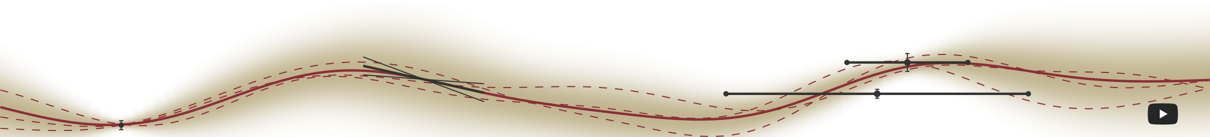
Probabilistic Machine Learning
Lecture 14
Generalized Linear Models

Philipp Hennig

08 June 2020

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

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



#	date	content	Ex	#	date	content	Ex
1	20.04.	Introduction	1	14	09.06.	Generalized Linear Models	
2	21.04.	Reasoning under Uncertainty		15	15.06.	Exponential Families	8
3	27.04.	Continuous Variables	2	16	16.06.	Graphical Models	
4	28.04.	Monte Carlo		17	22.06.	Factor Graphs	9
5	04.05.	Markov Chain Monte Carlo	3	18	23.06.	The Sum-Product Algorithm	
6	05.05.	Gaussian Distributions		19	29.06.	Example: Topic Models	10
7	11.05.	Parametric Regression	4	20	30.06.	Mixture Models	
8	12.05.	Learning Representations		21	06.07.	EM	11
9	18.05.	Gaussian Processes	5	22	07.07.	Variational Inference	
10	19.05.	Understanding Kernels		23	13.07.	Topics	
11	26.05.	Gauss-Markov Models		25	20.07.	Example: Kernel Topic Models	
12	25.05.	An Example for GP Regression	6	24	14.07.	Example: Inferringevision	
13	08.06.	GP Classification	7	26	21.07.	Revision	

$$p(y \mid f_x) = \sigma(yf_x) = \frac{1}{1+e^{-f_x}} \qquad p(f) = \mathcal{GP}(f; 0, k)$$

Today:

- Connection to the *Support Vector Machine*
- Beyond the logistic link function: *Generalized Linear Models*
- Laplace approximations for *Bayesian Deep Learning*

Training Points are not Uniformly Informative

within-class points have little influence

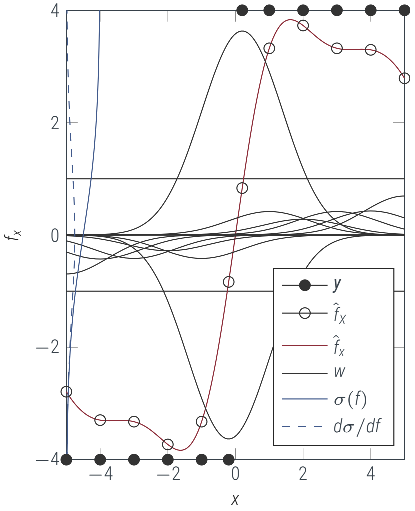
- Recall $\sigma(x) = \frac{1}{1+e^{-x}}$ with $\frac{d\sigma(x)}{dx} = \sigma(x)(1-\sigma(x))$
  Note that, at extremum

$$\nabla \log p(\boldsymbol{f}_X \mid \boldsymbol{y}) = \sum_{i=1}^{n} \nabla \log \sigma(y_i f_{x_i}) - K_{XX}^{-1}(\boldsymbol{f}_X - \boldsymbol{m}_X) = 0$$

$$K_{XX}^{-1}(\boldsymbol{f}_X - \boldsymbol{m}_X) = \sum_{i=1}^{n} \nabla \log \sigma(y_i f_{x_i}) = \nabla \log p(\boldsymbol{y} \mid \boldsymbol{f}_X) = \boldsymbol{r}$$

$$\Rightarrow \mathbb{E}_q(f_x) = m_x + k_{xX} K_{XX}^{-1}(\hat{\boldsymbol{f}}_X - \boldsymbol{m}_X) = m_x + k_{xX}\boldsymbol{r}$$

- So x_i with $|f_i| \gg 1$, where $\nabla \log p(y_i \mid f_i) \approx 0$, contribute almost nothing to $\mathbb{E}_q(f_x)$.
- The x_i with $|f_i| < 1$ are *"support points"*



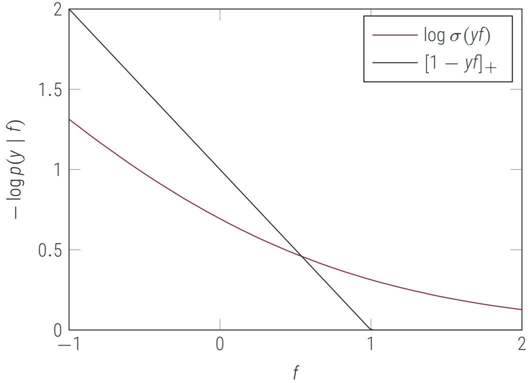
The Support Vector Machine

a powerful computational trick, with probabilistic caveats

[Boser, Guyon & Vapnik, 1992, Schölkopf et al., 1997]

$$-\log p(\boldsymbol{f}_X \mid \boldsymbol{y}) = \sum_i -\log \sigma(y_i f_i) + \|\boldsymbol{f}_X\|^2_{K_{XX}}$$
$$= \sum_i \ell(y_i; f_i) + \|\boldsymbol{f}_X\|^2_{K_{XX}}$$

- $\ell(y_i; f_i) = [1 - y_i f_i]_+$ – the **Hinge Loss**, yields the **Support Vector Machine (SVM)**



The Support Vector Machine

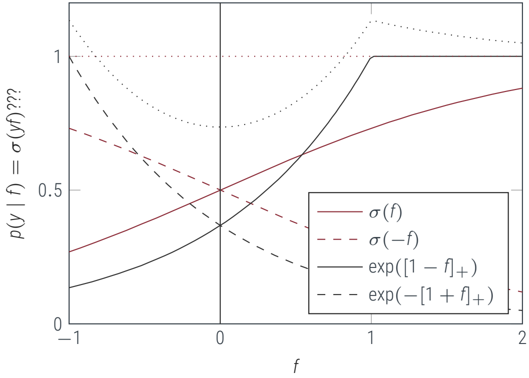
a powerful computational trick, with probabilistic caveats

[Boser, Guyon & Vapnik, 1992, Schölkopf et al., 1997]

$$
\begin{aligned}
-\log p(f_X \mid y) &= \sum_i -\log \sigma(y_i f_i) + \|f_X\|^2_{K_{XX}} \\
&= \sum_i \ell(y_i; f_i) + \|f_X\|^2_{K_{XX}}
\end{aligned}
$$

- $\ell(y_i; f_i) = [1 - y_i f_i]_+$ – the **Hinge Loss**, yields the **Support Vector Machine (SVM)**
- Unfortunately, this is not a log likelihood [Seeger, 2000]

$$
\begin{aligned}
&\exp(\ell(y_i; f_i)) + \exp(\ell(-y_i; f_i)) \\
&= \exp(\ell(f_i)) + \exp(\ell(-f_i)) \neq \text{const.}
\end{aligned}
$$



The SVM is a (further) computational simplification to logistic regression.
Unfortunately, it is not associated with (natural) predictive uncertainty.

Support Vector Machines

- arise from empirical losses that are *flat* (zero gradient) “within” the classes
- they can be thought of as a certain limit of logistic regression
- unfortunately, this particular limit loses the probabilistic interpretation
- but there are good theoretical guarantees for the point estimate!

SVMs are an example of a machine learning algorithm without a proper probabilistic interpretation. Note that, nevertheless, the probabilistic view can *help* with intuition for the statistical interpretation.

further reading: Rasmussen & Williams, 2006, §6.4.1

Multi-Class Classification

Extending the Laplace Approximation

- What if we have C class labels $[c_1, \dots, c_C]$?
- Generative Model:
  - use C **outputs** of the latent function. So at the n locations, the latent variables are

$$\boldsymbol{f}_X = [f_1^{(1)}, \dots, f_n^{(1)}, f_1^{(2)}, \dots, f_n^{(2)}, \dots, f_1^{(C)}, \dots, f_n^{(C)}]$$

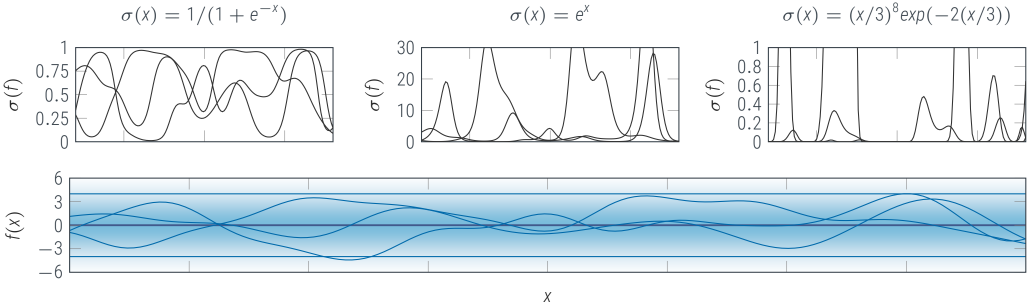
  - At location x_i, generate probabilities for each class by taking the **softmax**

$$p(y_i^{(c)} \mid \boldsymbol{f}_i) = \pi_i^{(c)} = \frac{\exp(f_i^{(c)})}{\sum_{\tilde{c}=1}^{C} \exp(f_i^{(\tilde{c})})}$$

The remaining derivations are analogous to the binary case.

Generalized Linear Models

Choose your link function



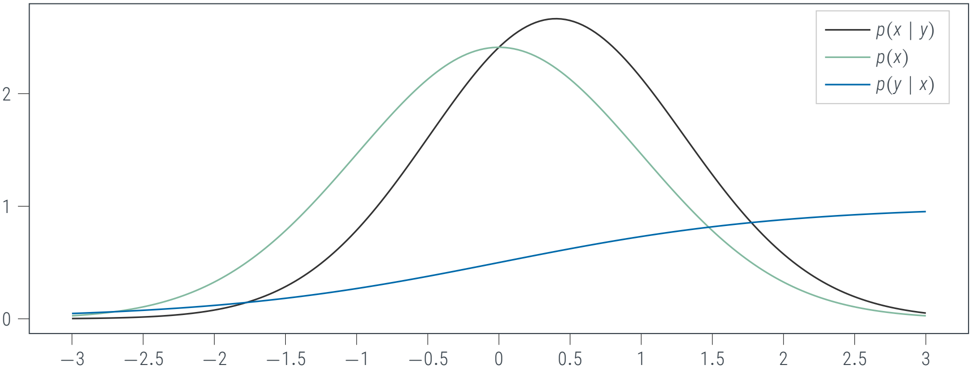
Definition (Generalized Linear Model)

(For our purposes,) a **generalized linear model (GLM)** is a probabilistic regression model for a function f with a Gaussian process prior $p(f)$ and a **non-Gaussian** likelihood $p(y \mid f_x)$. Note the distinction to a **general** linear model (GP prior and likelihood, with non-linear kernel k)

The Laplace Approximation

brief reminder

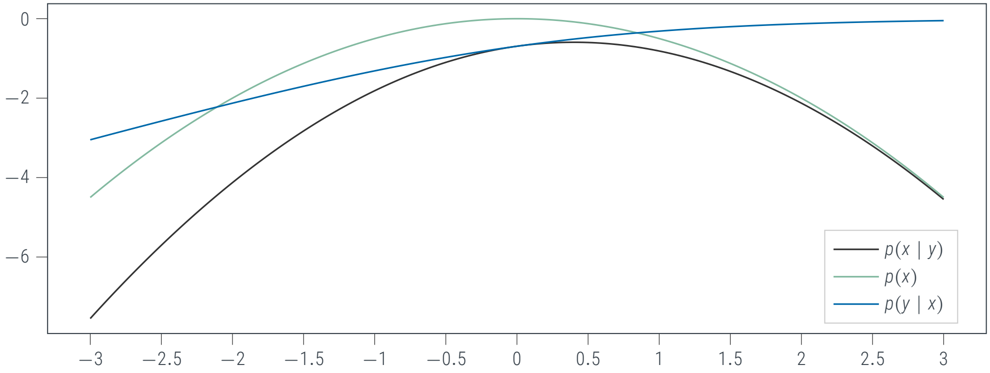
$$p(x) \approx q(x) = \mathcal{N}(x; x^* := \arg\max p, \Psi := -(\nabla\nabla \log p(x^*))^{-1})$$



The Laplace Approximation

brief reminder

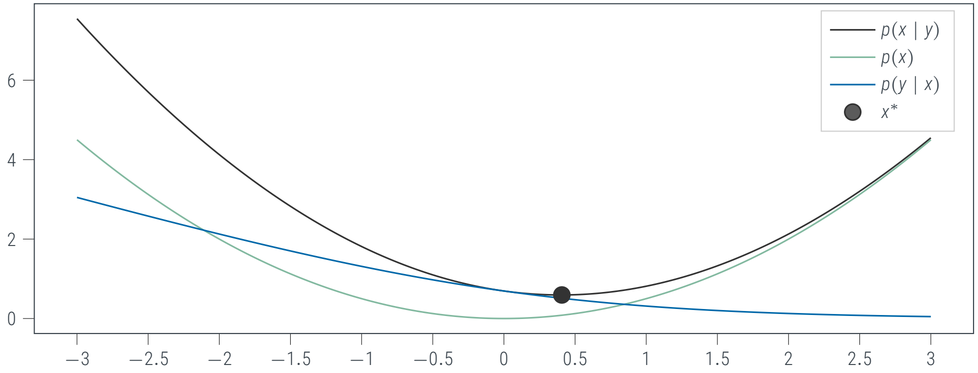
$$p(x) \approx q(x) = \mathcal{N}(x; x^* := \arg\max p, \Psi := -(\nabla\nabla \log p(x^*))^{-1})$$



The Laplace Approximation

brief reminder

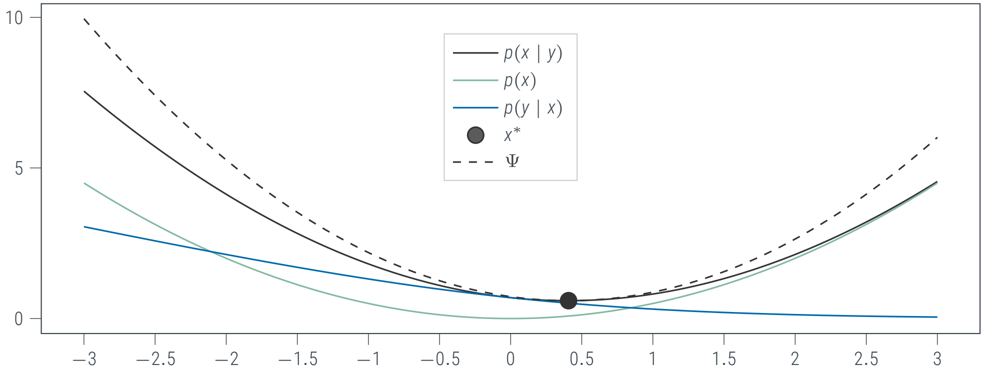
$$p(x) \approx q(x) = \mathcal{N}(x; x^* := \arg\max p, \Psi := -(\nabla\nabla \log p(x^*))^{-1})$$



The Laplace Approximation

brief reminder

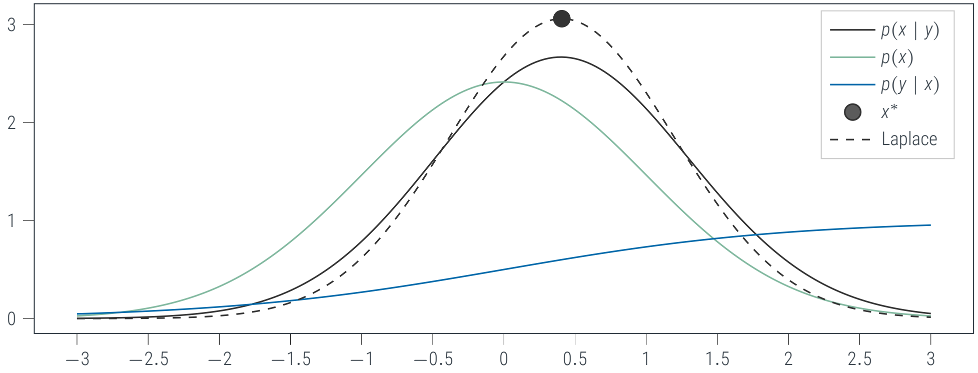
$$p(x) \approx q(x) = \mathcal{N}(x; x^* := \arg\max p, \Psi := -(\nabla\nabla \log p(x^*))^{-1})$$



The Laplace Approximation

brief reminder

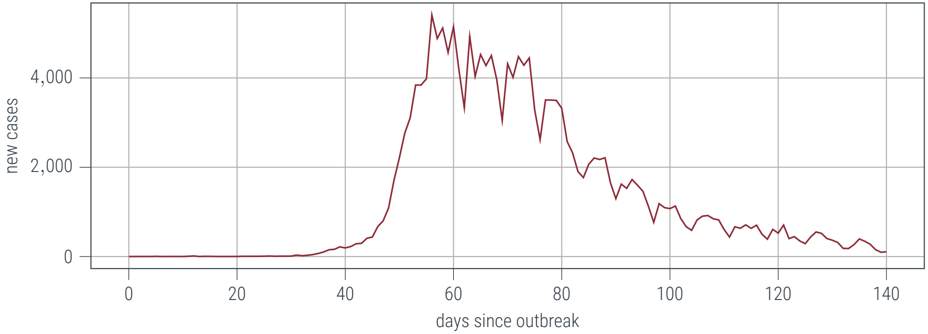
$$p(x) \approx q(x) = \mathcal{N}(x; x^* := \arg\max p, \Psi := -(\nabla\nabla \log p(x^*))^{-1})$$



A recent example

count data

data: Robert Koch Institut, 22 May 2020

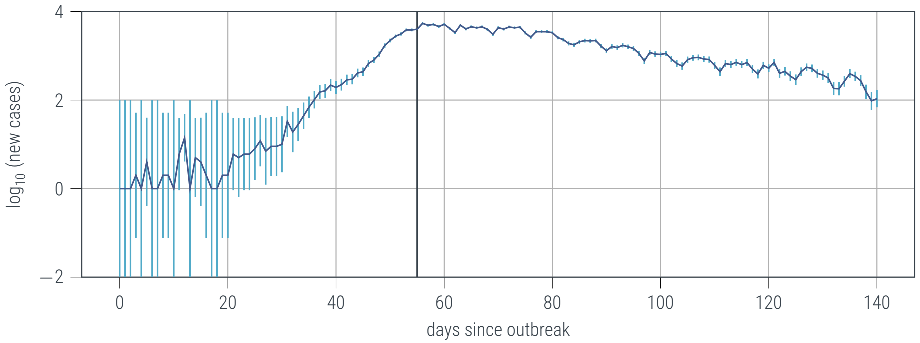


$$p(\boldsymbol{y} \mid f_T) = \mathcal{N}(\boldsymbol{y}; f_T, \sigma^2 I) \qquad p(f) = \mathcal{GP}(f; 0, k)$$

A recent example

count data

data: Robert Koch Institut, 22 May 2020



$$p(\boldsymbol{y} \mid f_T) = \mathcal{N}(\boldsymbol{y}; \exp(f_T), \sigma^2 I) \approx q(\boldsymbol{y} \mid f_T) = \mathcal{N}(\log \boldsymbol{y}; f_T, \sigma^2 \operatorname{diag}(1/\boldsymbol{y})) \quad \text{because}$$

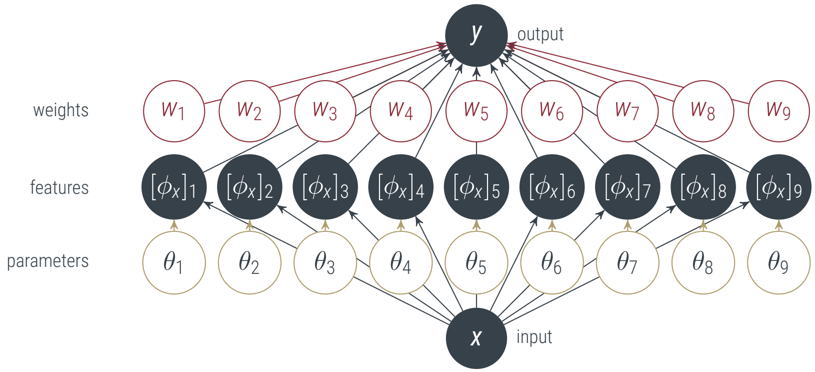
$$\left.\frac{\partial \log p(\boldsymbol{y} \mid f_T)}{\partial f_T}\right|_{f_T = \hat{f}_T} = 0 \quad \Rightarrow \quad \hat{f}_T = \log \boldsymbol{y} \qquad \text{and} \qquad \left.\frac{\partial^2 \log p(\boldsymbol{y} \mid f_T)}{\partial^2 f_T}\right|_{f_t = \hat{f}_T} = \frac{\boldsymbol{y}^2}{\sigma^2}$$

Generalized Linear Models

- extend the idea discussed for *classification* in the previous lecture to general *link functions*. That is, *non-Gaussian likelihoods* of general form.
- a simple (approximate) probabilistic version can be constructed by analogously extending the *Laplace approximation* from the previous lecture
- note that, for arbitrary link functions, the Laplace approximation may well be quite bad

How about Bayesian Deep Learning for Classification?

and what about uncertainty on the features?



y
output
weights
w_1 w_2 w_3 w_4 w_5 w_6 w_7 w_8 w_9
features
$[\phi_x]_1$ $[\phi_x]_2$ $[\phi_x]_3$ $[\phi_x]_4$ $[\phi_x]_5$ $[\phi_x]_6$ $[\phi_x]_7$ $[\phi_x]_8$ $[\phi_x]_9$
parameters
θ_1 θ_2 θ_3 θ_4 θ_5 θ_6 θ_7 θ_8 θ_9
x
input

Evidence / Marginal Likelihood

For hyperparameter adaptation

[Rasmussen & Williams, 2006, §3.4]

$$p(\boldsymbol{y} \mid X) = \int p(\boldsymbol{y}, \boldsymbol{f} \mid X)\, d\boldsymbol{f} = \int \exp\left(\log p(\boldsymbol{y} \mid \boldsymbol{f}) p(\boldsymbol{f} \mid X)\right)\, d\boldsymbol{f}$$

Laplace: $$\log p(\boldsymbol{y} \mid \boldsymbol{f}) p(\boldsymbol{f} \mid X) \approx \log\left(p(\boldsymbol{y} \mid \hat{f}) p(\hat{f} \mid X)\right) - \frac{1}{2}(\boldsymbol{f} - \hat{f})^\intercal (K^{-1} + W)(\boldsymbol{f} - \hat{f}) = \log q(\boldsymbol{y}, \boldsymbol{f} \mid X)$$

thus $$p(\boldsymbol{y} \mid X) \approx q(\boldsymbol{y} \mid X) = \exp\left(\log\left(p(\boldsymbol{y} \mid \hat{f}) p(\hat{f} \mid X)\right)\right) \int \exp\left(-\frac{1}{2}(\boldsymbol{f} - \hat{f})^\intercal (K^{-1} + W)(\boldsymbol{f} - \hat{f})\right) d\boldsymbol{f}$$

$$= \exp\left(\log\left(p(\boldsymbol{y} \mid \hat{f})\right)\right) \mathcal{N}(\hat{f}; m_X, k_{XX}) (2\pi)^{n/2} |(K^{-1} + W)^{-1}|^{1/2}$$

$$\log q(y \mid X) = \log p(\boldsymbol{y} \mid \boldsymbol{f}) - \frac{1}{2}(\hat{f} - m_X)^\intercal K_{XX}^{-1} (\hat{f} - m_X) - \frac{1}{2} \log(|K| \cdot |K^{-1} + W|)$$

$$= \sum_{i=1}^{n} \sigma(y_i f_{x_i}) - \frac{1}{2}(\hat{f} - m_X)^\intercal K_{XX}^{-1} (\hat{f} - m_X) - \frac{1}{2} \log |B|$$

Evidence / Marginal Likelihood

For hyperparameter adaptation

[Rasmussen & Williams, 2006, §3.4]

```
procedure GP-Logistic-Train($K_{XX}, m_X, \boldsymbol{y}$)
    $f \leftarrow m_X$                                                     // initialize
    while not converged do
        $r \leftarrow \frac{y+1}{2} - \sigma(f)$                           // $= \nabla \log p(\boldsymbol{y} \mid f_X)$, gradient of log likelihood
        $W \leftarrow \operatorname{diag}(\sigma(f) \odot (1 - \sigma(f)))$  // $= -\nabla\nabla \log p(\boldsymbol{y} \mid f_X)$, Hessian of log likelihood
        $R \leftarrow \textsc{Cholesky}(I + W^{1/2} K_{XX} W^{1/2})$       // $B$ from previous slide
        $b \leftarrow W(f - m_X) + r$
        $a \leftarrow b - W^{1/2} R^{-\intercal} R^{-1} (W^{1/2} K b)$     // $a = K^{-1}(K^{-1} + W)^{-1} b$
        $f \leftarrow Ka + m_X$   // $f \leftarrow (K^{-1} + W)^{-1}((K^{-1} + W)(f - m_X) - K^{-1}(f - m_X) + r + m_X = f + (W + K^{-1})^{-1} g$
    end while                                                              // The objective is $-\frac{1}{2} a^\intercal (f - m_X) + \sum_i \log \sigma(y_i f_i)$
    $\log q(y \mid X) \leftarrow -1/2 a^\intercal (f - m_X) + \sum_i \log \sigma(y_i f_i) - \sum_i \log R_{ii}$   // (Cholesky $B = R^\intercal R \Rightarrow |B| = \prod_i R_{ii}^2$)
    return $f, W, R, r$
end procedure
```

Parametric Form — Last layer Laplace approximation

Towards Bayesian Deep Learning

$$p(v) = \mathcal{N}(v, \mu, \Sigma) \qquad p(\boldsymbol{y} \mid \boldsymbol{f}_X) = \prod_{i=1}^{n} \sigma(y_i f_{x_i}) \quad \text{with } v \in \mathbb{R}^F, \phi_X \in \mathbb{R}^{n \times F}$$

$$\log p(v \mid \boldsymbol{y}) = \log p(\boldsymbol{y} \mid v) + \log p(v) - \log p(\boldsymbol{y})$$

$$= \sum_{i=1}^{n} \log \sigma(y_i \phi_{x_i}^\intercal v) - \frac{1}{2}(v - \boldsymbol{\mu})^\intercal \Sigma^{-1}(v - \boldsymbol{\mu}) + \text{const.}$$

$$\nabla \log p(v \mid \boldsymbol{y}) = \sum_{i=1}^{n} \nabla \log \sigma(y_i \phi_{x_i}^\intercal v) - \Sigma^{-1}(v - \boldsymbol{\mu}) \quad \text{with} \quad \frac{\partial \log \sigma(y_i \phi_{x_i}^\intercal v)}{\partial v_j} = [\phi_{x_i}]_j \left(\frac{y_i + 1}{2} - \sigma(\phi_{x_i}^\intercal v) \right)$$

$$\nabla \nabla^\intercal \log p(v \mid \boldsymbol{y}) = \sum_{i=1}^{n} \nabla \nabla^\intercal \log \sigma(y_i \phi_{x_i}^\intercal v) - \Sigma^{-1} \quad \text{with} \quad \frac{\partial^2 \log \sigma(y_i \phi_{x_i}^\intercal v)}{\partial v_a \partial v_b} = -[\phi_{x_i}]_a [\phi_{x_i}]_b \underbrace{\sigma(\phi_{x_i}^\intercal v)(1 - \sigma(\phi}_{=: w_i}$$

$$=: -(W + \Sigma^{-1}) = -\left(\phi_X \operatorname{diag}(\boldsymbol{w}) \phi_X^\intercal + \Sigma^{-1}\right) \in \mathbb{R}^{F \times F}$$

still convex minimization / concave maximization! All computations $\mathcal{O}(F^2 n)$

Deep Learning

At least, one kind of it

- consider a *deep (feedforward) neural network*

$$p(y_i \mid W) = \prod_{i=1}^{n} \sigma(f_W(\mathbf{x}_i)) \qquad f_W(x) = \mathbf{w}_L^\intercal \phi(\mathbf{w}_{L-1}\phi(\dots(\mathbf{w}_1\mathbf{x})\dots))$$

- standard deep learning amounts to ("type-I") **maximum a-posteriori** estimation

$$W^* = \arg\max_W p(W \mid \mathbf{y}) = \arg\min_W - \sum_{i-1}^{n} \log \sigma(f_W(x_i)) - \log p(W)$$

$$= \arg\min_W - \sum_{i-1}^{n} \log \sigma(f_W(x_i)) - \beta^2 \|W\|^2 =: \arg\min_W J(W)$$

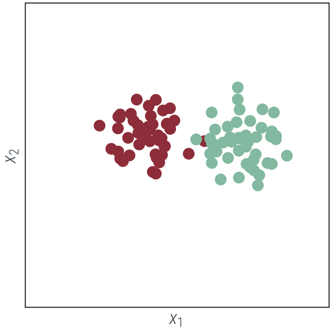
A problem with Deep Learning

overconfidence

[Hein et al., 2019]. Figure by Agustinus Kristiadi

Theorem (Hein et al. 2019)

Let $\mathbb{R}^d = \cup_{r=1}^{R} Q_r$ *and* $f|_{Q_r}(\boldsymbol{x}) = \boldsymbol{U}_r\boldsymbol{x} + \boldsymbol{c}_r$ *be the piecewise affine representation of the output of a ReLU network on* Q_r*. Suppose that* $\boldsymbol{U}_r$ *does not contain identical rows for all* $r = 1, \dots, R$*, then for almost any* $\boldsymbol{x} \in \mathbb{R}^n$ *and any* $\epsilon > 0$*, there exists a* $\delta > 0$ *and a class* $i \in \{1, \dots, k\}$ *such that it holds* $\operatorname{softmax}(f(\delta\boldsymbol{x}), i) \geq 1 - \epsilon$*. Moreover,* $\lim_{\delta \to \infty} \operatorname{softmax}(f(\delta\boldsymbol{x}), i) = 1$.



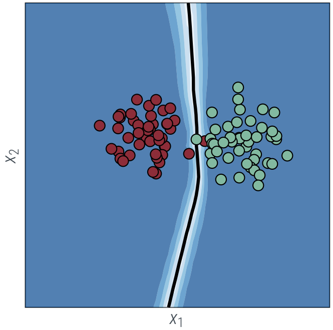
A problem with Deep Learning

overconfidence

[Hein et al., 2019]. Figure by Agustinus Kristiadi

Theorem (Hein et al. 2019)

Let $\mathbb{R}^d = \cup_{r=1}^R Q_r$ *and* $f|_{Q_r}(\boldsymbol{x}) = \boldsymbol{U}_r\boldsymbol{x} + \boldsymbol{c}_r$ *be the piecewise affine representation of the output of a ReLU network on* Q_r*. Suppose that* $\boldsymbol{U}_r$ *does not contain identical rows for all* $r = 1, \dots, R$*, then for almost any* $\boldsymbol{x} \in \mathbb{R}^n$ *and any* $\epsilon > 0$*, there exists a* $\delta > 0$ *and a class* $i \in \{1, \dots, k\}$ *such that it holds* $\operatorname{softmax}(f(\delta \boldsymbol{x}), i) \geq 1 - \epsilon$*. Moreover,* $\lim_{\delta \to \infty} \operatorname{softmax}(f(\delta \boldsymbol{x}), i) = 1$.



Why be Bayesian in Deep Learning?

Just one of many possible motivations

- A strong point estimate doesn't matter if it's uncertain
- replace $p(y = 1 \mid x) = \sigma(f_W(x))$ with the *marginal*

$$p(y = 1 \mid x) = \int \sigma(f_W(x))\, p(W \mid \mathbf{y})\, dW$$

- approximate posterior on W by Laplace as

$$p(W \mid \mathbf{y}) \approx \mathcal{N}(W; W^*, -(\nabla\nabla^\intercal J(W))^{-1}) =: \mathcal{N}(W; W^*, \Psi)$$

- and on f by linearizing with $G(x) = \frac{df_{W*}(x)}{dW}$ as $f_W(x) \approx f_{W*}(x) + G(x)(W - W^*)$, thus

$$p(f_W(x)) = \int p(f \mid W)p(W)\, dW \approx \mathcal{N}(f(x); f_{W*}(x), G(x)\Psi G(x)^\intercal) =: \mathcal{N}(f(x); m(x), v(x))$$

- and approximate the marginal (MacKay, 1992) as

$$p(y = 1 \mid x) \approx \sigma\left(\frac{m(x)}{\sqrt{1 + \pi/8\, v(x)}}\right).$$

Problem solved (asymptotically)!

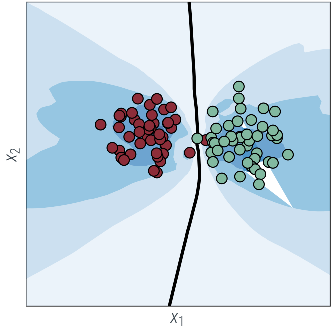
Kristiadi et al. 2020 `https://arxiv.org/abs/2002.10118`

Theorem (Kristiadi et al., 2020)

Let $f_W : \mathbb{R}^n \to \mathbb{R}$ be a binary ReLU classification network parametrized by $W \in \mathbb{R}^p$ with $p \geq n$, and let $\mathcal{N}(W|W^*, \Psi)$ be the approximate posterior. Then for any input $\mathbf{x} \in \mathbb{R}^n$, there exists an $\alpha > 0$ such that for any $\delta \geq \alpha$, the confidence $\sigma(|z(\delta\mathbf{x})|)$ is bounded from above by the limit $\lim_{\delta \to \infty} \sigma(|z(\delta\mathbf{x})|)$. Furthermore,

$$\lim_{\delta \to \infty} \sigma(|z(\delta\mathbf{x})|) \leq \sigma\left(\frac{|\mathbf{u}|}{s_{min}(\mathbf{J})\sqrt{\pi/8\,\lambda_{min}(\Psi)}}\right),$$

where $\mathbf{u} \in \mathbb{R}^n$ is a vector depending only on W and the $n \times p$ matrix $\mathbf{J} := \frac{\partial \mathbf{u}}{\partial W}\Big|_{W^*}$ is the Jacobian of $\mathbf{u}$ w.r.t. W at W^*.



Isn't Bayesian Deep Learning Costly?

Not necessarily, if you're willing to approximate

Not if you use

- a low-rank approximation of the Hessian
- a block-diagonal approximation of the Hessian
- the Hessian of the last layer
- or even just the diagonal of the Hessian

Code example

`BNN_Laplace.ipynb`

Using **backpack for pytorch**, a collection of lightweight extensions for second order quantities (curvature and variance) available at

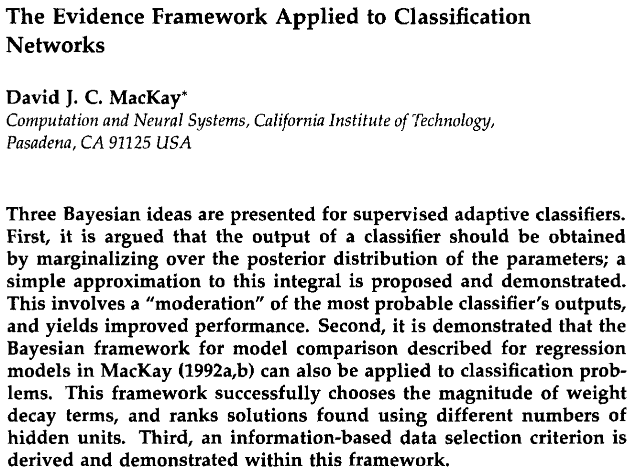
`http://backpack.pt`

F. Dangel, F. Künstner, P. Hennig
BackPACK: Packing more into Backprop
ICLR 2020

Bayesian Deep Learning via Laplace

not a new idea, but still a good one

MacKay, 1992



The Evidence Framework Applied to Classification Networks

David J. C. MacKay*
Computation and Neural Systems, California Institute of Technology, Pasadena, CA 91125 USA

Three Bayesian ideas are presented for supervised adaptive classifiers. First, it is argued that the output of a classifier should be obtained by marginalizing over the posterior distribution of the parameters; a simple approximation to this integral is proposed and demonstrated. This involves a "moderation" of the most probable classifier's outputs, and yields improved performance. Second, it is demonstrated that the Bayesian framework for model comparison described for regression models in MacKay (1992a,b) can also be applied to classification problems. This framework successfully chooses the magnitude of weight decay terms, and ranks solutions found using different numbers of hidden units. Third, an information-based data selection criterion is derived and demonstrated within this framework.

David JC MacKay
1967–2016

— Summary —

Support Vector Machines

- arise if the empirical risk has zero gradient for large values of f ($\rightharpoonup$ hinge-loss)
- unfortunately, this does not amount to a log likelihood, so there is no natural probabilistic interpretation (and thus uncertainty) for the SVM

Generalized Linear Models

- extend Gaussian (process) regression to non-Gaussian likelihoods
- the Laplace approximation yields a computationally lightweight approximate posterior for such models. It is better than a point-estimate, but one has to take care to ensure it is working, especially if the likelihood is not log-concave

Bayesian Deep Learning

- deep neural networks can have badly calibrated uncertainty when used as (MAP) point estimates
- Laplace approximations can fix this issue
- Laplace approximations are not for free, but feasible for many deep models, and easy to implement