Probabilistic Machine Learning
Lecture 19
Uses for Uncertainty in Deep Learning II

Philipp Hennig

6 July 2023

Eberhard Karls Universität Tübingen

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

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 $f(\boldsymbol{x}, \boldsymbol{\theta}) \approx f(\boldsymbol{x}, \boldsymbol{\theta}_*) + J(\boldsymbol{x}, \boldsymbol{\theta}_*)(\boldsymbol{\theta} - \boldsymbol{\theta}_*)$

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

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

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

Computing the *exact* Hessian Ψ is $\mathcal{O}(ND^2)$, and inverting it is $\mathcal{O}(D^3)$.
Ideas for Approximations:

- Sub-sample the dataset ($\mathcal{O}(MD^2)$ with $M \ll N$)
- structural approximations to the Hessian:
  - diagonal approximation: $\mathcal{O}(D)$ (inverse $\mathcal{O}(D)$)
  - last-layer approximation: $\mathcal{O}(D_L^2)$ (inverse $\mathcal{O}(D_L^3)$))
  - Kronecker factorized approximate curvature (KFAC): $\Psi \approx \operatorname{diag}([\Lambda_\ell \otimes \Omega_\ell]_{\ell=1,\dots,L})$ with $\Lambda_\ell \in \mathbb{R}^{\mathrm{in}_\ell \times \mathrm{in}_\ell}, \Omega_\ell \in \mathbb{R}^{\mathrm{out}_{\ell-1} \times \mathrm{out}_{\ell-1}}$ and thus inverse $\mathcal{O}\left(\sum_\ell \mathrm{in}_\ell^3 + \mathrm{out}_{\ell-1}^3\right)$)
  - Generalized Gauss-Newton (homework this week): $\Psi \approx \alpha I + GG^\intercal$ with $G \in \mathbb{R}^{D \times M}$
  - approximate eigenvalue decompositions using the Lanczos algorithm (cf. Lecture 13)

What’s not to like about deep learning?

What people **don’t** like about deep learning:

- deep learning has certain conceptual pathologies, leading to adversarial and out-of-distribution brittleness
- Training a deep net is fiddly, and requires *many* choices
- Once the model is trained, it’s unclear how to *update* it if new data arrives

Geometric / probabilistic interpretation of deep learning can help address these issues.

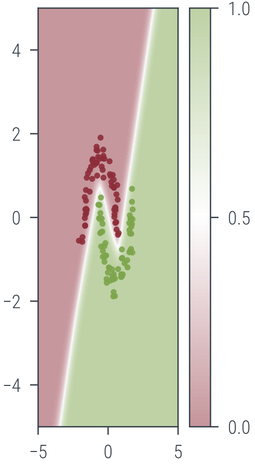
$$\boldsymbol{\theta}_* = \underset{\boldsymbol{\theta}\in\mathbb{R}^D}{\arg\min}\, \underbrace{\mathcal{L}(\boldsymbol{\theta})}_{\text{Loss}} = \underset{\boldsymbol{\theta}\in\mathbb{R}^D}{\arg\min} \left(\frac{1}{N} \sum_{i=1}^{N} \underbrace{\ell(y_i; \overbrace{f(x_i, \boldsymbol{\theta})}^{\text{deep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

<u>Theorem:</u> [Hein et al. 2019] If $f(x, \boldsymbol{\theta})$ is a **ReLU** network and $\ell = \sigma$ is the **cross-entropy** then, for almost any $x \in \mathbb{R}^d$ and $\varepsilon > 0$, there exists an $\alpha > 0$ and a class $k \in \{1, \dots, K\}$ such that for $z = \alpha x$

$$\frac{e^{f_k(x)}}{\sum_{r=1}^{K} e^{f_r(z)}} \geq 1 - \varepsilon \quad \text{and} \quad \lim_{\alpha \to \infty} \frac{e^{f_k(\alpha x)}}{\sum_{r=1}^{K} e^{f_r(\alpha x)}} = 1$$

Far from the data, the model is always confident.

<u>Intuition:</u> ReLU networks are piecewise linear, so far from the data, $f_i(\boldsymbol{x}, \theta) = A_i(\theta)\boldsymbol{x}$ and $f_{\arg\max_i A_i}(\boldsymbol{x}) - f_j = \mathcal{O}(\|\boldsymbol{x}\|)$, so $\lim_{\|\boldsymbol{x}\| \to \infty} \text{softmax}(f(\boldsymbol{x})) = 1$.



Being Bayesian, even just a bit, fixes overconfidence

Kristiadi, Hein, Hennig, ICML 2020

Sketch: (Proof in Kristiadi, Hein, Hennig, ICML 2020)

- Far from the data, $f_i(\boldsymbol{x}, \boldsymbol{\theta}) = A_i(\boldsymbol{\theta})\boldsymbol{x}$
- Consider *any* Gaussian measure on the weights $p(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}, \boldsymbol{\theta}_*, \Psi)$ (e.g. from Laplace approximation). The associated push-forward GP is

$$p(f(\boldsymbol{x})) = \mathcal{GP}(f(\boldsymbol{x}), f(\boldsymbol{x}, \boldsymbol{\theta}_*), J(\boldsymbol{x}, \boldsymbol{\theta}_*)\Psi J(\boldsymbol{x}, \boldsymbol{\theta}_*))$$

$$\stackrel{\|x\| \to \infty}{=} \mathcal{GP}(f(\boldsymbol{x}); A(\boldsymbol{\theta}_*)\boldsymbol{x}, \boldsymbol{x}^\intercal A'(\boldsymbol{\theta}_*)\Psi A'(\boldsymbol{\theta}_*)^\intercal \boldsymbol{x})$$

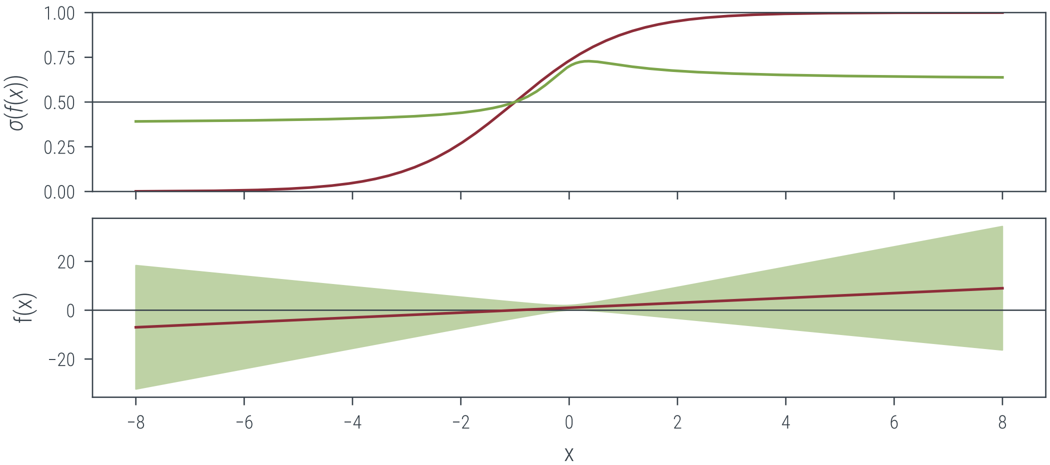
- The class prediction of this network is

$$\mathbb{E}(\sigma(f(\boldsymbol{x}))) = \int \sigma(f(\boldsymbol{x}))p(f(\boldsymbol{x}))\, df(\boldsymbol{x}) \approx \sigma\left(\frac{\mathbb{E}(f(\boldsymbol{x}))}{\sqrt{1 + \frac{\pi}{8}\operatorname{var}(f(\boldsymbol{x}))}}\right) \stackrel{\|x\| \to \infty}{=} \sigma\left(\frac{\mathcal{O}(\|x\|)}{\sqrt{1 + \mathcal{O}(\|x\|^2)}}\right)$$

$$\to \sigma(c) < 1 \quad \text{and} \quad > 0$$

Being Bayesian, even just a bit, fixes overconfidence

Kristiadi, Hein, Hennig, ICML 2020

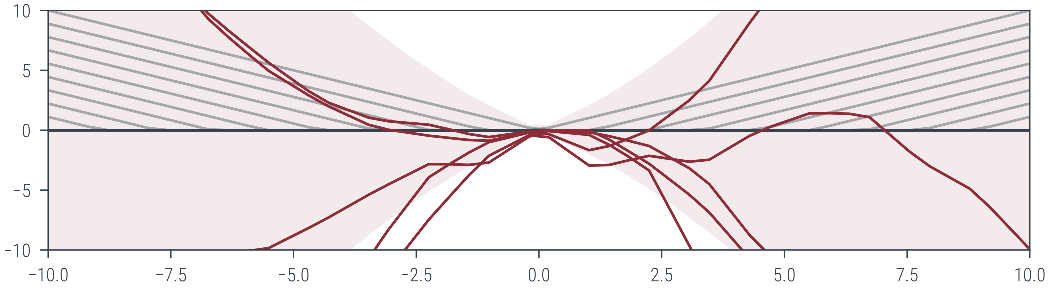


Finite networks have finite capacity

a role for nonparametric modeling in deep learning

Kristiadi, Hein, Hennig, NeurIPS 2021

- To achieve $\mathbb{E}(\sigma(f(\boldsymbol{x}))) = 1/C$ ("calibrated confidence") at $\|x\| \to \infty$, we need $\operatorname{var} f(\boldsymbol{x}) = \omega(\|\boldsymbol{x}\|^2)$.
- This can *not* be achieved with finitely many ReLU features in the network (there will always be a "last ReLU to switch on", and $f(\boldsymbol{x}) = A\boldsymbol{x}$ beyond that).
- But remember from Lecture 9 there's a limit process $\mathcal{GP}(0, k_{IWP})$ with $k_{IWP}(x, x) = \mathcal{O}(x^3)$

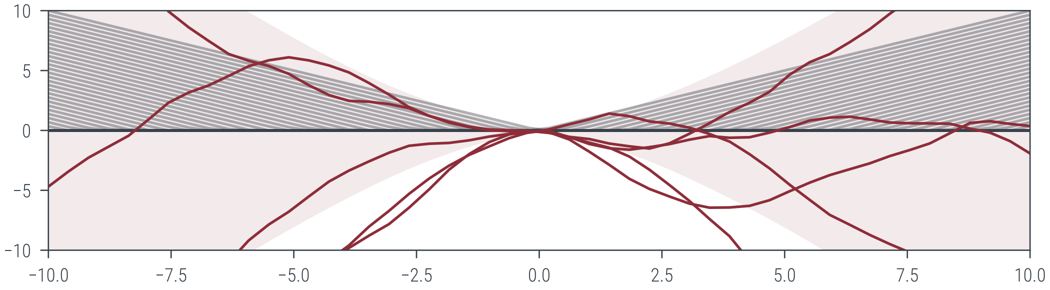


Finite networks have finite capacity

a role for nonparametric modeling in deep learning

Kristiadi, Hein, Hennig, NeurIPS 2021

- To achieve $\mathbb{E}(\sigma(f(\boldsymbol{x}))) = 1/C$ ("calibrated confidence") at $\|x\| \to \infty$, we need var $f(\boldsymbol{x}) = \omega(\|\boldsymbol{x}\|^2)$.
- This can *not* be achieved with finitely many ReLU features in the network (there will always be a "last ReLU to switch on", and $f(\boldsymbol{x}) = A\boldsymbol{x}$ beyond that).
- But remember from Lecture 9 there's a limit process $\mathcal{GP}(0, k_{IWP})$ with $k_{IWP}(x, x) = \mathcal{O}(x^3)$

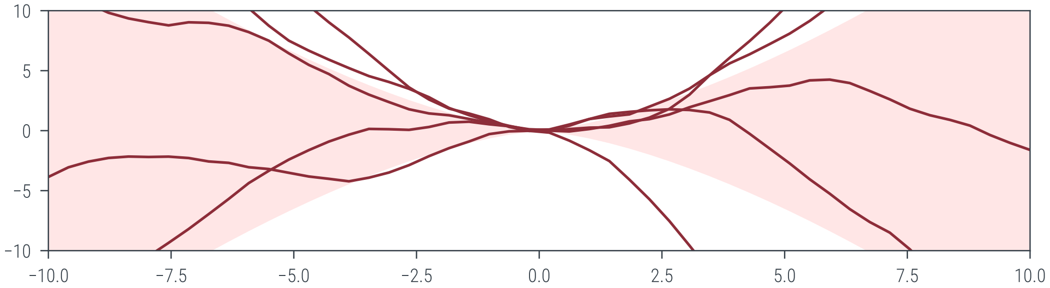


Finite networks have finite capacity

a role for nonparametric modeling in deep learning

Kristiadi, Hein, Hennig, NeurIPS 2021

- To achieve $\mathbb{E}(\sigma(f(\boldsymbol{x}))) = 1/C$ ("calibrated confidence") at $\|x\| \to \infty$, we need $\operatorname{var} f(\boldsymbol{x}) = \omega(\|\boldsymbol{x}\|^2)$.
- This can *not* be achieved with finitely many ReLU features in the network (there will always be a "last ReLU to switch on", and $f(\boldsymbol{x}) = A\boldsymbol{x}$ beyond that).
- But remember from Lecture 9 there's a limit process $\mathcal{GP}(0, k_{IWP})$ with $k_{IWP}(x, x) = \mathcal{O}(x^3)$



[G. Wahba, *Spline Models for Observational Data*, 1990]

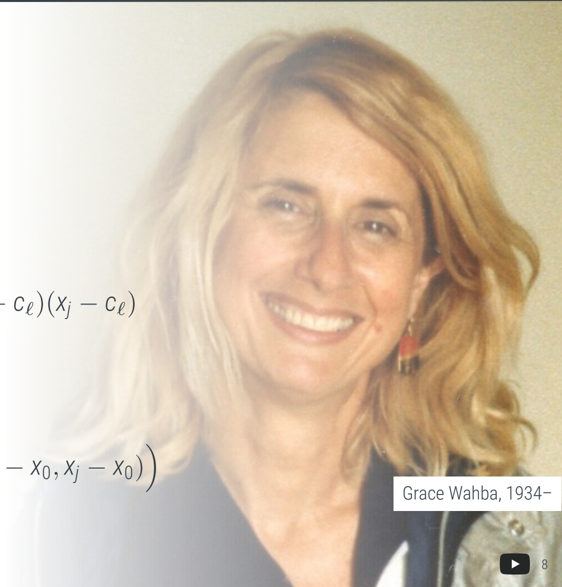
$$\phi_i(x) = \int_{-\infty}^{x} \theta(\tilde{x} - c_i)\, d\tilde{x} = \max(0, x - c_i)$$

$$k(x_i, x_j) = \frac{\sigma^2 (c_{\max} - c_0)}{F} \sum_{\ell} \phi_\ell(x_i)\phi_\ell(x_j)$$

$$= \frac{\sigma^2 (c_{\max} - c_0)}{F} \sum_{\ell} \max(0, \min(x_i, x_j) - c_\ell)(x_i - c_\ell)(x_j - c_\ell)$$

$$\to \sigma^2 \int_{c_0}^{\min(x_i, x_j)} (x_i - c)(x_j - c)\, dc$$

$$= \sigma^2 \left(\frac{1}{3} \min^3(x_i - x_0, x_j - x_0) + \frac{1}{2} |x_i - x_j| \min^2(x_i - x_0, x_j - x_0) \right)$$



Grace Wahba, 1934–

- Consider a dataset $X, \boldsymbol{y}$ w.l.o.g. scaled such that, $\|\boldsymbol{x}\| \leq 1$ for all $\boldsymbol{x} \in X$.
- Model classifier function as $\tilde{f}(\boldsymbol{x}) = f(\boldsymbol{x}) + \hat{f}(\boldsymbol{x})$ with a ReLU network $f(\boldsymbol{x})$ and $\hat{f}(\boldsymbol{x}) \sim \mathcal{GP}(0, k_{IWP})$.
- Then (Lemma in op.cit.), for $0 < \delta < 1$, if $\|a\|^2, \|b\|^2 \leq \delta$, then $k_{IWP}(a, b) \in \mathcal{O}(\delta^3)$.

<u>Proposition:</u> Assume the ReLU networks weights are distributed as $\mathcal{N}(\boldsymbol{\theta}, \boldsymbol{\theta}_*, \Sigma)$ and linearize $f(x) = f(x, \boldsymbol{\theta}_*) + J(x, \boldsymbol{\theta}_*)(\boldsymbol{\theta} - \boldsymbol{\theta}_*)$. Then the predictive GP posterior on $\tilde{f}$ has mean and covariance

$$\mathbb{E}(\tilde{f}(\bullet) \mid X, \boldsymbol{y}) = f(\bullet, \boldsymbol{\theta}_*) + k_{\bullet,X} C^{-1} (\boldsymbol{y} - f(X, \boldsymbol{\theta}_*))$$

$$\operatorname{var}(\tilde{f}(\bullet) \mid X, \boldsymbol{y}) = J(\bullet, \boldsymbol{\theta}_*) \Sigma J^\intercal(\bullet, \boldsymbol{\theta}_*) + k_{\bullet,\bullet} - k_{\bullet,X} C^{-1} k_{X,\bullet}$$

with $C := k_{IWP}(X, X) + \sigma^2 I + J(X, \boldsymbol{\theta}_*) \Sigma J^\intercal(X, \boldsymbol{\theta}_*)$. **If the network is well-trained ($f(X, \boldsymbol{\theta}_*) \approx \boldsymbol{y}$), then the residual is negligible.** We also have (simplified argument)

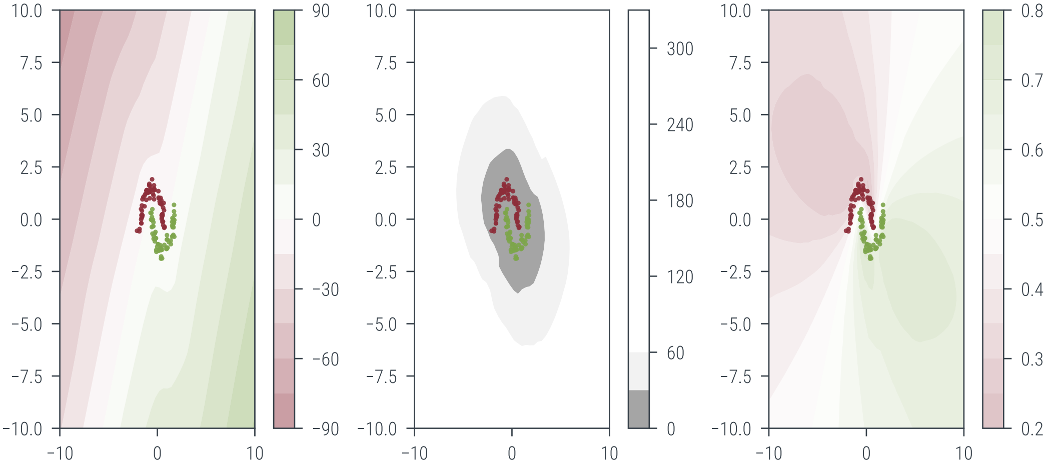
$$k_{\bullet,X} C^{-1} k_{X,\bullet} \leq \lambda_{\max}(C^{-1}) \|k_{\bullet,X}\|^2 = \lambda_{\max}(C^{-1}) \sum_{i=1}^{N} k^2_{\bullet,x_i} \leq \lambda_{\max}(C^{-1}) \sum_{i=1}^{N} k_{\bullet,\bullet} \underbrace{k_{x_i,x_i}}_{\approx 0}$$

Thus: $p(\tilde{f}(\bullet)) \approx \mathcal{GP}(\tilde{f}(\bullet); f(\bullet, \boldsymbol{\theta}_*), J(\bullet, \boldsymbol{\theta}_*) \Sigma J^\intercal(\bullet, \boldsymbol{\theta}_*) + k_{\bullet,\bullet})$.

ReLU-GP

Adding "infinitely many" untrained units to a deep net

Kristiadi, Hein, Hennig, NeurIPS 2021



Continual Learning

How to update a trained model when new data arrives

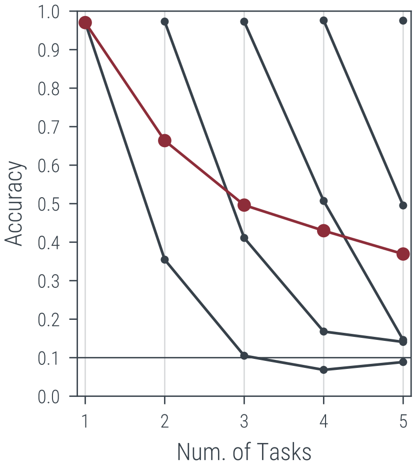
Kirkpatrick et al., PNAS 2017, Ritter et al., NeurIPS 2018

Consider a sequence of datasets $\mathcal{D}_i$, $i = 1, \dots, T$. At time i, previous datasets are no longer available. What do you do?

1. Just keep training™

(common alternative: "replay" old data

(e.g. https://openreview.net/forum?id=Q4aAITDgdP). Which is expensive)



Setup:

- Permuted copies of MNIST
- (784, 100, 10)-ReLU net
- Adam(lr=10^{-3}), 10 epochs each

Probabilistic inference naturally deals with continual learning

"yesterday's posterior is today's prior"

$$p(\boldsymbol{\theta} \mid \boldsymbol{y}_1, y_2) = \frac{p(\boldsymbol{\theta} \mid \boldsymbol{y}_1) \cdot p(\boldsymbol{y}_2 \mid \boldsymbol{\theta})}{p(\boldsymbol{y}_2)} \qquad \text{(assuming } \boldsymbol{y}_2 \perp\!\!\!\perp \boldsymbol{y}_1 \mid \boldsymbol{\theta}\text{)}$$

$$-\log p(\boldsymbol{\theta} \mid \boldsymbol{y}_1, y_2) = -\log p(\boldsymbol{y}_2 \mid \boldsymbol{\theta}) - \log p(\boldsymbol{\theta} \mid \boldsymbol{y}_1)$$

$$= \sum_{i=1}^{N_2} \ell(y_{2,i}, f(x_{2,i}, \boldsymbol{\theta})) + \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}_1)^\intercal \Psi_1 (\boldsymbol{\theta} - \boldsymbol{\theta}_1)$$

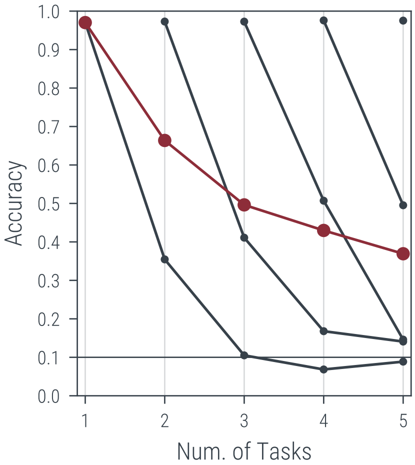
Continual Learning

How to update a trained model when new data arrives

Kirkpatrick et al., PNAS 2017, Ritter et al., NeurIPS 2018

Consider a sequence of datasets $\mathcal{D}_i$, $i = 1, \dots, T$. At time i, previous datasets are no longer available. What do you do?

1. Just keep training™

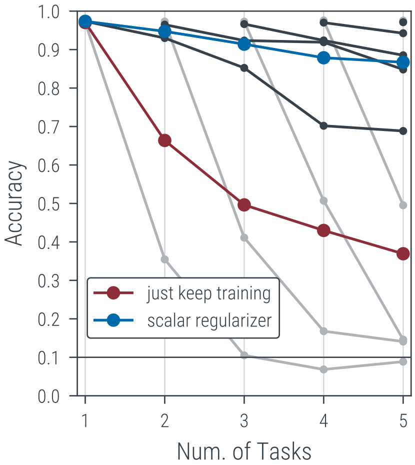


Setup:

- Permuted copies of MNIST
- (784, 100, 10)-ReLU net
- Adam(lr=10^{-3}), 10 epochs each

Consider a sequence of datasets $\mathcal{D}_i$, $i = 1, \dots, T$. At time i, previous datasets are no longer available. What do you do?

1. Just keep training™
2. regularizer $r(\boldsymbol{\theta}) = \frac{\lambda}{2}\|\boldsymbol{\theta} - \boldsymbol{\theta}_{i-1}\|^2$



Setup:

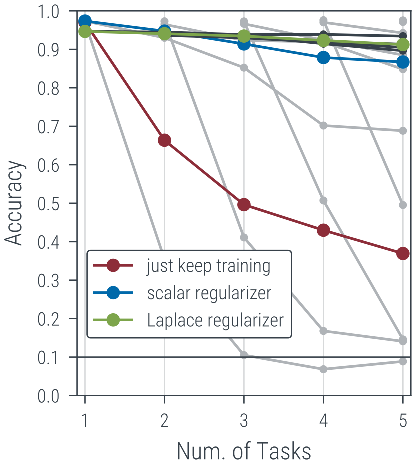
- Permuted copies of MNIST
- (784, 100, 10)-ReLU net
- Adam(lr=10^{-3}), 10 epochs each
- $\lambda = 5 \cdot 10^{-4}$

Consider a sequence of datasets $\mathcal{D}_i$, $i = 1, \dots, T$. At time i, previous datasets are no longer available. What do you do?

1. Just keep training™
2. regularizer $r(\boldsymbol{\theta}) = \frac{\lambda}{2}\|\boldsymbol{\theta} - \boldsymbol{\theta}_{i-1}\|^2$
3. Laplace posterior $r(\boldsymbol{\theta}) = \frac{\lambda}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}_{i-1})^\intercal \Psi (\boldsymbol{\theta} - \boldsymbol{\theta}_{i-1})$

Setup:

- Permuted copies of MNIST
- (784, 100, 10)-ReLU net
- Adam(lr=10^{-3}), 10 epochs each
- $\lambda = 5 \cdot 10^{-4}$



Uncertainty in Deep Learning

- fixes (asymptotic and local) overconfidence
- yields the functionality for continual learning
- many other applications not discussed here

Laplace approximations turn deep networks into GPs, inheriting all functionality of GPs

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}}
```