AEROSP 588 - Project

EFFICIENT SEQUENTIAL SAMPLING FOR NEURAL NETWORK-BASED SURROGATE MODELING

Adhithiaram Hariharan

Abstract

Surrogate modeling [4] is one of the effective ways for global optimization of expensive black-box functions. This project checks the accuracy and improvements of using a Neural Network (NN) based surrogate model for Efficient Global Optimization (EGO) when compared to the standard Gaussian Process Regression (GPR) method. According to the paper [3], the Efficient Global Optimization using Neural Networks (EGONN) yields comparable results as EGO with the same number of initial and infill sample budget. The project is undertaken to recreate the results from this paper.

Background and Motivation

We know that many model evaluations are required for performing design exploration, aero-structural optimization, uncertainty quantification, and various other design activities. This process costs a lot of computational time. Therefore, we employ surrogate modeling techniques to produce an approximate simulation response as a function of input variables close to the actual result.

To create a surrogate model for this project, we use the sequential sampling method, which uses a part of the computational cost for creating an initial surrogate model. Then, the remaining cost is used to add new samples to the data based on an infill criterion. This method can improve the surrogate model since the new points are added at locations where the surrogate model has poor accuracy or in regions of interest.

To use the infill criteria, we need to calculate the uncertainty and prediction at all the points of design space. A widely used technique for sequential sampling is efficient global optimization (EGO). Gaussian Process Regression (GPR) is a standard method for EGO because it can provide uncertainty estimates in the prediction. The cost of creating a GPR model for large data sets is high. So, we will use neural network (NN) models to scale better than GPR as the number of samples increases. However, the uncertainty estimates for NN prediction are not available.

Hence, a scalable algorithm is developed for EGO using NN-based prediction and uncertainty (EGONN). Initially, two different NNs are created using two different data sets. The first NN models the output based on the input values in the first data set, while the second NN models the prediction error of the first NN using the second data set. The next infill point is added to the first data set based on expected improvement criteria and prediction uncertainty.

Problem Formulation

To validate this surrogate modeling method, we will test it by optimizing the one-dimensional Forrester function and the two-dimensional constrained Branin function.

CASE 1: Forrester function:

The underlying optimization task is to find the global minimum of Forrester function $f(x)$ over a bounded domain, specifically $x \in [0, 1]$. The function is defined as:

$$f(x) = \left((6x - 2)^2\right) \sin(12x - 4),$$

which is a nonlinear, multimodal function. The goal is to identify the point $x^* \in [0, 1]$ that minimizes $f(x)$:

$$x^* = \arg \min_{x \in [0,1]} f(x).$$

This optimization problem can be described as continuous, constrained, nonlinear, multimodal, nonconvex, noise-free, stochastic optimization problem.

CASE 2: Branin Function

The second case under consideration is a two-dimensional constrained optimization problem. The objective is to find the global minimum of a nonlinear function $y(x_1, x_2)$ over a specified rectangular domain, while satisfying an inequality constraint. The problem can be formulated as follows:

$$\min_{x_1, x_2} \; y(x_1, x_2) \quad \text{subject to} \quad g(x_1, x_2) \leq 0,$$

where

$$y(x_1, x_2) = \left(x_2 - \frac{5.1}{4\pi^2} x_1^2 + \frac{5}{\pi} x_1 - 6\right)^2 + 10 \left(1 - \frac{1}{8\pi}\right) \cos(x_1) + 10,$$

and

$$g(x_1, x_2) = 1 - \frac{x_1 x_2}{x_1^* x_2^*}.$$

The design variables are constrained to: $x_1 \in [-5, 10]$, $x_2 \in [0, 15]$. The objective function $y(x_1, x_2)$ is nonconvex and nonlinear. The presence of the inequality constraint $g(x_1, x_2) \leq 0$ restricts the feasible region of the design space, adding complexity to the search for the global optimum. The known global minimum is located at: $(x_1^*, x_2^*) = (3\pi, 2.475)$, with

$$g(x_1^*, x_2^*) = 1 - (3\pi)(2.475) = 0,$$

satisfying the constraint exactly at the optimal point. Because this is a constrained optimization problem, a penalty function is introduced to transform the problem into an unconstrained one. Here, using a penalty parameter $\lambda = 100$, the new objective function is defined as:

$$H(x_1, x_2) = y(x_1, x_2) + \lambda \Delta g(x_1, x_2),$$

This modification ensures that any violation of the constraint significantly increases the objective value, thus penalizing infeasible solutions.

So this is a continuous, nonlinear, constrained, nonconvex, noise-free optimization problem. The complex objective function and the nonlinear inequality constraint make it a difficult problem for standard optimization techniques, and it is often solved using surrogate-based optimization or global optimization algorithms.

GPR based Optimization Algorithm

Consider a black-box function $f : \mathbb{R}^n \to \mathbb{R}$ defined over a domain $\mathcal{D} \subseteq \mathbb{R}^n$. The goal is to find a global minimum of $f(\mathbf{x})$ without relying on gradient information. The Efficient Global Optimization (EGO) algorithm provides a framework for solving this problem by iteratively refining a surrogate model of the objective function using Gaussian Process Regression (GPR) [1].

EGO begins with an initial dataset $(\mathbf{X}, \mathbf{Y})$, where $\mathbf{X} = [\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(p)}]^T$ are sampled points in $\mathcal{D}$ and $\mathbf{Y} = [y(\mathbf{x}^{(1)}), y(\mathbf{x}^{(2)}), \ldots, y(\mathbf{x}^{(p)})]^T$ are the corresponding function evaluations with $y(\mathbf{x}^{(i)}) = f(\mathbf{x}^{(i)})$. A Gaussian Process (GP) prior is placed on $f(\mathbf{x})$:

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')),$$

where $m(\mathbf{x})$ is the mean function and $k(\mathbf{x}, \mathbf{x}')$ is the covariance (kernel) function. For this project, the squared exponential kernel was used:

$$k(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left(-\sum_{j=1}^{n} \theta_j (x_j - x_j')^2 \right),$$

where σ_f^2 is the variance and $\theta_j > 0$ are length-scale parameter. These hyperparameters are typically estimated by maximizing the marginal likelihood of the observed data. The negative log-likelihood function is optimized in the inner loop of the GPR class through Scipy's minimize function with the L-BFGS optimizer.

The squared exponential kernel is chosen due to its smoothness and differentiability properties, making it easier to model a variety of functions. Once the kernel hyperparameters are fitted, the GPR model provides a posterior predictive distribution for any new point $\mathbf{x}_*$. Given the observed data $(\mathbf{X}, \mathbf{Y})$ and $\mathbf{m} = [m(\mathbf{x}^{(1)}), \ldots, m(\mathbf{x}^{(p)})]^T$, define:

$$K \quad \text{with} \quad K_{ij} = k(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}),$$

and

$$\mathbf{r}_* = [k(\mathbf{x}_*, \mathbf{x}^{(1)}), \ldots, k(\mathbf{x}_*, \mathbf{x}^{(p)})]^T.$$

The posterior mean and variance at $\mathbf{x}_*$ are given by:

$$\hat{y}(\mathbf{x}_*) = m(\mathbf{x}_*) + \mathbf{r}_*^T K^{-1} (\mathbf{Y} - \mathbf{m}),$$

$$s^2(\mathbf{x}_*) = k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{r}_*^T K^{-1} \mathbf{r}_*.$$

Thus, GPR provides not only a surrogate prediction $\hat{y}(\mathbf{x}_*)$ but also an associated uncertainty $s(\mathbf{x}_*) = \sqrt{s^2(\mathbf{x}_*)}$[2]. The ability to quantify uncertainty in the predictions makes GPR an appealing surrogate model for global optimization, as it guides the EGO algorithm to explore promising domain regions.

With the GPR model in place, the EGO algorithm iteratively updates the surrogate model and selects new points to evaluate based on a suitable acquisition function. The key idea is to balance the exploitation of the current best solution and the exploration of uncertain regions.

The EGO loop:

1. Fit a GPR model to the current data $(\mathbf{X}, \mathbf{Y})$.
2. Select the next infill point by maximizing an acquisition function (Expected Improvement).
3. Evaluate the true function $f(\mathbf{x}_{\text{next}})$, augment $(\mathbf{X}, \mathbf{Y})$, and update the GPR model.
4. Repeat until a stopping criterion is met (low uncertainty or maximum number of iterations).

EGO employs the Expected Improvement (EI) criterion to select the next evaluation point. Let:

$$f_{\text{best}} = \min\{y(\mathbf{x}^{(1)}), \ldots, y(\mathbf{x}^{(p)})\}$$

be the best (lowest) objective value observed so far. At a candidate point $\mathbf{x}$, the predictive distribution of the unknown function value $F(\mathbf{x})$ under the GPR model is normal with mean $\hat{y}(\mathbf{x})$ and variance $s^2(\mathbf{x})$. The EI is defined as:

$$\text{EI}(\mathbf{x}) = \mathbb{E}\left[\max(f_{\text{best}} - F(\mathbf{x}), 0)\right].$$

With $Z = \frac{f_{\text{best}} - \hat{y}(\mathbf{x})}{s(\mathbf{x})}$, the EI can be expressed in closed form:

$$\text{EI}(\mathbf{x}) = (f_{\text{best}} - \hat{y}(\mathbf{x}))\Phi(Z) + s(\mathbf{x})\phi(Z),$$

where $\Phi(\cdot)$ and $\phi(\cdot)$ are the cumulative distribution and probability density functions of the standard normal distribution, respectively.

Maximizing EI identifies points that are likely to improve upon the best solution or are sufficiently uncertain to avoid exploration. As iterations proceed, the algorithm reduces uncertainty and refines its search toward the global minimum.

EGONN based Optimization Algorithm

The proposed EGONN algorithm follows the main steps of the GPR algorithm but employs two deep neural networks (DNNs) in place of the Gaussian process regression (GPR) surrogate model. One DNN models the function output with respect to the input parameters, and the other DNN models the uncertainty in the function output prediction.

This is realized by having two initial data sets, one for the function prediction model and another for the uncertainty model. Let $(X, Y)_u$ be the data set used for the uncertainty model. Here, $X_u = \{x^{(1)}, \ldots, x^{(q)}\}$ is the set of input sample points and $Y_u = (y^{(1)}(x^{(1)}), \ldots, y^{(q)}(x^{(q)}))$ are the corresponding function outputs.

The data sets (X, Y) and $(X, Y)_u$ are distinct, and both are created using Latin hypercube sampling (LHS). Differential evolution is employed to maximize the expected improvement (EI). Other steps of the EGONN algorithm remain the same as in EGO.

In each cycle of the EGONN algorithm, a DNN is fitted to the current data set (X, Y) to construct a surrogate prediction model $\hat{y}$ of the function output y. DNNs are universal function approximators that consist of multiple layers. Hidden layers lie between the input and output layers, and each hidden layer contains neurons equipped with nonlinear activation functions. The DNN parameters are tuned by solving an unconstrained optimization problem, using a gradient-based optimizer.

Algorithm 1 Proposed EGO algorithm with DNN-based prediction and uncertainty (EGONN)

Require: Initial data sets (X, Y) and $(X, Y)_u$
  repeat
    Fit a DNN to (X, Y) to construct a surrogate prediction model $\hat{y}$.
    Compute the prediction uncertainty using the data $(X, Y)_u$.
    Fit another DNN to (X_u, u^2), where

$$u^2(x) = [\hat{y}(x) - y(x)]^2,$$

    to obtain the spatial error model $\hat{u}^2(x)$.
    Evaluate the uncertainty $s(x) = \sqrt{\hat{u}^2(x)}$.
    Compute the expected improvement:

$$\mathsf{EI}(x) = (f^* - \hat{y}(x))\Phi\left(\frac{f^* - \hat{y}(x)}{s(x)}\right) + s(x)\phi\left(\frac{f^* - \hat{y}(x)}{s(x)}\right),$$

    where $f^* = \min(Y)$, and $\Phi(\cdot)$ and $\phi(\cdot)$ are the CDF and PDF of the standard normal distribution, respectively.
    Find the new sample point $P = \arg\max_x \mathsf{EI}(x)$ using differential evolution.
    Update the data set $(X, Y) \leftarrow (X \cup P, Y \cup y(P))$.
  until convergence
  $y^* \leftarrow \min(Y)$
  $x^* \leftarrow \arg\min_{x \in X} y(x)$
  return (x^*, y^*)

In this work, the adaptive moments (ADAM) algorithm is employed alongside the backpropagation algorithm to compute gradients. The mean squared error (MSE) is used as the loss function:

$$L = \frac{1}{p}\sum_{l=1}^{p}\left(\hat{y}^{(l)} - y^{(l)}\right)^2,$$

where p is the number of training samples. The learning rate is set to 0.001. The number of hidden layers, the number of neurons per hidden layer, and other DNN hyperparameters are problem-dependent. In this work, the DNN implementation is carried out using TensorFlow.

The EGONN algorithm computes the spatial error in each cycle as: $u^2(x) = [\hat{y}(x) - y(x)]^2$. A second DNN is trained to model the spatial error using (X_u, u^2), resulting in the spatial error model $\hat{u}^2(x)$. The prediction uncertainty of the DNN-based model $\hat{y}$ at the sample x is then taken as: $s(x) = \sqrt{\hat{u}^2(x)}$.

The DNN-based $\hat{y}$ and $s(x)$ models are used to compute the expected improvement $\mathsf{EI}(x)$, and a new sample is obtained by maximizing the EI. Note that the DNN-based approach does not assume the underlying function or uncertainty to be Gaussian, unlike GPR-based methods. In spite of this, the same EI formulation is retained. The new sample is appended only to (X, Y), while $(X, Y)_u$ remains fixed. The termination criterion for the EGONN algorithm is the same as for EGO.

Code Structure

Algorithm 2 EGO and EGONN Optimization Framework

Input: Black-box function f, domain, initial sets $\{X_{\text{init}}, Y_{\text{init}}\}$, max iterations k_{max}, uncertainty training points X_{uncert}, true values $Y_{\text{uncert\_true}}$, tolerance ϵ.

Initialization:
Set $(X_{\text{EGO}}, Y_{\text{EGO}}) := (X_{\text{init}}, Y_{\text{init}})$.
Set $(X_{\text{EGONN}}, Y_{\text{EGONN}}) := (X_{\text{init}}, Y_{\text{init}})$.
for $k = 1$ to k_{max} **do**
  EGO Step:
  Fit GP to $(X_{\text{EGO}}, Y_{\text{EGO}})$ to get parameters $(\theta, \beta, \sigma^2)$.
  Compute $f_{\text{min}} = \min(Y_{\text{EGO}})$.
  Define EI using GP predictions and standard deviations.
  Find $x_{\text{next}} := \arg\max \text{EI}(x)$ over $[0, 1]$.
  Evaluate $f(x_{\text{next}})$ and update $(X_{\text{EGO}}, Y_{\text{EGO}})$.

  EGONN Step:
  Train a DNN prediction model on $(X_{\text{EGONN}}, Y_{\text{EGONN}})$.
  Use the DNN to predict $Y_{\text{uncert\_pred}} := \hat{f}(X_{\text{uncert}})$.
  Compute $U2 = (Y_{\text{uncert\_true}} - Y_{\text{uncert\_pred}})^2$.
  Train a DNN uncertainty model on $(X_{\text{uncert}}, U2)$.
  Compute $f_{\text{min}} = \min(Y_{\text{EGONN}})$.
  Define EGONN EI using DNN predictions and uncertainty estimates.
  Find $x_{\text{next}}^{\text{EGONN}} := \arg\max \text{EGONN EI}(x)$ over $[0, 1]$.
  Evaluate $f(x_{\text{next}}^{\text{EGONN}})$ and update $(X_{\text{EGONN}}, Y_{\text{EGONN}})$.

  Check stopping criteria (e.g., $k = k_{\text{max}}$, improvement $< \epsilon$).
  if stopping criteria met **then**
    Stop.
  end if
end for

Output:
$x_{\text{best,EGO}} = \arg\min Y_{\text{EGO}}, \quad f_{\text{best,EGO}} = \min(Y_{\text{EGO}})$.
$x_{\text{best,EGONN}} = \arg\min Y_{\text{EGONN}}, \quad f_{\text{best,EGONN}} = \min(Y_{\text{EGONN}})$.
Convergence history, surrogate models, and final approximations.

Optimization Results

CASE 1: Forrester function:

The surrogate modeling setup for the EGONN algorithm is the same as the setup in the paper [3]. Four uniformly distributed initial data samples are used with an infill budget of eight samples. Ten uniformly distributed data samples are used for the DNN-based prediction uncertainty model. Both DNNs have three hidden layers with 50 neurons. The hyperbolic tangent function is used for activation. The number of epochs is set to 3000.

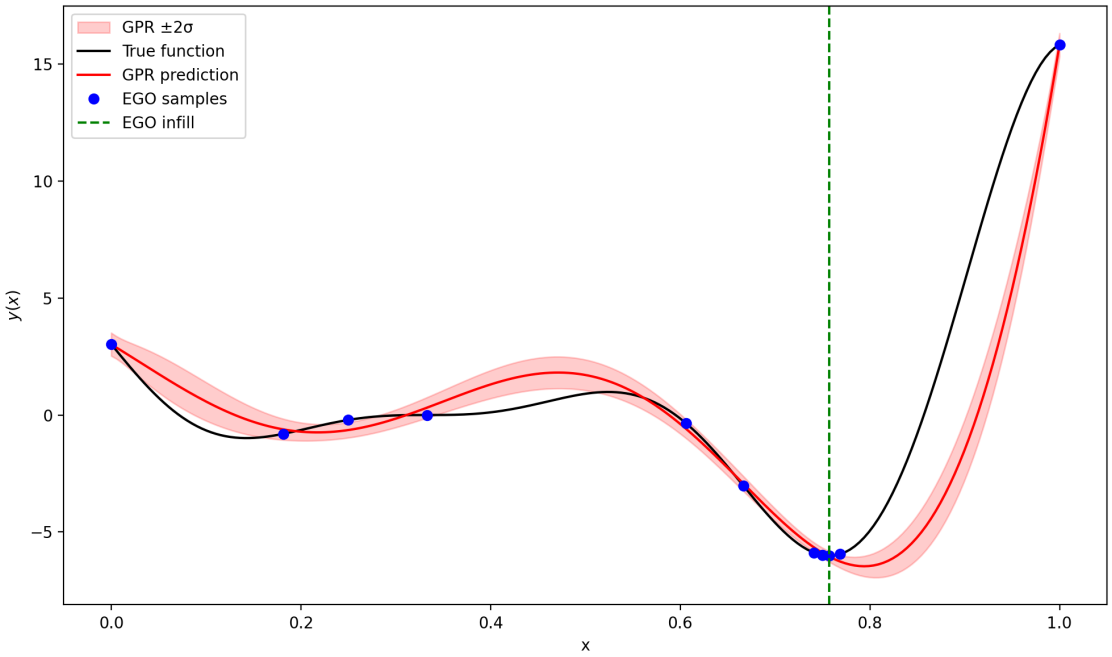

Figure 1: The EGO using the GPR surrogate model found the global minimum, and it closely approximates the true function. The uncertainty is higher in regions far from sample points.

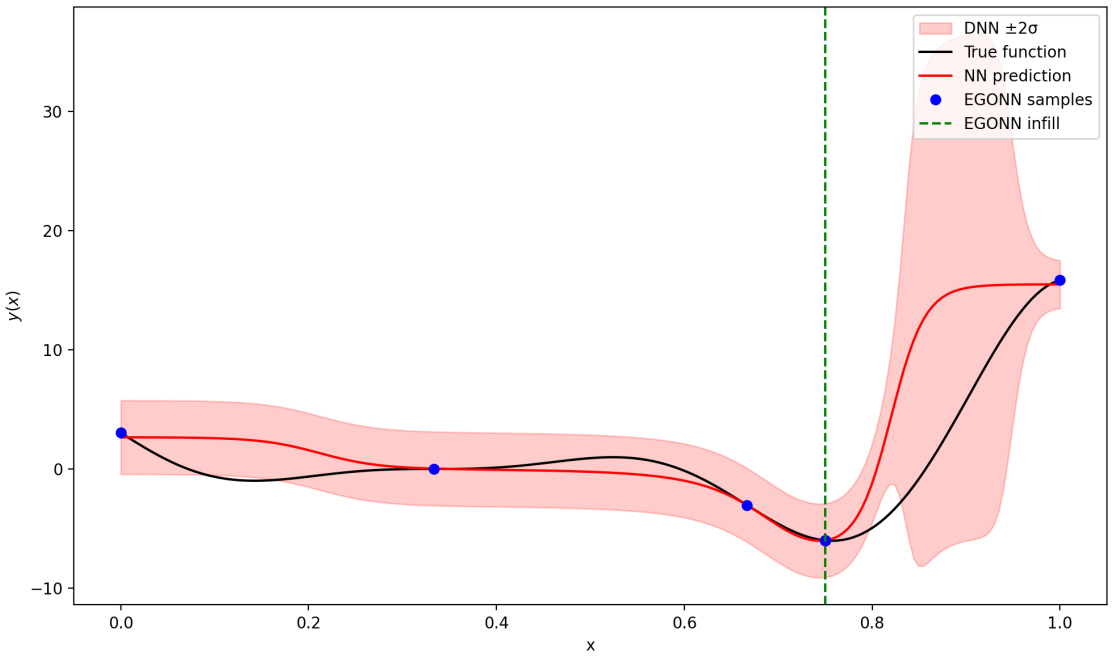

Figure 2: The 95% confidence interval is wider than the GPR model, indicating that the number of samples used is insufficient. But the optimizer converged to the global minimum.

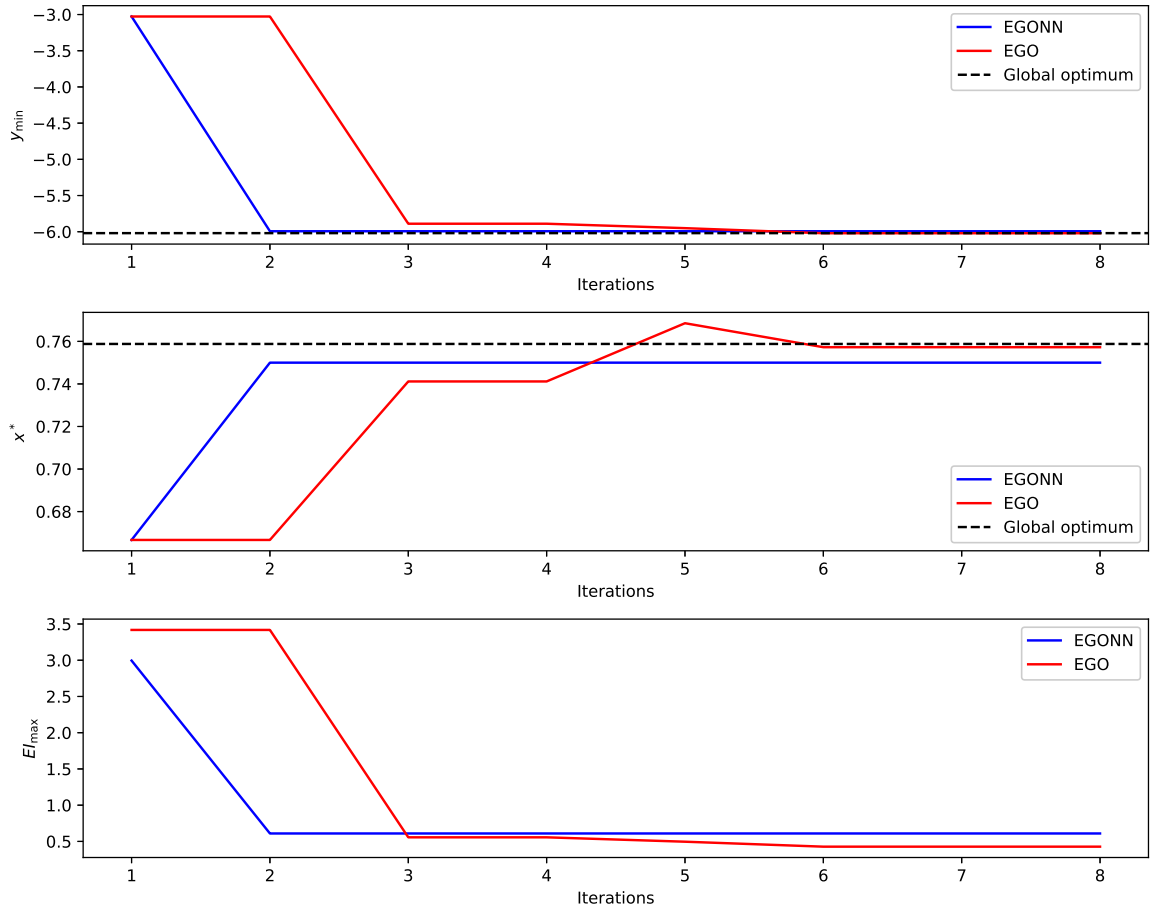

Figure 3: This is the optimization history of GPR and NN-based surrogate models. The global minimum is found at (0.76, -6), and the maximum expected improvement reduces from 3.5 to less than 0.5 within eight iterations.

The evolution of the surrogate model of Case 1, showing the function prediction (left), prediction uncertainty (middle), and expected improvement (right) for the eight iterations, will be shown below. The first four iterations (where the maximum change is observed) are studied to validate the results with the journal paper.

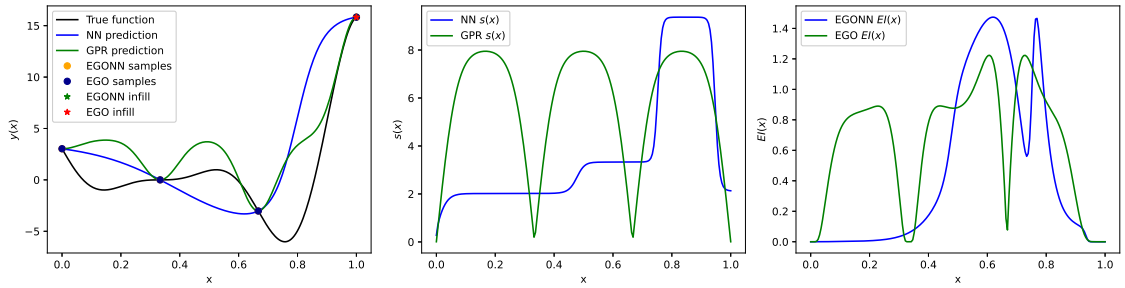

Figure 4: Iteration 1

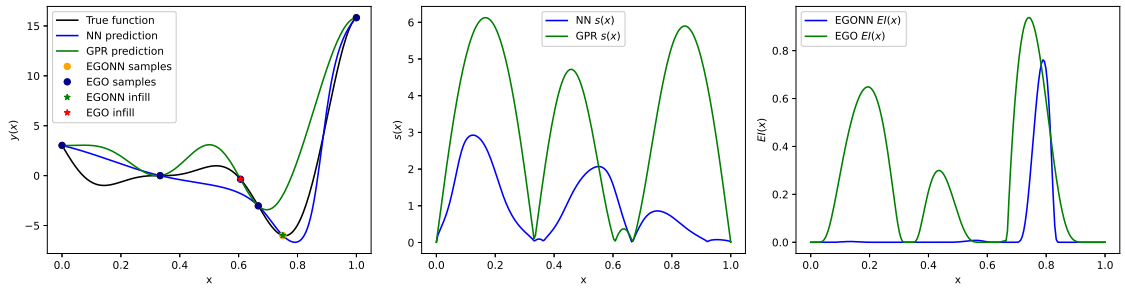

Figure 5: Iteration 2

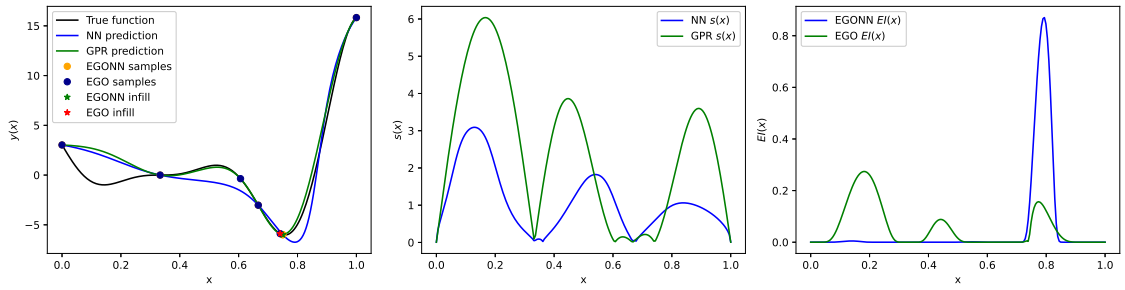

Figure 6: Iteration 3

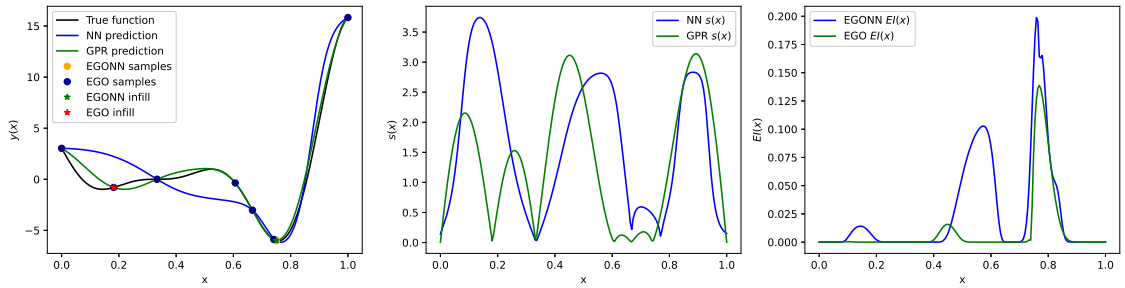

Figure 7: Iteration 4

The first four modeling cycles are analyzed, as the maximum expected improvement decreases significantly within these iterations. During these cycles, both EGO and EGONN show a considerable reduction in the maximum prediction uncertainty, dropping from approximately 9 to 3.5. At the same time, the magnitude of the maximum expected improvement reduces by two orders of magnitude, from around 1.4 to 0.2.

It is also observed that the prediction uncertainty and the location of the maximum expected improvement differ noticeably between the Gaussian Process Regression (GPR) and the Deep Neural Network (DNN). This difference influences the placement of new samples in each cycle. Although both EGO and EGONN successfully locate the approximate global minimum, the DNN provides a much closer fit to the true function during the early iterations than the GPR.

CASE 2: Constrained Branin function:

The EGONN surrogate model setup: 30 Latin Hypercube Sampling (LHS) initial data samples are used with an infill budget of 40 samples. 100 uniformly distributed data samples are used for the EGONN model. The DNNs have four hidden layers. The first layer has 8 neurons, the second layer has 16, the third layer has 32, and the fourth layer has 64. The rectified linear activation function (ReLu) is used for all neurons. The number of epochs is set to 2000.

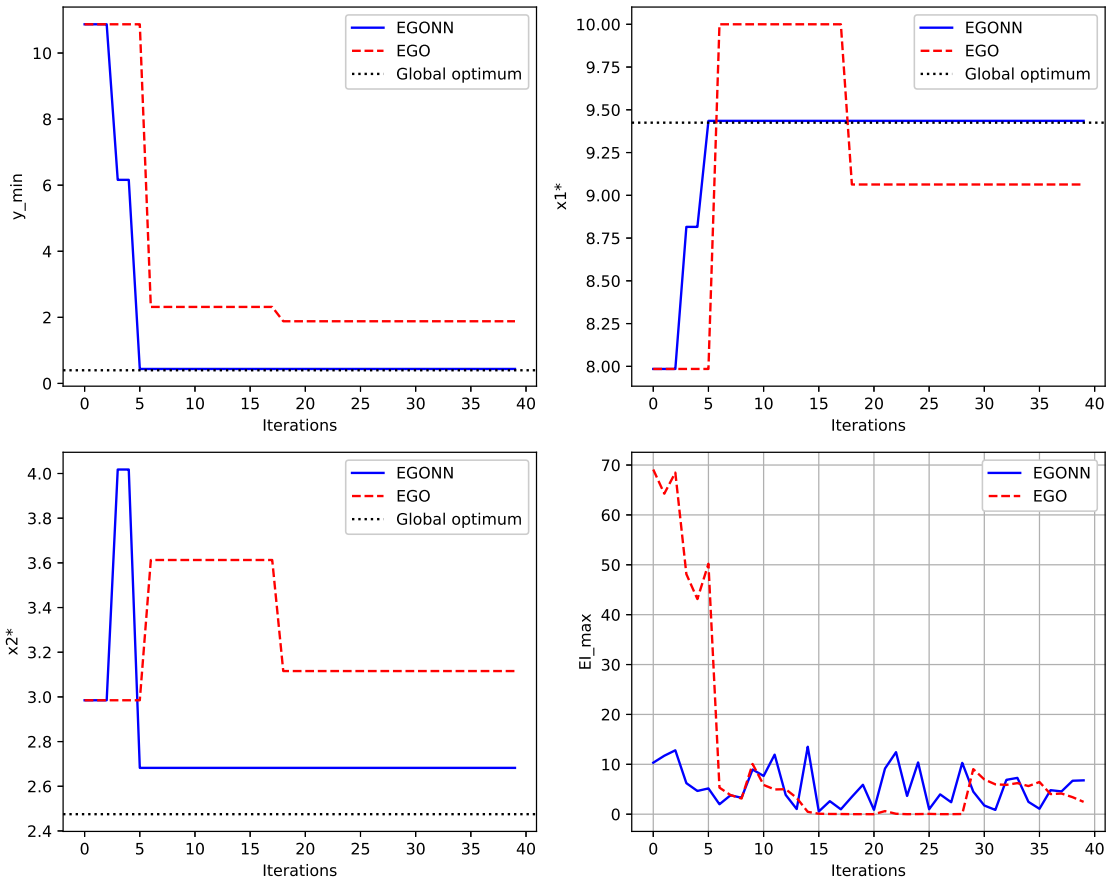

Figure 8: Optimization history of Branin function with minimum function value y_{min}, x_1^* and x_2^* locations of minimum observation, and maximum expected improvement.

The plot of y_{min} shows that EGO requires approximately 20 iterations to approach the minimum value y_{min} =0, whereas EGONN reaches close to this value within 5 iterations. This shows the faster convergence of EGONN compared to EGO. The x_1^* and x_2^* plots show that EGONN is more accurate and faster to converge to the global minimum. The maximum expected improvement (EI) reduces drastically for EGO whereas EI of the EGONN algorithm hovers between 0 to 10.

From the contours, we can see that the infill samples for both the EGO and EGONN algorithm are all clustered around the global minimum. The contour plot represents the true function, with the global minimum marked as the yellow diamond. The EGONN has much better infill points distribution than EGO algorithm, which shows how EGONN explored a lot of function domain.

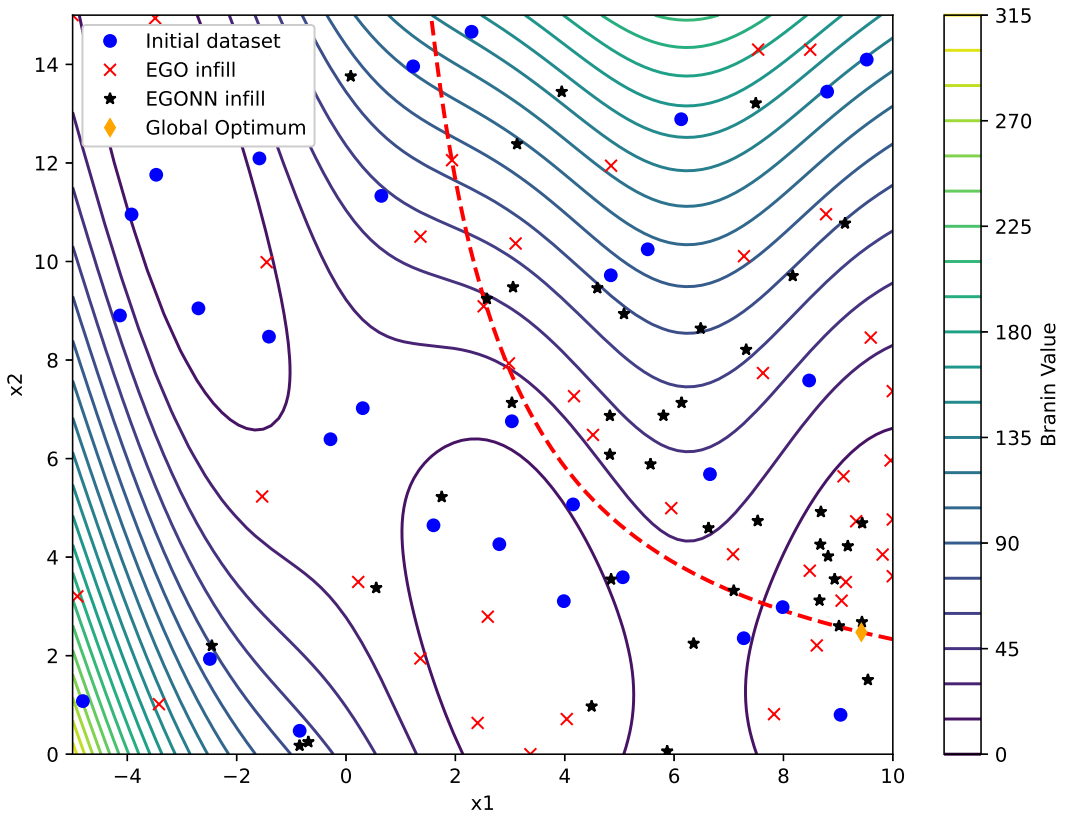

Figure 9: Contours of the the true function and constraint along with samples and infill points.

Conclusion

These results show that EGONN is much better algorithm when compared to GPR for a 2-dimensional function, but this is only with following surrogate model setup. For a different kernel or random point (numpy seed) selection, the results will vary a lot. Even though the results from journal paper [3] was not exactly replicated, we can observe that both the algorithms are efficient in surrogate optimization and can be implemented for higher dimensional functions and complex design problems like aerodynamic shape optimization.

References

[1] Alexander Forrester and Andy Keane. “Recent advances in surrogate-based optimization”. In: *Progress in Aerospace Sciences* 45.1 (2009), pp. 50–79. DOI: 10.1016/j.paerosci.2008.11.001.

[2] Donald Jones, Matthias Schonlau, and William Welch. “Efficient global optimization of expensive black-box functions”. In: *Journal of Global Optimization* 13.4 (1998), pp. 455–492. DOI: 10.1023/A:1008306431147.

[3] Pavankumar Koratikere et al. “Efficient Global Optimization Algorithm Using Neural Network-Based Prediction and Uncertainty”. In: *AIAA SCITECH 2023 Forum*. 2023. DOI: 10.2514/6.2023-2683.

[4] Nestor Queipo et al. “Surrogate-based analysis and optimization”. In: *Progress in Aerospace Sciences* 41.1 (2005), pp. 1–28. DOI: 10.1016/j.paerosci.2005.02.001.