# Cross-validation based adaptive sampling for Gaussian process models

Peter Challenor, Hossein Mohammadi

University of Exeter, UK

Mascot-Num

28 April 2021

## Outline

## Outline

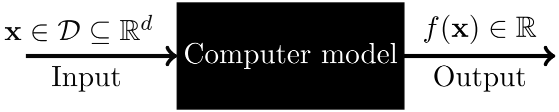
## Decision making with models/simulators

- Let $f(\mathbf{x})$ represent the output of a complex computer model (or simulator).
- We want to use this model to make some real world decision.
- For that we need to produce calibrated predictions with the uncertainties quantified.
- One step on the way to such simulator predictions is build a fast surrogate simulator with known uncertainty.
- We refer to this as an *emulator*.

## Problem Statement

We wish to approximate $f$ which is

- Black-box (analytically unknown)
- Computationally expensive $\rightarrow$ a parsimonious model run strategy is required



$\mathbf{x} \in \mathcal{D} \subseteq \mathbb{R}^d$
Input
Computer model
$f(\mathbf{x}) \in \mathbb{R}$
Output

## Surrogate modelling

One possible way around this problem is to use surrogate models, e.g. Gaussian process (GP) emulators.

- Surrogate models provide a “fast” approximation of the true model
- They are buildt using a limited number of simulation runs

## Experimental design

- Simulator runs are expensive
- The location of samples has a significant effect on the accuracy of a surrogate model
- We want the ‘best’ emulator for the minimum number of runs
- The design of computer experiments (DoE) has become an integral part of the analysis of computer experiments
- Samples can be chosen at once (one-shot) or sequentially (adaptive)

## One-shot sampling

- Full factorial
- Latin hypercube sampling (LHS)
- Minimax and maximin-distance
- Low discrepancy sequences
- . . .

## One-shot sampling

- Full factorial
- Latin hypercube sampling (LHS)
- Minimax and maximin-distance
- Low discrepancy sequences
- ...

Drawback: may result in under/oversampling and wasting computational resources ☹

## Adaptive sampling algorithm

1. Fit a surrogate model to an initial DoE
2. While stopping criterion not met:
    - (a) Find a new point using an "appropriate" criterion
    - (b) Run the simulator at that point
    - (c) Add the new point to the DoE
    - (d) Update the surrogate model using the new point

## Sequential Design

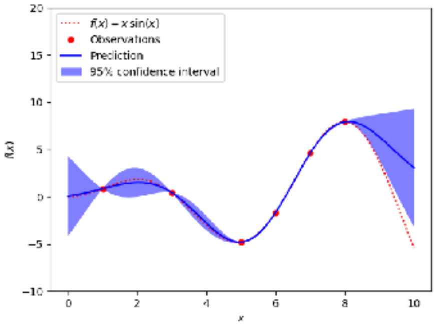
- An obvious sequential design is to add new design points where the uncertainty of our current emulator is a maximum.
- Or to minimise the integrated uncertainty
- Tends to add points at the boundaries
- Poor at space filling

## The Perfect Sequential Design

- Start with a ‘small’ space filling design
- Will not put points on the boundaries (fill space)
- Will reduce the integrated uncertainty (or other metric) rapidly
- Can be used in a ‘batch symmetric’ way

In our new algorithm:

- GP emulators serve as the surrogate model
- A new sample is selected based on a Leave-One-Out cross validation criterion



## Outline

## Gaussian processes

A GP is denoted by $Z_0(\mathbf{x})$ and is fully determined by its mean ($m_0$) and covariance function/kernel ($k_0$).

- $m_0(\mathbf{x}) = \mathbb{E}\left[Z_0(\mathbf{x})\right]$
- $k_0(\mathbf{x}, \mathbf{x}') = \mathbb{C}\mathrm{ov}\left(Z_0(\mathbf{x}), Z_0(\mathbf{x}')\right)$

Note: The mean function is assumed (wlog) to be a constant: $m_0(\mathbf{x}) = \mu$.

## GP prediction

The GP prediction is obtained by conditioning it on the observations: $Z_n(\mathbf{x}) = Z_0(\mathbf{x}) \mid \mathbf{y}_n$ where

- $\mathbf{X}_n = (\mathbf{x}_1, \dots, \mathbf{x}_n)^\top \rightarrow n$ locations in the input space
- $\mathbf{y}_n = (f(\mathbf{x}_1), \dots, f(\mathbf{x}_n))^\top \rightarrow$ corresponding outputs
- $\mathcal{A} = \{\mathbf{X}_n, \mathbf{y}_n\} \rightarrow$ training set (our initial experimental design)

## GP predictive mean & variance

The predictive (conditional) mean and variance are:

- $m_n(\mathbf{x}) = \mathbb{E}\left[Z_n(\mathbf{x})\right] = \mu + \mathbf{k}(\mathbf{x})^\top \mathbf{K}^{-1}(\mathbf{y}_n - \mu\mathbf{1})$
- $s_n^2(\mathbf{x}) = \mathbb{V}\mathrm{ar}\left(Z_n(\mathbf{x})\right) = k_0(\mathbf{x}, \mathbf{x}) - \mathbf{k}(\mathbf{x})^\top \mathbf{K}^{-1}\mathbf{k}(\mathbf{x})$

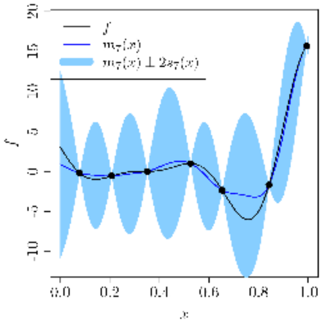
where

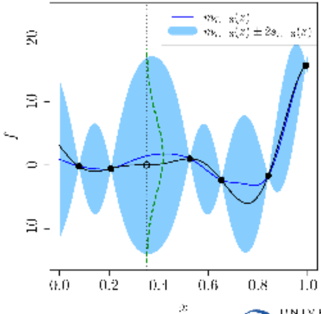
- $\mathbf{k}(\mathbf{x}) = (k_0(\mathbf{x}, \mathbf{x}_1), \ldots, k_0(\mathbf{x}, \mathbf{x}_n))^\top$
- $\mathbf{K} \rightarrow$ covariance matrix; $\mathbf{K}_{ij} = k_0(\mathbf{x}_i, \mathbf{x}_j)$ for $1 \leq i, j \leq n$
- $\mathbf{1} \rightarrow$ vector of ones

## Outline

# Leave-One-Out (LOO) error

1. Remove the $i$th data point from the training set
2. Fit the emulator to the remaining points, $Z_{n,-i}(\mathbf{x})$
3. Obtain the prediction at $\mathbf{x}_i$, $m_{n,-i}(\mathbf{x}_i)$
4. $e_L(\mathbf{x}_i) = |m_{n,-i}(\mathbf{x}_i) - f(\mathbf{x}_i)| \rightarrow$ LOO cross-validation error





- The LOO error measures the sensitivity of the emulator to the left out point
- The next sample should be selected where the LOO error is maximum
- However, this measure can be misleading

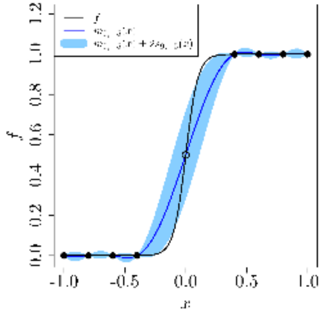


**Figure:** Removing the 5th sample does not change the prediction at $x = 0$ and $e_L(x_5) = 0$.

## Expected squared LOO error

The expected squared LOO (ES-LOO) error at $\mathbf{x}_i$ is defined as[1]

$$\mathcal{E}_L(\mathbf{x}_i) = \frac{\mathbb{E}\left[\left(Z_{n,-i}(\mathbf{x}_i) - f(\mathbf{x}_i)\right)^2\right]}{\sqrt{\mathbb{V}\text{ar}\left(\left(Z_{n,-i}(\mathbf{x}_i) - f(\mathbf{x}_i)\right)^2\right)}}.$$

- $Z_{n,-i}(\mathbf{x}_i) \sim \mathcal{N}\left(m_{n,-i}(\mathbf{x}_i), s^2_{n,-i}(\mathbf{x}_i)\right)$
- $\mathbb{E}\left[\left(Z_{n,-i}(\mathbf{x}_i) - f(\mathbf{x}_i)\right)^2\right] = s^2_{n,-i}(\mathbf{x}_i) + \left(m_{n,-i}(\mathbf{x}_i) - f(\mathbf{x}_i)\right)^2$
- $\mathbb{V}\text{ar}\left(\left(Z_{n,-i}(\mathbf{x}_i) - f(\mathbf{x}_i)\right)^2\right) =$
  $2s^4_{n,-i}(\mathbf{x}_i) + 4s^2_{n,-i}(\mathbf{x}_i)\left(m_{n,-i}(\mathbf{x}_i) - f(\mathbf{x}_i)\right)^2$

[1] [Mohammadi et al. (2021)]

## ES-LOO error (out-of-sample)

- Next sample → where ES-LOO error is maximum
- ES-LOO errors → only defined at design points ☹

We extend the ES-LOO error to be a function defined over the whole domain, $\mathcal{E}_L(\mathbf{x}), \mathbf{x} \in \mathcal{D}$, that we have observed at the design points, and we model this function with a GP. The interpretation of $\mathcal{E}_L(\mathbf{x})$ is the value of the ES-LOO error we think we would see if $(\mathbf{x}, f(\mathbf{x}))$ were part of our data set.

## Where to sample next?

- $Z_n^e(\mathbf{x}) \rightarrow$ GP model to predict $\mathcal{E}_L(\mathbf{x}), \mathbf{x} \in \mathcal{D}$
- $\{\mathbf{X}_n, \mathbf{y}_n^e\} \rightarrow$ data set to train $Z_n^e(\mathbf{x})$ where $\mathbf{y}_n^e = (\mathcal{E}_L(\mathbf{x}_1), \ldots, \mathcal{E}_L(\mathbf{x}_n))^\top$
- To find $\max\left(\mathcal{E}_L(\mathbf{x})\right) \rightarrow$ trade-off between exploration and exploitation is required
- *Expected improvement* (EI) $\rightarrow$ can be used to achieve the exploration/exploitation trade-off
- Next sample $\rightarrow \mathbf{x}_{n+1} = \underset{\mathbf{x} \in \mathcal{D}}{\operatorname{argmax}} \; EI(\mathbf{x})$

## Expected improvement

$$EI(\mathbf{x}) = \begin{cases} (m_n^e(\mathbf{x}) - \max(\mathbf{y}_n^e))\,\Phi(u) + s_n^e(\mathbf{x})\phi(u) & \text{if} \quad s_n^e(\mathbf{x}) > 0 \\ 0 & \text{if} \quad s_n^e(\mathbf{x}) = 0 \end{cases}$$

- $u = \frac{m_n^e(\mathbf{x}) - \max(\mathbf{y}_n^e)}{s_n^e(\mathbf{x})}$
- $m_n^e(\mathbf{x})$ and $s_n^e(\mathbf{x}) \rightarrow$ predictive mean and variance of $Z_n^e(\mathbf{x})$
- $\phi(\cdot)$ and $\Phi(\cdot) \rightarrow$ PDF and CDF of the standard normal distribution

EI tends toward exploitation and using it to select the next design point results in clustering.

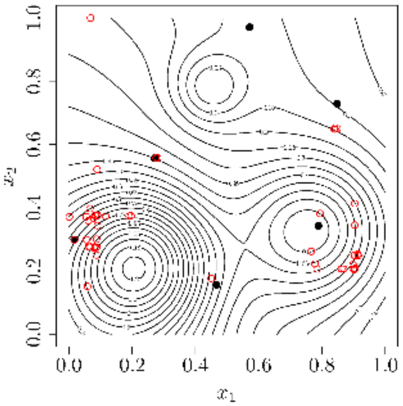


**Figure:** Initial design (filled circles) and adaptive designs (red circles) obtained by EI as the selection criterion.

## Pseudo expected improvement (PEI)

To avoid clustering, we use *pseudo expected improvement* (PEI):

$$\mathbf{x}_{n+1} = \underset{\mathbf{x}\in\mathcal{D}}{\operatorname{argmax}}\ PEI(\mathbf{x}).$$

PEI is obtained by multiplying EI by a repulsion function (RF) defined as:

$$RF(\mathbf{x};\mathbf{X}_n) = \prod_{i=1}^{n} \left[1 - \mathbb{C}\mathrm{orr}\left(Z_n^e(\mathbf{x}), Z_n^e(\mathbf{x}_i)\right)\right],$$

which is a measure of the (canonical) distance between $\mathbf{x}$ and the data points $\mathbf{X}_n$.

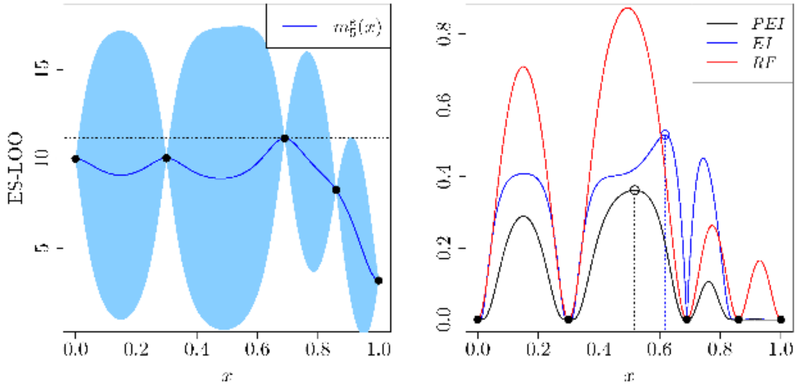


**Figure:** Left: A GP is fitted to ES-LOO errors. Right: Pseudo expected improvement (black), expected improvement (blue) and repulsion function (red). PEI is more explorative than EI.

## Pseudo points

Often, in adaptive sampling strategies many points lie along the boundaries of the input space. This problem can be alleviated in our approach by introducing "pseudo points" at appropriate locations on the boundaries.
Pseudo points, denoted by $\mathbf{X}_p$, are used to update the repulsion function, i.e. $RF(\mathbf{x}; \mathbf{X}_n \cup \mathbf{X}_p)$, but the true function is not evaluated there.

## Pseudo point locations

1. Corners of the input region
2. Closest point on each face of the (rectangular) bounding region to the initial design

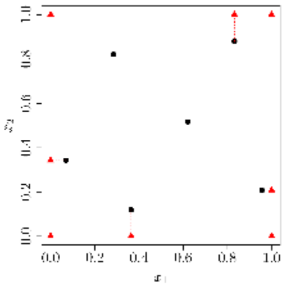


**Figure:** Initial DoE (black points) and pseudo points (red triangles).

## Batch sampling

- A set of inputs $\{\mathbf{x}_{n+1}, \ldots, \mathbf{x}_{n+q}\}$ is evaluated at each iteration
- Can save the user time when parallel computing is available
- Our approach can be easily extended to a batch mode thanks to the repulsion function
- Only the repulsion function is updated, see next slides

## Batch mode example ($q = 3$)

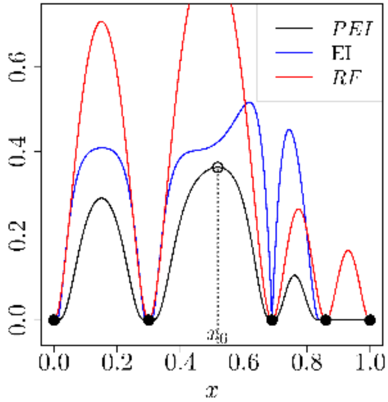


**Figure:** Five initial DoE and $x_6$ is selected where PEI is maximum.

## Batch mode example ($q = 3$)

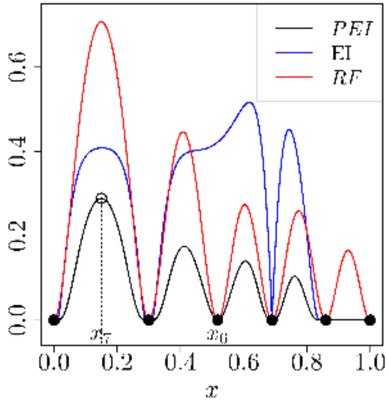


**Figure:** RF is updated by $x_6$. $x_7$ is selected where PEI is maximum.

## Batch mode example ($q = 3$)

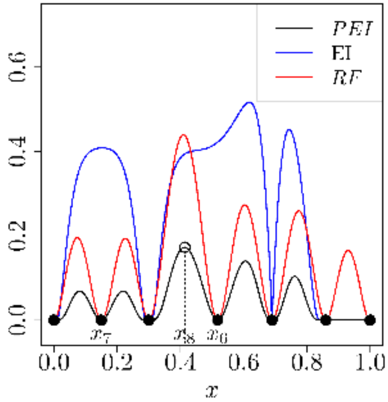


**Figure:** RF is updated by $x_7$. $x_8$ is selected where PEI is maximum.

## Outline

We compare our proposed approach with:

- Maximum mean squared error (MSE)
- Expected improvement for global fit (EIGF) [Lam (2008)]
- Mutual information for computer experiments (MICE) [Beck and Guillas (2016)]
- LHS design

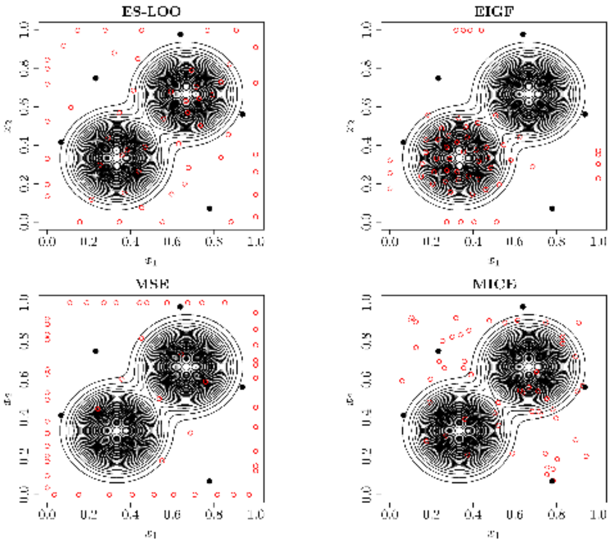
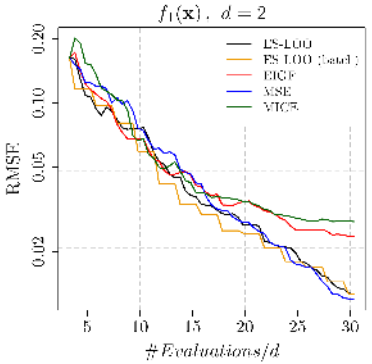


**Figure:** Typical behaviour of four adaptive sampling methods. Initial design (filled circles) and adaptive designs (red circles).

## Problem setting

- Six test functions: $f_1, \ldots, f_6$
- Problem dimension: $d = 2, 3, 5, 6, 7$
- Total budget: $30 \times d$
- Size of initial DoE: $3 \times d$ (using LHS)
- Assessment tool: root mean squared error (RMSE)
- Test points: 3000 (space-filling) points

## Results



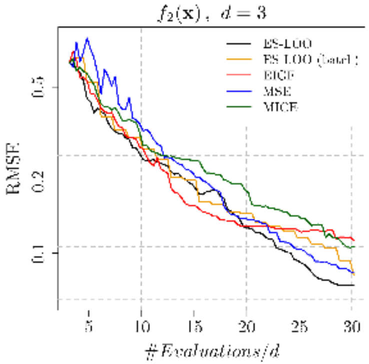
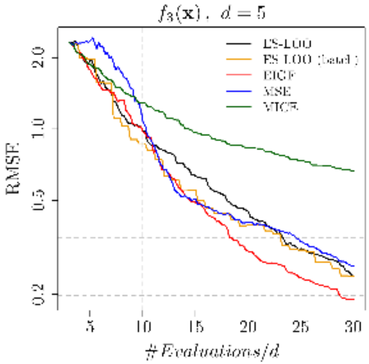


**Figure:** The median of RMSEs using 10 different initial DoEs. Franke's (left) and Hartman functions (right).

## Results



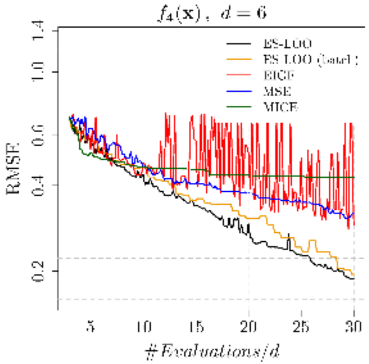
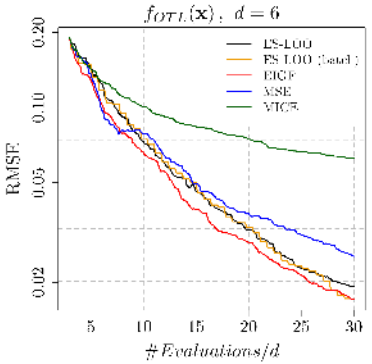


**Figure:** The median of RMSEs using 10 different initial DoEs. Friedman (left) and Gramacy & Lee functions (right).

## Results



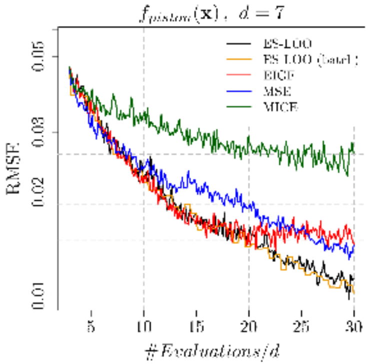


**Figure:** The median of RMSEs using 10 different initial DoEs. The OTL circuit (left) and piston simulation (right) functions.

## Conclusions

- Sequential design can offer a number of improvements over one shot designs
- We are proposing a new method based on leave one out and pseudo-expected improvement
- Our method is consistently among the best in comparisons.

Thank you for your attention!

## References

Beck, J. and Guillas, S. (2016). Sequential design with mutual information for computer experiments (MICE): Emulation of a tsunami model. *SIAM/ASA Journal on Uncertainty Quantification*, 4(1):739–766.

Lam, C. Q. (2008). *Sequential adaptive designs in computer experiments for response surface model fit*. PhD thesis, Columbus, OH, USA. AAI3321369.

Mohammadi, H., Challenor, P., Williamson, D., and Goodfellow, M. (2021). Cross-validation based adaptive sampling for Gaussian process models.