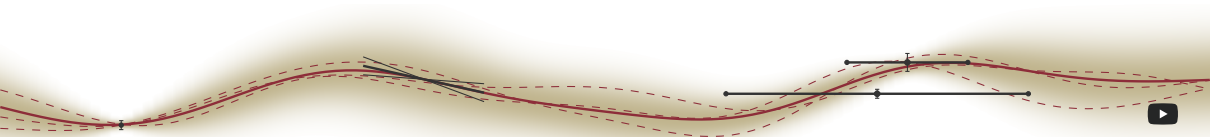
Probabilistic Machine Learning
Lecture 15
Gaussian Process Classification

Philipp Hennig

22 June 2023

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

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



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

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

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

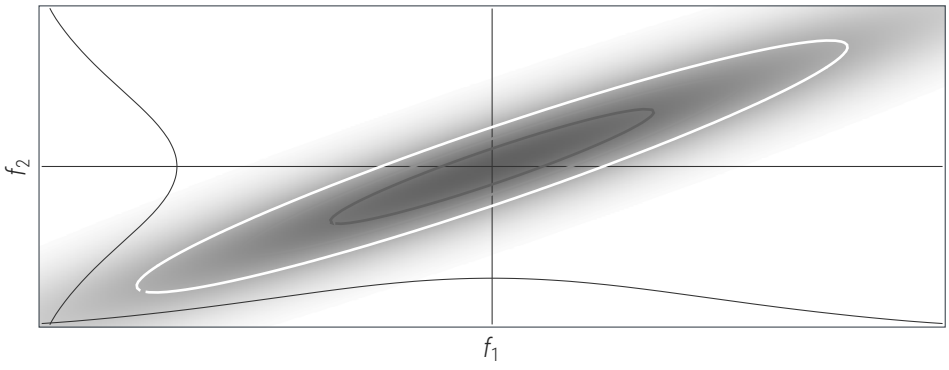
The problem: The posterior is not Gaussian!

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

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

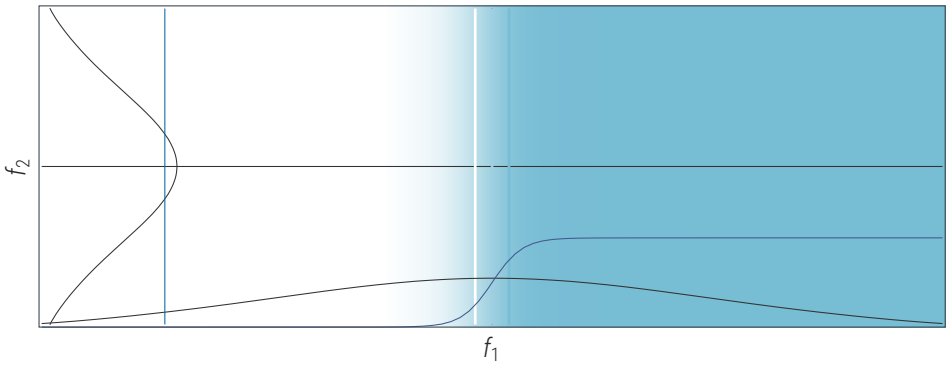
Logistic Regression is not analytic

We'll have to break out the toolbox



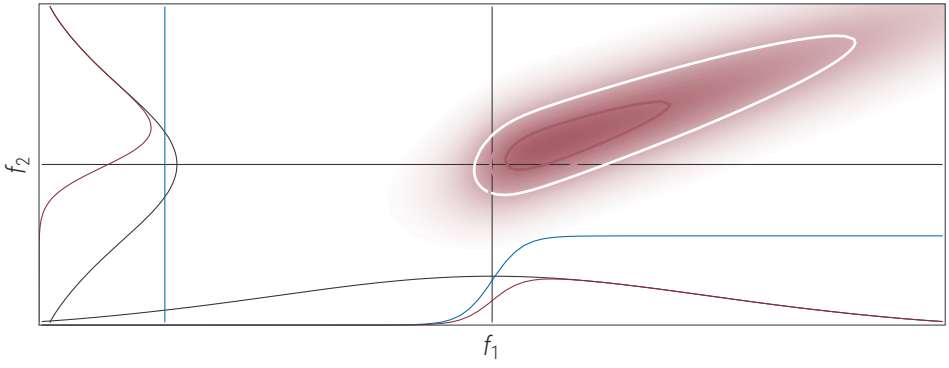
Logistic Regression is not analytic

We'll have to break out the toolbox



Logistic Regression is not analytic

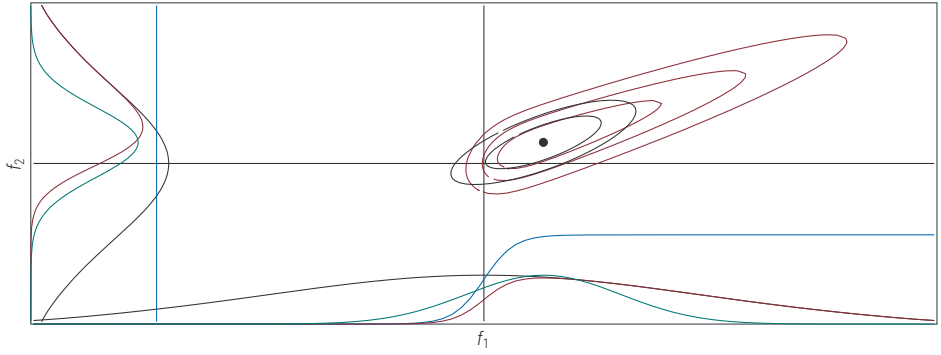
We'll have to break out the toolbox



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**

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

- 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

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

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 focus on the Hessian/uncertainty

Loss, Gradient, Hessian for Logistic Regression

Derivation

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

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

$$\begin{aligned}
\log p(\boldsymbol{f}_X \mid \boldsymbol{y}) &= \log p(\boldsymbol{y} \mid \boldsymbol{f}_X) + \log p(\boldsymbol{f}_X) - \log p(\boldsymbol{y}) \quad \text{with} \quad \log \sigma(y_i f_{x_i}) = -\log(1 + e^{-y_i f_{x_i}}) \\
&= \sum_{i=1}^{n} \log \sigma(y_i f_{x_i}) - \frac{1}{2}(\boldsymbol{f}_X - \boldsymbol{m}_X)^\intercal K_{XX}^{-1}(\boldsymbol{f}_X - \boldsymbol{m}_X) + \text{const.} \\
\nabla \log p(\boldsymbol{f}_X \mid \boldsymbol{y}) &= \sum_{i=1}^{n} \nabla \log \sigma(y_i f_{x_i}) - K_{XX}^{-1}(\boldsymbol{f}_X - \boldsymbol{m}_X) \quad \text{with} \quad \frac{\partial \log \sigma(y_i f_{x_i})}{\partial f_{x_j}} = \delta_{ij}\left(\frac{y_i + 1}{2} - \sigma(f_{x_i})\right) \\
\nabla\nabla^\intercal \log p(\boldsymbol{f}_X \mid \boldsymbol{y}) &= \sum_{i=1}^{n} \nabla\nabla^\intercal \log \sigma(y_i f_{x_i}) - K_{XX}^{-1} \quad \text{with} \quad \frac{\partial^2 \log \sigma(y_i f_{x_i})}{\partial f_{x_a} \partial f_{x_b}} = -\delta_{ia}\delta_{ib} \underbrace{\sigma(f_{x_i})(1 - \sigma(f_{x_i}))}_{=: w_i \text{ with } 0 < w_i < 1} \\
&=: -\operatorname{diag} \boldsymbol{w} - K^{-1} = -(W + K^{-1}) \quad \leftarrow \text{convex minimization / concave maximization!}
\end{aligned}$$

Newton Optimization

more efficient optimization for convex problems

[Rasmussen & Williams, 2006, §3.4]

- since we're already computing the Hessian, we can use it to optimize the log posterior, with updates

$$f \leftarrow f - \left(\nabla\nabla^\intercal \log p(\boldsymbol{f}_X \mid \boldsymbol{y})\right)^{-1} \nabla \log p(\boldsymbol{f}_X \mid \boldsymbol{y})$$

- Newton-Raphson optimization converges in *one step* on quadratic functions
- For non-quadratic, convex problems, it asymptotically converges with quadratic convergence rate:

$$\|f_{t+1} - f^*\| \leq C \cdot \|f_t - f^*\|^2$$

- Newton's method has (at least on paper) no learning rate, and no other parameters to tune.
- It requires computing and decomposing the Hessian, though

Implementing Newton Optimization

Concrete Algorithm, preliminary form

[Rasmussen & Williams, 2006, §3.4]

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

$$\nabla\nabla^\intercal \log p(\boldsymbol{f}_X \mid \boldsymbol{y}) = \sum_{i=1}^{n} \nabla\nabla^\intercal \log \sigma(y_i f_{x_i}) - K_{XX}^{-1} \quad \text{with} \quad \frac{\partial^2 \log \sigma(y_i f_{x_i})}{\partial f_{x_a} \partial f_{x_b}} = -\delta_{ia}\delta_{ib} \underbrace{\sigma(f_{x_i})(1-\sigma(f_{x_i}))}_{=:w_i \text{ with } 0<w_i<1}$$

$$=: -\operatorname{diag} \boldsymbol{w} - K^{-1} = -(W + K^{-1})$$

procedure GP-CLASSIFY-TRAIN($K_{XX}, m_X, \boldsymbol{y}$)
  $f \leftarrow m_X, \Delta \leftarrow \infty$ // initialize
  while $\Delta > \epsilon$ **do**
    $W \leftarrow \operatorname{diag}(\sigma(f) \odot (1-\sigma(f)))$ // $= -\nabla\nabla \log p(\boldsymbol{y} \mid f_X)$, Hessian of log likelihood
    $g \leftarrow \frac{y+1}{2} - \sigma(f) - K_{XX}^{-1}(f - m_X)$ // compute gradient
    $\Delta \leftarrow -(W + K^{-1})^{-1} g$ // Newton step
    $f \leftarrow f - \Delta$ // perform step
  end while
  return f
end procedure

Numerical stability

some advanced observations

[Rasmussen & Williams, 2006, §3.4]

- The Newton step can be numerically unstable when $(W + K^{-1})$ has very small eigenvalues. It is helpful to consider the matrix

$$B := I + W^{\frac{1}{2}} K W^{\frac{1}{2}}$$

which is symmetric positive definite, and has eigenvalues ≥ 1.

- From the matrix inversion lemma, we get

$$(W + K^{-1})^{-1} = K - K W^{\frac{1}{2}} B^{-1} W^{\frac{1}{2}} K$$

and thus

$$\begin{aligned}
\boldsymbol{f}_{t+1} &= \boldsymbol{f}_t - (\nabla\nabla^\intercal \log p(\boldsymbol{f}_t \mid \boldsymbol{y}))^{-1} \nabla \log p(\boldsymbol{f}_t \mid \boldsymbol{y}) \\
&= \boldsymbol{f}_t + (W + K^{-1})^{-1} (\nabla \log p(\boldsymbol{y} \mid \boldsymbol{f}_t)) - K^{-1} \boldsymbol{f}_t \\
&= (W + K^{-1})^{-1} (\nabla \log p(\boldsymbol{y} \mid \boldsymbol{f}_t) + W \boldsymbol{f}_t) \\
&= (K - K W^{\frac{1}{2}} B^{-1} W^{\frac{1}{2}} K)(\nabla \log p(\boldsymbol{y} \mid \boldsymbol{f}_t) + W \boldsymbol{f}_t)
\end{aligned}$$

Computing Predictions

from the Laplace approximation

[Rasmussen & Williams, 2006, §3.4]

From above:

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

$$= \mathcal{N}(f_x;\ m_x + k_{xX}K_{XX}^{-1}(\hat{\boldsymbol{f}} - m_X),\ k_{xx} - k_{xX}K_{XX}^{-1}k_{Xx} + k_{xX}K_{XX}^{-1}\hat{\Sigma}K_{XX}^{-1}k_{Xx})$$

$\log p(\boldsymbol{f}_X \mid \boldsymbol{y})$

with variance

$$= k_{xx} - k_{xX}K_{XX}^{-1}k_{Xx} + k_{xX}K_{XX}^{-1}\hat{\Sigma}K_{XX}^{-1}k_{Xx}$$
$$= k_{xx} - k_{xX}[K_{XX}^{-1} + K_{XX}^{-1}(K^{-1} + W)^{-1}K_{XX}^{-1}]k_{Xx}$$
$$= k_{xx} - k_{xX}(K + W^{-1})^{-1}k_{Xx}$$
$$= k_{xx} - k_{xX}W^{\frac{1}{2}}W^{-\frac{1}{2}}(K + W^{-1})^{-1}W^{-\frac{1}{2}}W^{\frac{1}{2}}k_{Xx}$$
$$= k_{xx} - k_{xX}W^{\frac{1}{2}}B^{-1}W^{\frac{1}{2}}k_{Xx}$$

Computing predictive *probabilities*

DJC MacKay, 1992

- We can compute the pdf of $\sigma(f)$ analytically as

$$p(\sigma(f(x))) = \mathcal{N}(\sigma^{-1}(f(x)); m_x, s_x^2) \left| \frac{\partial \sigma^{-1}(f(x))}{\partial f} \right| = \mathcal{N}(\sigma^{-1}(f); m_x, s_x^2) \frac{1}{(\sigma(1-\sigma))}$$

- but this distribution has no analytic expected value.
- Various approximations have been proposed. A simple (but imperfect) one [MacKay, 1992] is

$$\mathbb{E}_{\mathcal{N}(f;m,s^2)}(\sigma(f)) = \sigma\left(\frac{m}{\sqrt{1+\frac{\pi}{8}s^2}}\right)$$

- a more general, but also more expensive option is to build a regular "Gaussian grid" and compute an approximate integral

Code

Gaussian Process Classification:

- 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
- tune code for numerical stability, efficient computations
- Laplace approximation provides Gaussian posterior on training points, hence evidence, predictions

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}}
```