Chapter 3
The Surrogate Model

This Chapter presents the first key component of BO, that is, the probabilistic surrogate model. Section 3.1 is focused on Gaussian processes (GPs); Sect. 3.2 introduces the sequential optimization method known as Thompson sampling, also based on GP; finally, Sect. 3.3 presents other probabilistic models which might represent, in some cases, a suitable alternative to GP.

3.1 Gaussian Processes

Gaussian processes are a powerful formalism for implementing both regression and classification algorithms: we focus on regression. While most of the regression algorithms provide a deterministic output, GPs also offer a reliable estimate of uncertainty. This chapter presents the basic mathematics underlying this powerful tool.

3.1.1 Gaussian Processes Regression

One way to interpret a Gaussian process (GP) regression model is to think of it as defining a distribution over functions and with inference taking place directly in the space of functions (i.e. *function-space view*) (Williams and Rasmussen 2006). A GP is a collection of random variables, any finite number of which have a joint Gaussian distribution. A GP is completely specified by its mean function $\mu(x)$ and covariance function $\text{Cov}\big(f(x), f\big(x'\big)\big) = k\big(x, x'\big)$:

$$\mu(x) = \mathbb{E}[f(x)]$$
$$\text{Cov}\big(f(x), f\big(x'\big)\big) = k\big(x, x'\big) = \mathbb{E}\big[(f(x) - \mu(x))\big(f\big(x'\big) - \mu\big(x'\big)\big)\big]$$

and it is defined as:

F. Archetti and A. Candelieri, *Bayesian Optimization and Data Science*,
SpringerBriefs in Optimization, https://doi.org/10.1007/978-3-030-24494-1_3

$$f(x) \sim GP\big(\mu(x), k\big(x, x'\big)\big)$$

Usually, for notational simplicity we will take the prior of the mean function to be zero, although this is not necessary.

A simple example of a Gaussian process can be obtained from a Bayesian linear regression model $f(x) = \phi(x)^T w$ with prior $w = \mathcal{N}\big(0, \Sigma_p\big)$, where $\phi(x)$ and w are p-dimensional vectors. More precisely, $\phi(x)$ is a function mapping the d-dimensional vector x into a p-dimensional vector.

Thus, the equations for mean and covariance become:

$$\mathbb{E}[f(x)] = \phi(x)^T \mathbb{E}[w] = 0$$
$$\mathbb{E}\big[f(x) f\big(x'\big)\big] = \phi(x)^T \mathbb{E}\big[ww^T\big]\phi\big(x'\big) = \phi(x)^T \Sigma_p \phi\big(x'\big)$$

This means that $f(x)$ and $f\big(x'\big)$ are jointly Gaussian with zero mean and covariance given by $\phi(x)^T \Sigma_p \phi\big(x'\big)$.

As consequence, the function values $f(x_1), \ldots, f(x_n)$ obtained at n different points $x_1, \ldots, x_n$ are jointly Gaussian.

The covariance function assumes a critical role int the GP modelling, as it specifies the distribution over functions. To see this, we can draw samples from the distribution of functions evaluated at any number of points; in detail, we choose a set of input points $X_{1:n} = (x_1, \ldots, x_n)^T$ and then compute the corresponding covariance matrix element wise. This operation is usually performed by using pre-defined covariance functions allowing to write covariance between *outputs* as a function of *inputs* (i.e. $\mathrm{Cov}\big(f(x), f\big(x'\big)\big) = k\big(x, x'\big)$). Finally, we can generate a random Gaussian vector as:

$$f(X_{1:n}) \sim \mathcal{N}(\mathbf{0}, \mathrm{K}(X_{1:n}, X_{1:n}))$$

and plot the generated values as a function of the inputs. This is basically known as *sampling from prior*, whose core is sampling from a multivariate Gaussian distribution (Tong 1990).

Following an example of five different GP samples drawn from the GP prior: the covariance function used is known as the squared exponential (SE) kernel, introduced in Chap. 1 and that will be detailed in the following Sect. 3.1.2 (Fig. 3.1).

We are usually not primarily interested in drawing random functions from the prior but want to incorporate the knowledge about the function obtained through the evaluations performed so far. Such a knowledge will be then used by the *acquisition function* (presented in Chap. 4) in order to associate an informational utility to each point $x \in X$. We have usually access only to noisy function values, denoted by $y = f(x) + \varepsilon$. Assuming additive independent identically distributed Gaussian noise ε with variance λ^2, the prior on the noisy observations becomes:

$$\mathrm{Cov}\big(f(x), f\big(x'\big)\big) = k\big(x, x'\big) + \lambda^2 \delta_{xx'}$$

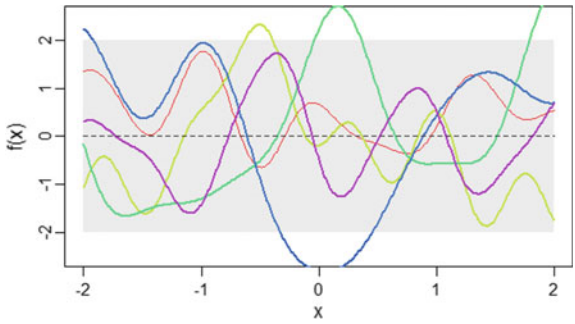

Fig. 3.1 Five different samples from the prior of a GP with squared exponential kernel as covariance function

where $\delta_{xx'}$ is a Kronecker delta which is equal to 1 if and only if $x = x'$. Thus, the covariance over all the function values $\mathrm{y} = (y_1, \ldots, y_n)$ is:

$$\mathrm{Cov}(\mathrm{y}) = \mathrm{K}(X_{1:n}, X_{1:n}) + \lambda^2 I$$

Therefore, the predictive equations for GP regression, that are $\mu(x)$ and $\sigma^2(x)$, can be easily updated, by conditioning the joint Gaussian prior distribution on the observations:

$$\mu(x) = \mathbb{E}[f(x)|D_{1:n}, x] = \mathrm{k}(x, X_{1:n})\left[\mathrm{K}(X_{1:n}, X_{1:n}) + \lambda^2 I\right]^{-1}\mathrm{y}$$
$$\sigma^2(x) = k(x, x) - \mathrm{k}(x, X_{1:n})\left[\mathrm{K}(X_{1:n}, X_{1:n}) + \lambda^2 I\right]^{-1}\mathrm{k}(X_{1:n}, x)$$

These equations are the same reported in Sect. 1.2.1.

Following, a simple example of five different samples drawn at random from a GP prior and posterior, respectively. Posterior is conditioned to six function observations (Fig. 3.2).

Sampling from posterior can be, ideally, considered as generating functions from the prior and rejecting the ones that disagree with the observations. Naturally, this strategy would not be computationally very efficient. A more formal definition of sampling from posterior will be presented in Sect. 3.2.

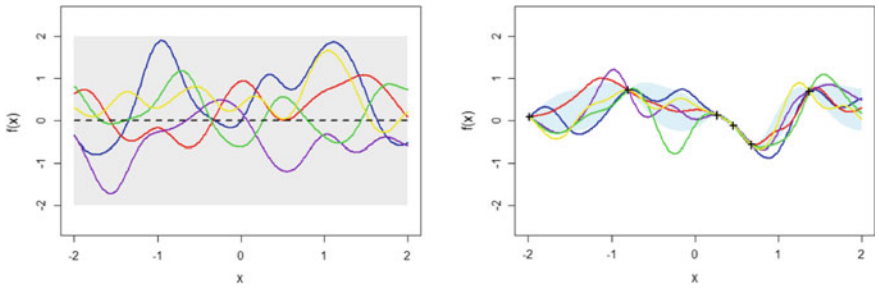

Fig. 3.2 Sampling from prior versus sampling from posterior (for the sake of simplicity, we consider the noise-free setting)

It is easy to notice that the mean prediction is a linear combination of n functions, each one centred on an evaluated point. This allows to write $\mu(x)$ as:

$$\mu(x) = \sum_{i=1}^{n} \alpha_i k(x, x_i)$$

where the vector $\alpha = \left[\mathrm{K}(X_{1:n}, X_{1:n}) + \lambda^2 I\right]^{-1}\mathrm{y}$ and α_i is the i-th component of the vector α, given by the product between the i-th row of the matrix $\left[\mathrm{K}(X_{1:n}, X_{1:n}) + \lambda^2 I\right]^{-1}$ and the vector y.

This means that, to make a prediction at a given x, we only need to consider the $(n + 1)$-dimensional distribution defined by the n function evaluations performed so far and the new point x to evaluate.

Several covariance functions have been proposed and some of the most widely adopted will be presented in the following subsection 3.1.2. Every covariance function has some hyperparameters to be set up, defining shape features of the GP, such as smoothness and amplitude. The values of the hyperparameters are usually unknown a priori and are set up depending on the observations $D_{1:n}$. Summarizing with γ the vector of the covariance's hyperparameters, their values are usually set up via marginal likelihood maximization, which is given, in the case of GP for regression, in closed form:

$$p(y \mid X_{1:n}, \gamma) = \int p(y \mid f, X_{1:n}) p(f|X_{1:n}) df$$

The GP's hyperparameters γ appear non-linearly in the kernel matrix K and a closed-form solution maximizing the marginal likelihood cannot be found in general. In practice, gradient-based optimization algorithms are adopted to find a (local) optimum of the marginal likelihood (e.g. conjugate gradients or BFGS).

An interesting generalization has been recently proposed in Berkenkamp et al. (2019) where the values of hyperparameters are modified by iteratively reducing the characteristic length-scale instead of setting them up through marginal likelihood maximization.

3.1.2 Kernel: The Data Geometry of Bayesian Optimization

A covariance function is the crucial ingredient in a GP predictor, as it encodes assumptions about the function to approximate. From a slightly different viewpoint, it is clear that both in supervised and unsupervised learning the notion of similarity between data points is crucial; it is a basic assumption that points which are close in x are likely to have similar target values y, and thus function evaluations that are near to a given point should be informative about the prediction at that point. Under the GP view, it is the covariance function that defines nearness or similarity.

A *stationary* covariance function is a function of $x - x'$. Thus, it is invariant to translations in the input space. If further the covariance is a function only of $|x - x'|$ then it is called *isotropic* and it is invariant to all rigid motions. Finally, a *dot product* covariance function depends only on $x \cdot x'$. Dot product covariance functions are invariant to a rotation of the coordinates about the origin but not translations.

A general name for a function k of two arguments mapping a pair of inputs x and x' into a scalar is *kernel*. For a kernel to be a covariance function the following conditions must be satisfied:

- kernel symmetry if $k(x, x') = k(x', x)$
- the matrix K with entries $K_{ij} = k(x_i, x_j)$, also known as *Gram matrix* must be positive semidefinite (PSD).

Examples of covariance (aka kernel) functions:

Squared Exponential (SE) kernel:

$$k_{SE}(x, x') = e^{-\frac{\|x-x'\|^2}{2\ell^2}}$$

with ℓ known as *characteristic length-scale*. The role of this hyperparameter is to rescale any point x by $1/\ell$ before computing the kernel value.

A large length-scale implies long-range correlations, whereas a short length-scale makes function values strongly correlated only if their respective inputs are very close to each other.

This kernel is infinitely differentiable, meaning that the GP is very "smooth". Note, that the covariance between the outputs is written as a function of the inputs. For this particular covariance function, we see that the covariance is almost unity between variables whose corresponding inputs are very close and decreases as their distance in the input space increases.

Matérn kernels:

$$k_{\text{Mat}}(x, x') = \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{|x - x'|\sqrt{2\nu}}{\ell} \right)^{\nu} K_{\nu} \left(\frac{|x - x'|\sqrt{2\nu}}{\ell} \right)$$

with two hyperparameters ν and ℓ, and where K_{ν} is a modified Bessel function. Note that for $\nu \to \infty$ we obtain the SE kernel.

The Matérn covariance functions become, particularly simple when ν is half-integer: $\nu = p + 1/2$, where p is a non-negative integer. In this case, the covariance function is a product of an exponential and a polynomial of order p. The most widely adopted versions, specifically in the machine learning community, are $\nu = 3/2$ and $\nu = 5/2$.

$$k_{\nu=3/2}(x, x') = \left(1 + \frac{|x-x'|\sqrt{3}}{\ell}\right) e^{-\frac{|x-x'|\sqrt{3}}{\ell}}$$
$$k_{\nu=5/2}(x, x') = \left(1 + \frac{|x-x'|\sqrt{5}}{\ell} + \frac{(x-x')^2}{3\ell^2}\right) e^{-\frac{|x-x'|\sqrt{5}}{\ell}}$$

Rational Quadratic Covariance Function:

$$k_{RQ}(x, x') = \left(1 + \frac{(x - x')^2}{2\alpha\ell^2}\right)^{-\alpha}$$

where α and ℓ are two hyperparameters. This kernel can be considered as an infinite sum (*scale mixture*) of SE kernels, with different characteristic length-scales. Indeed, one of the most important properties of kernel functions is that a sum of kernels is a kernel.

The following figure summarizes how the value of the four kernels decreases with x moving away from $x' = 0$ (on the left side) and which are possible resulting samples with different shape properties (on the right side) (Fig. 3.3).

The aforementioned kernels are just the most widely adopted in GP regression. More details and a most comprehensive set of covariance functions are reported in Williams and Rasmussen (2006), including non-stationary kernels and dot product kernels. Kernel for BO is a very actively researched area, relevant papers are Duvenaud et al. (2013; Shilton et al. (2018). Schultz et al. (2016) note that a kernel mismatch about the smoothness of the objective function leads to a substantial drop in accuracy and sample efficiency, which increases with the dimensions and cannot be mitigated by GP's hyperparameter tuning and the choice of acquisition functions.

A space-temporal kernel has been proposed in Nyikosa et al. (2018) to allow the GP to capture all the instances of the function over time and track a temporally evolving minimum. Some kernel issues have been considered in recent publications, such as: kernel composition, safe optimization in relation to cognition (Schultz et al. 2018) as well as kernel learning, adaptation and sparsity in order to deal with functions

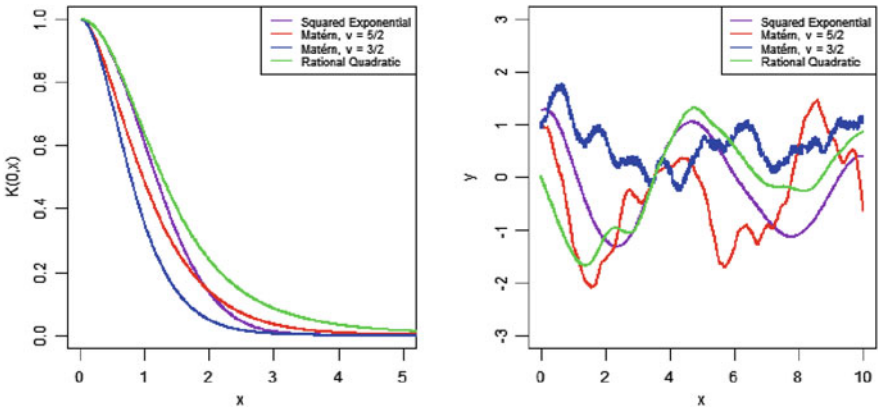

Fig. 3.3 Value of four different kernels with x moving away from $x' = 0$ (left) and four samples from GP prior, one for each kernel considered (right). The value of the characteristic length-scale is ℓ 1 for all the four kernels; α of the RQ kernel is set to 2.25

that are smooth in a subset of their domain and vary rapidly in another is analysed in Peifer et al. (2019).

3.1.3 *Embedding Derivative Observations in the Gaussian Process*

A relatively new part of GP regression, specifically useful for BO, is the integration of derivative information of $f(x)$ in the modelling and optimization process. Differentiation is a linear operator so that the derivative of a GP is also a GP:, if the covariance function is sufficiently smooth, one can derive the joint distribution over a function and its derivatives and then perform their Bayesian updating given observations of function, derivatives and Hessians (Wu et al. 2017).

We consider a d-dimensional function sampled from a GP, $f(\cdot) \sim \mathcal{GP}(0, K)$. Let us denote the first order partial derivative operator as $\nabla(\cdot) = \left[\frac{\partial}{\partial_{x_1}} \frac{\partial}{\partial_{x_2}} \dots \frac{\partial}{\partial_{x_d}}\right]^T$.

Then the joint process $[f(x), \nabla f(x)]$ has a GP distribution (see for instance Solak et al. (2003)) with a covariance function given by four blocks:

$$\begin{aligned} k_{[f,f]}(x, x') &= \operatorname{cov}\big(f(x), f(x')\big) = k(x, x') \\ k_{[f,\nabla f]}(x, x') &= \operatorname{cov}\big(f(x), \nabla f(x')\big) = \nabla_{x'} k(x, x') \\ k_{[\nabla f,f]}(x, x') &= \operatorname{cov}\big(\nabla f(x), f(x')\big) = \nabla_{x} k(x, x') \\ k_{[\nabla f,\nabla f]}(x, x') &= \operatorname{cov}\big(\nabla f(x), \nabla f(x')\big) = \nabla_{x} \nabla_{x'} k(x, x') \end{aligned}$$

We can write this joint process more compactly as

$$\begin{bmatrix} f(x) \\ \nabla f(x) \end{bmatrix} \sim \mathcal{GP}\left(0, \begin{bmatrix} \mathrm{k} & \mathrm{k}_{[f,\nabla f]} \\ \mathrm{k}_{[\nabla f,f]} & \mathrm{k}_{[\nabla f,\nabla f]} \end{bmatrix}\right)$$

We can now apply the formulas to update $\mu(x)$ and $\sigma(x)$ and to derive the posterior over function values $f(x_{n+1})$ given a set of observations of function values and gradients. Let $\mathrm{K}_{[\mathrm{f},\nabla\mathrm{f}]}$ denote the joint kernel matrix for a set of observations of function values and gradients. The joint distribution for $\big[f(X_{1:n}), \nabla f(X_{1:n}), f(x_{n+1})\big]$ is

$$\begin{bmatrix} f(X_{1:n}) \\ \nabla f(X_{1:n}) \\ f(x_{n+1}) \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} \mathrm{K}_{[\mathbf{f},\nabla\mathbf{f}]} & \bar{\mathrm{k}}_{n+1} \\ \bar{\mathrm{k}}_{n+1}{}^{\intercal} & k(x_{n+1}, x_{n+1}) \end{bmatrix}\right)$$

where $\bar{\mathrm{k}}_{n+1} = \left[\mathrm{k}(x_{n+1}, X_{1:n})^{\intercal}, \mathrm{k}_{[f_{n+1},\nabla f]}\right]^{\intercal}$, and the posterior over $f(x_{n+1})$ is:

$$f(x_{n+1}) | x_{1:n}, [f, \nabla f]_{1:n}, x_{n+1} \sim \mathcal{N}\big(\bar{\mu}(x_{n+1}), \bar{\sigma}^2(x_{n+1})\big)$$

where

$$\bar{\mu}(x_{n+1}) = \bar{\mathrm{k}}_{n+1}^{\mathsf{T}} \mathrm{K}_{[f,\nabla f]}^{-1} [f, \nabla f]_{1:n}^{\mathsf{T}}$$
$$\bar{\sigma}^2(x_{n+1}) = k(x_{n+1}, x_{n+1}) - \bar{\mathrm{k}}_{n+1}^{\mathsf{T}} \mathrm{K}_{[f,\nabla f]}^{-1} \bar{\mathrm{k}}_{n+1}$$

We use a similar derivation to incorporate Hessian (second derivative) information into a GP along with the function and gradient information.

Then the joint Gaussian of $\left[f, \nabla f, \nabla^2 f(x)\right]$ is defined as,

$$\begin{bmatrix} f(x) \\ \nabla f(x) \\ \nabla^2 f(x) \end{bmatrix} \sim \mathcal{GP}\left(0, \mathrm{K}_{[f,\nabla f,\nabla^2 f]}\right)$$

where

$$\mathrm{K}_{[f,\nabla f,\nabla^2 f]} = \begin{bmatrix} \mathrm{k} & \mathrm{k}_{[f,\nabla f]} & \mathrm{k}_{[f,\nabla^2 f]} \\ \mathrm{k}_{[\nabla f,f]} & \mathrm{k}_{[\nabla f,\nabla f]} & \mathrm{k}_{[\nabla f,\nabla^2 f]} \\ \mathrm{k}_{[\nabla^2 f,f]} & \mathrm{k}_{[\nabla^2 f,\nabla f]} & \mathrm{k}_{[\nabla^2 f,\nabla^2 f]} \end{bmatrix}$$

The resulting joint kernel matrix is partitioned into nine blocks corresponding to the covariances and cross-covariances over function values, gradients and Hessians.

We can now derive the posterior over function values $f(x_{n+1})$ given a set of observations of function, its gradients and Hessians. The joint distribution of $\left[f(X_{1:n}), \nabla f(X_{1:n}), \nabla^2 f(X_{1:n}), f(x_{n+1})\right]$ is given by:

$$\begin{bmatrix} f(X_{1:n}) \\ \nabla f(X_{1:n}) \\ \nabla^2 f(X_{1:n}) \\ f(x_{n+1}) \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} \mathrm{K}_{[f,\nabla f,\nabla^2 f]} & \bar{\mathrm{k}}_{n+1} \\ \bar{\mathrm{k}}_{n+1}{}^{\mathsf{T}} & k(x_{n+1}, x_{n+1}) \end{bmatrix}\right)$$

where $\bar{\mathrm{k}}_{n+1} = \left[\mathrm{k}(x_{n+1}, X_{1:n})^{\mathsf{T}}, \mathrm{k}_{[f_{n+1},\nabla f]}, \mathrm{k}_{[f_{n+1},\nabla^2 f]}\right]^{\mathsf{T}}$ and the posterior over $f(x_{n+1})$ is:

$$f(x_{n+1}) | x_{1:n}, \left[f, \nabla f, \nabla^2 f\right]_{1:n}, x_{n+1} \sim \mathcal{N}\left(\bar{\mu}(x_{n+1}), \bar{\sigma}^2(x_{n+1})\right)$$

where

$$\bar{\mu}(x_{n+1}) = \bar{\mathrm{k}}_{n+1}^{\mathsf{T}} \mathrm{K}_{[f,\nabla f,\nabla^2 f]}^{-1} \left[f, \nabla f, \nabla^2 f\right]_{1:n}^{\mathsf{T}}$$
$$\bar{\sigma}^2(x_{n+1}) = k(x_{n+1}, x_{n+1}) - \bar{\mathrm{k}}_{n+1}^{\mathsf{T}} \mathrm{K}_{[f,\nabla f,\nabla^2 f]}^{-1} \bar{\mathrm{k}}_{n+1}$$

The idea of using derivative information for optimization problems has been taken up in Hennig and Kiefel (2013; Hennig (2013) and in Wills and Thomas (2017) which has developed it along two different directions. The first, originally suggested in the

papers by Henning, brings to the reinterpretation in probabilistic terms of standard quasi-Newton like methods considered as particular instances of Gaussian process or Bayesian regression. The second, more relevant for the subject of this book, is to use the joint GP as a global model of the objective function and its derivatives.

A first remark is that adding a derivative observation reduces the uncertainty-standard deviation of GP prediction. The following figure, drawn from Solak et al. (2003) shows, from left to right, four samples from prior, four samples from posterior of standard GP and four samples from posterior of a GP with derivative observations. Given the same set of observations, the variance of the resulting GP is lower when derivative observations are included (Fig. 3.4).

The same pay-off effect of gradient estimation is shown in the following picture, drawn from Eriksson et al. (2018). On the left, the level sets of the objective function (Branin 2D) is depicted, in the middle and on the right, the mean of the GP without and with derivative observations, is reported, respectively (Fig. 3.5).

The same authors propose a BO algorithm with derivatives, inspired by REMBO (Wang et al. 2016). The algorithm estimates the active subspace spanned by the dominant eigenvalues of the gradient covariance matrix and fits a GP on the set of observations in this subspace. The optimization of the acquisition function,

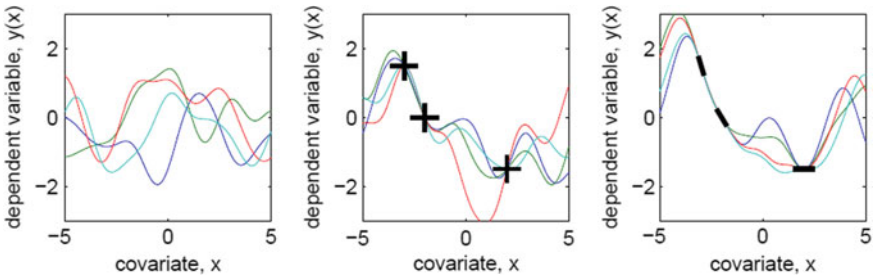

Fig. 3.4 Four samples from prior, four samples from posterior of a standard GP and four samples from posterior of a GP with derivative observations. (*Source* Solak et al. 2003)

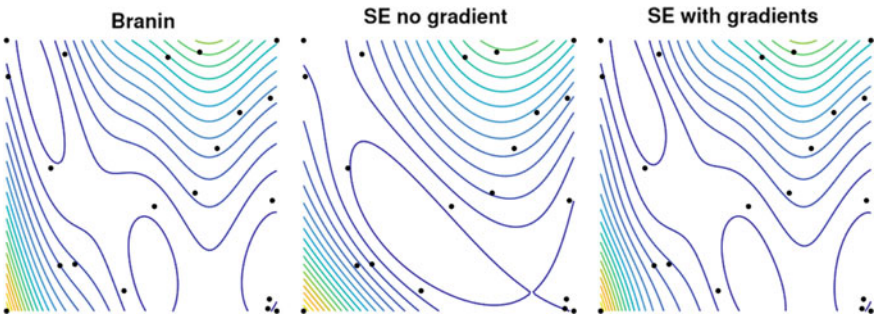

Fig. 3.5 Original function (left) mean of a GP without derivative observations (middle) and mean of a GP with derivative observations. (*Source* Eriksson et al. 2018)

built on the updated model, brings the new point x_{n+1}. The dataset is updated with $\left[x_{n+1}, f(x_{n+1}), \nabla f(x_{n+1})\right]$ and, consequently, the GP's hyperparameters are updated considering also the information on gradients.

3.1.4 Numerical Instability

Numerical instability issues arise due to the inversion of the matrix $\left[\mathrm{K}(X_{1:n}, X_{1:n}) + \lambda^2 I\right]$, required to update $\mu(x)$ and $\sigma^2(x)$. The *condition number* of this matrix, that is the ratio between the highest and lowest eigenvalue, gives a bound on the accuracy of the matrix inversion. Large condition numbers are indicators for numerical instability. Extreme eigenvalues appear when function values are strongly correlated. Strong positive correlation between function values will result in near-identical corresponding rows and columns in the kernel matrix $\mathrm{K}(X_{1:n}, X_{1:n})$, making it close to singular. Higher values of λ, on one side mitigate this instability, on the other side lead to a degradation of the model's accuracy.

Some heuristics have been proposed to control the condition number of the matrix $\left[\mathrm{K}(X_{1:n}, X_{1:n}) + \lambda^2 I\right]$. The most common consists of adding an explicit penalty on the signal-to-noise ratio, when fitting the GP or adding a jitter term. Both are presented in https://drafts.distill.pub/gp/.

The first method derives from the observation that the condition number grows linearly with the number n of data points and quadratically with the signal-to-noise ratio $\frac{\sigma_f}{\lambda}$ (where σ_f^2 is a multiplier used to regulate the maximum kernel value). More precisely, $\frac{e_{max}}{e_{min}} = n\frac{\sigma_f^2}{\lambda^2} + 1$, where e_{max} and e_{min} are the largest and smallest eigen value, respectively. It is easy to understand that a high signal-to-noise ratio makes quickly the matrix $\left[\mathrm{K}(X_{1:n}, X_{1:n}) + \lambda^2 I\right]$ ill-conditioned. Therefore, although most of the research works consider the noise-free setting, this should be avoided in practical applications for numerical stability reasons. When fitting the GP by maximizing the log-marginal likelihood, a penalty term is added to the log-marginal likelihood that discourages extreme signal-to-noise ratios.

The second heuristic method, consisting of adding a jitter term, is also based on the previous consideration: it basically adds some artificial "noise" to the function evaluations.

In Zhigljavsky and Žilinskas (2019), authors studied the properties of a GP generated bya SE kernel where the exponent 2 has been replaced by 2-ε, with $\varepsilon > 0$.

An interesting solution, called K-optimality and aimed at choosing the next point x_{n+1} to reduce the condition number, has been recently suggested in Yan et al. (2018) and will be outlined in Chap. 4.

Finally, it is important to remark that using derivatives, as reported in previous section, induces a faster onset of the numerical instability issue.

3.2 Thompson Sampling

As previously mentioned, given a kernel k, there exists a feature map $\phi(x)$ such that $k(x, x') = \phi(x)^T \phi(x')$. In practice, one can use the spectral decomposition of the matrix $K = \mathbf{\Phi}\mathbf{\Phi}^T$ to estimate the feature map $\phi(x) \in \mathbb{R}^m$, where $\mathbf{\Phi}^T = [\phi(x_1), ..., \phi(x_n)]$ and $\{x_1, ..., x_n\}$ are the points evaluated so far. The decomposition we considered is derived from the Bochner's theorem, as reported in (Basu and Ghosh 2017), and based on the concept of *spectral density* of a kernel. For instance, the spectral density *S(s)* of the SE covariance function is given by:

$$S(s) = (2\pi\ell^2)^{\frac{d}{2}} e^{-2\pi^2\ell^2 s^2}$$

The spectral density can be treated as a probability density $p(s) = S(s)/\alpha$ with $\alpha = \int S(s)ds$ a normalizing constant. The following figure shows *p(s)* for different values of the value of the length scale of the SE kernel (Fig. 3.6).

In TS, the feature map $\phi(x)$ is a m-dimensional vector sampled as $\sqrt{2/m}\cos(\mathbf{W}x + \mathbf{b})$, where $[\mathbf{W}]_i \sim p(s)$ and $[\mathbf{b}]_i \sim \mathcal{U}(0, 2\pi)$, with i denoting the ith component of the vectors $\mathbf{W}$ and $\mathbf{b}$, and $i = 1, ..., m$.

Accordingly, the following figures show how the shape of the (prior) sample modifies depending on the length scale (i.e. ℓ is 0.2, 0.3 and 0.4, from left to right). Every sample is generated by sampling $[\mathbf{W}_i] \sim p(s)$ and $[\mathbf{b}]_i \sim \mathcal{U}(0, 2\pi)$ (Figs. 3.7 and 3.8).

Although Thompson sampling (TS) dates to (Thompson 1933), it has been recently attracting attention in several studies on sequential optimization (Chapelle and Li 2011; Kandasamy et al. 2017, 2018). Ouyang et al. (2017) propose a TS-based

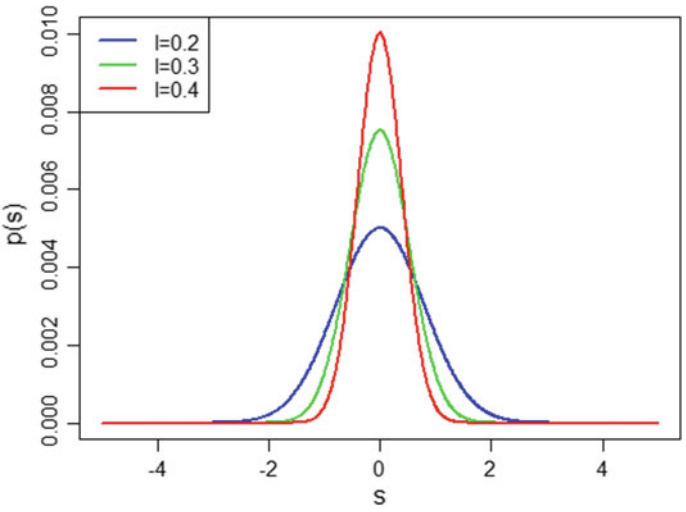

Fig. 3.6 Spectral density of SE kernel with different values of the characteristic length-scale

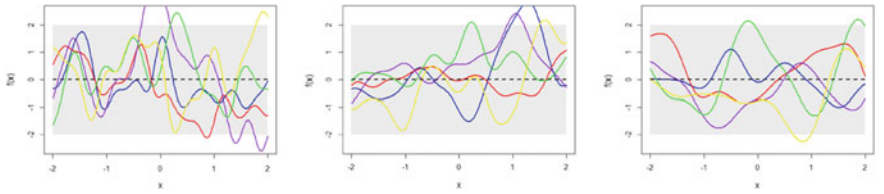

Fig. 3.7 Five different samples for GP prior for, respectively, $\ell = 0.2$, $\ell = 0.3$ and $\ell = 0.4$, respectively

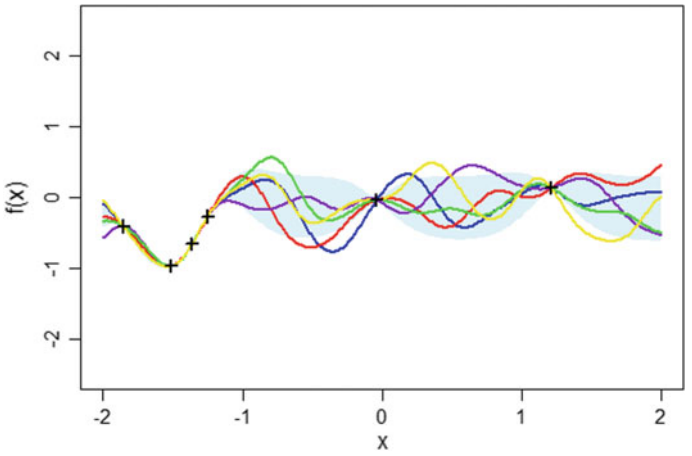

Fig. 3.8 Five different samples for GP posterior (ℓ is learned through marginal likelihood maximization on the observations)

approach to learn the structure of an unknown Markov decision process (MDP) in a reinforcement learning framework.

TS is an algorithm to sequentially optimize a black box function, coherently with the optimization problem considered in this book:

$$\min_{x \in X \subset \mathbb{R}^d} f(x)$$

As BO, TS uses a GP as a probabilistic surrogate model of the unknown objective function. The main difference is that TS samples, at every iteration, just one mapping function $\phi(x)$ and a random vector $\boldsymbol{\theta}$ drawn from the posterior distribution $\boldsymbol{\theta}|(D_{1:n}, \phi) = \mathcal{N}\left(\boldsymbol{A}^{-1}\boldsymbol{\Phi}^T\boldsymbol{y}, \sigma^2\boldsymbol{A}^{-1}\right)$, where $\boldsymbol{A} = \boldsymbol{\Phi}^T\boldsymbol{\Phi} + \sigma^2\boldsymbol{I}$. Then the new point x_{n+1} to evaluate is just obtained as the minimizer of the GP sample $f(x) = \phi(x)^T\boldsymbol{\theta}$. The function is evaluated at the new point, eventually with noise, and the function evaluations dataset is updated consequently: $D_{n+1} = D_n \cup \{(x_{n+1}, y_{n+1})\}$.

The process is initialized by assuming a non-informative prior on the distribution of the minimizer and will stop when the variance of the distribution becomes considerably small.

The following figure shows four different samples from GP posterior (after five observations) and the corresponding minimizers to use as next point to evaluate. We have decided to plot four different scenarios to highlight the probabilistic nature of TS in selecting the next point. Although samples are different, giving some chance to exploration, the bias towards exploitation is evident, motivating the ε-greedy approach proposed in (Basu and Ghosh 2017) (Fig. 3.9).

As a final remark, it is clear that TS is a sequential optimization process per se, and sampling from posterior is just one component of the method. It is again based on a probabilistic surrogate model but, differently from BO, the next promising point to evaluate is identified by minimizing a sample from the GP instead of a specific acquisition function. Another way to look at this is to consider the sample function as an acquisition function (as described in Chap. 4).

While the theory underlying TS and related converge issue are rooted in results in functional analysis, like Bochner's theorem and Winer–Khintchine theorem, its implementation is relatively straightforward as shown before and demonstrated in the pseudo code which follows, again inspired by (Basu and Ghosh 2017). There are different implementations, notable (Bijl et al. 2016)

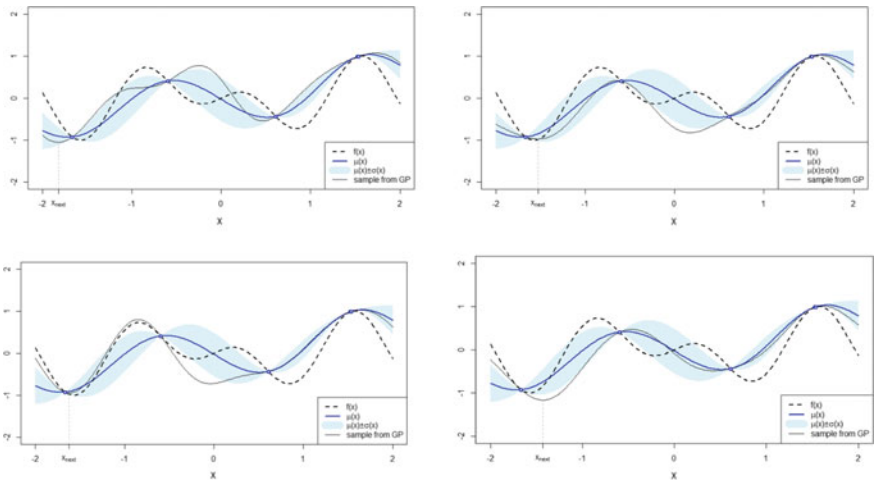

Fig. 3.9 An example of one iteration of Thompson sampling: four different samples for GP posterior are reported along with the next point to evaluate chosen as the minimizer of the corresponding GP sample

Algorithm - Thompson Sampling with SE kernel

Input: function $f(x)$, kernel k, domain X
Output: x^+ the optimal solution minimizing $f(x)$
Observe $y_1 \leftarrow f(x_1) + \varepsilon$, with $\varepsilon \sim \mathcal{N}(0, \lambda^2)$
$D_1 \leftarrow \{(x_1, y_1)\}$
$x^+ \leftarrow x_1$
$y^+ \leftarrow f(x^+)$
for $n = 1, 2, \ldots$ **do**
  estimate the kernel's hyperparameters depending on $D_{1:n}$(e.g., the length scale ℓ in the case of SE kernel)
  sample a random feature map $\phi(x)$ (i.e., from $\sqrt{2/m}\cos(\mathbf{W}x + \mathbf{b})$, as previously described)
  sample $\boldsymbol{\theta}$ from $\boldsymbol{\theta}|(D_{1:n}, \phi) \sim \mathcal{N}(\boldsymbol{A}^{-1}\boldsymbol{\Phi}^T\boldsymbol{y}, \sigma^2\boldsymbol{A}^{-1})$, with $\boldsymbol{A} = \boldsymbol{\Phi}^T\boldsymbol{\Phi} + \sigma^2\boldsymbol{I}$ and $\boldsymbol{\Phi}^T = [\phi(x_1), \ldots, \phi(x_n)]$
  select $x_{n+1} = \text{argmin}_{x \in X}\, \phi(x)^T\boldsymbol{\theta}$
  observe $y_{n+1} \leftarrow f(x_{n+1}) + \varepsilon$, with $\varepsilon \sim \mathcal{N}(0, \lambda^2)$
  $D_{n+1} \leftarrow D_n \cup \{(x_{n+1}, y_{n+1})\}$
  break from the loop when x_{n+1} converges to x^+
  if $y_{n+1} < y^+$ **then**
    $x^+ \leftarrow x_{n+1}$
    $y^+ \leftarrow y_{n+1}$
  end
endfor
return x^+

TS can be biased towards exploitation, especially when σ^2 is large, leading the process to potentially converge to a local minimum. To reduce the risk to get stuck in a local optimum, an ε-greedy approach can be considered, as in (Basu and Ghosh 2017). That is, at every iteration n, we explore the entire region X uniformly (i.e. we sample $x_{n+1} \sim \mathcal{U}(X)$), with probability $\varepsilon > 0$ or we sample $x_{n+1} \sim min_{x \in X}\phi(x)^T\boldsymbol{\theta}$, with probability $1 - \varepsilon$.

3.3 Alternative Models

A basic issue with GP is the assumption that optimization variables take real values. In the case of integer-valued variables, such as the number of layers of a DNN, the basic idea is rounding the solution to the closest integer while, in the case of categorical variables, as many extra variables as possible categories are added and chosen according to the largest one (as in Spearmint). These operations can be applied in two different ways. In the so-called naïve approach they are performed before updating the GP, so x_{n+1} is a mixed-integer vector, while in the "basic" approach they are performed just before evaluating the objective function, so $x_{n+1} \in \mathbb{R}^d$. The naïve approach might lead to a relevant mismatch between the (continuous) optimizer of the objective function and the point evaluated. The basic approach overcomes this limitation because GP ignores the approximation but it is expected to provide a suboptimal solution. A more principled approach is suggested in Garrido-Merchán and Hernández-Lobato (2018), who proposes a kernel that essentially embeds the same transformation of the basic approach.

A radical solution consists in avoiding GP altogether and adopting alternative models like random forests.

3.3.1 Random Forest

Random forest (RF) is an ensemble learning method, based on decision trees, for both classification and regression problems (Ho 1995). According to the originally proposed implementation, RF aims at generating a multitude of decision trees, at training time, and providing as output the mode of the classes (classification) the mean/median prediction (regression) of the individual trees. The following figure provides a schematic representation of the output provisioning mechanism of a RF (Fig. 3.10).

Decision trees are ideal candidates for ensemble methods since they usually have low bias and high variance, making them very likely to benefit from the averaging process. RF mostly differs from other typical ensemble methods in the way it introduces random perturbations into the training phase. The basic idea is to combine bagging, to sample examples from the original dataset, and random selection of features, in order to train every tree of the forest. Injecting randomness simultaneously with both strategies yields one the most effective off-the-shelf methods in machine learning, working surprisingly well for almost any kind of problems, allowing for generating a collection of decision trees with controlled variance.

Although designed and presented as a machine learning algorithm, RF is also an effective and efficient alternative to GP for implementing BO. To better understand how RF, for regression, can replace GP, one has to consider that:

- The dataset, in the case of Machine Learning, is replaced by the design $D_{1:n}$ of the BO process
- The features correspond to the dimensions of the global optimization problem.

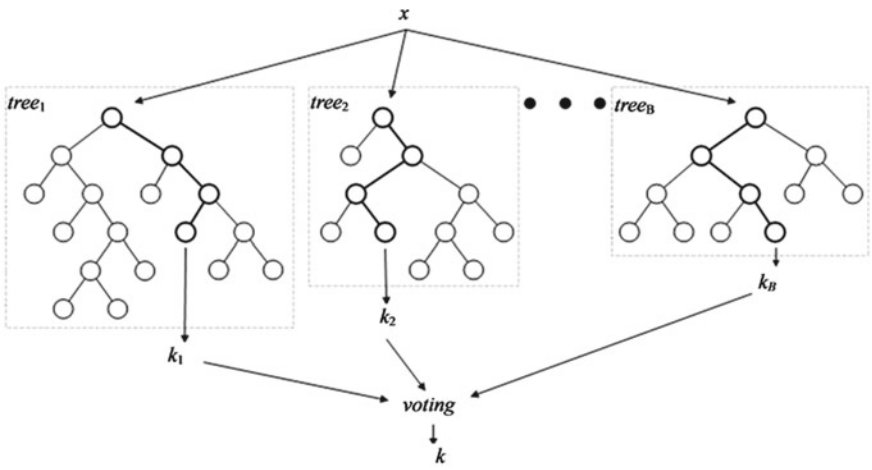

Fig. 3.10 A schematic representation of the regression/classification performed by a random forest. Voting mechanism is different for classification (e.g. simple or weighted majority) and regression (e.g. mean or median)

Moreover, since RF consists of an ensemble of different regressors, it is possible to compute—as for GP—both $\mu(x)$ and $\sigma(x)$, simply as mean and variance among the individual outputs provided by the regressor. Due to the different nature of RF and GP, the associated probabilistic surrogate models will also result significantly different. The following figure offers an example. While GP is well-suited to model smooth functions in search space spanned by continuous variables, RF can deal with discrete and conditional variables. Discontinuity in the RF-based surrogate is given by the underlying decision tree models (Fig. 3.11).

Another important property of RF is that the variance can be controlled, as explained in the following.

Let's denote with S the size of the forest (i.e. the number of decision trees in the forest) and $T_i(x)$ the output provided by the i-th decision tree for the input x, the variance of the RF-based estimator, for the regression case, can be easily computed as follows:

$$\begin{aligned}
\mathrm{Var}\left(\frac{1}{S}\sum_{i=1}^{S} T_i(x)\right) &= \mathrm{Cov}\left(\frac{1}{S}\sum_{i=1}^{S} T_i(x), \frac{1}{S}\sum_{j=1}^{S} T_j(x)\right) \\
&= \frac{1}{S^2}\sum_{i=1}^{S}\left(\sum_{j\neq i}^{S} \mathrm{Cov}\big(T_i(x), T_j(x)\big) + \mathrm{Var}(T_i(x))\right) \\
&\leq \frac{1}{S^2}\sum_{i=1}^{S}\left((S-1)\rho\sigma^2 + \sigma^2\right) \\
&= \frac{S(S-1)\rho\sigma^2 + S\sigma^2}{S^2} = \rho\sigma^2 + \sigma^2\frac{1-\rho}{S}
\end{aligned}$$

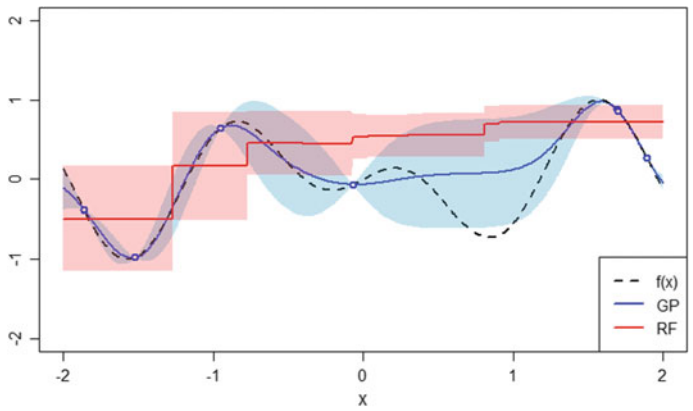

Fig. 3.11 A graphical comparison between probabilistic surrogate models offered by a GP (blue) and a RF (red), both fitted on the same set of observations

where σ^2 is the maximum variance computed over all the $T_i(x)$ and $\rho\sigma^2 = \max_{i,j} \mathrm{Cov}\big(T_i(x), T_j(x)\big)$. The variance of the RF estimator is proportional to σ^2 and ρ (i.e. if the number m of selected features decreases also σ^2 and ρ decrease) and with the size S the forest increasing.

Finally, a basic description of the RF learning algorithm is provided. Thanks to the bagging and random feature selection—step 2 and 4, respectively—RF results more computationally efficient than GP, specifically when the number of decision variables is larger than 20. Indeed, RF does not require to invert any kernel matrix and training of every decision tree can be performed in parallel.

Random Forest learning algorithm
Given:
· S the size of the forest (i.e. number of decision trees in the forest)
· N the number of examples in the dataset (i.e. evaluations of the objective functions)
· M the number of features (i.e. decision variables) representing every example
· m the number of features (i.e. decision variables) to be randomly selected from the M available for each leaf node of the tree to be split
· $n_{\min}$ the minimum node size in the tree
1: for $i = 1$ to S
2: sample, with replacement, a subset of N instances from the dataset
3: for every terminal node (leaf) of the i-th decision tree with size less than $n_{\min}$
4: select m features (i.e. decision variables) at random from the M available
5: pick the best feature (i.e. decision variable) and split among the possible m
6: split the node in two children nodes (new leaves of the decision tree)
7: end for
8: end for

3.3.2 *Neural Networks: Feedforward, Deep and Bayesian*

The role of neural networks in BO is two-fold: as already discussed they are machine learning algorithms typically optimized in the AutoML setting; on the other hand, they can offer a well-suited alternative to GP. Indeed, as already highlighted in this book, GPs are sample efficient but their computational complexity is cubic in the number of points and do not scale well in high dimensions.

A relevant approach, Snoek et al. (2015) aims to replace the GP with a model that scales better but retains most of the GP desirable properties such as flexibility and well-calibrated uncertainty. More specifically, deep learning models are investigated adding a Bayesian linear regressor to the last hidden layer of a deep neural

network (DNN), marginalizing only the output weights of the net while using a point estimate for the remaining parameters. This results in adaptive basis regression, a well-established statistical technique which scales linearly in the number of observations, and cubically in the basis function dimensionality. This allows to explicitly trade-off evaluation time and model capacity. The resulting algorithm DNGO (Deep Networks for Global Optimization, https://github.com/Anmol6/DNGO-BO) has been extensively tested for optimizing the hyperparameters of deep convolutional neural networks. Empirical results show that DNGO provides the same modelling properties of a GP but with a significantly lower computational cost.

The following figure (Fig. 3.12) shows ANN and DNN for creating alternative surrogate models within BO framework and allows for a comparison of predictive mean (solid blue line) and uncertainty (shaded blue region). The first line figures show the surrogate models generated by DNN with three hidden layers, the first one using only tanh as activation function and the second one using only sigmoid as the activation function. The second line figures show the surrogate models generated by ANN with one hidden layer with the same activation functions

The application of Bayesian methods to neural networks has a rich history in machine learning, the goal of Bayesian neural networks is to uncover the full posterior distribution over the network weights in order to capture uncertainty, to act as a regularizer, and to provide a framework for model comparison. The full posterior

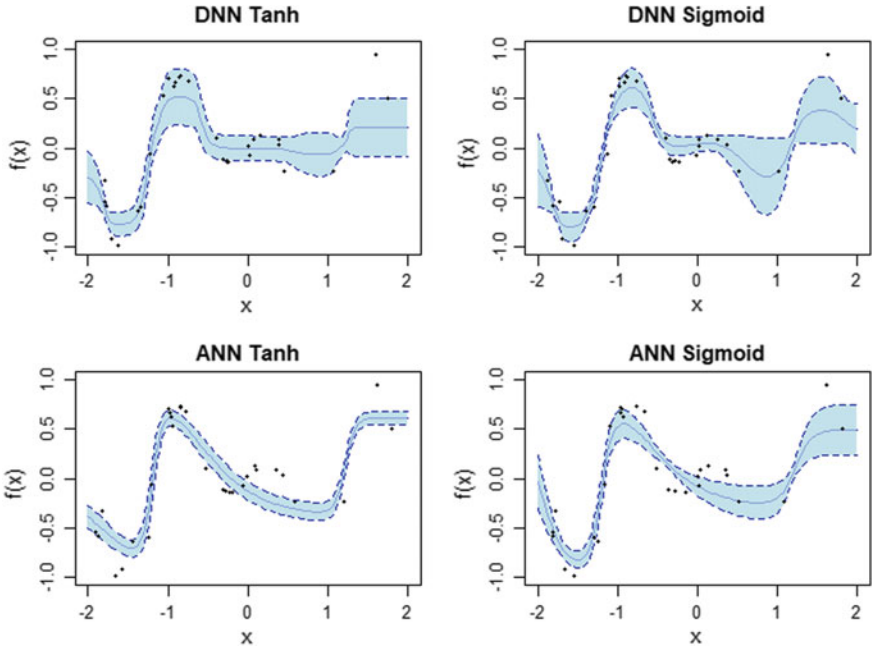

Fig. 3.12 Probabilistic surrogate models based on deep and artificial neural networks, with different activation functions (i.e. sigmoid and tanh) for the neurons

is, however, intractable for most forms of neural networks, necessitating expensive approximate inference or Markov Chain Monte Carlo simulation.

In Springenberg et al. (2016), BNN have been suggested as a principled alternative to GP. The algorithm is called BOHAMIANN (Bayesian Optimization with HAMIltonian Artificial Neural Network) and has been tested with a three layers neural network with 50 tanh units. The method has been presented also for multi-task optimization (i.e. finding the set of optimizers for k black box functions, each with the same domain X). An implementation of the algorithm is provided in RoBO, a BO software reported in Chap. 6.

In order to deal with non-stationarity, a possible approach is to use deep Gaussian processes. In Hebbal et al. (2019), functional composition of stationary GPs is proposed, providing a multiple layer structure.

References

Basu, K., Ghosh, S.: Analysis of Thompson sampling for Gaussian process optimization in the bandit setting (2017). arXiv preprint arXiv:1705.06808

Bergstra, J., Bardenet, R., Bengio, Y., Kégl, B.: Algorithms for hyper-parameter optimization. In: Advances in Neural Information Processing Systems, pp. 2546–2554 (2011)

Berkenkamp, F., Schoellig, A.P., Krause, A.: No-regret Bayesian Optimization with unknown Hyperparameters(2019). arXiv preprint arXiv:1901.03357

Bijl, H., Schön, T.B., van Wingerden, J.W., Verhaegen, M.: A sequential Monte Carlo approach to Thompson sampling for Bayesian optimization (2016). arXiv preprint arXiv:1604.00169

Chapelle, O., Li, L.: An empirical evaluation of Thompson sampling. In: NIPS, pp. 2249–2257 (2011)

Duvenaud, D., Lloyd, J.R., Grosse, R., Tenenbaum, J.B., Ghahramani, Z.: Structure discovery in nonparametric regression through compositional kernel search (2013). arXiv preprint arXiv:1302.4922

Eriksson, D., Dong, K., Lee, E., Bindel, D., Wilson, A.G.: Scaling Gaussian process regression with derivatives. In: Advances in Neural Information Processing Systems, pp. 6868–6878 (2018)

Garrido-Merchán, E.C., Hernández-Lobato, D.: Dealing with categorical and integer-valued variables in Bayesian optimization with Gaussian processes (2018). arXiv preprint arXiv:1805.03463

Hebbal, A., Brevault, L., Balesdent, M., Talbi, E.G., Melab, N.: Bayesian Optimization using Deep Gaussian Processes (2019). arXiv preprint arXiv:1905.03350

Hennig, P., Kiefel, M.: Quasi-Newton method: a new direction. J. Mach. Learn. Res. **14**, 843–865 (2013)

Hennig, P.: Fast probabilistic optimization from noisy gradients. In: International Conference on Machine Learning, pp. 62–70 (2013, February)

Ho, T.K.: Random decision forests. In: Conference in Document Analysis and Recognition, pp. 278–282 (1995)

Kandasamy, K., Krishnamurthy, A., Schneider, J., Póczos, B. Parallelised bayesian optimisation via Thompson sampling. In: International Conference on Artificial Intelligence and Statistics, pp. 133–142 (2018, March)

Kandasamy, K., Krishnamurthy, A., Schneider, J., Poczos, B.: Asynchronous parallel Bayesian optimisation via Thompson sampling (2017). arXiv preprint arXiv:1705.09236

Nyikosa, F.M., Osborne, M.A., Roberts, S. J.: Bayesian optimization for dynamic problems (2018). arXiv preprint arXiv:1803.03432

Ouyang, Y., Gagrani, M., Nayyar, A., Jain, R.: Learning unknown markov decision processes: a Thompson sampling approach. In: Advances in Neural Information Processing Systems, pp. 1333–1342 (2017)

Peifer, M., Chamon, L., Paternain, S., Ribeiro, A.: Sparse multiresolution representations with adaptive kernels (2019). arXiv preprint arXiv:1905.02797

Shilton, A., Gupta, S., Rana, S., Vellanki, P., Li, C., Park, L., Venkatesh, S., Sutti, A., Rubin, D., Dorin, T., Vahid, A., Height, M.:. Covariance function pre-training with m-kernels for accelerated Bayesian optimisation (2018). arXiv preprint arXiv:1802.05370

Schulz, E., Speekenbrink, M., Krause, A.: A tutorial on Gaussian process regression: modelling, exploring, and exploiting functions. J. Math. Psychol. **85**, 1–16 (2018)

Snoek, J., Rippel, O., Swersky, K., Kiros, R., Satish, N., Sundaram, N., Patwary, M., Prabhat, M., Adams, R.: Scalable bayesian optimization using deep neural networks. In: International Conference on Machine Learning, pp. 2171–2180 (2015, June)

Schulz, E., Speekenbrink, M., Hernández-Lobato, J.M., Ghahramani, Z., Gershman, S.J.: Quantifying mismatch in Bayesian optimization. In: Nips Workshop on Bayesian Optimization: black-box Optimization and Beyond (2016)

Solak, E., Murray-Smith, R., Leithead, W.E., Leith, D.J., Rasmussen, C.E.: Derivative observations in Gaussian process models of dynamic systems. In: Advances in Neural Information Processing Systems, pp. 1057–1064 (2003)

Springenberg, J.T., Klein, A., Falkner, S., Hutter, F.: Bayesian optimization with robust Bayesian neural networks. In: Advances in Neural Information Processing Systems, pp. 4134–4142 (2016)

Tong Y.L.: Fundamental properties and sampling distributions of the multivariate normal distribution. In: The Multivariate Normal Distribution. Springer Series in Statistics. Springer, New York, NY (1990)

Thompson, W.R.: On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. Biometrika **25**, 285–294 (1933)

Wang, Z., Hutter, F., Zoghi, M., Matheson, D., de Feitas, N.: Bayesian optimization in a billion dimensions via random embeddings. J. Artif. Intell. Res. **55**, 361–387 (2016)

Williams, C.K., Rasmussen, C.E.: Gaussian Processes for Machine Learning, vol. 2, no. 3, p. 4. MIT Press, Cambridge (2006)

Wills, A.G., Schön, T.B.: On the construction of probabilistic Newton-type algorithms. In: IEEE 56th Annual Conference on Decision and Control (CDC), pp. 6499–6504, IEEE. (2017, December)

Wu, A., Aoi, M.C., Pillow, J.W.: Exploiting gradients and Hessians in Bayesian optimization and Bayesian quadrature (2017). arXiv preprint arXiv:1704.00060

Yan, L., Duan, X., Liu, B., Xu, J.: Bayesian optimization based on k-optimality. Entropy **20**(8), 594 (2018)

Zhigljavsky, A., Žilinskas, A.: Selection of a covariance function for a Gaussian random field aimed for modeling global optimization problems. Optim. Lett. 1–11 (2019)