Motivation

Consider a black-box function $f(\vec{x})$ of the form

$$\begin{aligned} f \ &: \ \mathcal{S} \subseteq \mathbb{R}^D \to \mathbb{R} \\ \vec{x} \ &\mapsto \ f(\vec{x}) \end{aligned} \tag{1}$$

and assume that, while evaluating $f(\vec{x})$ is always possible,

1. f does not have an explicit, closed form expression; and,
2. f is expensive to compute.

If the goal is to seek $\vec{x}^* \in \mathcal{S}$ such that f is minimized,[1] then this motivates finding a surrogate function $\widehat{f}(\vec{x})$ such that

1. $\widehat{f}(\vec{x}) \cong f(\vec{x})$; and,
2. $\widehat{f}$ is cheap to compute.

This then allows one to perform surrogate optimization (i.e., using $\widehat{f}$ as the objective function within some optimization algorithm in order to approximate the optima of f).

Question

The construction of any surrogate model involves first sampling the problem space. That is, choose a set of $N > 0$ sample points $\{\vec{s}_i\}$, generate the corresponding image set $\{f(\vec{s}_i)\}$, and then train a surrogate model using this data set. This then begs the question: *what is the most efficient sampling scheme?*

Methodology

For the sake of scope control, this work is focussed on scalar functions f as stated above. The idea here is to compare two sampling schemes; namely

1. Simple random sampling (the "null sampling"); and,
2. Latin hypercube sampling (the "alternate sampling").

For a given benchmark problem (I'll just pick one, for the sake of scope.), the *efficiency* of a sampling scheme might be defined as something like

$$\eta_s = \frac{\text{surrogate utility}}{\text{surrogate cost}} = \exp\left[-\frac{N(\mu + \sigma)}{M}\right] \tag{2}$$

where $\mu \geq 0$ is some performance (error) mean, $\sigma \geq 0$ is some performance (error) standard deviation, and $M > 0$ is some normalizing factor (perhaps just $M = \max\{|f(\vec{s}_i)|\}$?). Observe that (2) is defined such that

[1] So $f(\vec{x}^*) \leq f(\vec{x})$ for all $\vec{x} \in \mathcal{S}$.

1. For any $\mu, \sigma, M > 0$, $\eta_s \to 0^+$ as $N \to \infty$. So, efficiency tends to zero for larger and larger sample sizes (the cost penalty).

2. For any $N, M > 0$, $\eta_s \to 1^-$ as $\mu + \sigma \to 0$. So, efficiency tends to one in the case of "perfect surrogate utility" (the utility reward).

What remains, then, is defining and computing μ and σ.

As for μ and σ, one might choose $\mu = \mu_{\text{RMSE}}$ and $\sigma = \sigma_{\text{RMSE}}$ (that is, work in terms of surrogate root mean squared error). One way forward in this regard is a training/test split approach; namely

1. Randomly partition the surrogate data set into two sub-sets: training and test. (Perhaps an 80%/20% split here.)

2. Train the surrogate on the training set (e.g., least squares).

3. Assess the surrogate on the test set (compute RMSE).

4. Repeat 1 - 3 a sufficient number of times (say, 1000) to generate a population of RMSE values. (So essentially *bootstrapping*.)

5. Compute μ_{RMSE} and σ_{RMSE} from the population generated in 4.

Steps 1 - 5 would allow one to generate a single (N, η_s) pair under some choice of dimensionality D and sampling scheme. *One of the key deliverables here would be η_s vs N curves for various D values, one plot for each sampling scheme (or maybe a single plot with dashed curves for random sampling and solid curves for LHS). I expect the "curse of dimensionality" will present in the plot(s).*

Taking this one step further, if I were to vary the benchmark problem as well, then I could gain some insight into whether or not the No Free Lunch Theorem also holds with respect to sampling scheme (i.e., there is no overall "best scheme"). I suspect that it does not, actually, but that remains to be seen.