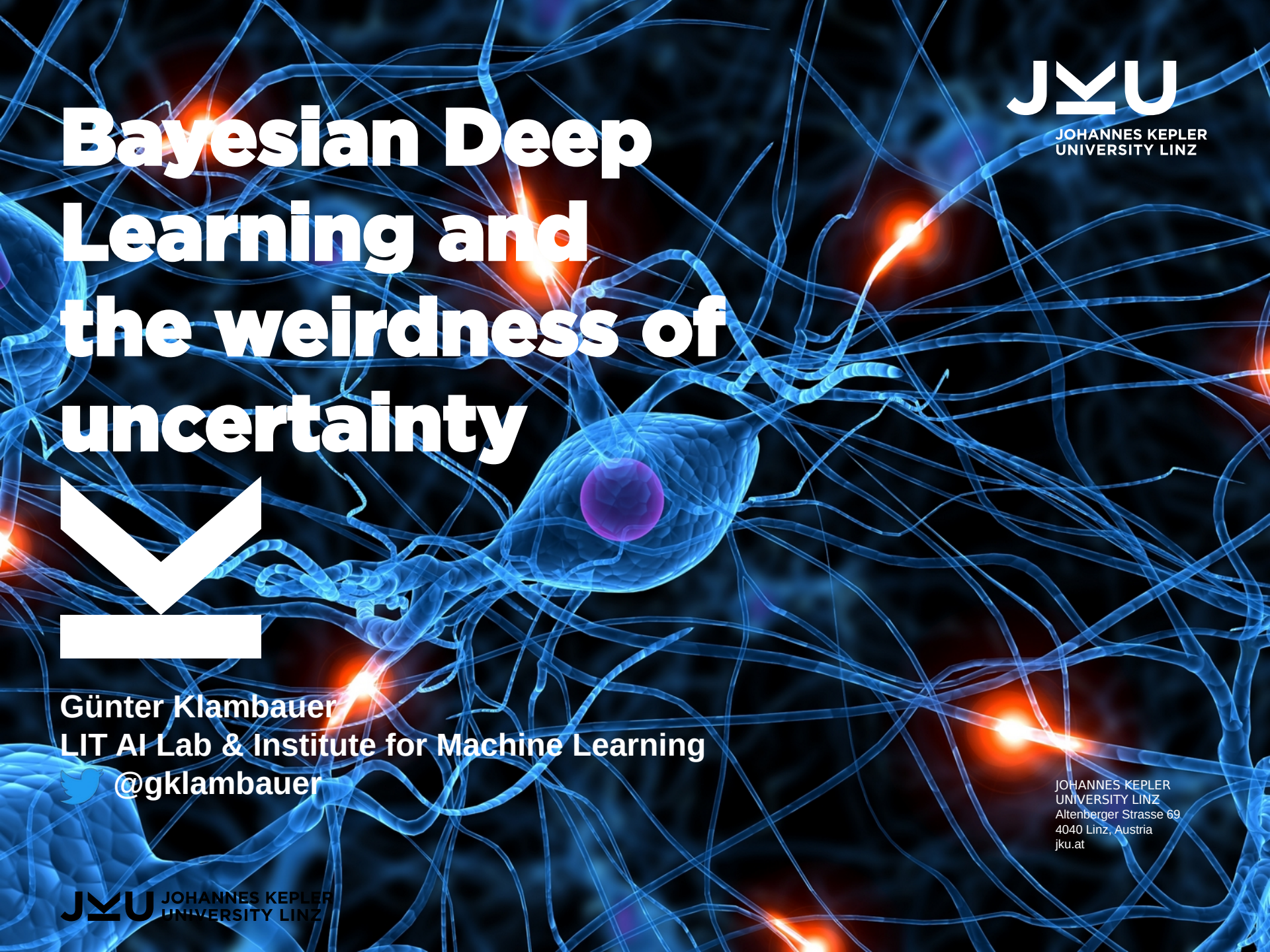


JOHANNES KEPLER
UNIVERSITY LINZ



Bayesian Deep Learning and the weirdness of uncertainty

Günter Klambauer
LIT AI Lab & Institute for Machine Learning
@gklambauer

JOHANNES KEPLER
UNIVERSITY LINZ
Altenberger Strasse 69
4040 Linz, Austria
jku.at

JOHANNES KEPLER UNIVERSITY LINZ

Sources

- NeurIPS Tutorial 2020: Practical Uncertainty Estimation and Out-of-Distribution Robustness in Deep Learning
  https://nips.cc/virtual/2020/protected/tutorial_0f190e6e164eafe66f011073b4486975.html
- Eyke Hüllermeier's recent talks & publications,e.g.
  - Hüllermeier, E., & Waegeman, W. (2021). Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods. Machine Learning, 110(3), 457-506.
- The main papers
  - Hendrycks, D., & Gimpel, K. (2016). A baseline for detecting misclassified and out-of-distribution examples in neural networks. arXiv preprint arXiv:1610.02136.
  - Blundell, C., Cornebise, J., Kavukcuoglu, K., & Wierstra, D. (2015, June). Weight uncertainty in neural network. In International conference on machine learning (pp. 1613-1622). PMLR.
  - Gal, Y., & Ghahramani, Z. (2016, June). Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In international conference on machine learning (pp. 1050-1059). PMLR.
  - Lakshminarayanan, B., Pritzel, A., & Blundell, C. (2017). Simple and scalable predictive uncertainty estimation using deep ensembles. Advances in neural information processing systems, 30.
  - Malinin, A., & Gales, M. (2018). Predictive uncertainty estimation via prior networks. Advances in neural information processing systems, 31.

Literature map:
https://app.litmaps.com/shared/map/9e9fa355-5c04-4ef8-b133-570c1b2dfc8a

Outline

- 0. Background
- 1. The weirdness of uncertainty
- 2. Intro to Bayesian Deep Learning
- 3. Bayesian Deep Learning approaches
  - 3.1 Weight uncertainty in DNNs (“Bayes by backprop”)
  - 3.2 Deep networks and Gaussian processes
  - 3.3 Monte-Carlo Dropout
  - 3.4. Deep Ensembles
- 4. Comparison
- 5. Summary

0. Notation

- The machine learning model $g(\boldsymbol{x}; \boldsymbol{w})$ will be $p(\boldsymbol{y} \mid \boldsymbol{w}, \boldsymbol{x})$ or at times $p_{\boldsymbol{w}}(\boldsymbol{y} \mid \boldsymbol{x})$.
- We will use $\mathcal{D} = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), \ldots, (\boldsymbol{x}_N, \boldsymbol{y}_N)\}$ to denote the labeled training data.
- The prediction $\hat{y}$ will become $\hat{p}$ or $\hat{p}_k$.
- A new data point that we aim to predict will be denoted as $(\boldsymbol{x}^{\text{test}}, \boldsymbol{y}^{\text{test}})$ or just: $(\boldsymbol{x}, \boldsymbol{y})$
- A model class will be denoted with $\mathcal{M}$

0. Likelihood, Prior, Posterior, Evidence

- **Maximum likelihood:** find the most likely parameters given data

$$L(\mathcal{D}; \boldsymbol{w}) = p(\mathcal{D} \mid \boldsymbol{w})$$

- However, for complex models this leads to overfitting. Therefore, some $\boldsymbol{w}$ should be more likely than others
  - Prior distribution over parameters: $p(\boldsymbol{w})$
  - Bayes formula $p(\boldsymbol{w} \mid \mathcal{D}) = \dfrac{p(\mathcal{D} \mid \boldsymbol{w})\ p(\boldsymbol{w})}{p_{\boldsymbol{w}}(\mathcal{D})}$
  - Important note! Usually: $p(\mathcal{D}) \neq p_{\boldsymbol{w}}(\mathcal{D})$
- Maximum A Posteriori: (last term does not depend on params)

$$-\log p(\boldsymbol{w} \mid \mathcal{D}) = -\log p(\mathcal{D}|\boldsymbol{w}) - \log p(\boldsymbol{w}) + \log p_{\boldsymbol{w}}(\mathcal{D})$$

0. Maximum A Posteriori (MAP) approach for Deep Nets: a usual case

- Usually, the aim is to find the most likely parameters given data

$$\begin{aligned}\tilde{\boldsymbol{w}} &= \underset{\boldsymbol{w}}{\operatorname{argmax}}\; p(\boldsymbol{w} \mid \boldsymbol{x}, \boldsymbol{y}) \\ &= \underset{\boldsymbol{w}}{\operatorname{argmin}} - \log p(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{w}) - \log p(\boldsymbol{w})\end{aligned}$$

- For example, softmax & cross-entropy at output and L2 regularization lead to

$$\tilde{\boldsymbol{w}} = \underset{\boldsymbol{w}}{\operatorname{argmin}} \sum_k y_k \log(\hat{p}_k) + \lambda \|\boldsymbol{w}\|^2$$

- Note: $p(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{w})$ should represent uncertainty about the label. However, we can make only a single prediction with the parameters
- Still a point estimate! No full Bayesian treatment! We want a full distribution over parameters!

0. A Bayesian approach: a distribution of parameters is wanted; supervised!

- Bayesian setting:

$$p(\boldsymbol{w} \mid \boldsymbol{x}, \boldsymbol{y}) = \frac{p(\boldsymbol{y} \mid \boldsymbol{w}, \boldsymbol{x}) \cdot p(\boldsymbol{w})}{\int_{\mathcal{W}} p(\boldsymbol{y} \mid \boldsymbol{w}, \boldsymbol{x}) \cdot p(\boldsymbol{w}) d\boldsymbol{w}} = \frac{p(\boldsymbol{y} \mid \boldsymbol{w}, \boldsymbol{x}) \cdot p(\boldsymbol{w})}{p(\boldsymbol{y} \mid \boldsymbol{x})}$$

- Posterior: $p(\boldsymbol{w} \mid \boldsymbol{x}, \boldsymbol{y})$
- Prior: $p(\boldsymbol{w})$
- Likelihood: $p(\boldsymbol{y} \mid \boldsymbol{w}, \boldsymbol{x})$
- Evidence: $\int_{\mathcal{W}} p(\boldsymbol{y} \mid \boldsymbol{w}, \boldsymbol{x}) \cdot p(\boldsymbol{w}) d\boldsymbol{w}$
  - Usually intractable; especially for DNNs
  - If all other distributions are Gaussian: tractable. “Gaussian process”

0. Example: Gaussian data, Gaussian prior

- Data: $x \mid \mu \sim \mathcal{N}(\mu, \sigma_x^2)$ Pior: $\mu \sim \mathcal{N}(\nu, \sigma_\mu^2)$
- Parameter posterior

$$p(\mu \mid x) = \frac{p(x \mid \mu)p(\mu)}{\int_{-\infty}^{\infty} p(x \mid m)p(m)dm} \propto p(x \mid \mu)p(\mu)$$

$$p(x \mid \mu)p(\mu) = \frac{1}{\sqrt{2\pi\sigma_x^2}} e^{-\frac{(x-\mu)^2}{2\sigma_x^2}} \frac{1}{\sqrt{2\pi\sigma_\mu^2}} e^{-\frac{(\mu-\nu)^2}{2\sigma_\mu^2}}$$

- **Completing squares in exponent** and assuming normalized distributions, we see that this is again a Gaussian

$$\mu \mid x \sim \mathcal{N}\left(\frac{\mu\sigma_\mu^2 + \nu\sigma_x^2}{\sigma_\mu^2 + \sigma_x^2}, \frac{1}{1/\sigma_x^2 + 1/\sigma_\mu^2}\right)$$

0. Difference between frequentist and Bayesian approach: marginalization

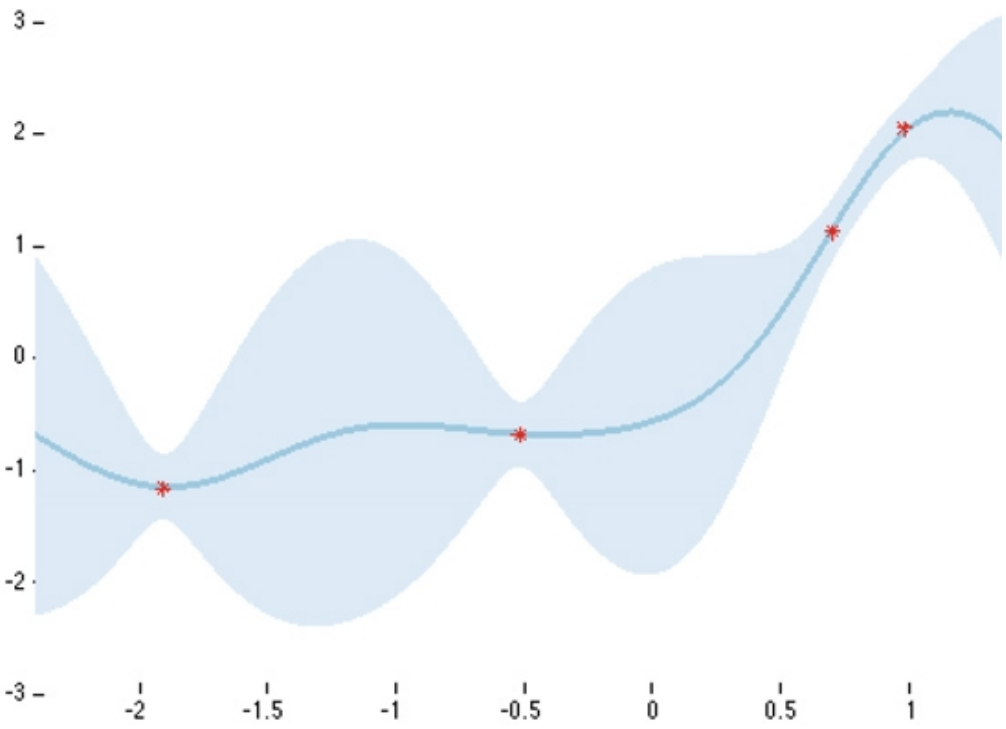
- **Full predictive distribution** / distribution of outputs:

$$p(\boldsymbol{y}^{\text{test}} \mid \boldsymbol{x}^{\text{test}}, \mathcal{D}) = \mathbb{E}_{\boldsymbol{w} \sim p(\boldsymbol{w}|\mathcal{D})} \left[p(\boldsymbol{y}^{\text{test}} \mid \boldsymbol{w}, \boldsymbol{x}^{\text{test}}) \right],$$
$$= \int_{\mathcal{W}} p(\boldsymbol{y}^{\text{test}} \mid \boldsymbol{w}, \boldsymbol{x}^{\text{test}}) p(\boldsymbol{w} \mid \mathcal{D}) d\boldsymbol{w}$$

- Defines probability for class label given input and dataset
- Marginalization over parameters
- “Bayesian model averaging (BMA)”

0. Full predictive distribution

- Linear regression example
- Intuitively:
  - Uncertainty close to data points is small
  - High uncertainty elsewhere
- However: high-dimensional spaces



1. Introductory example: Intuition

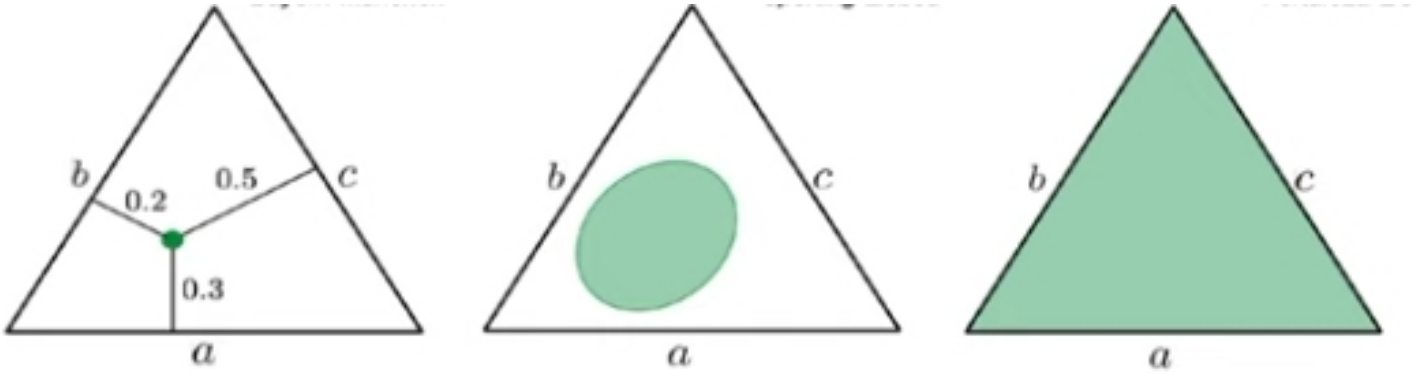
- Assume you train a QSAR model $p(y \mid \boldsymbol{x}, \boldsymbol{w})$ that should predict binary activity (“active” vs “inactive”).
- It provides for a particular molecule:

$$p(y = 1 \mid \boldsymbol{x}, \boldsymbol{w}) = 0.5$$

- Is the model uncertain about it’s prediction?
- Correct answer: we don’t know!
  - A) It could be the perfect model, but the assay is random (for this molecule or in general)
    the prediction is correct: everytime one measures the molecule it is 50% active and 50% inactive; → **aleatoric uncertainty**
  - B) The model is garbage and the molecule is indeed inactive (or: active); uncertainty about the parameters → **epistemic uncertainty**

1. Weirdness of uncertainy

- Levels of uncertainty representations

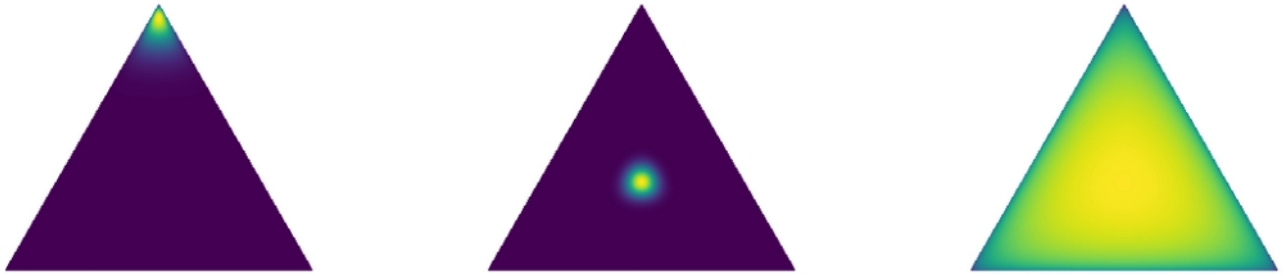


- Note: the softmax output of a neural network “pretends” to have no *epistemic uncertainty* and to just provide *aleatoric uncertainty*

Eyke Hüllermeier: "Uncertainty Quantification in Machine Learning: From Aleatoric to Epistemic I"; https://www.youtube.com/watch?v=vvUUmk9qfuA
Malinin, A., & Gales, M. (2018). Predictive uncertainty estimation via prior networks. Advances in neural information processing systems, 31.

1. Weirdness of uncertainy

- Levels of uncertainty representations



(a) Confident Prediction (b) High data uncertainty (c) Out-of-distribution

- Note: the softmax output of a neural network “pretends” to have no *epistemic uncertainty* and to just provide *aleatoric uncertainty*

Eyke Hüllermeier: "Uncertainty Quantification in Machine Learning: From Aleatoric to Epistemic I"; https://www.youtube.com/watch?v=vvUUmk9qfuA
Malinin, A., & Gales, M. (2018). Predictive uncertainty estimation via prior networks. Advances in neural information processing systems, 31.

1. Weirdness of uncertainy
Definition of uncertainty

- *Total uncertainty* often defined as entropy of the predictive distribution (e.g. Gal, 2016; Hüllermeier, 2021):

$$\mathrm{H}\left[p(\boldsymbol{y} \mid \boldsymbol{x}, \mathcal{D})\right] \;=\; \int_{\mathcal{W}} \mathrm{H}\left[p(\boldsymbol{y} \mid \boldsymbol{x}, \tilde{\boldsymbol{w}})\right] \; p(\tilde{\boldsymbol{w}} \mid \mathcal{D}) \; \mathrm{d}\tilde{\boldsymbol{w}} \;+\; \mathrm{I}(Y, W) \;.$$

- The *aleatoric uncertainty* is given by

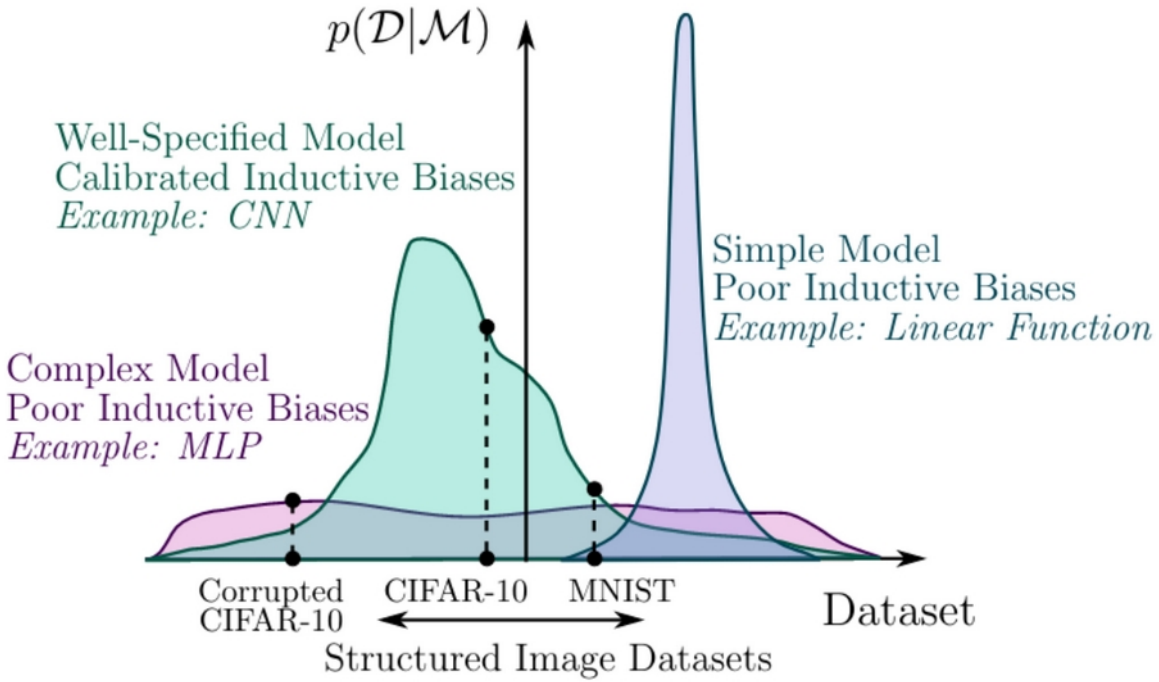
$$\mathbb{E}_{p(\tilde{\boldsymbol{w}}|\mathcal{D})}\left[\mathrm{H}\left[p(\boldsymbol{y} \mid \boldsymbol{x}, \tilde{\boldsymbol{w}})\right]\right] \;= -\; \int_{\mathcal{W}} \mathrm{H}\left[p(\boldsymbol{y} \mid \boldsymbol{x}, \tilde{\boldsymbol{w}})\right] \; p(\tilde{\boldsymbol{w}} \mid \mathcal{D}) \; \mathrm{d}\tilde{\boldsymbol{w}}$$

- *Epistemic uncertainty* obtained as difference

$$\begin{aligned}\mathrm{I}(Y; W) \;&=\; \mathrm{H}\left[p(\boldsymbol{y} \mid \boldsymbol{x}, \mathcal{D})\right] \;-\; \mathbb{E}_{p(\tilde{\boldsymbol{w}}|\mathcal{D})}\left[\mathrm{H}\left[p(\boldsymbol{y} \mid \boldsymbol{x}, \tilde{\boldsymbol{w}})\right]\right] \\ &=\; \int_{\mathcal{W}} \left(\mathrm{H}\left[p(\boldsymbol{y} \mid \boldsymbol{x}, \mathcal{D})\right] \;-\; \mathrm{H}\left[p(\boldsymbol{y} \mid \boldsymbol{x}, \tilde{\boldsymbol{w}})\right]\right) \; p(\tilde{\boldsymbol{w}} \mid \mathcal{D}) \; \mathrm{d}\tilde{\boldsymbol{w}}\end{aligned}$$

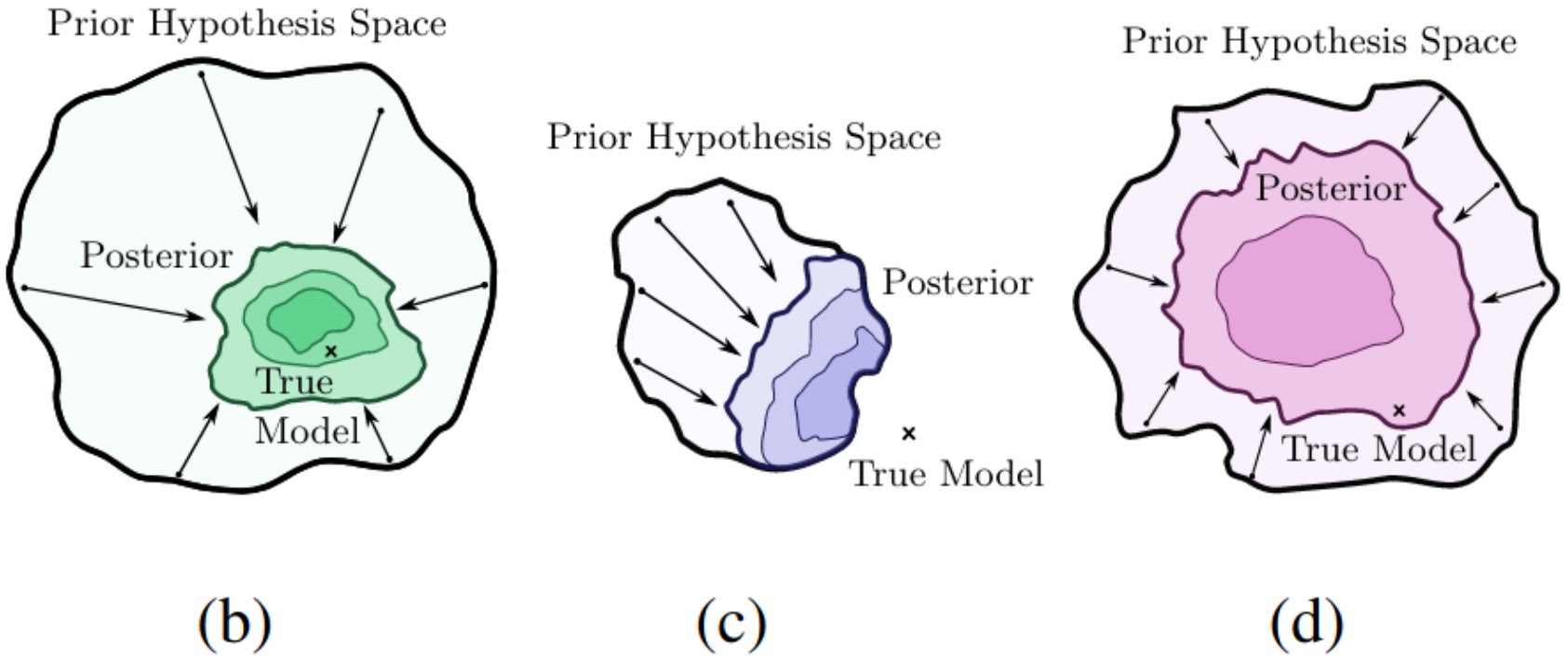
$$p(\boldsymbol{y} \mid \boldsymbol{x}, \mathcal{D}) \;=\; p(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{w}^*) \;=\; \int_{\mathcal{W}} p(\boldsymbol{y} \mid \boldsymbol{x}, \tilde{\boldsymbol{w}}) \; p(\tilde{\boldsymbol{w}} \mid \mathcal{D}) \; \mathrm{d}\tilde{\boldsymbol{w}}$$

2. Neural network generalization



- How is model class performance (~*inductive bias*) distributed over the range of all possible datasets (*support*)

2. A probabilistic perspective of generalization



3. A selection of Bayesian DL approaches

- 1. Weight uncertainty in DNNs (variational approach)
- 2. Infinite Width Bayesian Deep Networks are Gaussian Processes
- 3. Monte-Carlo Dropout
- 4. Deep Ensembles
- 5. Hessian-based approach

3.1 Variational approach: Weight Uncertainty in DNNs

- Approximate predictive distribution
- Approach: **“Bayes-By-Backprop”**
  Weight uncertainty in neural networks (Blundell et al., 2015).
- Variational approach, learning finds the parameters $\boldsymbol{w}$ of a parameterized distribution of the weights $q(\boldsymbol{w} \mid \boldsymbol{\theta})$
- New new set of parameters $\boldsymbol{\theta}$ that we did not have before.
  - parameters determine the distribution of weights,
  - e.g. mean and standard deviation of Gaussian

Blundell, C., Cornebise, J., Kavukcuoglu, K., & Wierstra, D. (2015, June). Weight uncertainty in neural network. In International Conference on Machine Learning (pp. 1613-1622). PMLR.

3.1 Weight Uncertainty in DNNs

- Gaussian approximation of the parameter posterior
- Minimization of KL divergence with true posterior

$$
\begin{aligned}
\tilde{\boldsymbol{\theta}} &= \operatorname{argmin}_{\boldsymbol{\theta}} \boxed{\mathrm{KL}\left(q(\boldsymbol{w} \mid \boldsymbol{\theta}) \mid\mid p(\boldsymbol{w} \mid \mathcal{D})\right)} \\
&= \operatorname{argmin}_{\boldsymbol{\theta}} \int_{\mathcal{W}} q(\boldsymbol{w} \mid \boldsymbol{\theta}) \log \frac{q(\boldsymbol{w} \mid \boldsymbol{\theta})}{p(\boldsymbol{w}) p(\mathcal{D} \mid \boldsymbol{w})} d\boldsymbol{w} \\
&= \operatorname{argmin}_{\boldsymbol{\theta}} \boxed{\mathrm{KL}\left(q(\boldsymbol{w} \mid \boldsymbol{\theta}) \mid\mid p(\boldsymbol{w})\right)} - \mathbb{E}_{\boldsymbol{w} \sim q(\boldsymbol{w} \mid \boldsymbol{\theta})}[\log p(\mathcal{D} \mid \boldsymbol{w})]
\end{aligned}
$$

3.1 Weight Uncertainty in DNNs

- Objective function (ELBO):

$$L(\boldsymbol{\theta}, \mathcal{D}) = L(\boldsymbol{\theta}, \boldsymbol{X}, \boldsymbol{Y}) = \mathrm{KL}\left(q(\boldsymbol{w} \mid \boldsymbol{\theta}) \mid\mid p(\boldsymbol{w})\right) - \mathbb{E}_{\boldsymbol{w} \sim q(\boldsymbol{w} \mid \boldsymbol{\theta})}[\log p(\mathcal{D} \mid \boldsymbol{w})]$$

  - Similar to objective in VAEs
  - Involves sampling from posterior estimates
  - How to approach this with gradient descent? We cannot backpropagate to a sampling procedure

3.1 Weight Uncertainty in DNNs: Reparametrization trick

- We assume that weights are determined by some function that has both a deterministic and a random part:

$$\boldsymbol{w} = h(\boldsymbol{\theta}, \boldsymbol{\epsilon})$$

  - ϵ: noise
  - $\boldsymbol{\theta}$: parameters of probability distribution
  - Similar to reparametrization trick in VAEs
  - E.g. mean and variance of Gaussian plus standard noise

3.1 Weight Uncertainty in DNNs: Reparametrization trick

- Backpropagation to parameters of probability distribution

$$\frac{\partial}{\partial \boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{w} \sim q(\boldsymbol{w}|\theta)}[f(\boldsymbol{w}, \boldsymbol{\theta})] = \frac{\partial}{\partial \boldsymbol{\theta}} \int f(\boldsymbol{w}, \boldsymbol{\theta}) q(\boldsymbol{w} \mid \boldsymbol{\theta}) d\boldsymbol{w} =$$

$$= \frac{\partial}{\partial \boldsymbol{\theta}} \int f(\boldsymbol{w}, \boldsymbol{\theta}) q(\boldsymbol{\epsilon}) d\boldsymbol{\epsilon}$$

$$= \mathbb{E}_{\boldsymbol{\epsilon} \sim q(\boldsymbol{\epsilon})} \left[\frac{\partial f(\boldsymbol{w}, \boldsymbol{\theta})}{\partial \boldsymbol{w}} \frac{\partial \boldsymbol{w}}{\partial \boldsymbol{\theta}} + \frac{\partial f(\boldsymbol{w}, \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \right]$$

- We used Leibnitz integral rule to switch integration and differentiation
- We use: $f(\boldsymbol{w}, \boldsymbol{\theta}) = \log q(\boldsymbol{w} \mid \boldsymbol{\theta}) - \log p(\boldsymbol{w}) - \log p(\mathcal{D} \mid \boldsymbol{w})$

3.1 Weight Uncertainty in DNNs: Reparametrization trick

- With $f(\boldsymbol{w}, \boldsymbol{\theta}) = \log q(\boldsymbol{w} \mid \boldsymbol{\theta}) - \log p(\boldsymbol{w}) - \log p(\mathcal{D} \mid \boldsymbol{w})$ we approximate the objective function by

$$L(\boldsymbol{\theta}, \mathcal{D}) \approx \sum_{m=1}^{M} \log q(\boldsymbol{w}^{(m)} \mid \boldsymbol{\theta}) - \log p(\boldsymbol{w}^{(m)}) - \underbrace{\log p(\mathcal{D} \mid \boldsymbol{w}^{(m)})}_{\text{usual obj of a DNN}}$$

- Where $\boldsymbol{w}^{(m)}$ is a parameter vector randomly drawn from the estimated posterior distribution $q(\boldsymbol{w} \mid \boldsymbol{\theta})$

3.1 Weight Uncertainty in DNNs: Algorithm

- Sample $\varepsilon \sim \mathcal{N}(0, 1)$,
- set $\boldsymbol{w} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \varepsilon$
- let $\boldsymbol{\theta} = (\boldsymbol{\mu}, \boldsymbol{\sigma})$,
- let $f(\boldsymbol{w}, \boldsymbol{\theta}) = \log q(\boldsymbol{w} \mid \boldsymbol{\theta}) - \log p(\boldsymbol{w}) - \log p(\mathcal{D} \mid \boldsymbol{w})$,
- calculate gradients of $f(\boldsymbol{w}, \boldsymbol{\theta})$ with respect to $\boldsymbol{\mu}$ using backprop:

$$\Delta_{\boldsymbol{\mu}} = \frac{\partial f(\boldsymbol{w}, \boldsymbol{\theta})}{\partial \boldsymbol{w}} \odot \mathbf{1} + \frac{\partial f(\boldsymbol{w}, \boldsymbol{\theta})}{\partial \boldsymbol{\mu}}$$

- calculate gradients of $f(\boldsymbol{w}, \boldsymbol{\theta})$ with respect to $\boldsymbol{\sigma}$ using backprop:

$$\Delta_{\boldsymbol{\sigma}} = \frac{\partial f(\boldsymbol{w}, \boldsymbol{\theta})}{\partial \boldsymbol{w}} \odot \varepsilon + \frac{\partial f(\boldsymbol{w}, \boldsymbol{\theta})}{\partial \boldsymbol{\sigma}}$$

- and update the variational parameters:

$$\boldsymbol{\mu}^{\text{new}} = \boldsymbol{\mu}^{\text{old}} - \eta \Delta_{\boldsymbol{\mu}}$$
$$\boldsymbol{\sigma}^{\text{new}} = \boldsymbol{\sigma}^{\text{old}} - \eta \Delta_{\boldsymbol{\sigma}}$$

These are the usual gradients from backprop

3.2 Infinite Width Bayesian Deep Networks are Gaussian Processes

- We revisit linear regression:

$$g(\boldsymbol{x}) = \hat{y} = \boldsymbol{w}^T f(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{h}$$

- Now we assume a Gaussian prior over the parameters

$$p(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{w} \mid \boldsymbol{0}, \sigma^2 \boldsymbol{I})$$

- We stack representations and labels of different samples

$$\hat{\boldsymbol{y}} = \boldsymbol{H}\boldsymbol{w}$$

- Then we calculate mean and covariance to define $\boldsymbol{K}$

$$\mathbb{E}[\hat{\boldsymbol{y}}] = \boldsymbol{H}\,\mathbb{E}[\boldsymbol{w}] = \boldsymbol{0}$$

$$\mathrm{Cov}[\hat{\boldsymbol{y}}] = \mathbb{E}[\hat{\boldsymbol{y}}\hat{\boldsymbol{y}}^T] = \boldsymbol{H}\,\mathbb{E}[\boldsymbol{w}\boldsymbol{w}^T]\boldsymbol{H}^T = \sigma^2 \boldsymbol{H}\boldsymbol{H}^T = \boldsymbol{K}$$

J

3.2 Gaussian processes

- A Gaussian process is defined as probability distribution over functions $g(\boldsymbol{x})$
- such that the set of values of $g(\boldsymbol{x})$ evaluated at a set $\{\boldsymbol{x}^1, \ldots, \boldsymbol{x}^n\}$ is jointly a Gaussian distribution.
- Gaussian distribution is fully determined by its first two moments, **mean** and **covariance**
- The mean vector must be assumed to be zero 0, such that the specification is mostly determined by the **covariance**
- Covariance is determined by the kernel function and thus the pairwise similarity of data points in some space

3.2 Connection of neural networks and Gaussian processes

- Discovered by Neal (1994) in his PhD thesis as pessimistic result
- Two-layer network with weights $v_{jd} \sim \mathcal{N}(0, \sigma_w^2)$ and $w_{kj} \sim \mathcal{N}(0, \sigma_w^2)$.
- Pre-activations have identical means and second moments

$$\mathbb{E}[\textstyle\sum_{d=1}^{D} x_d v_{jd}] = 0 \qquad \mathbb{E}[s_j^2(\boldsymbol{x})] = \mathbb{E}[\textstyle\sum_{d=1}^{D} x_d^2 v_{jd}^2] = \mathbb{E}[s_l^2(\boldsymbol{x})]$$

- Activations in hidden layer have some mean m_h and variance V_h.
- Distributions of outputs:

$$\mathbb{E}[o_k(\boldsymbol{x})] = \mathbb{E}[\sum_{j=1}^{J} w_{kj} h_j(\boldsymbol{x})] = J\,\mathbb{E}[w_{kj}]\,\mathbb{E}[h_1(\boldsymbol{x})] = 0.$$

$$\mathbb{E}[(o_k(\boldsymbol{x})^2] = \mathbb{E}[\sum_{j=1}^{J} (w_{kj} h_j(\boldsymbol{x}))^2] = J\,\mathbb{E}[w_{kj}^2]\,\mathbb{E}[(h_1(\boldsymbol{x}))^2] = J\sigma_w^2 V_h.$$

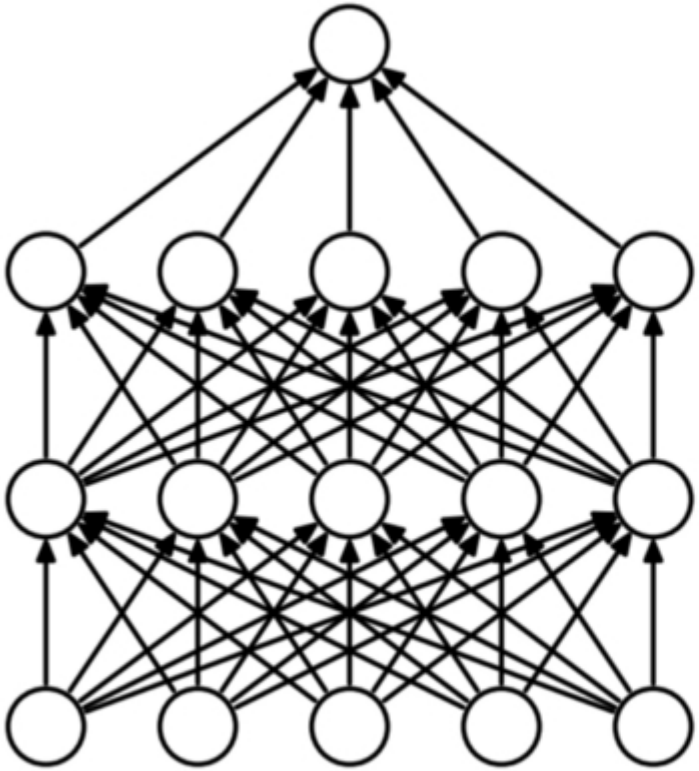
3.2 Connection of neural networks and Gaussian processes

- We can use the *Central Limit Theorem* to show that outputs are Gaussian distributed
- The joint distribution of two data points then is

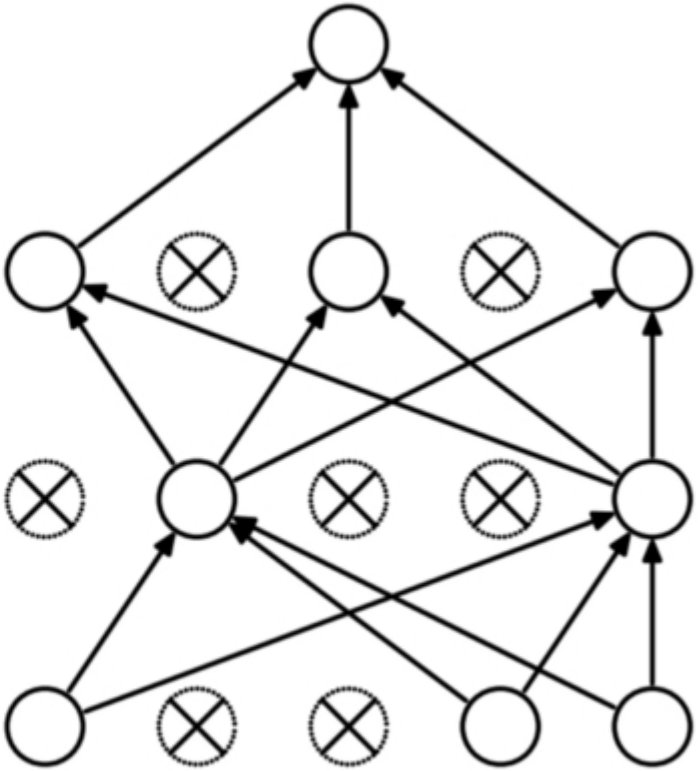
$$\mathbb{E}[o_k(\boldsymbol{x}_n)o_k(\boldsymbol{x}_m)] = \sum_{j=1}^{J} \mathbb{E}[w_{jk}h_j(\boldsymbol{x}_n)w_{jk}h_j(\boldsymbol{x}_m)] = \sigma_w^2 V_h K(\boldsymbol{x}_n, \boldsymbol{x}_m)$$

- Gaussian for multiple data points, thus a Gaussian process whose covariance depends on kernel between data points
- Covariance between two different output units is zero
- Connection is often used for Deep Learning theory
  - Also for *Neural Tangent Kernels*

3.3 Monte-Carlo (MC) Dropout



(a) Standard Neural Net



(b) After applying dropout.

3.3 MC Dropout

- Similar derivation to *Weight Uncertainty in DNNs*
- Use of the equivalence to Gaussian processes
- Consider a two-layer network
- Intractable posterior; approximation with $q(\boldsymbol{w})$
  - Weights assumed to come from mixture

$$q(\mathbf{W}_1) = \prod_{q=1}^{Q} q(\mathbf{w}_q),$$

$$q(\mathbf{w}_q) = p_1 \mathcal{N}(\mathbf{m}_q, \boldsymbol{\sigma}^2 \mathbf{I}_K) + (1 - p_1) \mathcal{N}(0, \boldsymbol{\sigma}^2 \mathbf{I}_K)$$

  - Applying Dropout is like drawing from this posterior estimates
  - Objective is again ELBO approximated with sampling

3.3 MC Dropout

- Two-layer network (one hidden layer):

$$p(y \mid \boldsymbol{x}, \boldsymbol{W}^{[1]}, \boldsymbol{W}^{[2]}) = \text{softmax}(\boldsymbol{W}^{[2]} f(\boldsymbol{W}^{[1]} \boldsymbol{x}))$$

- Gaussian prior on weights

$$p(W^{[1]}) = \mathcal{N}(0, \boldsymbol{I}) \quad \text{and} \quad p(W^{[2]}) = \mathcal{N}(0, \boldsymbol{I})$$

- Full predictive distribution

$$p(\boldsymbol{y}^{\text{test}} \mid \boldsymbol{x}^{\text{test}}, \mathcal{D}) = \int_{\mathcal{W}} p(\boldsymbol{y}^{\text{test}} \mid \boldsymbol{W}^{[1]}, \boldsymbol{W}^{[2]}, \boldsymbol{x}^{\text{test}}) \underbrace{p(\boldsymbol{W}^{[1]}, \boldsymbol{W}^{[2]} \mid \mathcal{D})}_{\text{untractable}} d\boldsymbol{W}^{[1]} d\boldsymbol{W}^{[2]}$$

- Minimizing KL-divergence and connection to ELBO (see VAEs)

$$D_{KL}(q(\boldsymbol{W}^{[1]}, \boldsymbol{W}^{[2]}) || p(\boldsymbol{W}^{[1]}, \boldsymbol{W}^{[2]} \mid \mathcal{D}))$$

$$D_{KL}(q(\boldsymbol{W}^{[1]}, \boldsymbol{W}^{[2]}) || p(\boldsymbol{W}^{[1]}) p(\boldsymbol{W}^{[2]})) - \sum_{n=1}^{N} \int q(\boldsymbol{W}^{[1]}, \boldsymbol{W}^{[2]}) \log(y_n \mid x_n, \boldsymbol{W}^{[1]}, \boldsymbol{W}^{[2]})$$

- With variational distribution; mixture of Gaussians

$$q(\boldsymbol{W}^{[1]}, \boldsymbol{W}^{[2]}) = q(\boldsymbol{W}^{[1]}) q(\boldsymbol{W}^{[2]})$$

$$q(\boldsymbol{W}) = p\mathcal{N}(\boldsymbol{M}, \sigma^2 \boldsymbol{I}) + (1-p)\mathcal{N}(0, \sigma^2 \boldsymbol{I})$$

3.3 MC Dropout

- KL divergence between variational and prior

$$D_{KL}(q(\boldsymbol{W}^{[l]})||p(\boldsymbol{W}^{[l]})) \approx dI(\sigma^2 - \log(\sigma^2) - 1) + p/2||\boldsymbol{W}^{[l]}||_2^2 + \text{const}$$

- Hence: weight matrices sampled from

$$\boldsymbol{W}^{[l]} \approx \text{diag}(\boldsymbol{z}^{[l]})\boldsymbol{M}^{[l]}, \boldsymbol{z} \sim \text{Bernoulli}(p)$$

- Objective:

$$D_{KL}(q(\boldsymbol{W}^{[1]}, \boldsymbol{W}^{[2]}||p(\boldsymbol{W}^{[1]}, \boldsymbol{W}^{[2]} \mid \mathcal{D})) \leq -\sum_{n=1}^{N} \log p(y_n \mid x_n, \boldsymbol{W}_n^{[1]}, \boldsymbol{W}_n^{[2]}) + p^{[1]}/2||\boldsymbol{M}_1||_2^2 + p^{[2]}/2||\boldsymbol{M}_2||_2^2$$

where $\boldsymbol{W}_n = \text{diag}(\boldsymbol{z}_n)\boldsymbol{M}, \quad \boldsymbol{z} \sim \text{Bernoulli}(p)$ are Monte Carlo samples

Same loss as used when training with dropout! (and reg.)

3.4 Deep Ensembles

- Main idea: approximate expectation with averages

$$p(\boldsymbol{y}^{\text{test}} \mid \boldsymbol{x}^{\text{test}}, \mathcal{D}) = \mathbb{E}_{\boldsymbol{w} \sim p(\boldsymbol{w}|\mathcal{D})} \left[p(\boldsymbol{y}^{\text{test}} \mid \boldsymbol{w}, \boldsymbol{x}^{\text{test}}) \right]$$

- Train several DNNs with different random seeds, obtain parameters $\boldsymbol{w}^{(m)}$ and take average

$$p(\boldsymbol{y}^{\text{test}} \mid \boldsymbol{x}^{\text{test}}, \mathcal{D}) \approx \frac{1}{M} \sum_{m=1}^{M} p(\boldsymbol{y}^{\text{test}} \mid \boldsymbol{w}^{(m)}, \boldsymbol{x}^{\text{test}},)$$

- Averages can be weighted differently
- Appears as simple technique, but performs well in practice

Lakshminarayanan, B., Pritzel, A., & Blundell, C. (2017, December). Simple and scalable predictive uncertainty estimation using deep ensembles. In Proceedings of the 31st International Conference on Neural Information Processing Systems (pp. 6405-6416).

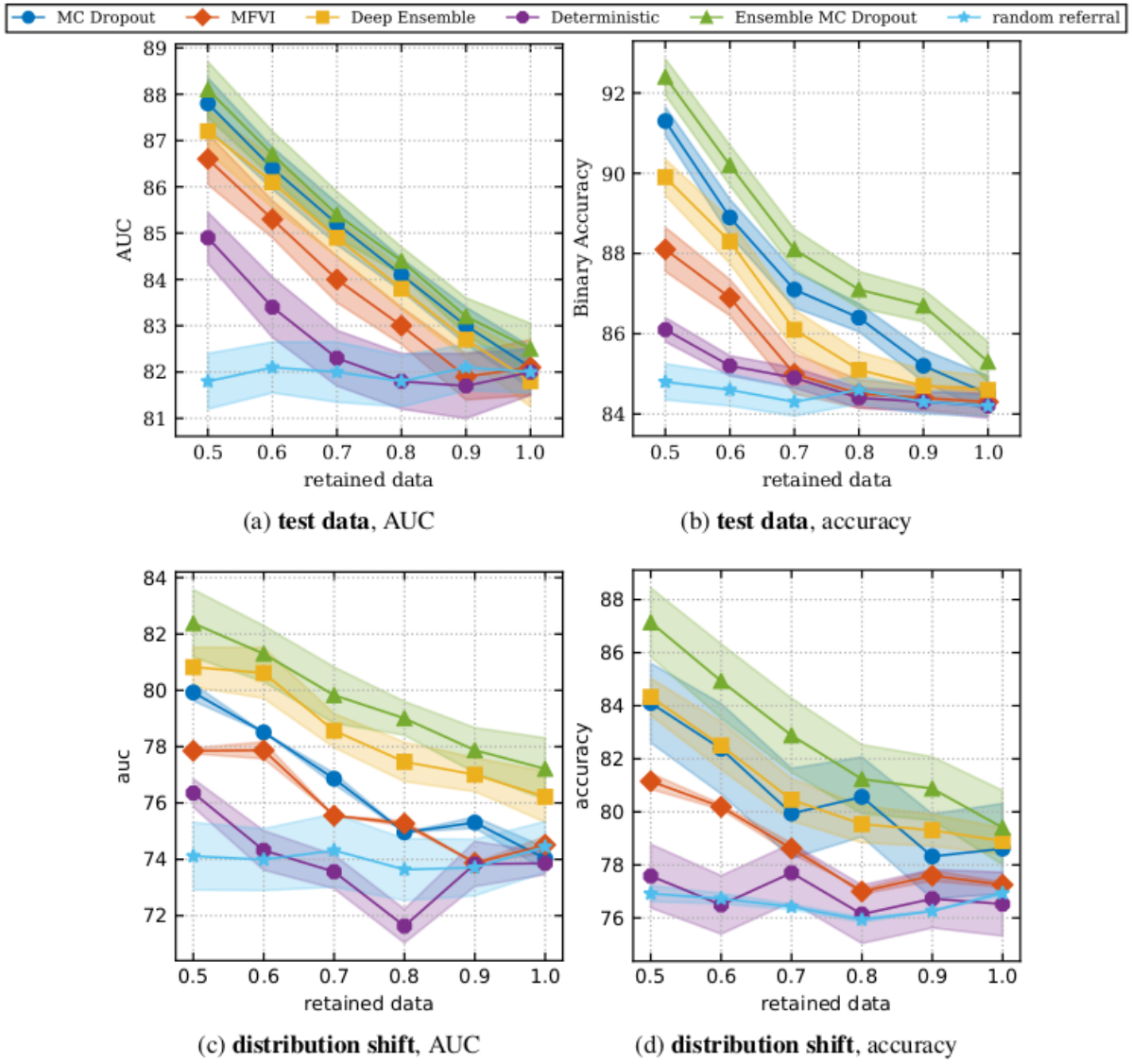
3.4 Deep Ensembles

Algorithm 1 Pseudocode of the training procedure for our method

1: ▷ *Let each neural network parametrize a distribution over the outputs, i.e.* $p_\theta(y|\mathbf{x})$. *Use a proper scoring rule as the training criterion* $\ell(\theta, \mathbf{x}, y)$. *Recommended default values are* $M = 5$ *and* $\epsilon = 1\%$ *of the input range of the corresponding dimension (e.g 2.55 if input range is [0,255]).*
2: Initialize $\theta_1, \theta_2, \ldots, \theta_M$ randomly
3: **for** $m = 1 : M$ **do** ▷ *train networks independently in parallel*
4: Sample data point n_m randomly for each net ▷ *single* n_m *for clarity, minibatch in practice*
5: Generate adversarial example using $\mathbf{x}'_{n_m} = \mathbf{x}_{n_m} + \epsilon \operatorname{sign}\left(\nabla_{\mathbf{x}_{n_m}} \ell(\theta_m, \mathbf{x}_{n_m}, y_{n_m})\right)$
6: Minimize $\ell(\theta_m, \mathbf{x}_{n_m}, y_{n_m}) + \ell(\theta_m, \mathbf{x}'_{n_m}, y_{n_m})$ w.r.t. θ_m ▷ *adversarial training (optional)*

Lakshminarayanan, B., Pritzel, A., & Blundell, C. (2017, December). Simple and scalable predictive uncertainty estimation using deep ensembles. In Proceedings of the 31st International Conference on Neural Information Processing Systems (pp. 6405-6416).

4. Bayesian DL methods compared



(a) **test data**, AUC

(b) **test data**, accuracy

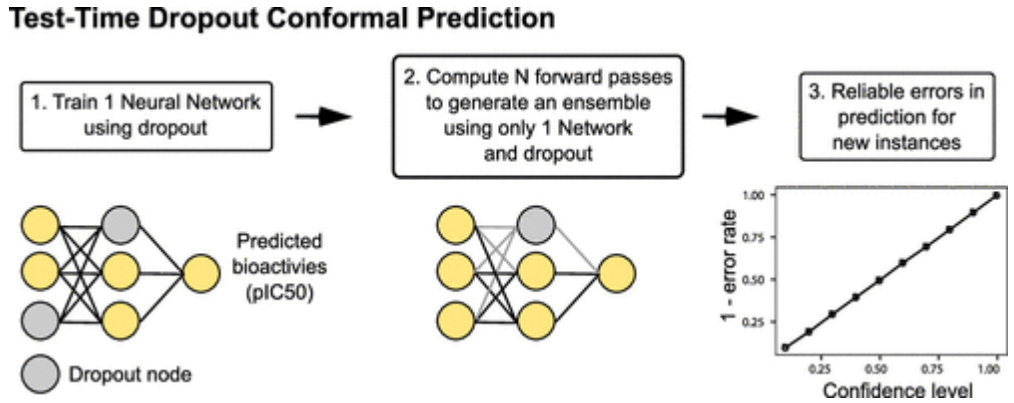
(c) **distribution shift**, AUC

(d) **distribution shift**, accuracy

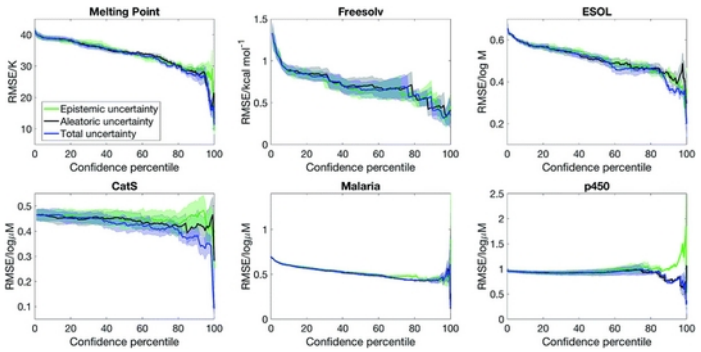
Filos, A., Farquhar, S., Gomez, A. N., Rudner, T. G., Kenton, Z., Smith, L., ... & Gal, Y. (2019). A systematic comparison of Bayesian deep learning robustness in diabetic retinopathy tasks. arXiv preprint arXiv:1912.10481.

4. Applications of Bayesian DL in drug discovery

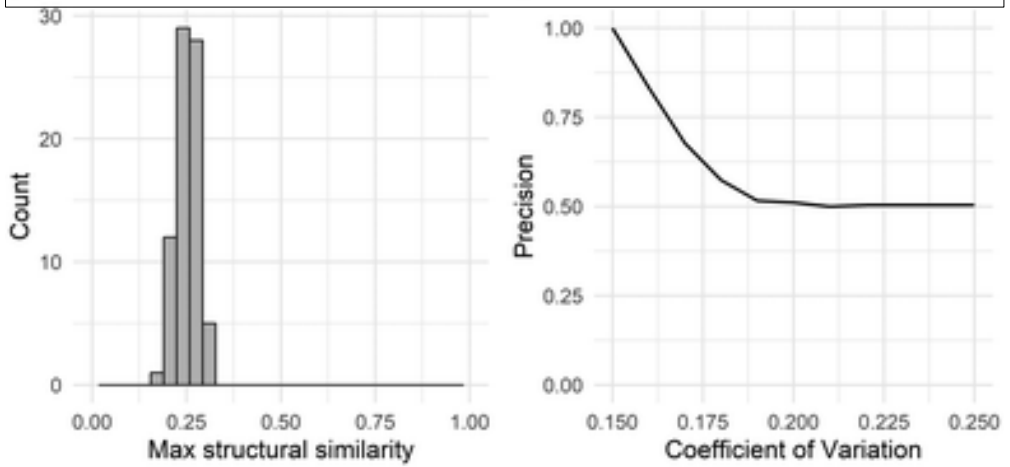
- Plenty of work on uncertainty estimation, calibration, etc.
- Bayesian DL in DD: several publications



Cortes-Ciriano, I., & Bender, A. (2019). Reliable prediction errors for deep neural networks using test-time dropout. Journal of chemical information and modeling, 59(7), 3330-3339



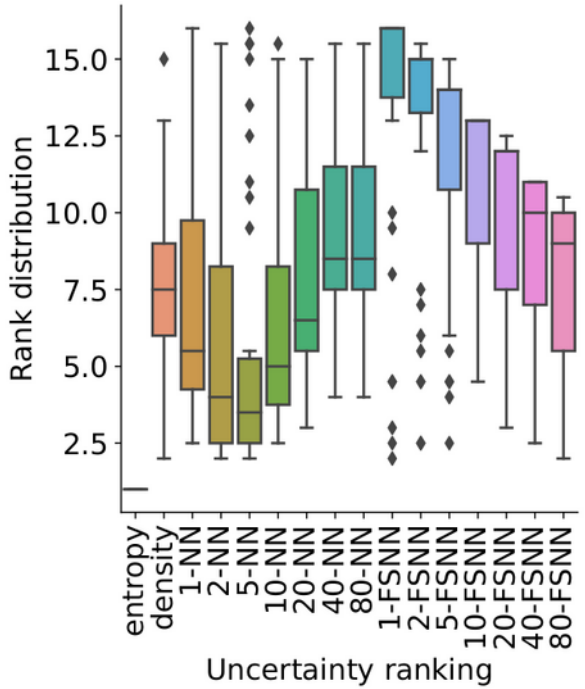
Zhang, Y. (2019). Bayesian semi-supervised learning for uncertainty-calibrated prediction of molecular properties and active learning. Chemical science, 10(35), 8154-8163.

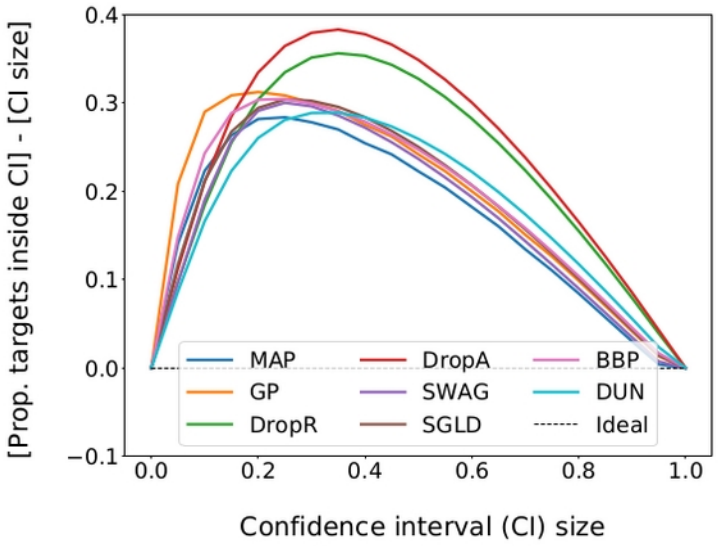


Fotis, C., Meimetis, N., Sardis, A., & Alexopoulos, L. G. (2021). DeepSIBA: chemical structure-based inference of biological alterations using deep learning. Molecular Omics, 17(1), 108-120.

JOHANNES KEPLER UNIVERSITY LINZ

4. Applications of Bayesian DL in drug discovery





Renz, P., Hochreiter, S., & Klambauer, G. (2019). Uncertainty Estimation Methods to Support Decision-Making in Early Phases of Drug Discovery. In Workshop on Safety and Robustness in Decision-making at 33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

Lamb, G., & Paige, B. (2020). Bayesian graph neural networks for molecular property prediction. arXiv preprint arXiv:2012.02089.

Summary

- We have investigated Bayesian Deep Learning approaches
- Discussed main difference to frequentist approaches
- Can capture uncertainty about parameters
- Usually approximation of parameter posterior
- Variational approach, Monte-Carlo Dropout, Deep Ensembles

5. Hessian-based approach

- Regression task: Gaussian noise on labels

$$p(y \mid \boldsymbol{x}, \boldsymbol{w}, \sigma_\epsilon^2) = \mathcal{N}(y \mid g(\boldsymbol{x}; \boldsymbol{w}), \sigma_\epsilon^2)$$

- Gaussian prior on weights

$$p(\boldsymbol{w} \mid \sigma_w^2) = \mathcal{N}(\boldsymbol{w} \mid \mathbf{0}, \sigma_w^2)$$

- Dataset $\mathcal{D}$ of data points and labels; posterior and log post.:

$$p(\boldsymbol{w} \mid \mathcal{D}, \sigma_w^2, \sigma_\epsilon^2) \propto p(\boldsymbol{w} \mid \sigma_w^2) p(\mathcal{D} \mid \boldsymbol{w}, \sigma_\epsilon^2)$$

$$\log p(\boldsymbol{w} \mid \mathcal{D}, \sigma_w^2, \sigma_\epsilon^2) = \frac{1}{2\sigma_w^2} \boldsymbol{w}^T \boldsymbol{w} - \frac{1}{2\sigma_\epsilon^2} \sum_{n=1}^{N} (g(\boldsymbol{x}; \boldsymbol{w}) - y_n)^2 + C$$

The last function is optimized with the usual gradient descent techniques and we obtain parameters: $\boldsymbol{w}_{\mathrm{MAP}}$

5. Hessian-based approach

- Having found $\boldsymbol{w}_{\mathrm{MAP}}$, we can make a local Gaussian approximation by evaluation the matrix of second derivatives of the negative log posterior:

$$\boldsymbol{A} = -\nabla^2 \log p(\boldsymbol{w} \mid \mathcal{D}, \sigma_w^2, \sigma_\epsilon^2) = 1/\sigma_w^2 \boldsymbol{I} + 1/\sigma_\epsilon^2 \boldsymbol{H}$$

  where $\boldsymbol{H}$ is the usual Hessian of the loss function of the neural network. Can be approximated in various ways for small networks (see later lecture)

- Gaussian approximation of posterior

$$q(\boldsymbol{w} \mid \mathcal{D}) = \mathcal{N}(\boldsymbol{w} \mid \boldsymbol{w}_{\mathrm{MAP}}, \boldsymbol{A}^{-1})$$

- Predictive distribution (still intractable, sampling from approx)

$$p(y \mid \boldsymbol{x}, \mathcal{D}) = \int \mathcal{N}(y \mid g(\boldsymbol{x}, \boldsymbol{w}), \sigma_\epsilon^2) q(\boldsymbol{w} \mid \mathcal{D}) d\boldsymbol{w}$$

Intractable because distribution on weights is inside the neural net

5. Hessian-based approach

- If we would have a linear model instead of $g(\boldsymbol{x}; \boldsymbol{w})$, this integral would be analytically tractable.
- Let us approximate the neural net around $\boldsymbol{w}_{\mathrm{MAP}}$ with a Taylor approximation up to first degree

$$g(\boldsymbol{x}; \boldsymbol{w}) \approx g(\boldsymbol{x}; \boldsymbol{w}_{\mathrm{MAP}}) + \boldsymbol{g}^T(\boldsymbol{w} - \boldsymbol{w}_{\mathrm{MAP}})$$

  where $\boldsymbol{g} = \nabla_{\boldsymbol{w}} g(\boldsymbol{x}; \boldsymbol{w})|_{\boldsymbol{w}_{\mathrm{MAP}}}$

- Linear Gaussian model with Gaussian distribution of weights and prior whose mean is linear function of weights

$$p(y \mid \boldsymbol{x}, \boldsymbol{w}, \sigma_\epsilon^2) \approx \mathcal{N}(y \mid g(\boldsymbol{x}, \boldsymbol{w}_{\mathrm{MAP}}) + \boldsymbol{g}^T(\boldsymbol{w} - \boldsymbol{w}_{\mathrm{MAP}}), \sigma_\epsilon^2)$$

  General results on marginal Gaussian provide:

$$p(y \mid \boldsymbol{x}, \mathcal{D}, \sigma_\epsilon^2, \sigma_w^2) \approx \mathcal{N}(y \mid g(\boldsymbol{x}, \boldsymbol{w}_{\mathrm{MAP}}), \sigma_\epsilon^2 + \boldsymbol{g}^T \boldsymbol{A} \boldsymbol{g})$$