SCRIPT: BAYESIAN SIMULATION OPTIMIZATION WITH INPUT UNCERTAINTY

Juan Ungredda Juergen Branke

November 12, 2018

1 1th Slide: presentation slide

Thank you, so let's start with the Agenda

2 2nd Slide: Agenda

First we'll give some motivation for the project, followed by a brief background in order to explain the actual problem, finishing with the numerical experiments of the proposed algorithm and final conclusions.

3 3rd Slide: Motivation

When we deal with optimisation of complex systems we try to find the design variables that results in the best performance. Often we need to run simulations or physical experiments.
Examples of different applications are:

- Tuning of optimisation algorithms. Many optimisation algorithms have parameters that need to be tuned specific to the problem.
- scheduling and planning: a factory may be using dispatching rules for real-time scheduling, and the dispatching rules have some parameters whose optimal setting depends on shop floor conditions such as utilisation level or product mix.
- To capture the uncertainty present in the real world, these simulation models are usually stochastic. This is a particular challenge for optimisation tools, as evaluating the same solution multiple times yields different performance values
- Simulations are computationally expensive, thus the number of simulation runs that can be performed during optimisation is very limited.

So, in order to explain Input uncertainty let's see this example.
Hugh Grant runs a bakery that is open from 6am till 3pm.

- Customers arrive: Poisson process at rate Λ that varies from day to day in an i.i.d
- Λ: gamma distributed with parameters $\alpha > 0$ and $\beta > 0$
- Service time: exponentially distributed known mean μ^{-1}

so we can give answer to question related to the system such as this one where we try to allocate staff.
In contrast, Elton John despite of having the system, the poisson process is fixed but he doesn't know the parameter but can model the uncertainty.

4 5th Slide: Surrogate Model, Gaussian process

As we said before, usually simulations take a huge amount of resources such as time or money, so we are constraints on how many samples we can get.
The idea is building an auxiliary model that mimics the behaviour of the main model which will be used to make predictions.
One good surrogate model that is starting to be highly used because of its properties are Gaussian Processes
So, if you look at this line, this is one point in x. We say that the intersection of the function and this line is Gaussian. If this is true at...probably is true in the rest of the axis. Now let's take infinite many slices. You'll see something like this, where we can define a mean tendency and a constant variance.
Now, what happens if I know that the value at 4 and 6 are these two. So what can we say at places where I don't have points? Yes, if we go closer to the points we are more certain about our predictions but if we go far the model will be less certain.
So a Gaussian process can be seen as a collection of random variables that follows a normal distribution.

5 6th Slide: Adquisition Functions

Now, How can we sample in a iterative way in order to sample close where a minimum is.
Let's take one possible function that could pass through those points. The minimum should be around 6. Even intuitively one is temptep to choose a value closer to the second point.
if we start taking more sample functions we can get an histogram of the frequency where we find the minimum, which is somehow closer to our intuition.
So in the long run we have now a function we can easily maximise to find the design to sample next.

6 7th Slide: Problem Formulation

We have now some intuition about the tools. Now the problem at hand is the following.
Consider the possible designs $a \in A$ that we can choose from the simulation model and in the other hand the xes which represent the possible variables that the input that defines the simulation model could take
So, when running a simulation with design a and input x, we observe an output following an unknown noisy function $f(x, a)$ that we can decompose between a deterministic function defining and normally distributed and independent noise ϵ with constant variance.
If we knew the true value of the parameter x, say x^*, the best expected performance is given by the design a that approximates the true optimal.
but since we only have access to a distribution of where we believe is the true x, we identify the design a that maximises the expected performance simultaneously across the input parameter distribution.
As you can see, the design that maximises the function is different, so the idea is that we not only take samples to update the Gaussian process, but also take samples to update our beliefs about the input distribution.

$$F(a) = \int_X \theta(x, a)\mathbb{P}[x|D]dx \tag{1}$$

Since we are dealing with an expensive simulation model, we estimate $F(a)$ using the mean $\mu(x, a)$ of the learned Gaussian Process model.
where X_R are samples drawn from the posterior input distribution $\mathbb{P}[x|D]$.
We assume we have a

- Fixed budget of N samples we can take from the simulation model

- From the overall budget N, a part is destined to collect data samples to compute $\mathbb{P}[x|D]$ and the remaining budget is used to run more simulations.
- data samples to compute $\mathbb{P}[x|D]$ are i.i.d

So, the problem is when to take more data samples to update the input distribution. In the picture we see that a wide distribution might converge to a different design than narrower distributions.
So how do we incorporate the trade-off in the algorithm

7 8th Slide: Input Parameter Update

And we have a solution for the problem. So for a particular value x_0, we would choose a_r instead of $a = \arg\max_a \mu(x, a)$. The difference between both terms generate a non-negative loss of $\max_a \mu(x_0, a) - \mu(x_0, a_r)$. Thus, we can calculate an expected overall loss by integrating across all the input distribution.

$$Loss^m = \int_X (\max_a \mu(x, a) - \mu(x, a_r))\mathbb{P}[x|D]dx$$

Now, if we sampled additional data, say z^{m+1}, the posterior input distribution will change. So in order to give an expected Loss after sampling z^{m+1}, we calculate the expected Loss given all possible realisations by

$$Loss^{m+1} = \int_Z \int_X (\max_a \mu(x, a) - \mu(x, a'_r))\mathbb{P}[x|D, z^{m+1}]dx\mathbb{P}[z^{m+1}|D]dz^{m+1} \tag{2}$$

Finally, we may write the expected difference reduction as follows:

$$\Delta Loss = Loss^m - Loss^{m+1} \tag{3}$$

So during the sampling, we can decide to collect samples according to Knowleadge Gradient and each value will have some expected improvement that shows how much the Gaussian Process averaged over $P[x|D]$ will increase, or we can collect data to change the $P[x|D]$ which will narrow the distribution and the overall difference will be reduced. Therefore, we propose if there is an a that produces a bigger value of the Knowleadge Gradient function compared to the current value of ΔLoss we should collect a sample $(x^{n+1}, a^{n+1}, y^{n+1})$ and update the Gaussian Process, otherwise we collect a sample z^{m+1} and update the input distribution.

8 9th Slide: Numerical Experiments

The quality of the sampling procedure is determined by the Opportunity Cost, the difference in true performance between the design with the highest predicted value and the true best design, which is measured over the true input value x^*. So, if the value of a given alternative is calculated by $F(a) = \theta(x^*, a)$, the Opportunity Cost (OC) is given by

$$OC = \max_a F(a) - F(a_r) \tag{4}$$

Figure shows the average oportunity cost for the different numerical experiments.
We compare our algorithm against taking from a total budget of N a percentage p to sample and update the input distribution before the simulation optimisation. The remaining budget N(1-p) is used to train the Gaussian Process. We vary p as $p = 1, 3, 5, 10, 20, 50$ percent.

For the OC curves where a fixed proportion p was used, the increase of the percentage results in a decrease of the oportunity cost. Nonetheless, setting bigger initial portions Np also reduce the improvement in the OC curves convergence, mainly because the adquisition of new samples to update the input parameter distribution does not significantly change $\hat{F}(a)$
Compared to fixed proportions, the OC curve of the proposed method converges to similar values for $p = 20$ and 50% without having to set any parameter before the simulation optimisation process. On average, the proposed method takes between 30.70 and 31.84% of the total budget using a 95 $\%$ confidence interval. So, the algorithm is capable to take into account the trade-off between running additional simulations and collecting more data to reduce the input uncertainty.

9 10th Slide: Conclusions and Future Work

Numerical experiments showed that the new metric indeed reduce the input uncertainty and produce similar or better results compared to taking fixed portions of the overall budget.
Therefore, the algorithm is capable of evaluate between running additional simulations and reduce the input uncertainty in order to identify the design that has the best expected performance over the posterior distribution of input parameters. Also, the developed metric does not depend of parameters set by the user.
There are some interesting extensions of this work with concrete practical applications which are possible to pursue. While we assumed in this paper that the design space and the input distribution parameter space can each be described by a single continuous parameter, the proposed methods should also be tested in higher dimensions and with discrete parameters. Finally, one could consider worst case performance rather than expected performance.