Foundations of Deep Learning, Winter Term 2021/22

Week 13: Uncertainty in Deep Learning

Uncertainty in Deep Learning

Frank Hutter Abhinav Valada

University of Freiburg

Overview of Week 13

Lecture Overview

Foundations of Deep Learning, Winter Term 2021/22

Week 13: Uncertainty in Deep Learning

Why we Need Uncertainty in Neural Networks

Frank Hutter Abhinav Valada

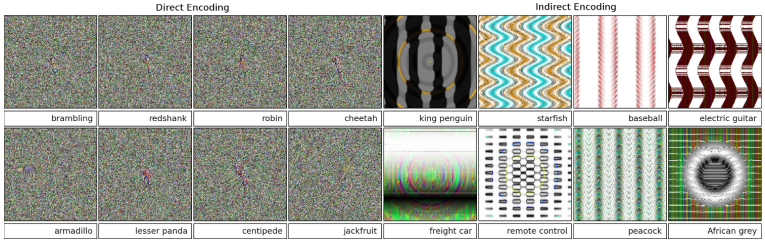
University of Freiburg

Motivation: Why Does Uncertainty Matter?

- Knowing model's uncertainty is crucial in safety-critical applications
  - Medical diagnosis
  - Autonomous driving
  - Operations in space
  - Household robotics
  - ...
- Decision theory = probability theory + utility theory

Uncertainty in Neural Networks for Classification

- Standard neural nets *do* output a probability distribution (softmax)
  - But this probability distribution is often not reliable
- Uncertainty should grow away from the typical data seen, but doesn't
  - E.g., for ReLU networks there are points far away from the training data for which the networks are arbitrarily certain [Hein et al., 2019]
  - E.g., deep neural nets can be overconfident for unrecognizable images:

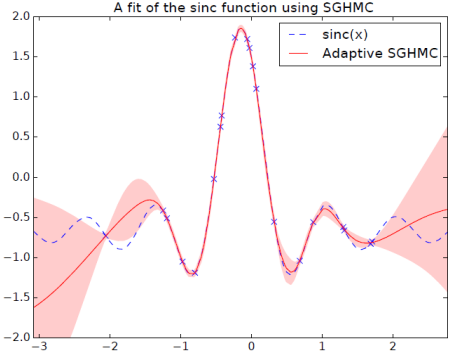


Evolved images that are classified with $\geq 99.6\%$ certainty to be a familiar object

[Image source: Nguyen et al., 2015]

Uncertainty in Neural Networks for Regression

- Standard NNs for regression only output a single value, no uncertainty
- Uncertainty should grow away from the typical data seen



[Image source: Springenberg et al., 2016]

Types of Uncertainty

- Aleatoric uncertainty
  - Intrincic observation noise; cannot be reduced
- Epistemic uncertainty
  - Uncertainty about the model
  - Relates to our knowledge of the world; can be reduced by more data
- We can express uncertainty about different quantities
  - Uncertainty about the true state of the world (epistemic)
  - Uncertainty about what we will measure (epistemic + aleatoric)

Questions to Answer for Yourself / Discuss with Friends

- Repetition:
  In the case of classification, our standard softmax is already a probabilistic prediction. Why do we need more?
- Discussion:
  Can you think of an application area of neural networks where it is really important to quantify our uncertainty and "know when we don't know"?

Lecture Overview

Foundations of Deep Learning, Winter Term 2021/22

Week 13: Uncertainty in Deep Learning

Background: Properties of the Gaussian Distribution

Frank Hutter Abhinav Valada

University of Freiburg

The Gaussian Distribution

The univariate form

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}\right)$$

Named after Carl Friedrich Gauss (1777-1855)

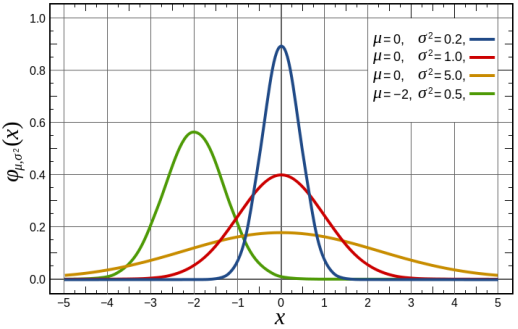
(Exposition follows Philipp Hennig's Gaussian process tutorial)

[Image source: Wikimedia Commons]

The Gaussian Distribution

The univariate form

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}\right)$$



[Image source: Wikimedia Commons]

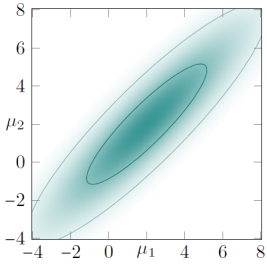
The Gaussian Distribution

The multivariate Gaussian (in N dimensions)

$$f(\boldsymbol{x}) = \frac{1}{(2\pi)^{N/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^{\mathsf{T}}\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right)$$

- $\boldsymbol{x}$ and $\boldsymbol{\mu}$ are $N \times 1$ (column) vectors
- $\boldsymbol{\Sigma}$ is an $N \times N$ matrix

An example in 2 dimensions:



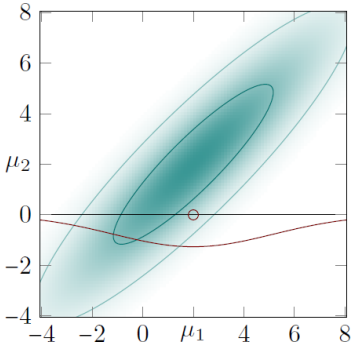
$$\boldsymbol{\mu} = \begin{pmatrix}\mu_1\\ \mu_2\end{pmatrix} \qquad \boldsymbol{\Sigma} = \begin{pmatrix}\sigma_{11}^2 & \sigma_{12}^2\\ \sigma_{21}^2 & \sigma_{22}^2\end{pmatrix}$$

Property 1: Closure Under Marginalization

Gaussian distributions are closed under marginalization

Let $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ be jointly Gaussian distributed:

$$\begin{bmatrix}\boldsymbol{x}_1\\ \boldsymbol{x}_2\end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix}\boldsymbol{\mu}_1\\ \boldsymbol{\mu}_2\end{bmatrix}, \begin{bmatrix}\boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12}\\ \boldsymbol{\Sigma}_{12}^{\mathsf{T}} & \boldsymbol{\Sigma}_{22}\end{bmatrix}\right) \qquad \text{Then } \boldsymbol{x}_1 \sim \mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_{11}).$$



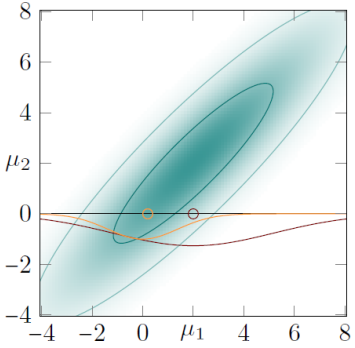
$$\boldsymbol{\mu} = \begin{pmatrix}\mu_1\\ \mu_2\end{pmatrix} \qquad \boldsymbol{\Sigma} = \begin{pmatrix}\sigma_{11}^2 & \sigma_{12}^2\\ \sigma_{21}^2 & \sigma_{22}^2\end{pmatrix}$$

Property 2: Closure Under Conditioning

Gaussian distributions are closed under conditioning

Let $\boldsymbol{a}$ and $\boldsymbol{b}$ be jointly Gaussian distributed: $\begin{bmatrix} \boldsymbol{a} \\ \boldsymbol{b} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}_a \\ \boldsymbol{\mu}_b \end{bmatrix}, \begin{bmatrix} A & C \\ C^\intercal & B \end{bmatrix} \right)$

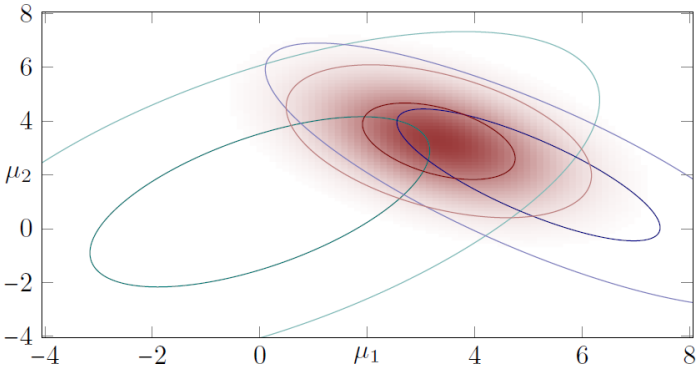
Then $\boldsymbol{b}|\boldsymbol{a} \sim \mathcal{N}\left(\boldsymbol{\mu}_b + C^\intercal A^{-1}(\boldsymbol{a} - \boldsymbol{\mu}_a), B - C^\intercal A^{-1} C\right)$.



Property 3: Closure Under Multiplication

$$\mathcal{N}(x; a, A)\mathcal{N}(x; b, B) \propto \mathcal{N}(x; c, C)$$

$$C := (A^{-1} + B^{-1})^{-1} \quad c := C(A^{-1}a + B^{-1}b)$$



Property 4: Closure Under Linear Maps

Gaussian distributions are closed under linear maps

- Let $\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w)$. Then, $\boldsymbol{x}^\mathsf{T}\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{x}^\mathsf{T}\boldsymbol{\mu}_w, \boldsymbol{x}^\mathsf{T}\boldsymbol{\Sigma}_w\boldsymbol{x})$

Why Gaussians?

- The central limit theorem tells us that means of large populations are distributed according to a Gaussian
- However, individual measurements are rarely Gaussian-distributed
  - So, this isn't really the main reason we use Gaussians
- The use of Gaussians is mostly a mathematical convenience that lets us solve integrals in closed form!

Questions to Answer for Yourself / Discuss with Friends

- Repetition: Write down the formula for a multivariate Gaussian.
- Repetition: List four operations under which Gaussians are closed.

Lecture Overview

Foundations of Deep Learning, Winter Term 2021/22

Week 13: Uncertainty in Deep Learning

Bayesian Linear Regression

Frank Hutter Abhinav Valada

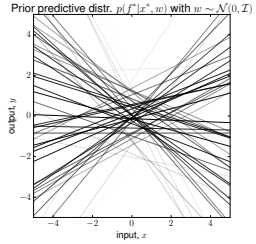
University of Freiburg

Principles of Bayesian Reasoning

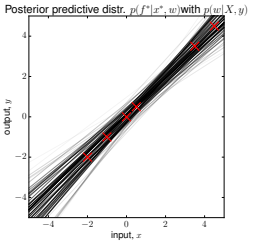
Don't select a single model, but acknowledge all sources of uncertainty

- Prior over possible models
- Likelihood of the data under these models, making certain models more likely
- Posterior over possible models

$$\text{posterior} = \frac{\text{likelihood} \times \text{prior}}{\text{marginal likelihood}} \propto \text{likelihood} \times \text{prior}$$

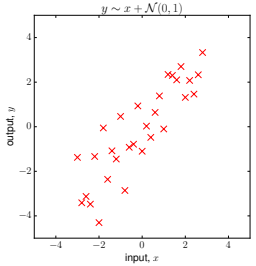


Prior over models (here: lines)



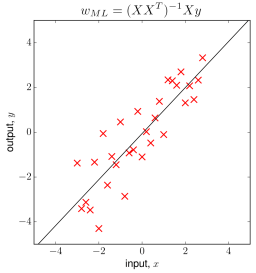
Posterior over models

Linear Regression



- We have a dataset of n observations, $\mathcal{D} = \{(\boldsymbol{x}_i, y_i) | i = 1, \ldots, n\}$
- Each $\boldsymbol{x}_i$ denotes a $D \times 1$ column vector and each y_i is a scalar output or target
- We collect all inputs $\boldsymbol{x}_i$ into the $D \times n$ design matrix $\boldsymbol{X} = [\boldsymbol{x}_1 \cdots \boldsymbol{x}_n]$ and the targets into the $n \times 1$ vector $\boldsymbol{y}$.
  - Note: here, we use the notation from the excellent book Gaussian processes for Machine Learning by Rasmussen & Williams so that you can read up on all derivations. (Note that $\boldsymbol{X}$ is transposed from our usual notation.)

Standard Linear Regression Model



$$f(\boldsymbol{x}) = \boldsymbol{x}^{\mathsf{T}}\boldsymbol{w}, \qquad y = f(\boldsymbol{x}) + \epsilon \tag{2.1}$$

$$\epsilon \sim \mathcal{N}(0, \sigma_n^2) \tag{2.2}$$

(all equation numbers from Chapter 2 of
Gaussian processes for Machine Learning by Rasmussen & Williams)

Maximum Likelihood Estimation of $\boldsymbol{w}$

The likelihood $p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{w})$ is then defined as:

$$\begin{aligned} p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{w}) &= \prod_{i=1}^{n} p(y_i|\boldsymbol{x}_i, \boldsymbol{w}) \\ &= \prod_{i=1}^{n} \frac{1}{(\sqrt{2\pi}\sigma_n)} \exp\left(-\frac{(y_i - \boldsymbol{x}_i^\mathsf{T}\boldsymbol{w}_i)^2}{2\sigma_n^2}\right) \\ &= \frac{1}{(2\pi\sigma_n^2)^{n/2}} \exp\left(-\frac{1}{2\sigma_n^2}|\boldsymbol{y} - \boldsymbol{X}^\mathsf{T}\boldsymbol{w}|^2\right) \\ &= \mathcal{N}(\boldsymbol{y}; \boldsymbol{X}^\mathsf{T}\boldsymbol{w}, \sigma_n^2 I) \end{aligned} \tag{2.3}$$

Maximum likelihood estimation seeks to maximize this likelihood:

$$\Rightarrow \boldsymbol{w}_{\mathsf{ML}} = (\boldsymbol{X}\boldsymbol{X}^\mathsf{T})^{-1}\boldsymbol{X}\boldsymbol{y}$$

This is known as the normal equations.

The Bayesian Way of Setting $\boldsymbol{w}$

- Don't commit to a single $\boldsymbol{w}_{\mathsf{ML}}$,
  but rather integrate over an infinite number of possibilities

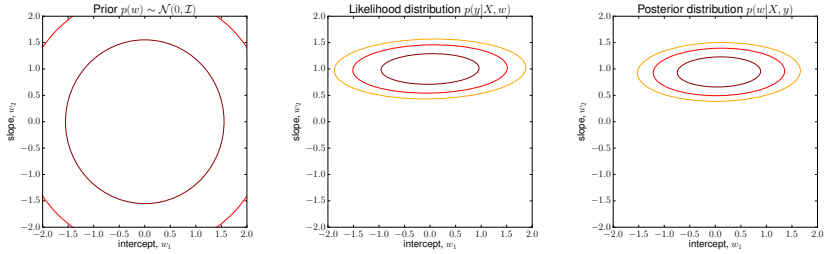
- Define a prior distribution $p(\boldsymbol{w})$ over $\boldsymbol{w}$ quantifying our expectations
- Compute the likelihood $p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{w})$ of the data under each model

- Compute posterior $p(\boldsymbol{w}|\boldsymbol{y}, \boldsymbol{X})$ by Bayes' rule: proportional to likelihood times prior.

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

$$p(\boldsymbol{w}|\boldsymbol{y}, \boldsymbol{X}) = \frac{p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{w})p(\boldsymbol{w})}{p(\boldsymbol{y}|\boldsymbol{X})} \tag{2.5}$$

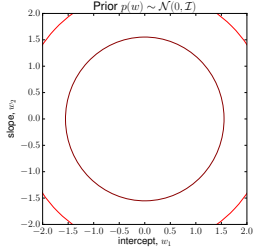
The Bayesian Way of Setting $\boldsymbol{w}$

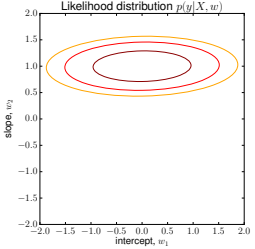


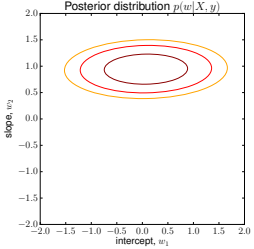
$$\text{posterior} = \frac{\text{likelihood} \times \text{prior}}{\text{marginal likelihood}}$$

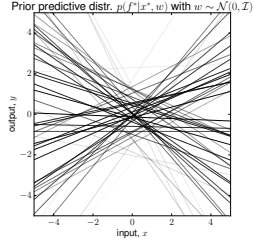
$$p(\boldsymbol{w}|\boldsymbol{y}, \boldsymbol{X}) = \frac{p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{w})p(\boldsymbol{w})}{p(\boldsymbol{y}|\boldsymbol{X})} \tag{2.5}$$

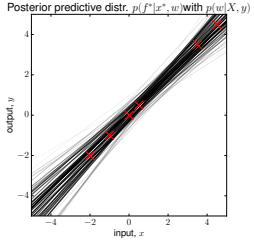
Weights Correspond to Functions











Computation of the Posterior Under a Gaussian Prior

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

$$p(\boldsymbol{w}|\boldsymbol{y}, \boldsymbol{X}) = \frac{p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{w})p(\boldsymbol{w})}{p(\boldsymbol{y}|\boldsymbol{X})} \tag{2.5}$$

The denominator generally requires approximating a complicated integral $p(\boldsymbol{y}|\boldsymbol{X}) = \int p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{w})p(\boldsymbol{w})d\boldsymbol{w}$. But everything simplifies if we choose a prior that is conjugate to the likelihood; in this case another Gaussian:

$$\boldsymbol{w} \sim \mathcal{N}(0, \Sigma_p) \tag{2.4}$$

In that case, the posterior is Gaussian again:

$$p(\boldsymbol{w}|\boldsymbol{X}, \boldsymbol{y}) \sim \mathcal{N}\left(\bar{\boldsymbol{w}} = \frac{1}{\sigma_n^2} A^{-1} \boldsymbol{X}\boldsymbol{y}, A^{-1}\right), \tag{2.7}$$

with A given by:

$$\mathsf{A} = \sigma_n^{-2} \boldsymbol{X}\boldsymbol{X}^\mathsf{T} + \Sigma_p^{-1}$$

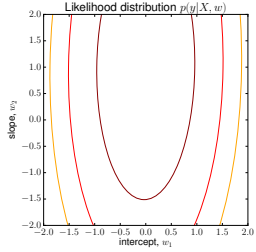
The Posterior Predictive Distribution

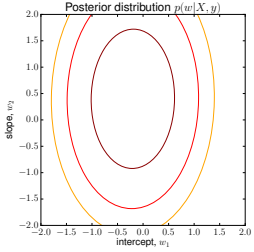
- Each weight vector $\boldsymbol{w}$ corresponds to a linear model
- For a new data point $\boldsymbol{x}_\star$, each weight vector $\boldsymbol{w}$ gives rise to a linear model prediction $f_\star = \boldsymbol{x}_\star^\mathsf{T}\boldsymbol{w}$
- We simply average the predictions of linear models with all possible $\boldsymbol{w}$, weighting each model by the posterior probability of its corresponding $\boldsymbol{w}$:

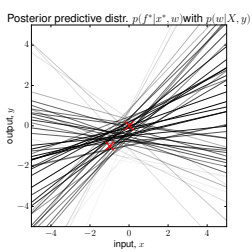
$$\begin{aligned} p(f_\star|\boldsymbol{x}_\star, \boldsymbol{X}, \boldsymbol{y}) &= \int p(f_\star|\boldsymbol{x}_\star, \boldsymbol{w}) p(\boldsymbol{w}|\boldsymbol{X}, \boldsymbol{y}) d\boldsymbol{w} \\ &= \mathcal{N}(f_\star; \frac{1}{\sigma_n^2}\boldsymbol{x}_\star^\mathsf{T} A^{-1}\boldsymbol{X}\boldsymbol{y}, \boldsymbol{x}_\star^\mathsf{T} A^{-1}\boldsymbol{x}_\star) \end{aligned} \tag{2.9}$$

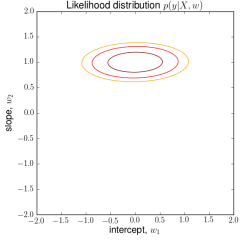
$\rightarrow$ This is a Gaussian because Gaussians are closed under linear maps and marginalization.

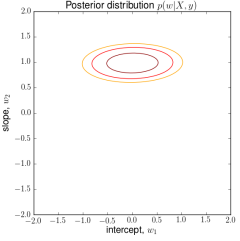
Enough Data Overwhelms the Prior

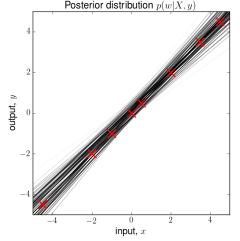












Bayesian Linear Regression With Basis Functions

The same analysis goes through if we replace $\boldsymbol{x}$ with some so-called basis functions $\phi(\boldsymbol{x})$ of $\boldsymbol{x}$, e.g., powers of x

$$\phi(x) = (1, x, x^2, x^3, \ldots)^\mathsf{T}$$

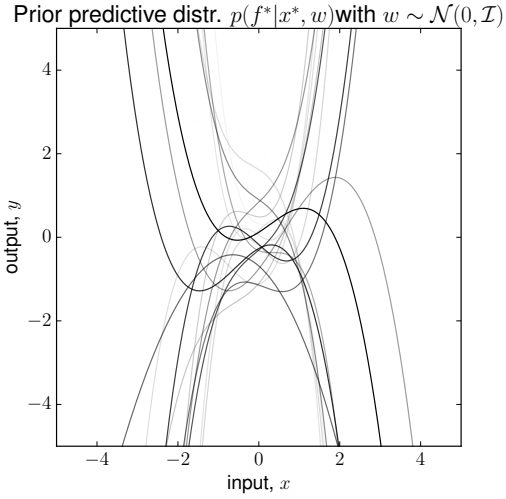
Basis function linear regression is then:

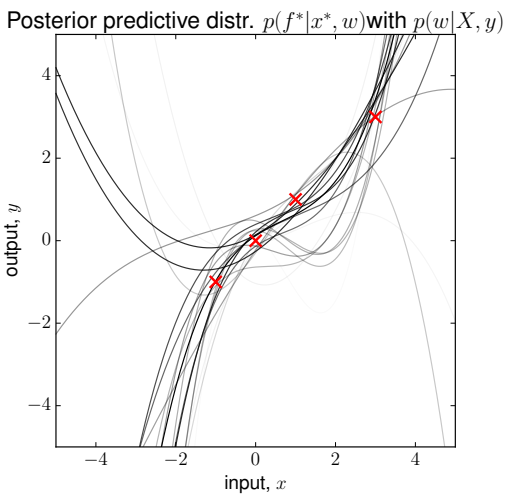
$$f(\boldsymbol{x}) = \phi(\boldsymbol{x})^\mathsf{T}\boldsymbol{w} \tag{2.10}$$

And the equivalent of 2.9 is:

$$f_\star|\boldsymbol{x}_\star, \boldsymbol{X}, \boldsymbol{y} \sim \mathcal{N}\left(\frac{1}{\sigma_n^2}\phi(\boldsymbol{x}_\star)^\mathsf{T}A^{-1}\Phi\boldsymbol{y}, \phi(\boldsymbol{x}_\star)^\mathsf{T}A^{-1}\phi(\boldsymbol{x}_\star)\right) \tag{2.11}$$
$$\text{with } \Phi = \phi(\boldsymbol{X}) \text{ and } A = \sigma_n^{-2}\Phi\Phi^\mathsf{T} + \Sigma_p^{-1}$$

Bayesian Cubic Regression





. . . and you can do the same based on learned features from the last layer of a neural network

Summary of Bayesian Linear Regression

- General Bayesian principle: we hedge our bets, integrate over infinitely many models
- In the special case of Bayesian linear regression, we can do this in closed form:
  - We use a Gaussian prior $p(\boldsymbol{w})$ that quantifies our expectations
  - We use a Gaussian likelihood $p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{w})$
  - This results in a Gaussian posterior $p(\boldsymbol{w}|\boldsymbol{X}, \boldsymbol{y})$
  - This also results in a Gaussian predictive distribution $p(f_\star|\boldsymbol{x}_\star, \boldsymbol{X}, \boldsymbol{y})$
- Bayesian linear regression still works with basis functions $\phi(\boldsymbol{x})$ instead of $\boldsymbol{x}$ itself
  - These basis functions can also be nonlinear
  - The computation just needs to be linear in $\boldsymbol{w}$

Questions to Answer for Yourself / Discuss with Friends

- Repetition:
  Write down the formula for the posterior $p(\boldsymbol{w}|\boldsymbol{X}, \boldsymbol{y})$ in terms of the prior $p(\boldsymbol{w})$ and the likelihood $p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{w})$
- Repetition:
  In Bayesian cubic regression, which components dominate if all weights have a uniform weight in the prior?

Lecture Overview

Foundations of Deep Learning, Winter Term 2021/22

Week 13: Uncertainty in Deep Learning

Being Bayesian about Neural Networks

Frank Hutter Abhinav Valada

University of Freiburg

What Does it Mean to be Bayesian About Neural Nets?

- Dealing with all sources of parameter uncertainty:
  - More than one weight vector can explain the observed data
  - Take into account all possible explanations

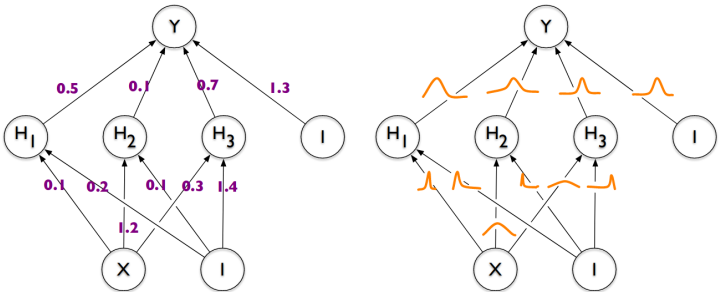


Figure: *Left*: Standard NN with parameter point estimates. *Right*: Bayesian NN with parameter distributions.

[Image source: Blundell et al., 2015]

The DNGO Model: Being Bayesian At Least About the Output Layer

1. Fit a standard neural network to the data
2. Use the representation in the last hidden layer as basis functions $\phi(\boldsymbol{x})$ of the input $\boldsymbol{x}$
3. Use Bayesian linear regression for the output layer

$\rightsquigarrow$ Successfully used as a probabilistic model for Bayesian optimization [Snoek et al., 2015]

Frequentist Approach: MLE and MAP Estimates

- Data: $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N = (\boldsymbol{X}, \boldsymbol{y})$
- Parameters $\boldsymbol{\theta}$: (all) weights of the neural network
- Prior: $p(\boldsymbol{\theta})$
- Likelihood: $p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta})$
- Posterior: $p(\boldsymbol{\theta}|\mathcal{D}) \equiv p(\boldsymbol{\theta}|\boldsymbol{X}, \boldsymbol{y})$
  - MAP estimation:

$$\begin{aligned}\boldsymbol{\theta}^{MAP} &= \arg\max_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}|\boldsymbol{X}, \boldsymbol{y}) \\ &= \arg\max_{\boldsymbol{\theta}} \log p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta}) + \log p(\boldsymbol{\theta})\end{aligned}$$

    → if prior $p(\boldsymbol{\theta})$ is Gaussian (Laplacian), this is L_2 (L_1) regularization
  - Maximum Likelihood Estimation (MLE):

$$\boldsymbol{\theta}^{MLE} = \arg\max_{\boldsymbol{\theta}} \log p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta})$$

    → corresponds to a uniform prior $p(\boldsymbol{\theta})$

Bayesian Neural Networks

- Data: $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N = (\boldsymbol{X}, \boldsymbol{y})$
- Parameters $\boldsymbol{\theta}$: (all) weights of the neural network
- Prior: $p(\boldsymbol{\theta})$
- Likelihood: $p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta})$
- Posterior: $p(\boldsymbol{\theta}|\mathcal{D}) \equiv p(\boldsymbol{\theta}|\boldsymbol{X}, \boldsymbol{y}) = \frac{p(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\boldsymbol{y}|\boldsymbol{X})} = \frac{p(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{\theta})p(\boldsymbol{\theta})}{\int p(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{\theta})p(\boldsymbol{\theta})d\boldsymbol{\theta}}$
  - Problem: $p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta})$ is not as simple as in linear models
    - E.g., its mean is not a linear function of $\boldsymbol{\theta}$
  - The posterior is therefore not a Gaussian anymore
    - One possibility: approximate $p(\boldsymbol{\theta}|\mathcal{D})$ with a simple distribution (e.g., a Gaussian again) $\rightarrow$ variational inference
    - Another possibility: sample from $p(\boldsymbol{\theta}|\mathcal{D})$ $\rightarrow$ MCMC
- Prediction (inference): $p(y'|\mathcal{D}, \boldsymbol{x}') = \int p(\boldsymbol{y}'|\boldsymbol{x}', \boldsymbol{\theta})p(\boldsymbol{\theta}|\mathcal{D})d\boldsymbol{\theta}$
  - Equivalent to averaging predictions from an ensemble of an infinite number of NNs (weighted by the posterior probabilities of $\boldsymbol{\theta}$)

Questions to Answer for Yourself / Discuss with Friends

- Repetition:
  How can you use Bayesian linear regression to at least be Bayesian about the output layer of a neural network for regression?
- Repetition:
  Why can't we use Bayesian linear regression to be Bayesian about all the parameters $\boldsymbol{\theta}$ of a neural network?

Lecture Overview

Foundations of Deep Learning, Winter Term 2021/22

Week 13: Uncertainty in Deep Learning

Variational Inference

Frank Hutter Abhinav Valada

University of Freiburg

Variational Inference

- Goal: approx. an intractable posterior distribution $p(\boldsymbol{\theta}|\mathcal{D})$ by a simpler distribution $q_\phi(\boldsymbol{\theta})$
- Approach: pick a model family $q_\phi(\boldsymbol{\theta})$ and optimize its parameters ϕ to make $q_\phi(\boldsymbol{\theta})$ as close as possible to $p(\boldsymbol{\theta}|\mathcal{D})$
  - Unfortunately, $p(\boldsymbol{\theta}|\mathcal{D})$ is intractable, so we also can't compute $KL(q_\phi(\boldsymbol{\theta})||p(\boldsymbol{\theta}|\mathcal{D}))$
  - But we can implicitly minimize this KL divergence by rewriting it:

$$
\begin{aligned}
KL(q_\phi(\boldsymbol{\theta})||p(\boldsymbol{\theta}|\mathcal{D})) &= \int q_\phi(\boldsymbol{\theta}) \log \frac{q_\phi(\boldsymbol{\theta})}{p(\boldsymbol{\theta}|\mathcal{D})} d\boldsymbol{\theta} \\
&= \mathbb{E}_{q_\phi(\boldsymbol{\theta})}\left[\log \frac{q_\phi(\boldsymbol{\theta})}{p(\boldsymbol{\theta}|\mathcal{D})}\right] = \mathbb{E}_{q_\phi(\boldsymbol{\theta})}\left[\log \frac{q_\phi(\boldsymbol{\theta})}{p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})} p(\mathcal{D})\right] \\
&= \mathbb{E}_{q_\phi(\boldsymbol{\theta})}\left[\log q_\phi(\boldsymbol{\theta}) - \log p(\mathcal{D}|\boldsymbol{\theta}) - \log p(\boldsymbol{\theta}) + \log p(\mathcal{D})\right] \\
&= \mathbb{E}_{q_\phi(\boldsymbol{\theta})}\left[\log q_\phi(\boldsymbol{\theta}) - \log p(\mathcal{D}|\boldsymbol{\theta}) - \log p(\boldsymbol{\theta})\right] + \log p(\mathcal{D}) \\
&= \mathbb{E}_{q_\phi(\boldsymbol{\theta})}\left[\log q_\phi(\boldsymbol{\theta}) - \log p(\mathcal{D}|\boldsymbol{\theta}) - \log p(\boldsymbol{\theta})\right] + \log p(\mathcal{D}) \\
&= KL(q_\phi(\boldsymbol{\theta})||p(\boldsymbol{\theta})) - \mathbb{E}_{q_\phi(\boldsymbol{\theta})}\left[\log p(\mathcal{D}|\boldsymbol{\theta})\right] + \log p(\mathcal{D})
\end{aligned}
$$

Evidence Lower Bound (ELBO)

$$KL(q_\phi(\boldsymbol{\theta})||p(\boldsymbol{\theta}|\mathcal{D})) = KL(q_\phi(\boldsymbol{\theta})||p(\boldsymbol{\theta})) - \mathbb{E}_{q_\phi(\boldsymbol{\theta})}\left[\log p(\mathcal{D}|\boldsymbol{\theta})\right] + \log p(\mathcal{D})$$

$$\Rightarrow$$

$$KL(q_\phi(\boldsymbol{\theta})||p(\boldsymbol{\theta}|\mathcal{D})) + \underbrace{\mathbb{E}_{q_\phi(\boldsymbol{\theta})}\left[\log p(\mathcal{D}|\boldsymbol{\theta})\right] - KL(q_\phi(\boldsymbol{\theta})||p(\boldsymbol{\theta}))}_{\text{evidence lower bound (ELBO)}} = \log p(\mathcal{D})$$

- In order to minimize $KL(q_\phi(\boldsymbol{\theta})||p(\boldsymbol{\theta}|\mathcal{D}))$ w.r.t. $\boldsymbol{\theta}$, we only need to maximize ELBO since $p(\mathcal{D})$ is a constant (it does not depend on θ)
- ELBO embodies a trade-off between satisfying the complexity of the data and satisfying the simplicity of the prior
  - $\mathbb{E}_{q_\phi(\boldsymbol{\theta})}\left[\log p(\mathcal{D}|\boldsymbol{\theta})\right] \rightarrow$ data likelihood: data dependent (max!)
  - $KL(q_\phi(\boldsymbol{\theta})||p(\boldsymbol{\theta})) \rightarrow$ difference from true prior: prior dependent (min!)
- Since $KL(q_\phi(\boldsymbol{\theta})||p(\boldsymbol{\theta}|\mathcal{D})) \geq 0$, ELBO is a lower bound on $\log p(\mathcal{D})$
- For an appropriate choice of q, ELBO is tractable to compute.

Variational inference is a very active field of research

- ELBO is very popular
  - Approximates the true posterior with the variational posterior and therefore replaces the difficult integral
  - Wide range of applications
- Summary of ongoing research efforts on variational inference
  - ELBO requires calculations over the whole dataset
    - Circumvented by Monte Carlo estimates (Stochastic VI)
    - Methods for reducing variance in the gradients
  - The choice of $q_\phi(\boldsymbol{\theta})$ is important; need to trade off:
    + Expressive power
    + Computational feasibility: memory cost and speed

Questions to Answer for Yourself / Discuss with Friends

- Repetition: Why does it make sense to replace computing one KL divergence $KL(q_\phi(\boldsymbol{\theta})||p(\boldsymbol{\theta}|\mathcal{D}))$ with a computation that involves another KL-divergence $KL(q_\phi(\boldsymbol{\theta})||p(\boldsymbol{\theta}))$?
- Repetition: Which two terms does ELBO consist of?
- Repetition: Derive the fact that ELBO is a lower bound to $\log p(\mathcal{D})$.

Lecture Overview

Foundations of Deep Learning, Winter Term 2021/22

Week 13: Uncertainty in Deep Learning

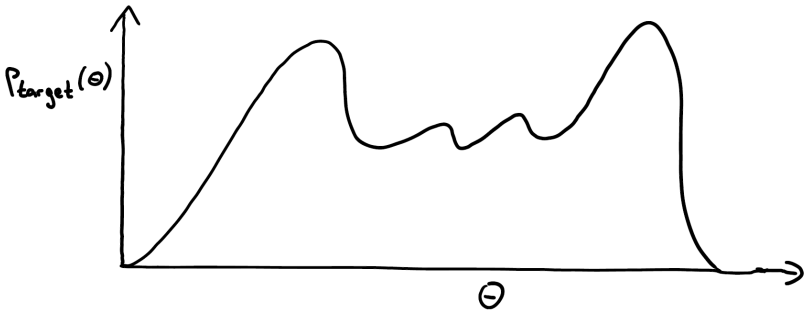
Markov Chain Monte Carlo

Frank Hutter Abhinav Valada

University of Freiburg

Markov Chain Monte Carlo (MCMC) Sampling: Intuition

- Problem: given a probability distribution $p_{target}(\boldsymbol{\theta})$ that we can evaluate up to a normalization constant, draw samples according to $p_{target}(\boldsymbol{\theta})$
  - In our case $p_{target}(\boldsymbol{\theta}) = p(\boldsymbol{\theta}|\mathcal{D})$



- Move along a chain according to a proposal distribution
- Accept improving moves
- Only accept worsening moves with a certain probability
- Keep a history (the "chain") of visited states

Markov Chain Monte Carlo (MCMC) Sampling: Key Idea

- Key idea: a chain that samples $\boldsymbol{\theta}^{(t)} \rightarrow \boldsymbol{\theta}^{(t+1)} \rightarrow \boldsymbol{\theta}^{(t+2)} \rightarrow \ldots$
  - such that the samples converge to the target distribution $p_{target}(\boldsymbol{\theta})$

- We use every t-th sample $\boldsymbol{\theta}^{(k+it)}$ to approximate samples from $p_{target}(\boldsymbol{\theta})$
  - E.g., with $k = 50$ and $t = 10$, our sample sequence is $\boldsymbol{\theta}^{(50)}$, $\boldsymbol{\theta}^{(60)}$, $\boldsymbol{\theta}^{(70)}$, $\boldsymbol{\theta}^{(80)}$, $\ldots$
  - A histogram of this sequence then approximates $p_{target}(\boldsymbol{\theta})$
  - The sequence can also be used to approximate expectations of a function $f(\boldsymbol{\theta})$ under $p_{target}(\boldsymbol{\theta})$:

$$\mathbb{E}_{p_{target}}(f) \approx \frac{1}{M} \sum_{i=1}^{M} f(\boldsymbol{\theta}^{(k+it)})$$

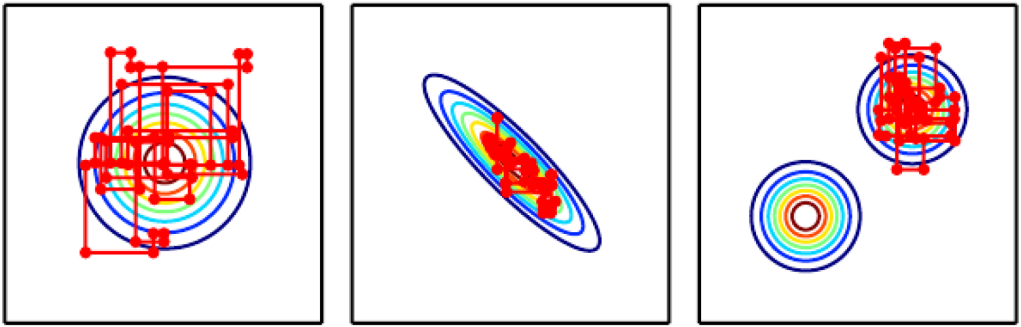
Markov Chain Monte Carlo (MCMC) Sampling: Theory

- Ingredients
  - A state space $\boldsymbol{\Theta}$
  - A transition distribution $p(\boldsymbol{\theta}_2|\boldsymbol{\theta}_1)$ for $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2 \in \boldsymbol{\Theta}$
    - For all $\boldsymbol{\theta}_1 \in \boldsymbol{\Theta}$: $\sum_{\boldsymbol{\theta}_2 \in \boldsymbol{\Theta}} p(\boldsymbol{\theta}_2|\boldsymbol{\theta}_1) = 1$
    - For $\boldsymbol{\theta}_2 \neq \boldsymbol{\theta}_1$, $p(\boldsymbol{\theta}_2|\boldsymbol{\theta}_1) = g(\boldsymbol{\theta}_2|\boldsymbol{\theta}_1)A(\boldsymbol{\theta}_2, \boldsymbol{\theta}_1)$
- We want the chain to sample from a stationary distribution π
  - For all $\boldsymbol{\theta}_2 \in \boldsymbol{\Theta}$: $\sum_{\boldsymbol{\theta}_1 \in \boldsymbol{\Theta}} \pi(\boldsymbol{\theta}_1) p(\boldsymbol{\theta}_2|\boldsymbol{\theta}_1) = \pi(\boldsymbol{\theta}_2)$
  - This stationary distribution $\pi(\boldsymbol{\theta})$ should be $\pi(\boldsymbol{\theta}) \equiv p_{target}(\boldsymbol{\theta})$
- Sufficient condition for a stationary distribution to exist: detailed balance
  - For any $\boldsymbol{\theta}, \boldsymbol{\theta}' \in \boldsymbol{\Theta}$: probability for being in $\boldsymbol{\theta}$ and transitioning to $\boldsymbol{\theta}'$ must be same as for being in $\boldsymbol{\theta}'$ and transitioning to $\boldsymbol{\theta}$
  - Formally: $\pi(\boldsymbol{\theta})p(\boldsymbol{\theta}'|\boldsymbol{\theta}) = \pi(\boldsymbol{\theta}')p(\boldsymbol{\theta}|\boldsymbol{\theta}')$

Markov Chain Monte Carlo (MCMC) Sampling: Practice

- Even if we satisfy detailed balance, it may take a while until we sample from the stationary distribution
  - This time is called the mixing time
  - Intuitively, the time until the chain forgot where it started
- In practice, the mixing time is not known
  - We only start taking samples from the chain after a burning in phase (the length of this is a hyperparameter)

The Mixing Time Depends on the Distribution



[Image source: Goodfellow et al., 2016 (Figure 17.1)]

Famous MCMC Algorithm (from 1953): Metropolis-Hastings

- Proposal distribution: $g(\boldsymbol{\theta}_2|\boldsymbol{\theta}_1)$ for generating new states
- Acceptance ratio $A(\boldsymbol{\theta}_2, \boldsymbol{\theta}_1)$
- Transition probability is then: $p(\boldsymbol{\theta}_2|\boldsymbol{\theta}_1) = g(\boldsymbol{\theta}_2|\boldsymbol{\theta}_1) \cdot A(\boldsymbol{\theta}_2, \boldsymbol{\theta}_1)$
- Acceptance ratio selected to always satisfy detailed balance:

$$A(\boldsymbol{\theta}_2, \boldsymbol{\theta}_1) = \min(1, \frac{p_{target(\boldsymbol{\theta}_2)}}{p_{target(\boldsymbol{\theta}_1)}} \frac{g(\boldsymbol{\theta}_1|\boldsymbol{\theta}_2)}{g(\boldsymbol{\theta}_2|\boldsymbol{\theta}_1)})$$

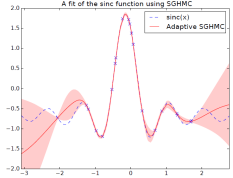
Full algorithm:

- Pick an initial $\boldsymbol{\theta}_0$ (e.g., at random); set $t = 0$
- For $t = 0, \ldots, T$:
  - Generate a next candidate state $\boldsymbol{\theta}' \sim g(\boldsymbol{\theta}'|\boldsymbol{\theta}_t)$
  - Accept with probability $\min(1, \frac{p_{target(\boldsymbol{\theta}_1)}}{p_{target(\boldsymbol{\theta}_2)}} \frac{g(\boldsymbol{\theta}_2|\boldsymbol{\theta}_1)}{g(\boldsymbol{\theta}_1|\boldsymbol{\theta}_2)})$

MCMC Can Be Sped Up By Gradients

- Gradient-based updates of $\boldsymbol{\theta}$ are much faster than sampling following a simpler proposal distribution
  - Hamiltonian Monte Carlo (HMC) takes into account gradients
  - Special case: Langevin Dynamics: $\Delta\boldsymbol{\theta}_t = \frac{\epsilon_t}{2}\left(\nabla \log p(\boldsymbol{\theta}_t) + \nabla \log p(\mathcal{D}|\boldsymbol{\theta}_t)\right) + \eta_t$, with $\eta_t \sim \mathcal{N}(0, \epsilon)$
- $p(\mathcal{D}|\boldsymbol{\theta})$ depends on the full data, like batch gradient descent
  - Can be sped up by considering mini batches of n data points: $p(\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n|\boldsymbol{\theta})$ $\rightsquigarrow$ Stochastic Gradient Langevin Dynamics (SGLD) [Welling Teh, 2011]
  - SGLD is almost identical to SGD, just adds $\eta_t \sim \mathcal{N}(0, \epsilon)$
- General HMC can also work with mini-batches

  $\rightarrow$ SGHMC [Chen et al, 2014]
  - We've also extended SGHMC for Bayesian optimization with Bayesian neural networks [Springenberg et al., 2016]



[Image source: Springenberg et al., 2016]

Questions to Answer for Yourself / Discuss with Friends

- Repetition: Explain MCMC in your own words.
- Repetition: What is the condition for a "stationary distribution"?
- Repetition: What is the relation between the mixing time and the length of the burning in phase?

Lecture Overview

Foundations of Deep Learning, Winter Term 2021/22

Week 13: Uncertainty in Deep Learning

Prior-Fitted Networks for Bayesian Inference

Frank Hutter Abhinav Valada

University of Freiburg

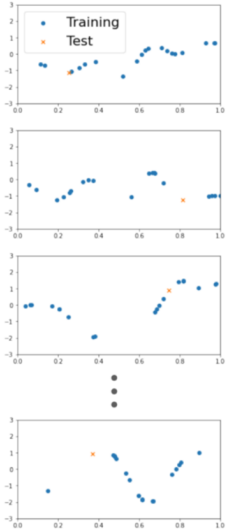
Motivation for Prior-Fitted Networks (PFNs) [Müller et al, 2022]

- So far, we estimated $p(\boldsymbol{\theta}|\mathcal{D}_{train})$ via variational inference or MCMC
- To make predictions for a new data point $\boldsymbol{x}_{test}$, we would then compute the predictive distribution as:

$$p(y_{test}|\boldsymbol{x}_{test}, \mathcal{D}_{train}) = \int_{\boldsymbol{\theta}} p(\boldsymbol{\theta}|\mathcal{D}_{train}) p(y_{test}|\boldsymbol{x}_{test}, \boldsymbol{\theta}) d\boldsymbol{\theta}$$

- In prior-fitted networks, we will directly approximate $p(y_{test}|\boldsymbol{x}_{test}, \mathcal{D}_{train})$
  - We will assume that we can sample datasets $\mathcal{D}^{(i)} \sim p(\mathcal{D})$
  - We will draw many datasets $\mathcal{D}^{(i)}$ and split each into $\mathcal{D}^{(i)}_{train}$ and $\mathcal{D}^{(i)}_{test} = \{\boldsymbol{x}^{(i)}_{test}, y^{(i)}_{test}\}$
  - We will then learn a model that predicts $y^{(i)}_{test}$ from $\langle \boldsymbol{x}^{(i)}_{test}, \mathcal{D}^{(i)}_{train} \rangle$
  - This model has then learned to predict $p(y_{test}|\boldsymbol{x}_{test}, \mathcal{D})$ in a single forward pass
  - PFNs might be a disruptive alternative to variational inference and MCMC

Illustration of Prior-Fitted Networks (PFNs)

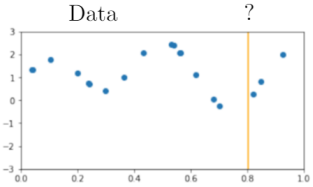


Samples from the prior

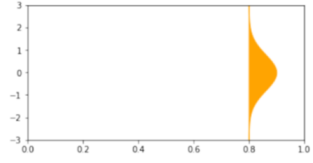
Learn a model to predict test from train

We call this model a prior-fitted network(PFN)

We have thus meta-learned to approximate Bayesian inference, purely by supervised learning with set-valued inputs

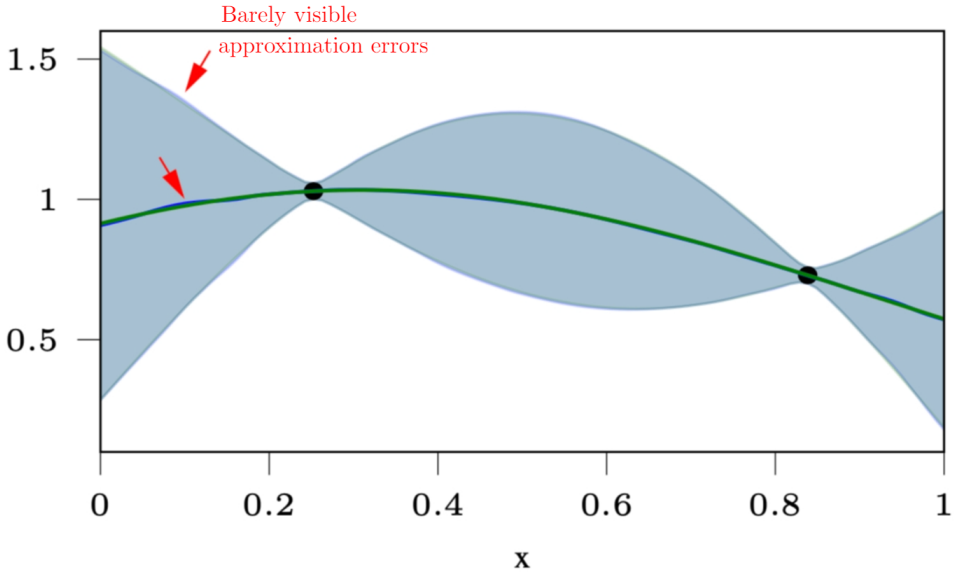


Perform a forward pass with the PFN

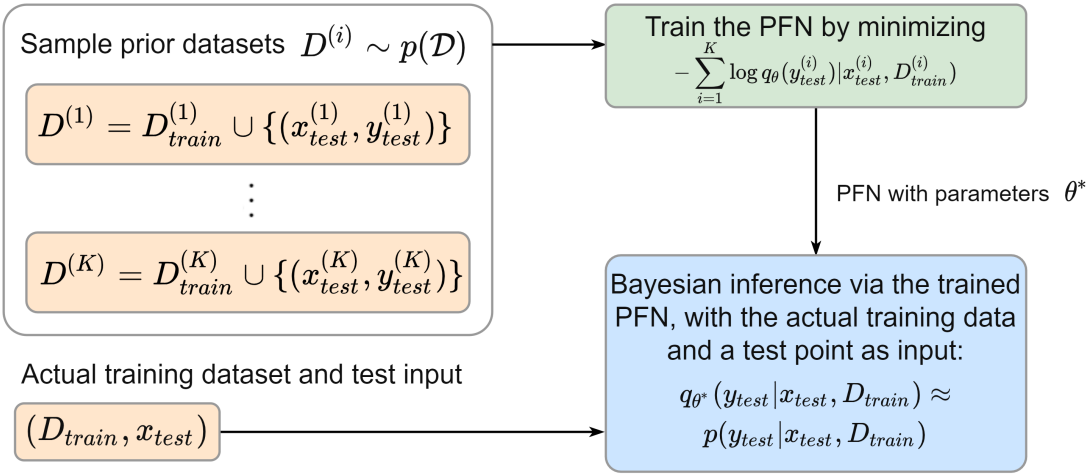


Posterior predictive distribution

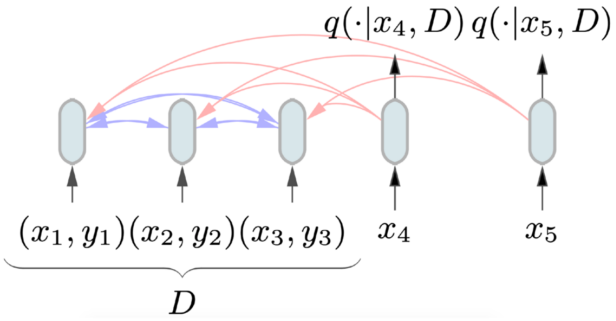
Gaussian Process Approximation



Some Details for Prior-Fitted Networks (PFNs)

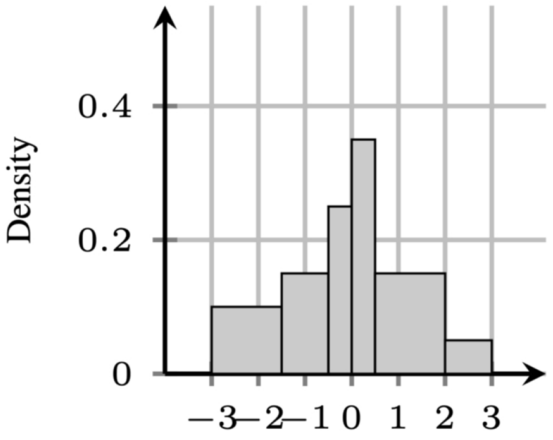


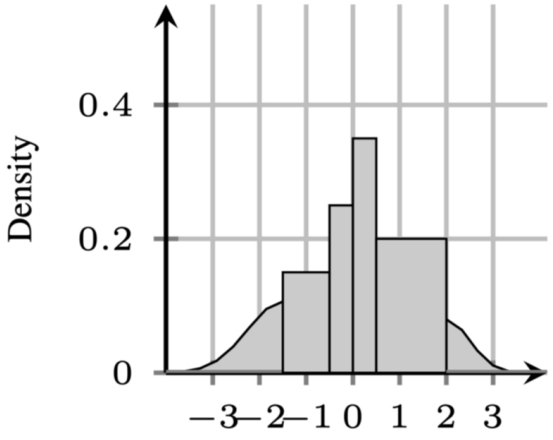
The Model We Use for PFNs: a Permutation-Invariant Transformer



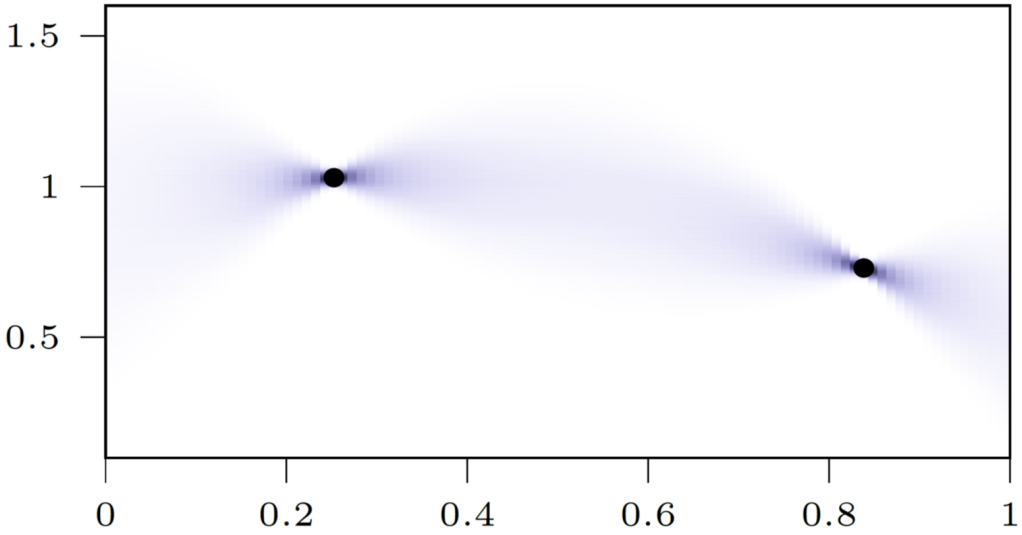
- We simply drop the positional embedding to obtain permutation invariance
- We do not model correlations between outputs
- To draw joint samples, we would draw one at a time and then include it in the input

The Model Head: Regression as Classification

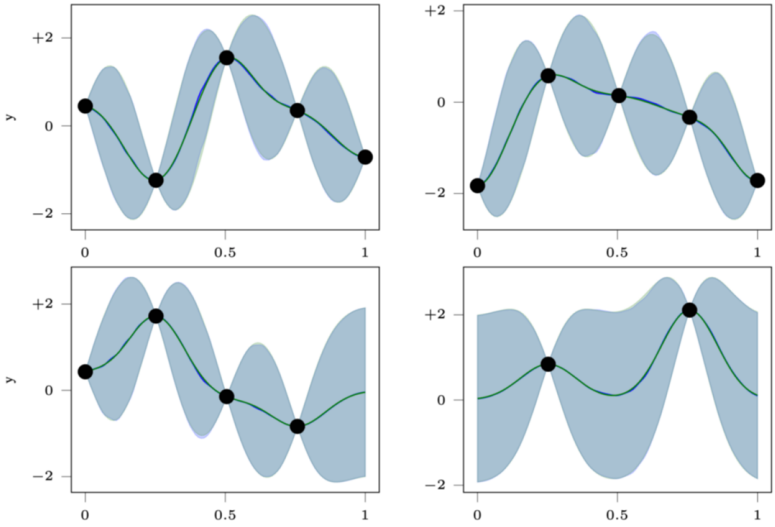




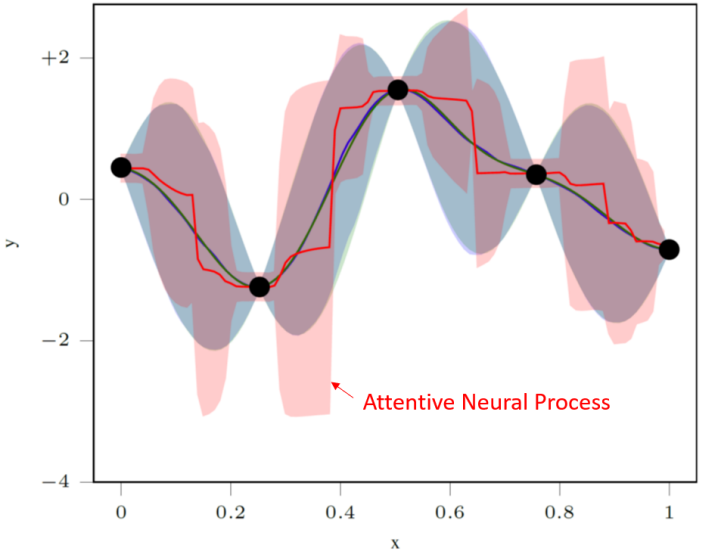
Regression as Classification



Results: Gaussian Process (GP) Approximation

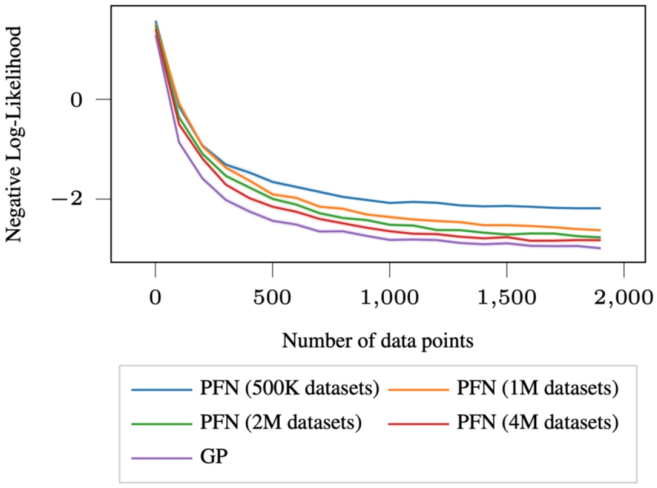


Results: Comparison to Attentive Neural Processes

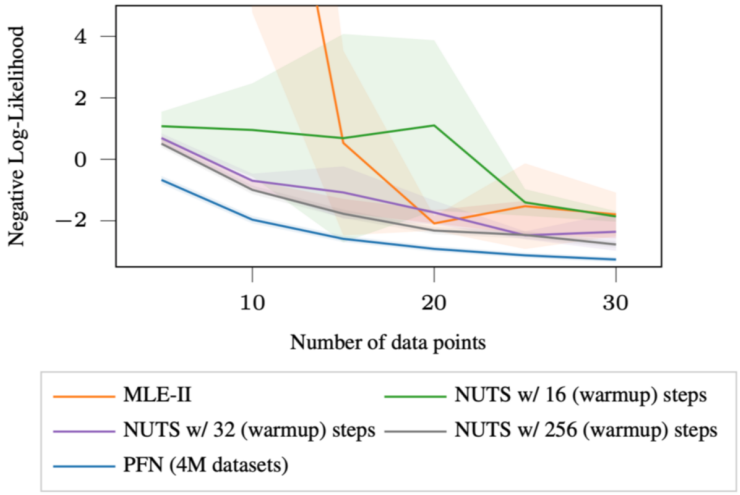


Results: Scalability

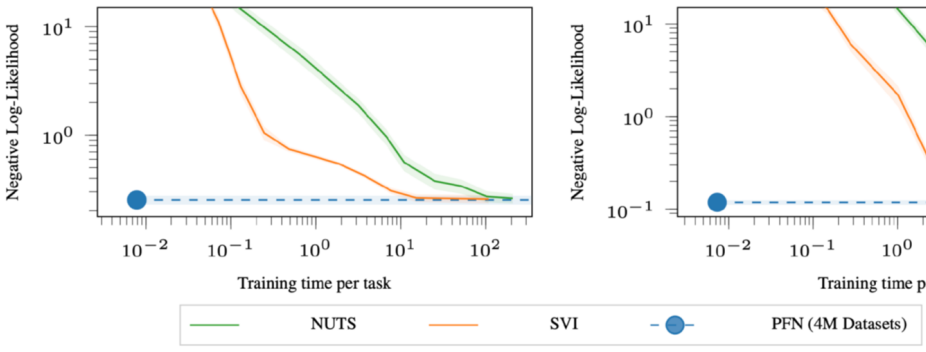
- Longer training (=seeing more datasets sampled from the prior) improves the approximation
- Direct scalability to about 4000 data points; more would need Transformer efficiency tricks



Results: GP Approximation with Unknown Hyperparameters



Results: Bayesian Neural Networks Approximation



- Results for two different network architectures
- (a) a BNN with 3 features, 2 layers and a hidden dimensionality of 5 and
- (b) a BNN with 8 features, 2 layers and a hidden dimensionality of 64.

Application to Binary Classification in Tabular Datasets

Experiment setup

- Binary, balanced classification
- 20 datasets from OpenML
- 30 training data points, 70 test
- Up to 60 features

Prior for generating D synthetic bin. class. datasets:

- Sample a neural architecture
- Sample all its weights from $N(0,1)$
- Sample entries of input matrix $Xi.i.d$ from $N(0,1)$
- Forwardprop X to obtain logits
- Binarize: set y to 1 for top 50% logits, to 0 for others

This is the first time real data enters the picture

Learn a PFN to minimize

$$\sum_{i=1:D} p(y_{31}^{(i)},\ldots,y_{100}^{(i)}|(x_1^{(i)},y_1^{(i)}),\ldots,(x_{30}^{(i)},y_{30}^{(i)}),x_{31}^{(i)},\ldots,x_{100}^{(i)})$$

Real training dataset & test x values

$$(x_1,y_1),\ldots,(x_{30},y_{30}),x_{31},\ldots,x_{100})$$

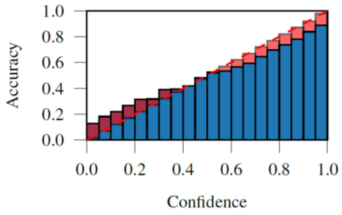
Perform a forward pass with the PFN

$$p(y_{31},\ldots,y_{100}|(x_1,y_1),\ldots,(x_{30},y_{30}),x_{31},\ldots,x_{100})$$

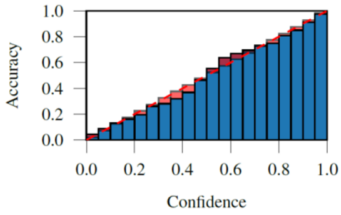
PFNs Outperform all the Baselines, with a Single Forward Prop

Metric	PFN-BNN	PFN-GP	Log. Reg.	GP	KNN	BNN	Catboost	XGB
Mean rank ROC AUC	**2.786**	3.833	4.690	5.286	6.214	5.000	4.833	3.357
Loss/Tie/Win vs PFN-BNN	–	14/1/6	16/1/4	17/0/4	17/1/3	17/1/3	13/1/7	12/2/7
Wilcoxon p. vs PFN-BNN	–	3.6e-13	3.6e-13	3.9e-13	3.6e-13	3.6e-13	1.9e-12	1.6e-06
Expected Calibration Error	**0.025**	0.067	0.157	0.095	0.093	0.089	0.157	0.066
Benchmark Time	GPU: **0:00:13** CPU: 0:04:23		0:09:56	0:24:30	0:00:34	12:04:41	2:05:20	20:59:46

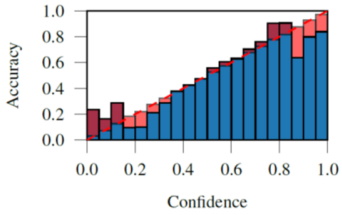
(a) Bayesian Neural Network



(b) PFN-BNN



(c) XGBoost



Questions to Answer for Yourself / Discuss with Friends

- Repetition:
  Can you explain the idea of PFNs in your own words?
- Repetition:
  Why are PFNs permutation-invariant?
- Transfer:
  If you wanted a PFN to behave like a random forest, how would you define the prior?

Lecture Overview

Foundations of Deep Learning, Winter Term 2021/22

Week 13: Uncertainty in Deep Learning

Alternative Treatments of Uncertainty in Neural Networks

Frank Hutter Abhinav Valada

University of Freiburg

Empirical Uncertainty Estimates via Ensembling

1. Construct an ensemble of M neural networks on the training data
   - More on how to do this on the next slides
2. For each test data point $\mathbf{x}$, predict with each of the M networks
   - Let $f_i(\mathbf{x})$ denote the output of the i-th network
3. Combine outputs of the M networks to obtain probabilistic predictions
   - For classification: average probabilistic (softmax) outputs $f_i(\mathbf{x})$:

$$f(\mathbf{x}) = \frac{1}{M} \sum_{i=1}^{M} f_i(\mathbf{x})$$

   - For regression: compute empirical mean and variance across the point predictions $f_i(\mathbf{x})$:

$$\begin{aligned} \boldsymbol{\mu}_{\mathbf{x}} &= \frac{1}{M} \sum_{i=1}^{M} f_i(\mathbf{x}) \\ \boldsymbol{\sigma}^2_{\mathbf{x}} &= \frac{1}{M} \sum_{i=1}^{M} (f_i(\mathbf{x}) - \boldsymbol{\mu}_{\mathbf{x}})^2 \end{aligned}$$

Different Ways to Construct Ensembles of M Neural Networks

- Via dropout $\rightarrow$ MC-DropOut [Gal Ghahramani, 2015]
  - Very popular, since training does not change at all
  - To obtain M networks, simply sample M random dropout masks
  - No overhead at training time; of course, still M-fold cost at test time
- Via Markov Chain Monte Carlo (MCMC)
- Via different random seeds for initialization and SGD [Lakshminarayanan et al., 2017]
  - These deep ensembles are surprisingly effective: state-of-the-art performance in 2019 [Ovadia et al, 2019]
  - Only disadvantage: M-fold overhead at training time
- Snapshot ensemble obtained via SGD with restarts (SGDR) [Huang et al., 2017]
  - No overhead at training time; of course, still M-fold cost at test time

Different Ways to Construct Ensembles of M Neural Networks

- Via different hyperparameter settings for the training pipeline
  - Hyperdeep ensembles [Wenzel et al., 2020]
  - Can directly reuse the hyperparameter evaluations from HPO [Feurer et al., 2015]

- Via different architectures
  - Recall that we said the Bayesian spirit is to deal with all sources of uncertainty
  - Different architectures often yield very different predictions
  - Neural ensemble search [Zaidi et al, 2020]

Predictive Uncertainty Estimates for Regression Tasks via Parametric Output Distributions

- We can train a network (with weights $\boldsymbol{\theta}$) to output the parameters $\mathbf{w}$ of a parametric model $p(\mathbf{y}|\mathbf{x}, \mathbf{w})$
  - Most common: Gaussian $\mathcal{N}(\mu, \sigma^2)$ ($\mathbf{w}$ parameterizes μ and σ^2)
  - For robustness: rather have network output μ and $\log(\sigma^2)$

- Simply maximize the log-likelihood w.r.t. network weights $\boldsymbol{\theta}$:

$$\log p(\mathcal{D} \mid \boldsymbol{\theta}) = \frac{1}{N} \sum_{i=1}^{N} \log p(\mathbf{y}_i \mid \mathbf{w}(\mathbf{x}_i, \boldsymbol{\theta}))$$

- The predictive distribution for an input $\mathbf{x}$ is then defined as:

$$p(\mathbf{y} \mid \mathbf{x}, \boldsymbol{\theta}) \equiv p(\mathbf{y} \mid \mathbf{w}(\mathbf{x}, \boldsymbol{\theta})).$$

Combining Predictive and Empirical Uncertainty Estimates

- We can combine predictive uncertainties with ensembling
- Each network i with weights $\boldsymbol{\theta}_i$ predicts a probability distribution

$$p(\mathbf{y} \mid \mathbf{x}, \boldsymbol{\theta}_i) \equiv p(\mathbf{y} \mid \mathbf{w}(\mathbf{x}, \boldsymbol{\theta}_i))$$

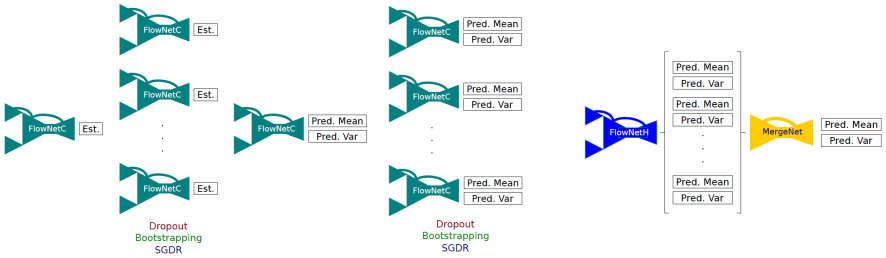
  with mean $\boldsymbol{\mu}_{\mathbf{x},i}$ and variance $\boldsymbol{\sigma}^2_{\mathbf{x},i}$

- We can combine these predictions in a mixture distribution
  - Intuitively, we need to combine the uncertainty of each network and the disagreement between networks
  - We use the law of total variance to compute the mean $\boldsymbol{\mu}_{\mathbf{x}}$ and variance $\boldsymbol{\sigma}^2_{\mathbf{x}}$ of the mixture distribution as:

$$\boldsymbol{\mu}_{\mathbf{x}} = \frac{1}{M}\sum_{i=1}^{M} \boldsymbol{\mu}_{\mathbf{x},i} \qquad \boldsymbol{\sigma}^2_{\mathbf{x}} = \frac{1}{M}\sum_{i=1}^{M} \left((\boldsymbol{\mu}_{\mathbf{x},i} - \boldsymbol{\mu}_{\mathbf{x}})^2 + \boldsymbol{\sigma}^2_{\mathbf{x},i} \right)$$

Overview of Alternatives + an Alternative in a Single Net

- Application: optical flow (thus the name FlowNet)
  - Collaboration between our CV + ML labs, ECCV 2018

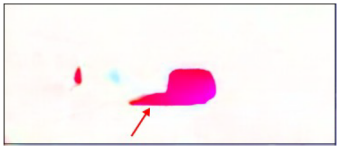


Standard | ensemble | predictive | predictive + ensembles | predictive + ensembles in a single net

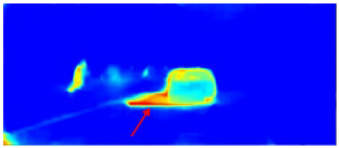
[Image source: Ilg et al, 2018]

As Desired, Large Uncertainty in Regions of Large Error

Image from KITTI dataset



Estimated optical flow



Estimated uncertainty

Also see the video

No network will ever be perfect,
but especially in safety-critical applications
it helps to know when we don't know

[Image source: Ilg et al, 2018]

Questions to Answer for Yourself / Discuss with Friends

- Repetition:
  How can you obtain a predictive distribution from an ensemble of networks?
- Repetition:
  Name four ways to construct an ensemble of networks
- Repetition:
  How can you construct a network that outputs a predictive distribution for a regression task?

Lecture Overview

Foundations of Deep Learning, Winter Term 2021/22

Week 13: Uncertainty in Deep Learning

Summary, Further Reading, References

Frank Hutter Abhinav Valada

University of Freiburg

Summary by learning goals

Having heard this lecture, you can now ...

- motivate the study of uncertainty in deep learning
- list some properties of Gaussian distributions
- derive Bayesian linear regression
- describe variational inference and derive the evidence lower bound
- describe MCMC and some MCMC algorithms
- explain prior-fitted networks and how they provide an alternative to variational inference and MCMC
- describe alternative methods to handle uncertainty in deep learning

Acknowledgement of Sources

Main sources for this lecture:

- Bayesian linear regression:
  - Chapter 2 of the excellent book Gaussian processes for Machine Learning
  - Philipp Hennig's Gaussian process tutorials
- MCMC: Chapter 17 of the Deep Learning book
- Prior-fitted networks: the original paper [Müller et al, 2022]

References

Hein, M., Andriushchenko, M., Bitterwolf, J. (2018)
Why ReLU networks yield high-confidence predictions far away from the training data and how to mitigate the problem
International Conference on Machine Learning
https://arxiv.org/pdf/1812.05720.pdf

Nguyen, A., Yosinski, J., Clune, J. (2015)
Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images
Proceedings of the IEEE conference on computer vision and pattern recognition
https://arxiv.org/pdf/1412.1897.pdf

Springenberg, J., Klein, A., Falkner, S., Hutter, F. (2016)
Bayesian optimization with robust Bayesian neural networks
Advances in neural information processing systems, 4134–4142
https://papers.nips.cc/paper/2016/file/a96d3afec184766bfeca7a9f989fc7e7-Paper.pdf

Blundell, C., Cornebise, J., Kavukcuoglu, K., Wierstra, D. (2015)
Weight uncertainty in neural network
International Conference on Machine Learning, 1613–1622
http://proceedings.mlr.press/v37/blundell15.pdf

Snoek, J., Rippel, O., Swersky, K., Kiros, R., Satish, N., Sundaram, N., Patwary, M., Prabhat, M., Adams, R. (2015)
Scalable bayesian optimization using deep neural networks
International Conference on Machine Learning, 2171–2180
http://proceedings.mlr.press/v37/snoek15.pdf

Goodfellow, I., Bengio, Y., Courville, A. (2016)
Deep Learning
MIT Press
https://www.deeplearningbook.org/

Welling, M., Teh, Y. (2011)
Bayesian learning via stochastic gradient Langevin dynamics
Proceedings of the 28th international conference on machine learning (ICML-11), 681–688
https://www.stats.ox.ac.uk/~teh/research/compstats/WelTeh2011a.pdf

Chen, T., Fox, E., Guestrin, C. (2014)
Stochastic gradient hamiltonian monte carlo
International conference on machine learning, 1683–1691
http://proceedings.mlr.press/v32/cheni14.pdf

Müller, S., Hollman, N., Arango Pineda, S., Grabocka, J., Hutter, F. (2022)
Transformers Can Do Bayesian Inference
https://openreview.net/forum?id=KSugKcbNf9

Gal, Y., Ghahramani, Z. (2015)
Dropout as a bayesian approximation: Representing model uncertainty in deep learning
International conference on machine learning, 1050–1059
http://proceedings.mlr.press/v48/gal16.pdf

Lakshminarayanan, B., Pritzel, A., Blundell, C. (2017)
Simple and scalable predictive uncertainty estimation using deep ensembles
https://arxiv.org/pdf/1612.01474.pdf

References

Ovadia, Y., Fertig, E., Ren, J., Nado, Z., Sculley., D, Nowozin, S., Dillon, J., Lakshminarayanan, B., Snoek, J. (2019)
Can You Trust Your Model's Uncertainty? Evaluating Predictive Uncertainty Under Dataset Shift
https://arxiv.org/pdf/1906.02530.pdf

Huang, G., Li, Y., Pleiss, G., Liu, Z., Hopcroft, J., Weinberger, K. (2017)
Snapshot Ensembles: Train 1, get M for free
https://arxiv.org/pdf/1704.00109.pdf

Wenzel, F., Snoek, J., Tran, D., Jenatton, R. (2020)
Hyperparameter ensembles for robustness and uncertainty quantification
https://arxiv.org/pdf/2006.13570.pdf

Feurer, M., Klein, A., Eggensperger, K., Springenberg, J., Blum, M., Hutter, F. (2015)
Efficient and Robust Automated Machine Learning
https://papers.nips.cc/paper/2015/file/11d0e6287202fced83f79975ec59a3a6-Paper.pdf

Zaidi, S., Zela, A., Elsken, T., Holmes, C., Hutter, F., Teh, Y. (2015)
Neural Ensemble Search for Uncertainty Estimation and Dataset Shift
https://arxiv.org/abs/2006.08573

Ilg, E., Cicek, O., Galesso, S., Klein, A., Makansi, O., Hutter, F., Brox, T. (2018)
Uncertainty estimates and multi-hypotheses networks for optical flow
Proceedings of the European Conference on Computer Vision (ECCV), 652–667

https://lmb.informatik.uni-freiburg.de/Publications/2018/ICKMB18/paper-Uncertainty-FlowNetH.pdf