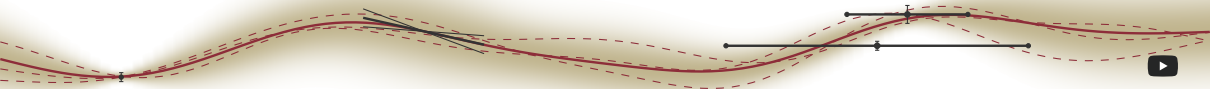
Probabilistic Machine Learning
Lecture 14
Logistic Regression

Philipp Hennig

19 June 2023

Eberhard Karls
Universität
Tübingen

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



Algorithm 1 Iterative GP regression (Numerics Layer)

Input: sufficient statistics $K = k_{XX} + \sigma^2 I, \bar{y} = y - \mu_X$, initial guesses α_0, C_0
Output: defragmented statistics S, C_i, α_i

```
procedure TRAIN(K, y, C_0 = 0, α_0 = 0)
    for i ∈ {1, ..., n} do
        s_i ⇐ POLICY(S = [s_{j<i}], Z = [z_{j<i}])        // Action – load, s_i ∈ ℝ^{N×k_i}
        z_i ⇐ K s_i                                       // Observation – compute, z_i ∈ ℝ^{N×k_i}
        d_i ⇐ (I − C_{i−1} K) s_i = s_i − C_{i−1} z_i     // low-rank update, d_i ∈ ℝ^{N×k_i}
        H_i ⇐ s_i^T K d_i = z_i^T d_i                     // Schur complement, H_i ∈ ℝ^{k_i×k_i}
        C_i ⇐ C_{i−1} + d_i H_i^{−1} d_i^T                // Inverse estimate
        α_i ⇐ C_i y = α_{i−1} + d_i H_i^{−1} d_i^T ȳ      // Solution Estimate
    end for
    return S = [s_j]_{j≤i}, α_i, C_i
end procedure
procedure PREDICT(x, S, α, C)
    k_xS ⇐ k[x, S]                                        // Covariance to Observations
    μ_x ⇐ μ_x + k_xS α                                    // Point estimate
    v_xx ⇐ k_xx − k_xS C k_Sx                             // Uncertainty
end procedure
```


Computation and Inference

There is really no difference

- In least-squares regression, "training" the algorithm reduces to a *linear algebra* computation
- We have now seen that the corresponding *numerical methods* essentially reduce to smartly loading *projections* of the data in the right order
  - $s_i = e_{j(i)}$ yields the **pivoted** Cholesky decomposition, where $j(i)$ is the pivoting policy
  - the Lanczos process, initialized with $s_0 = \boldsymbol{K}\alpha_0 - \boldsymbol{C}_0\boldsymbol{y}$, yields the **preconditioned** conjugate gradient method, where $\boldsymbol{C}_0$ is the preconditioner.
  - We can think of $\alpha_0, \boldsymbol{C}_0$ as a **prior guess** for $\alpha, \boldsymbol{K}^{-1}$. Then our algorithm computes updated estimates of $\boldsymbol{K}^{-1}$.
- Side note: It turns out there are Bayesian interpretations of these point estimates with associated uncertainty for α and $\boldsymbol{K}^{-1}$ [Hennig, 2015, Wenger & Hennig, 2021, Hennig, Osborne, Kersting, 2022]

Relationship to Gradient Descent

connection to previous lecture

Lecture 12: Jacobian/Sensitivity of minimizer

$$\boldsymbol{\alpha}^\star = \arg\min_{\alpha \in \mathbb{R}^N} f(\boldsymbol{\alpha}) = \arg\min_{\alpha \in \mathbb{R}^N} \frac{1}{2}\boldsymbol{\alpha}^\intercal (k_{XX} + \sigma^2 I_N)\boldsymbol{\alpha} - \underbrace{(\boldsymbol{y} - \mu_X)}_{=\tilde{\boldsymbol{y}}}{}^\intercal \boldsymbol{\alpha}$$

$$= (k_{XX} + \sigma^2 I_N)^{-1}(\boldsymbol{y} - \mu_X)$$

$$\Rightarrow \qquad \frac{\mathrm{d}\boldsymbol{\alpha}^\star}{\mathrm{d}\boldsymbol{y}} = (k_{XX} + \sigma^2 I_N)^{-1}$$

Compare to gradient descent:

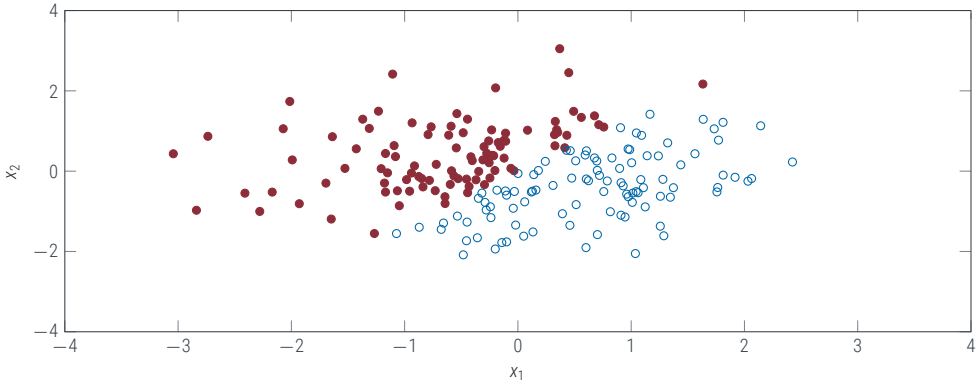
$$\frac{\partial \boldsymbol{\alpha}^*}{\partial \boldsymbol{y}} = \frac{\partial \nabla_\alpha f(\alpha)}{\partial \boldsymbol{y}} = \eta I$$

Cholesky iteratively computes a *total* derivative (a matrix inverse), while GD computes a *partial* derivative. This is why gradient descent does not converge in finite time. We pay for reducing complexity to $\mathcal{O}(N)$ and below, by losing convergence, and uncertainty.

Classification

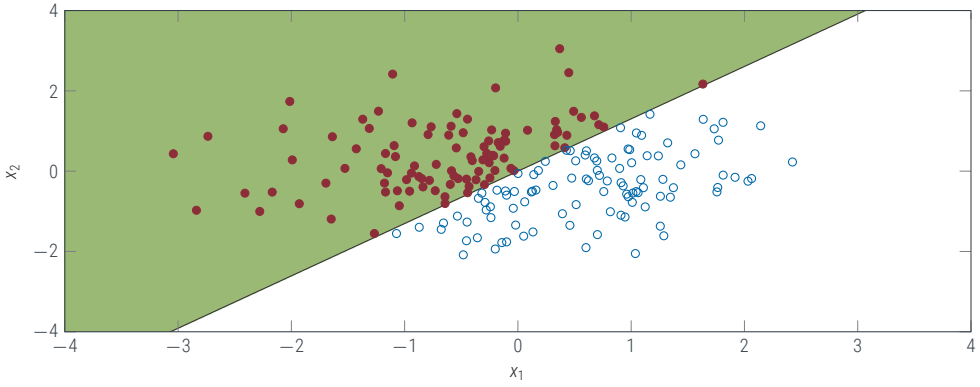
Classification Problems

a typography of supervised learning problems



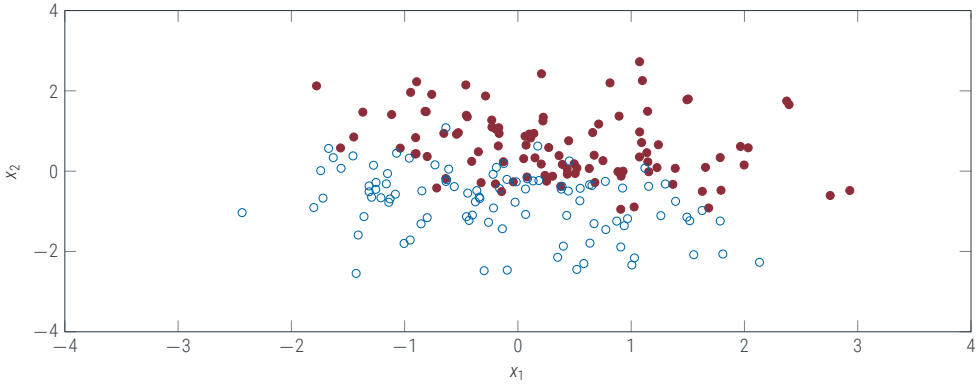
Classification Problems

a typography of supervised learning problems



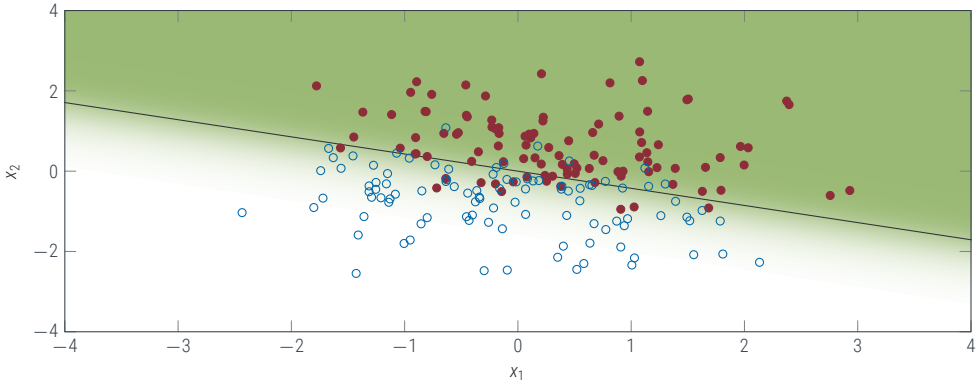
Classification Problems

a typography of supervised learning problems



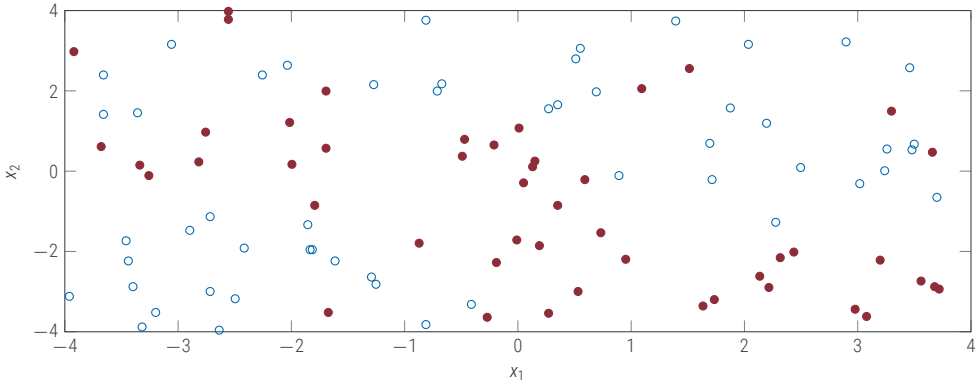
Classification Problems

a typography of supervised learning problems



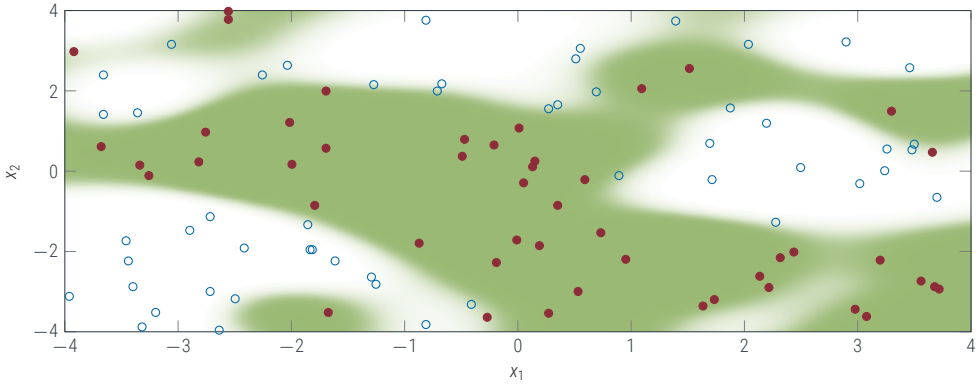
Classification Problems

a typography of supervised learning problems



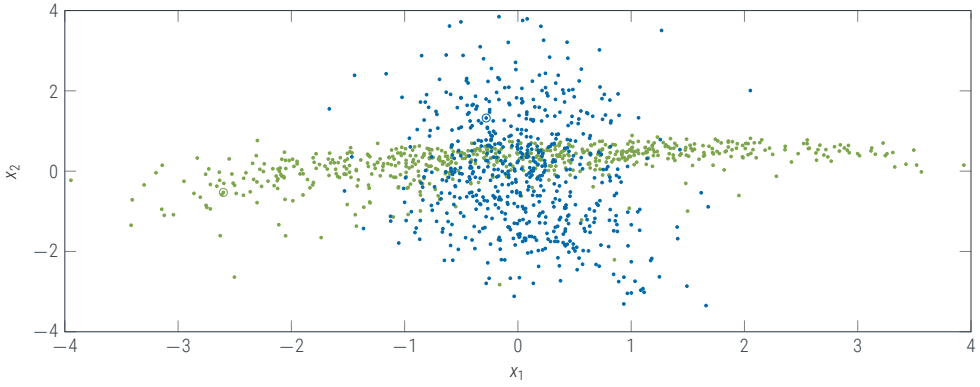
Classification Problems

a typography of supervised learning problems



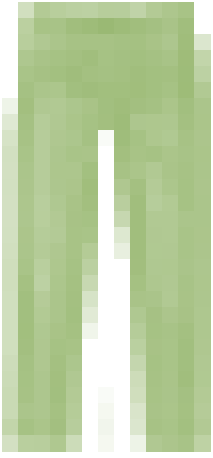
Classification Problems

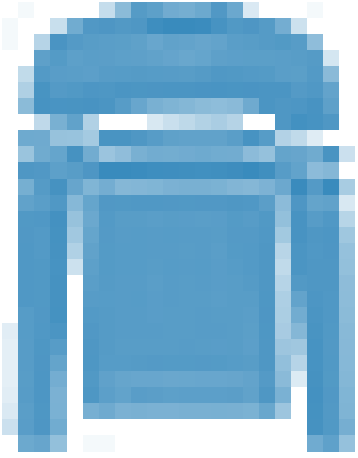
a typography of supervised learning problems



Classification Problems

a typography of supervised learning problems





https://github.com/zalandoresearch/fashion-mnist

Classification vs. Regression

Two types of supervised learning problems

Regression:

Given supervised *data* (special case $d = 1$: univariate regression)

$$(X, Y) := (x_i, y_i)_{i=1,\ldots,n} \text{ with } x_i \in \mathbb{X}, y_i \in \mathbb{R}^d$$

find function $f : \mathbb{X} \to \mathbb{R}^d$ such that f "models" $Y \approx f(X)$.

Classification:

Given supervised *data* (special case $d = 2$: binary classification)

$$(X, Y) := (x_i, c_i)_{i=1,\ldots,n} \text{ with } x_i \in \mathbb{X}, c_i \in \{1, \ldots, d\}$$

find probability $\pi : \mathbb{X} \to U^d$ ($U^d = \{p \in [0,1]^d : \sum_{i=1}^d p_i = 1\}$) such that π "models" $y_i \sim \pi_{x_i}$.

Regression predicts a **function**, *classification* predicts a **probability**.

Until further notice, consider only discriminative binary classification:

$$y \in \{-1; +1\} \qquad x \mapsto \pi(x) =: \pi_x \in [0, 1]$$

$$p(y \mid x) = \begin{cases} \pi(x) & \text{if } y = 1 \\ 1 - \pi(x) & \text{if } y = -1 \end{cases}$$

Until further notice, consider only discriminative binary classification:

$$y \in \{-1; +1\} \qquad x \mapsto \pi(x) =: \pi_x \in [0, 1]$$

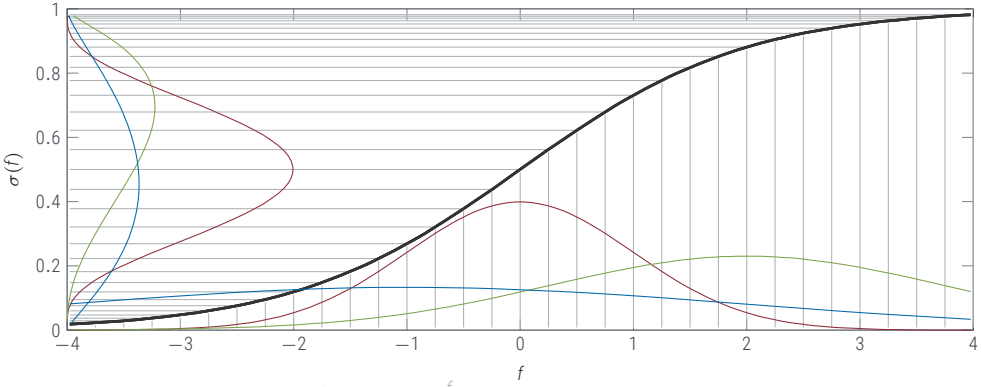
$$p(y \mid x) = \begin{cases} \pi(x) & \text{if } y = 1 \\ 1 - \pi(x) & \text{if } y = -1 \end{cases}$$

Discriminative learning phrased probabilistically:

- We would like to *learn* $\pi_x(y) = p(y \mid x)$
- This is *almost* like regression: $p(y \mid x) = \mathcal{N}(y; f_x, \sigma^2) = \pi_x(y)$
- only the *domain* is wrong: $y \in \{-1; 1\}$ vs. $y \in \mathbb{R}$.

Link Functions

sigmoids — the **logistic** link function



$$\pi_f = \sigma(f) = \frac{1}{1+\exp(-f)} = \int_{-\infty}^{f} \frac{1}{4}\,\mathrm{sech}^2\left(\frac{x}{2}\right)\,dx$$

$$\sigma(f) = 1 - \sigma(-f) \qquad f(\pi) = \ln \pi - \ln(1-\pi) \qquad \frac{d\pi}{df} = \pi(f)\cdot(1-\pi(f))$$

DEMO

- `git clone https://github.com/philipphennig/ProbML_Apps.git`
- `cd ProbML_Apps/14`
- `pip install -r requirements.txt`
- `streamlit run Lecture_14.py`

A Gaussian Process model for Classification

Logistic Regression

$$p(f) = \mathcal{GP}(f; m, k)$$

$$p(y \mid f_x) = \sigma(y f_x) = \begin{cases} \sigma(f) & \text{if } y = 1 \\ 1 - \sigma(f) & \text{if } y = -1 \end{cases} \qquad \text{using } \sigma(x) = 1 - \sigma(-x).$$

A Gaussian Process model for Classification

Logistic Regression

$$p(f) = \mathcal{GP}(f; m, k)$$

$$p(y \mid f_x) = \sigma(y f_x) = \begin{cases} \sigma(f) & \text{if } y = 1 \\ 1 - \sigma(f) & \text{if } y = -1 \end{cases} \qquad \text{using } \sigma(x) = 1 - \sigma(-x).$$

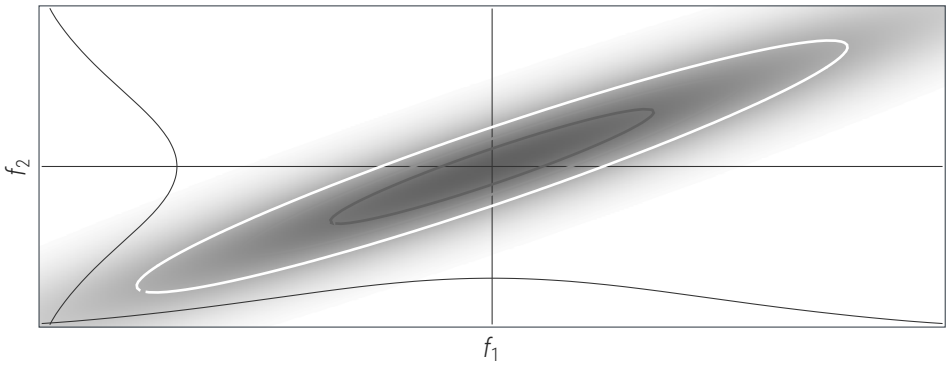
The problem: The posterior is not Gaussian!

$$p(f_X \mid Y) = \frac{p(Y \mid f_X) p(f_X)}{p(Y)} = \frac{\mathcal{N}(f_X; m, k) \prod_{i=1}^n \sigma(y_i f_{x_i})}{\int \mathcal{N}(f_X; m, k) \prod_{i=1}^n \sigma(y_i f_{x_i}) \, df_X}$$

$$\log p(f_X \mid Y) = -\frac{1}{2} f_X^\intercal k_{XX}^{-1} f_X + \sum_{i=1}^n \log \sigma(y_i f_{x_i}) + \text{const.}$$

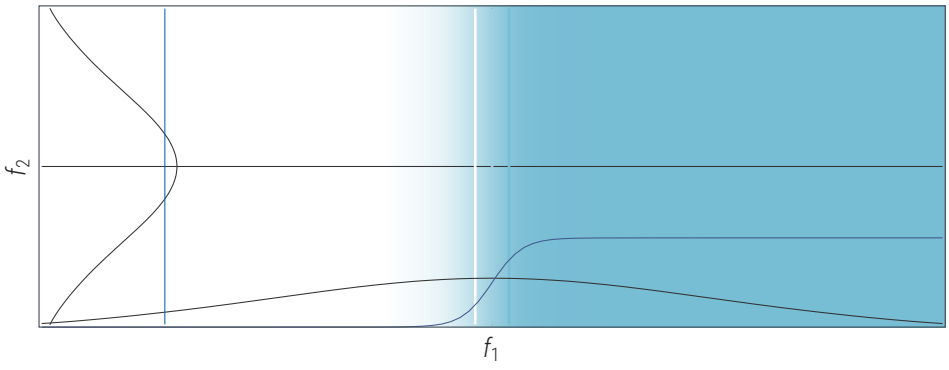
Logistic Regression is not analytic

We'll have to break out the toolbox



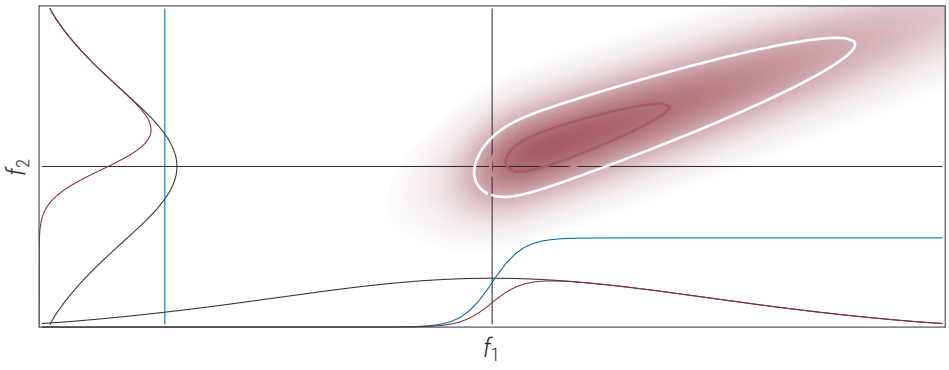
Logistic Regression is not analytic

We'll have to break out the toolbox



Logistic Regression is not analytic

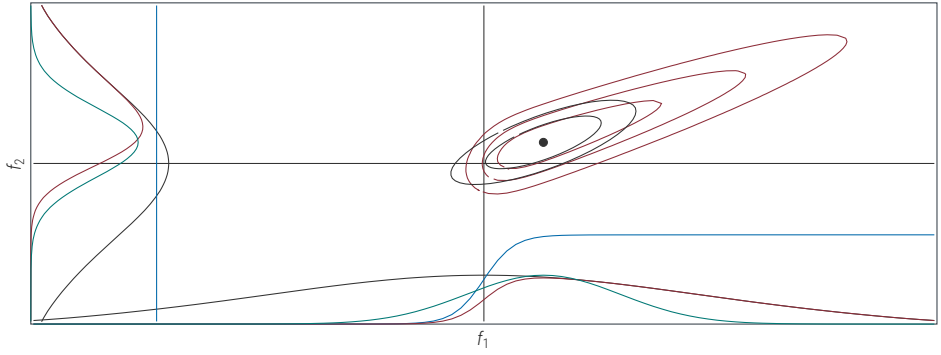
We'll have to break out the toolbox



The Laplace Approximation

A local Gaussian approximation

Pierre Simon M. de Laplace, 1814



The Laplace Approximation

recap

- Consider a probability distribution $p(\theta)$ (may be a posterior $p(\theta \mid D)$ or something else)
- find a (local) **maximum** of $p(\theta)$ or (equivalently) $\log p(\theta)$

$$\hat{\theta} = \arg\max \log p(\theta) \qquad \Rightarrow \qquad \nabla \log p(\hat{\theta}) = 0$$

- perform **second order Taylor expansion** around $\theta = \hat{\theta} + \delta$ in log space

$$\log p(\delta) = \log p(\hat{\theta}) + \frac{1}{2}\delta^\intercal \left(\underbrace{\nabla\nabla^\intercal \log p(\hat{\theta})}_{=:\Psi} \right) \delta + \mathcal{O}(\delta^3)$$

- define **the Laplace approximation** q **to** p

$$q(\theta) = \mathcal{N}(\theta; \hat{\theta}, -\Psi^{-1})$$

The Laplace Approximation for GP Classification

conceptual step (implementation details coming up)

[based on Rasmussen & Williams, 2006, §3.4]

- Find maximum posterior probability for **latent** f at **training points**

$$\hat{f} = \arg\max \log p(\mathbf{f}_X \mid y)$$

- Assign approximate Gaussian posterior at training points

$$q(\mathbf{f}_X) = \mathcal{N}(\mathbf{f}_X; \hat{\mathbf{f}}, -(\nabla\nabla^\intercal \log p(\mathbf{f}_X \mid y)|_{\mathbf{f}_X=\hat{\mathbf{f}}})^{-1}) =: \mathcal{N}(\mathbf{f}_X; \hat{\mathbf{f}}, \hat{\Sigma})$$

- approximate posterior **predictions** at f_x for **latent function**

$$\begin{aligned} q(f_x \mid y) &= \int p(f_x \mid \mathbf{f}_X) q(\mathbf{f}_X)\, d\mathbf{f}_X = \int \mathcal{N}(f_x; m_x + k_{xX}K_{XX}^{-1}(\mathbf{f}_X - m_X), k_{xx} - k_{xX}K_{XX}^{-1}k_{Xx}) q(\mathbf{f}_X)\, d\mathbf{f}_X \\ &= \mathcal{N}(f_x;\ m_x + k_{xX}K_{XX}^{-1}(\hat{\mathbf{f}} - m_X),\ k_{xx} - k_{xX}K_{XX}^{-1}k_{Xx} + k_{xX}K_{XX}^{-1}\hat{\Sigma}K_{XX}^{-1}k_{Xx}) \end{aligned}$$

Compare with *exact* predictions

$$\mathbb{E}_{p(f_x, f_X \mid y)}(f_x) = \int (\mathbb{E}_{p(f_x \mid f_X)}(f_x)) p(f_X \mid y)\, df_X = m_x + k_{xX}K_{XX}^{-1}(\mathbb{E}_{p(f_X \mid y)}(f_X) - m_X) =: \bar{f}_x$$

Recall: $p(x) = \mathcal{N}(x; m, V), p(z \mid x) = \mathcal{N}(z; Ax, B) \ \Rightarrow\ p(z) = \int p(z \mid x) p(x)\, dx = \mathcal{N}(z; Am, AVA^\intercal + B)$.

The Laplace Approximation for GP Classification

conceptual step (implementation details coming up)

[based on Rasmussen & Williams, 2006, §3.4]

- Find maximum posterior probability for **latent** f at **training points**

$$\hat{f} = \arg\max \log p(f_X \mid y)$$

- Assign approximate Gaussian posterior at training points

$$q(f_X) = \mathcal{N}(f_X; \hat{f}, -(\nabla\nabla^\intercal \log p(f_X \mid y)|_{f_X=\hat{f}})^{-1}) =: \mathcal{N}(f_X; \hat{f}, \hat{\Sigma})$$

- approximate posterior **predictions** at f_x for **latent function**

$$\begin{aligned} q(f_x \mid y) &= \int p(f_x \mid f_X) q(f_X)\, df_X = \int \mathcal{N}(f_x; m_x + k_{xX}K_{XX}^{-1}(f_X - m_X), k_{xx} - k_{xX}K_{XX}^{-1}k_{Xx}) q(f_X)\, df_X \\ &= \mathcal{N}(f_x;\ m_x + k_{xX}K_{XX}^{-1}(\hat{f} - m_X),\ k_{xx} - k_{xX}K_{XX}^{-1}k_{Xx} + k_{xX}K_{XX}^{-1}\hat{\Sigma}K_{XX}^{-1}k_{Xx}) \end{aligned}$$

Compare with *exact* predictions

$$\mathrm{var}_{p(f_x, f_X \mid y)}(f_x) = \int (f_x - \bar{f}_x)^2\, dp(f_x \mid f_X)\, dp(f_X) = k_{xx} - k_{xX}K_{XX}^{-1}k_{Xx} + k_{xX}K_{XX}^{-1}\mathrm{var}_{p(f_X \mid y)}(f_X)K_{XX}^{-1}k_{Xx})$$

Recall: $p(x) = \mathcal{N}(x; m, V), p(z \mid x) = \mathcal{N}(z; Ax, B) \ \Rightarrow\ p(z) = \int p(z \mid x)p(x)\, dx = \mathcal{N}(z; Am, AVA^\intercal + B)$.

The Laplace Approximation for GP Classification

conceptual step (implementation details coming up)

[based on Rasmussen & Williams, 2006, §3.4]

- Find maximum posterior probability for **latent** f at **training points**

$$\hat{f} = \arg\max \log p(\mathbf{f}_X \mid y)$$

- Assign approximate Gaussian posterior at training points

$$q(\mathbf{f}_X) = \mathcal{N}(\mathbf{f}_X; \hat{\mathbf{f}}, -(\nabla\nabla^\intercal \log p(\mathbf{f}_X \mid y)|_{f_X=\hat{f}})^{-1}) =: \mathcal{N}(\mathbf{f}_X; \hat{\mathbf{f}}, \hat{\Sigma})$$

- approximate posterior **predictions** at f_x for **latent function**

$$\begin{aligned} q(f_x \mid y) &= \int p(f_x \mid \mathbf{f}_X) q(\mathbf{f}_X)\, d\mathbf{f}_X = \int \mathcal{N}(f_x; m_x + k_{xX}K_{XX}^{-1}(\mathbf{f}_X - m_X), k_{xx} - k_{xX}K_{XX}^{-1}k_{Xx}) q(\mathbf{f}_X)\, d\mathbf{f}_X \\ &= \mathcal{N}(f_x;\ m_x + k_{xX}K_{XX}^{-1}(\hat{\mathbf{f}} - m_X),\ k_{xx} - k_{xX}K_{XX}^{-1}k_{Xx} + k_{xX}K_{XX}^{-1}\hat{\Sigma}K_{XX}^{-1}k_{Xx}) \end{aligned}$$

- compute predictions for **label probabilities**:

$$\mathbb{E}_{p(f|y)}[\pi_x] \approx \mathbb{E}_q[\pi_x] = \int \sigma(f_x) q(f_x \mid y)\, df_x \quad \text{or (not the same!)} \quad \hat{\pi}_x = \sigma(\mathbb{E}_q(f_x))$$

- the Laplace approximation is only very roughly motivated (see above)
- it can be **arbitrarily wrong**, since it is a **local** approximation
- but it is still better than a point estimate!
- and it is typically the most computationally efficient thing to try, because it uses only auto-diff and linear algebra
- for logistic regression, it tends to work relatively well, because
  - the log posterior is concave (see below)
  - the algebraic structure of the link function yields "almost" a Gaussian posterior (cf. picture above)

Today, we will only implement the *mode-finding*, and leave the Hessian/uncertainty for next time

Implementing the Laplace Approximation

Derivation

[based on Rasmussen & Williams, 2006, §3.4]

$$p(f) = \mathcal{GP}(f, m, k) \qquad p(\boldsymbol{y} \mid \boldsymbol{f}_X) = \prod_{i=1}^{n} \sigma(y_i f_{x_i}) \qquad \sigma(z) = \frac{1}{1+e^{-x}}$$

$$\begin{aligned}\log p(\boldsymbol{f}_X \mid \boldsymbol{y}) &= \log p(\boldsymbol{y} \mid \boldsymbol{f}_X) + \log p(\boldsymbol{f}_X) - \log p(\boldsymbol{y}) \quad \text{with} \quad \log \sigma(y_i f_{x_i}) = -\log(1+e^{-y_i f_{x_i}}) \\ &= \sum_{i=1}^{n} \log \sigma(y_i f_{x_i}) - \frac{1}{2}(\boldsymbol{f}_X - \boldsymbol{m}_X)^\intercal K_{XX}^{-1}(\boldsymbol{f}_X - \boldsymbol{m}_X) + \text{const.}\end{aligned}$$

$$\nabla \log p(\boldsymbol{f}_X \mid \boldsymbol{y}) = \sum_{i=1}^{n} \nabla \log \sigma(y_i f_{x_i}) - K_{XX}^{-1}(\boldsymbol{f}_X - \boldsymbol{m}_X) \quad \text{with} \quad \frac{\partial \log \sigma(y_i f_{x_i})}{\partial f_{x_j}} = \delta_{ij}\left(\frac{y_i + 1}{2} - \sigma(f_{x_i})\right)$$

Gaussian Process Classification — (Probabilistic) Logistic Regression:

- Supervised classification phrased in a **discriminative** model with probabilistic interpretation
- model binary outputs as a **transformation** of a **latent function** with a Gaussian process prior
- due to **non-Gaussian likelihood**, the posterior is non-Gaussian; exact inference **intractable**
- **Laplace approximation:** Find MAP estimator, second order expansion for Gaussian approximation

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}}
```