Resampling Methods for Meta-Model Validation with Recommendations for Evolutionary Computation

B. Bischl[*] bischl@statistik.tu-dortmund.de
Faculty of Statistics, TU Dortmund University, Germany

O. Mersmann olafm@statistik.tu-dortmund.de
Faculty of Statistics, TU Dortmund University, Germany

H. Trautmann trautmann@statistik.tu-dortmund.de
Faculty of Statistics, TU Dortmund University, Germany

C. Weihs weihs@statistik.tu-dortmund.de
Faculty of Statistics, TU Dortmund University, Germany

Abstract

Meta-modeling has become a crucial tool in solving expensive optimization problems. Much of the work in the past has focused on finding a good regression method to model the fitness function. Examples include classical linear regression, splines, neural networks, Kriging and support vector regression. This paper specifically draws attention to the fact that assessing model accuracy is a crucial aspect in the meta-modeling framework. Resampling strategies such as cross-validation, subsampling, bootstrapping, and nested resampling are prominent methods for model validation and are systematically discussed with respect to possible pitfalls, shortcomings, and specific features. A survey of meta-modeling techniques within evolutionary optimization is provided. In addition, practical examples illustrating some of the pitfalls associated with model selection and performance assessment are presented. Finally, recommendations are given for choosing a model validation technique for a particular setting.

Keywords

Resampling, meta-models, model validation, regression, evolutionary optimization, evolutionary computation.

1 Introduction

Many real-world continuous optimization problems are of a complex nature, which often requires expensive fitness function evaluations or computationally intensive simulations in case the functions cannot be provided explicitly in analytical form (black box optimization). Approximations of such functions, often called meta-models or surrogate models, are an important means to reduce the computational effort by fitting a regression function based on a number of evaluated design points, in particular when the required number of explicit fitness evaluations should be kept as small as possible.

Most prominent meta-modeling approaches are polynomial models as used in response surface analysis (Myers and Montgomery, 1995), Kriging together with the

[*]Corresponding Author.

expected improvement approach (e.g., Huang et al., 2006), for example, as used for algorithm parameter tuning in the well-known framework of sequential parameter optimization (SPO; Bartz-Beielstein, 2006), neural networks (e.g., Ong et al., 2003), or support vector regression (e.g., Yun et al., 2009; Bhattacharya, 2008).

A key aspect, however, is that a meta-model will always only approximate the original problem (Y. Jin and Branke, 2005), which possibly leads to bias, wide confidence bands, and also error in the obtained optimum. These are properties that cannot be neglected in evaluating a modeling approach. Resampling methods can be efficiently used for the following purposes.

Accuracy Assessment of a Given Meta-Model. A meta-model without sufficient accuracy can have detrimental effects on the optimization process and biased results might be generated which will not have any relevance for the problem at hand. This stresses the importance of a systematic model validation which indicates the reliability and accuracy of the estimated model. In case the current meta-model does not fulfill the requirements, possible alternatives are either a model update based on additional values obtained by more fitness function evaluations, or using a different, possibly less complex, modeling technique.

Model Selection. In many cases, several model classes are candidates for fitting the desired meta-model. Resampling methods and the related accuracy assessment efficiently support the selection process of the most appropriate and reliable model type. It should be noted that it is usually advisable to choose a less complex model in order to achieve good results with the desired small sample sizes, since more complex models usually require larger datasets to be sufficiently accurate.

Tuning of Hyperparameters. Most modeling strategies require the setting of so-called hyperparameters (e.g., the parameters of the covariance kernel in Kriging). Thus, a tuning of the hyperparameters is desired to determine the settings leading to highest model quality which can be evaluated using the discussed resampling techniques.

In the following, model quality is solely reflected by model accuracy, which in our view is the most relevant aspect, although other aspects of model quality could also be relevant in some settings. For example, in the early stages of an optimization process, a smooth model might be considered more appropriate than a rough model since it will provide sufficient information to drive the optimization process and simultaneously diminishes the probability of getting stuck in local optima. However, smoothing an accurate but rough model provides this possibility as well and can at the same time, in its accurate form, also be used at the end of the optimization process. It should also be taken into account that the ability to interpret a meta-model is often seen as an important benefit that other optimization techniques lack (e.g., see Smit and Eiben, 2009). A model that is merely helpful in controlling the optimization but does not approximate the true structure of the data generating process well enough might therefore mislead a researcher to draw wrong conclusions in a later stage of analysis.

Therefore, this paper focuses on stressing the importance of evaluating model accuracy in meta-model optimization together with providing related practical guidelines and recommendations. We thoroughly discuss state of the art approaches with respect to their advantages and shortcomings. While we cannot provide a comprehensive model validation approach covering all optimization-specific aspects in general, we give an overview of how model validation and evolutionary computing have been combined in Section 5.

An overview of the most important resampling techniques for model validation such as cross-validation, bootstrapping, subsampling, and nested resampling as a combination of the former approaches is available in Sections 2 and 3. Guidelines are presented for choosing an adequate resampling technique suitable for small sample sizes resulting from a small number of fitness evaluations. In Section 4, specific attention is drawn to a discussion of potential pitfalls and important assumptions that have to be fulfilled. A special focus on model validation in the evolutionary optimization field is provided in Section 5, supplemented by illustrative examples. Conclusions are drawn in Section 7. All of these methods and most of the comments are generally applicable for any regression or classification setting in machine learning, and for comparison, some remarks are made regarding the latter.

2 Basic Resampling Techniques and Statistical Properties

Assume we are given a fitness function f, a set of points $D = \{\mathbf{d}_1, \ldots, \mathbf{d}_n\}$ with $\mathbf{d}_i = (\mathbf{x}_i\ y_i)'$, decision variables or parameters $\mathbf{x}_i \in \mathbb{R}^m$ and associated function values $y_i = f(\mathbf{x}_i) \in \mathbb{R}$. Our aim now is to find a meta-model which approximates f using the information contained in D, that is, we want to fit a regression function $\hat{f}_D$ to our data D so that $\hat{y} = \hat{f}_D(\mathbf{x}, \varepsilon)$ with minimum approximation error ε.

Now the question arises, how well our model $\hat{f}_D$ approximates the real fitness function f. This requires the definition of an appropriate loss function $L(y, \hat{f}(\mathbf{x}))$. For regression this will be the squared loss $(y - \hat{f}(\mathbf{x}))^2$ or the absolute loss $|y - \hat{f}(\mathbf{x})|$, if robustness is required with respect to outliers. These loss values are aggregated by the mean to form the mean squared error (MSE) and the mean absolute error (MAE).

For modeling fitness functions in optimization, one might also be interested in estimators, which do not measure how close the regression model is to the true fitness function, but whether it preserves the ranking of the data points w.r.t. their fitness. A severely biased regression model that exhibits a high MSE might still properly predict the ranking of the data points. Such a model can still be beneficial for the optimization algorithm, if only rank-based information is required. Note that the last statement might not be completely true, if the regression model should also be interpreted. Then we might require a reasonably accurate model by all means. A simple way to measure the similarity of two rankings (e.g., of the true and the predicted one) is Spearman's rank correlation coefficient, also called Spearman's rho, which calculates the correlation between the two rankings. Its negative value can be interpreted as a rank-based error measure.

Given a loss function, we can now define the risk associated with our model of the fitness function

$$R(f, p) = \int_{\mathbb{R}} \int_{\mathbb{R}^m} L(y, f(\mathbf{x})) p(\mathbf{x}, y) d\mathbf{x}\, dy.$$

Here $p(\mathbf{x}, y)$ is the joint probability density function (PDF) of decision space and function values. Since models are usually based on data, we can plug in the estimator $\hat{f}_D$ of f into the definition of the risk to obtain

$$GE(\hat{f}_D, p) = \int_{\mathbb{R}} \int_{\mathbb{R}^m} L(y, \hat{f}_D(\mathbf{x})) p(\mathbf{x}, y)\, d\mathbf{x}\, dy \tag{1}$$

which is usually called the generalization error or conditional risk associated with the predictor. Note that the generalization error now also depends on the data used to fit the

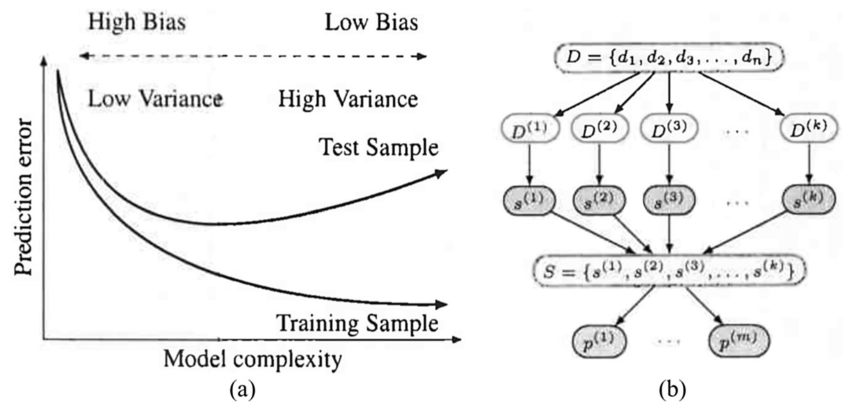

Figure 1: (a) With increasing model complexity, overfitting on the training set becomes more likely (Hastie et al., 2001). (b) Generic resampling scheme. A model validation function takes a training set ($D^{(i)}$) and an accompanying test set ($D \setminus D^{(i)}$). Different generators for the data subsets $D^{(i)}$ are cross-validation, bootstrapping, and subsampling. S equals the set of aggregated loss function values which are again aggregated by performance measures $p^{(i)}$.

model. First, let us estimate the generalization error of a model. Since the underlying distribution P of the data is neither known nor can it be efficiently estimated, it is replaced with a test dataset D^*. We then get

$$\widehat{GE}(\hat{f}_D, D^*) = \sum_{(\mathbf{x}\ y)' \in D^*} \frac{L(y, \hat{f}_D(\mathbf{x}))}{|D^*|}$$

If we plug in the dataset D used to fit the model into the previous definition, we obtain the so-called resubstitution error

$$\widehat{GE}_{\text{resub}} = \widehat{GE}(\hat{f}_D, D).$$

Here we use the training data both to learn the predictor and estimate the generalization error. This leads to a biased estimate of the generalization error which, if we use this estimate of the generalization error for model selection, will favor more adaptive, complex models in an unintended way. In fact, for some models, the resubstitution error will always be zero, for example, imagine the case of an interpolating spline. This general problem is illustrated in Figure 1(a): The more complex and adaptive a model is, the lower its bias is, meaning it is more able to accommodate for the shape of the underlying function which should be modeled. But these models might overfit the training data, leading to a low resubstitution error, although they are not necessarily able to predict future data with the desired accuracy, visualized in Figure 1(a) as the growing gap between training and test error. In conclusion, we cannot use the performance on the training data if we are interested in an honest estimator of the true performance of a model.

One way to rectify this is to split the data into a training set D_{train} and a corresponding test set D_{test} such that $D_{\text{train}} \cup D_{\text{test}} = D$ and $D_{\text{train}} \cap D_{\text{test}} = \emptyset$. One would then train the model on D_{train} to obtain $\hat{f}_{D_{\text{train}}}$ and calculate its generalization error using the data points in D_{test}. This approach is called the hold-out or test set method, which is trivial to implement and work with from a statistical perspective, as the test set observations are completely independent from the training observations. The estimate of the generalization error is then given by

$$\widehat{GE}_{\text{hold-out}} = \widehat{GE}(f_{D_{\text{train}}}, D_{\text{test}}).$$

Two important problems remain: A large dataset D is required, since we need enough data in the training set to build an adequate model, but also enough samples in the test set to get statistically valid performance results. Such a large sample size will usually not be available if evaluating f is expensive. Also, one will not detect variance and model instability due to changes in the training set. Some models, especially more complex, nonlinear ones, might produce very different results even when the training data change only slightly. Note that although we use the term model complexity informally in this text (where a linear model might be considered to be of lower complexity, compared to, e.g., a neural network with a high number of hidden units), different rigorous definitions of this notion exist, among them VC dimension and Rademacher complexity (Bartlett et al., 2002).

To deal with this situation, resampling techniques have been developed. All of these techniques repeatedly generate training and test sets from the dataset at hand, fit a model to each training set, and judge its quality on the corresponding test set (and possibly also on the training set). This general framework is depicted in Figure 1(b). In the next three subsections we will introduce three different methods to generate these training/test pairs and will show how to estimate performance values. The estimate of the generalization error then becomes

$$\widehat{GE}_{\text{samp}} = \frac{1}{k}\sum_{i=1}^{k} \widehat{GE}(f_{D^{(i)}}, D \setminus D^{(i)}). \tag{2}$$

Note that the value of the generalization error estimate given in Equation (1) depends upon both the size of the dataset used to fit the model and the size of the test set. Therefore, we need to make sure that if we want to assess the generalization error for a model fit on n data points, the training set used in estimating the model should not be much larger or smaller than n. If, for example, the training set is much smaller than the complete dataset, we will usually pessimistically overestimate the error of a model fitted on all available data, as less information is used during the training while calculating the estimator. Keeping this in mind, it is clear that the quality of the generalization error estimate in Equation (2) obtained by a resampling strategy largely depends upon how the two datasets are chosen. Important factors influencing the quality are the size of the two sets compared to the original dataset D, the number of subsamples k drawn, and the dependence structure between the subsamples $D^{(i)}$. The error of the estimator (with respect to a fixed training set size of n) is usually measured by its mean squared error (MSE) regarding the true performance value:

$$\text{MSE}(\widehat{GE}_{\text{samp}}) = \underset{|D|=n}{\text{E}}((\widehat{GE}_{\text{samp}} - GE(\hat{f}_D, P))^2)$$

The MSE of an estimator can equivalently be expressed as the sum of its squared bias and variance:

$$\text{MSE}(\widehat{GE}_{\text{samp}}) = \text{Bias}(\widehat{GE}_{\text{samp}})^2 + \text{Variance}(\widehat{GE}_{\text{samp}})$$

These two quantities are defined as follows:

$$\text{Bias}(\widehat{GE}_{\text{samp}}) = \text{E}\left[\widehat{GE}_{\text{samp}}\right] - \underset{|D|=n}{\text{E}}\left[GE(\hat{f}_D, p)\right]$$

Algorithm 1 Generic resampling

```
input: A dataset $D$ of $n$ observations $d_1$ to $d_n$, the number of subsets $k$ to generate and a loss function $L$.
output: Summary of the validation statistics.
1 Generate $k$ subsets of $D$ named $D^{(1)}$ to $D^{(k)}$
2 $S \leftarrow \emptyset$
3 for $i \leftarrow 1$ to $k$ do
4     $\bar{D}^{(i)} \leftarrow D \setminus D^{(i)}$
5     $\hat{f} \leftarrow$ FitModel ($D^{(i)}$)
6     $s_i \leftarrow \sum_{(\mathbf{x}\ y)' \in \bar{D}^{(i)}} L(y, \hat{f}(\mathbf{x}))$
7     $S \leftarrow S \cup \{s_i\}$
8 Summarize $S$, for example, $\mathrm{mean}(S)$, $\mathrm{median}(S)$
```

Algorithm 2 Subsets for k-fold CV

```
input: A dataset $D$ of $n$ observations $d_1$ to $d_n$ and the number of subsets $k$ to generate.
output: $k$ subsets of $D$ named $D^{(1)}$ to $D^{(k)}$.
1 $D \leftarrow$ Shuffle ($D$)
2 for $i \leftarrow 1$ to $k$ do
3     $D^{(i)} \leftarrow D$
4 for $j \leftarrow 1$ to $n$ do
5     $i \leftarrow (j \bmod k) + 1$
6     $D^{(i)} \leftarrow D^{(i)} \setminus \{d_j\}$
```

and

$$\text{Variance}(\widehat{GE}_{\text{samp}}) = \mathrm{E}\left[\left(\widehat{GE}_{\text{samp}} - \mathrm{E}\left[\widehat{GE}_{\text{samp}}\right]\right)^2\right]$$

The bias expresses the average difference between an estimator and the true value, while the variance measures the average spread of the estimator. Note that we have now defined an MSE (or bias or variance) of an estimator as well as the MSE (or bias or variance) of a regression model. It should be clear from the context which of the two is meant.

2.1 Cross-Validation

Cross-validation (CV; Stone, 1974) is probably one of the oldest resampling techniques. Like all other methods presented in this paper it uses the generic resampling strategy as described in Algorithm 1. The k subsets (line 1 of Algorithm 1) are generated according to Algorithm 2. The idea is to divide the data set into k equally sized blocks and then use $k-1$ blocks to fit the model and validate it on the remaining block. This is done for all possible combinations of $k-1$ of the k blocks. The k blocks are usually called folds in the cross-validation literature. Such a cross-validation with $k = 10$ would be called a 10-fold cross-validation. Usual choices for k are 5, 10, and n.

This last case of $k = n$ is also referred to as leave-one-out cross-validation (LOO-CV) because the model is fitted on the subsets of D which arise if we leave out exactly one observation. In general, this is computationally hard, but for certain classes of models, the LOO-CV estimate can be calculated efficiently (Fast LOO-CV). Examples include linear regression (Wu, 1986), splines (Wahba, 1980), support vector regression (Cawley and Talbot, 2004), and Kriging models (Sundararajan and Keerthi, 2001).

Algorithm 3 Subsets for bootstrap

```
input: A dataset $D$ of $n$ observations $\mathbf{d}_1$ to $\mathbf{d}_n$ and the number of subsets $k$ to generate.
output: $k$ subsets of $D$ named $D^{(1)}$ to $D^{(k)}$.
for $i \leftarrow 1$ to $k$ do
    $D^{(i)} \leftarrow \emptyset$
    for $j \leftarrow 1$ to $n$ do
        $d \leftarrow$ RandomElement ($D$)
        $D^{(i)} \leftarrow D^{(i)} \cup \{\mathbf{d}\}$
```

2.2 Bootstrap

The development of the bootstrap resampling strategy (Efron, 1979) is almost as old as the idea of cross-validation. Again, Algorithm 1 is the basis of the method, but the k subsets are generated using Algorithm 3. In this case, k is usually chosen much larger than for CV. Values of $k = 100$ up to $k = 1000$ are not uncommon and there is no relevant upper limit[1] on k.

The subset generation is based on the idea that instead of sampling from D without replacement, as is done in the CV case, we sample with replacement. One of the advantages of this approach is that the size of the training set, in the bootstrap literature often also called the "in bag observations," is equal to the actual dataset size. On the other hand, this entails that some observations can and likely will be present multiple times in the training set $D^{(i)}$. In fact, asymptotically, only about $1 - e^{-1} \approx 63.2\%$ of the data points in D will be present in the training set (Efron, 1979). The remaining $\approx$36.8% of observations are called "out of bag," and form the test set as in CV.

The fact that some observations are present multiple times in the training set can be problematic for some meta-modeling techniques. Several approaches have been proposed to deal with this. Most add a small amount of random noise to the observations. For details see Efron (1979).

Another effect of adding some observations multiple times to the training set is that we overemphasize their importance, which is called oversampling. This leads to an estimation bias for our validation statistic since all data points in the test set are now, in some sense, outliers. A first attempt to counter this was the so called .632 bootstrap procedure by Efron (1983). Here the estimated error of the model is a weighted average of the error on the training set and the test set. The fallacy in this approach is that some modeling techniques will always have an error of 0 on the training set. An example of such a method would be an interpolating spline.

To counter this, Efron proposed a further variant of the bootstrap named the .632+ bootstrap (see Efron and Tibshirani, 1997). This strategy is a bit more involved and deviates somewhat from the framework proposed in Algorithm 1. The details are given in Algorithm 4. The main difference here is that instead of fixed weights, as in the .632 bootstrap, the weights are calculated for each model to reflect how well the model can reproduce the training set.[2]

[1]Note that there are n^n different bootstrap samples. So for very small n, there are limits to the number of bootstrap samples you can generate.

[2]Note that we use the definition for the resubstitution error given in Efron (1983) here, which, for $k \to \infty$, is equivalent to the definition used in Efron and Tibshirani (1997; as they also point out). The reason for using this definition is that it can be implemented much more space efficiently.

Algorithm 4 .632+bootstrap

```
input : A dataset $D$ of $n$ observations $d_1$ to $d_n$, the number of subsets $k$ to generate, and a loss function $L$.
output: $k$ values of the validation statistic.
1  Generate $k$ subsets of $D$ named $D^{(1)}$ to $D^{(k)}$
2  $S \leftarrow \emptyset$
3  for $i \leftarrow 1$ to $k$ do
4      $\bar{D}^{(i)} \leftarrow D \setminus D^{(i)}$
5      $\hat{f} \leftarrow \texttt{FitModel}(D^{(i)})$
6      $\hat{\gamma} \leftarrow \frac{1}{n^2} \sum_{p,q=1}^{n} L(y_p, \hat{f}(\mathbf{x}_q))$
7      $s^{\text{in}} \leftarrow \frac{1}{|D^{(i)}|} \sum_{(x,y) \in D^{(i)}} L(y, \hat{f}(\mathbf{x}))$
8      $s^{\text{out}} \leftarrow \frac{1}{|\bar{D}^{(i)}|} \sum_{(x,y) \in \bar{D}^{(i)}} L(y, \hat{f}(\mathbf{x}))$
9      $\hat{R} \leftarrow \frac{s^{\text{out}} - s^{\text{in}}}{\hat{\gamma} - s^{\text{in}}}$
10     $\hat{w} \leftarrow \frac{0.632}{1 - 0.368\hat{R}}$
11     $s_i \leftarrow (1 - \hat{w})s_i^{\text{in}} + \hat{w}s_i^{\text{out}}$
12     $S \leftarrow S \cup \{s_i\}$
```

The k subsets of D are again generated using Algorithm 3. Then, as in the general framework, the model $\hat{f}$ is calculated in line 5. Lines 6 to 11 are new. In line 6 the loss is estimated for the hypothetical case that our model has no predictive power. This is done by calculating the loss for each possible combination of x and y from D. Because now there is no direct dependence between x and y, the loss one observes tells us something about the error rate the model would achieve even if there were no link between x and y, that is, if our function were pure noise. This quantity $\hat{\gamma}$ is called the "no-information error rate." Using this and the in-bag error rate from line 7, as well as the usual out-of-bag error rate from line 8, the relative overfitting rate $\hat{R}$ is calculated in line 9. $\hat{R}$ lies between 0 and 1. If $\hat{R} = 1$, then the model is completely overfitted, that is, it only has predictive power on the training set. If $\hat{R}$ is almost 0 on the other hand, then $\hat{f}$ has great predictive power on the test set and we can use the error on the training set to increase the accuracy of our error estimate. The final performance criterion is the weighted average of the in-bag and out-of-bag error rates as calculated in line 11 using the weight derived in line 10 from the relative overfitting rate.

2.3 Subsampling

Subsampling (SS) is very similar to the classical bootstrap. The only difference is that observations are drawn from D without replacement. Therefore, the training set has to be smaller than D or no observations would remain for the test set. The usual choices for the subsampling rate $|D^{(i)}|/|D|$ are 4/5 or 9/10. This corresponds to the usual number of folds in cross-validation ($k = 5$ or $k = 10$). As in bootstrapping, k has to be selected a priori by the user. Choices for k are also similar to bootstrapping, for example, in the range of 100 to 1000.

2.4 Further Methods

Many variants, extensions, and combinations of the above algorithms exist, which we cannot present as comprehensively as the standard methods presented in the previous section. Some examples of these methods are as follows.

Algorithm 5 Subset generation algorithm for subsampling

```
input: A dataset $D$ of $n$ observations $d_1$ to $d_n$, the number of subsets $k$ to generate, and the subsampling
       rate $r$.
output: $k$ subsets of $D$ named $D^{(1)}$ to $D^{(k)}$.
1 $m \leftarrow \lfloor r|D| \rfloor$
2 for $i \leftarrow 1$ to $k$ do
3     $D' \leftarrow D$
4     $D^{(i)} \leftarrow \emptyset$
5     for $j \leftarrow 1$ to $m$ do
6         $d \leftarrow$ RandomElement ($D'$)
7         $D^{(i)} \leftarrow D^{(i)} \cup \{d\}$
8         $D' \leftarrow D' \setminus \{d\}$
```

Repeated Cross-Validation (REPCV). The REPCV performs a usual *k*-fold CV multiple times and aggregates the results (normally by the mean) to reduce the variance of randomly splitting the data.

Bootstrap Cross-Validation (BCV). The BCV generates *b* bootstrap samples of D and performs cross-validation on each bootstrap sample, then aggregates the resulting error rates by averaging them. This has been shown to be advantageous for small sample sizes. For a detailed description, see Fu et al. (2005) and Hothorn et al. (2005), but note comment 10 in Section 4 and the variation proposed in Hothorn et al. (2005), where those entries in the test set are removed, which also occurs in the training set.

Stratified Cross-Validation (SCV). The SCV tries to ensure that the data distribution of each single fold matches the distribution of the complete dataset most closely. While in classification mainly the sizes of the classes are controlled in SCV, in regression it is either ensured that the mean response value is roughly equal in all partitions or that the distribution of the input data in each fold is as close as possible to the distribution of the input data of the complete dataset. For the latter, clustering algorithms are often employed (see Diamantidis et al., 2000, for one of the possible approaches). Such a technique can be helpful to reduce the bias of an accuracy estimator when only a small number of data samples are available.

3 Nested Resampling

Often, the process of deriving a model requires a more complex approach (including model selection) than simply fitting a regression model. This will be the case when hyperparameters (e.g., consider support vector regression) or covariates have to be selected (e.g., screening the covariates for the ones which have a relevant influence on the target value y). Since the optimal hyperparameters are often data-dependent, they cannot be chosen without looking at the data. On the other hand, one cannot perform these model selection steps using the same resampling sets that are used for evaluating the model itself, as this can lead to severely biased performance results (see comments 4 and 5 in Section 4). This effect is sometimes called "training on the test set," since through the repeated evaluation of the model on the test data, information regarding its structure enters the model, which might lead to overfitting.

Another explanation of why overfitting will occur is illustrated by the following thought experiment: Imagine a model which randomly assigns predictions without using the covariates and with a hyperparameter σ which has no influence on the quality of the fit. “Optimizing” σ over a large number of iterations now corresponds to choosing the minimum statistic of the error values of all generated random models. This minimum will be lower the more iterations we perform. It will, however, not be a reasonable estimator regarding the performance on new data (which would be the mean of all observed errors or expected error rate on an unused test set).

Instead, the model selection process has to be treated as part of the fitting and therefore has to be repeated for every training set. Unfortunately, as this kind of model selection usually requires a resampling approach as well, one ends up with a nested process of sampling. While the model is evaluated in an outer loop, every training set is resampled again in an inner loop to select an appropriate model; for example, see Simon (2007). As an example, consider using subsampling with $k = 100$ for evaluation and fivefold cross-validation for hyperparameter selection. For each of the 100 training sets $D^{(i)}$ from subsampling, a fivefold cross-validation on the training set is employed as the internal fitness evaluation to select the best setting for the hyperparameters of the regression function. This is an optimization problem itself, but we are not going to cover the differences between the usual approaches here. Typically, optimization methods are used for which no parameters have to be specified a priori in order to avoid another level of hyperparameter optimization. The resulting optimization quality of these approaches usually is sufficient for this purpose. The best obtained hyperparameters are used to fit the model on the complete training set and calculate the error on the test set of the outer resampling strategy. Figure 2 shows this process schematically for two nested cross-validations.

While this course of action is computationally very expensive, it is currently the only known way to ensure unbiased results. If runtime has to be reduced, more efficient optimization or a coarser resampling for model selection in the inner loop should be employed. If the latter produces a negative effect for model selection, at least this will be noticeable in a valid way in the outer evaluation.

4 Common Pitfalls, Recommendations, and Statistical Properties

In practical applications, choosing the right resampling method and experimental setup in order to evaluate estimated models might become a tough decision—even for an experienced data analyst. We will describe the most common misconceptions and pitfalls in this section and try to give reasonable recommendations and some theoretical results relevant for applied work. There are a number of other papers with a similar goal, often from the field of classification, which we build upon. See, for example, Good (2005), Kohavi (1995), Simon (2007), Molinaro et al. (2005), Kim (2009), Weihs (1993) and, for more general recommendations, Salzberg (1997).

1 **Terminology.** Make sure to check formal definitions of resampling methods in papers. Unfortunately, even in the ones cited in this paper, different names are often used and terminologies are sometimes even interchanged. For example, in Binder and Schumacher (2008), what we call out-of-bag bootstrap is called bootstrap cross-validation (and therefore used differently than what we defined as BCV) and in Nadeau and Bengio (2003), what we call subsampling is called cross-validation. Other terms for subsampling also include Monte Carlo cross-validation (Shao, 1993), and repeated hold-out (Kim, 2009).

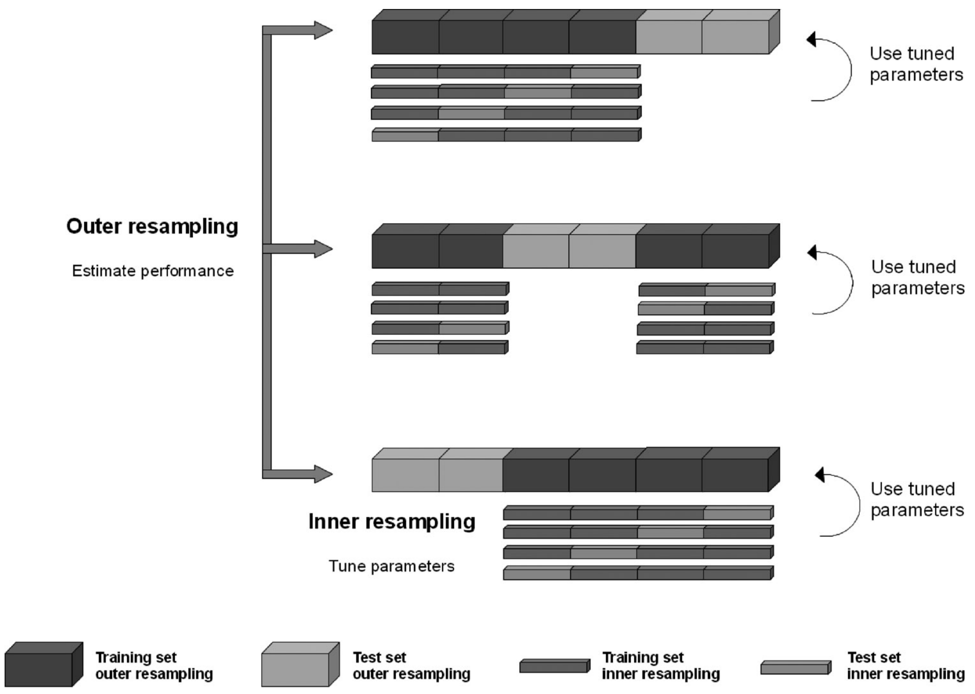

Figure 2: Nested resampling with two nested cross-validations.

2 **Stochastic Performance Values Are Not a Disadvantage.** We have encountered a specific preference toward the leave-one-out estimator among nonstatisticians "because it does not produce results subject to stochastic variation," but instead delivers a reproducible, deterministic value. This is a misconception which stems from the notion that your given dataset completely determines your task. On the contrary, the data were obtained by a stochastic (real-world or simulated) process itself. The task is not to optimize a machine learning model w.r.t. this dataset, but to use the data efficiently to minimize the loss on new, unseen observations. This is and will always be a stochastic optimization problem. Information about uncertainty and statistical variation due to slight changes in the data and unstable models are an advantage and not a drawback.

3 **Be Precise When Stating How Model Selection and Evaluation Were Performed.** As the interpretation of results relies strongly on the chosen experimental setup, the necessity of this statement should be self-evident. Still, some authors seem to treat this issue as a "minor detail, only worth brief mentioning in passing." Also, when code is not published along with an article, reproducibility of results sometimes becomes difficult if not impossible.

4 **Do Not Report Performances from Repeated Tuning on the Same Resampling.** Even though this problem and its remedy have already been discussed in Section 3, we emphasize this common error here again, as from our experience this still repeatedly turns up in applied work. Disregarding this makes interpretations and comparisons to other results nearly impossible, as one can never be

sure how much positive results are due to the “training on the test set” effect. Quite a number of publications had to be rectified because of this mistake, see Fiebrink and Fujinaga (2006) for a recent example. Use nested resampling.

5 **Include all Modeling Steps in the Resampling.** If data-dependent preprocessing or preselection of variables is performed, these have to be included in the resampling as well to avoid biased, optimistic results. Although not doing so is basically the same mistake as the one described in the comment above, this happens more frequently as preprocessing is usually considered independently of the model fitting and performed once in the beginning of the analysis. Simon et al. (2003) demonstrate how the reevaluation of experiments from a published paper, in which covariates were excluded by a statistical test in preprocessing, leads to much worse results than would have been the case if the selection were included in a complete resampling. They also confirm this conclusion by further simulation studies regarding the selection of covariates.

6 **Do Not Use Hold-Out, CV with Few Iterations, or Subsampling with a Low Subsampling Rate for Small Samples.** These methods have a large bias resulting from the reduced training set size (and a larger variance due to the one or only few random split-ups, except for subsampling), which is especially detrimental when data are already scarce.

7 **Comments Regarding Leave-One-Out and Cross-Validation.** Leave-one-out cross-validation has better properties for the squared loss used in regression than for its usual 0-1 counterpart in classification and is an almost unbiased estimator for the mean loss (Kohavi, 1995). It can be shown that it is asymptotically equivalent to the well-known AI criterion (AIC), the jackknife, and the bootstrap (Stone, 1977; Efron, 1983). Although these asymptotic equivalences are important and interesting in their own right, we want to point out that many practically relevant differences exist between the mentioned methods, and some are discussed in the comments in this section. The many possibilities for its efficient computation mentioned in Section 2.1 and its deterministic, reproducible value (but see comment 2) and its near unbiasedness make LOO-CV an attractive candidate among the presented algorithms, especially when only a few samples are available. But one should be aware of the following facts: LOO-CV has a high variance (Kohavi, 1995; Weiss and Kulikowski, 1991) as an estimator of the mean loss, meaning quite different values are produced if the data used for cross-validation slightly change. It also is asymptotically inconsistent and tends to select too complex models. In Shao (1993) theoretical reasons for this effect are presented, and subsampling and balanced leave-d-out CV are shown to be superior estimators in a simulation study. Kohavi (1995) arrives at similar results regarding LOO-CV and demonstrates empirically that 10-fold CV is often superior. He suggests a stratified version. For these reasons, we recommend LOO-CV mainly for efficient model selection, still keeping in mind that this might lead to somewhat suboptimal choices. Repeated and stratified CV will usually produce more reliable results in practice.

8 **Comments on Different Bootstrap Variants.** Many variations on the algorithms presented in Section 2.2 exist. We will here restrict ourselves to comparing the simple out-of-bag bootstrap, using only s^{out}, the older .632, which always sets

$\hat{w} = 0.632$, and the .632+. While the out-of-bag bootstrap is pessimistically biased in the sense that it bases its performance values on models which use only about 63.2% of the data, .632 can be optimistic in a much worse way, as complex models can easily achieve $s^{\text{in}} = 0$ (or consider adding a memory of the training data to your model). Both estimators are known to have a low variance, and out-of-bag is especially good when the sample size is small and the error or noise in the data is high (Weiss and Kulikowski, 1991). The .632+ combines the best properties of both estimators and can generally be trusted to achieve very good results with small sample sizes. Its main drawback is the difficult combination with tuning (see comment 10) and that it might result in an optimistic bias when more complex models are considered (Kim, 2009; Molinaro et al., 2005).

9 **Choosing the Number of Iterations in Bootstrapping, Subsampling, and others.** These questions seem to remain an open research issue, and the general consensus seems to be, to err on the safe side by choosing a large k. For example, 100, 250, and 500 are common values in published papers, and Weiss and Kulikowski (1991) recommend at least 200 to obtain a good estimate. But a reasonable setting always has to be data-dependent, relying on the number of available samples, the complexity of the task, and applied models. One should be aware that this will be computationally very expensive. There exist a couple of heuristics to select the number of iterations adaptively, especially in model selection and optimization. For example, consider the repeated testing performed in F-Races (Birattari et al., 2002). But for valid statistical inference, one will have to employ methods from sequential analysis. To our knowledge, this is not very often done in machine learning, and a comprehensive overview of these techniques is beyond the scope of this paper. We refer the reader to Mukhopadhyay and de Silva (2009) for an accessible introduction.

10 **Bootstrapping or Subsampling? Repeated Observations Can Be Problematic.** When combining model or complexity parameter selection with bootstrapped datasets in the outer loop of nested resampling, repeated observations can lead to a substantial bias toward more complex models. This stems from the fact that in the inner tuning loop with a high probability, measurements will occur both in the training set as well as in the test set, so that more complex models “memorizing” the training data will seem preferable. This effect was documented in the setting of boosting for high-dimensional microarray data (Binder and Schumacher, 2008) and subsampling was proposed and evaluated as a remedy.

11 **Independence, Confidence Intervals, and Testing.** In general, the generated training and test sets, and therefore the obtained performance statistics, will not be independent when sampling from a finite dataset. This has negative consequences if confidence intervals for the performance measure should be calculated. The dependence structure is especially complicated for the commonly used cross-validation, where the split-up of the data in one iteration completely depends on all other split-ups. It can be shown that in this setting, no unbiased estimator of the variance exists (Bengio and Grandvalet, 2004) and pathological examples can be constructed where the variance estimator performs arbitrarily badly. Extensions on corrections for subsampling exist which take the dependence between sampled datasets into account and provide a much better foundation for interval estimators and subsequent statistical tests regarding location parameters (Nadeau

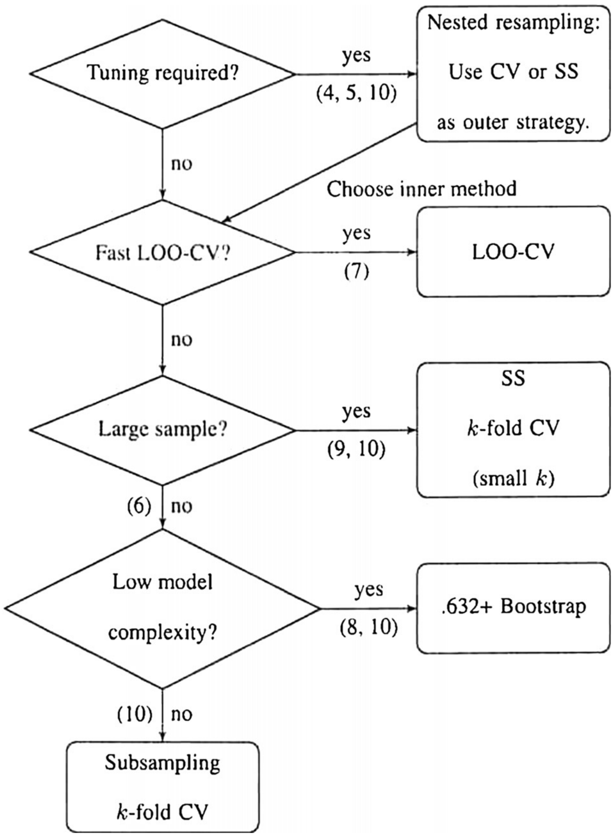

Figure 3: Guidelines for selecting a resampling strategy. The numbers below or to the left of the arrows are references to the common pitfalls discussed in Section 4.

and Bengio, 2003). The focus in this paper has largely been on model selection and not necessarily on model comparison. For this, several procedures using statistical hypothesis tests have been proposed by Dietterich (1998), Alpaydin (1999), Nadeau and Bengio (2003), and Hothorn et al. (2005).

12 **Software and Illustrating Examples.** While many machine learning toolboxes contain at least some of the mentioned resampling strategies, we would like to point out that our own *R* package *mlr* (Bischl, 2010) contains all of the methods mentioned in this paper, including generic nested resampling, and an interface to many regression, classification, and optimization algorithms in *R*.

Figure 3 is a short guideline to choosing a model validation strategy. Due to the complexity of the topic at hand, there are certainly cases where it does not apply. We do, however, feel that it provides good guidelines for someone not intimately familiar with the procedures to choose a method for validating the models.

5 Model Validation within Evolutionary Optimization Methods

A comprehensive overview of meta-modeling techniques with respect to evolutionary computation and multiobjective optimization in general is given in El-Beltagy et al. (1999), Y. Jin (2005), Tenne and Armfield (2008), Knowles and Nakayama (2008), Nakayama et al. (2009), Shi and Rasheed (2010), and Lim et al. (2010). A short overview is provided in the following. A detailed discussion of these methods, however, is beyond the scope of this paper.

Meta-modeling strategies can be implemented by constructing a global meta-model based on initial data points (offline data sampling), for example, as generated by an experimental design, and replacing the fitness function by a global meta-model for the whole optimization process. However, the success of this strategy strongly depends on the initial sample used for model estimation. In general, the meta-model should be used in combination with the original fitness function in order to control the convergence to the meta-model optimum which does not necessarily coincide with the optimum of the original fitness function. Different strategies for this so-called evolution control or model management exist (Y. Jin, 2005; Ulmer et al., 2003; Y. Jin et al., 2002). Interactive approaches combining the explicit function and the meta-model can be employed where local meta-models are estimated using progressively generated evaluations of the explicit function (e.g., Younis and Dong, 2010; Emmerich et al., 2006; Zhou et al., 2007; Büche et al., 2004).

The trade-off between curse and blessing of meta-model accuracy is addressed in Ong et al. (2006) and Lim et al. (2010). The first aspect refers to possibly biased optimization results due to high model uncertainties, whereas more inaccurate models, which simultaneously exhibit smoothing properties, could make it less likely that the optimization process gets trapped by a local optimum and provide an adequate search direction. This relates to the question discussed in Section 1 whether an optimization algorithm could profit from using a rougher meta-model in the early stages of the optimization process. Loshchilov et al. (2010) and Kern et al. (2006) incorporate rank-based meta-modeling techniques within the CMA-ES. The aim here is not primarily on reaching maximum model accuracy, but rather on preserving the rankings of the offspring population values. However, it is shown that despite good performance on unimodal functions, the algorithms fail to produce competitive results on multimodal and noisy functions.

In multiobjective evolutionary optimization, meta-models indeed play an important role as well. For example, in Giotis and Giannakoglou (1999) ANNs are used as single neural networks with multiple output neurons as meta-models within multi-objective airfoil optimization to overcome the difficulties stemming from extremely expensive fitness function evaluations. One major area as well is prescreening points based on their predicted values and confidence of these predictions (e.g., Emmerich et al., 2006). A comparative study of different approaches is given in Wagner et al. (2010). In addition, hierarchical models (Kampolis et al., 2007; Zhou et al., 2007; Lim et al., 2010) which pick up the meta-modeling approach and expand it to multiple models of different complexity are a straightforward approach and a matter of current research.

Unfortunately, so far, not much attention has been given to model accuracy assessment in the evolutionary computation field. The focus is rather on training a selected meta-model based on the so far generated explicit evaluations of the original function in the course of the algorithm. A validation on test data is not specifically considered, which will likely result in an overfitting of the—possibly local—meta-models. Strategies for sequentially updating the meta-model, once new exact fitness function values are generated, are discussed, including the decision between local (e.g., trust region strategies, Zhou et al., 2007) and global modeling approaches. Therefore, in most cases, additional exact function evaluations are used as a means for evaluating the meta-model accuracy and to decide whether an update of the meta-model is necessary or not.

To our best knowledge, the resampling methodology in this paper is rarely used to assess and improve meta-models in optimization. In Lim et al. (2010) a hold-out approach is presented in which the test data is used for validating the prediction quality of the meta-model. In fact, Y. Jin et al. (2000) state that "re-sampling during fitness evaluation . . . can reduce the influence of noise to a certain extent." On the other hand, leave-one-out cross validation is mentioned as a suitable method for validating a Kriging model (Simpson et al., 1998), but it is declared to be "uncertain how well this measure correlates with model accuracy," which is questionable. Santers et al. (2003) recommend to use leave-one-out cross-validation for adapting the correlation parameter θ of the isotropic Gaussian kernel of a Kriging meta-model.

We are only aware of one specific paper by Tenne and Armfield (2008) where a similar approach to the one proposed in this paper is followed and will compare to their work. The authors propose to use hold-out, 10-fold CV, LOO-CV and the .632 bootstrap to estimate the prediction error of a quadratic, a Kriging, and a radial basis function network model on three different test functions. They do not include the .632+ bootstrap or repeated CV in their study, although these estimators might be more appropriate for their small sample sizes (less than 100). They also optimize the hyperparameters of the anisotropic Kriging model on the full dataset in the beginning, which might lead to a bias in the following resampling caused by overfitting.

Otherwise, their findings agree with our comments above: they state that hold-out has a pessimistic bias, since less data are used for training, LOO-CV has a large variance, the 10-fold CV also has a large variance due to the random splitting of the few samples and .632 has a low variance and an optimistic bias. However, they do not provide references for these facts, which have also been observed by other authors in comparison studies in machine learning (Wu, 1986; Shao, 1993; Kohavi, 1995; Efron and Tibshirani, 1997; Molinaro et al., 2005; Simon, 2007; Kim, 2009). They report LOO-CV as the most suited but computationally expensive resampling method for model assessment, but without mentioning the many approaches for its efficient calculation (see Section 2.1).

Evolutionary optimization methods can benefit from meta-model accuracy assessment based on resampling methods. Model type selection is facilitated as the accuracy of all candidate meta-models can be evaluated at each desired stage of the optimization process. A fair model comparison becomes possible so that the most appropriate meta-model type can be selected. This could either be done in the initial optimization phase, using initial sample points; or it could be done sequentially during the course of the optimization process so that switching from one model type to another of higher accuracy can be considered. Model type selection can also be addressed with respect to the trade-off between model accuracy, computational complexity, and the difficulty for the optimization algorithm. For instance, optimizing a meta-model based on random forests will in general raise more problems than the optimization of a Kriging model. In a comparative study of different meta-modeling techniques (Y. Jin et al., 2000), it is explicitly stated that multiple metrics such as accuracy, robustness, efficiency, transparency, and conceptual simplicity should be used for meta-model assessment. In addition, it has to be ensured that no false optima are introduced by the meta-model (Y. Jin et al., 2000), which may generate more severe optimization bias than a lower model accuracy. An aspect often neglected is that some models are numerically unstable or have constraints on the types of data they accept. Our observation is that dealing with these issues can become a large burden for the practitioner. Therefore, we believe that this should also be considered when choosing candidate modeling strategies.

Many meta-model-based EAs rely on evolution control and thus on the principle of updating the estimated meta-model parameters once the meta-model fails to predict new points in the enriched training dataset with a sufficient accuracy. In order to be able to make a statistically sound evaluation of the model fit, a suitable resampling strategy for assessing the (local) model accuracy should be chosen. Additionally, uncertainty estimates of Kriging meta-model predictions can be used for prescreening promising candidate solutions for single objective and multi-objective EAs (Emmerich et al., 2006). By incorporating the confidence information of the model, the convergence to the Pareto optimal front is enforced and premature convergence to local optima can be overcome. Uncertainty estimates of different surrogate models are also required in Sanchez et al. (2008), where an optimal ensemble of kernel-based meta-models is desired by weighting the candidate surrogate models inversely proportional to the estimated local prediction variance. Finding robust solutions by means of EAs fits into this context as well. Especially in this case, additional samples of the fitness functions as well as uncertainty estimates are required, which would result in long algorithm runtimes in the case of expensive fitness function evaluations. By means of local meta-models, for which regression models are recommended in Paenke et al. (2006), the expected solution quality and variance can be estimated. Kruisselbrink et al. (2010) employ Kriging meta-models in combination with the CMA-ES in order to efficiently search for robust solutions.

The ability to interpret the parameters of a meta-model is another crucial aspect. For instance, in the sequential parameter optimization (SPO) framework used for tuning EA parameters, the resulting meta-model at the end of the tuning phase is consulted to get an idea about the influence and interaction of the EA parameters. This is only possible in the case where the meta-model is of sufficient accuracy, which can be evaluated by the presented methodology. By this means, a global error rate of the resulting meta-model can be computed, which can be referred to in order to justify the suitability of the SPO tuning process.

Furthermore, an accurate meta-model can be used for the analysis of the fitness landscape of the optimization problem at hand as the meta-model evaluations are cheap in comparison to evaluating the original fitness functions. Thus, generating a quite large number of sample points in a well defined and structured manner and the calculation of appropriate features for characterizing the landscape becomes possible. This approach of exploratory landscape analysis as contemplated in Mersmann et al. (2010, 2011) offers a possibility for selecting the most appropriate optimization algorithm with respect to the characteristics of the optimization problem considered.

Another difficulty lies in the strong relatedness of the design points generated during sequential optimization. While the assumption of statistical independence when sampling points from this design by resampling is still formally valid, we have to be careful with the resulting interpretation. The problem lies in the fact that after a sequential optimization, the design points will not be drawn regularly from the fitness function, but will likely cluster around the (local) optimum where the algorithm has converged. Each performance estimator we compute at this stage now measures a property of this local distribution of design points and not the fitness function as a whole. We offer the following comments with regard to this issue: (a) This problem does not arise by construction when the regression model is resampled based solely on an (initial) regular design. (b) It is less prevalent if the sequential optimization does include an explorative mechanism (e.g., the expected improvement criterion in

Kriging), as design points will now be gradually sampled in unexplored regions of the space. (c) This issue can never be completely avoided if we want to access the quality of the regression model after optimization. Disregarding the model evaluation is also not an option, as it is clear that models might be wrong, and we need a way to assess their quality. Possible remedies might be to restrict the design points further to the area surrounding the best obtained solution and explicitly state that the measured performance values and subsequent interpretations hold only for this region. Measuring the performance and interpreting the model near the best obtained solution instead of across the whole input space might be desired at this stage anyway. Another option is to use lower-dimensional projections of the data to visualize the distribution of points the estimators are based upon. (d) The problem has to our knowledge not been addressed in other relevant papers from the optimization field and most likely requires further research.

6 Illustrative Examples

Four well known test functions are considered in the subsequent examples. However, the original ranges of the decision variables have been adapted for illustrative purposes of the meta-modeling framework and are listed in the following together with the definitions of the functions. Visualizations for the 2D case are provided in Figure 4.

Ackley test function:

$$f(\mathbf{x}) = 20 + \exp(1) - 20\exp\left(-0.2\sqrt{\frac{1}{m}\sum_{i=1}^{m} x_i^2}\right) - \exp\left(\frac{1}{m}\sum_{i=1}^{m}\cos(2\pi x_i)\right) \qquad \mathbf{x} \in [-2, 2]^m$$

Rosenbrock test function:

$$f(\mathbf{x}) = \sum_{i=1}^{m-1}\left(100\left(x_{i+1} - x_i^2\right)^2 + (1 - x_i)^2\right) \qquad \mathbf{x} \in [-2, 2]^m$$

Kotancheck test function (non-noisy; Vladislavleva et al., 2007):

$$f(\mathbf{x}) = \frac{\exp(-(x_2 - 1.2)^2)}{1 + (x_1 - 2.5)^2} \qquad \mathbf{x} \in [1.5, 3.5] \times [0, 2] \times \mathbb{R}^{m-2}$$

Rastrigin test function:

$$f(\mathbf{x}) = 10m + \sum_{i=1}^{m}\left(x_i^2 - 10\cos(2\pi x_i)\right) \qquad \mathbf{x} \in [-2, 2]^m$$

6.1 Example 1: Validating a Kriging Model

In order to demonstrate the subtle dangers of applying resampling methods wrongly when validating a model, we chose an approach which has also been used in Jones et al. (1998) and Tenne and Armfield (2008). Let us assume that for a given small dataset we would like to study the predictive accuracy of a regression model. The dataset might be the initial design of a beginning optimization run with surrogate modeling or the complete design when only modeling is sought. As the regression model we consider Kriging with an anisotropic Gaussian covariance kernel. A not uncommon approach for researchers is to fit all parameters of the covariance kernel by maximum likelihood on the full dataset, interpret the model, and then justify the fitted model and

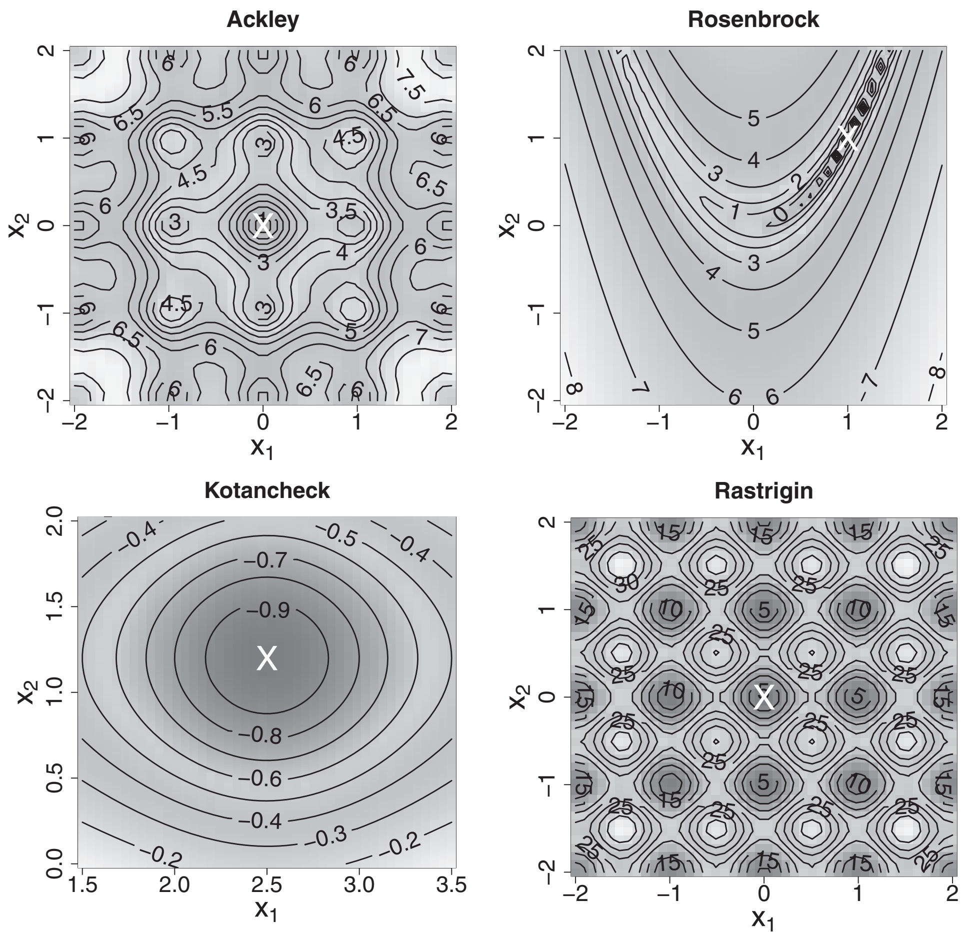

Figure 4: Visualization of considered test functions (2D input dimension) within predefined bounds. The Rosenbrock function is displayed in log-scale. The optimum of the function is marked by a white cross.

one's subsequent interpretations by presenting a low root mean square error (RMSE) obtained by leave-one-out cross-validation. The problem here is that during LOO-CV, the parameters of the Kriging model are not fitted again in each cross-validation iteration, but instead the ones which have been obtained from the fit on the complete data are used. Rather one should cross-validate the whole model selection including the fitting of kernel parameters (see comments 4 and 5 in Section 4). We compared the bias of both approaches for error estimation on the four test functions detailed above. The results were obtained in the following way: 50 points were randomly sampled from the displayed region of interest of the 3D versions of the functions for training and also 500,000 additional points as an independent test set. The Kriging model was fit on the training set, its RMSE was assessed by both LOO-CV approaches, and the bias of both estimators was calculated by comparing it with the true RMSE of the model when predicting the large test set.

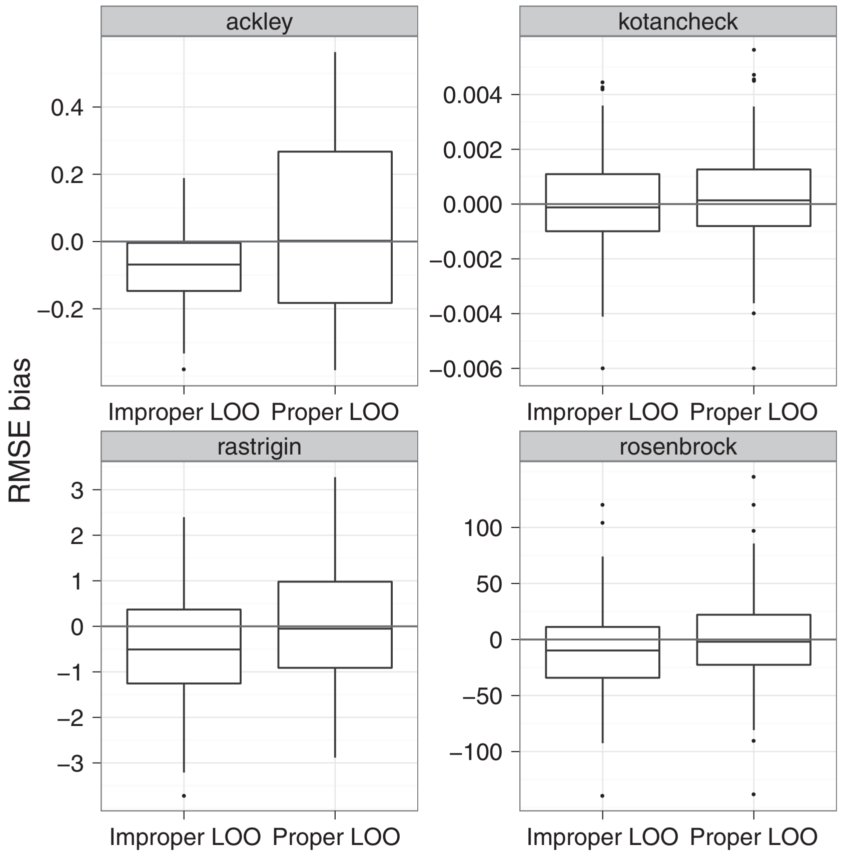

Figure 5: Results for both LOO-CV approaches on the four test functions.

As can be seen clearly in Figure 5, employing the improper cross-validation after the kernel parameters are selected by looking at the full dataset can lead to a substantially optimistic error estimator while the fully cross-validated model selection is nearly unbiased. The former method is quite problematic when using its results as the justification for the appropriateness of a model, as there is no way to judge exactly how far off the results are. Note that the bias discussed here does not refer to the bias of the regressing model, but concerns the bias of the accuracy estimator of the model, which is the statistical measure we use to evaluate and justify a model.

6.2 Example 2: Sequential Parameter Optimization

Sequential parameter optimization (SPO) is a popular sequential optimization framework which heavily relies on meta-modeling techniques internally. Popular statistical modeling strategies include tree-based methods, Kriging, multivariate adaptive regression splines (MARS), and neural networks. Starting with a small number of initial function evaluations, termed an initial design, SPO performs the following operations in every iteration: It fits a meta-model to the points evaluated so far, optimizes this meta-model (which is cheap to evaluate), and adds the obtained optimum and its true fitness value to the current design. This procedure is repeated until the maximally allowed number of iterations is reached.

The aim of the following example is to show the benefit of embedding a resampling technique into the SPO framework. This is done by optimizing the four functions listed above in five dimensions using SPO with different meta-models. For each of the functions, five meta-model candidates are considered as surrogates for the true functional relationship between the input variables and the function output. These are a full quadratic model, a regression tree, a random regression forest, MARS, and a Kriging model with a Matern 5/2 kernel function.

Initially, the meta-model is estimated using a latin hypercube design of 50 points, where the minimal distance of each design point to its neighbors is maximized. For each function, all five regression models use the same initial design to reduce variance in later comparisons. SPO is allowed a budget of 100 sequential iterations. The meta-model is optimized by generating 50,000 points by random latin hypercube sampling and selecting the best one with respect to the meta-model predictions. The whole procedure is repeated 20 times for every test function and meta-model to capture the stochastic nature of the algorithm.

Model accuracy is measured after the initial design has been evaluated. The regression models are resampled by hold-out with 2/3 of the data for training, a 10-fold cross-validation, LOO-CV, and subsampling with 9/10 of the data for training and 100 iterations. The resampled predictions of the models are assessed by root mean square error (RMSE), median absolute error (MAE), and Spearman's rho, which measures the correlation between the true and predicted ranking of the design points. These values are then compared to the minimal fitness value obtained during the optimization. This allows one to study whether a model with better predictive accuracy will also be superior in guiding SPO to a superior solution during optimization and what influence the different resampling strategies and performance measures have on the results. In general, we try to keep the experimental setup simple, avoiding models which have to be tuned by an external optimizer (e.g., support vector regression and neural networks).

Comparing the best obtained fitness values for all five meta-model strategies in Figure 6, Kriging seems to perform best overall, but it seems to be quite different for each test function which meta-model comes second best or even outperforms Kriging.

To determine which combination of resampling strategy and performance measure is most informative about the usefulness of the regression model for the optimization task, we compute for each strategy-measure pair the rank correlation between the resampled performance values and the finally obtained fitness value after sequential optimization. The results are displayed in Table 1.

We make the following observations: In all cases there is a positive correlation between the performance rankings obtained by resampling after the initial design and the rankings of the obtained final fitness values, although, for example, Rastrigin for this is less strongly expressed. Regarding the resampling strategies: Hold-out is almost completely dominated by subsampling and cross-validation as it uses the few data points of the initial design less efficiently and can therefore not be recommended. LOO-CV is not completely dominated, but it does not score too well overall, possibly due to its higher variance. Subsampling is often a bit better then cross-validation; at least these results suggest it here as the method of choice. It is, of course, computationally more expensive than CV, but as the initial design is often quite small, it will still be feasible. Regarding the performance measures: Here the picture is less clear, as no measure clearly dominates the others. If we consider only the results in combination

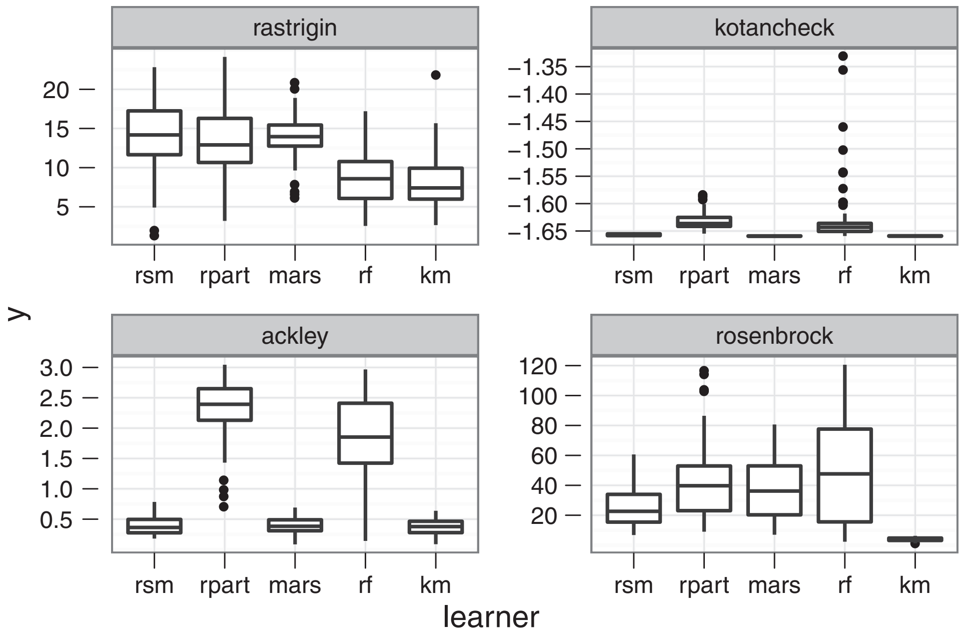

Figure 6: Boxplots of the best function values found by each SPO run using the five different meta-models.

Table 1: Rank correlation between resampled performance values and optimal obtained fitness values for the five regression models.

Function	Resampling	RMSE	MAE	Spearman's rho
Rastrigin	Hold-out	0.24	0.26	0.08
Rastrigin	10CV	0.44	0.29	0.35
Rastrigin	LOO-CV	0.20	0.17	0.23
Rastrigin	Subsampling	0.48	0.27	0.41
Kotancheck	Hold-out	0.84	0.85	0.80
Kotancheck	10CV	0.86	0.85	0.83
Kotancheck	LOO-CV	0.82	0.81	0.79
Kotancheck	Subsampling	0.83	0.84	0.83
Ackley	Hold-out	0.13	0.26	0.09
Ackley	10CV	0.49	0.49	0.45
Ackley	LOO-CV	0.53	0.41	0.50
Ackley	Subsampling	0.53	0.62	0.43
Rosenbrock	Holdout	0.11	0.11	0.17
Rosenbrock	10CV	0.48	0.41	0.44
Rosenbrock	LOO-CV	0.24	0.25	0.26
Rosenbrock	Subsampling	0.50	0.50	0.52

with subsampling, no substantial differences can be observed, as only MAE does not score very well on Rastrigin. In paricular, we cannot conclude that the ranking-based measure leads to better results than selecting the default RMSE.

7 Conclusions and Outlook

Meta-modeling techniques play a prominent role in modern optimization procedures with expensive evaluations. Much research has focused on the design of powerful modeling methods. In this paper we have given an overview of well-known resampling methods from statistics and how they can be used to choose or tune a meta-model.

Proper model validation in this context is crucial. First of all, employing a meta-model which badly approximates the original objective function cannot lead to reliable optimization results. Secondly, hyperparameter tuning for a specific model class, selecting from a finite amount of different models and possibly choosing relevant features are important steps in practical optimization. It is well known that their correct application leads to an improvement in generalization performance. As in model validation, a properly evaluated performance measure is needed in order to compare among the potential models. But when very few sample points are combined with very powerful and flexible regression models, overfitting becomes likely. Therefore, the model validation procedure plays a key role for determining a useful model, subsequently performing efficient optimization, and being able to draw reliable interpretations from the obtained models.

After introducing the three basic strategies, cross-validation, bootstrapping, and subsampling, nested resampling is discussed and common pitfalls and design mistakes are highlighted when using these methods. As a summary recommendation, we suggest consideration of the efficient leave-one-out cross-validation for fast model tuning, the .632+ bootstrap for small samples sizes with low complexity models, and when no tuning is required, and subsampling or repeated cross-validation in all other cases (see Figure 3).

We provided two simple examples to demonstrate how the presented algorithms and the general methodology might be applied in a modeling or optimization task. Further work is needed to study the relationship between the accuracy of a meta-model and its influence on the obtained final fitness in a sequential optimization process with surrogate modeling. Other ranking-based performance measures might be worthwhile instead of the traditional types of losses known from regression. Furthermore, the influence of the distribution of points obtained during optimization on the performance estimators and possible adjustments need to be better studied. Also, model combination techniques seem very promising to hopefully merge the advantages of predictive performance and optimization process guidance of different meta-models.

Acknowledgments

This work was partly supported by the Collaborative Research Center SFB 823, the Graduate School of Energy Efficient Production and Logistics and the Research Training Group "Statistical Modelling" of the German Research Foundation.

References

Alpaydin, E. (1999). Combined 5 × 2CV F test for comparing supervised classification learning algorithms. *Neural Computation*, 11:1885–1892.

Bartlett, P., Boucheron, S., and Lugosi, G. (2002). Model selection and error estimation. *Machine Learning*, 48(1–3):85–113.

Bartz-Beielstein, T. (2006). *Experimental research in evolutionary computation; The new experimentalism*. Berlin: Springer.

Bengio, Y., and Grandvalet, Y. (2004). No unbiased estimator of the variance of k-fold cross-validation. *Journal of Machine Learning Research*, 5:1089–1105.

Bhattacharya, M. (2008). Meta model based EA for complex optimization. *International Journal of Computational Intelligence*, 1(4):36–45.

Binder, H., and Schumacher, M. (2008). Adapting prediction error estimates for biased complexity selection in high-dimensional bootstrap samples. *Statistical Applications in Genetics and Molecular Biology*, 7(1):12.

Birattari, M., Stützle, T., Paquete, L., and Varrentrapp, K. (2002). A racing algorithm for configuring metaheuristics. In W. Langdon (Ed.), *Genetic and Evolutionary Computation Conference (GECCO)*, pp. 11–18.

Bischl, B. (2010). MLR: Machine learning in R. Retrieved from http://mlr.r-forge.r-project.org

Büche, D., Schraudolph, N. N., and Koumoutsakos, P. (2004). Accelerating evolutionary algorithms with Gaussian process fitness function models. *IEEE Transactions on Systems, Man and Cybernetics*, 35:183–194.

Cawley, G., and Talbot, N. (2004). Fast exact leave-one-out cross-validation of sparse least-squares support vector machines. *Neural Networks*, 17(10):1467–1475.

Diamantidis, N., Karlis, D., and Giakoumakis, E. (2000). Unsupervised stratification of cross-validation for accuracy estimation. *Artificial Intelligence*, 116(1–2):1–16.

Dietterich, T. (1998). Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10(7):1895–1923.

Efron, B. (1979). Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, 7(1):1–26.

Efron, B. (1983). Estimating the error rate of a prediction rule: Improvement on cross-validation. *Journal of the American Statistical Association*, 78(382):316–331.

Efron, B., and Tibshirani, R. (1997). Improvements on cross-validation: The 0.632 + bootstrap method. *Journal of the American Statistical Association*, 92(438):548–560.

El-Beltagy, M., Nair, P. B., and Keane, A. J. (1999). Metamodeling techniques for evolutionary optimization of computationally expensive problems: Promises and limitations. In *Genetic and Evolutionary Computation Conference (GECCO)*, pp. 196–203.

Emmerich, M., Giannakoglou, K., and Naujoks, B. (2006). Single- and multiobjective evolutionary optimization assisted by Gaussian random field metamodels. *IEEE Transactions on Evolutionary Computation*, 10(4):421–439.

Fiebrink, R., and Fujinaga, I. (2006). Feature selection pitfalls and music classification. In *Proceedings of the Eleventh International Society for Music Information Retrieval Conference (ISMIR)*, pp. 340–341.

Fu, W. J., Carroll, R. J., and Wang, S. (2005). Estimating misclassification error with small samples via bootstrap cross-validation. *Bioinformatics*, 21(9):1979–1986.

Giotis, A. P., and Giannakoglou, K. C. (1999). Single- and multi-objective airfoil design using genetic algorithms and artificial intelligence. In *EUROGEN 99, Evolutionary Algorithms in Engineering and Computer Science*.

Good, P. (2005). *Resampling methods: A practical guide to data analysis*. Basel, Switzerland: Birkhauser.

Hastie, T., Tibshirani, R., and Friedman, J. (2001). *The elements of statistical learning*. Berlin: Springer.

Hothorn, T., Leisch, F., Zeileis, A., and Hornik, K. (2005). The design and analysis of benchmark experiments. *Journal of Computational and Graphical Statistics*, 14:675–699.

Huang, D., Allen, T., Notz, W., and Zeng, N. (2006). Global optimization of stochastic black-box systems via sequential Kriging meta-models. *Journal of Global Optimization*, 34(3):441–466.

Jin, R., Chen, W., and Simpson, T. (2000). Comparative studies of metamodeling techniques under multiple modeling criteria. *Structural and Multidisciplinary Optimization,* 23:1–13.

Jin, Y. (2005). A comprehensive survey of fitness approximation in evolutionary computation. *Soft Computing*, 9(1):3–12.

Jin, Y., and Brank, J. (2005). Evolutionary optimization in uncertain environments: A survey. *IEEE Transactions on Evolutionary Computation*, 9(3):303–318.

Jin, Y., Olhofer, M., and Sendhoff, B. (2000). On evolutionary optimization with approximate fitness functions. In *Genetic and Evolutionary Computation Conference (GECCO)*, pp. 786–793.

Jin, Y., Olhofer, M., and Sendhoff, B. (2002). A framework for evolutionary optimization with approximate fitness functions. *IEEE Transactions on Evolutionary Computation*, 6:481–494.

Jones, D., Schonlau, M., and Welch, W. (1998). Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13(4):455–492.

Kampolis, I. C., Zymaris, A. S., Asouti, V. G., and Giannakoglou, K. C. (2007). Multilevel optimization strategies based on metamodel-assisted evolutionary algorithms, for computationally expensive problems. In K. C. Tan and J.-X. Xu (Eds.), *IEEE Congress on Evolutionary Computation (CEC)*, pp. 4116–4123.

Kern, S., Hansen, N., and Koumoutsakos, P. (2006). Local meta-models for optimization using evolution strategies. In *Parallel problem solving from nature, PPSN IX*, pp. 939–948.

Kim, J.-H. (2009). Estimating classification error rate: Repeated cross-validation, repeated hold-out and bootstrap. *Computational Statistics and Data Analysis*, 53(11):3735–3745.

Knowles, J., and Nakayama, H. (2008). Meta-modeling in multiobjective optimization. In *Multi-objective optimization: Interactive and evolutionary approaches* (pp. 245–284). Berlin: Springer.

Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 1137–1143.

Kruisselbrink, J. W., Emmerich, M. T. M., Deutz, A. H., and Bäck, T. (2010). A robust optimization approach using Kriging metamodels for robustness approximation in the CMA-ES. In *IEEE Congress on Evolutionary Computation (CEC)*, pp. 1–8.

Lim, D., Jin, Y., Ong, Y.-S., and Sendhoff, B. (2010). Generalizing surrogate-assisted evolutionary computation. *IEEE Transactions on Evolutionary Computation*, 14(3):329–355.

Loshchilov, I., Schoenauer, M., and Sebag, M. (2010). Comparison-based optimizers need comparison-based surrogates. In *Proceedings of the 11th International Conference on Parallel Problem Solving from Nature* (pp. 364–373). Berlin: Springer.

Mersmann, O., Bischl, B., Trautmann, H., Preuss, M., Weihs, C., and Rudolph, G. (2011). Exploratory landscape analysis. In *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation (GECCO)*, pp. 829–836.

Mersmann, O., Preuss, M., and Trautmann, H. (2010). Benchmarking evolutionary algorithms: Towards exploratory landscape analysis. In *Parallel problem solving from nature (PPSN). Lecture notes in computer science*, Vol. 6238 (pp. 73–82). Berlin: Springer.

Molinaro, A., Simon, R., and Pfeiffer, R. (2005). Prediction error estimation: A comparison of resampling methods. *Bioinformatics*, 21(15):3301–3307.

Mukhopadhyay, N., and de Silva, B. M. (2009). *Sequential methods and their applications*. New York: Chapman & Hall/CRC.

Myers, R., and Montgomery, D. (1995). *Response surface methodology*. New York: Wiley.

Nadeau, C., and Bengio, Y. (2003). Inference for the generalization error. *Machine Learning*, 52(3):239–281.

Nakayama, H., Yun, Y., and Yoon, M. (2009). *Sequential approximate multiobjective optimization using computational intelligence*. Berlin: Springer.

Ong, Y., Nair, P., and Keane, A. (2003). Evolutionary optimization of computationally expensive problems via surrogate modeling. *AIAA Journal*, 41(4):687–696.

Ong, Y., Zhou, Z., and Lim, D. (2006). Curse and blessing of uncertainty in evolutionary algorithms using approximation. In *IEEE Congress on Evolutionary Computation (CEC)*, pp. 2928–2935.

Paenke, I., Branke, J., and Jin, Y. (2006). Efficient search for robust solutions by means of evolutionary algorithms and fitness approximation. *IEEE Transactions on Evolutionary Computation*, 10(4):405–420.

Salzberg, S. (1997). On comparing classifiers: Pitfalls to avoid and a recommended approach. *Data Mining and Knowledge Discovery*, 1(3):317–328.

Sanchez, E., Pintos, S., and Queipo, N. (2008). Toward an optimal ensemble of kernel-based approximations with engineering applications. *Structural and Multidisciplinary Optimization*, 36(3):247–261.

Santers, T., Williams, N., and Notz, W. (2003). *The design and analysis of computer experiments*. Berlin: Springer.

Shao, J. (1993). Linear model selection by cross-validation. *Journal of the American Statistical Association*, 88(422):486–494.

Shi, L., and Rasheed, K. (2010). A survey of fitness approximation methods applied in evolutionary algorithms. *Computational intelligence in expensive optimization problems* (pp. 3–28). Berlin: Springer.

Simon, R. (2007). Resampling strategies for model assessment and selection. In *Fundamentals of data mining in genomics and proteomics* (pp. 173–186). Berlin: Springer.

Simon, R., Radmacher, M. D., Dobbin, K., and McShane, L. M. (2003). Pitfalls in the use of DNA microarray data for diagnostic and prognostic classification. *Journal of the National Cancer Institute*, 95(1):14–18.

Simpson, T., Mauery, T., Korte, J., and Mistree, F. (1998). Comparison of response surface and Kriging models for multidisciplinary design optimization. In *Proceedings of the 7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, pp. 98–4755.

Smit, S., and Eiben, A. (2009). Comparing parameter tuning methods for evolutionary algorithms. In A. Tyrrell (Ed.), *IEEE Congress on Evolutionary Computation (CEC)*, pp. 399–406.

Stone, M. (1974). Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society, Series B*, 36(1):111–147.

Stone, M. (1977). An asymptotic equivalence of choice of model by cross-validation and Akaike's criterion. *Journal of the Royal Statistical Society, Series B*, 39:44–47.

Sundararajan, S., and Keerthi, S. (2001). Predictive approaches for choosing hyperparameters in Gaussian processes. *Neural Computation*, 13(5):1103–1118.

Tenne, Y., and Armfield, S. (2008). Metamodel accuracy assessment in evolutionary optimization. In *IEEE Congress on Evolutionary Computation (CEC)*, pp. 1505–1512.

Ulmer, H., Streichert, F., and Zell, A. (2003). Evolution strategies assisted by Gaussian processes with improved pre-selection criterion. In *IEEE Congress on Evolutionary Computation (CEC)*, pp. 692–699.

Vladislavleva, E., Smits, G., and Kotanchek, M. (2007). Better solutions faster: Soft evolution of robust regression models in Pareto genetic programming. In R. L. Riolo, T. Soule, and B. Worzel (Eds.), *Genetic programming theory and practice V*, *Genetic and evolutionary computation* (chap. 2, pp. 13–32). Berlin: Springer.

Wagner, T., Emmerich, M., Deutz, A., and Ponweiser, W. (2010). On expected-improvement criteria for model-based multi-objective optimization. In R. Schaefer (Ed.), *Parallel Problem Solving from Nature (PPSN)*, pp. 718–727.

Wahba, G. (1980). Spline bases, regularization, and generalized cross validation for solving approximation problems with large quantities of noisy data. In *International Conference on Approximation Theory in Honour of George Lorenz*.

Weihs, C. (1993). Canonical discriminant analysis: Comparison of resampling methods and convex-hull approximation. *Information and Classification* (pp. 225–238). Berlin: Springer.

Weiss, S., and Kulikowski, C. (1991). *Computer systems that learn: Classification and prediction methods from statistics, neural nets, machine learning, and expert systems*. San Mateo, CA: Morgan Kaufmann.

Wu, C. (1986). Jackknife, bootstrap and other resampling methods in regression analysis. *Annals of Statistics*, 14:1261–1295.

Younis, A., and Dong, Z. (2010). Metamodelling using search space exploration and unimodal region elimination for design optimization. *Engineering Optimization*, 6(42):517–533.

Yun, Y., Yoon, M., and Nakayama, H. (2009). Multi-objective optimisation based on meta-modelling using support vector regression. *Optimization and Engineering*, 10(2):167–181.

Zhou, Z., Ong, Y.-S., Nair, P., Keane, A., and Lum, K. (2007). Combining global and local surrogate models to accelerate evolutionary optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, 37(1):66–76.