Chapter 6
Fitness Meta-Modeling

6.1 Introduction

In expensive optimization problems, the reduction of the number of fitness function calls has an important part to play. Evolutionary operators produce candidate solutions $\mathbf{x}_i$ in the solution space that are evaluated on fitness function f. If a regression method $\hat{f}$ is trained with the pattern-label pairs $(\mathbf{x}_i, f(\mathbf{x}_i)), i = 1, \ldots, N$, the model $\hat{f}$ can be used to interpolate and eventually extrapolate the fitness of a novel solution $\mathbf{x}'$ that has been generated by the evolutionary operators. Model $\hat{f}$ is also known as meta-model or surrogate in this context. Many different regression methods can be employed for this purpose. An important question is how to manage the meta-model. Past fitness function evaluations are stored in a training set. The questions come up, which patterns to use for training the meta-model, how often to tune the meta-model, and when to use the fitness function or the meta-model as surrogate. Some methods directly use the fitness evaluations of the meta-model for the evolutionary process. Others use the meta-model for pre-screening, i.e., each solution is first evaluated on the meta-model and the most successful ones are evaluated on the real fitness function before being selected as parents for the following generation.

In this chapter, we analyze nearest neighbor regression when optimizing with a (1+1)-ES and Rechenberg's step size control technique. Nearest neighbor regression makes use of the idea that the label of an unknown pattern $\mathbf{x}'$ should get the label of the k closest patterns in the training set. This method is also known as instance-based and non-parametric approach. It does not induce a functional model like linear regression. The objective of this chapter is to show that a comparatively simple hybridization can result in a very effective optimization strategy. The chapter is structured as follows. Nearest neighbor regression is introduced in Sect. 6.2. The integration of the kNN meta-model is presented in Sect. 6.3. Related work is discussed in Sect. 6.4. The approach is experimentally analyzed in Sect. 6.5. Conclusions are drawn in Sect. 6.6.

O. Kramer, *Machine Learning for Evolution Strategies*,
Studies in Big Data 20, DOI 10.1007/978-3-319-33383-0_6

6.2 Nearest Neighbors

Nearest neighbor regression, also known as kNN regression, is based on the idea that the closest patterns to a target pattern $\mathbf{x}'$, for which we seek the label, deliver useful information for completing it. Based on this idea, kNN assigns the class label of the majority of the k-nearest patterns in data space. For this sake, we have to be able to define a similarity measure in data space. In $\mathbb{R}^d$, it is reasonable to employ the Minkowski metric (p-norm)

$$\|\mathbf{x}' - \mathbf{x}_j\|_p = \left(\sum_{i=1}^{d} |(x_i)' - (x_i)_j|^p \right)^{1/p} \tag{6.1}$$

with parameter $p \in \mathbb{N}$. The distance measure corresponds to the Euclidean distance for $p = 2$ and the Manhattan distance for $p = 1$. In other data spaces, adequate distance functions have to be chosen, e.g., the Hamming distance in $\mathbb{B}^d$. For regression tasks, kNN can also be applied. As continuous variant, the task is to learn a function $\hat{f} : \mathbb{R}^d \to \mathbb{R}$ known as regression function. For an unknown pattern $\mathbf{x}'$, kNN regression computes the mean of the function values of its k-nearest neighbors

$$\hat{f}(\mathbf{x}') = \frac{1}{K} \sum_{i \in \mathcal{N}_K(\mathbf{x}')} y_i \tag{6.2}$$

with set $\mathcal{N}_K(\mathbf{x}')$ containing the indices of the k-nearest neighbors of pattern $\mathbf{x}'$ in the training data set $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$. Normalization of patterns is usually applied before the machine learning process, e.g., because different variables can come in different units.

The choice of k defines the locality of kNN. For $k = 1$, little neighborhoods arise in regions, where patterns from different classes are scattered. For larger neighborhood sizes, e.g. $k = 20$, patterns with labels in the minority are ignored. Neighborhood size k is usually chosen with the help of cross-validation. For the choice of k, grid-search or testing few typical choices like [1, 2, 5, 10, 20, 50] may be sufficient. This restriction reduces the effort for tuning the model significantly. Nearest neighbor methods are part of the SCIKIT-LEARN package.

- The command `from sklearn import neighbors` imports the SCIKIT-LEARN implementation of kNN.
- `clf = neighbors.KNeighborsRegressor(n_neighbors=k)` calls kNN with k neighbors using uniform weights bei default. Optionally, an own distance function can be defined.
- `clf.fit(X,y)` trains kNN with patterns `X` and corresponding labels `y`.
- `y_pred = clf.predict(X_test)` predicts the class labels of test data set `X_test`.

The method kNN demonstrates its success in numerous applications, from classification of galaxies in digital sky surveys to the problem of handwritten digits and EKG data [1]. As kNN uses the training points that are nearest to the target patterns, kNN employs high variance and low bias, see the discussion in Chap. 4. For example, if the target patterns lie at the location of a training pattern, the bias is zero. Cover and Hart [2] show that the error rate of kNN with $k = 1$ is asymptotically bound by twice the Bayes error rate. The proof of this result is also sketched in Hastie et al. [1].

6.3 Algorithm

In this section, we introduce the meta-model-based ES. The main ingredients of meta-model approaches are the training set, which stores past fitness function evaluations, the meta-model maintenance mechanism, e.g., for parameter tuning and regularization, and the meta-model integration mechanism that defines how it is applied to save fitness function evaluations. Figure 6.1 illustrates the meta-model principle. Solutions are evaluated on the real fitness function f resulting in the blue squares. The meta-model is trained with these examples resulting in the red curve. This is basis of the meta-model evaluation of solutions, represented as red little squares.

One alternative to use the meta-model is to test each candidate solution with a certain probability on the meta-model and to use the predicted value instead of the real fitness function evaluation in the course of the evolutionary optimization process. We employ a different meta-model management that is tailored to the (1+1)-ES.

Algorithm 4 shows the pseudocode of the (1+1)-ES with meta-model (MM-ES) and Rechenberg's adaptive step size control [3]. If the solution $\mathbf{x}'$ has been evaluated on f, both will be combined to a pattern and as pattern-label pair $(\mathbf{x}', f(\mathbf{x}'))$ included to the meta-model training set. The last N solutions and their fitness function evaluations build the training set $\{(\mathbf{x}_i, f(\mathbf{x}_i))\}_{i=1}^{N}$. After each training set update, model $\hat{f}$ can be re-trained. For example in case of kNN, a new neighborhood size k may be chosen with cross-validation.

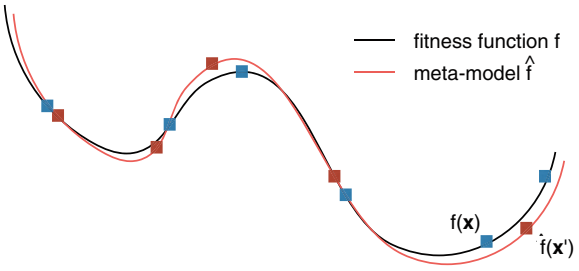


Fig. 6.1 Illustration of fitness meta-model. Solutions (*blue squares*) like $\mathbf{x}$ are evaluated on f (*black curve*). The meta-model $\hat{f}$ (*red curve*) is trained with the evaluated solutions. It estimates the fitness $\hat{f}(\mathbf{x}')$ of new candidate solutions (*red squares*) like $\mathbf{x}'$

Algorithm 4 MM-ES

initialize $\mathbf{x}$
repeat
  adapt σ with Rechenberg
  $\mathbf{z} \sim \sigma \cdot \mathcal{N}(\mathbf{0}, \mathbf{I})$
  $\mathbf{x}' = \mathbf{x} + \mathbf{z}$
  if $\hat{f}(\mathbf{x}') \leq f(\mathbf{x}_{-t})$ **then**
    evaluate $\mathbf{x}' \rightarrow f(\mathbf{x}')$
    last N solutions $\rightarrow \{(\mathbf{x}_i, f(\mathbf{x}_i))\}_{i=1}^{N}$
    train $\hat{f}$
    if $f(\mathbf{x}') \leq f(\mathbf{x})$ **then**
      replace $\mathbf{x}$ with $\mathbf{x}'$
    end if
  end if
until termination condition

The meta-model integration we employ is based on the idea that solutions are only evaluated, if they are promising. Let $\mathbf{x}$ be the solution of the last generation and let $\mathbf{x}'$ be the novel solution generated with the mutation operator. If the fitness prediction $\hat{f}(\mathbf{x}')$ of the meta-model indicates that $\mathbf{x}'$ employs a better fitness than the tth last solution $\mathbf{x}_{-t}$ that has been generated in the past evolutionary optimization progress, the solution is evaluated on the real fitness function f. The tth last solution defines a fitness threshold that assumes $\hat{f}(\mathbf{x}')$ may underestimate the fitness of $\mathbf{x}'$. The evaluations of candidate solutions that are worse than the threshold are saved and potentially lead to a decrease of the number of fitness function evaluations. Tuning of the model, e.g., the neighborhood size k of kNN with cross-validation, may be reasonable in certain optimization settings.

Last, the question for the proper regression model has to be answered. In our blackbox optimization scenario, we assume that we do not know anything about the curvature of the fitness function. For example, it is reasonable to employ a polynomial model in case of spherical fitness function conditions. But in general, we cannot assume to have such information.

6.4 Related Work

Meta-models, also known as surrogates, are prominent approaches to reduce the number of fitness function evaluations in evolutionary computation. Most work in meta-modeling concentrates on fitness function surrogates, few also on reducing constraint functions evaluations. In this line of research, early work concentrates on neural networks [4] and on Kriging [5]. Kriging belongs to the class of Gaussian process regression models and is an interpolation method. It uses covariance information and is based on piecewise-polynomial splines. Neural networks and Kriging

meta-models are compared in [6]. An example for the recent employment of Kriging models is the differential evolution approach by Elsayed et al. [7].

Various kinds of mechanisms allow the savings of fitness function evaluations. Cruz-Vega et al. [8] employ granular computing to cluster points and adapt the parameters with a neuro-fuzzy network. Verbeeck et al. [9] propose a tree-based meta-model and concentrate on multi-objective optimization. Martínez and Coello [10] also focus on multi-objective optimization while employing a support vector regression meta-model. Loshchilov et al. [11] combine a one-class SVM with a regression approach as meta-model in multi-objective optimization. Ensembles of support vector methods are also used for in the approach by Rosales-Pérez [12] in multi-objective optimization settings. Ensembles combine multiple classifiers to reduce the fitness prediction error.

Kruisselbrink et al. [13] apply the Kriging model in CMA-ES-based optimization. The approach puts an emphasis on the generation of archive points for improving the meta-model. Local meta-models for the CMA-ES are learned in the approach by Bouzarkouna et al. [14], who train a full quadratic local model for each sub-function in each generation. Also Liao et al. [15] propose a locally weighted meta-model, which only evaluates the most promising candidate solutions. The local approach is similar to the nearest neighbor method we use in the experimental part, as kNN is a local method.

There is a line of research that concentrates on surrogate-assisted optimization for the CMA-ES. For example, the approach by Loshchilov et al. [16] adjusts the life length of the current surrogate model before learning a new surrogate as well as its hyper-parameters. A variant with larger population sizes [17] leads to a more intensive exploitation of the meta-model.

Preuss et al. [18] propose to use a computationally cheap meta-model of the fitness function and tune the parameters of the evolutionary optimization approach on this surrogate. Kramer et al. [19] combine two nearest neighbor meta-models, one for the fitness function, and one for the constraint function with an adaptive penalty function in a constrained continuous optimization scenario.

Most of the work sketched here positively reported savings in fitness function evaluations, although machine learning models and meta-model managing strategies vary significantly.

6.5 Experimental Analysis

In this section, we experimentally analyze the meta-model-based ES. Besides the convergence behavior, we analyze the influence of the neighborhood size k and the training set size N. Table 6.1 shows the experimental results of 25 runs of the (1+1)-ES and the MM-ES on the Sphere function and on Rosenbrock for kNN as meta-model with $k = 1$. Both algorithms get a budget of 5000 function evaluations. The results show the mean values and the corresponding standard deviations. On the Sphere function, we employ the setting $t = 10$. We can observe that the ES with

Table 6.1 Experimental comparison of (1+1)-ES and the MM-ES on the Sphere function and on Rosenbrock

Problem		(1+1)-ES		MM-ES		Wilx.
	d	Mean	Dev	Mean	Dev	p-value
Sphere	2	2.067e-173	0.0	2.003e-287	0.0	0.0076
	10	1.039e-53	1.800e-53	1.511e-62	2.618e-62	0.0076
Rosenbrock	2	0.260	0.447	8.091e-06	7.809e-06	0.0076
	10	0.519	0.301	2.143	2.783	0.313

meta-model leads to a significant reduction of fitness function evaluations. For $d = 2$, the standard deviation even falls below a value that is measurable due to a limited machine accuracy. The results are statistically significant, which is confirmed by the Wilcoxon test. On Rosenbrock, we set $t = 50$. The MM-ES significantly outperforms the (1+1)-ES for $d = 2$, but only in few runs on Rosenbrock leading to no statistical superiority for $d = 10$.

Figure 6.2 compares the evolutionary runs of the (1+1)-ES and the MM-ES on the Sphere function for (a) $d = 2$ and (b) $d = 10$. As of the very beginning of the optimization process, even the worst evolutionary runs with meta-model are better than the best evolutionary runs of the (1+1)-ES without meta-model. This effect is even more significant for $d = 2$.

Now, we analyze the training set size N as it is has a significant influence on the quality of regression model $\hat{f}$. In Fig. 6.3, we compare the two training set sizes $N = 20$ and $N = 500$ on the Sphere function. Again, we use the same settings, i.e., 5000 fitness function evaluations in each run for $d = 2$ and $d = 10$. The runs show that a too small training set lets the search become less stable in both cases. Some runs may get stuck because of inappropriate fitness function estimates. Further analyses show that a training set size of $N = 500$ is an appropriate choice.

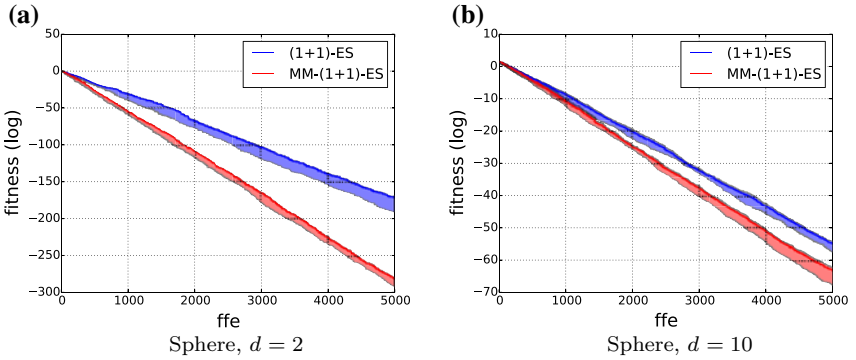
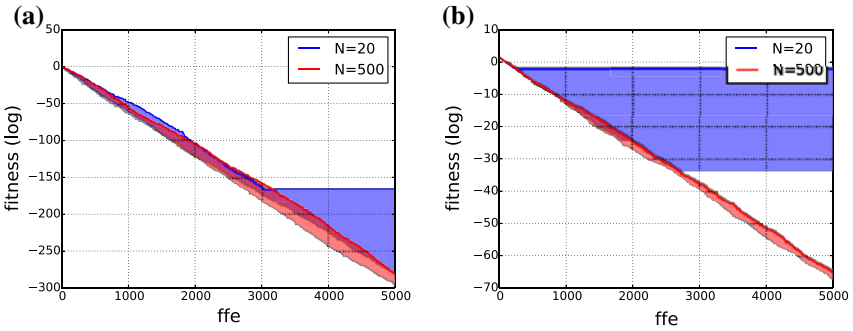


Fig. 6.2 Comparison of (1+1)-ES and MM-ES on the Sphere function with **a** $d = 2$ and **b** $d = 10$

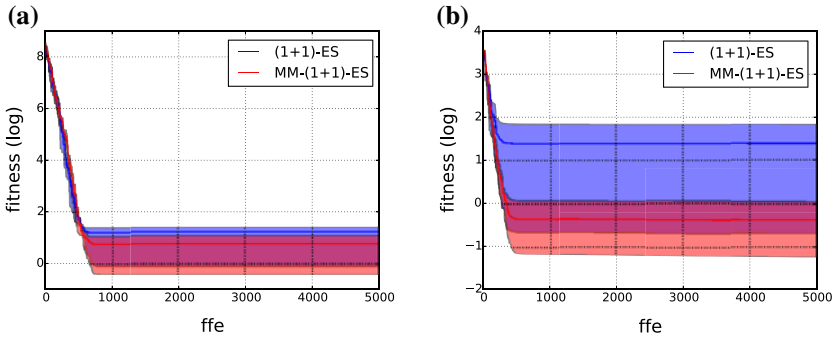


Sphere, $d = 2$ Sphere, $d = 10$

Fig. 6.3 Comparison of meta-model sizes $N = 20$ and $N = 500$ on the Sphere function with **a** $d = 2$ and **b** $d = 10$

Our analysis of the neighborhood size k have shown that the choice $k = 1$ yields the best results in all cases. Larger choices slow down the optimization or let the optimization process stagnate, similar to the stagnation we observe for small training set sizes. Hence, we understand the nearest neighbor regression meta-model with $k = 1$ as local meta-model, which also belongs to the most successful in literature, see Sect. 7.4.

Figure 6.4 shows the experimental results of the MM-ES on the Cigar function and on Rosenbrock for $d = 10$ dimensions and 5000 fitness function evaluations. It confirms that the MM-ES reduces the number of fitness function evaluations in comparison to the standard (1+1)-ES.



Cigar, $d = 10$ Rosenbrock, $d = 10$

Fig. 6.4 Analysis of MM-ES on **a** Cigar and **b** on Rosenbrock for $d = 10$

6.6 Conclusions

Meta-modeling in evolutionary optimization is a frequent and well-known technique that allows saving expensive fitness function evaluations. In this chapter, we apply nearest neighbor regression in a (1+1)-ES with Gaussian mutation and Rechenberg's adaptive step size control. Fitness function evaluations are saved, if the predicted fitness is worse than the tth best element in the training set. The comparison with the tth best element of the past search and not with the best element is a pessimistic perspective. The search might get stuck or at least decelerate, if the model's quality is overestimated. Further, it turns out that the small neighborhood size $k = 1$ is the best choice. In our experiments, we observe significant performance wins on the Sphere function. The optimization runs of the (1+1)-ES with meta-model outperform the native ES. The Wilcoxon test confirms the statistical significance of all results.

Meta-models as fitness function surrogates can lead to various difficulties. The training set should be large enough to offer diversity that can be exploited for fitness predictions during the evolutionary optimization process. However, one has to keep in mind that inexact fitness predictions based on bad surrogates may disturb the evolutionary process and result in deteriorations instead of improvements. Various other ways to employ a regression model as meta-model are possible instead of pre-selection. For example, an interesting approach is the use of the meta-model as fitness function with a certain probability, but without further checks on the real fitness function.

References

1. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning. Springer, New York (2009)
2. Cover, T., Hart, P.: Nearest neighbor pattern classification **13**, 21–27 (1967)
3. Rechenberg, I.: Evolutionsstrategie - Optimierung technischer Systeme nach Prinzipien der biologischen Evolution. Frommann-Holzboog, Stuttgart (1973)
4. Jin, Y., Olhofer, M., Sendhoff, B.: On evolutionary optimization with approximate fitness functions. In: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2000, pp. 786–793 (2000)
5. Armstrong, M.: Basic Linear Geostatistics. Springer (1998)
6. Willmes, L., Bäck, T., Jin, Y., Sendhoff, B.: Comparing neural networks and kriging for fitness approximation in evolutionary optimization. In: Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2003, pp. 663–670 (2003)
7. Elsayed, S.M., Ray, T., Sarker, R.A.: A surrogate-assisted differential evolution algorithm with dynamic parameters selection for solving expensive optimization problems. In: Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2014, pp. 1062–1068 (2014)
8. Cruz-Vega, I., Garcia-Limon, M., Escalante, H.J.: Adaptive-surrogate based on a neuro-fuzzy network and granular computing. In: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2014, pp. 761–768 (2014)
9. Verbeeck, D., Maes, F., Grave, K.D., Blockeel, H.: Multi-objective optimization with surrogate trees. In: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2013, pp. 679–686 (2013)

10. Martínez, S.Z., Coello, C.A.C.: A multi-objective meta-model assisted memetic algorithm with non gradient-based local search. In: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2010, pp. 537–538 (2010)
11. Loshchilov, I., Schoenauer, M., Sebag, M.: A mono surrogate for multiobjective optimization. In: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2010, pp. 471–478 (2010)
12. Rosales-Pérez, A., Coello, C.A.C., Gonzalez, J.A., García, C.A.R., Escalante, H.J.: A hybrid surrogate-based approach for evolutionary multi-objective optimization. In: Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2013, pp. 2548–2555 (2013)
13. Kruisselbrink, J.W., Emmerich, M.T.M., Deutz, A.H., Bäck, T.: A robust optimization approach using kriging metamodels for robustness approximation in the CMA-ES. In: Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2010, pp. 1–8 (2010)
14. Bouzarkouna, Z., Auger, A., Ding, D.Y.: Local-meta-model CMA-ES for partially separable functions. In: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2011, pp. 869–876 (2011)
15. Liao, Q., Zhou, A., Zhang, G.: A locally weighted metamodel for pre-selection in evolutionary optimization. In: Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2014, pp. 2483–2490 (2014)
16. Loshchilov, I., Schoenauer, M., Sebag, M.: Self-adaptive surrogate-assisted covariance matrix adaptation evolution strategy. In: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2012, pp. 321–328 (2012)
17. Loshchilov, I., Schoenauer, M., Sebag, M.: Intensive surrogate model exploitation in self-adaptive surrogate-assisted cma-es (saacm-es). In: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2013, pp. 439–446 (2013)
18. Preuss, M., Rudolph, G., Wessing, S.: Tuning optimization algorithms for real-world problems by means of surrogate modeling. In: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2010, pp. 401–408 (2010)
19. Kramer, O., Schlachter, U., Spreckels, V.: An adaptive penalty function with meta-modeling for constrained problems. In: Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2013, pp. 1350–1354 (2013)