Advanced Machine Learning
Gaussian processes
Lecture 3

Mattias Villani

Division of Statistics and Machine Learning
Department of Computer and Information Science
Linköping University

LECTURE OVERVIEW

- Lecture 3
  - **Gaussian Process Classification**
  - **More GP models**

Classification with logistic regression

- **Classification**: **binary response** $y \in \{-1, 1\}$ predicted by features $\mathbf{x}$.

- Example: linear logistic regression

$$Pr(y = 1|\mathbf{x}) = \lambda(\mathbf{x}^T\mathbf{w})$$

where $\lambda(z)$ is the logistic **link function**

$$\lambda(z) = \frac{1}{1 + \exp(-z)}$$

- $\lambda(z)$ 'squashes' the linear prediction $\mathbf{x}^T\mathbf{w} \in \mathbb{R}$ into $\lambda(\mathbf{x}^T\mathbf{w}) \in [0, 1]$.
- Logistic regression has **linear decision boundaries**.

GP CLASSIFICATION

- Obvious **GP extension** of logistic regression: replace $\mathbf{x}^T\mathbf{w}$ by $f(\mathbf{x})$ where

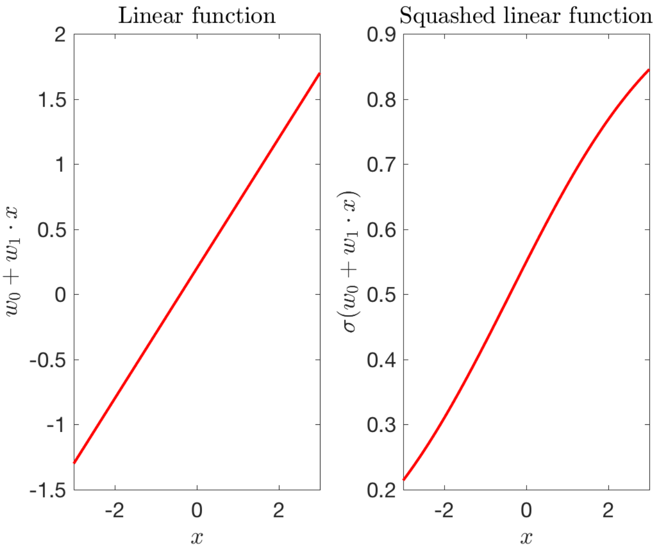
$$f(\mathbf{x}) \sim GP(0, k(\mathbf{x}, \mathbf{x}'))$$

  and squash f through logistic function (or normal CDF)

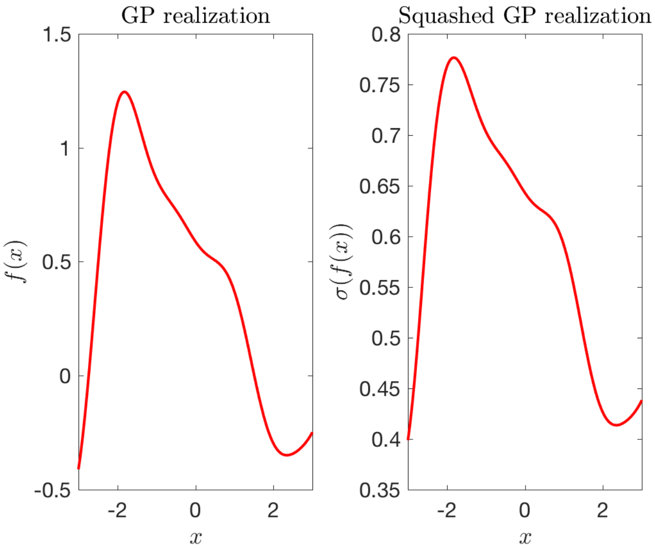
$$Pr(y = 1|\mathbf{x}) = \lambda(f(\mathbf{x}))$$

- Decision boundaries are now non-parametric (GP). Flexible.

SQUASHING A LINEAR FUNCTION



Squashing a GP function



GP classification - inference

- **Prediction** for a test case $\mathbf{x}_*$:

$$Pr(y_* = +1|\mathbf{X}, \mathbf{y}, \mathbf{x}_*) = \int \sigma(f_*) p(f_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y}) df_*$$

  where $\sigma(f_*)$ is some sigmoidal function (logistic, normal CDF...) and f_* is the latent f at the test input $\mathbf{x}_*$.
- The posterior distribution of f_* is

$$p(f_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y}) = \int p(f_*|\mathbf{X}, \mathbf{x}_*, \mathbf{f}) p(\mathbf{f}|\mathbf{X}, \mathbf{y}) d\mathbf{f}$$

  where

$$p(\mathbf{f}|\mathbf{X}, \mathbf{y}) \propto p(\mathbf{y}|\mathbf{f}) p(\mathbf{f}|\mathbf{X})$$

  is the posterior of $\mathbf{f}$ from the training data.
- Note that $p(\mathbf{y}|\mathbf{f})$ is no longer Gaussian in classification problems. Posterior $p(\mathbf{f}|\mathbf{X}, \mathbf{y})$ is not analytically tractable.

The Laplace approximation

- Approximates $p(\mathbf{f}|\mathbf{X},\mathbf{y})$ with $N(\hat{\mathbf{f}},\mathbf{A}^{-1})$, where $\hat{\mathbf{f}}$ is the posterior mode and $\mathbf{A}$ is the negative Hessian of the log posterior at $\mathbf{f}=\hat{\mathbf{f}}$.
- The log posterior is (proportional to)

$$
\begin{aligned}
\Psi(\mathbf{f}) &= \log p(\mathbf{y}|\mathbf{f}) + \log p(\mathbf{f}|\mathbf{X}) \\
&= \log p(\mathbf{y}|\mathbf{f}) - \frac{1}{2}\mathbf{f}^T K^{-1}\mathbf{f} - \frac{1}{2}\log|K| - \frac{n}{2}\log 2\pi
\end{aligned}
$$

- Differentiating wrt $\mathbf{f}$

$$
\begin{aligned}
\nabla\Psi(\mathbf{f}) &= \nabla \log p(\mathbf{y}|\mathbf{f}) - K^{-1}\mathbf{f} \\
\nabla\nabla\Psi(\mathbf{f}) &= \nabla\nabla \log p(\mathbf{y}|\mathbf{f}) - K^{-1} = -W - K^{-1}
\end{aligned}
$$

  where W is a diagonal matrix since each y_i only depends on its f_i.
- Use **Newton's method** to iterate to the mode.
- **Approximate predictions** of f_* are **possible**.
- Predictions of y_* require one-dimensional numerical integration.

Markov Chain Monte Carlo

- Metropolis-Hastings (or Hamiltonian MC) to **sample from training posterior**

$$\mathbf{f}|\mathbf{x},\mathbf{y},\theta$$

Produces $\mathbf{f}^{(1)}, ..., \mathbf{f}^{(N)}$ draws.

- For each $\mathbf{f}^{(i)}$, **sample the test posterior** $\mathbf{f}_*$ from

$$\mathbf{f}_*|\mathbf{f}^{(i)},\mathbf{x},\mathbf{x}_* \sim N\left(K(\mathbf{x}_*,\mathbf{x})K(\mathbf{x},\mathbf{x})^{-1}\mathbf{f}^{(i)}, K(\mathbf{x}_*,\mathbf{x}_*) - K(\mathbf{x}_*,\mathbf{x})K(\mathbf{x},\mathbf{x})^{-1}\right.$$

Note that this does not depend on $\mathbf{y}$ since we condition on $\mathbf{f}$. Noise-free GP fit. Produces $\mathbf{f}_*^{(1)}, ..., \mathbf{f}_*^{(N)}$ draws.

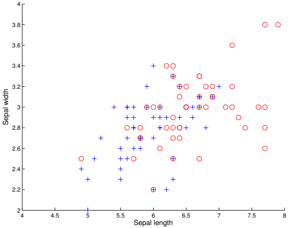
- For each $\mathbf{f}_*^{(i)}$, **sample a prediction** from

$$p(\mathbf{y}_*|\mathbf{f}_*^{(i)},\theta).$$

Produces a draws from the predictive distribution $p(\mathbf{y}_*|\mathbf{x}_*,\mathbf{x},\mathbf{y},\theta)$.

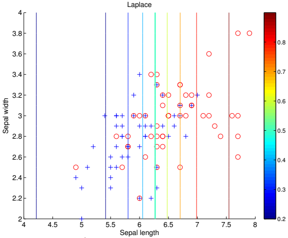
- Straightforward (at least in principle) to also **sample the hyperparameters** θ. Slice sampling.

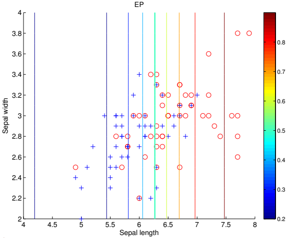
IRIS DATA - SEPAL - SE KERNEL WITH ARD



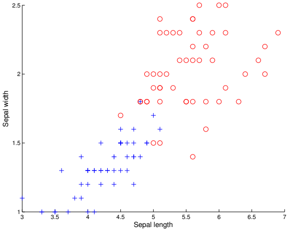
$$\text{Laplace: } \hat{\ell}_1 = 1.7214, \hat{\ell}_2 = 185.5040, \sigma_f = 1.4361$$

$$\text{EP: } \hat{\ell}_1 = 1.7189, \hat{\ell}_2 = 55.5003, \sigma_f = 1.4343$$



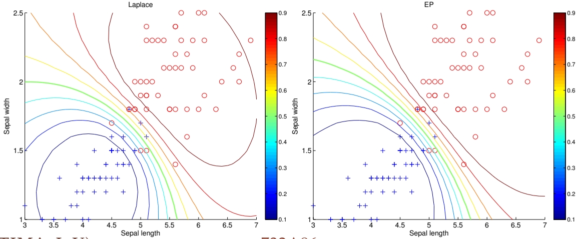


Iris data - petal - SE kernel with ARD

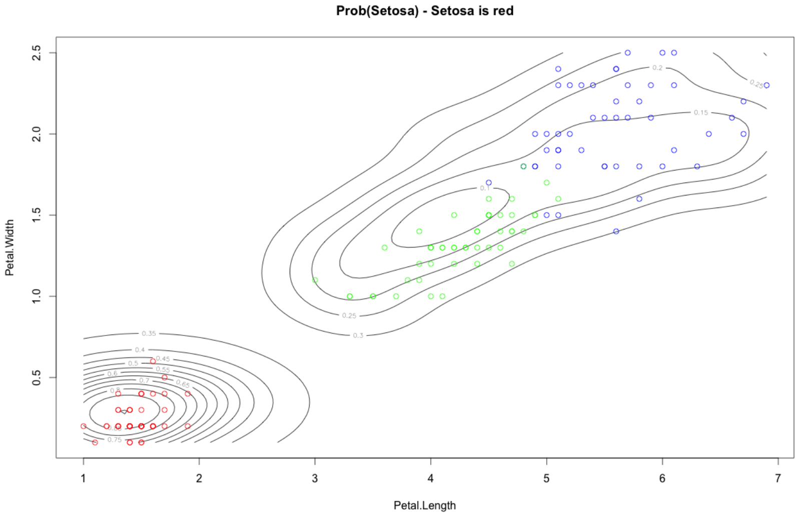


Laplace: $\hat{\ell}_1 = 1.7606, \hat{\ell}_2 = 0.8804, \sigma_f = 4.9129$

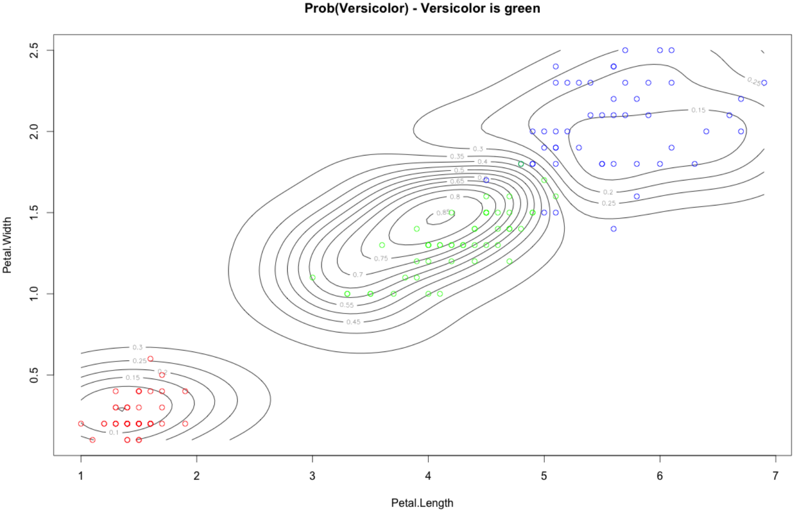
EP: $\hat{\ell}_1 = 2.1139, \hat{\ell}_2 = 1.0720, \sigma_f = 5.3369$



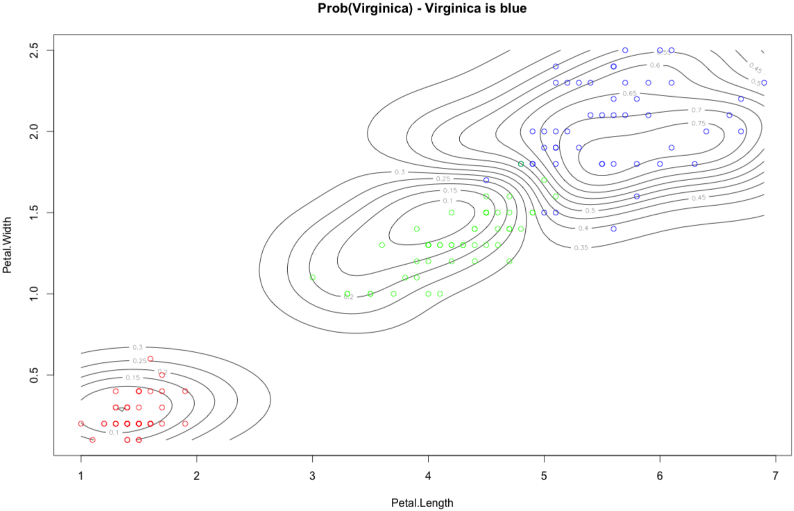
Iris data - petal - all three classes



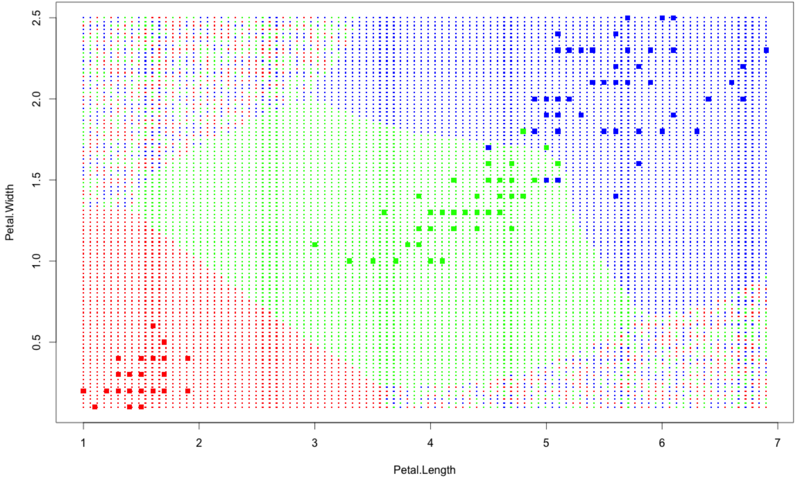
Iris data - petal - all three classes



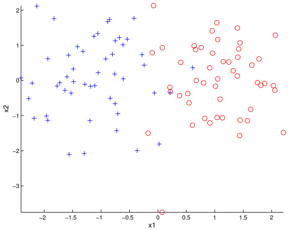
IRIS DATA - PETAL - ALL THREE CLASSES



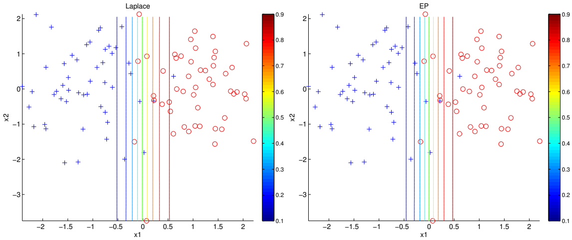
Iris data - petal - decision boundaries



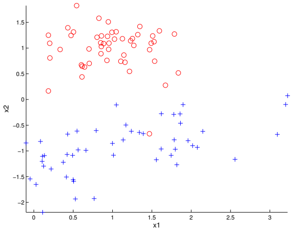
Toy data 1 - SE kernel with ARD



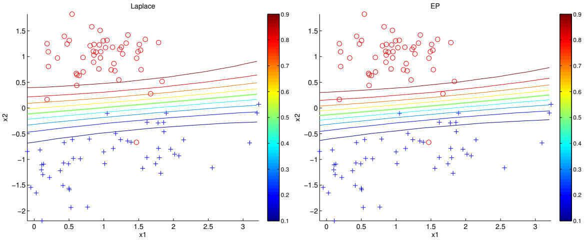
$$\text{EP: } \hat{\ell}_1 = 2.4503, \hat{\ell}_2 = 721.7405, \sigma_f = 4.7540$$



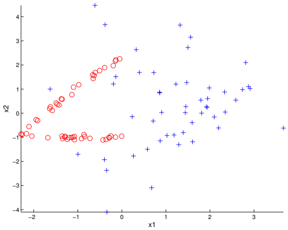
TOY DATA 2 - SE KERNEL WITH ARD



$$\text{EP: } \hat{\ell}_1 = 8.3831, \hat{\ell}_2 = 1.9587, \sigma_f = 4.5483$$

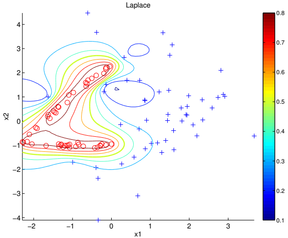


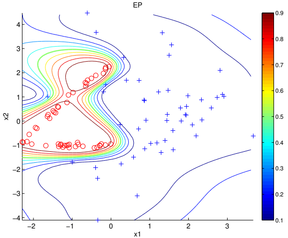
Toy data 3 - SE kernel with ARD



Laplace: $\hat{\ell}_1 = 0.7726, \hat{\ell}_2 = 0.6974, \sigma_f = 11.7854$

EP: $\hat{\ell}_1 = 1.2685, \hat{\ell}_2 = 1.0941, \sigma_f = 17.2774$





Gaussian Process Optimization (GPO)

- **Aim**: minimization of **expensive** function

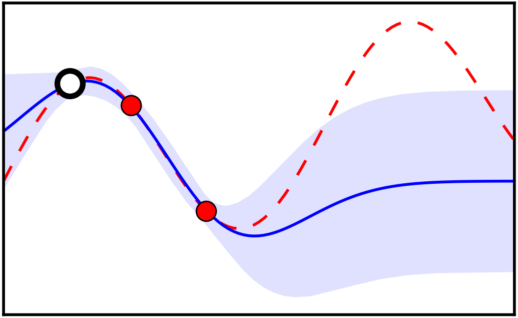
$$\text{argmin}_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x})$$

- Typical applications: **hyperparameter estimation**.

- **GPO idea**:
  - Assign GP prior to the unknown function f.
  - Evaluate the function at some values $x_1, x_2, ..., x_n$.
  - Update to posterior $f|x_1, ..., x_n \sim GP(\mu, K)$. Noise-free model.
  - Use the GP posterior of f to find a new evaluation point x_{n+1}. **Explore** vs **Exploit**.
  - Iterate until the change in optimum is lower that some tolerance.

- **Bayesian Optimization**. Bayesian Numerics. Probabilistic numerics.

Explore-Exploit Illustration



Acquisition functions

- **Probability of Improvement** (**PI**)

$$a_{PI}\left(\mathbf{x};\{\mathbf{x}_n,y_n\},\theta\right)\equiv \Pr\left(f(\mathbf{x})<f(\mathbf{x}_{best})\right)=\Phi(\gamma(\mathbf{x}))$$

  where

$$\gamma(\mathbf{x})=\frac{f(\mathbf{x}_{best})-\mu\left(\mathbf{x};\{\mathbf{x}_n,y_n\},\theta\right)}{\sigma\left(\mathbf{x};\{\mathbf{x}_n,y_n\},\theta\right)}$$

- **Expected Improvement** (**EI**)

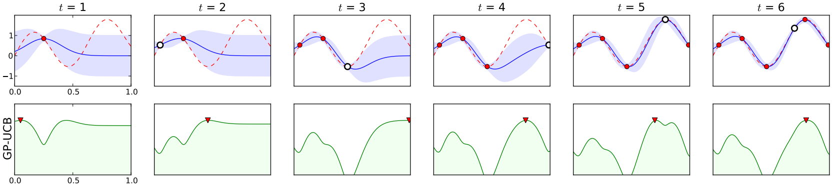
$$a_{EI}\left(\mathbf{x};\{\mathbf{x}_n,y_n\},\theta\right)=\sigma\left(\mathbf{x};\{\mathbf{x}_n,y_n\},\theta\right)\left[\gamma(\mathbf{x})\Phi(\gamma(\mathbf{x}))+\mathcal{N}\left(\gamma(\mathbf{x});0,1\right)\right]$$

- **Lower Confidence Bound** (**LCB**)

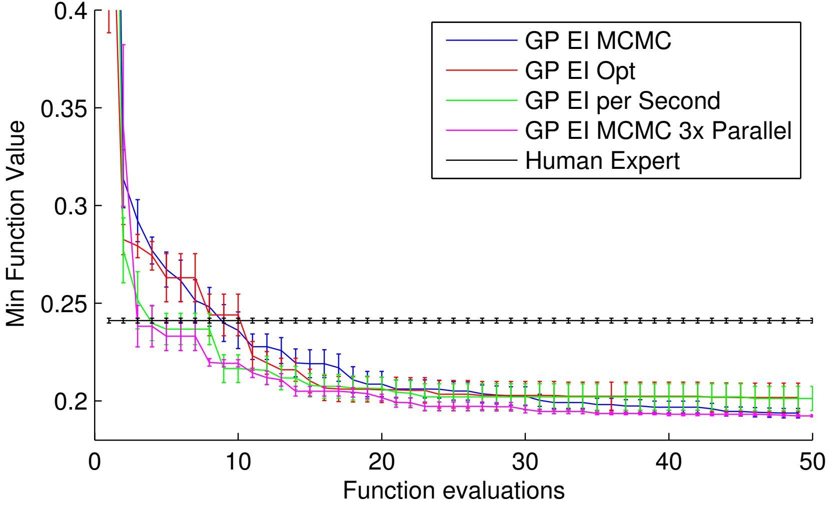
$$a_{EI}\left(\mathbf{x};\{\mathbf{x}_n,y_n\},\theta\right)=\mu\left(\mathbf{x};\{\mathbf{x}_n,y_n\},\theta\right)-\kappa\cdot\sigma\left(\mathbf{x};\{\mathbf{x}_n,y_n\},\theta\right)$$

- Note: need to maximize the acquisition function to choose $\mathbf{x}_{next}$. Non-convex, but cheaper and simpler than original problem.

Acquisition functions from Brochu et al



ConvNets - Snoek et al (NIPS, 2012)



More GP models

- **Heteroscedastic GP regression**

$$y = f(x) + \exp\left[g(x)\right]\epsilon$$

so where $f \sim GP\left[m_f(x), k_f(x, x^{'})\right]$ independently of $g \sim GP\left[m_g(x), k_g(x, x^{'})\right]$.

- GP for **density estimation**

$$p(x) = \frac{\exp\left[f(x)\right]}{\int_{\mathbb{R}} \exp\left[f(t)\right] dt}$$

where $f \sim GP\left[m(x), k(x, x^{'})\right]$. Appealing mean function: $m(x) = -\frac{1}{2\theta_2}(x - \theta_1)^2$ [i.e. best guess is a normal density].

- **Shared latent GP** for **dependent multivariate data** ($k \ll p$)

$$\begin{pmatrix} y_1(\mathbf{x}) \\ \vdots \\ y_p(\mathbf{x}) \end{pmatrix} = \underset{p \times k}{\mathbf{L}} \begin{pmatrix} f_1(\mathbf{x}) \\ \vdots \\ f_k(\mathbf{x}) \end{pmatrix} + \begin{pmatrix} g_1(\mathbf{x}) \\ \vdots \\ g_p(\mathbf{x}) \end{pmatrix} + \begin{pmatrix} \epsilon_1 \\ \vdots \\ \epsilon_p \end{pmatrix}$$