Machine-Learning in Simulation-Driven Optimization
Yoel TENNE
Ariel University, Israel

Keywords: Optimization, Computer simulation, Machine learning.

Abstract. Modern engineering design optimization often uses computer simulations to evaluate candidate designs. For some of these designs the simulation can fail for an unknown reason, which in turn hampers the optimization process. To handle such scenarios more effectively this study proposes the integration of classifiers, borrowed from the domain of machine learning, into the optimization process. Numerical experiments show that the proposed approach improves the effectiveness of the optimization search.

Introduction

Modern engineering increasingly replaces laboratory experiments with computer simulations to improve the design efficiency. This setup transforms the design process into an optimization problem having several unique features [1]:

- The simulation acts as the objective function since it assigns an output value to a candidate design, but it's analytic expression is unknown (a black-box function).
- Each simulation evaluation is computationally expensive and this severely restricts the number of possible evaluations.
- Due to the real-world physics being modelled and the numerical simulation process the black-box function may have a complicated landscape which exacerbates the optimization difficulty.

Such problems are termed in the literature as expensive black-box optimization problems and have received significant attention in recent years [1]. Beyond the challenges mentioned a further issue is that for some candidate designs the simulation will fail and will not provide an output value. In this study such designs are termed as simulator-infeasible (SI), while valid designs are termed simulator-feasible (SF). The existence of SI designs implies that: i) the objective function becomes discontinuous, which introduces an additional optimization difficulty, and ii) such designs can consume large computational resources without assisting the search, thereby degrading the search effectiveness. In this study it is assumed that simulation evaluations are deterministic, namely, a SF design will consistently succeed while a SI one will consistently result in a simulation failure.

While multiple studies such as [2, 3, 4] have reported encountering SI designs, prevalent methodologies to handle such vectors display significant shortcomings, for example they discard such vectors all together or they assign such vectors a fictitious penalized value. Both approaches can severely degrade the effectiveness of the optimization search, and accordingly to handle such vectors effectively this study proposes to a machine-learning approach in which a classifier is incorporated into the search to predict if a candidate design is likely to be SI or not. These predictions are then used by the optimization algorithm to divert the search to designs which are likely to be SF, and avoids the issues associated with the approaches mentioned. The remainder of this paper is organized as follows: Section 2 provides the pertinent background information, Section 3 describes a working implementation, while Section 4 gives a performance evaluation and summary.

Background

Simulation-driven optimization problems arise often in engineering and several approaches have been proposed to effectively handle them. An established methodology is that of using approximation models, also termed as *metamodels or surrogates*, which approximate the true
expensive function and provide predicted objective values at a lower computational cost [1]. Metamodels variants include radial basis functions (RBF) and Kriging from geostatistics, artificial neural networks from the domain of machine learning, and polynomial approximations from applied mathematics. During the optimization the optimizer will receive the predicted objective values from the metamodel, thereby reducing the number of calls to the expensive simulation.

Also as mentioned, an additional challenge is that the computer simulation may consistently fail for some candidate designs. Such simulation-infeasible (SI) designs have been referred to in the literature, for example in references [3,4,5,6,7,14]. Given their harmful impact on the optimization, several techniques have been examined in an attempt to handle them, for example by assigning them a penalized value [4,5] or by discarding them altogether [6]. However, such techniques exhibit several demerits, namely, using a penalized vectors during the metamodel training can severely degrade its prediction accuracy, while discarding SI designs can result in a loss of expensive and potentially beneficial information. As an example, Figure 1 shows two Kriging metamodels which were used to approximate the Rosenbrock function: (a) shows the resultant metamodel when 30 regular vectors were used, while (b) shows the result metamodel when 20 penalized vectors were added to the previous training sample, with a penalized value which taken as the worst objective value from the initial sample. The resultant metamodel shape shows a severely deformed landscape with many artificial optima, and this clearly exacerbates the optimization difficulty.

![Figure 1. Example of the impact of adding penalized vectors on the resultant metamodel.](image)

**Implemented Framework**

Following the above discussion a framework has been implemented to leverage on machine learning concepts to bias the search more effectively towards SF vectors without degrading the metamodel prediction accuracy. The framework begins by generating a sample of vectors with the Latin hypercube sampling (LHS) method [14] which yields a space-filling sample which improves the prediction accuracy of the resultant metamodel. The sample vectors are evaluated with the true expensive function and are cached.

The main optimization loop then begins where a Kriging metamodel is trained by using only the SF vectors in the cache, and then a classifier is trained by using all the cached, namely, both the SF and SI ones. The classifier is a machine-learning component which is used here to predict if a candidate design is simulation-infeasible or not. Mathematically, given a set of vectors which have been assigned to $n_s$ distinct sets $s_j, j = 1 \ldots n_s$, a classifier maps a new vector into one of these sets, namely:

$$c(x): \mathbb{R}^n \to s_j$$

A variety of classifier variants have been proposed, but typically the variant best suited to the problem being solved is unknown. To address this the proposed framework uses an autonomous classifier-selection step based on the cross-validation (CV) procedure from statistical theory. Specifically, the cache of previously evaluated vectors (both SI and SF) are split into a training set and a testing set. In turn for each of the prescribed classifier types a classifier is trained by using the training set and its prediction accuracy is estimated by using the testing set. After all candidate classifiers have been evaluated, the algorithm selects the one having the smallest prediction error. In
this study three candidate classifiers were considered: kNN (k nearest neighbours, \( k=3 \)), linear discrete analysis (LDA), and a support vector machine (SVM) [1].

After training the metamodel and classifier an optimization step is performed. However, since only a small cache of vectors is available the resultant metamodel is likely to be inaccurate. To ensure convergence of the optimization search to a valid optimum of the true expensive function the search is performed with a trust-region (TR) approach [13], where a restricted step is taken and the search is updated based on the success (or failure) of this step to produce a new best solution. The search itself is performed with a real-coded EA [8], and where the EA received objective values from the following modified objective function

\[
\hat{m}(x) = \begin{cases} 
    m(x) \text{if } c(x) \text{ predicts } x \text{ is simulation feasible} \\
    p \text{if } c(x) \text{ predicts } x \text{ is simulation infeasible}
\end{cases}
\]

where \( p \) is a penalized objective value. In this manner the optimization algorithm receives the metamodel prediction for candidate vectors which are predicted to be simulation-feasible, but it receives a penalized objective value for those predicted to be simulation-infeasible. As such the search is diverted towards vectors which are expected to be valid but without deforming the metamodel and degrading the search effectiveness. For the trust-region updates, the following updates procedures were implemented:

- If the solution found by the EA is better than the current best solution: The metamodel is considered accurate, and therefore, the trust-region radius is doubled.
- If the solution found isn't better than the current best solution and the number of vectors in the trust-region is deemed as too small: The failure of the optimization step is attributed to poor metamodel accuracy arising from too few vectors in the trust-region. In this case, a new vector is generated in the trust-region and evaluated with the true expensive function.
- If the solution found isn't better than the current best and the number of vectors in the trust-region is deemed as sufficient: As above the optimization step failed to find a better solution but here the metamodel is deemed as sufficiently accurate and hence the failure is attributed to the TR being too large. Accordingly, the trust-region radius is halved to restrict the search range.

At the end of the optimization loop all the vectors which have evaluated in the current iteration are added to the memory cache, and the process repeats until the number of simulation reaches the prescribed limit, or the trust-region radius is sufficiently small.

**Performance Evaluation**

For evaluation the proposed framework it was applied to an engineering problem of airfoil shape optimization. Here the goal is to find an airfoil shape which minimizes the drag (aerodynamic friction) while maximizing the lift (the force which keeps the aircraft airborne). There is also a structural constraint that the thickness of the airfoil (\( t \)) between 20\% to 80\% of its chord line must be larger than a critical value \( t^* = 0.1 \). To accomplish this, the following test function was used:

\[
f = -\frac{c_l}{c_d} + \Omega, \Omega = \begin{cases} 
    t^*, & c_l, c_d \text{ if } t < t^* \\
    0 & \text{otherwise}
\end{cases}
\]

where \( c_l, c_d \) are the lift and drag coefficients, respectively, and \( \Omega \) is a penalty term which added to airfoil which violate the thickness constraint. Candidate airfoils were generated with the Hicks-Henne method, namely:

\[
y = y_b + \sum_{i=1}^{n} \theta_i \tau_i(x), \tau_i(x) = \sin\left(\pi x \log_{10}(1/(k+i))\right), i = 1...k
\]

where \( y_b \) is a baseline airfoil shape, taken as the NACA0012, \( \theta_i \in [0,1] \) are coefficients, and \( \tau_i(x) \) are basis functions, where \( k \) is the user-prescribed number of basis of basis functions to use. The goal is then to find the values of the coefficients \( \theta_i \) which yield the optimal airfoil.
of candidate airfoils was performed with the Xfoil aerodynamic analysis code for subsonic airfoils [10], and each evaluation required up to 30 seconds on a desktop computer. Figure 2 gives the layout of the airfoil problem formulation.

The airfoil problem is a valid test case since it contains simulation-infeasible (SI) vectors. Furthermore, numerical tests have established that the percentage of SI vectors increases with the angle of attack (AOA) parameter, and particularly that there is a high failure rate from 20° onwards. Accordingly, the numerical tests were performed with the settings $AOA = 20^\circ, 30^\circ, 40^\circ$ to ensure challenging test cases. The other settings were a flight speed of Mach=0.75 and an altitude of 32 Kft. For comparison the proposed approach was benchmarked against two reference algorithms from the literature: i) KEAS which is an evolutionary algorithm which uses periodic sampling of new vectors to refresh a Kriging metamodel [11], and ii) EICMAS: An algorithm which combines the Expected Improvement approach [4] with a covariance matrix adaption evolutionary strategies search.

In the benchmarking tests there was a limit of 200 calls to the Xfoil simulation, and the size of the initial sample was 20, namely 10% of the total optimization budget. To ensure a valid statistical analysis 30 runs were repeated with each algorithm in each AOA setting, and Table 1 gives the resultant test statistics of mean and median.

<table>
<thead>
<tr>
<th>AOA [degrees]</th>
<th>Statistic</th>
<th>Proposed algorithm</th>
<th>KEAS</th>
<th>EICMAS</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>Mean</td>
<td>-1.0E+1</td>
<td>-6.8E+0</td>
<td>-0.9E+1</td>
</tr>
<tr>
<td></td>
<td>Median</td>
<td>-1.1E+1</td>
<td>-6.7E+0</td>
<td>-0.8E+0</td>
</tr>
<tr>
<td>30</td>
<td>Mean</td>
<td>-3.3E+0</td>
<td>-3.0E+0</td>
<td>-2.8E+0</td>
</tr>
<tr>
<td></td>
<td>Median</td>
<td>-3.1E+0</td>
<td>-2.9E+0</td>
<td>-2.7E+0</td>
</tr>
<tr>
<td>40</td>
<td>Mean</td>
<td>-2.9E+0</td>
<td>-2.6E+0</td>
<td>-2.5E+0</td>
</tr>
<tr>
<td></td>
<td>Median</td>
<td>-2.7E+0</td>
<td>-2.5E+0</td>
<td>-2.4E+0</td>
</tr>
</tbody>
</table>

The best statistics in each test case are shown in bold.

The test results show that the proposed framework consistently outperformed the reference algorithms as evident from the mean and median statistics. These results are attributed to the enhanced optimization process brought by the classifier and the resulting effective diversion of the search away from SI vectors, instead of simply penalizing vectors or discarding them. Overall, the
analysis shows the merit of the proposed approach and the gains achieved by integrating a machine learning component into simulation-driven optimization.

References


