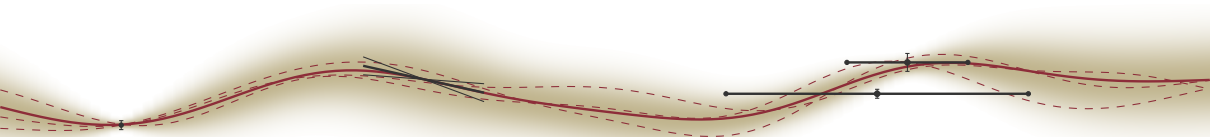
Probabilistic Inference and Learning
Lecture 10
Gaussian Process Classification and other Non-Gaussian Observation Models

Philipp Hennig

19 November 2018

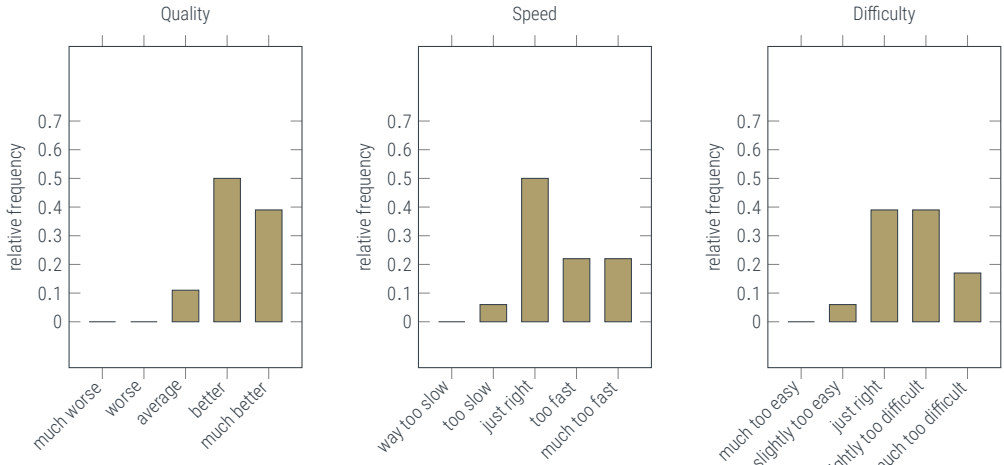
Eberhard Karls
Universität
Tübingen

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



Last Lecture: Debrief

Feedback dashboard



Last Lecture: Debrief

Detailed Feedback

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Things you did not like:

- **"please no break [on Wednesday]"**
- **Lecture has to end by 10:50 at latest**
- "can you just report the results and let us do the derivations?"
- **long derivation**
- lecture hall is too cold
- **the code should contain equations!**
- use something other than color to highlight

Things you did not understand:

- "I'm not sure what we're assuming for the Kalman filter to work"
- why $\int p(x_t \mid x_{t+1}, Y_{0:t})\, dx_t = 1$?

Things you enjoyed:

- detailed derivations
- plots
- code
- slides
- "the concepts — really getting into it"
- recaps and connections (but too long now)

Overview of Lectures so far:

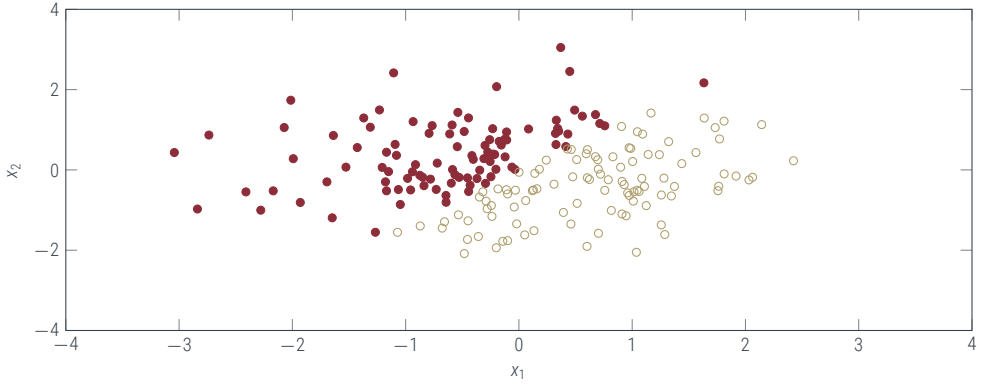
0. Introduction to Reasoning under Uncertainty
1. Probabilistic Reasoning
2. Probabilities over Continuous Variables
3. Gaussian Probability Distributions
4. Gaussian Parametric Regression
5. More on Parametric Regression
6. Gaussian Processes
7. More on Kernels & GPs
8. A practical GP example
9. Markov Chains, Time Series, Filtering
10. Classification (TODAY)
11. Empirical Example of Classification (WED)
12. More Connections to SLT (NEXT MON)
13. Theory of Filtering / SDEs (NEXT WED)

Today:

- Classification: Supervised Learning with Discrete Outputs
- *Conceptually*, a simple generalization of GP regression
- *Computationally*, we will have to introduce new approximation methods

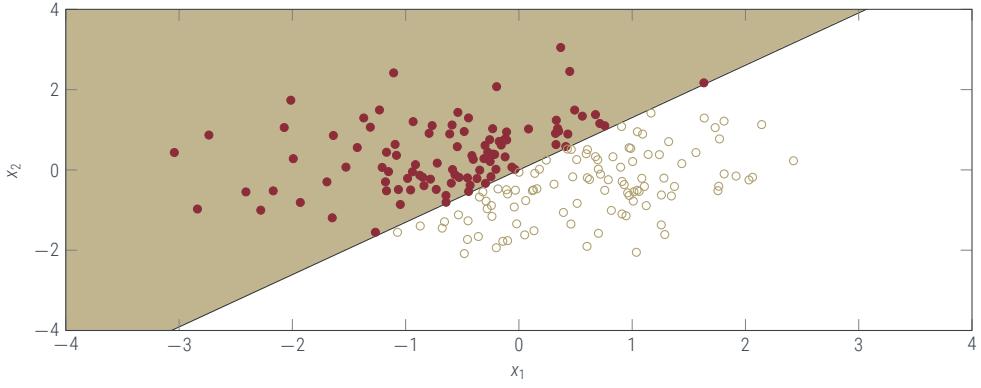
Classification Problems

a typography of supervised learning problems



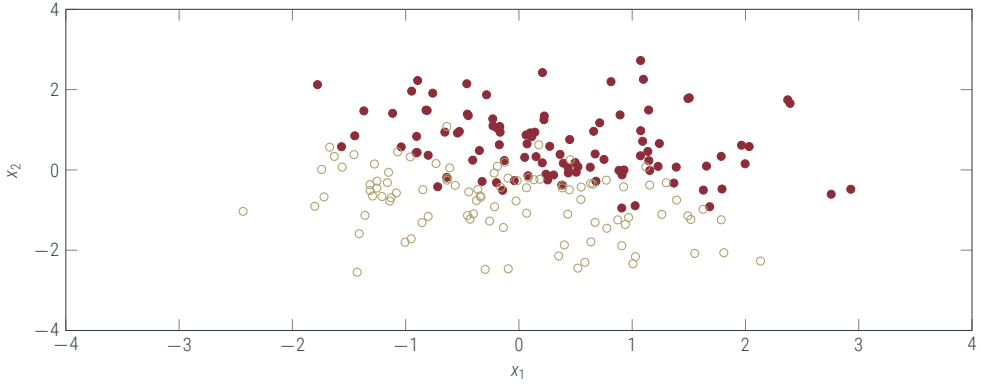
Classification Problems

a typography of supervised learning problems



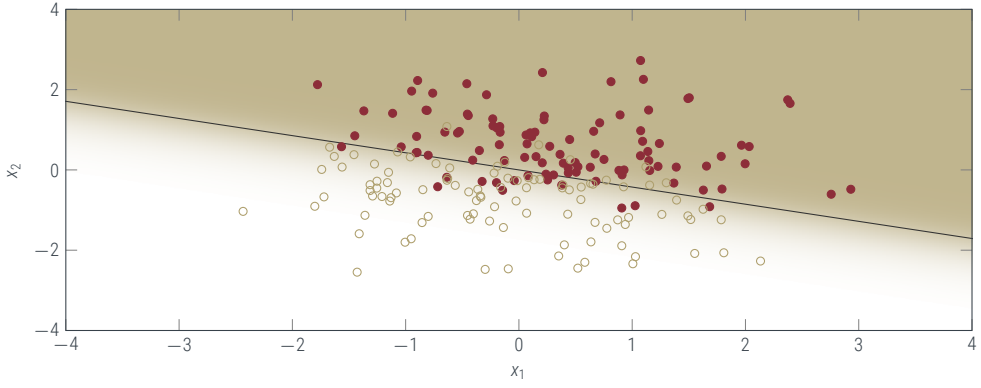
Classification Problems

a typography of supervised learning problems



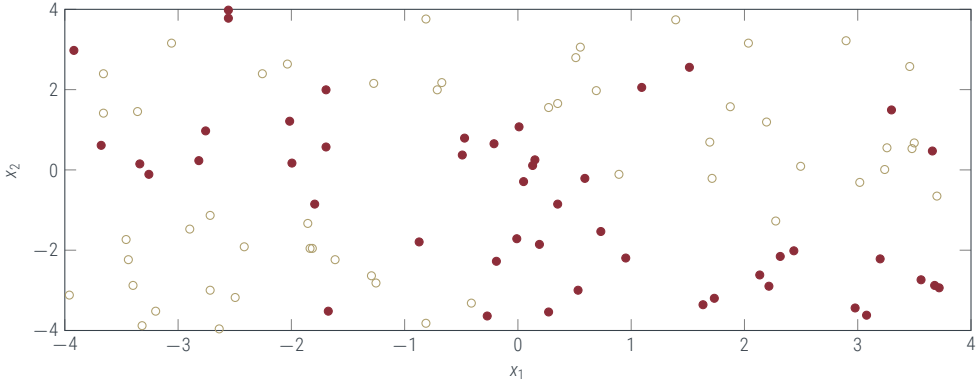
Classification Problems

a typography of supervised learning problems



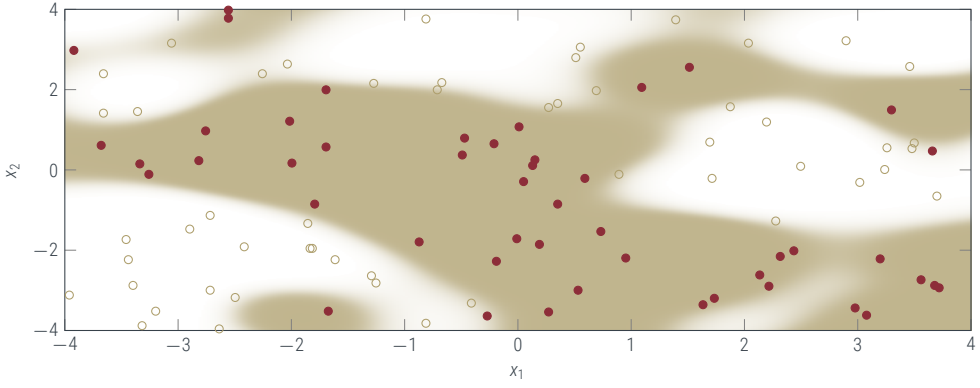
Classification Problems

a typography of supervised learning problems



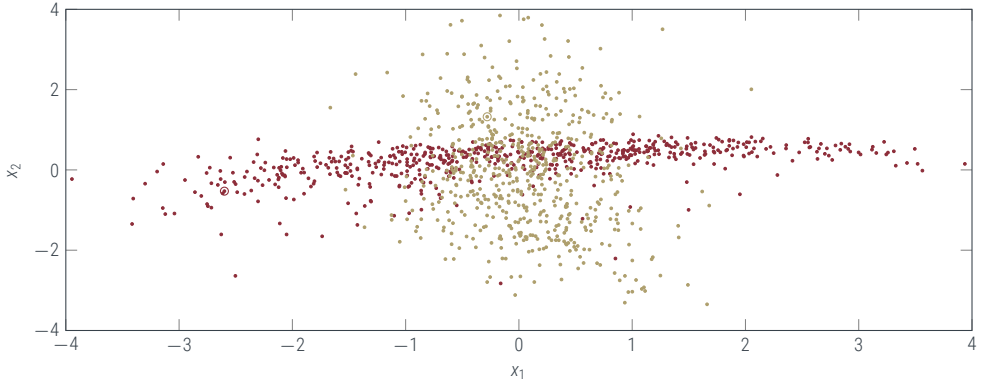
Classification Problems

a typography of supervised learning problems



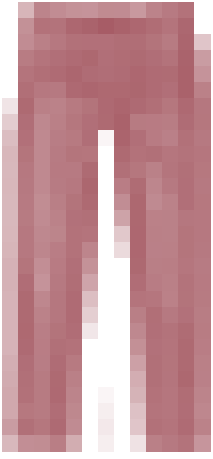
Classification Problems

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)

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

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

Classification:

Given supervised *data* (special case $m = 1$: binary classification)

$$(X, Y) := (x_i, c_i)_{i=1,\dots,n} \text{ with } x_i \in \mathbb{X}, c_i \in \{0, \dots, d-1\}$$

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" $y_i \sim \pi_{x_i}$.

Classification and regression are quite similar at first sight. But classification is a bit more subtle…

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* $\pi_x(y) = p(y \mid x)$

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: $p(y \mid x) = \mathcal{N}(y; f_x, \sigma^2) = \pi_x(y)$

Until further notice, consider only discriminative binary classification:

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

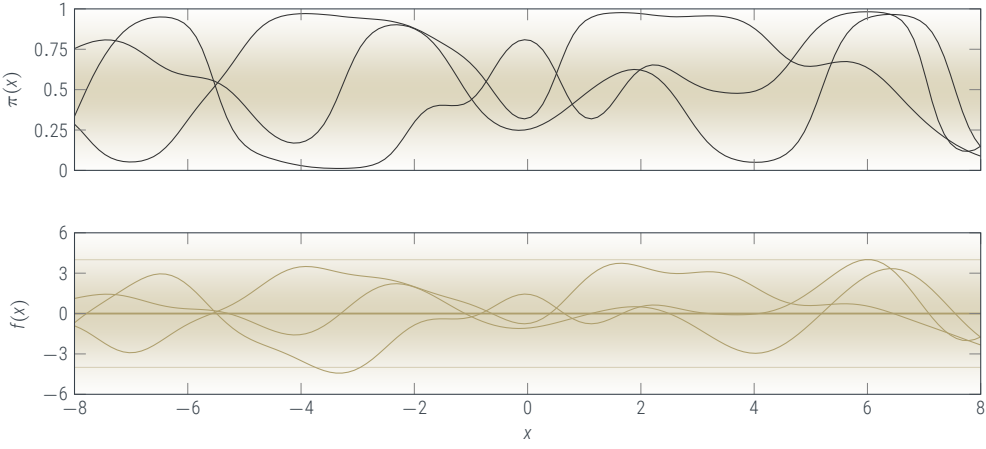
$$p(y \mid x) = \begin{cases} \pi(x) & \text{if } y = 1 \\ 1 - \pi(x) & \text{if } y = -1 \end{cases}$$

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. $y \in \mathbb{R}$.

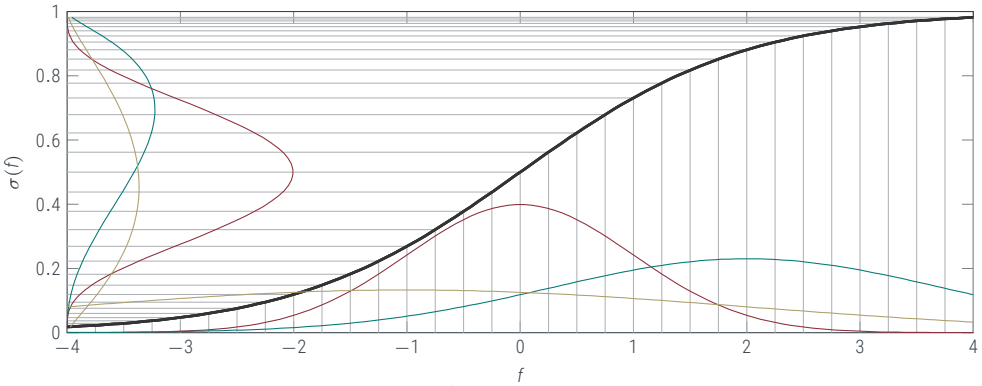
Let's not throw out Gauss just yet

Turning Gaussian process models into discrete likelihoods



Link Functions

sigmoids – the **logistic** link function



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

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

CODE

Generative Model for Logistic Regression.ipynb

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

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

A Gaussian Process model for Classification

Logistic Regression

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

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

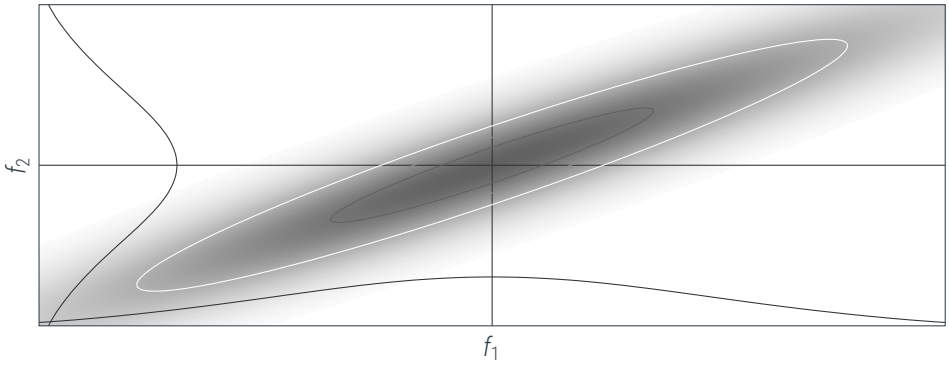
The problem: The posterior is not Gaussian!

$$p(f_X \mid Y) = \frac{p(Y \mid f_X) p(f_X)}{p(Y)} = \frac{\sigma(Y \odot f_X) \mathcal{N}(f_X; m, k)}{\int \sigma(Y \odot f_X) \mathcal{N}(f_X; m, k)\, df_X}$$

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

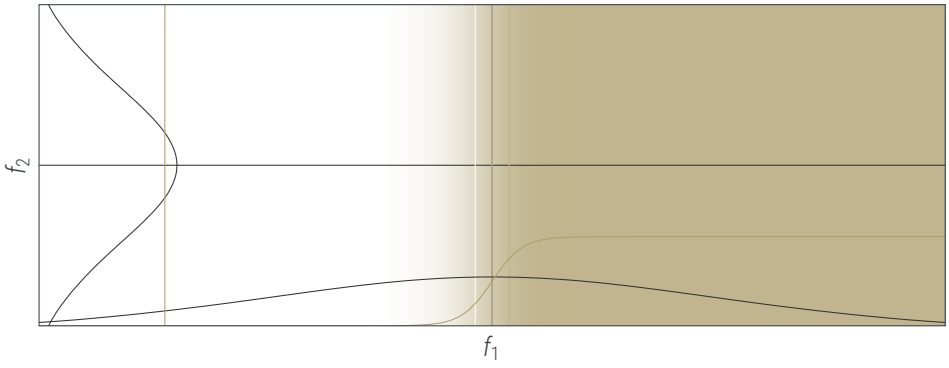
Logistic Regression is non-analytic

We'll have to break out the toolbox



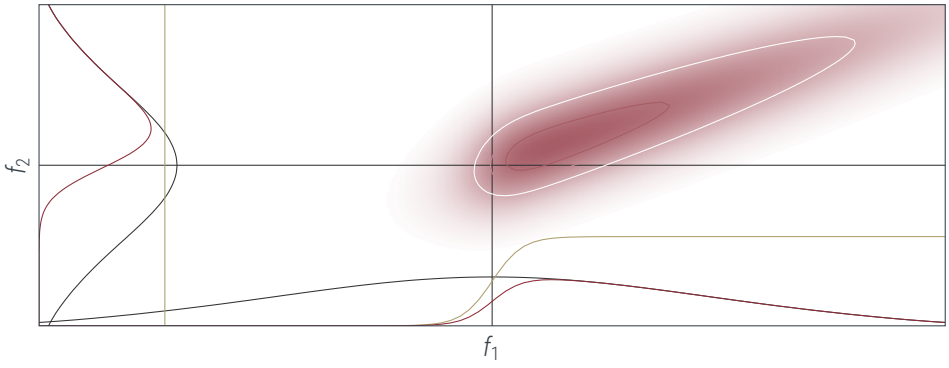
Logistic Regression is non-analytic

We'll have to break out the toolbox



Logistic Regression is non-analytic

We'll have to break out the toolbox



We do not always care about all details of the posterior, just "key aspects".

- Remember that the Gaussian choice was also one of convenience.
- **Moments** of $p(f, y) = p(y \mid f)p(f)$ we may be interested in

$$\mathbb{E}_p(1) = \int p(y, f)\, df = \int 1 \cdot p(y, f)\, df = Z \quad \text{the } \textbf{evidence}$$

$$\mathbb{E}_{p(f|y)}(f) = \int f \cdot p(f \mid y)\, df = \frac{1}{Z} \int f \cdot p(f, y)\, df = \bar{f} \quad \text{the } \textbf{mean}$$

$$\mathbb{E}_{p(f|y)}(f^2) - \bar{f}^2 = \int f^2 \cdot p(f \mid y)\, df - \bar{f}^2 = \frac{1}{Z} \int f^2 \cdot p(f, y)\, df - \bar{f}^2 = \mathrm{var}(f) \quad \text{the } \textbf{variance}$$

Z for hyperparameter tuning $\quad$ $\bar{f}$ as a point estimator $\quad$ $\mathrm{var}(f)$ as an error estimator

Unfortunately, all these are usually intractable. But we can aim to approximate them.

The Toolbox

Five principal methods for dealing with computational complexity in probabilistic inference

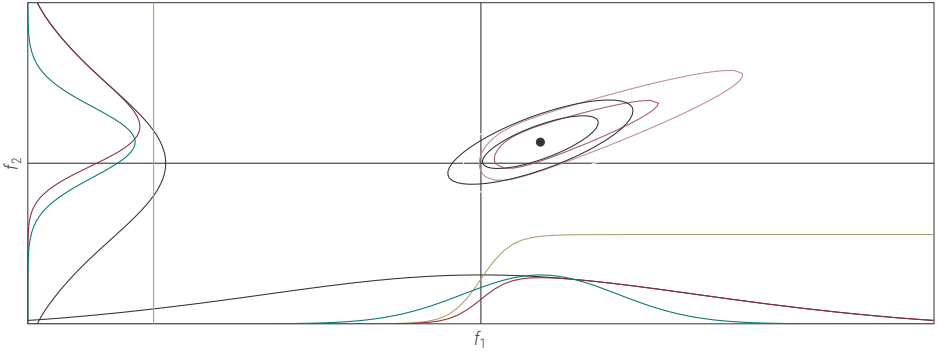
1. **Maximum Likelihood (ML) / Maximum A-Posteriori (MAP)** estimation:
   To estimate θ in $p(D \mid \theta)$ or $p(\theta \mid D)$, set $\hat{\theta} = \arg\max_\theta p$.
2. **Laplace** Approximation: $p(\theta \mid D) \approx \mathcal{N}\left(\theta; \hat{\theta}, -(\nabla\nabla^\intercal \log p(\theta \mid D))^{-1}\right)$
3. ????
4. ????
5. **Numerical Quadrature:**
   To marginalize θ, compute $\int p(f \mid \theta)\, d\theta \approx \sum_i w_i \cdot p(f \mid \theta_i)$

Disclaimer: The listed items are neither mutually exclusive nor collectively exhaustive. Some of the methods are intricately interrelated.

The Laplace Approximation

A local Gaussian approximation

Pierre Simon M. de Laplace, 1814



An idea as old as Probabilistic Inference

Théorie analytique des probabilités, 1814

Pierre-Simon, marquis de Laplace

(1749-1827)

l'intégrale du numérateur étant prise depuis $x=\theta$ jusqu'à $x=\theta'$, et celle du dénominateur étant prise depuis $x=0$ jusqu'à $x=1$?

La valeur de s la plus probable, est celle qui rend y un *maximum*. Nous la désignerons par a. Si aux limites de x, y est nul, alors chaque valeur de y a une valeur égale correspondante de l'autre côté du *maximum*.

Quand les valeurs de x, considérées indépendamment du résultat observé, ne sont pas également possibles; en nommant z la fonction de x qui exprime leur probabilité; il est facile de voir, par ce qui a été dit dans le premier chapitre de ce Livre, qu'en changeant dans la formule (1), y dans yz, on aura la probabilité que la valeur de x est comprise dans les limites $x=\theta$ et $x=\theta'$. Cela revient à supposer toutes les valeurs de x également possibles *à priori*, et à considérer le résultat observé, comme étant formé de deux résultats indépendans, dont les probabilités sont y et z. On peut donc ramener ainsi tous les cas à celui où l'on suppose *à priori*, avant l'événement, une égale possibilité aux différentes valeurs de x, et par cette raison, nous adopterons cette hypothèse dans ce qui va suivre.

Nous avons donné dans les n^os^ 22 et suivans du premier Livre, les formules nécessaires pour déterminer par des approximations convergentes, les intégrales du numérateur et du dénominateur de la formule (1), lorsque les événemens simples dont se compose l'événement observé, sont répétés un très-grand nombre de fois; car alors y a pour facteurs, des fonctions de x élevées à de grandes puissances. Nous allons, au moyen de ces formules, déterminer la loi de probabilité des valeurs de x, à mesure qu'elles s'éloignent de la valeur a, la plus probable, ou qui rend y un *maximum*. Pour cela, reprenons la formule (c) du n^o^ 27 du premier Livre,

$$\int y\,dx = Y.\left\{U+\frac{1}{2}.\frac{d^2U^2}{1.2.dx^2}+\frac{1.3}{2^2}.\frac{d^4U^3}{1.2.3.4.dx^4}+\text{etc.}\right\}.f\,dt.c^{-t^2}$$
$$+\frac{Y}{2}.c^{-T^2}.\left\{\frac{dU^2}{dx}-T.\frac{d^2U^2}{1.2.dx^2}+(T^2+1).\frac{d^3U^3}{1.2.3.dx^3}-\text{etc.}\right\}; \quad (2)$$
$$-\frac{Y}{2}.c^{-T'^2}.\left\{\frac{dU^2}{dx}+T'.\frac{d^2U^2}{1.2.dx^2}+(T'^2+1).\frac{d^3U^3}{1.2.3.dx^3}+\text{etc.}\right\}$$

ψ est égal à $\frac{x-a}{\sqrt{\log Y-\log y}}$, et U, $\frac{dU^2}{dx}$, $\frac{d^2U^3}{dx^2}$, etc. sont ce que deviennent v, $\frac{dv^2}{dx}$, $\frac{d^2v^3}{dx^2}$, etc., lorsqu'on y change après les différentiations, x en a, a étant la valeur de x qui rend y un *maximum*: T est égal à ce que devient la fonction $\sqrt{\log Y-\log y}$, lorsqu'on change x en $a-\theta$ dans y, et T' est ce que devient la même fonction, lorsqu'on y change x dans $a+\theta'$. L'expression précédente de $\int y\,dx$ donne la valeur de cette intégrale, dans les limites $x=a-\theta$ et $x=a+\theta'$; l'intégrale $\int dt.c^{-t^2}$ étant prise depuis $t=-T$ jusqu'à $t=T'$.

Le plus souvent, aux limites de l'intégrale $\int y\,dx$, étendue depuis $x=0$ jusqu'à $x=1$, y est nul; ou lorsque y n'est pas nul, il devient si petit à ces limites, qu'on peut le supposer nul. Alors, on peut faire à ces limites T et T' infinis, ce qui donne pour l'intégrale $\int y\,dx$, étendue depuis $x=0$ jusqu'à $x=1$,

$$\int y\,dx = Y.\left\{U+\frac{1}{2}.\frac{d^2U^2}{1.2.dx^2}+\frac{1.3}{2^2}.\frac{d^4U^3}{1.2.3.4.dx^4}+\text{etc.}\right\}.\sqrt{\pi};$$

ainsi la probabilité que la valeur de x est comprise dans les limites $x=a-\theta$ et $x=a+\theta'$, est égale à

$$\frac{\int dt.c^{-t^2}}{\sqrt{\pi}}+\frac{\left\{\begin{array}{l}\frac{1}{2}.c^{-T^2}.\left\{\frac{dU^2}{dx}-T.\frac{d^2U^2}{1.2.dx^2}+(T^2+1).\frac{d^3U^3}{1.2.3.dx^3}-\text{etc.}\right\}\\ -\frac{1}{2}.c^{-T'^2}.\left\{\frac{dU^2}{dx}+T'.\frac{d^2U^2}{1.2.dx^2}+(T'^2+1).\frac{d^3U^3}{1.2.3.dx^3}+\text{etc.}\right\}\end{array}\right.}{\left\{U+\frac{1}{2}.\frac{d^2U^2}{1.2.dx^2}+\frac{1.3}{2^2}.\frac{d^4U^3}{1.2.3.4.dx^4}+\text{etc.}\right\}.\sqrt{\pi}}; \quad (3)$$

On voit par le n^o^ 23 du premier Livre, que dans le cas où y a pour facteurs, des fonctions de x élevées à de grandes puissances de l'ordre $\frac{1}{\alpha^2}$, α étant une fraction extrêmement petite, alors U est le plus souvent de l'ordre $\sqrt{\alpha}$, ainsi que ses différences successives; U, $\frac{dU^2}{dx}$, $\frac{d^2U^3}{dx^2}$, etc. sont respectivement des ordres $\sqrt{\alpha}$, α, $\alpha^{\frac{3}{2}}$, etc.; d'où il suit que la convergence des séries de la formule (3), exige que T et T' ne soient pas d'un ordre supérieur à $\frac{1}{\sqrt{\alpha}}$.

The Laplace Approximation

Once again, formally

- 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

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

- Note that, if $p(\theta) = \mathcal{N}(\theta; m, \Sigma)$, then $p(\theta) = q(\theta)$

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**

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

- 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**

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

- 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 often among the most computationally efficient things to try
- 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)

Implementing the Laplace Approximation

Derivation

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

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

Implementing the Laplace Approximation I

Newton Optimization

[Rasmussen & Williams, 2006, §3.4]

```
1  procedure OPTIMIZE(L(·), f_0)
2      f ⇐ f_0                                   // initialize
3      while not converged do
4          g ⇐ ∇L(f)                             // compute gradient
5          H ⇐ (∇∇ᵀL(f))^{-1}                    // compute inverse Hessian
6          Δ ⇐ Hg                                // Newton step
7          f ⇐ f − Δ                             // perform step
8          converged ⇐ ‖Δ‖ < ϵ                   // check for convergence
9      end while
10     return f
11 end procedure
```

Implementing the Laplace Approximation II

Concrete Algorithm, preliminary form

[Rasmussen & Williams, 2006, §3.4]

procedure GP-LOGISTIC-TRAIN($K_{XX}, m_X, \boldsymbol{y}$)
  $f \leftarrow m_X$ // initialize
  while not converged **do**
    $r \leftarrow \frac{\boldsymbol{y}+1}{2} - \sigma(f)$ // $= \nabla \log p(\boldsymbol{y} \mid f_X)$, gradient of log likelihood
    $W \leftarrow \mathrm{diag}(\sigma(f) \odot (1 - \sigma(f)))$ // $= -\nabla\nabla \log p(\boldsymbol{y} \mid f_X)$, Hessian of log likelihood
    $g \leftarrow r - K_{XX}^{-1}(f - m_X)$ // compute gradient
    $H \leftarrow -(W + K^{-1})^{-1}$ // compute inverse Hessian
    $\Delta \leftarrow Hg$ // Newton step
    $f \leftarrow f - \Delta$ // perform step
    converged $\leftarrow \|\Delta\| < \epsilon$ // check for convergence
  end while
  return f
end procedure

This can be numerically unstable as it (repeatedly) requires $(W + K^{-1})^{-1}$. For a numerically stable alternative, see end of this pdf.

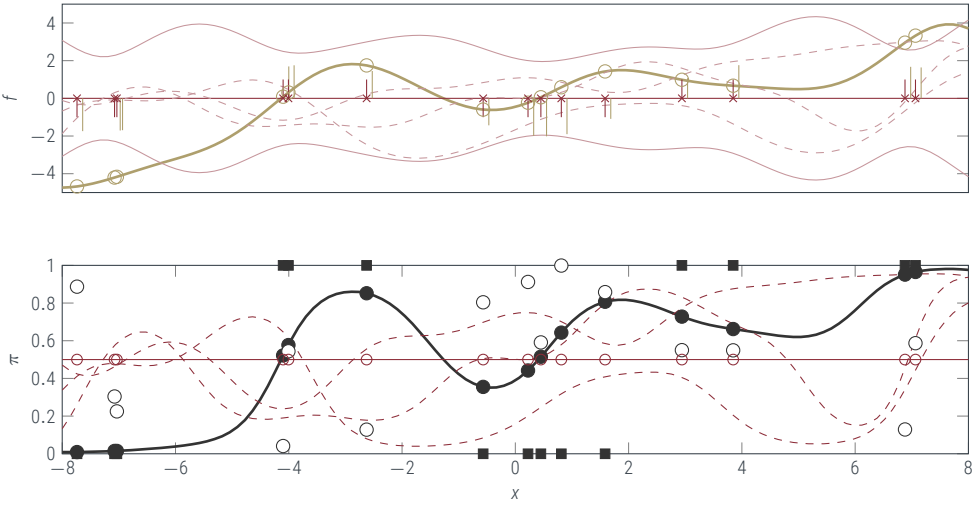
$$\log p(\boldsymbol{f}_X \mid \boldsymbol{y}) = \sum_{i=1}^{n} \log \sigma(y_i f_{x_i}) - \frac{1}{2}(\boldsymbol{f}_X - \boldsymbol{m}_X)^\intercal K_{XX}^{-1}(\boldsymbol{f}_X - \boldsymbol{m}_X) + \text{const.}$$

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

procedure GP-LOGISTIC-PREDICT($\hat{\boldsymbol{f}}, W, R, r, k, \boldsymbol{x}$) // $\hat{f}, W, R, r$ handed over from training
  for $i = 1, \ldots,$ LENGTH($\boldsymbol{x}$) **do**
    $\bar{f}_i \leftarrow k_{x_i X} r$ // mean prediction (note at minimum, $0 = \nabla p(f_X \mid y) = r - K_{XX}^{-1}(f_X - m_X)$).
    $s \leftarrow R^{-1}(W^{1/2} k_{X x_i})$ // pre-computation allows this step in $\mathcal{O}(n^2)$
    $v \leftarrow k_{x_i x_i} - s^\intercal s$ // $v = \text{cov}(f_x)$
    $\bar{\pi}_i \leftarrow \int \sigma(f_i)\mathcal{N}(f_i, \bar{f}_i, v)\, df_i$ // predictive probability for class 1 is $p(y \mid \bar{f}) = \int p(y_x \mid f_x) p(f_x \mid \bar{f})\, df_x$
  end for // entire loop is $\mathcal{O}(n^2 m)$ for m test cases.
  return $\bar{\pi}_x$
end procedure

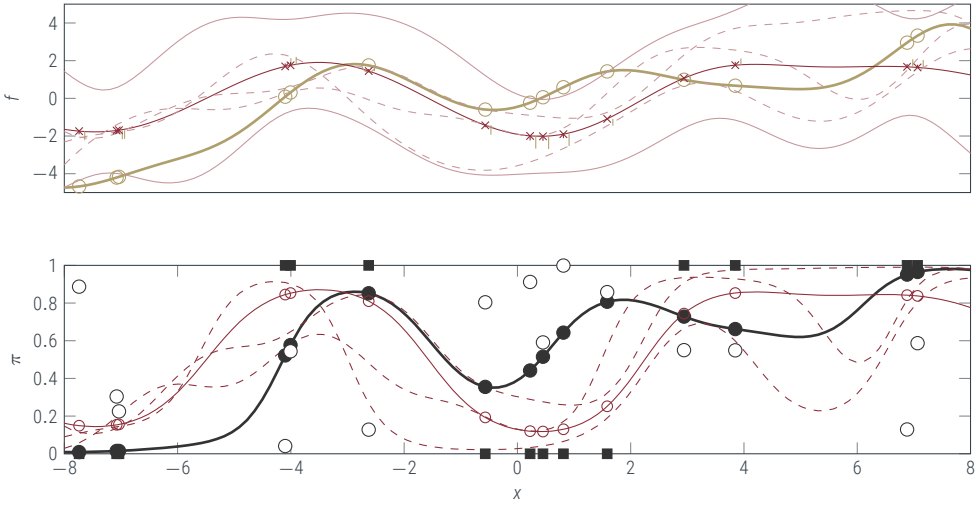
Pictorial View

Gaussian process logistic regression



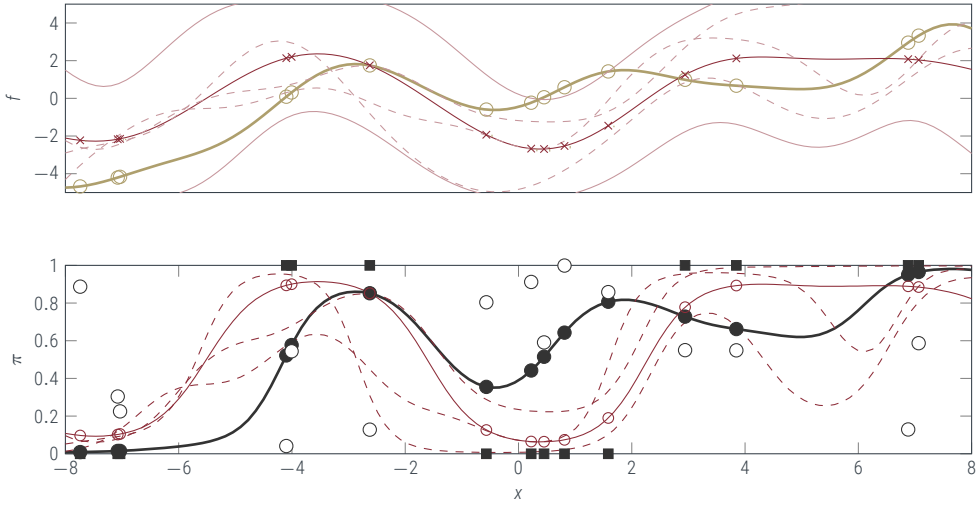
Pictorial View

Gaussian process logistic regression



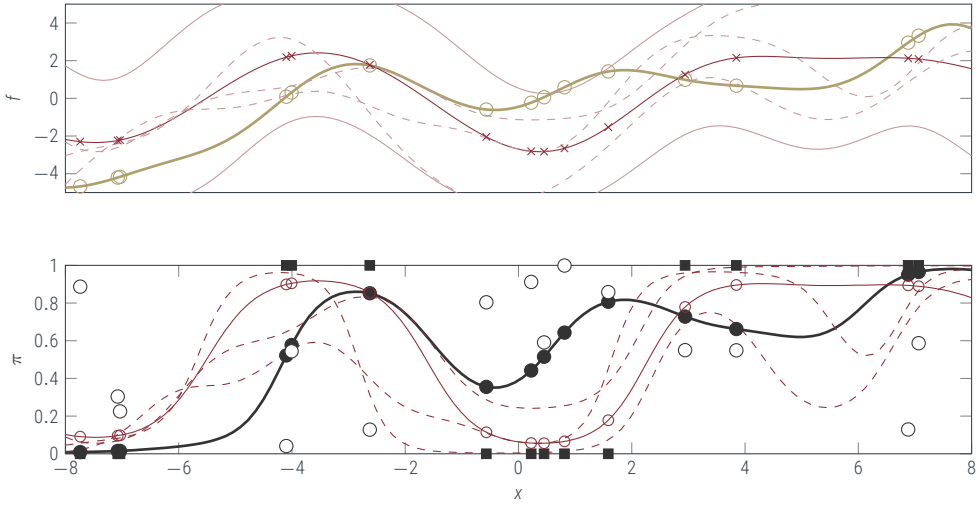
Pictorial View

Gaussian process logistic regression



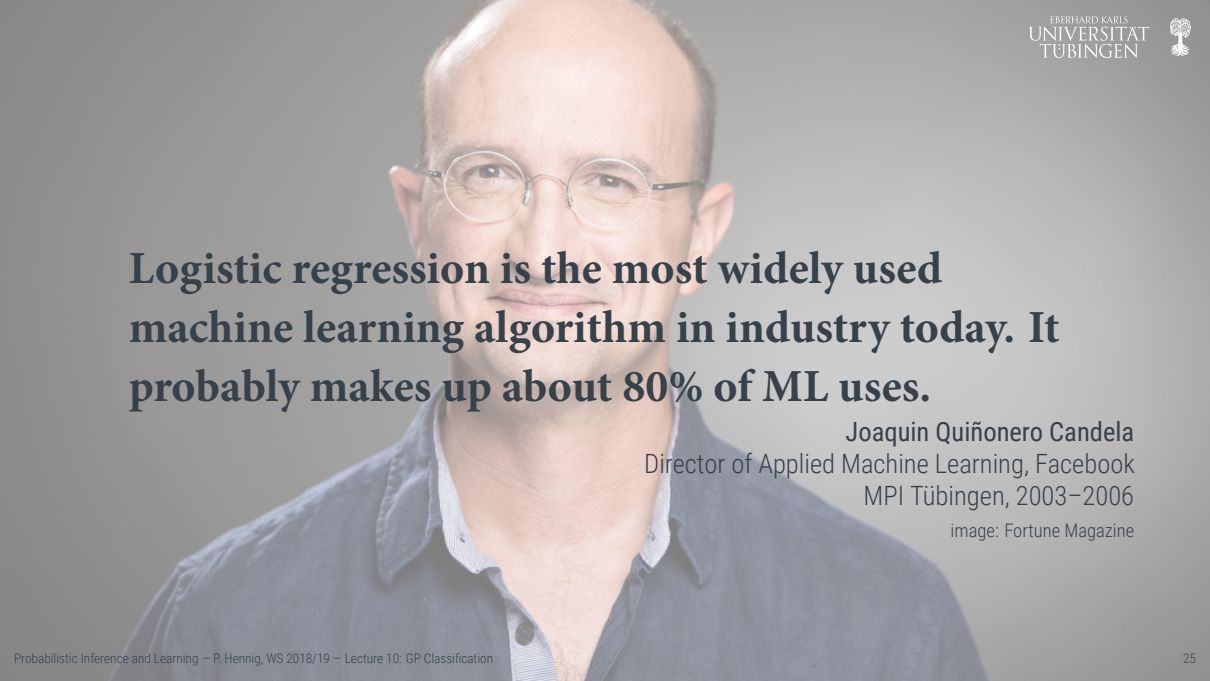
Pictorial View

Gaussian process logistic regression



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



Logistic regression is the most widely used machine learning algorithm in industry today. It probably makes up about 80% of ML uses.

Joaquin Quiñonero Candela
Director of Applied Machine Learning, Facebook
MPI Tübingen, 2003–2006

image: Fortune Magazine

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

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

    $R \leftarrow \text{CHOLESKY}(I + W^{1/2} K_{XX} W^{1/2})$ // 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

For completeness – analogous to above

$$p(v) = \mathcal{N}(v, \mu, \Sigma) \qquad p(\boldsymbol{y} \mid v) = \sigma(\boldsymbol{y} \odot \phi_X^\intercal v) \quad \text{with } v \in \mathbb{R}^F, \phi_X \in \mathbb{R}^{n \times F}$$

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

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

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

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

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

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

Exercise: Re-write Algorithms GP-Logistic-Train & GP-Logistic-Predict in weight-space.

Multi-Class Classification

Extending the Laplace Approximation

- What if we have C class labels $[c_1, \ldots, c_C]$?

Multi-Class Classification

Extending the Laplace Approximation

- What if we have C class labels $[c_1, \ldots, c_C]$?
- Generative Model:

- 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

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

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**

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

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

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

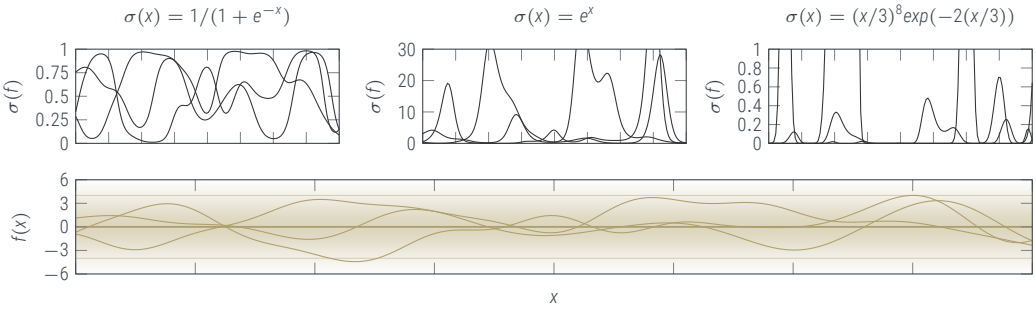
  - At location x_i, generate probabilities for each class by taking the **softmax**

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

The remaining derivations are analogous to the binary case. They are this week's **exercise**.

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)

Summary:

- **Gaussian process classification** provides a probabilistic, discriminative model for classification
- The MAP solution is also known as **logistic regression**, and may be the most widely used machine learning algorithm in industry
- It can be found, for example by **Newton optimization**, or by stochastic gradient descent. The **Laplace approximation** provides a simple approximate posterior (it requires evaluating or estimating the *Hessian* of the log posterior)
- Other non-linearities / link-functions can be used, too, to model other output types (bounded, strictly positive, structured, etc.)

next time: Concrete example implementation

Some practical numerical linear algebra

no, this will not be in the exam …

[Rasmussen & Williams, 2006, §3.4]

The matrix $(W + K^{-1})^{-1}$ can be numerically unstable. Instead, consider

$$B := I + W^{1/2} K_{XX} W^{1/2}.$$

- can be computed in $\mathcal{O}(n^2)$ (because W is diagonal, spd)
- eigenvalues $\lambda_i(B)$ are bounded $1 < \lambda_i < 1 + n \max_{ij} \frac{K_{ij}}{4}$, thus typically numerically stable
- Matrix inversion lemma yields

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

Lemma (The matrix inversion lemma / Sherman-Morrison-Woodbury Identity)

If all the inverses exist, then

$$(A^{-1} + U D^{-1} V^\intercal)^{-1} = A - AU(D + V^\intercal A U)^{-1} V^\intercal A.$$

Some practical numerical linear algebra

no, this will not be in the exam …

[Rasmussen & Williams, 2006, §3.4]

The matrix $(W + K^{-1})^{-1}$ can be numerically unstable. Instead, consider

$$B := I + W^{1/2} K_{XX} W^{1/2}.$$

- can be computed in $\mathcal{O}(n^2)$ (because W is diagonal, spd)
- eigenvalues $\lambda_i(B)$ are bounded $1 < \lambda_i < 1 + n \max_{ij} \frac{K_{ij}}{4}$, thus typically numerically stable
- Matrix inversion lemma yields

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

Lemma (The matrix inversion lemma / Sherman-Morrison-Woodbury Identity)

Secret Weapon!

If all the inverses exist, then

$$(A^{-1} + U D^{-1} V^\intercal)^{-1} = A - AU(D + V^\intercal A U)^{-1} V^\intercal A.$$

Implementing the Laplace Approximation III

Efficient form

[Rasmussen & Williams, 2006, §3.4]

procedure GP-LOGISTIC-TRAIN($K_{XX}, m_X, \boldsymbol{y}$)

  $f \leftarrow m_X$ // initialize

  while not converged **do**

    $r \leftarrow \frac{\boldsymbol{y}+1}{2} - \sigma(f)$ // $= \nabla \log p(\boldsymbol{y} \mid f_X)$, 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

    $R \leftarrow \text{CHOLESKY}(I + W^{1/2} K_{XX} W^{1/2})$ // 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 $-\frac{1}{2} a^\intercal f + \sum_i \log \sigma(y_i f_i)$

  return f, W, R, r

end procedure