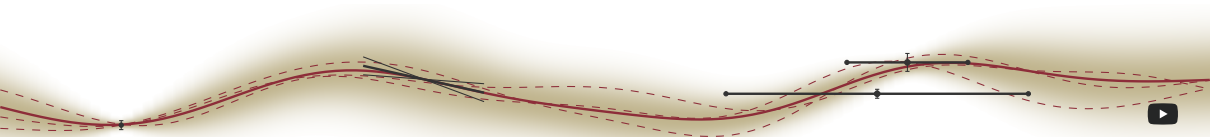
Probabilistic Machine Learning
Lecture 17
Probabilistic Deep Learning

Philipp Hennig

29 June 2023

Eberhard Karls
Universität
Tübingen

Faculty of Science
Department of Computer Science
Chair for the Methods of Machine Learning



For our purposes, a **deep neural network** is a function

$$f(\boldsymbol{x}, \boldsymbol{\theta}) : \mathbb{X} \times \mathbb{R}^D \to \mathbb{R}^F$$

parametrized by *parameters* $\boldsymbol{\theta} \in \mathbb{R}^D$ and mapping inputs $\boldsymbol{x} \in \mathbb{X}$ to outputs $f(\boldsymbol{x}, \boldsymbol{\theta}) \in \mathbb{R}^F$. The network is trained by **empirical risk minimization** (ERM), to find parameters $\boldsymbol{\theta}_*$ on a training set $\mathcal{D} = [(x_i, y_i)]_{i=1,\dots,N}$

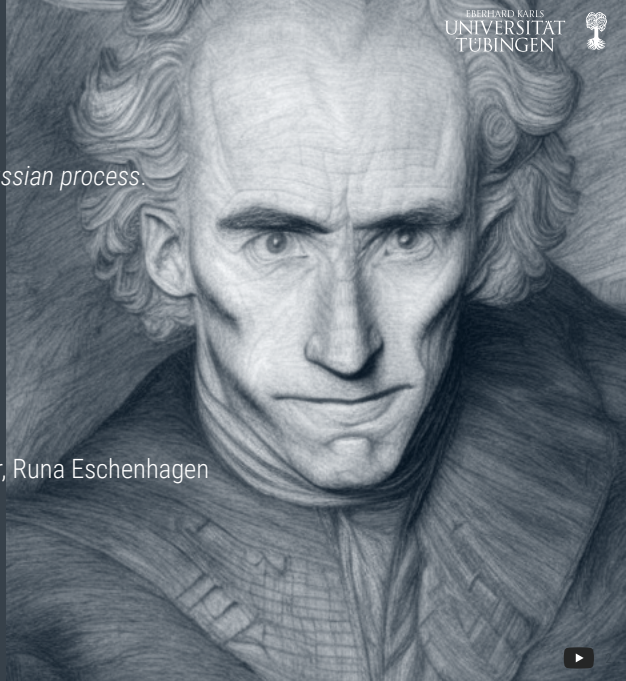
$$\boldsymbol{\theta}_* = \arg\min_{\boldsymbol{\theta} \in \mathbb{R}^D} \underbrace{\mathcal{L}(\boldsymbol{\theta})}_{\text{Loss}} = \arg\min_{\boldsymbol{\theta} \in \mathbb{R}^D} \left(\frac{1}{N} \sum_{i=1}^{N} \underbrace{\ell(y_i; \overbrace{f(x_i, \boldsymbol{\theta})}^{\text{deep net}})}_{\text{empirical risk}} + \underbrace{r(\boldsymbol{\theta})}_{\text{regularizer}} \right).$$

Today:

How to turn *any* deep network into a *Gaussian process*.

Courtesy of

- Pierre-Simon, Marquis de Laplace, 1810
- David JC MacKay, 1998
- Emtiyaz Khan, 2019
- Agustinus Kristiadi, 2020
- Alex Immer, 2021
- Matthias Bauer, Vincent Fortuin, Eric Daxberger, Runa Eschenhagen
- and many others whose papers I missed



Deep networks *are* GPs

1. Realise that the loss is a **negative log-posterior**

$$\mathcal{L}(\boldsymbol{\theta}) = \left(\frac{1}{N} \sum_{i=1}^{N} \underbrace{\ell(y_i; \overbrace{f(x_i, \boldsymbol{\theta})}^{\text{deep net}})}_{\text{empirical risk}} + \underbrace{r(\boldsymbol{\theta})}_{\text{regularizer}} \right) = -\sum_{i=1}^{N} \log p(\boldsymbol{y} \mid \boldsymbol{\theta}) - \log p(\boldsymbol{\theta}) = -\log p(\boldsymbol{\theta} \mid \boldsymbol{y}) + \text{const.}$$

2. **Train the deep net *as usual*** to find $\boldsymbol{\theta}_* = \arg\max_{\boldsymbol{\theta} \in \mathbb{R}^D} p(\boldsymbol{\theta} \mid \boldsymbol{y})$
3. At $\boldsymbol{\theta}_*$, compute a **Laplace approximation** of the log-posterior, with $\Psi := -\nabla\nabla^\intercal \log p(\boldsymbol{\theta}_* \mid \boldsymbol{y})$

$$\log p(\boldsymbol{\theta} \mid \boldsymbol{y}) + \text{const.} = \mathcal{L}(\boldsymbol{\theta}) \approx \mathcal{L}(\boldsymbol{\theta}_*) + 1/2(\boldsymbol{\theta} - \boldsymbol{\theta}_*)^\intercal \Psi (\boldsymbol{\theta} - \boldsymbol{\theta}_*) = \log \mathcal{N}(\boldsymbol{\theta}; \boldsymbol{\theta}_*, -\Psi^{-1})$$

4. **Linearize** $f(\boldsymbol{x}, \boldsymbol{\theta})$ around θ_*, with $[J(\boldsymbol{x})]_{ij} = \frac{\partial f_i(\boldsymbol{x}, \boldsymbol{\theta}_*)}{\partial \theta_j}$ as $f(\boldsymbol{x}, \boldsymbol{\theta}) \approx f(\boldsymbol{x}, \boldsymbol{\theta}_*) + J(\boldsymbol{x}, \boldsymbol{\theta}_*)(\boldsymbol{\theta} - \boldsymbol{\theta}_*)$

$$\text{thus } p(f(\bullet) \mid \mathcal{D}) = \int p(f \mid w)\, dp(w) \approx \mathcal{GP}(f(\bullet); f(\bullet, \boldsymbol{\theta}_*), -J(\bullet)\Psi^{-1}J(\circ)) \quad \text{with}$$

$$\mathbb{E}(f(\bullet)) = f(\bullet, \boldsymbol{\theta}_*) \qquad \text{the trained net as the mean function}$$

$$\text{cov}(f(\bullet), f(\circ)) = -J(\bullet)\Psi^{-1}J(\circ)^\intercal \qquad \text{the Laplace tangent kernel as the covariance function}$$

Another Look

What are the computational steps necessary?

$$p(f(\bullet) \mid \mathcal{D}) = \mathcal{GP}(f(\bullet);\ \underbrace{f(\bullet, \boldsymbol{\theta}_*)}_{\text{trained network}},\ \underbrace{-J(\bullet)\Psi^{-1}J(\circ)^\intercal}_{\text{Laplace tangent kernel}})$$

Overhead:

- At **Train Time**: After training is completed, compute

$$\Psi \in \mathbb{R}^{D\times D} = \nabla_\theta \nabla_\theta^\intercal \mathcal{L} = \nabla_\theta \nabla_\theta \left(\frac{1}{N} \sum_{i=1}^{N} \ell(y_i; f(x_i, \boldsymbol{\theta})) + r(\boldsymbol{\theta}) \right)$$

  and its matrix-decomposition.

- At **Test Time**: For a new input $\boldsymbol{x}$, compute *backward pass* in addition to forward pass, to get

$$J(\boldsymbol{x}) = \nabla_\theta f(\boldsymbol{x}, \boldsymbol{\theta}_*)$$

Code

Laplace Approximations for Deep Learning

What's to like?

If you are a deep learning person…

- you get to keep your beloved point estimate $f(\bullet, \boldsymbol{\theta}_*)$
- you get to keep your beloved training procedure. Laplace is constructed *post-hoc*, even for pre-trained networks downloaded from the net (assuming model, data, and loss are available)
- you need only *auto-diff* and *numerical linear algebra*. No sampling, no stochasticity, no ensembles
- But the result is a GP, with all the trimmings (evidence, sampling, sparse decompositions, etc.)!

Laplace Approximations for Deep Learning

Challenges

- Hessian decomposition is $\mathcal{O}(D^3)$. But there are approximations. It's linear algebra!
- Laplace approximations are local. They can be "arbitrarily wrong" to the full posterior.
  - but they're still better than a point estimate!
  - and the loss functions of deep learning where never constructed to be good generative models in the first place!

The Laplace Tangent kernel

Some context

Whence the name?

- Jacot, Gabriel, Hongler, NeurIPS 219 introduce the *Neural tangent kernel*

$$k(\bullet, \circ) = J_{\theta_0}(\bullet) J_{\theta_0}^\intercal(\circ) = \sum_{d=1}^{D} \frac{\partial f_i(\bullet, \theta_0)}{\partial [\theta_0]_d} \frac{\partial f_j(\bullet, \theta_0)}{\partial [\theta_0]_d}$$

But they *do not make any connection to manifolds or tangents!* Their paper is a theoretical analysis of the flow of gradient descent, it does not yield meaningful uncertainty quantification.

Laplace approximations for Deep learning:

- Laplace approximations turn (nearly) *any* deep neural network into a Gaussian process
- they involve only *auto-diff* and *linear algebra*, both of which are robust and scalable
- Deep nets thus approximately inherit the probabilistic functionality
- for large-scale deep networks, care must be taken to find *approximate* solutions to the Hessian decomposition

Please cite this course, as

```
@techreport{Tuebingen_ProbML23,
    title =
    {Probabilistic Machine Learning},
    author = {Hennig, Philipp},
    series = {Lecture Notes
        in Machine Learning},
    year = {2023},
    institution = {Tübingen AI Center}}
```