Explanation of Mathematical Methods Behind the First Half of the Project

AUTHOR
Emma Bowen

Introduction

What mathematical methods will I explain in this document?

POSSIBLY WANT TO SHOW THE BITS OF CODE ASSOCIATED WITH EACH METHOD IN THIS DOCUMENT - SHOW IN EACH SECTION, JUST NOTING HERE

Context

We are building on the paper "Using history matching to speed up management strategy evaluation grid searches". This paper is looking to find the Harvest Control Rule parametrised by F_{target} and $B_{trigger}$ that maximises the catch whilst keeping the risk below 0.05 for a single-stock fishery. The paper does not consider fleet dynamics. We have an objective function in the paper which combines the risk and the catch and so this is the function we want to maximise to get our maximal catch with the constraint of keeping the risk below 0.05.

To do this, the paper takes a Bayesian History Matching (BHM) approach. Firstly, we sample our first eight points which are spaced evenly throughout the sample space to get some initial data. Then, for each round we do the following:

We repeat this process until there is only one plausible point left and then we will accept this as being the F_{target} and $B_{trigger}$ that maximise the catch whilst keeping the risk below 0.05.

Now, we will look at the mathematics behind the methods above and also at the acquisition functions of Expected Improvement, Augmented Expected Improvement and Knowledge Gradient which I added to the original code.

Gaussian Processes

NEED TO ADD/CHANGE HERE

We use a GP to emulate the objective function because it is much cheaper to evaluate compared to our objective function. If we let the emulator be denoted as $\hat{f}$ then we can calculate $\hat{f}(x)$ for any x in the design space as our estimate of $f(x)$ based on our current beliefs. This is true even for the evaluated points $x_1, \dots, x_n$ as the emulator is fitted to these points.

In the Bayesian approach, a GP is a generalisation of the Gaussian distribution to a function space of infinite dimension and it is created by placing a prior distribution over this function space. As such, the GP is specified by a mean function and a covariance function. "For inputs that have been evaluated with nearby points, the output is more certain than points that are further away from the points that have been evaluated". - THIS IS USEFUL, BUT NOT SPECIFIC ENOUGH.

Universal Kriging

NEED TO ADD/CHANGE HERE SECTION 3 IN A Tutorial on Bayesian Optimization FOCUSES ON THIS

The paper uses Gaussian processes, Kriging and Bayesian history matching. Kriging is an interpolation method. It makes use of a set of pre-evaluated points in the dataset, which can be quite small, to create a predictor function $\hat{f}$, which gives an estimate for the values of all the data points in the dataset. This enables us to estimate values for currently unevaluated data points in the dataset.

The type of Kriging used in the paper is Universal Kriging. This uses a polynomial $p(x)$, which is the expectation of the response variable $y(x)$, and then adds $M(x)$ which is a zero-mean stationary Gaussian Process to generate noise. Thus we have:

$$y(x) = p(x) + M(x)$$

The GP being zero-mean stationary in Kriging means that it has a mean of zero, a constant variance, and covariances that depend only on the distances between points.This is important because it affects what we get for the predictor function $\hat{y}$ which is determined by the three parameters:

The GP allows us to look at which functions could be the true $y(x)$, usually under some assumptions about $y(x)$ such as smoothness and continuity.

Bayesian History Matching

Once the objective function has been evaluated at eight points for the first round, Bayesian history matching speeds up the process by removing any points that are implausible. This is done by using Bayes' Theorem to update our beliefs about the value of every unevaluated point based on the values of the points we have evaluated so far. The general process is described below.

Let f be the objective function and x be a point in the sample space. We begin with some uncertainty about $f(x)$. However, we can make probabilistic statements such as:

$$P(f(x) > a) = \int_a^\infty P(f(x))df(x)$$

Once we evaluate another point x', we are able to improve our integral to:

$$P(f(x) > a|f(x')) = \int_a^\infty P(f(x)|f(x'))df(x)$$

We now let $a = max\{f(x_1), \dots, f(x_n)\}$ where n is the number of points we have evaluated so far. For the first round, $n = 8$ but as the rounds increase we make sure to include all previous points of the objective function that have been evaluated.

We remove the point x if:

$$P(f(x) > a|f(x_1), \dots, f(x_n)) < \varepsilon$$

We repeat this using $\varepsilon = 0.00001$ until only one plausible point remains as the optimum x^*. In our case, x^* is the F$_{target}$ and B$_{trigger}$ that will give the highest catch whilst following the precautionary principle.

COULD FIND OUT MORE ABOUT HOW WE FIND $P(f(x)|f(x'))$ BUT O/W LOOKS GOOD

Expected Improvement

The first type of acquisition function we will look at is Expected Improvement (EI). At iteration n, we sample the point x_n and observe the value $f(x_n)$. Then, if we were to return a solution at this point, bearing in mind we observe the objective function f without noise and we can only return points we have already evaluated, we would return $f_n^* = max\{f(x_1), \ldots, f(x_n)\}$. If we evaluate at another point x_{n+1} and observe $f(x_{n+1})$ this allows us to define the expected improvement as:

$$EI_n(x_{n+1}) := E_n[f(x_{n+1}) - f_n^*]^+$$

where $[f(x_{n+1}) - f_n^*]^+$ is the positive part of $[f(x_{n+1}) - f_n^*]$ and E_n indicates the expectation taken under the posterior distribution given evaluations of f at $x_1, \ldots, x_n$ so that we are using the previous points we have evaluated to help us find which new point is best to evaluate. This acquisition function is relatively easy to optimise and many different methods have been developed for doing this.

COULD EXPLAIN POSTERIOR DISTRIBUTION BUT OTHERWISE THIS LOOKS VERY GOOD.

Augmented Expected Improvement

NEED TO READ UP ON - NOT SURE I HAVE A SOURCE YET

Knowledge Gradient

NEED TO GET AND CONSIDER CHANGING. GET PROCESS DOWN BEFORE ADDING CODE. MAIN ISSUE HERE SEEMS TO BE CONSISTENCY OF NOTATION

A Tutorial on Bayesian Optimization HAS A SECTION ON THIS

We revise the assumption of EI that we have to return a pre-evaluated point as our best point. This allows us to do some different computations to the ones in EI. We also now start by saying that the solution we would choose if we have to stop sampling after n points would be the point in the design space with the largest $\mu_n(x)$ value, where μ_n is the value of the posterior mean. We call this maximum x_n^* and then can say that $f(x_n^*)$ has the conditional expected value:

$$\mu_n^* := \mu_n(x_n^*) = max_x \mu_n(x)$$

where x is any point in the sample space.

Then, we imagine that we are now allowed to sample a new point x_{n+1}. This changes the conditional expected value for $f(x_n^*)$ to be:

$$\mu^*_{n+1} := \mu_{n+1}(x^*_n) = max_x \mu_{n+1}(x)$$

?? SANITY CHECK $\mu_{n+1}(x^*_n)$ IN THE ABOVE

So, we can see that the increase in the conditional expected value of $f(x^*_n)$ by sampling the new point x_{n+1} is:

$$\mu^*_{n+1} - \mu^*_n$$

While this quantity is unknown before we sample x_{n+1} we can calculate it’s expected value given our observations $x_1, \ldots, x_n$. The Knowledge Gradient for sampling at a new point x is defined as:

$$KG_n(x) := E_n[\mu^*_{n+1} - \mu^*_n | x = x_{n+1}]$$

where again E_n indicates the expectation taken under the posterior distribution at the n^{th} iteration.

The easiest way to calculate the KG is via simulation. This can be done by simulating one possible value for $f(x_{n+1})$. Then, we calculate μ^*_{n+1} and subtract μ^*_n. We iterate this process many times so that we can find the average of $\mu^*_{n+1} - \mu^*_n$ and this allows us to estimate $KG_n(x)$.

However, we also have to include the update process in these iterations. This requires us to add in some extra code to make sure that the mean of the prior distribution, μ_n, is updated properly. Firstly, we examine the update formula:

$$\mu_{n+1}(x) = \mu_n(x) + \frac{\mathrm{cov}_n(x_{n+1}, x)}{\mathrm{var}_n(x_{n+1}) + \sigma^2_{\mathrm{obs}}}(F_{n+1} - \mu_n(x_{n+1})).$$

where F_{n+1} is the random distribution for $f(x_{n+1})$ and σ^2_{obs} is a noise variable which can be determined by the user.

This requires us to define the covariance matrix, find the standard deviation and to include the observation noise. Part of finding the covariance matrix involves summing over the number of inputs into the GP model for catch, denoted by d. Thus, we have a value θ which is a length d vector that weights each of these inputs appropriately.

In review, we now have all the tools to compute equation (4) - CHECK WHAT THIS IS IN ORIGINAL AS WILL BE A DIFFERENT NUMBER NOW. Then, we take the maximum of this as μ^*_{n+1} as before. By putting this all together, we now have the full method of calculating the $KG_n(x)$.