Proceedings of the 2020 Winter Simulation Conference
K.-H. Bae, B. Feng, S. Kim, S. Lazarova-Molnar, Z. Zheng, T. Roeder, and R. Thiesing, eds.

SIMULATION OPTIMIZATION BASED FEATURE SELECTION, A STUDY ON DATA-DRIVEN OPTIMIZATION WITH INPUT UNCERTAINTY

Kimia Vahdat
Sara Shashaani

Edward P. Fitts Department of Industrial and Systems Engineering
North Carolina State University
915 Partners Way,
Raleigh, NC 27606, USA

ABSTRACT

In machine learning, removing uninformative or redundant features from a dataset can significantly improve the construction, analysis, and interpretation of the prediction models, especially when the set of collected features is extensive. We approach this challenge with simulation optimization over a high dimensional binary space in place of the classic greedy search in forward or backward selection or regularization methods. We use genetic algorithms to generate scenarios, bootstrapping to estimate the contribution of the intrinsic and extrinsic noise and sampling strategies to expedite the procedure. By including the uncertainty from the input data in the measurement of the estimators' variability, the new framework obtains robustness and efficiency. Our results on a simulated dataset exhibit improvement over state-of-the-art accuracy, interpretability, and reliability. Our proposed framework provides insight for leveraging Monte Carlo methodology in probabilistic data-driven modeling and analysis.

1 INTRODUCTION

Simulation Optimization (SO) entails optimizing problems based on stochastic simulations and requires estimating the intrinsic or stochastic error for success. This error characterizes the deviation between the mean and the simulated output using a finite sample size. Extrinsic error or input uncertainty is another source of error that is often, especially when the search space is large, overlooked in the simulation optimization solvers.

One class of high demand stochastic optimization problems is the data-driven models that are stochastic due to the random distribution of data despite being treated statically within machine learning approaches. Hence, accounting for intrinsic and extrinsic errors in estimating the models' outcome is crucial for rendering robustness in the optimization. This paper aims to investigate this concept for a feature selection problem, which is a challenging topic in the machine learning domain. In the presence of a large number of explanatory features in supervised learning algorithms, feature selection aims to find a subset of the features that provide the *best* predictive results and interpretability power. This search is considered hard due to the enormity of $2^p - 1$ possible subsets of p features.

Moreover, finding features that lead to robust predictions is not trivial. Cross-validation is a popular way to measure external performance in machine learning. But to gain more robustness, one needs to address the variance of all sources of error. Other commonly used methods are Jackknife sampling and bootstrapping, where the former is equivalent to leave-one-out cross-validation while the latter is a more generalized resampling technique. Of course, conditional distributions of data cannot be resampled with either, when only joint distributions are available in the original dataset (sample). Existing sampling-based approaches to feature selection do not leverage additional information in the optimization, such as variance estimates of the original dataset's performance. This motivates a study of bringing the model and input

uncertainty into simulation optimization for a data-driven case within empirical high dimensional space for the inputs with limited data availability.

1.1 Feature Selection

Feature selection (FS) is the task of removing uninformative and/or redundant features from a dataset. This is an important task because contrary to what may be understood, more data does not lead to better predictions. The inclusion of non-contributing features in the process of learning causes overfitting in the training set, more computationally expensive development and updating of the model, and less inference or interpretation power (Guyon and Elisseeff 2003). The main categories of feature selection methods are filter methods, wrapper methods, and embedded methods. Filter methods are applied as a pre-processing step and are generally considered an unsupervised approach (Kuhn and Johnson 2013).

Wrapper methods are considered a supervised approach in which, a search algorithm is wrapped around the learning algorithm. The embedded methods are specific to some learning algorithms with FS as part of the model construction. The two categories of embedded methods are built-in mechanisms such as tree based models or support vector machines (Chen and Lin 2006), and regularization models such as LASSO or Ridge regression (Singh et al. 2016; Tibshirani 1996). Filter methods are independent of the response variables, and embedded methods are too dependent on the learning algorithm. Wrapper methods are optimization problems mostly solved with greedy approaches. In addition to the three main methods, there are hybrid methods that combine two or multiple categories together with the aim of achieving better performances.

The goal of wrapper methods, as in the classic statistical learning, is to choose a subset of features that the learning algorithm utilizes to build a model such that its generated predictions are closest to the unknown response for a given set of feature values. In formulating FS as an optimization, we search for a feature subset that distinguishes uninformative or redundant features from relevant and contributing features. The search space is finite but large in size and expensive when the data is large as well. We extend this foundation with a probabilistic and Monte Carlo approach described in Section 2.

1.2 Relation to Robustness

Selection bias or overfitting is an issue if we evaluate our selected feature subset's performance on the same set of data points that train the prediction model. Therefore, one must perform a correct validation on a separate dataset from the modeling dataset. However, using a single modeling and validation dataset does not resolve the issue, considering the likelihood that a subset of features can perform well on one validation data set but poorly on many others. Thus, FS's more important goal is to find a robust subset of features that generalizes the predictions. As mentioned earlier, a common way in practice to handle the robustness objective is cross-validation, where the available dataset is divided into k folds and in k steps, the performance of a model trained with $k-1$ folds is tested on the remaining fold for validation. Other methods, such as boosting and bagging, can take resamples of the data into two folds of train and test sets to make the training sets less dependent. Both of these approaches speak to the importance of choosing the correct estimation of the optimized objective function. Robustness is direr when the available data is not large, but larger datasets are also prone to overfitting. With the advancement of sampling and uncertainty estimation techniques, we find this area to have significant improvement potential with more probabilistic approaches.

1.3 Solver

The mapping between the inputs and outputs builds a surrogate model for optimization. In other words we are dealing with a derivative-free model-based solution methods for this SO. Given the nature of the problem that entails high dimensional binary decision variables, random search solvers appear to be more suitable than others. Recent ranking and selection enhancements allow up to 10^6 design points, which

equivalently can solve an FS for up to $p = 20$ features. In real applications, p is much larger. In addition, mixed integer programming is not reliable for simulation optimization and integer lattice SO solvers are not applicable here because, defining structure in high dimensional binary spaces has not been visited before. For the reason of easy implementation, Genetic Algorithms are chosen for this study but employing more guided search algorithms that take advantage of the history of the trajectory may reduce the computational burden and are left for future research. In other words, if we divide the FS into subset generation, subset evaluation and result validation, we mainly focus on subset evaluation and result validation in `SOFS` and leave the subset generation to GA. With GA there is assurance from previous studies (Derrac et al. 2012; Li et al. 2010; Cano et al. 2003; Kudo and Sklansky 2000) that better results than many other solvers, especially in large-scale FS problems, can be obtained. Additionally there is less chance to arrive at a local optimal solution than other solvers.

In the remainder of this paper, we formulate FS as a simulation optimization and state the problem and notations (Section 2), incorporate input uncertainty with bootstrapping from the empirical data distribution and propose a new SO-based algorithm (Section 3), provide the supporting performance results in comparison with the alternative feature selection methods (Section 4), and conclude (Section 5).

2 Simulation Optimization Formulation and Notations

We let every dataset in this paper have the form $\mathcal{D} = \{< \boldsymbol{z}_i, y_i >\}_i$, where $\boldsymbol{z}_i$ is a p-dimensional variable containing the values of features and y_i is the response or target value. For a newly collected set of feature values $\boldsymbol{z}$, we denote the predicted response by $r_{\mathcal{D}}(\boldsymbol{z}, \boldsymbol{x})$, where $r_{\mathcal{D}}(\cdot, \boldsymbol{x})$ is the prediction model that a learning algorithm, such as linear regression, has constructed from an available dataset $\mathcal{D}$ by utilizing only a subset of features $\boldsymbol{x}$. $\boldsymbol{x}$ is a p-dimensional binary vector, each of its components taking a value of 1 if the corresponding feature is used in training and 0 otherwise. Suppose $\mathcal{D}$ is the dataset available for modeling, and $\mathcal{D}_0$ is the unknown dataset that we wish to predict. Let $Q_{\mathcal{D}_0}(r_{\mathcal{D}}(\boldsymbol{z}, \boldsymbol{x}), y) := Q(r_{\mathcal{D}}(\{\boldsymbol{z}_j, \boldsymbol{x}\}_{j\in\mathcal{D}_0}), \{y_j\}_{j\in\mathcal{D}_0})$ be a generic error measure summarizing the deviation of predicted and observed responses over another set of data $\mathcal{D}_0$. Therefore, Q is a closed-form deterministic function; in the case of a mean square error (MSE) Q is simply an ℓ_2 norm, i.e. $Q(\boldsymbol{a}, \boldsymbol{b}) = m^{-1}\|\boldsymbol{a} - \boldsymbol{b}\|_2^2$ for two m-dimensional vectors $\boldsymbol{a}$ and $\boldsymbol{b}$.

It is understood that $\mathcal{D}_0 \cap \mathcal{D} = \emptyset$ is essential in avoiding implicit bias on the error measure. Letting P and P_0 be the distributions of the data used for training ($\mathcal{D}$) and prediction ($\mathcal{D}_0$), a general optimization problem, akin to an early suggestion by Efron and Tibshirani (1997), then follows

$$\min_{\boldsymbol{x}\in\{0,1\}} f(\boldsymbol{x}) := \mathbb{E}_{\mathcal{D}\sim P}\left[\mathbb{E}_{\mathcal{D}_0\sim P_0}\left[Q_{\mathcal{D}_0}\left(r_{\mathcal{D}}(\boldsymbol{z}, \boldsymbol{x}), y\right)\right]\right]. \tag{1}$$

Given that P and P_0 are unknown (often with the assumption that $P = P_0 = P^c$), the objective function in (1) is a computationally expensive stochastic black box on a binary space of p dimensions. Each of the expectations in the equation (1) needs to be estimated only using the on-hand dataset. In (1), there are two sources of uncertainty: the probability distribution of the data on hand and test dataset denoted as P and P_0, respectively. The combination of both is then estimated using the sample average approximation (SAA) (Kim et al. 2015).

In the SO context, validation is done through post-processing, where the solution of each solver at the end or at the point of exhausting a predefined simulation budget is re-evaluated with a fixed number of fresh simulation replications. This step avoids the optimization bias that is present in the estimated values from the solver (Mak et al. 1999). In the data-driven context this higher separation of modeling and validation step can be considered a *macro-replication* under which the *micro-replications* are drawn for objective function estimation. Through the macro-replications we compare the results of different FS methods in terms of reliability and robustness by looking at how the performance of each FS method varies across macro-replications. While macro-replications can be used for comparing any group of FS methods and a given dataset, the micro-replications are unique to `SOFS`. By using the bootstrapped $\hat{\mathcal{D}}_b$, $b = 1, 2, \cdots, B$ one replicates different training sets from $\mathcal{M}$, and from their complements ($\mathcal{M}\backslash\hat{\mathcal{D}}_b$) the test sets $\hat{\mathcal{D}}_{b,r}$,

$r = 1, 2, \cdots, R$ are drawn. This is to mimic having each input of this stochastic system as a whole dataset instead of a single data point. Figure 1 clarifies the sampling approach for a data-driven SO framework.

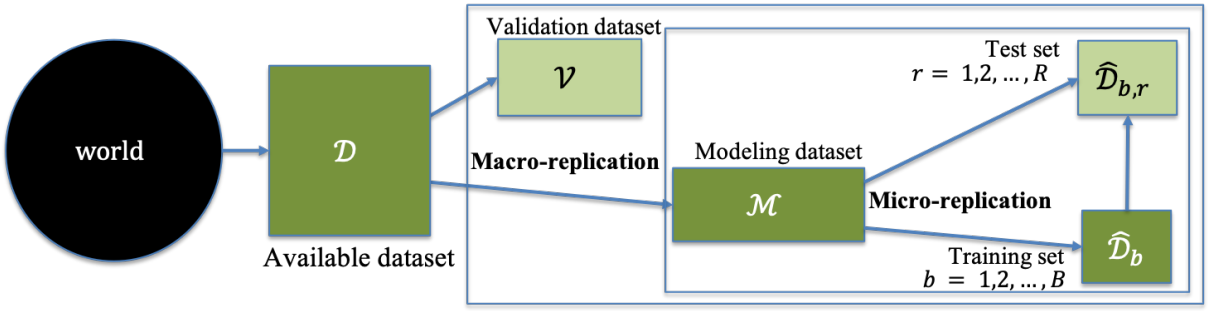

Figure 1: Akin SO framework is the sampling from the available dataset $\mathcal{D}$. The micro-replications enable subset evaluations, measuring accuracy, and contain training sets $\hat{\mathcal{D}}_b$ and test sets $\hat{\mathcal{D}}_{b,r}$ bootstrapped from $\mathcal{D}$ and $\mathcal{M} \backslash \hat{\mathcal{D}}_b$ respectively. The macro-replications $\mathcal{M}$ and $\mathcal{V}$ produce result validations, measuring robustness.

3 Input Uncertainty with Empirical Data Distribution

We wish for our prediction model to predict response values close to the observed response y for feature values $\boldsymbol{z}$ coming from an unknown dataset $\mathcal{D}_0$. In other words we seek $\text{argmin}_{\boldsymbol{x} \in \{0,1\}^p} \; Q_{\mathcal{D}_0}\left(r_{\mathcal{D}}(\boldsymbol{z}, \boldsymbol{x}), y\right)$, which is unattainable. But if $\hat{\mathcal{D}}_0$ (an estimate of $\mathcal{D}_0$) comes from the correct data distribution P^c, one can search for

$$\boldsymbol{x}^* = \text{argmin}_{\boldsymbol{x} \in \{0,1\}^p} \; f(\boldsymbol{x}|P^c) := \mathbb{E}_{\hat{\mathcal{D}}_0 \sim P^c}\left[Q_{\hat{\mathcal{D}}_0}\left(r_{\mathcal{D}}(\boldsymbol{z}, \boldsymbol{x}), y\right)\right].$$

This is again not possible because P^c is also unknown. The best we can do is to use the empirical distribution of the available dataset $\mathcal{D}$, and estimate

$$\hat{\boldsymbol{x}}^* = \text{argmin}_{\boldsymbol{x} \in \{0,1\}^p} \; f(\boldsymbol{x}|P) := \mathbb{E}_{\hat{\mathcal{D}}, \hat{\mathcal{D}}_0 \sim P}\left[Q_{\hat{\mathcal{D}}_0}\left(r_{\hat{\mathcal{D}}}(\boldsymbol{z}, \boldsymbol{x}), y\right)\right], \tag{2}$$

where P is the empirical distribution of the available dataset $\mathcal{D}$. In (2), consider the random dataset $\hat{\mathcal{D}}_0$ represents the simulation error and $\hat{\mathcal{D}}$ represents the extrinsic error. The interesting observation is that both are using the same distribution and are also interdependent, as we need to ensure $\hat{\mathcal{D}}_0$ does not overlap with $\hat{\mathcal{D}}$ to avoid an overfit. Following the structure illustrated in Figure 1, given a macro-replication we denote $\hat{\mathcal{D}}_b$ for every bootstrap of the input modeling data, and $\hat{\mathcal{D}}_{b,r}$ for every sample of the output generation. Also note their dependence with the shared subscript b. Hence, $F_r(\boldsymbol{x}|P_b) := Q_{\hat{\mathcal{D}}_{b,r}}(r_{\hat{\mathcal{D}}_b}(\boldsymbol{z}, \boldsymbol{x}))$ will be a single stochastic output using the bootstrapped input model P_b and can be written as

$$\begin{aligned} F_r(\boldsymbol{x}|P_b) &= f(\boldsymbol{x}|P_b) + \epsilon_r(\boldsymbol{x}|P_b) + \delta_r(\boldsymbol{x}|P_b) \\ &= f(\boldsymbol{x}|P^c) + (f(\boldsymbol{x}|P_b) - f(\boldsymbol{x}|P^c)) + \epsilon_r(\boldsymbol{x}|P_b) + \delta_r(\boldsymbol{x}|P_b). \end{aligned} \tag{3}$$

In (3), $f(\boldsymbol{x}|P_b) - f(\boldsymbol{x}|P^c)$ is the uncertainty that is forced due to deviance of the inputs coming from the empirical distribution P_b instead of P^c or the extrinsic error, $\epsilon_r(\boldsymbol{x}|P_b)$ represents the stochastic error due to the finiteness of outputs and $\delta_r(\boldsymbol{x}|P_b)$ represents the discrepancy due to the departure of the choice of training algorithm (simulation) from the real association of the features and the response variable. Discrepancy is often represented by a Gaussian Process for physical models (Kennedy and O'Hagan 2001) but not data-driven models and we will not explore it in this paper.

The sample average approximation of the objective function in (2) is

$$\bar{F}_R(\boldsymbol{x}|P_b) = \frac{1}{R}\sum_{r=1}^{R} F_r(\boldsymbol{x}|P_b).$$

To clarify, in classic machine learning the training is done over a training set which is equivalent to above with one bootstrap and estimates the prediction accuracy on one test set, that is usually the exact complement of the training set, i.e., $\hat{\mathbb{E}}_{\hat{\mathcal{D}},\hat{\mathcal{D}}_0\sim P_b}\left[Q_{\hat{\mathcal{D}}_0}\left(r_{\hat{\mathcal{D}}}(\boldsymbol{z},\boldsymbol{x}),y\right)\right]$. Here, we allow the random outputs to vary for a single training set and additionally will enable the input uncertainty to be quantified using multiple bootstraps $b = 1, 2, \cdots, B$ for the input model.

To measure the robustness of the performance of a selected decision variable $\boldsymbol{x}$ to the input model, we can then use a specific summary of

$$\bar{F}_R(\boldsymbol{x}|P_1), \bar{F}_R(\boldsymbol{x}|P_2), \cdots, \bar{F}_R(\boldsymbol{x}|P_B), \tag{4}$$

as the estimated objective function. Note that this summary is simply the worst case or the maximum value of (4) in robust optimization, which is a conservative approach. We denote this summary value by $\phi_{B,R}(\boldsymbol{x})$ and use that as the objective function. Consider two summary cases of bootstrapped results in (4):

(i) average: $\phi_{B,R}(\boldsymbol{x}) = B^{-1}\sum_{b=1}^{B}\bar{F}_R(\boldsymbol{x}|P_b)$,
(ii) $q\%$ quantile: $\phi_{B,R}(\boldsymbol{x}) = \bar{F}_R(\boldsymbol{x}|P_{\lceil B\times q\%\rceil})$.

The average reflects the mean performance over bootstraps while the quantile reflects the tail of the distribution, which arguably also clues to the variability across bootstraps. In (ii), observe that

$$\bar{F}_R(\boldsymbol{x}|P_{\lceil B\times q\%\rceil}) = \inf\{t:\ B^{-1}\sum_{b=1}^{B}\{\bar{F}_R(\boldsymbol{x}|P_b)\le t\}\ \ge q\},$$

where $B^{-1}\sum_{b=1}^{B}\{\bar{F}_R(\boldsymbol{x}|P_b)\le t\}$ is the empirical cdf of the distribution in (4). The quantile summary is equivalently used in risk analysis and referred to as Value at Risk.

3.1 Estimation with Reduced Variance

In this section we utilize a variance reduction strategy following (Barton et al. 2018), where a variant of the objective function

$$\hat{F}_R(\boldsymbol{x}|P_b) = c\times\frac{1}{B}\sum_{i=1}^{B}\bar{F}_R(\boldsymbol{x}|P_b) + (1-c)\times\bar{F}_R(\boldsymbol{x}|P_b),$$

is computed instead by choosing the constant c such that its variance is small; the smallest variance it can have is one due to the input uncertainty since that variance is irreducible. Hence, we derive

$$\begin{aligned}\text{Var}\left(\hat{F}_R(\boldsymbol{x}|P_b)\right) &= \left(\frac{c^2}{B^2}(B-1) + (\frac{c}{B}+1-c)^2\right)\text{Var}\left(\bar{F}_R(\boldsymbol{x}|P_b)\right)\\ &= \left((1-c)^2\frac{B-1}{B}+\frac{1}{B}\right)\text{Var}\left(\bar{F}_R(\boldsymbol{x}|P_b)\right)\\ &\overset{\text{set}}{=} \text{Var}\left(f(\boldsymbol{x}|P_b)\right).\end{aligned} \tag{5}$$

Solving for c we can write

$$c = 1 - \sqrt{\left(\frac{\text{Var}(f(\boldsymbol{x}|P_b))}{\text{Var}(\bar{F}_R(\boldsymbol{x}|P_b))} - \frac{1}{B}\right)\frac{B}{B-1}}. \tag{6}$$

Note that in (5), c is defined for a given training bootstrap b that we generalize by replacing P_b with P. Following the law of total variance, an unbiased estimator for $\text{Var}(\bar{F}_R(\boldsymbol{x}|P))$ is

$$\begin{aligned}\hat{\text{Var}}(\bar{F}_R(\boldsymbol{x}|P)) &= \hat{\mathbb{E}}_P\left[\text{Var}(\bar{F}_R(\boldsymbol{x}|P))\right] + \hat{\text{Var}}_P\left(\mathbb{E}[\bar{F}_R(\boldsymbol{x}|P)]\right), \\ &= \frac{1}{B}\sum_{b=1}^{B}\frac{1}{R(R-1)}\sum_{r=1}^{R}\left(F_r(\boldsymbol{x}|P_b) - \bar{F}_R(\boldsymbol{x}|P_b)\right)^2 \\ &\quad + \frac{1}{B-1}\sum_{b=1}^{B}\left(\bar{F}_R(\boldsymbol{x}|P_b) - \frac{1}{B}\sum_{b=1}^{B}\bar{F}_R(\boldsymbol{x}|P_b)\right)^2, \end{aligned} \tag{7}$$

which includes variance across training bootstraps and variance within the test bootstraps. Note that in (7), we assume that the covariances among the training and test sets are negligible. Furthermore,

$$\hat{\text{Var}}(f(\boldsymbol{x}|P)) = \hat{\text{Var}}(\bar{F}_R(\boldsymbol{x}|P)) - \frac{1}{B}\sum_{b=1}^{B}\left(\frac{1}{R(R-1)}\sum_{r=1}^{R}\left(F_r(\boldsymbol{x}|P_b) - \bar{F}_R(\boldsymbol{x}|P_b)\right)^2\right), \tag{8}$$

where the second right-hand side term is $\hat{\sigma}^2/R$ which is the average variance estimate within the test bootstraps. Consequently $\hat{c}$ can be computed by substituting the variances in equation (6) with their estimates in (7) and (8).

Note that if we work with the average across input models as the objective function, $B^{-1}\sum_{i=1}^{B}\hat{F}_R(\boldsymbol{x}|P_b) = B^{-1}\sum_{i=1}^{B}\bar{F}_R(\boldsymbol{x}|P_b)$. Then the effect of the variance reduction does not play out. In the quantile of the input models as the objective function, we expect to see the reduced variance improving the efficiency and requiring fewer micro-replications to reach a confidence interval half-width that is reasonable. Moreover, for the case of $R = 1$, that is, a sample of size 1 is used for estimation of an input model's performance, we let $c = 0$.

3.2 Common Random Numbers (CRN)

Several data-driven settings are different from the stochastic simulation settings with input uncertainty. First, the test sets are sampled from the training set's complement and hence dependent on them. The R outputs are estimated on distinct test sets for each bootstrapped input model P_b. As a result, common random numbers across input models to draw the same test sets for random output generation is unavailable.

As listed in Algorithm 1, we maintain CRN for the macro-replications using $\{\omega_j\}$. This allows for an external validation for a variety of scenarios for available dataset $\mathcal{M}$ and supposedly unknown $\mathcal{V}$. For every generated subset $\boldsymbol{x}$ within each macro-replication we draw bootstraps using $\{\xi_b\}$ and test samples using $\{\zeta_r\}$ hierarchically. Despite these random seeds being fixed, their resulting resamples on distinct modeling datasets $\mathcal{M}_j$ mimics having uniquely defined substreams and sub-substreams of random seeds. The CRN assists the optimization by enabling efficient comparison between two features subsets $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$:

$$\text{Var}\left(\phi_{B,R}(\boldsymbol{x}_1) - \phi_{B,R}(\boldsymbol{x}_2)\right) = \text{Var}\left(\phi_{B,R}(\boldsymbol{x}_1)\right) + \text{Var}\left(\phi_{B,R}(\boldsymbol{x}_2)\right) - 2\text{Cov}\left(\phi_{B,R}(\boldsymbol{x}_1), \phi_{B,R}(\boldsymbol{x}_2)\right),$$

where the covariance is maximized, knowing $\phi_{B,R}(\boldsymbol{x}) := \phi(\boldsymbol{x}, \{\xi_b\}_{b=1}^{B}, \{\zeta_r\}_{r=1}^{R})$ have the same random seeds for all $\boldsymbol{x}$ in one macro-replication.

With the solution reported from each macro-replication, denoted by $\boldsymbol{x}_j^*$, the performance measures of interest for the feature selection are computed by training the whole available dataset $\mathcal{M}_j$ and externally validating on $\mathcal{V}_j$ that remains unused in obtaining $\boldsymbol{x}_j^*$. The performance measures in this study include the average and variance of (i) deviation measure Q, (ii) the size of the optimal feature subset, and (iii) the percentage of detected true features.

Algorithm 1 `SOFS`(B, R)

Given optimization alg., training alg. $r(\cdot)$, dataset $\mathcal{D}$, fractions κ_1, κ_0, random streams $\{\omega_j\}, \{\xi_b\}, \{\zeta_r\}$. Tune the hyper-parameters of the optimization and training algorithms, if any.

for *macro-replication* $j = 1, 2, \cdots, J$ **do**

Draw $\mathcal{M}_j$ and $\mathcal{V}_j = \mathcal{D} \backslash \mathcal{M}_j$ using $\{\omega_j\}$.

Optimization starts:

for *iteration* $k = 1, 2, \cdots$ **do**

Generate feature subset(s) $\boldsymbol{x}_k$ via the optimization algorithm.

for *Training bootstrap* $b = 1, \cdots, B$ **do**

Draw $\hat{\mathcal{D}}_b$ from $\mathcal{M}_j$ of the size $\kappa_1 \times |\mathcal{M}_j|$, using $\{\xi_b\}$.

Train the model $r_{\hat{\mathcal{D}}_b}(.|\boldsymbol{x}_k)$.

for *Test bootstrap* $r = 1, 2, \cdots, R$ **do**

Draw $\hat{\mathcal{D}}_{b,r}$ from $\mathcal{M}_j \backslash \hat{\mathcal{D}}_b$ of the size $\kappa_0 \times |\mathcal{M}_j \backslash \hat{\mathcal{D}}_b|$, using $\{\zeta_r\}$.

Compute $F_r\left(\boldsymbol{x}_k|P_b\right) = Q_{\hat{\mathcal{D}}_{b,r}}(r_{\hat{\mathcal{D}}_b}(\boldsymbol{z}, \boldsymbol{x}_k), \boldsymbol{y})$.

end

Compute $\bar{F}_R(\boldsymbol{x}_k|P_b) = \frac{1}{R}\sum_{i=1}^{R} F_r\left(\boldsymbol{x}_k|P_b\right)$.

end

For all $b \in \{1, 2, \cdots, B\}$, and using computed $\hat{c}$ from (6) set the b-th micro-replication output as

$$\hat{F}_R(\boldsymbol{x}_k|P_b) = \hat{c}\left(\frac{1}{BR}\sum_{b=1}^{B}\sum_{r=1}^{R} F_r(\boldsymbol{x}_k|P_b)\right) + (1-\hat{c})\bar{F}_R(\boldsymbol{x}_k|P_b).$$

Report the objective function $\phi_{B,R}(\boldsymbol{x}_k)$ as some summary of $\hat{F}_R(\boldsymbol{x}|P_b),\ b = 1, 2, \cdots, B$.

end

Report the optimal subset $\boldsymbol{x}_j^*(B, R)$ and performance measures representing the mean squared error, number of features in the optimal subset, and the true positive rate (TPR) as

$$f_j = Q_{\mathcal{M}_j}\left(r_{\mathcal{V}_j}(\boldsymbol{z}, \boldsymbol{x}_j^*(B,R), y)\right);\quad g_j = \|\boldsymbol{x}_j^*(B,R)\|_1;\quad h_j = \|\boldsymbol{x}^*\|_1\left(\boldsymbol{x}_j^*(B,R)\right)^T\boldsymbol{x}^*,$$

respectively where $\boldsymbol{x}^*$ is the true solution.

end

Output: mean and standard deviations of the three performance measures $(\bar{f}, \hat{\sigma}_f), (\bar{g}, \hat{\sigma}_g)$, and $(\bar{h}, \hat{\sigma}_h)$.

4 NUMERICAL EXPERIMENTS

In this section, we experiment with the proposed `SOFS` and compare it with alternative FS methods.

4.1 Implementation

The choice of training algorithm (r) is kept open; here for simplicity, we utilize Generalized Linear Models or GLM (Nelder and Wedderburn 1972). GLM models assume the response variable follows one of many distributions in the exponential family (represented by link function ℓ), and its conditional mean has a linear

relationship with the features; i.e., $\ell(\mu_{y|\boldsymbol{z}}) = \boldsymbol{\beta z}$. Coefficients $\boldsymbol{\beta}$ are computed by fitting the model using the least-squares method. GLM models also assume that the error $y - \ell^{-1}(\boldsymbol{\beta z})$ is normally distributed, has a constant variance, and is independent per data points. Despite the limiting assumptions, the GLM structure's departure from that of the real system can be modeled by a bias generated from the discrepancy term δ in (3). To the best of our knowledge, empirical δ is not explored; we leave characterizing that for future research as well.

The simulated regression dataset that we generated for this study has 12 independent Gamma distributed features with both shape and scale parameters equal 2:

$$y = z_1 \times z_2 + z_{10}^2 - z_3 \times z_{17} - z_{15} \times z_4 + z_9 \times z_5 + z_{19} - z_{20}^2 + z_9 \times z_8,$$

following (Van der Laan et al. 2007), available in caret package in R under "LPH07_2", with added 108 standard normally distributed noise features and 100 correlated (auto-regressive) features with correlation value of 0.75. Use of the simulated dataset allows us to know beforehand which of the features are contributing in the prediction of the response, and detect the correctly and falsely selected features through each FS method.

We compare `SOFS` method with three other FS method, `RFE` or Recursive Feature Elimination (Guyon et al. 2002), `GAFS` (Kuhn and Johnson 2013), and `LASSO` (Tibshirani 1996). All of these methods are widely used in practice. The main characteristics in these methods are bagging in `RFE`, and GA based optimization in `GAFS`, which are combined in `SOFS`. `LASSO` is the benchmark for its known high performance. `RFE` finds the best subset size by ranking the features, whereas `SOFS` does this directly by solving the optimization problem. In most cases (we will show this in the results), `RFE` leads to smaller subsets due to this difference in the decision variables, but with significant variability in the best subsets. `GAFS` finds the best solution over different iterations of the GA. Hence it does not solve the problem directly. However, it evaluates the performance over internal and external levels using cross-validation or bootstrapping. `LASSO` is a regularization method that puts a penalty on the ℓ_1 norm of the coefficients. Usually, in this method, cross-validation is performed to fit the penalty coefficient. We compare the FS algorithms in Algorithm 1, by executing in the inner block marked with "Optimization starts" in place of `SOFS`. Hyper-parameters for each FS method is tuned using cross-validation and full factorial experimental design. These hyper-parameters include the probability of mutation, cross-over and population size for GA, and the penalty coefficient for `LASSO`. For all the cases recorded, the GA runs until either 5000 iterations complete, or no updates in the best fitness value is obtained in the last 150 iterations. The results of the two choices for the objective function, as described in Section 3 are shown in Table 1.

4.2 Role of Sample Size

B, and R are the two sample size choices for running an `SOFS` algorithm, which represents the micro-replications at a particular decision choice $\boldsymbol{x}$. We are interested in knowing whether these values achieve better results as they increase, comparable with any SAA method. We face a critical challenge here: the number of available data points is limited, and at some point, increasing B may only make the sample average look more and more like if the entire $\mathcal{M}$ was used instead, which leads to the old issue of overfitting. So finding the correct B is non-trivial. Additionally, each modeling bootstrap's size is κ_1 times the size of the available data. In contrast, the size of the test sets that computes the internal performance of $\boldsymbol{x}$ as stochastic outputs is κ_0 times the remainder of the available data. Although these decisions are somewhat heuristic, we would like to point out the trade-off between them: Larger κ_1 forces having fewer test sets and keeping R high imposes smaller test sets (κ_0) to prevent them from overlapping. But small test sets can hurt the estimation variance and bias. Smaller training sets enable more test sets and more extensive training sets.

Our experimental designs on $\kappa_1 \in \{0.5, 0.8\}$ and $\kappa_0 \in \{0.5, 1\}$ suggest $\kappa_1 = 0.5$ and $\kappa_0 = 0.5$ results in better performance measures. Intuitively, this setting leaves enough data points for the internal validation

and $R > 1$ stochastic outputs, although we realize that this may not be generalizable to other real-world datasets. We do not present the results of the comparison due to space limitations.

Table 1: Summary results for SR dataset where the average and standard deviation of the performance measures, representing MSE, subset size, and % of correct features detected, as described in Algorithm 1 are computed over 10 macro-replications. The $\bar{f}$ is $\times 10^{-2}$. The number of bootstraps, B, takes values 10, 20, and 30 and the number of testing resamples, R, is set to 1, 5 and 10. $\kappa_1 = 0.5$ and $\kappa_0 = 0.5$ for all rows. The objective function in the optimization is either the 90% quantile of the micro-replications or the sample average. Time is reported in minutes.

Obj. Func.	**FS**	B	R	$\bar{f} \pm \hat{\sigma}_f$	$\bar{g} \pm \hat{\sigma}_g$	$\hbar \pm \hat{\sigma}_h$	**Time**
-	RFE	-	-	53.23 ± 22.42	11.20 ± 25.32	0.03 ± 0.10	0.13
-	GAFS	-	-	16.17 ± 8.76	49.80 ± 5.59	0.79 ± 0.44	167.56
-	LASSO	-	-	12.56 ± 6.01	44.30 ± 7.04	0.92 ± 0.00	0.10
Average	SOFS	10	1	14.05 ± 5.99	43.50 ± 4.53	0.92 ± 0.00	12.69
			5	13.78 ± 4.79	39.50 ± 5.91	0.93 ± 0.03	28.37
			10	14.99 ± 6.44	36.20 ± 3.42	0.93 ± 0.03	38.16
		20	1	14.39 ± 6.05	35.10 ± 4.31	0.92 ± 0.02	15.95
			5	14.33 ± 5.23	30.30 ± 4.22	0.93 ± 0.03	34.84
			10	14.73 ± 5.41	30.30 ± 6.73	0.92 ± 0.00	74.50
		30	1	14.00 ± 6.00	28.80 ± 2.70	0.93 ± 0.03	24.61
			5	13.84 ± 5.71	27.40 ± 3.63	0.92 ± 0.00	45.92
			10	13.34 ± 5.17	29.30 ± 4.00	0.92 ± 0.00	97.91
90% Quantile	SOFS	10	1	15.34 ± 7.35	49.40 ± 6.15	0.90 ± 0.05	14.52
			5	14.80 ± 5.31	41.00 ± 3.19	0.92 ± 0.00	23.41
			10	13.70 ± 5.97	38.40 ± 5.72	0.92 ± 0.00	39.63
		20	1	13.80 ± 6.21	42.00 ± 6.03	0.93 ± 0.04	21.72
			5	14.14 ± 5.26	35.90 ± 5.89	0.93 ± 0.04	32.73
			10	14.28 ± 5.59	33.80 ± 3.79	0.91 ±0.03	61.78
		30	1	14.69 ± 6.08	37.90 ± 3.51	0.93 ± 0.03	27.07
			5	13.99 ± 6.55	34.70 ± 4.24	0.93 ± 0.03	42.62
			10	14.02 ± 5.66	33.10 ± 4.33	0.92 ±0.00	66.76

4.3 Discussion on Results

We emphasize that with the simulated dataset in our experiments, we can track the type of each features that is selected by different algorithms and know the exact performance irrespective of the accuracy estimates from MSE, henceforth measure of deviance, that can be misleading in reality. A few observations are noteworthy in Table 1.

First, comparing SOFS with other FS methods in terms of computation time, TPR, average performance measures, and their variability shows that it performs better in one or multiple aspects than others. RFE is faster and may seem to be more successful in selecting fewer features at first, but its near 0 TPR suggests that it fails to select the correct features. Also, its high variability reflects its random behavior across macro-replications. GAFS has a similar $\bar{f}$ or deviance measure to SOFS, but with higher risk (standard deviation) and less TPR. SOFS is more efficient than GAFS, although they both use Genetic Algorithms. Additionally, GAFS chooses more features on average than SOFS. LASSO has comparable time with RFE,

and a comparable TPR and deviance measure to SOFS. However, SOFS selects significantly fewer features than LASSO, without sacrificing accuracy.

Second, comparing SOFS with different values for R and B shows that the deviance measure does not significantly change with more bootstraps. Still, the number of selected features improves, implying that a better solution subset is obtained without losing accuracy. Note that based on $\hat{c}$ estimation in (6), $R = 1$ implies $\hat{c} = 0$, therefore, we cannot see the effect of the variance reduction in this case.

Third, with the objective function as average, we can observe that more bootstraps result in smaller subsets in some cases and larger or unchanged subsets in others. With the 90% quantile objective function, not only the subset size consistently decreases with more bootstraps, the computation time of almost all cases is less than the corresponding cases with the average objective function. This observation proves that the variance reduction adjustment effectively reduces the noise around the solutions and speeds the convergence. Although with the 90% quantile objective, optimal subsets are larger on average than those from the average objective function, the difference is not statistically significant. More notably, as B grows with a fixed R, variability across macro-replications lowers, implying more robustness in the results and more trust in the algorithm. A direct cost of more micro-replications is larger computation time; however, that does not increase linearly, suggesting that the optimization becomes more efficient due to better estimations. These results confirm that including input uncertainty and making appropriate adjustments achieves the goal of this study.

In another experiment, we focus on comparing LASSO and SOFS more closely. In a randomly singled out macro-replication using identical data subsets for both methods, we summarize the types of features selected by each in Table 2. While the number of contributing features selected by each is the same in this macro-replication, SOFS selects at most half as many noise features (features that are completely uninformative in reality) or redundant features (those only correlated with one or few of the main real features) selected by LASSO. Another interesting observation across all macro-replications reveals that one of the 12 contributing features, z_{19}, is the most challenging to identify. Across all methods, this feature only appears in a few macro-replications that suggests the high value of detecting it that is only achieved by SOFS (in cases with TPR above 0.92). Moreover, LASSO has higher variability across macro-replications, which is evident in its larger subset size standard deviation and deviance standard deviation. Lastly, LASSO's linearity assumption is a limitation of this method. At the same time, SOFS does not necessarily assume a linear relationship between the features and the response when $r(\cdot)$ is any learning algorithms other than linear regression. In other words, SOFS allows any parametric or nonparametric fitting that can be separately determined for the dataset of interest.

Table 2: Comparison between selected features by an instance of SOFS and LASSO in a macro-replication.

FS Method	**real +**	**redundant +**	**noise =**	**Total**
LASSO	11	4	29	44
SOFS(20,5)	11	2	12	25

5 CONCLUSION

In this paper, we propose an SO based FS algorithm, which searches for the best and most set of features by incorporating the input uncertainty that describes how the variability in the data propagates in the outcome of the prediction models. This problem, which commonly appears in practice and with the emergence of big data, is a data-driven stochastic optimization, currently formulated with binary decision variables. Genetic Algorithm is an easily implementable derivative-free solver, while any other search algorithm can replace it in the SOFS framework. Furthermore, it can infer about the robustness and reliability of the results. The adjusted variance reduction technique increases efficiency, although more extensive research on the proposed estimators' effect on the solution quality and reliability remains for future research. The results of

experimentation on a simulated dataset reveal a significant advantage over the existing optimization-based FS methods. A valuable investigation seeks adaptive sampling rules with incorporated input uncertainty that reduce the total number of micro-replications and attain comparatively good results within a shorter computation time and without sacrificing the robustness. Moreover, a deeper dive in model discrepancy and characterization of bias and variance of each source of uncertainty in the analysis is an area with little previous research.

REFERENCES

Barton, R. R., H. Lam, and E. Song. 2018. "Revisiting Direct Bootstrap Resampling for Input Model Uncertainty". In *Proceedings of the 2018 Winter Simulation Conference*, edited by M. Rabe, A. Juan, N. Mustafee, A. Skoogh, S. Jain, and B. Johansson, 1635–1645. Gothenburg, Sweden: Institute of Electrical and Electronics Engineers, Inc.

Cano, J. R., F. Herrera, and M. Lozano. 2003. "Using Evolutionary Algorithms as Instance Selection for Data Reduction in KDD: An Experimental Study". *Institute of Electrical and Electronics Engineers, Inc. transactions on evolutionary computation* 7(6):561–575.

Chen, Y.-W., and C.-J. Lin. 2006. "Combining SVMs with Various Feature Selection Strategies". In *Feature Extraction. Studies in Fuzziness and Soft Computing*, edited by G. I., N. M., G. S., and Z. L.A., Volume 207, 315–324. Berlin, Heidelberg: Springer.

Derrac, J., S. García, and F. Herrera. 2012. "A Survey on Evolutionary Instance Selection and Generation". In *Modeling, Analysis, and Applications in Metaheuristic Computing: Advancements and Trends*, 233–266. IGI Global.

Efron, B., and R. Tibshirani. 1997. "Improvements on Cross-validation: The 632+ Bootstrap Method". *Journal of the American Statistical Association* 92(438):548–560.

Guyon, I., and A. Elisseeff. 2003. "An Introduction to Variable and Feature Selection". *Journal of Machine Learning Research* 3:1157–1182.

Guyon, I., J. Weston, S. Barnhill, and V. Vapnik. 2002. "Gene Selection for Cancer Classification Using Support Vector Machines". *Machine Learning* 46(1-3):389–422.

Kennedy, M. C., and A. O'Hagan. 2001. "Bayesian Calibration of Computer Models". *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 63(3):425–464.

Kim, S., R. Pasupathy, and S. G. Henderson. 2015. "A Guide to Sample Average Approximation". In *Handbook of Simulation Optimization. International Series in Operations Research Management Science*, edited by F. M., Volume 216, 207–243. New York, NY: Springer.

Kudo, M., and J. Sklansky. 2000. "Comparison of Algorithms that Select Features for Pattern Classifiers". *Pattern recognition* 33(1):25–41.

Kuhn, M., and K. Johnson. 2013. *Applied Predictive Modeling*, Volume 26. New York, NY: Springer-Verlag.

Li, R., J. Lu, Y. Zhang, and T. Zhao. 2010. "Dynamic Adaboost Learning with Feature Selection Based on Parallel Genetic Algorithm for Image Annotation". *Knowledge-Based Systems* 23(3):195–201.

Mak, W.-K., D. P. Morton, and R. K. Wood. 1999. "Monte Carlo Bounding Techniques for Determining Solution Quality in Stochastic Programs". *Operations Research Letters* 24(1-2):47–56.

Nelder, J. A., and R. W. Wedderburn. 1972. "Generalized Linear Models". *Journal of the Royal Statistical Society: Series A (General)* 135(3):370–384.

Singh, D., S. Appavu Alias Balamurugan, and E. Jebamalar Leavline. 2016. "Literature Review on Feature Selection Methods for High-Dimensional Data". *International Journal of Computer Applications* 975(1):9–17.

Tibshirani, R. 1996. "Regression Shrinkage and Selection via the Lasso". *Journal of the Royal Statistical Society. Series B (Methodological)* 58(1):267 – 288.

Van der Laan, M. J., E. C. Polley, and A. E. Hubbard. 2007. "Super Learner". *Statistical applications in genetics and molecular biology* 6(1).

AUTHOR BIOGRAPHIES

KIMIA VAHDAT is a third year Ph.D. student at the ISE department of NC State University, pursuing a minor in statistics. Her main research is in the combination of stochastic optimization with data analysis. She received her B.Sc. in Industrial Engineering in 2018 from Sharif University of Technology, Tehran, Iran. Her e-mail address is kvahdat@ncsu.edu.

SARA SHASHAANI is an assistant professor in the Edward P. Fitts Department of Industrial and Systems Engineering at North Carolina State University. Her research interests are probabilistic data-driven modeling and simulation optimization. Her email address is sshasha2@ncsu.edu.