

A unified framework for closed-form nonparametric regression, classification, preference and mixed problems with Skew Gaussian Processes

Alessio Benavoli
Associate Professor
Discipline of Statistics and Information Systems
SCSS
alessio.benavoli@tcd.ie

Wednesday 10th February, 2021

Overview

- 1 Gaussian Process Regression
- 2 Gaussian Process Classification
- 3 Skew Gaussian Process
- 4 Bayesian Optimisation
- 5 Bayesian Preferential Optimisation

Bayesian Linear Regression

Consider a 1D linear regression model:

$$y = a + bx + \text{noise}$$

and assume *noise* is Gaussian with zero mean and known variance σ^2 .

Given iid data = $\{(x_i, y_i) : i = 1, \dots, n\}$, and the probabilistic model:

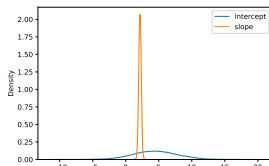
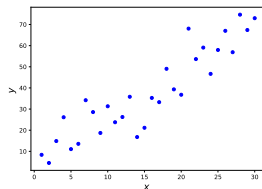
Likelihood: $p(\text{data} | a, b) = \prod_{i=1}^n N(y_i; a + bx_i, \sigma^2)$

Prior: $p(a, b) = N(a; 0, \sigma_a^2)N(b; 0, \sigma_b^2)$

the posterior

$$p(a, b | \text{data})$$

can be obtained in closed-form by Bayes' rule and is a bivariate Gaussian.



Predictive posterior

Given the Gaussian posterior,

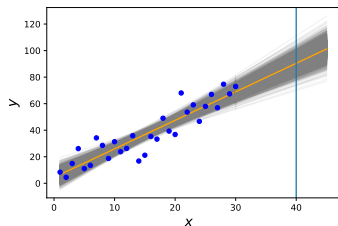
$$p(a, b | \text{data})$$

we can sample a, b and predict.

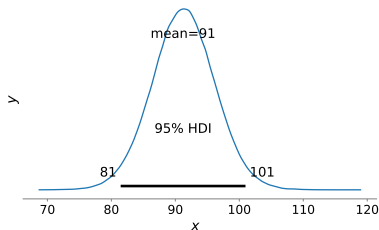
We can focus on the predictive posterior at x^* :

$$p(a + bx^* | \text{data})$$

which is also Gaussian.



Prediction at $x=40$



Equivalent form

We considered this prior

$$\text{Prior: } p(a, b) = N(a; 0, \sigma_a^2)N(b; 0, \sigma_b^2)$$

If we define

$$f(x) := a + bx.$$

Since f depends linearly on a, b , which are normal distributed, it follows that f is also normal distributed with

$$\text{mean: } E(f(x)) = E(a) + E(b)x = 0$$

$$\text{covariance: } E(f(x_i)f(x_j)) = E((a + bx_1)(a + bx_2)) = \sigma_a^2 + \sigma_b^2 x_1 x_2 \quad \forall x_1, x_2$$

Equivalent form

Denote the covariance as:

$$\Omega(x_i, x_j) := E(f(x_i)f(x_j))$$

and let X be the vector $[x_1, \dots, x_n]^T$, we define the covariance matrix:

$$\Omega(X, X) := \begin{bmatrix} \Omega(x_1, x_1) & \Omega(x_1, x_2) & \dots & \Omega(x_1, x_n) \\ \Omega(x_2, x_1) & \Omega(x_2, x_2) & \dots & \Omega(x_2, x_n) \\ \vdots & \vdots & \ddots & \vdots \\ \Omega(x_n, x_1) & \Omega(x_n, x_2) & \dots & \Omega(x_n, x_n) \end{bmatrix}$$

and the vectors

$$Y := \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}, \quad f(X) := \begin{bmatrix} f(x_1) \\ \vdots \\ f(x_n) \end{bmatrix}$$

Bayesian Linear Regression

We can equivalently write/derive Bayesian linear regression in this way:

Prior:

$$p(f(X)) = N \left(\begin{bmatrix} f(x_1) \\ \vdots \\ f(x_n) \end{bmatrix}; \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} \Omega(x_1, x_1) & \Omega(x_1, x_2) & \dots & \Omega(x_1, x_n) \\ \Omega(x_2, x_1) & \Omega(x_2, x_2) & \dots & \Omega(x_2, x_n) \\ \vdots & \vdots & \ddots & \vdots \\ \Omega(x_n, x_1) & \Omega(x_n, x_2) & \dots & \Omega(x_n, x_n) \end{bmatrix} \right)$$

Likelihood:

$$p(Y|f(X)) = \prod_{i=1}^n N(y_i; f(x_i), \sigma^2) = N(Y; f(X), I_n \sigma^2)$$

Posterior:

$p(f(X)|\text{data})$ (multivariate) Gaussian

The marginal on $f(x_i)$, for any $x_i \in X$, is equal to

$$p(f(x_i)|\text{data}) \left(= p(a + bx_i|\text{data}) \right)$$

Posterior predictive

Issue: from the previous model, we cannot compute

$$p(f(x^*)|\text{data})$$

for $x^* \notin X$, we only have the posterior on $f(X)$. However, if we start with $X' = [X^T, x^*]^T$ and the model:

Prior:

$$p(f(X')) = N(f(X'); 0, \Omega(X', X'))$$

Likelihood:

$$p(Y|f(X)) = N(Y; f(X), I_n \sigma^2)$$

Posterior:

$$p(f(X')|\text{data}) \text{ (multivariate) Gaussian}$$

We can then marginalise $p(f(X')|\text{data})$ to obtain

$$p(f(x^*)|\text{data})$$

Posterior predictive

The predictive distribution is also Gaussian:

$$p(f(x^*)|\text{data}) = N(f(x^*); \mu_p(x^*), \Omega_p(x^*, x^*))$$

with

$$\begin{aligned}\mu_p(x^*) &= \Omega(x^*, X)(\Omega(X, X) + \sigma^2 I_n)^{-1} Y \\ \Omega_p(x^*, x^*) &= \Omega(x^*, x^*) - \Omega(x^*, X)(\Omega(X, X) + \sigma^2 I_n)^{-1} \Omega(X, x^*)\end{aligned}$$

Note that, the mean and variance are functions of x^* .

Definition of GP

We have placed a prior on a function by specifying, for any finite set X' , the marginals:

$$f(X') \sim N(0, \Omega(X', X')).$$

By definition, this is a Gaussian Process:

A Gaussian process is a stochastic process (a collection of random variables indexed by $x \in \mathcal{X}'$), such that every finite collection of those random variables has a (consistent) multivariate normal distribution.

It is denoted as:

$$f \sim GP(0, \Omega(x, x')),$$

and it is completely defined by a mean function (zero in the example) and a covariance function.

This definition is consistent because the Gaussian distribution is closed to marginalisation and because

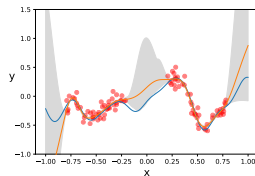
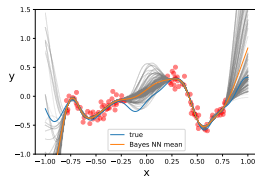
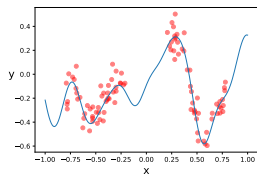
$$\Omega(X', X') > 0 \quad \forall X'$$

Why is this useful?

We can use GPs to perform closed-form Bayesian **nonlinear regression** by simply changing the kernel function

$$\Omega(x, x') = \sigma_r^2 e^{-\frac{\|x-x'\|}{2\ell^2}}, \quad \Omega(x, x') = \sigma_r^2 \frac{2}{\pi} \arcsin \left(\frac{\sigma_w^2 x^\top x' + \sigma_b^2}{\sqrt{\sigma_w^2 x^\top x + \sigma_b^2 + 1} \sqrt{\sigma_w^2 (x')^\top x' + \sigma_b^2 + 1}} \right)$$

which are called “square exponential” and NN kernel.



It also works for multivariate nonlinear regression, because $f(x) \in \mathbb{R}$, no matter what x is.

Everything is computed in closed-form!

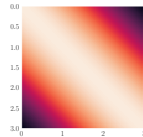
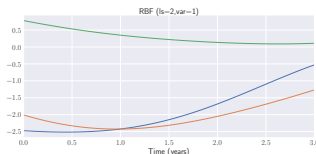
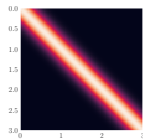
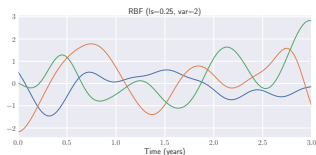
Hyperparameters

Kernels (covariance functions) have hyperparameters θ :

$$\Omega(x, x') = \sigma_r^2 e^{-\frac{\|x-x'\|}{2\ell^2}}, \quad \Omega(x, x') = \sigma_r^2 \frac{2}{\pi} \operatorname{asin} \left(\frac{\sigma_w^2 x^\top x' + \sigma_b^2}{\sqrt{\sigma_w^2 x^\top x + \sigma_b^2 + 1} \sqrt{\sigma_w^2 (x')^\top x' + \sigma_b^2 + 1}} \right)$$

and we also need to select the variance σ^2 of the Gaussian noise in the likelihood.

$$f \sim GP(0, \Omega(x, x'))$$



Hyperparameters

The marginal likelihood has a closed-form:

$$p(\text{data}|\theta) = N(Y; 0, \Omega(X, X) + \sigma^2 I_n)$$

which means we can “easily” learn θ via Bayesian model selection or MCMC.

Memory and Computational issues

Predictive posterior

$$p(f(x^*)|\text{data}) = N(f(x^*); \mu_p(x^*), \Omega_p(x^*, x^*))$$

with

$$\begin{aligned}\mu_p(x^*) &= \Omega(x^*, X)(\Omega(X, X) + \sigma^2 I_n)^{-1} Y \\ \Omega_p(x^*, x^*) &= \Omega(x^*, x^*) - \Omega(x^*, X)(\Omega(X, X) + \sigma^2 I_n)^{-1} \Omega(X, x^*)\end{aligned}$$

The issue of GPs is that we need to build the covariance matrix

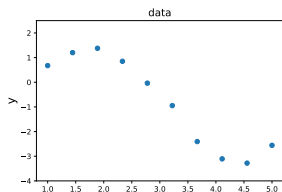
$$\Omega(X, X)$$

which has dimension n^2 (memory) and invert it, which costs $O(n^3)$ (time). This means that we cannot solve linear regression problems with 20000 data points, but there are approximations.¹

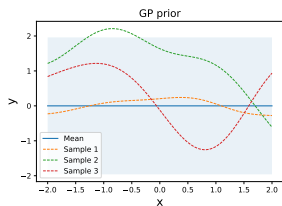
¹Schürch, M., Azzimonti, D., Benavoli, A., Zaffalon, M. "Recursive estimation for sparse Gaussian process regression." *Automatica*, 2020.

Summarizing: GP Regression

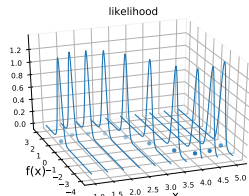
$$D = \{(x_i, y_i)\}_{i=1}^n$$



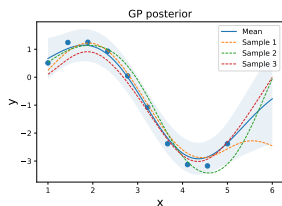
$$f \sim GP(\mu(x), \Omega(x, x'))$$



$$N(y_i; f(x_i), \sigma^2)$$



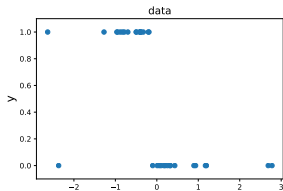
$$f|D \sim GP(\mu_p(x), \Omega_p(x, x'))$$



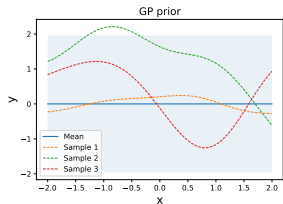
The success of Gaussian Processes for regression comes from the closed form solution of the posterior ($O(n^3)$ time, $O(n^2)$ memory complexity).

Gaussian Process Classification

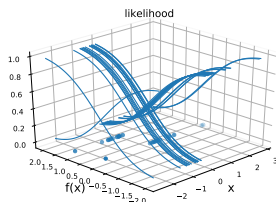
$$D = \{(x_i, y_i)\}_{i=1}^n$$



$$f \sim GP(\mu(x), \Omega(x, x'))$$



$$\Phi(f(x_i)) = \int_{-\infty}^{f(x_i)} N(z, 0, 1) dz$$



$$f|D \sim ?$$

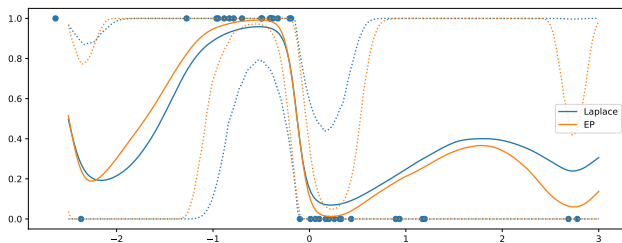
The posterior is not a Gaussian Processes!

Main two approximations

For binary classification, the likelihood is not conjugate to the prior and, therefore, the posterior is not a GP.

The posterior process can be approximated by a GP:

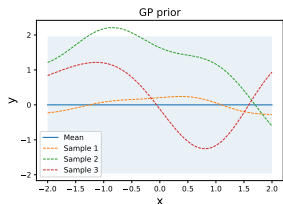
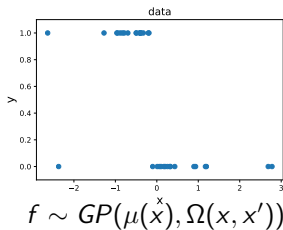
- Laplace's approximation;
- Expectation propagation (EP).



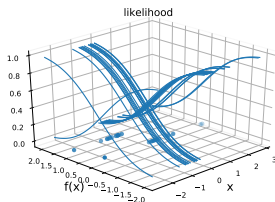
The predictive posterior probability $p(y = 1|x^*, D)$ using a RBF kernel.

New result

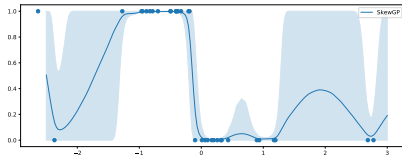
$$D = \{(x_i, y_i)\}_{i=1}^n$$



$$\Phi(f(x_i)) = \int_{-\infty}^{f(x_i)} N(x, 0, 1) dx$$



$$f|D \sim \text{SkewGP}(\xi_p, \Omega_p, \Delta_p, \gamma_p, \Gamma_p)$$

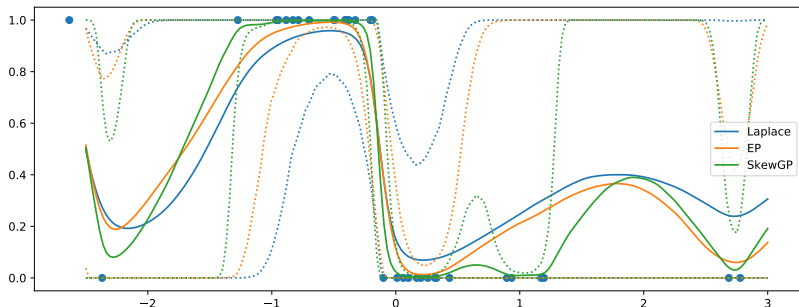


The posterior is a SkewGP²

²Benavoli, A., Azzimonti, D., and Piga, D. (2020b). "Skew Gaussian Processes for Classification." Machine Learning, 109:1877–1902.

Comparison

Laplace vs. EP vs. Exact (SkewGP)

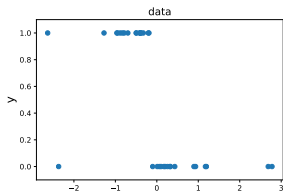


Comparison on 130 classification datasets³

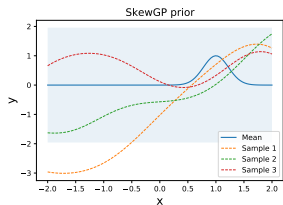
³Benavoli, A., Azzimonti, D., and Piga, D. (2020b). "Skew Gaussian Processes for Classification." *Machine Learning*, 109:1877–1902.

More general

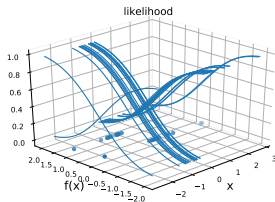
$$D = \{(x_i, y_i)\}_{i=1}^n$$



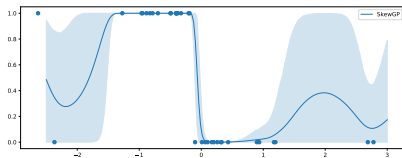
$$f \sim \text{SkewGP}_s(\xi, \Omega, \Delta, \gamma, \Gamma)$$



$$\Phi(f(x_i)) = \int_{-\infty}^{f(x_i)} N(x, 0, 1) dx$$



$$f|D \sim \text{SkewGP}(\xi_p, \Omega_p, \Delta_p, \gamma_p, \Gamma_p)$$



The posterior is a Skew Gaussian Process and predictions can be computed in $O(n^3)$ time, $O(n^2)$ memory complexity.

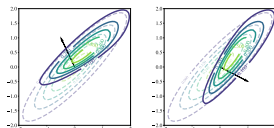
What is a Unified Skew-Normal distribution (SUN)?

A vector $z \in \mathbb{R}^p$ is said to have a SUN distribution with latent skewness dimension s , $z \sim \text{SUN}_{p,s}(\xi, \Omega, \Delta, \gamma, \Gamma)$, if its PDF is:

$$p(z) = \phi_p(z - \xi; \Omega) \frac{\Phi_s(\gamma + \Delta^T \bar{\Omega}^{-1} D_{\Omega}^{-1}(z - \xi); \Gamma - \Delta^T \bar{\Omega}^{-1} \Delta)}{\Phi_s(\gamma; \Gamma)} \quad (1)$$

where

- $\phi_p(z - \xi; \Omega)$ is the PDF of a multivariate Normal distribution with mean $\xi \in \mathbb{R}^p$ and covariance $\Omega = D_{\Omega} \bar{\Omega} D_{\Omega} \in \mathbb{R}^{p \times p}$ with $\bar{\Omega}$ being a correlation matrix and D_{Ω} a diagonal matrix containing the square root of the diagonal elements in Ω .
- $\Phi_s(m; M)$ denotes the CDF of $N_s(0, M)$ evaluated at $m \in \mathbb{R}^s$.



The parameters $\gamma \in \mathbb{R}^s$, $\Gamma \in \mathbb{R}^{s \times s}$, $\Delta^{p \times s}$ control the skewness.

What is a SkewGP?

We say that a real function $f : \mathbb{R}^p \rightarrow \mathbb{R}$ is distributed as a skew-Gaussian process with latent dimension s ,

$$f \sim \text{SkewGP}_s(\xi, \Omega, \Delta, \gamma, \Gamma)$$

if, for any sequence of n points $\mathbf{x}_1, \dots, \mathbf{x}_n \in \mathbb{R}^p$, the vector $f(X) = [f(\mathbf{x}_1), \dots, f(\mathbf{x}_n)]^T$ is SUN distributed with parameters γ, Γ , location, scale and skewness matrices, respectively, given by

$$\xi(X) := \begin{bmatrix} \xi(\mathbf{x}_1) \\ \xi(\mathbf{x}_2) \\ \vdots \\ \xi(\mathbf{x}_n) \end{bmatrix}, \quad \Omega(X, X) := \begin{bmatrix} \Omega(\mathbf{x}_1, \mathbf{x}_1) & \Omega(\mathbf{x}_1, \mathbf{x}_2) & \dots & \Omega(\mathbf{x}_1, \mathbf{x}_n) \\ \Omega(\mathbf{x}_2, \mathbf{x}_1) & \Omega(\mathbf{x}_2, \mathbf{x}_2) & \dots & \Omega(\mathbf{x}_2, \mathbf{x}_n) \\ \vdots & \vdots & \dots & \vdots \\ \Omega(\mathbf{x}_n, \mathbf{x}_1) & \Omega(\mathbf{x}_n, \mathbf{x}_2) & \dots & \Omega(\mathbf{x}_n, \mathbf{x}_n) \end{bmatrix},$$
$$\Delta(X) := \begin{bmatrix} \Delta(\mathbf{x}_1) & \Delta(\mathbf{x}_2) & \dots & \Delta(\mathbf{x}_n) \end{bmatrix}.$$

What is a SkewGP?

The process is well-defined provided that the matrix

$$M = \begin{bmatrix} \Gamma & \Delta(X) \\ \Delta(X) & \bar{\Omega}(X, X) \end{bmatrix}$$

is **positive definite**.

Note that a SkewGP

$$f \sim \text{SkewGP}_s(\xi, \Omega, \Delta, \gamma, \Gamma)$$

reduces to a GP when either the latent dimension is zero or when $\Delta(X) = 0$.

Binary Classification

Consider the training data $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$, where $\mathbf{x}_i \in \mathbb{R}^p$ and $y_i \in \{0, 1\}$. We assume that

$$f \sim \text{SkewGP}(\xi, \Omega, \Delta, \gamma, \Gamma)$$

and consider a *probit* model for the likelihood:

$$p(\mathcal{D}|f) = \prod_{i=1}^n \Phi((2y_i - 1)f(\mathbf{x}_i); 1) = \Phi_n(Wf(X); I_n),$$

where $W = \text{diag}(2y_1 - 1, \dots, 2y_n - 1)$.

Binary Classification with SkewGP prior

The (predictive) posterior of $f(\mathbf{x}^*)$ is SkewGP with mean, covariance and skewness functions:

$$\begin{aligned}\xi_p(\mathbf{x}^*) &= \xi(\mathbf{x}^*) \\ \Omega_p(\mathbf{x}^*, \mathbf{x}^*) &= \Omega(\mathbf{x}^*, \mathbf{x}^*), \\ \Delta_p(\mathbf{x}^*) &= [\Delta(\mathbf{x}^*) \quad D_{\Omega(\mathbf{x}^*, \mathbf{x}^*)}^{-1} \Omega(\mathbf{x}^*, X) W^T],\end{aligned}$$

and parameters

$$\begin{aligned}\gamma_p &= [\gamma, \quad W\xi(X)]^T, \\ \Gamma_p &= \begin{bmatrix} \Gamma & \Delta(X)^T D_{\Omega(X, X)} W^T \\ W D_{\Omega(X, X)} \Delta(X) & (W \Omega(X, X) W^T + I_n) \end{bmatrix}\end{aligned}$$

Binary Classification with GP prior

The (predictive) posterior of $f(x)$ for a GP classification problem (that is probit likelihood and GP prior):

$$\xi_p(x) = \xi(x)$$

$$\Omega_p(x, x) = \Omega(x, x),$$

$$\Delta_p(x) = \left[\cancel{\Delta(x)} \quad D_{\Omega(x, X)}^{-1} \Omega(x, X) W^T \right],$$

$$\gamma_p = \left[\cancel{X}, \quad W \xi(X) \right]^T,$$

$$\Gamma_p = \begin{bmatrix} \cancel{X} & \cancel{\Delta(X)^T D_{\Omega(x, X)} W^T} \\ \cancel{W D_{\Omega(x, X)} \Delta(X)} & (W \Omega(X, X) W^T + I_n) \end{bmatrix}$$

Hyperparameter learning and posterior sampling

The Marginal Likelihood (ML) is:

$$p(\mathcal{D}|\theta) = \frac{\Phi_{s+n}(\tilde{\gamma}; \tilde{\Gamma})}{\Phi_s(\gamma; \Gamma)}.$$

To compute the ML we need to compute a multivariate CDF of dimension n . To sample from the predictive posterior (this is necessary to compute $p(y = 1|x^*, D)$) we must be able to compute sample from a truncated n dimensional multivariate Gaussian distribution.

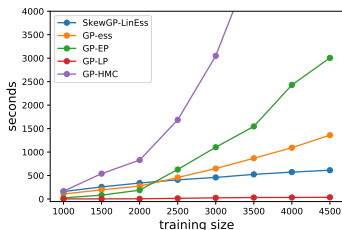
We can sample efficiently from the posterior using **linear elliptical slice sampling** ($O(n^3)$ time, $O(n^2)$ memory complexity). Moreover, we have provided a lower bound of $p(\mathcal{D})$ that allows us to compute efficiently an approximation of the ML.

Computational cost

Consider the independent random vectors $r_0 \sim \phi_p(0; \bar{\Omega} - \Delta\Gamma^{-1}\Delta^T)$ and $r_{1,-\gamma}$, the truncation below γ of $r_1 \sim \phi_s(0, \Gamma)$. Then the random variable

$$z_u = \xi + D_\Omega(r_0 + \Delta\Gamma^{-1}r_{1,-\gamma}), \quad (2)$$

Therefore, to compute the predictive SkewGP, we must find a way to sample from a multivariate truncated normal $r_{1,-\gamma}$.



All these algorithms have complexity $O(n^3)$ but the constant is different.

Nonparametric rest

Can we use the same approach for:

- nonlinear regression;
- logistic nonlinear regression (binary output);
- multinomial logistic nonlinear regression (categorical output);
- ordinal nonlinear regression;
- preference learning?

GPs are not conjugate with those likelihoods, but SkewGPs are!

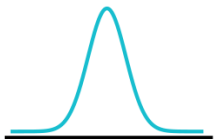

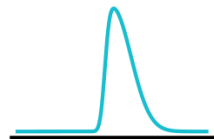
- Benavoli, A., Azzimonti, D., & Piga, D. (2021). “A unified framework for closed-form nonparametric regression, classification, preference and mixed problems with Skew Gaussian Processes”. arXiv:2012.06846.

Unification

regression

preference|classification

mixed

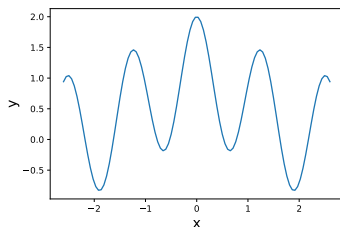
Likelihood		Normal	Affine Probit	Product
				
Posterior	Exact	SkewGP GP	SkewGP	SkewGP
	Approx.		GP	GP

Bayesian Optimisation

What is BO?

It is a methodology for global black-box optimisation of functions that are expensive to evaluate.

Imagine we want to find the maximum of this 1D function:



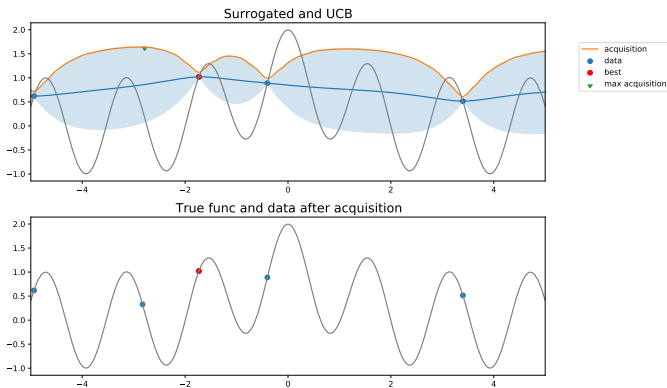
but we do not know the function: we can only evaluate it.

BO loop

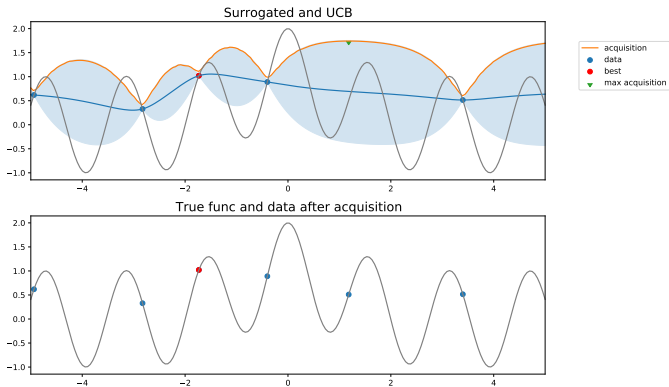
We start from some initial data points $\text{data} = \{(x_i, f(x_i)) \text{ for } i = 1, 2, 3\}$.

Loop:

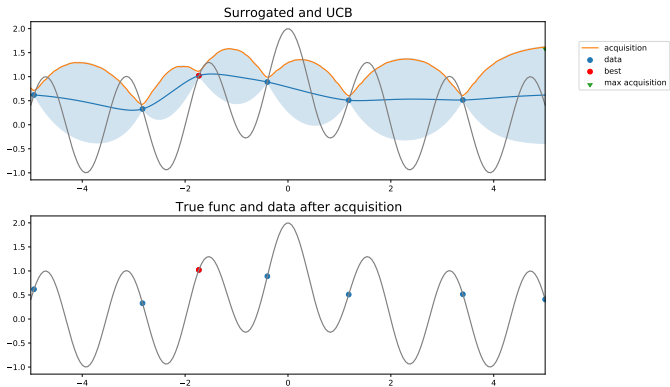
- 1 compute a GP regression model (surrogated model);
- 2 optimise an acquisition function to compute x_{next}
- 3 evaluate f at x_{next} and update $\text{data} = \text{data} \cup \{(x_{next}, f(x_{next}))\}$



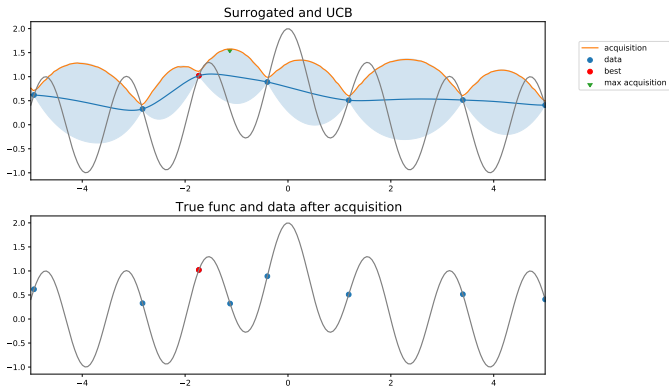
BO loop



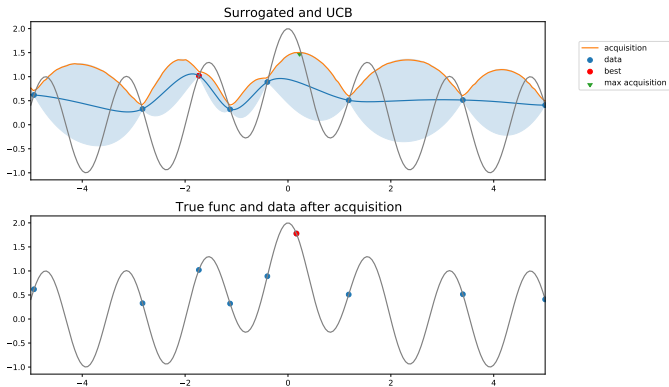
BO loop



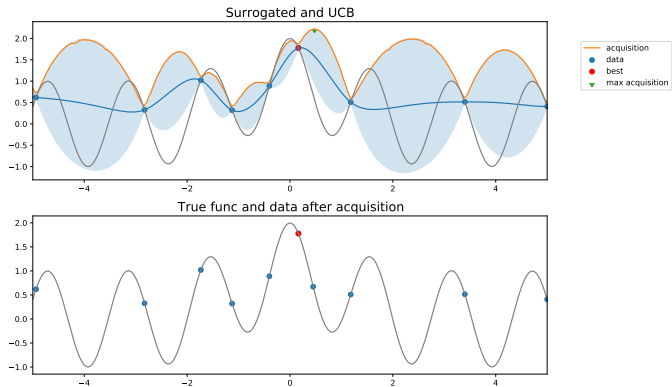
BO loop



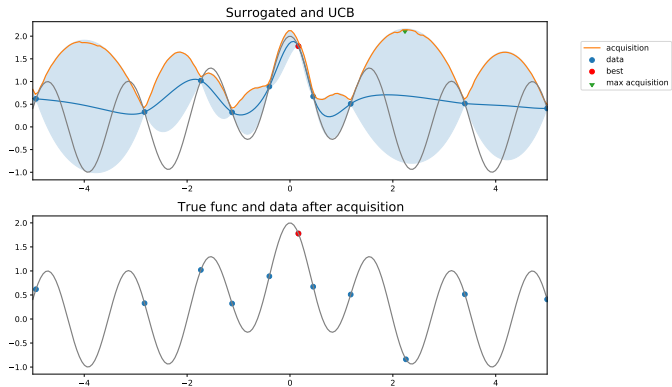
BO loop



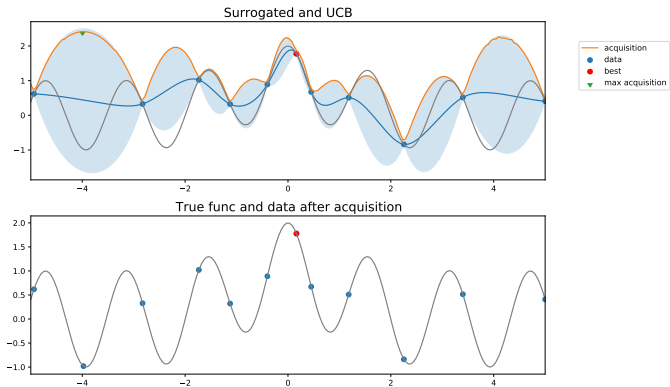
BO loop



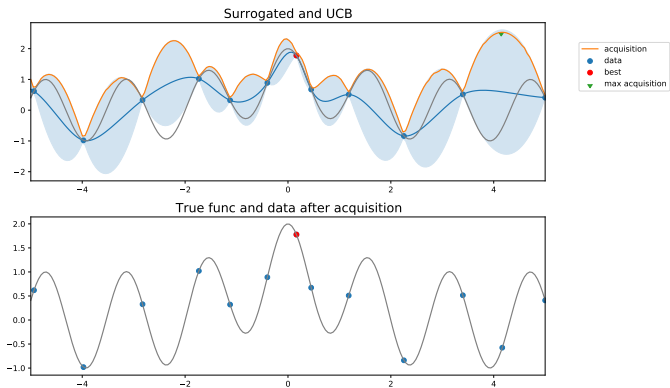
BO loop



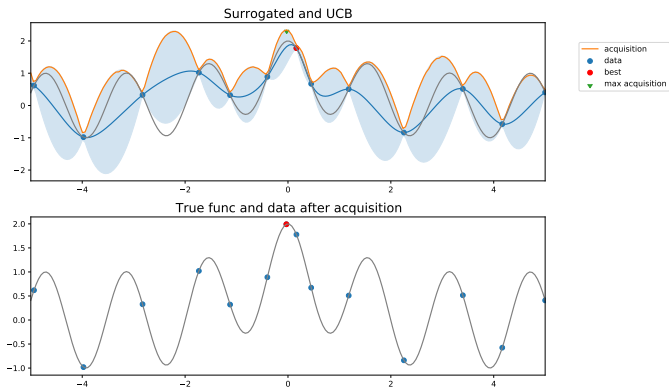
BO loop



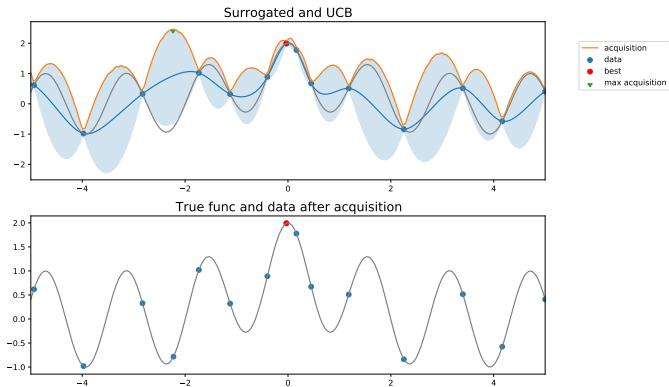
BO loop



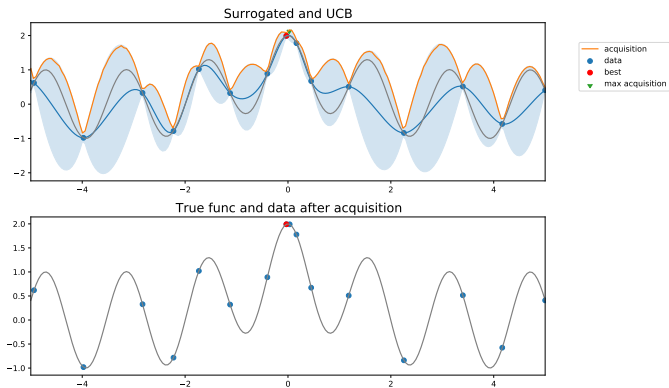
BO loop



BO loop

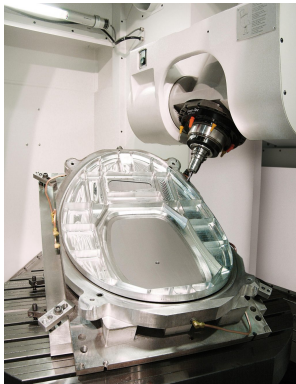
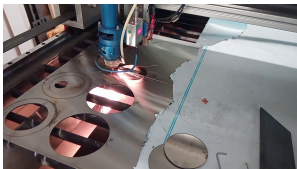


BO loop



Projects: smart Manufacturing

- Laser Cutting;
- Electrical Discharge Machine;
- 3D printing.



Goal: find the setting for the machine that optimises quality/speed/reliability

BO for coffee machine

Manufacturing process:

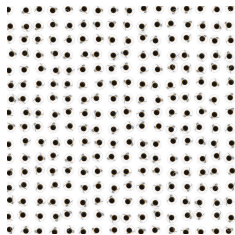


Setting:

$x = [\text{temp, press, coffee amount, coffee type}]$

we aim to make the best coffee.

DOE



How do we measure the quality of a cup of coffee?



Bayesian preferential BO

Before we consider the case that

$$\text{ev}_{x_{\text{next}}}(f) \rightarrow f(x_{\text{next}})$$

but there are situations where evaluating the function is difficult (costly):

$$\text{pref}_{x_{\text{next}}, x_{\text{ref}}}(f) = \begin{cases} x_{\text{next}} \succ x_{\text{ref}} & \text{if } f(x_{\text{next}}) > f(x_{\text{ref}}) \\ x_{\text{next}} \not\succeq x_{\text{ref}} & \text{if } f(x_{\text{next}}) \not> f(x_{\text{ref}}) \end{cases}$$

Likelihood:

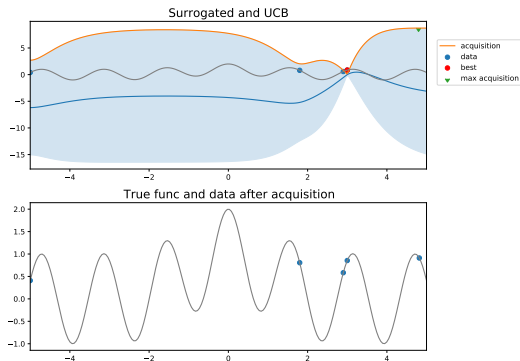
$$p(\text{data}|f) = \Phi\left(\frac{f(x_{\text{next}}) - f(x_{\text{ref}})}{\sigma}\right)^y \Phi\left(\frac{f(x_{\text{ref}}) - f(x_{\text{next}})}{\sigma}\right)^{1-y}$$

BO preferential loop

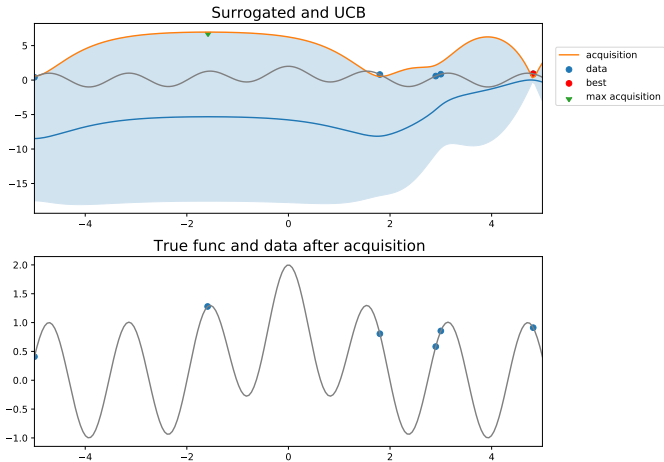
We start from some initial data points $\text{data} = \{x_1 \succ x_2, x_1 \succ x_3, x_1 \succ x_4\}$.

Loop:

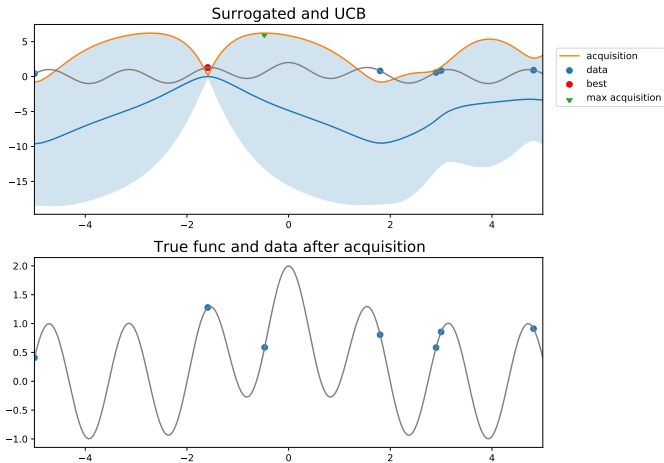
- 1 compute a SkewGP model (surrogated model);
- 2 optimise an acquisition function to compute x_{next}
- 3 query f for x_{next} versus x_{ref} and update $\text{data} = \text{data} \cup \{x_{next} \succ x_{ref}\}$



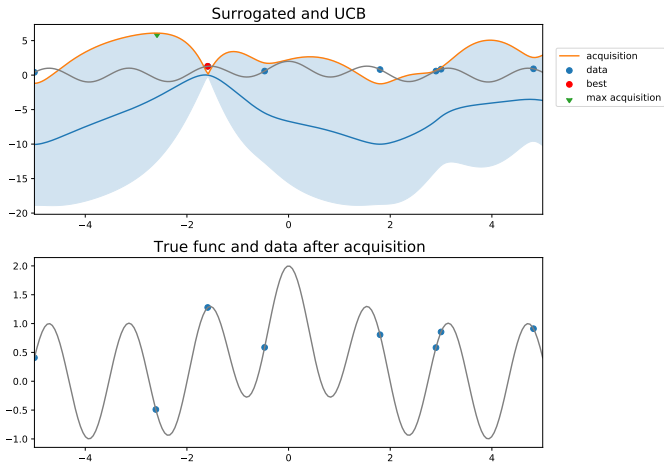
BO loop



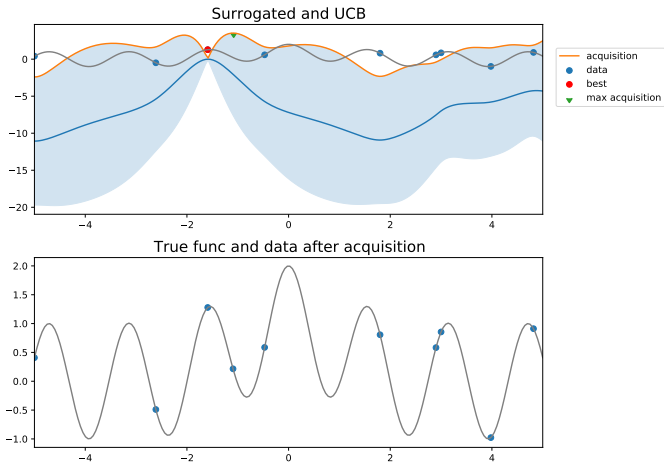
BO loop



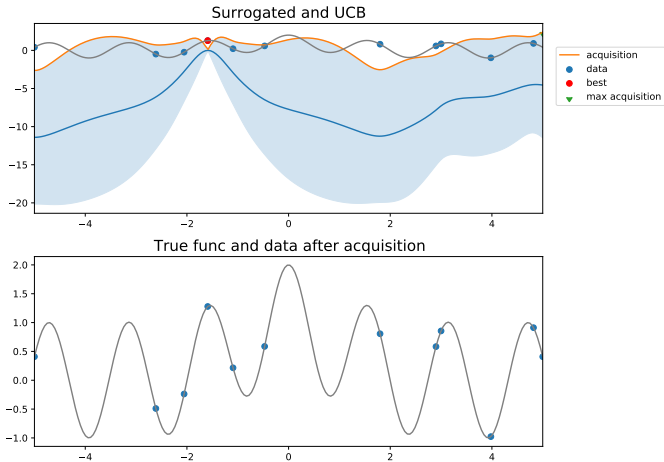
BO loop



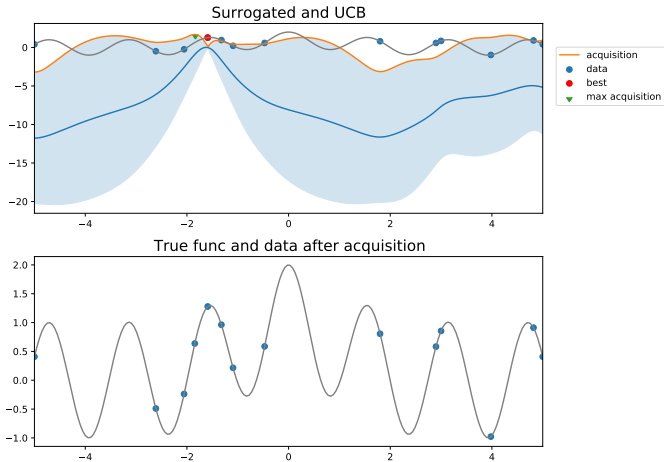
BO loop



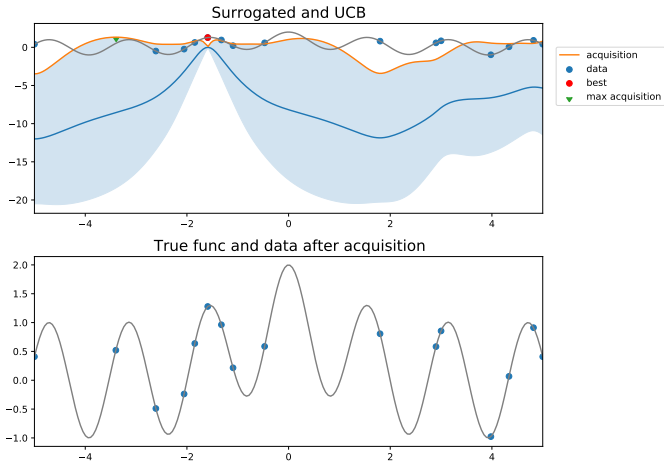
BO loop



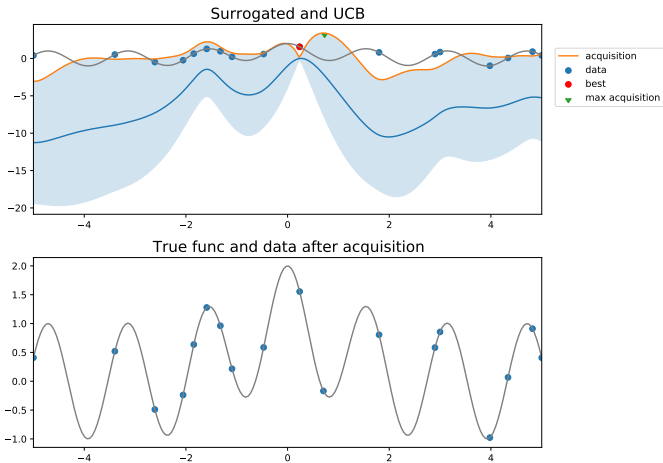
BO loop



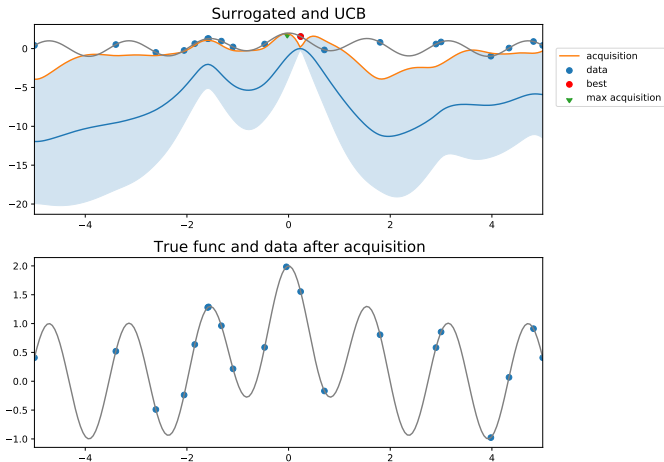
BO loop



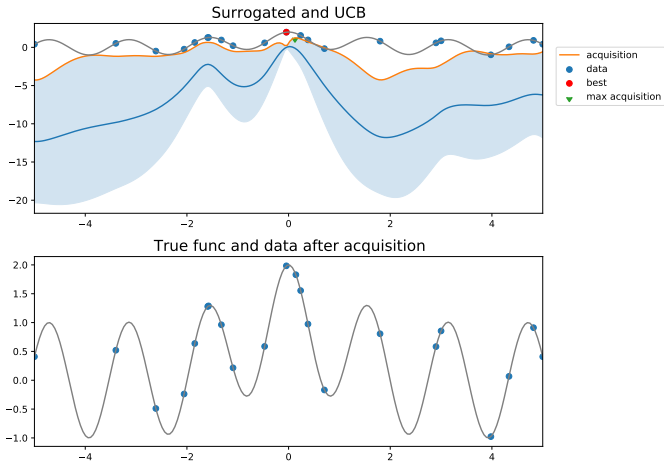
BO loop



BO loop



BO loop



Conclusions

- (Skew) Gaussian Processes and Bayesian Nonparametrics;
- Bayesian Hypothesis testing for comparing algorithms;
- Probabilistic Machine Learning;
- Probabilistic Programming (Stan, PyMC3);
- Foundation of rationality theories:
 - No Dutch book
 - No Arbitrage
 - Equilibrium Game theory
- Foundation of quantum theory.

Fashion MNIST dataset (Sneaker vs. rest)

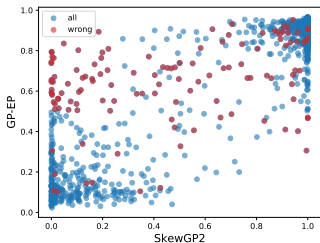


Fashion MNIST dataset (each image is $28 \times 28 = 784$ pixels and there are 10 classes).

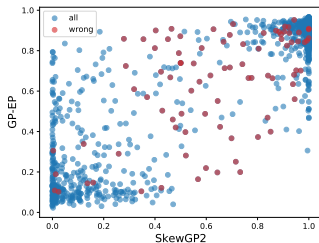
We randomly pooled 10000 images from the dataset and divided them into two sets, with 5000 cases for training and 5000 for testing.

For each one of the 10 classes, we have defined a binary classification sub-problem by considering one class against all the other classes.

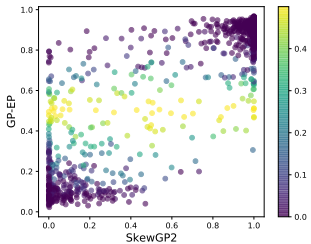
Fashion MNIST dataset (Sneaker vs. rest)



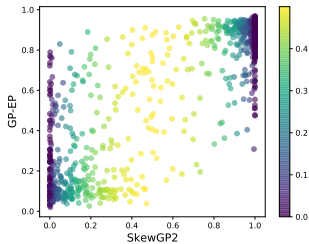
(a1)



(a2)



(b1)



(b2)