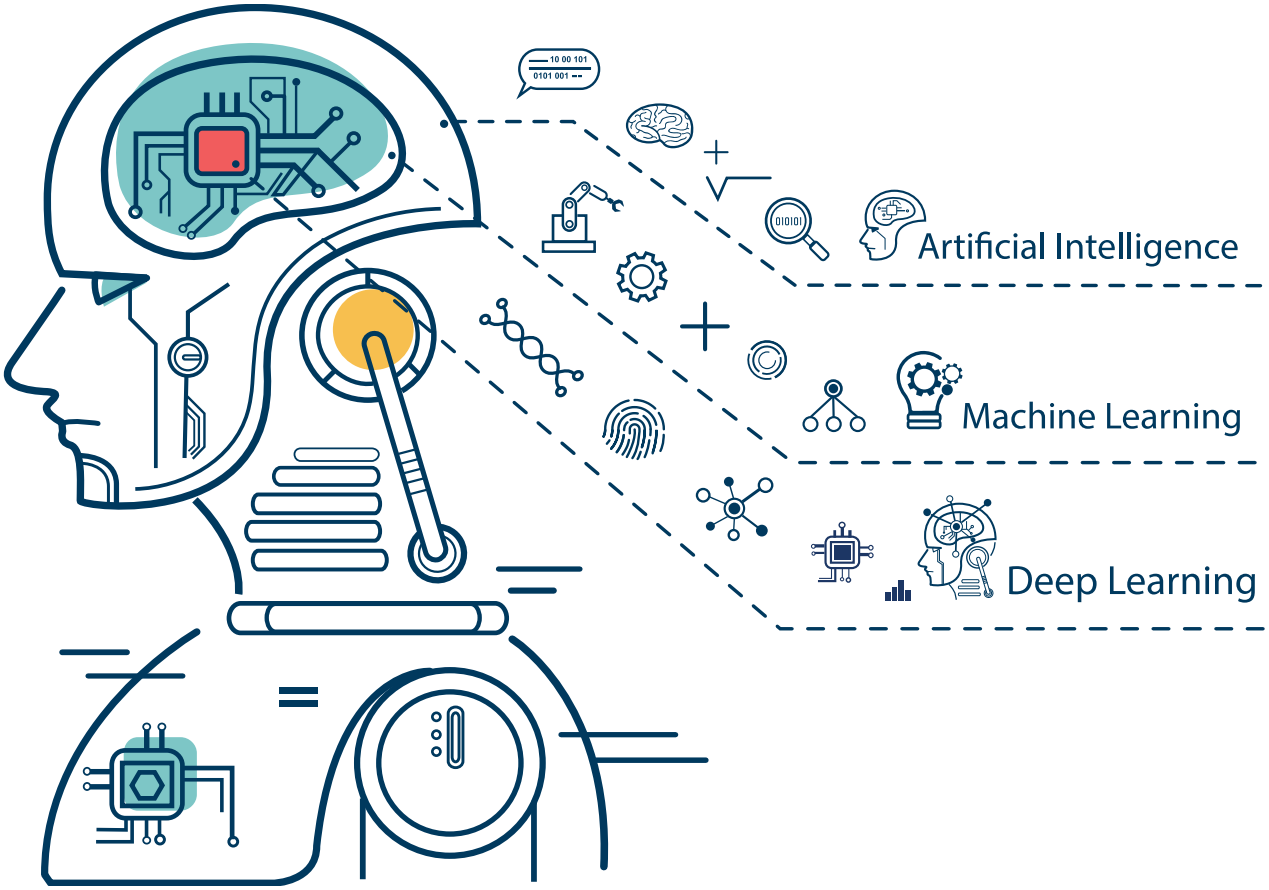
mind foundry

THE INTUITIONS BEHIND BAYESIAN OPTIMIZATION WITH GAUSSIAN PROCESSES

In certain applications the objective function is expensive or difficult to evaluate. In these situations, a general approach consists in creating a simpler surrogate model of the objective function which is cheaper to evaluate and will be used instead to solve the optimization problem. Moreover, due to the high cost of evaluating the objective function, an iterative approach is often recommended. Iterative optimizers work by iteratively requesting evaluations of the function at a sequence of points in the domain. Bayesian Optimization adds a Bayesian methodology to the iterative optimizer paradigm by incorporating a prior model on the space of possible target functions. This guide introduces the basic concepts and intuitions behind Bayesian Optimization with Gaussian Processes and introduces Mind Foundry Optimize, an API for Bayesian Optimization.



OPTIMIZATION

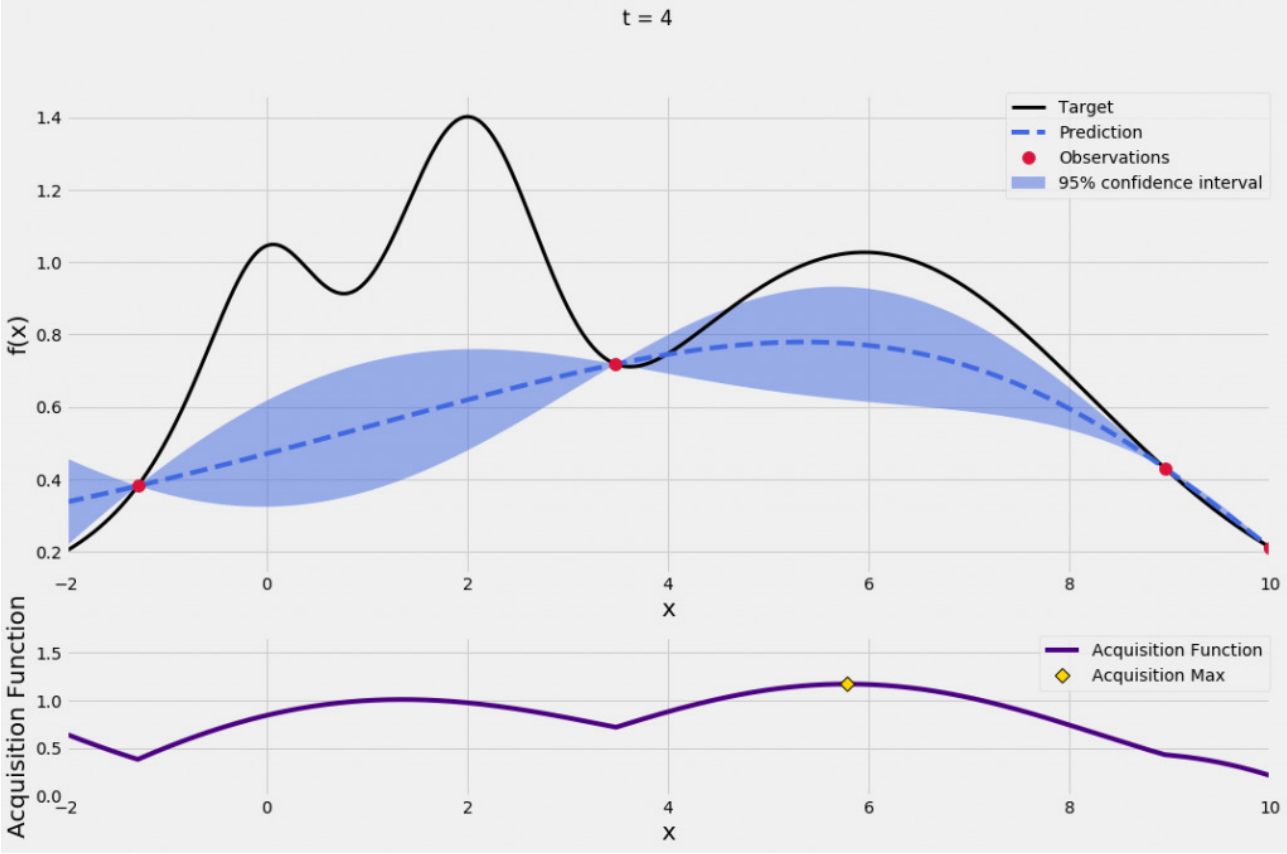
Optimization methods try to locate an input χ^* in a domain χ to a function f which maximizes (or minimizes) the value of this function over χ:

The general Optimization framework

In practice, the function f represents the outcome of a process that is required to be optimized, such as the overall profitability of a trading strategy, quality control metrics on a factory production line, or the performance of a data science pipeline with many parameters and hyper-parameters.

The input domain χ represents the valid parameter choices for the process needing optimization. These could be market predictors used in a trading strategy, quantities of raw materials used in a factory process, or the parameters of ML models in a data science pipeline. It is the description of the input domain, χ, together with the properties of the function, f, that characterize the optimization problem. The valid inputs to the process domain, χ, may be discrete, continuous, constrained, or any combination of these. Similarly, the outcome function, f, may be convex, differentiable, multi-modal, noisy, slowly changing, or have many other important properties.

In certain applications the objective function is expensive to evaluate (computationally or economically), difficult to evaluate (chemical experiments, oil drilling). In these situations, a general approach consists in creating a simpler surrogate model $\hat{f}$ of the objective function f which is cheaper to evaluate and will be used instead to solve the optimization problem.



Moreover, due to the high cost of evaluating the objective function, an iterative approach is often recommended. Iterative optimizers work by iteratively requesting evaluations of the function fat a sequence of points in the domain $\chi 1, \chi 2, \ldots \in \chi$. Through these evaluations the optimizer is able to build up a picture of the function f. For gradient descent algorithms this picture is local but for surrogate model approaches this picture is global. At any time, or at the end of a pre-allocated budget of function evaluations, the iterative optimizer will be able to state its best approximation to the true value of $\chi*$.

The surrogate model is trained using N known evaluations of $f: F = (f1, f2, \ldots, fN)$ at $XN = (x1, x2, \ldots, xN)$. There are many approaches used for building the surrogate model such as polynomial interpolation, neural networks, support vector machines, random forests and Gaussian processes. At Mind Foundry, our method of choice is regression using Gaussian Processes.

GAUSSIAN PROCESSES

Gaussian Processes (GPs) provide a rich and flexible class of non-parametric statistical models over function spaces with domains that can be continuous, discrete, mixed, or even hierarchical in nature. Furthermore, the GP provides not just information about the likely value of f, but importantly also about the uncertainty around that value.

The idea behind Gaussian Process Regression is for a set of observed values FN at some points XN we assume that these values correspond to the realisation of a multivariate Gaussian Process with a prior distribution:

where KN is a NxN covariance matrix and its coefficients are expressed in terms of a correlation

function (or kernel) $Kmn = K(xm, xn, \theta)$. The hyper- parameters θ of the kernel are calibrated according to a maximum likelihood principle. KN is chosen to reflect a prior assumption of the function and therefore the choice of the kernel will have a significant impact on the correctness of the regression. An illustration of several covariance functions is given in Figure 2.

Through mathematical transformations and using the conditional probability rule it is possible to estimate the posterior distribution $p(fN+1|FN, XN+1)$ and express $fN+1$ as a function of KN and FN with an uncertainty. This allows us to construct a probabilistic surrogate from our observations as illustrated in Figure 1:

BAYESIAN OPTIMIZATION

Bayesian Optimization is a class of iterative optimization methods that focuses on the general optimization setting, where a description of x is available, but knowledge of the properties of f is limited. Bayesian Optimization methods are characterized by two features:

- the ***surrogate model*** f, for the function f,
- and an ***acquisition function*** computed from the surrogate and used for guiding the selection of the next evaluation point

BO adds a Bayesian methodology to the iterative optimizer paradigm by incorporating a ***prior model*** on the space of possible target functions, f. By updating this model every time time a function evaluation is reported, a Bayesian optimization routine keeps a posterior model of the target function f. This posterior model is the surrogate f for the function f. The pseudo code for a Bayesian Optimization routine with a GP prior is:

Initialisation:

- Place a Gaussian process prior on f
- Observe f at n0 points according to an initial space-filling experimental design.

- Set n at n0

While n ≤ N do:

- Update the posterior probability distribution on f using all available data
- Identify the maximiser xn of the acquisition function over x, where the acquisition function is calculated using the current posterior distribution
- Observe $yn = f(xn)$
- Increment n

End while

Return either the point evaluated with the largest $f(x)$ or the point with the largest posterior mean. An example of a standard acquisition function is the ***Expected Improvement Criterion (EI)***, which, for any given point in $x \in x$ is the expected improvement in the value of f at x over the best value of f

yet seen, given that the function f at x is indeed higher than the best value of f yet seen: so if we are finding the maxima of f, EI can be written as:

$$EI(x) = IE(max(f(x) - f^*, 0))$$

where f^* is the maximum value of f seen so far.

Additional examples of acquisition functions are:

- Entropy search which consists in seeking to minimise the uncertainty we have in the location of the optimal value
- Upper confidence bound
- Expected loss criterion

Figure 3 illustrates the evolution of the surrogate and its interactions with the acquisition function as it improves its knowledge after each iteration of the underlying function it is trying to minimize.

OPERATIONALIZING BO WITH MIND FOUNDRY OPTIMIZE

Mind Foundry Optimize is a general purpose Bayesian optimizer which provides optimal parameter configurations via web-services. It can handle any parameter type and does not need to know the underlying process, models, or data. It requires the client to specify the parameters and their domains, and to post back the accuracy scores for each of the recommended parameter configurations. Mind Foundry Optimize uses these scores to model the underlying system and search for optimal configurations faster.

Mind Foundry have implemented an ensemble of surrogate models and acquisition functions within the product that it will automatically select and configure based on the nature and number of parameters it is provided as described in the Figure 4. This selection is based on thorough scientific testing and research so that the optimizer always makes the most appropriate choices. Moreover, Mind Foundry has the ability to design custom covariance functions for a client's specific problem which will significantly increase the speed and accuracy of the optimization process. Most users of Mind Foundry Optimize require the optimization of complex processes, which are expensive to run, and give limited feedback. For this reason, the product focuses its API on providing a simple iterative optimizer interface. However, if more information is available about the process being optimized, it can always be leveraged for faster convergence towards the optimum. Therefore, Mind Foundry Optimize also supports the communication of information about the domain x, such as constraints on inputs, and about the evaluations of the function f, such as noise or gradients or partially complete evaluations. Furthermore, clients are often able to leverage local infrastructure to distribute the optimization search, and can also be requested for batches to evaluate together.

The optimization process is the following:

1. Mind Foundry Optimize recommends a configuration to the customer
2. The customer evaluates the configuration on their machines
3. The customer sends back a score (accuracy, Sharpe ratio, Return on investment,...)
4. Mind Foundry Optimize uses the score to update its surrogate model and the cycle repeats until the optimal configuration has been reached.

During the whole process, Mind Foundry Optimize does not access the underlying data or models.

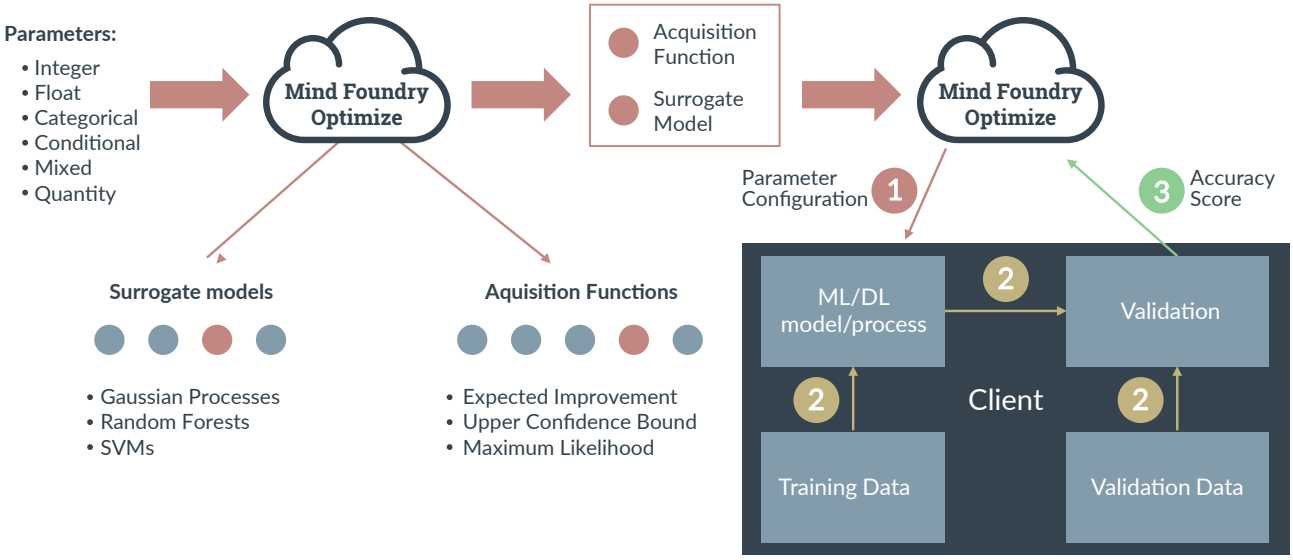


Figure 4: Optimization procedure with Mind Foundry Optimize

THIS GUIDE IS BROUGHT TO YOU BY MIND FOUNDRY

Organisations looking to harness the potential of their data are turning to Mind Foundry's human augmented machine learning solution to uncover answers to their business questions. Business problem owners use our easy-to-navigate SaaS platform to prepare their data, optimize it for machine learning, generate a model, test it, and then make predictions for business issues such as lead conversion, buyer profile fit, customer churn, credit risk, clinical trial outcomes and more. We co-create an achievable action plan with each customer, prioritizing opportunities for applying machine learning to maximize value creation in the shortest time possible.

Mind Foundry is a portfolio company of the University of Oxford, founded in 2015 by Stephen Roberts and Mike Osborne, two of the greatest minds in machine learning research and development. Investors include Oxford Sciences Innovation, the Oxford Technology and Innovations Fund, the University of Oxford Innovation Fund and Parkwalk Advisors.

For more information about how machine learning or Mind Foundry can help your organisation, please contact our team: info@mindfoundry.ai