Accelerating optimization via machine learning with different surrogate models

Ludovico Bessi

23 March 2019

Introduction

A surrogate model is an approximation method that mimics the behavior of a computationally expensive simulation.
The exact working of the simulation is not assumed to be known, solely the input-output behavior is important.
For this reason, a model is constructed based on the response of the simulator to a limited number of chosen data points.

In more mathematical terms: suppose we are attempting to optimize a function $f(p)$, but each calculation of f is very expensive. It may be the case we need to solve a PDE for each point or use advanced numerical linear algebra machinery which is usually costly. The idea is then to develop a surrogate model g which approximates f by training on previous data collected from evaluations of f.

There are many ways for building a surrogate model. Just to name a few, we can use: radial basis functions, neural networks, random forests, support vector machines and Gaussian processes. It is worth noting that the nature of the function is not known a priori so it usually is not clear *which* surrogate model will be the most accurate.

The construction of a surrogate model can be seen as a three steps process:

1 **Sample selection**

2 **Construction of the surrogate model and optimizing the parameters**

3 **Accuracy appraisal of the surrogate**

The accuracy of the surrogate depends on the number and location of samples in the design space.

Example

Let's consider the parametrized system of Lotka-Volterra equations:

$$\begin{cases} \frac{dx}{dt} = (A - By)x \\ \frac{dy}{dt} = (Cx - D)y \end{cases} \tag{1}$$

This system is made up of two non linear differential equations.
It can be solved numerically, but let's try to build the simplest possible surrogate model: a *linear model*.
Before diving into the overview of Julia implementation, it is worth noting that it cannot be expected that such a surrogate will work well, simply because it will try to model the system linearly, but (1) is non- linear.

Our surrogate should work like this: with $[A, B, C, D]$ as input, we want to be able to calculate the solution (x, y) at a time t^*.
We proceed as follows:

1 Solve the system in an interval (t_{min}, t_{max}) for n times with random input $[A, B, C, D]$.

2 Use the library *GLM* to build a linear model out of the samples.

3 Use model to find solution at time t^*.

Proposal

Timeline

6th May - 27th May – Bonding time

–

–

–

28th May - 28th June –

–

–

29th June - 26th July –

From 28th of June to approximately 3th of July I will be under exams, therefore my output will be reduced in this timeline.

–

–

27th July - 26th August –

I have left more time than necessary in this last month to account for problems that might have occurred along the way. I will be on vocation for a few days with my girlfriend as well.

–

– **Bonus:** Start developing surrogate models using Gaussian processes.

About me

I am a third year applied mathematics student at Politecnico di Torino, Italy. I know how to code in Python, C, R, Matlab and Julia. I have a strong background in **Probability theory** and **statistics**, thanks to a *measure theory* based course.

I am working as a Data scientist for "Policumbent", a team at my University that is building the fastest bike on earth.

I won an Hackaton with a machine learning project based on Keras. You can find my CV and more information about on my personal webite: https://ludoro.github.io

Contact information:

- Mobile phone: +39 3467172332
- Email: ludovicobessi@gmail.com
- GitHub: @ludoro
- Slack: @ludoro

Contact information in case of emergency: +39 360882130 (Father).