A Tutorial on Gaussian Process Exploration and Exploitation

Eric Schulz[*], Maarten Speekenbrink[*], Andreas Krause[**]

Computational Learning and Decision Making Laboratory
University College London

Learning and Adaptive Systems Group
ETH Zürich

Abstract

This tutorial introduces Gaussian Process Regression as a method to explore (learn) and exploit (maximize) unknown functions. Gaussian Process Regression is a popular non-parametric Bayesian approach to sequential function learning problems. This tutorial aims to provide an accessible introduction to these techniques. We will introduce and define Gaussian Processes as a distribution over functions used for Bayesian inference and demonstrate different applications thereof. Examples will focus on pure exploration within optimal design problems, bandit-like exploration-exploitation scenarios, and on scenarios with additional constraints, for example safe explorations, where the goal is to never sample below a certain threshold. Software pointers will be provided.

Keywords: Gaussian Process, Exploration-Exploitation, Bandit Problems

[*]Department of Experimental Psychology, University College London.
[**]Department of Computer Science, Swiss Federal Institute of Technology Zürich.

1. Introduction

Whether we try to find a function that describes participants' behavior (Cavagnaro, Aranovich, McClure, Pitt, and Myung, 2014), estimate parameters of psychological models (Wetzels, Vandekerckhove, Tuerlinckx, and Wagenmakers, 2010), sequentially optimize stimuli in an experiment (Myung and Pitt, 2009), or model how participants themselves learn to interact with their environment (Meder and Nelson, 2012), many research problems require us to explore (learn) or exploit (optimize) an unknown function that maps inputs to outputs (Mockus, 2010). Often times the underlying function might be unknown, hard to evaluate analytically, or other requirements such as design costs might complicate the process of information acquisition. In these situations, Gaussian Process regression can serve as a useful multi-purpose tool towards learning and optimization (Rasmussen, 2006). Gaussian Process regression is a non-parametric Bayesian approach (Gershman and Blei, 2012) to regression problems. It can capture a wide variety of functional input-output relations by utilizing a potentially infinite number of parameters and letting the data "decide" on the level of complexity through Bayesian posterior inference (Williams, 1998).

This tutorial will introduce Gaussian Process Exploration-Exploitation as an approach towards learning and optimization of unknown functions. It consists of 5 main parts: The first part will introduce the mathematical basis of Gaussian process regression. The second part will show how Gaussian processes can be used in problems of optimal experimental design, where the goal – learning a function as well as possible – is purely explorative. The third part will describe Bayesian optimization (*exploitation*) with Gaussian processes. **In the fourth part, we will discuss the use of Gaussian**

process exploration-exploitation in situations with additional requirements and show how they can be applied when the goal is to avoid areas that are below a certain threshold. We will conclude by sketching out the possibility of Gaussian Processes as a low level description of cognitive processes.

2. Problem Statement

Imagine a function $f : x \to y$ that maps an input x to an output y. Initially, the function is unknown. The task is to choose sequentially, at each time or trial t, an input value as wisely as possible, either to learn about the unknown function (exploration) or to find the input for which an optimum output is obtained *exploitation.* More formally, at each time t, the function is queried by choosing a point x^* for which the output of the function $y = f(x^*) + \epsilon$ at that point is observed with additional noise ϵ. It is assumed that this noise is independent and identically distributed, following a Gaussian distribution: $\epsilon \sim \mathcal{N}(0, \sigma^2)$.

2.1. Exploration

The goal within exploration is to chose x^* in a way that allows you to learn an unknown function as well as possible. This is sometimes referred to as an optimal experimental design problem (Goos and Jones, 2011). More formally, the task is to find an informative set of points $\mathcal{A} \subset \mathcal{D}$, that is a sequential query $\mathcal{A}$ out of the design set $\mathcal{D}$, chosen to maximize information gain over all potential steps. Generally, the information gain is measured as the difference in entropy between the state before a sample point was queried and afterwards as shown in Equation 1.

$$I(y_A; f) = H(y_A) - H(y_A|f) \quad (1)$$

In the discrete case, the entropy H can be calculated as shown in Equation 2.1.

$$H(X) = \sum_{i}^{n} p(x_i) \log p(x_i)$$

Finding the absolute information gain maximizer is generally NP-hard (Ko, Lee, and Queyranne, 1995), which means that one can never know the best strategy of how to pick all of the observations over time to certainly reach the maximum information gain at the end as this requires an exponentially increasing problem space to be evaluated.

Optimal design problems are ubiquitous in psychology. No matter if we have to find out how participants integrate information (Borji and Itti, 2013) or if we have to find stimuli in order to distinguish between different models (Cavagnaro, Myung, Pitt, and Kujala, 2010), often times we have to try and learn an underlying function that we do not know a priori and do so both adaptively and efficiently. As we will see later, Gaussian Process exploration algorithms provide us with an excellent tool for such tasks.

2.2. Exploration-Exploitation

The goal within exploration-exploitation tasks is to both learn and optimize a function. Here, it is essentially the intention to learn a function quickly so that we then can find the maximum of that function and keep on exploiting it; exploiting means entering the input that produces the highest output. This is normally measured by regret, the distance between what your current guess of the maximum has produced and what the best argument would have produced. If you are sequentially producing an estimate of what you currently think might be the maximum x^*, then it is possible to measure regret R by the difference to what the actual best argument x_{max} would have produced.

$$\begin{aligned} R &= \sum_{i=1}^{T} r_i \\ &= \sum_{i=1}^{T} f(x_{\text{max}}) - f(x^*) \end{aligned}$$

Problems where we have to optimize an unknown function are very common; be it to estimate parameters of a complex model (Snoek, Larochelle, and Adams, 2012) or finding the stimulus that produces the maximal response in an experiment (Daunizeau, Preuschoff, Friston, and Stephan, 2011), many problems require us to explore and exploit functions. Sometimes this approach is also called Bayesian Optimization (Brochu, Cora, and De Freitas, 2010).

3. Gaussian Processes-a distribution over functions

3.1. Motivation

If our goal is to learn or optimize an unknown function, then we need the following two ingredients to be successful.

1. A model for f, that is a model that learns what f looks like.
2. A method to select observations, given the problem statement.

In this section we want to find a good model for f before we can then use it to either explore or exploit the function.

3.2. Weight space view

Let us first start by considering the text book-approach to learn functions, which is linear Bayesian regression, before then transitioning over to Gaussian Process regression. Remember that it is the task to either learn or optimize an unknown function. Therefore, regressing observed input points to observed output points by standard regression seems to be an apparent thing to do. Once we are able to capture the function –so the intuition– we can easily quantify our uncertainty about it or try to find the point that produces the highest expected output. In classic linear regression, we assume normally distributed noise, $\epsilon \sim \mathcal{N}(0, \Sigma)$, and our goal is to predict a value y based on observations $\mathbf{x}$ using weights $\mathbf{w}$.

$$\begin{aligned} f(\mathbf{x}|\mathbf{w}) &= \mathbf{x}^\top \mathbf{w} \\ y &= f + \epsilon \end{aligned}$$

Here, $\mathbf{x}$ are our observations and $\mathbf{w}$ is a vector containing the weights (sometimes notated as β). If we assume a Gaussian prior over the parameters $p(\mathbf{w}) = \mathcal{N}(0, \Sigma)$ and the likelihood $p(\mathbf{y}|\mathbf{X}, \mathbf{w}) = \mathcal{N}(\mathbf{X}^\top \mathbf{w}, \boldsymbol{\sigma^2 \mathbf{I}})$, then –by applying standard Bayesian inference– we get the posterior

$$\begin{aligned} p(\mathbf{w}|\mathbf{y}, \mathbf{X}) &\propto p(\mathbf{y}|\mathbf{X}, \mathbf{w})p(\mathbf{w}) \\ &= \mathcal{N}(\frac{1}{\sigma^2}\mathbf{A}^{-1}\mathbf{X}\mathbf{y}, \mathbf{A}^{-1}) \end{aligned}$$

with $\mathbf{A} = \boldsymbol{\Sigma}^{-1} + \sigma^{-2}\mathbf{X}\mathbf{X}^\top$. In order to make predictions at a new test points $\mathbf{x}_\star$, we have to average over all posterior predictions

$$\begin{aligned} p(f_\star|\mathbf{x}_\star, \mathbf{X}, \mathbf{y}) &= \int p(f_\star|\mathbf{x}_\star, \mathbf{w}) p(\mathbf{w}|\mathbf{X}, \mathbf{y}) d\mathbf{w} \\ &= \mathcal{N}(\sigma^{-2}\mathbf{x}_\star^\top \mathbf{A}^{-1}\mathbf{X}\mathbf{y}, \mathbf{x}_\star^\top \mathbf{A}^{-1}\mathbf{x}_\star) \end{aligned}$$

Even though this approach is commonly chosen to model functions, the way it is set up above only allows to make linear predictions. However, only few relations in the real world are actually linear. Therefore, what we need is a way to model non-linear dependencies as well. One possible adjustment is to use a projection of the inputs $\mathbf{x}$ onto a feature space by using a function $\phi(\mathbf{x})$. One common projection is to use polynomials, resulting into polynomial regression. Take a cubic regression as an example, which assumes a function $f(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \epsilon$. Deriving the posterior for this model is similar to the linear regression described before, only that all $\mathbf{X}$ are replaced by the projection $\mathbf{\Phi} = \phi(\mathbf{X})$. However, the problem then becomes to find appropriate projections for our input variables as infinitely many projections might be possible and we have to choose one a priori (or by model comparison between finite sets of parametric forms). Especially if the problem is to explore and exploit a completely unknown function, this approach will not be beneficial as we would not know which projections we should try out (we do not know the parametric form a priori). Fortunately, there exists another approach to this problem which treats functions as distributions and models this distribution directly. This approach is called Gaussian Process regression and shifts the view away from a weight space perspective to a function space view.

3.3. Function space view

Another approach to regression problems is –instead of modeling distributions over weights– to focus on distributions over functions. A common way to describe a distribution over functions is a Gaussian Process. A Gaussian process is defined as a collection of random variables, any finite number of which have a jointly Gaussian distribution. Therefore, a function is distributed as

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

Here, $m(\mathbf{x})$ is a mean function modeling the expected output of the function and $k(\mathbf{x}, \mathbf{x}')$ is a kernel function modeling the covariance between different points.

$$\begin{aligned} m(\mathbf{x}) &= \mathbb{E}[f(\mathbf{x})] \\ k(x, x') &= \mathbb{E}\left[(f(\mathbf{x}) - m(x))(f(\mathbf{x}') - m(x'))\right] \end{aligned}$$

As $k(x, x')$ models the covariance between two different points, we have to choose a function that will produce sensible estimates. The function k is commonly called the *kernel* of the Gaussian Process (Jäkel, Schölkopf, and Wichmann, 2007). The choice of an appropriate kernel is normally based on assumptions such as smoothness and likely patterns to be expected in the data. If the data is not expected to be periodic, then a sensible assumption is normally that the correlation between two points decays according to a power function in dependency of the distance between the two points and that the covariance is symmetric, that is that only the distance between two points matters, but not the direction. This just means that closer points are expected to be more similar than points which are further away from each other in all possible directions. One very popular choice of a kernel fulfilling those requirement is the so-called squared exponential (sometimes also called Gaussian or Radial Basis Function) kernel.

$$k(x, x') = \sigma^2 \exp\left(-\frac{(x - x')^2}{2\lambda^2}\right)$$

The squared exponential is an expressive way to model smooth functions and the hyper parameters λ (called the length-scale) and σ^2 (the noise constant) are normally optimized by using the marginal likelihood.

This implies the aforementioned distribution over functions as we can easily generate samples for new input points at location $X_\star$.

$$\mathbf{f}_\star \sim \mathcal{N}(0, K\left(X_\star, X_\star\right))$$

Given observations $\mathcal{D} = \{\mathbf{X}, \mathbf{y}\}$ with a noise level σ, we can draw new predictions from our function $\mathbf{f}_\star$ for inputs $X_\star$ as described below.

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{f}_\star \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(X,X) + \sigma^2 I & K(X, X_\star) \\ K(X_\star, X) & K(X_\star, X_\star) \end{bmatrix}\right)$$

This means that we treat a function as a vector of infinite size. However, as we always only have to make predictions for finitely many points, we can simply draw outputs for these points by using a multivariate normal distribution with a covariance matrix generated by our kernel. Calculating the expectation of the Gaussian Process at the new points then is straight forward.

$$\mathbf{f}_\star | X, \mathbf{y}, X_\star \sim \mathcal{N}(\overline{\mathbf{f}}_\star, \mathrm{cov}(\mathbf{f}_\star)) \quad \text{where}$$

This means that predictions for new points are generated based on the expected mean value and covariance function of the posterior Gaussian Process.

$$\begin{aligned} \mathbb{E}[\mathbf{f}_\star | X, \mathbf{y}, X_\star] &= K(X_\star, X)[K(X, X) + \sigma^2 I]^{-1}\mathbf{y} \\ \mathrm{cov}(\mathbf{f}_\star) &= K(X_\star, X_\star) - K(X_\star, X)[K(X, X) + \sigma^2 I]^{-1} K(X, X_\star) \end{aligned}$$

Figure 1 shows an example of samples from a squared exponential Gaussian Process prior and the updated mean functions after some points have been observed.

If we look at how to generate predictions for single points, we get the following equation of the expectation for new points.

$$\mathbf{f}_\star(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i k(\mathbf{x}_i, \mathbf{x}_\star)$$

with $\alpha = \left(K(X, X) + \sigma^2 I\right)^{-1} \mathbf{y}$. What this equation tells us is that a Gaussian Process can be written as a sum of basis functions. This means that a potentially infinitely parametrized model boils down to a finite sum when making predictions. This sum only depends on the chosen kernel and the data observed thus far (Kac and Siegert, 1947).

This is also why Gaussian Process regression is referred to as non-parametric. It is not the case that this regression approach has no parameters. Instead, it has potentially infinitely many parameters (parametrized by the chosen kernel), but only manifests itself by a finite sum when making

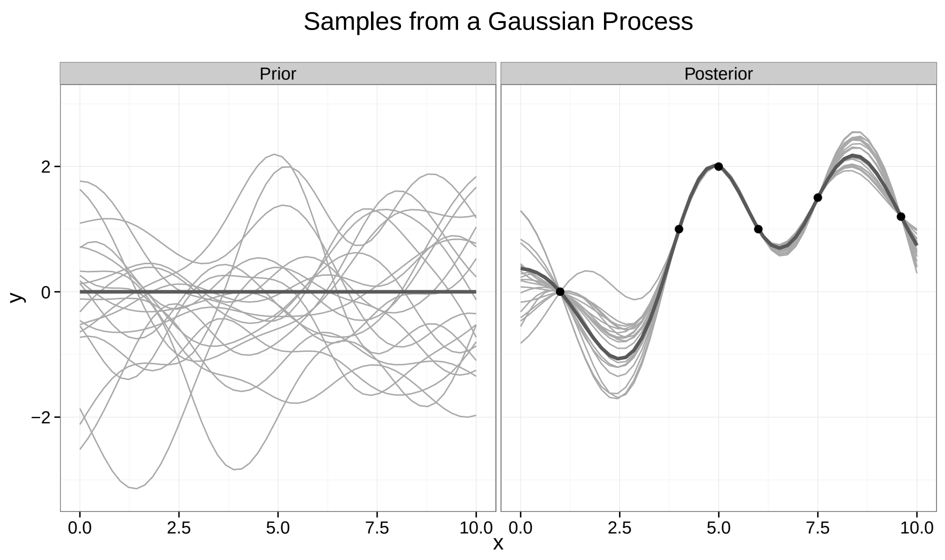


Figure 1: Example of samples from a Gaussian Process prior and posterior. Grey lines inidcate samples from the GP. Black dots mark empirical observations. The dark grey line marks the current mean of the GP.

predictions. Therefore, Gaussian Process regression is a powerful tool to capture many stationary functions. This in turn can be easily applied to contexts where the task is to explore or exploit these functions sequentially.

3.4. General setup

Having found a good model to learn functions, that is our first ingredient for successful sequential function learning, we now have to find a way to smartly explore or exploit the function we are learning over time. Within the Gaussian Process approach both pure *exploration* and *exploitation* can be viewed as two sides of the same coin. They both take a Gaussian Process to model an underlying unknown function [1] and then estimate the utility of all available queries (candidate points from which to sample next) by using what is called an acquisition function. An acquisition function can be seen as a measurement of usefulness (or utility) that candidate points under consideration promise to produce. The approach then goes on to choose as the next point the one that currently produces the highest utility. The way the general set up works is shown in Algorithm 1.

Algorithm 1 General $\mathcal{GP}$ optimization Algorithm

```
Require: Input space $\mathcal{D} = \{X, \mathbf{y}\}$; $\mathcal{GP}$-prior $\mu_0 = 0$, $\sigma_0$, $k$
  for $t = 1, 2, \ldots$ do
    Choose $x_t^* = \text{argmax } f_{\text{Acquisition}}(\mathbf{x})$
    Sample $y_t = f(x_t^*) + \epsilon_t$
    Perform Bayesian update for $\mu_t$ and $\sigma_t$
  end for
```

This algorithm starts out with a Gaussian Process distribution over functions, then assesses the usefulness of the available samples by utilizing the acquisition function and selects the point that currently maximizes this function. Afterwards, the new output at the chosen sample point is observed, the Gaussian Process is updated, and the process starts anew.

[1]Sometimes the Gaussian Process here is referred to as a surrogate model (Gramacy and Lee, 2008).

4. Exploration and Optimal Design

4.1. Acquisition function

The goal in an optimal design setting is to learn an unknown function as well and quickly as possible. As said before, this is the same as the attempt to maximize information gain over all trials. For a Gaussian, the entropy $H(\mathcal{N}(\mu,\Sigma) = \frac{1}{2}\log|2\pi e\Sigma|$, which means that in our setting $I(\mathbf{y};f) = \frac{1}{2}\log|I+\sigma^{-2}K|$, where $K = [k(x,x')]$. Even though finding the overall information gain maximizer is NP-hard, it can be approximated by an efficient greedy algorithm based on Gaussian Processes. If $F(A) = I(\mathbf{y}_A;f)$, then this algorithm picks $x_t = \text{argmax}F(A_{t-1}\cup\{\mathbf{x}\})$. This in turn can be shown to be equivalent to the following acquisition function

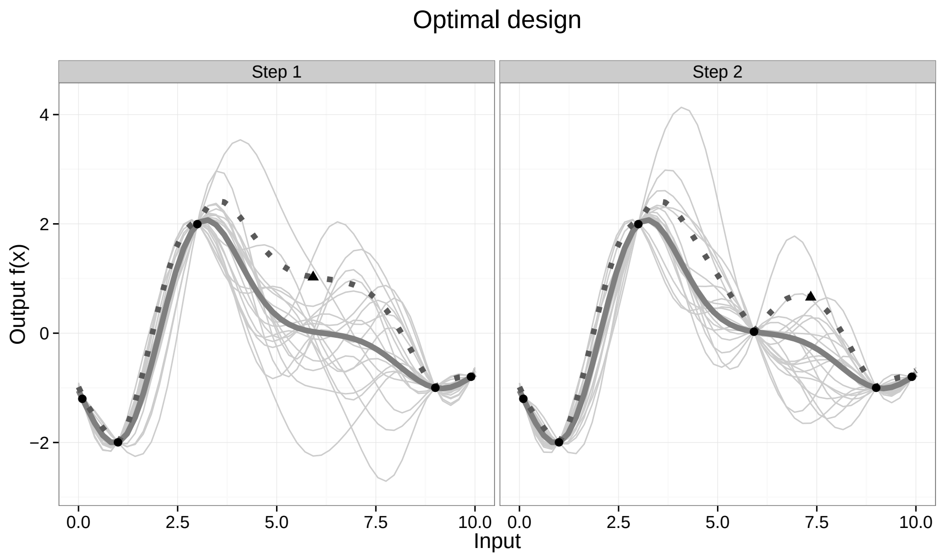
$$f_{\text{Acquisition}}(\mathbf{x}_t) = \text{argmax } \sigma_{t-1}(\mathbf{x}) \tag{2}$$

What this algorithm does, is to start out with a Gaussian Process prior, and to sequentially sample from the points that currently show the highest uncertainty in a greedy fashion. Even though this algorithm sounds naïve at first, it can actually be shown to converge to at least a constant fraction of the maximum information gainer (Krause, Singh, and Guestrin, 2008).

$$F(A_T) \geq \left(1-\frac{1}{e}\right)\max F(A) \tag{3}$$

This is based on a property of the acquisition function called *submodularity* (Krause and Golovin, 2012). Intuitively, submodularity here means that information never hurts (it is always helpful to observe more points), but also that the usefulness of newly acquired points decreases the more points have been sampled before. This diminishing returns property is crucial to show that the greedy algorithm can be successful over all (see Appendix). A simple example of the Gaussian Process uncertainty reduction sampler is shown in Figure 2 below. We have sampled a function from a Gaussian Process prior (a squared exponential) and let the algorithm select observations by picking as the next obvservation the one that currently has the highest standard deviation attached.

Figure 2: GP-uncertainty reduction example. The dark grey line marks the current mean of the GP. The dashed line shows the upper part of the standard deviation. The light grey lines are samples from the GP.



4.2. Example: Learning unknown functions

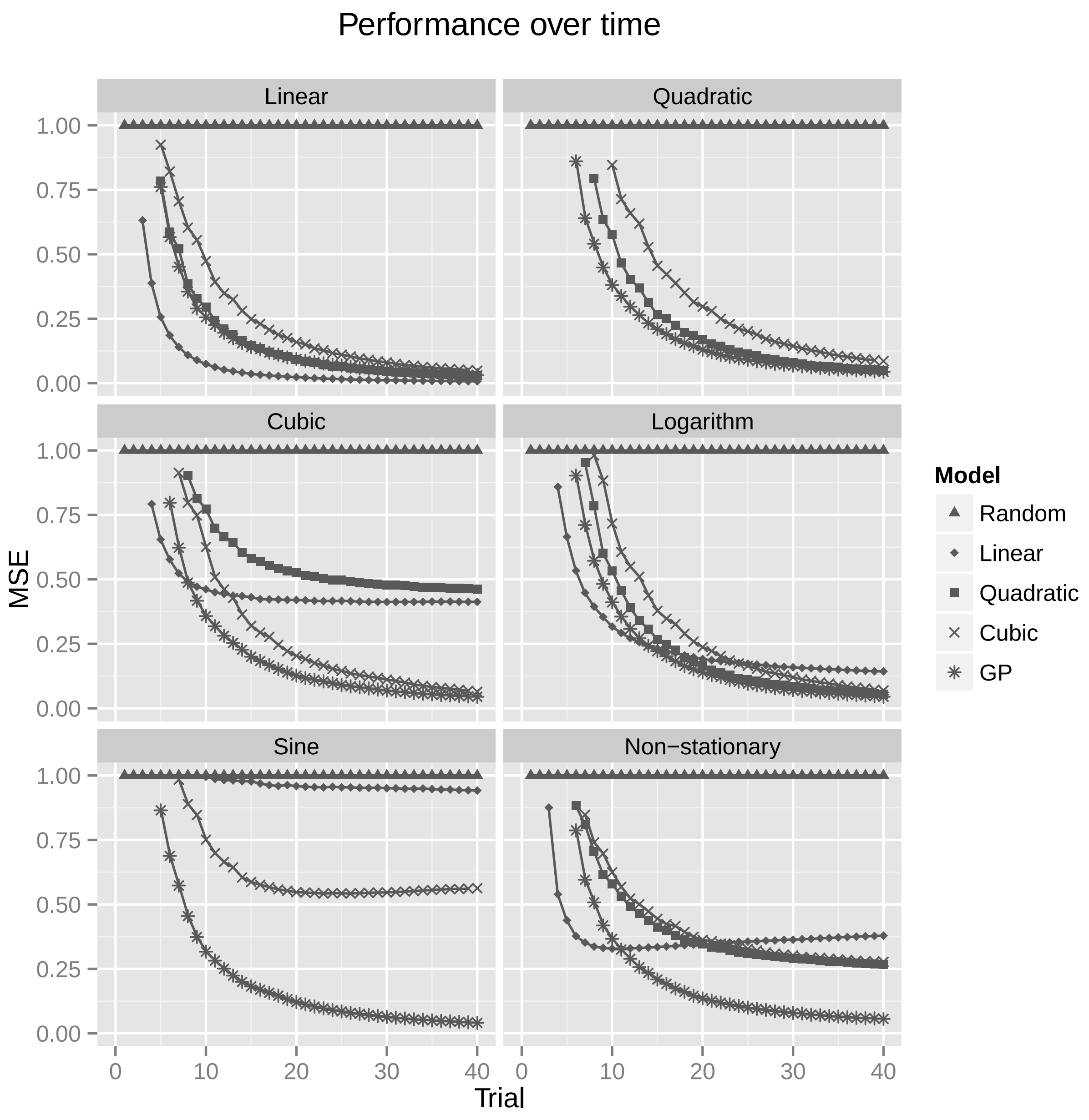
In order to demonstrate how Gaussian Process based exploration works, let us simulate how the algorithm learns unknown functions and compare it to other algorithms. If we assume that we have to learn an unknown function as quickly as possible and that the function f only takes a one-dimensional input $x = [0, 0.1, 0.2, \ldots, 10]$ and to which it maps an output y, then we can set up different functions that the models can learn actively over time. As the GP is considered to learn many different functions well, we will test the following different functions: a linear, quadratic, cubic, logarithmic, sine, and a non-stationary function (see Appendix for details).

We have deliberately chosen different parametric forms to assess the learning methods. The models we have used to learn the different function are a linear, a quadratic, a cubic, and a Gaussian Process (with a squared exponential kernel) regression. Each model was set up to learn the underlying function by picking as the next observation the one that currently has the highest uncertainty (standard deviation of the predicted mean). We let each model run 100 times for each underlying function and averaged the mean squared error over the whole discretized input space for each step. Results are shown in Figure 3.

It can be seen that the Gaussian Process model learns all functions both efficiently and well. Only in the cases in which the used learning function is indeed the same as the learning function (for example, using a linear function to learn an underlying linear function), does another model learn faster than the Gaussian Process.

This means that Gaussian Processes are especially useful in cases where the underlying function is not known or can be ignored. For examle, one could easily use Gaussian Processes to learn participants' utility function over different experiments or simply use them to generate stimuli that are most informative overall.

Figure 3: GP-uncertainty reduction example.



5. Exploration-Exploitation and Bayesian Optimization

In an exploration-exploitation scenario the goal is to find the argument to a function that produces the maximum output as quickly as possible. One way to measure the quality of this search process is to quantify regret. Regret is the distance between the output of the currently chosen argument and the maximum output.

$$r(\mathbf{x}) = f(\mathbf{x}^*) - f(\mathbf{x}) \tag{4}$$

The goal then is to minimize regret over all available trials.

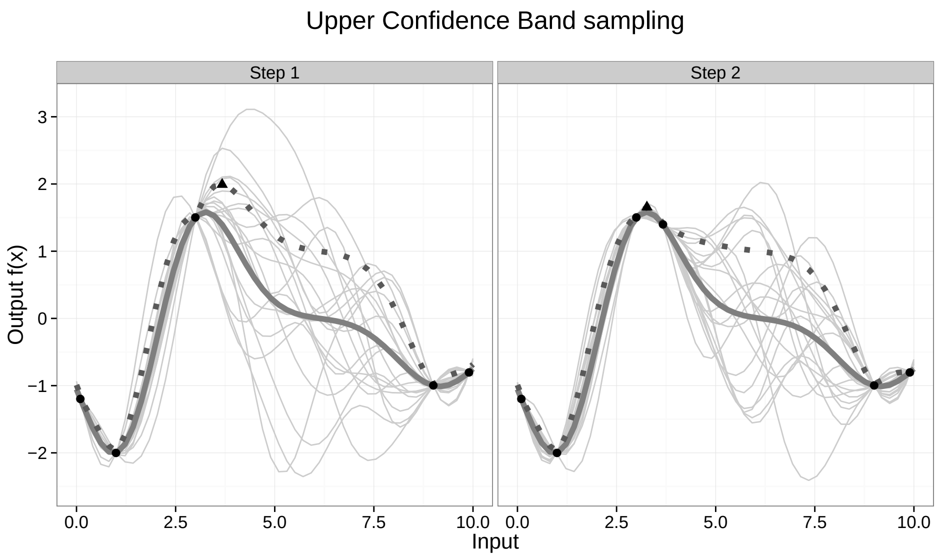
$$\min \sum_{t}^{T} r(x_t) = \max \sum_{t}^{T} f(x_t) \tag{5}$$

Again, finding the global function maximizer is NP hard. That is finding the minimum sequence of queries that leads to the lowest regret overall is almost impossible. However, there is again a greedy trick one can apply in this scenario. This trick is based on redefining the function maximization problem as a bandit task. In a bandit tasks the goal is to maximize output by playing the right arm out of many available arms (it is named after the one armed-bandits that can be found in casinos). As the tasks is to maximize output of a function, the discretized input points to that functions are seen as arms, that are correlated in dependency of the underlying covariance kernel. Taking this perspective, one easy way to approach these kind of problems is the following acquisition function called Upper Confidence Band sampling.

$$f_{\text{Acquisition}}(\mathbf{x}) = \mu_{t-1}(\mathbf{x}) + k\sigma_{t-1}(\mathbf{x}) \tag{6}$$

Upper Confidence Band sampling (UCB) plays the arm that currently shows the highest upper confidence interval. This strategy is sometimes called naive optimism in the face of uncertainty. Intuitively, the upper confidence band is determined by two factors, the current estimate of the mean at a particular point (the higher the estimate, the higher the band) and the uncertainty attached to that estimate (the higher the uncertainty, the higher the band). Therefore, the UCB algorithm trades off naturally between expectations and uncertainties. An example of how the UCB works is shown in Figure 4. Again, we have sampled a function from a Gaussian Process prior and have applied the algorithm to this function.

Figure 4: GP-UCB example. The dark grey line marks the current mean of the GP. The dashed line marks the GP's upper confidence bound. The light grey lines are samples from the GP.



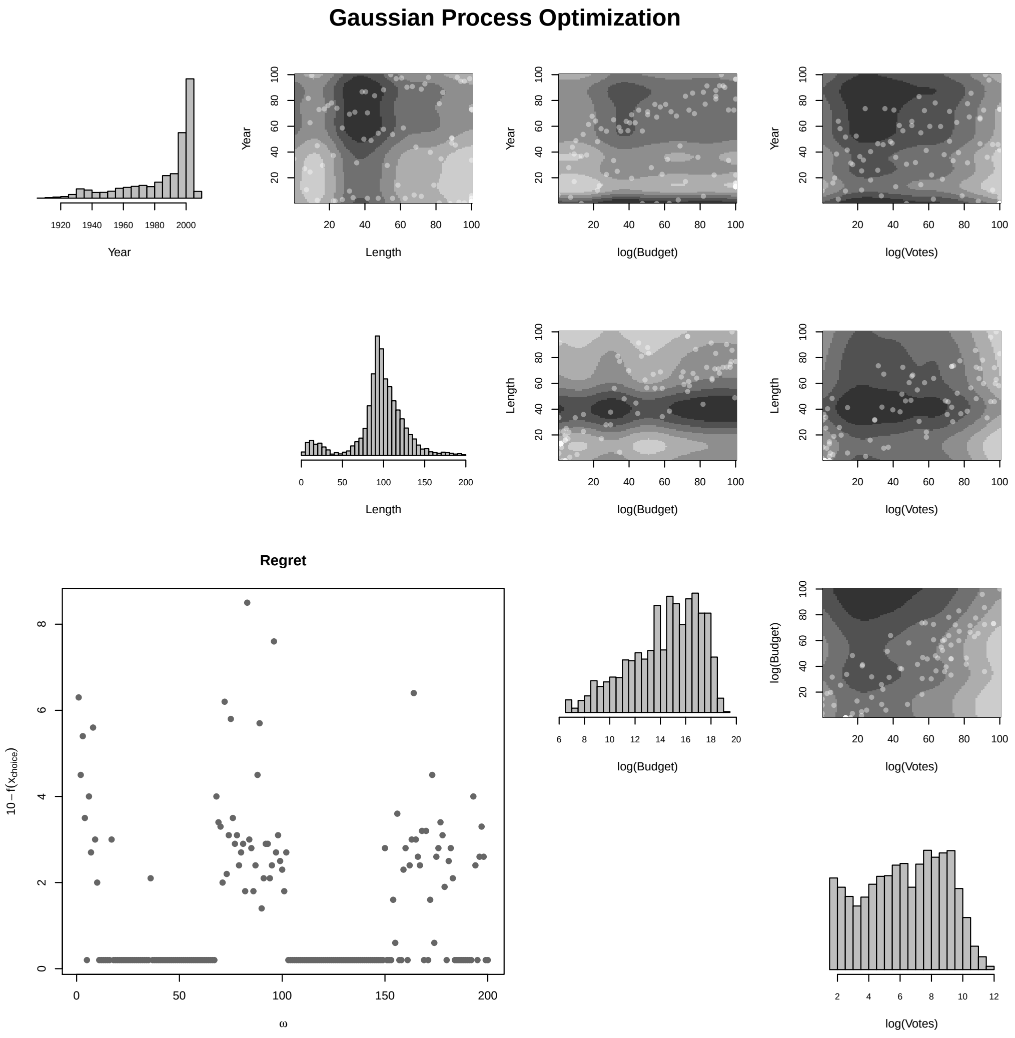
Even though the greedy UCB strategy is again naïve, it can be shown that its regret is sublinear, again based on the submodularity of the overall information gain (Srinivas, Krause, Kakade, and Seeger, 2009).

5.1. GP-UCB Example: Choosing movies

We use the imdb-movie data as an example for doing GP-UCB exploratio-exploitation. For this, we extracted 5141 movies from the data base, including the year they appeared, the budget that was used to make them, their length, as well as how many people had evaluated the movie on the imdb. The dependent variable was the actual imdb score and we set up a GP-UCB with a squared exponential kernel, set $\beta = 3$ and let the algorithm pick 200 movies sequentially. Results are shown in Figure 5.

It can be seen that the algorithm quickly starts picking movies that produce low to zero regret. However, it still keeps exploring other movies from time to time, just to alqays come back to recommending movies with really high scores. Additionally, it explores all of the given input areas for quite some time, just to find good areas in the end.

Figure 5: GP-UCB example.



6. Safe Exploration-Exploitation

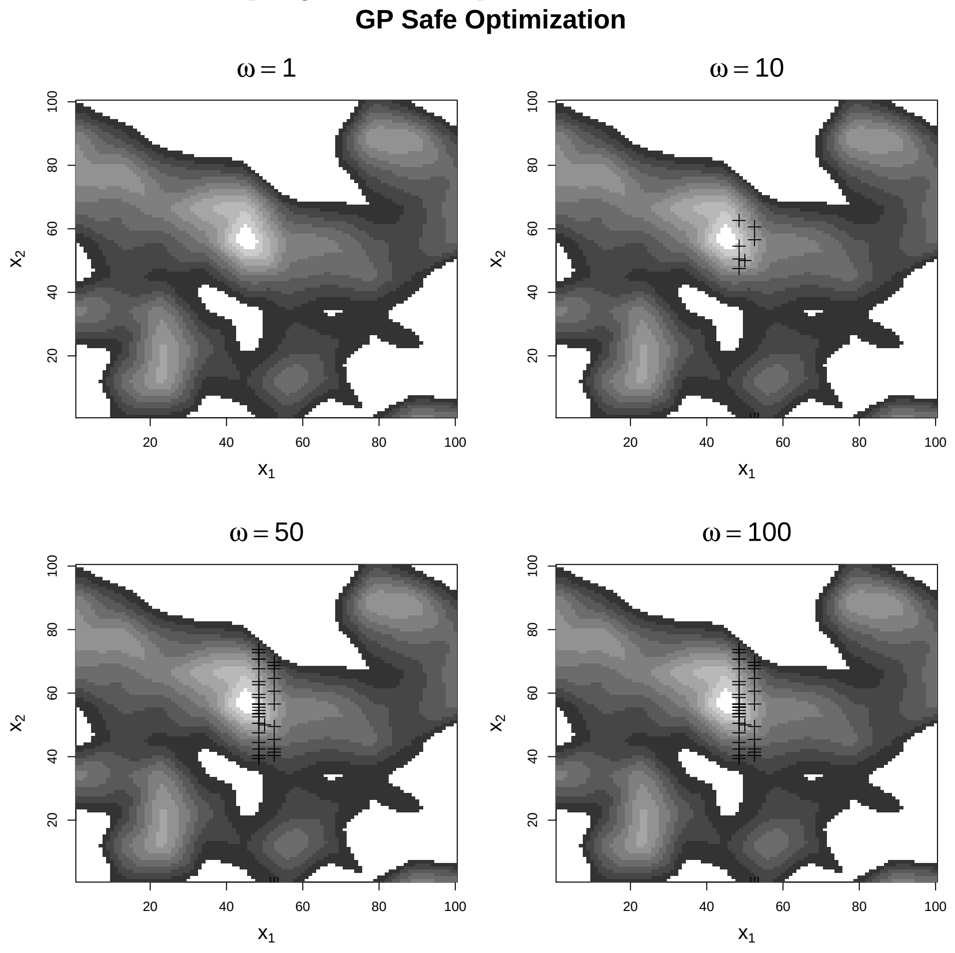
Sometimes an experimentalist's goal might not only be to learn or maximize a function, but additional requirements can also be important. One such requirement, for example, can be to avoid certain outputs as much as possible. One example for such a scenario is excitatory stimulation, especially in a physiologically setting. Imagine a medical scenario, where the task is to stimulate the spinal chord in such a way that certain movements are achieved (Desautels, 2014). Here, it is important to stimulate the spinal chord such that optimal recovery is achieved, but not too much as this might lead to painful reactions within the patients. Again, Gaussian Process optimization methods can be used here to learn the underlying function of what stimulations lead to what strength of reaction. However, an additional requirement now is to avoid these expectedly painful areas. It turns out that there is a smart way to adapt the GP-UCB approach described above while accommodating for this additional requirement. This is acchieved by an algorithm called SafeOpt (Sui, Gotovos, Burdick, and Krause, 2015). Leaving the detailed technical explanation of this algorithm for the interested the reader to look up, this algorithm basically works by trading-off two different things. Firstly, it keeps a set of safe options it considers to be above the given threshold and tries to expand this set as much as it can. Secondly, it maintains a set of potential maximizers that, if used as an input, would potentially achieve the highest output. It then choses as the next point, a point within the intersection of these two sets that has the highest predictive variance.

6.1. Example: Stimulus Optimisation

Imagine that you have to optimize a stimulus that can be described by two dimensions X_1 and X_2. This means you have to find the maximum output y. However, you also want to avoid sampling values below 0 at all costs. For this we have drawn a two dimensional function from a Gaussian Process prior with a squared exponential kernel. This can be seen as a similar to the case where one want to present stimuli to participants, but make sure that participants never react with an intensity below a certain threshold. Results are shown in Figure 6.

It can be seen that the SafeOpt algorithm finds the maximum of the function well, but also tries to expand the space of possible inputs more and more. At the same time, the algorithm does not sample from the white area (values below 0) for a single time. We therefore think that this algorithm could potentially applied in many optimal design settings that require additional constraints.

Figure 6: GP-SafeOpt example. White areas represent areas below 0. The black crosses show where the SafeOpt algorithm has sampled.



7. Gaussian Processes and cognition

As we have seen, Gaussian Processes (in combination with smart acquisition functions) are a powerful tool to learn and maximize unknown functions. However, they might also be applied in a different, even more psychological context, that is as as a model for human cognition within the tradition of Bayesian cognitive science (Chater and Oaksford, 2008). Non-parametric Bayesian approaches have been used before to describe associative learning (Gershman and Niv, 2012) and categorization learning (Kemp, Tenenbaum, Griffiths, Yamada, and Ueda, 2006), among others. Recently, Lucas, Griffiths, Williams, and Kalish (2015) have proposed to use Gaussian Processes as a rational model of human function learning for passive learning tasks. Schulz, Tenenbaum, Reshef, Speekenbrink, and Gershman (2015c) used Gaussian Processes to assess participants judgments about the predictability of functions in dependency of the smoothness of the underlying kernel. Taking recent developments within the active learning community into account, we expect Gaussian Processes to soon become a fruitful descriptive model for human cognition in many different domains where participants have to act actively, that is to select information sequentially. In a first attempt, we have found that participants' behavior can be well described by Gaussian Process algorithms when the task is to learn and maximize a function both in a static (Schulz, Konstantinidis, and Speekenbrink, 2015b) as well as in a dynamic environment (Schulz, Konstantinidis, and & Speekenbrink, 2015a). However, given GP's expressiveness and the mathematical guarantees that come with them, we expect them to be used more frequently as a model for human cognition in the near future and hope that this tutorial can help people to apply them more often.

Here, it is not only the assumption of not having to choose a parametric shape a priori, but letting the data speak directly, that makes these models powerful as a psychological model. It is also the attached measure of uncertainty that comes for free when doing computations with Gaussian Processes, a characteristic more and more used in numerics (Hennig, Osborne, and Girolami, 2015) and optimization problems (Hennig and Kiefel, 2013). The resulting uncertainty of computation can then also be propagated between different systems easily (Damianou and Lawrence, 2012). Using this fact, one could build of a model that involves control, learning, and optimization and in which uncertainty is modeled over the whole system, sort of a Bayesian Cognitivism approach.

8. Discussion

This tutorial has introduced Gaussian Process regression as a general purpose inference engine to learn about (explore) and maximize (exploit) unknown functions. Gaussian Processes have been introduced mathematically and the greedy variance minimization exploration algorithm as well as the upper confidence bound sampling method for exploration-exploitation scenarios have been introduced. Within a simulated exploration experiment we have shown how Gaussian Processes can be used to efficiently learn about unknown functions in an optimal design set-up and gradually presented informative stimuli. Within a parameter tuning example, we have shown how GP-UCB can outperform other commonly used methods when the goal is to optimize hyper-parameters of a neural network. Additionally, we have talked about introducing additional requirements to the used acquisition function and have shown one example within a scenario, where the task was to show the optimal excitatory stimulus while avoiding some areas of the input space. Finally, we have talked about utilizing Gaussian Process models as actual models of cognition in the hope that they will be picked up and used more frequently in the near future.

Of course a tutorial like this can never be fully comprehensive. Therefore, let us briefly point out some things that we have not covered here. First of all, using the Gaussian Process regression approach as we have parametrized here is only one possible approach towards regression problems. In fact, it can be shown that many standard Bayesian regression approaches can be re-parametrized to be equivalent to Gaussian Process regression, given specific assumptions about the kernel (Duvenaud, Lloyd, Grosse, Tenenbaum, and Ghahramani, 2013). The two chosen utility functions here also are just two options out of a pool of different acquisition functions. Another commonly used acquisition function for the pure exploration case is the attempt to minimize the expected variance summed up over the whole input space (Gramacy and Apley, 2014). Even though this method tends to sample less from the boundary cases as compared to the algorithm introduced here, it can be hard to compute, especially if the input space is large. There exist many different acquisition functions in the exploration-exploitation context, that are mostly discussed under the umbrella term Bayesian optimization (de Freitas, Smola, and Zoghi, 2012). One other common acquisition function that is frequently used here is the expected probability of improvement (Mockus, 2012), which choses as the next point the one that promises to have the highest likelihood of improving the currently expected maximum. Again, this method of assessing candidate points can be computationally expensive and mathematical guarantees are only given for the UCB algorithm. Last not least, there are also many different acquisition functions with additional constraints available. Important to mention hereby are Bayesian black box optimization with additional constraints, the estimation of level sets, as well as Bayesian Quadrature-based algorithms.

Here we have introduced Gaussian Process-based algorithms to explore and exploit unknown functions. We hope that this tutorial will help others to pick up these methods and that their usage will gradually become more common within psychology.

9. Software packages

Table 1 below contains further Gaussian Process software pointers.

Table 1: Gaussian Process Software Packages

Name	**Algorithm**	**Language**	**Author**
GPML	GP Toolbox	Matlab	Rasmussen & Williams
SFO	Submodular Optimization	Matlab	Krause
GPy	GP Toolbox	Python	Sheffield ML Group
gptk	GP Toolbox	R	Lawrence
tgp	Tree GPs, GP regression	R	Gramacy & Taddy

References

Borji, A., Itti, L., 2013. Bayesian optimization explains human active search. In: Advances in neural information processing systems. pp. 55–63.

Brochu, E., Cora, V. M., De Freitas, N., 2010. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. arXiv preprint arXiv:1012.2599.

Cavagnaro, D. R., Aranovich, G. J., McClure, S. M., Pitt, M. A., Myung, J. I., 2014. On the functional form of temporal discounting: An optimized adaptive test.

Cavagnaro, D. R., Myung, J. I., Pitt, M. A., Kujala, J. V., 2010. Adaptive design optimization: A mutual information-based approach to model discrimination in cognitive science. Neural computation 22 (4), 887–905.

Chater, N., Oaksford, M., 2008. The probabilistic mind: Prospects for Bayesian cognitive science. Oxford University Press.

Damianou, A. C., Lawrence, N. D., 2012. Deep gaussian processes. arXiv preprint arXiv:1211.0358.

Daunizeau, J., Preuschoff, K., Friston, K., Stephan, K., 2011. Optimizing experimental design for comparing models of brain function. PLoS Comput Biol 7 (11), e1002280.

de Freitas, N., Smola, A., Zoghi, M., 2012. Regret bounds for deterministic gaussian process bandits. arXiv preprint arXiv:1203.2177.

Desautels, T. A., 2014. Spinal cord injury therapy through active learning. Ph.D. thesis, California Institute of Technology.

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

Gershman, S. J., Blei, D. M., 2012. A tutorial on bayesian nonparametric models. Journal of Mathematical Psychology 56 (1), 1–12.

Gershman, S. J., Niv, Y., 2012. Exploring a latent cause theory of classical conditioning. Learning & behavior 40 (3), 255–268.

Goos, P., Jones, B., 2011. Optimal design of experiments: a case study approach. John Wiley & Sons.

Gramacy, R. B., Apley, D. W., 2014. Local gaussian process approximation for large computer experiments. Journal of Computational and Graphical Statistics (just-accepted), 1–28.

Gramacy, R. B., Lee, H. K., 2008. Bayesian treed gaussian process models with an application to computer modeling. Journal of the American Statistical Association 103 (483).

Hennig, P., Kiefel, M., 2013. Quasi-newton methods: A new direction. The Journal of Machine Learning Research 14 (1), 843–865.

Hennig, P., Osborne, M. A., Girolami, M., 2015. Probabilistic numerics and uncertainty in computations. Proc. R. Soc. A 471 (2179), 20150142.

Jäkel, F., Schölkopf, B., Wichmann, F. A., 2007. A tutorial on kernel methods for categorization. Journal of Mathematical Psychology 51 (6), 343–358.

Kac, M., Siegert, A., 1947. An explicit representation of a stationary gaussian process. The Annals of Mathematical Statistics, 438–442.

Kemp, C., Tenenbaum, J. B., Griffiths, T. L., Yamada, T., Ueda, N., 2006.

Learning systems of concepts with an infinite relational model. In: AAAI. Vol. 3. p. 5.

Ko, C.-W., Lee, J., Queyranne, M., 1995. An exact algorithm for maximum entropy sampling. Operations Research 43 (4), 684–691.

Krause, A., Golovin, D., 2012. Submodular function maximization. Tractability: Practical Approaches to Hard Problems 3, 19.

Krause, A., Singh, A., Guestrin, C., 2008. Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies. The Journal of Machine Learning Research 9, 235–284.

Lucas, C. G., Griffiths, T. L., Williams, J. J., Kalish, M. L., 2015. A rational model of function learning. Psychonomic bulletin & review, 1–23.

Meder, B., Nelson, J. D., 2012. Information search with situation-specific reward functions. Judgment and Decision Making 7 (2), 119–148.

Mockus, J., 2010. Bayesian Heuristic approach to discrete and global optimization: Algorithms, visualization, software, and applications. Springer-Verlag.

Mockus, J., 2012. Bayesian approach to global optimization: theory and applications. Vol. 37. Springer Science & Business Media.

Myung, J. I., Pitt, M. A., 2009. Optimal experimental design for model discrimination. Psychological review 116 (3), 499.

Rasmussen, C. E., 2006. Gaussian processes for machine learning.

Schulz, E., Konstantinidis, E., & Speekenbrink, M., 2015a. Learning and decisions in contextual multi-armed bandit tasks. In: Proceedings of the Thirty-Seventh Annual Conference of the Cognitive Science Society.

Schulz, E., Konstantinidis, E., Speekenbrink, M., 2015b. Exploration-exploitation in a contextual multi-armed bandit task. In: International Conference on Cognitive Modeling. pp. 118–123.

Schulz, E., Tenenbaum, J. B., Reshef, D. N., Speekenbrink, M., Gershman, S. J., 2015c. Assessing the perceived predictability of functions. In: Proceedings of the Thirty-Seventh Annual Conference of the Cognitive Science Society.

Snoek, J., Larochelle, H., Adams, R. P., 2012. Practical bayesian optimization of machine learning algorithms. In: Advances in neural information processing systems. pp. 2951–2959.

Srinivas, N., Krause, A., Kakade, S. M., Seeger, M., 2009. Gaussian process optimization in the bandit setting: No regret and experimental design. arXiv preprint arXiv:0912.3995.

Sui, Y., Gotovos, A., Burdick, J., Krause, A., 2015. Safe exploration for optimization with gaussian processes. In: Proceedings of the 32nd International Conference on Machine Learning (ICML-15). pp. 997–1005.

Wetzels, R., Vandekerckhove, J., Tuerlinckx, F., Wagenmakers, E.-J., 2010. Bayesian parameter estimation in the expectancy valence model of the iowa gambling task. Journal of Mathematical Psychology 54 (1), 14–27.

Williams, C. K., 1998. Prediction with gaussian processes: From linear regression to linear prediction and beyond. In: Learning in graphical models. Springer, pp. 599–621.