Stochastic Local Search for Efficient Hybrid Feature Selection

Ole Jakob Mengshoel
NTNU & CMU

Tong Yu
CMU

Jon Riege
NTNU

Eirik Flogard
NLIA & NTNU

ABSTRACT

There is a need to study not only accuracy but also computational cost in machine learning. Focusing on both accuracy and computational cost of feature selection, we develop and test stochastic local search (SLS) heuristics for hybrid feature selection.

CCS CONCEPTS

• **Computing methodologies** → **Randomized search**; **Discrete space search**; **Heuristic function construction**; *Feature selection*;

KEYWORDS

Pseudo-Boolean functions, feature selection, optimization, stochastic local search, wrapper, filter

ACM Reference Format:
Ole Jakob Mengshoel, Tong Yu, Jon Riege, and Eirik Flogard. 2021. Stochastic Local Search for Efficient Hybrid Feature Selection. In *2021 Genetic and Evolutionary Computation Conference Companion (GECCO '21 Companion), July 10–14, 2021, Lille, France.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3449726.3459438

1 INTRODUCTION

Context. We study search over bit-strings $\mathbb{B} = \{0,1\}^n$. Fitness f is a pseudo-boolean function (PBF) that maps from $\mathbb{B}$ to the non-negative real numbers $\mathbb{R}_{\geq 0}$. We seek to optimize (without loss of generality, maximize) the fitness function f:

$$\boldsymbol{b}^* = \arg\max_{\boldsymbol{b}\in\mathbb{B}} f(\boldsymbol{b}) . \quad (1)$$

In feature selection (FS) for machine learning (ML), each bit in a bitstring $\boldsymbol{b} \in \mathbb{B}$ indicates whether a feature is included "1" or not "0" when an ML model is induced. And f is a measure of ML model quality, in our case classifier accuracy [13, 15].

Typical benefits of FS include improved accuracy, interpretability, and computational efficiency of the resulting ML model [3, 9, 11]. For example, the Naive Bayes classifier's accuracy suffers if correlated features are used [9], and FS can help by removing them. In FS, one distinguishes between filter, wrapper, and embedded methods [2, 9]. The filter approach selects features in a preprocessing step, and the features selected do not depend on the ML algorithm used. The wrapper approach, in contrast, uses an ML algorithm as part of FS. To hybridize the filter and wrapper approaches to FS, we study an SLS algorithm in this paper, SLS4F.

Problem. We are interested in the problem of scaling SLS to handle large FS problems; large both in the sense of number of features and number of samples. What is the time it takes for an SLS algorithm to find $\boldsymbol{b}^*$ or a good approximation? There are several factors. First, it depends on the time $g(\boldsymbol{b})$ it takes to evaluate $f(\boldsymbol{b})$, where $\boldsymbol{b} \in \mathbb{B}$. Here, g maps from $\mathbb{B}$ to $\mathbb{R}_{\geq 0}$. Second, an SLS algorithm will in general evaluate $f(\boldsymbol{b})$ for many $\boldsymbol{b} \in \mathbb{B}$ when searching. Each state in an FS search process represents a potentially computationally costly (often on the order of seconds or minutes) induction over a potentially large and high-dimensional dataset. In contrast, computation time for each state of traditional SLS [6, 12] is fast (often on the order of microseconds or milliseconds). In addition, $g(\boldsymbol{b})$ may vary dramatically over the FS search space.

Contribution. Efficient hybrid FS using SLS needs to consider both maximizing ML accuracy and handling the computational cost of FS; that is our focus in this brief paper.

2 SLS4FS: STOCHASTIC LOCAL SEARCH

Input. The SLS4FS algorithm takes these *inputs*: probability of restart P_r and of noise P_n, adaptation rate of restart α_r and noise α_n, dataset D (with number of instances d and features n), ML algorithm L, filter F, fitness function $f(\boldsymbol{b})$, time limit τ, and number of neighbors N_r. For FS problems, f is defined via L as follows: For a subset $\boldsymbol{b}$, we run L on D using the features indicated by $\boldsymbol{b}$; $f(\boldsymbol{b})$ gives the estimated accuracy of the learned model [9].

Search. SLS4FS starts, using Restart(F;D), from a random initial state $\boldsymbol{b} \in \mathbb{B}$. Restart($F$;$D$) initializes a bitstring using a filter method F on dataset D. SLS4FS searches $\mathbb{B}$ while optimizing f. Let $\overline{P}_r = 1 - P_r$ and $\overline{P}_n = 1 - P_n$. SLS4FS performs either: a *greedy step* with probability $\overline{P}_r\overline{P}_n$; a *noise step* with probability $\overline{P}_r P_n$; or a *restart* step with probability P_r. During search, SLS4FS keeps track of a best-so-far $\boldsymbol{b}^+$. If, for the i-th search step $f(\boldsymbol{b}) > f(\boldsymbol{b}^+)$, then $\boldsymbol{b}$ is recorded as $\boldsymbol{b}^+$, the new best-so-far.[1]

Our SLS algorithm, SLS4FS, relates to existing research on SLS [7, 12–14, 18]. SLS4FS scales the most closely related algorithm, MarkovSLS [13], by adding three heuristics: (i) *FS filters*, (ii) *randomized neighborhoods*, and (iii) *soft greedy search*.

(i) *FS filters.* When a filter F is applied to a dataset for initialization or restart, a score is calculated for each feature. For each bit b_i, if the corresponding feature's score is in the 90th percentile, $b_i = 1$, else $b_i = 0$. We tested the runtime and accuracy of well-known and

[1] We distinguish between SLS4FS variants, depending on how hyperparameters α_r and α_n are set. When $\alpha_r = \alpha_n = 0$, in *static* SLS4FS, they are set via offline tuning. Otherwise, in *adaptive* SLS4FS, online control [10] is used. In offline tuning is the probability parameters P_r and P_n are fixed; this enables analysis via homogeneous Markov chains [12, 13]. Adaptive SLS4FS provides greater adaptivity to the problem and search process at hand but goes beyond homogeneous Markov chains.

GECCO '21 Companion, July 10–14, 2021, Lille, France

ACM ISBN 978-1-4503-8351-6/21/07.
https://doi.org/10.1145/3449726.3459438

prominent FS filters [1, 17]. Overall, chi-squared (F_{χ^2}) had the best performance in terms of runtime on almost every dataset. It also had high mean accuracy for the datasets. Thus, for our experiments we use F_{χ^2} and F_{0s}. F_{0s} means starting with all-0s.

(ii) *Randomized neighborhood.* Let $N_r \in \mathbb{N}^+$ with $0 < N_r \leq n$. A randomized neighborhood of size N_r is defined as a set $N(\boldsymbol{b}, N_r)$:

$$N(\boldsymbol{b}, N_r) = \{\boldsymbol{b'} \in N(\boldsymbol{b}) | \boldsymbol{b'} \text{ is picked from } N(\boldsymbol{b})\}, \quad (2)$$

such that $|N(\boldsymbol{b}, N_r)| = N_r$. Here, "picked" means "picked uniformly at random without replacement."

(iii) *Soft greedy.* A greedy step Greedy($\boldsymbol{b}$, N_r, f; L, D) starts from $\boldsymbol{b}$ and seeks among $\boldsymbol{b'} \in N(\boldsymbol{b}, N_r)$ for a state that maximizes the objective function $f(\cdot)$. This is done by applying L to D. If there is a tie in $f(\cdot)$ among maximizing neighbors in $N(\boldsymbol{b}, N_r)$, one of these is picked uniformly at random. A *strict* greedy step stays with $\boldsymbol{b}$ if $f(\boldsymbol{b}) \geq f(\boldsymbol{b'})$ for all $\boldsymbol{b'} \in N(\boldsymbol{b}, N_r)$, while a *soft* greedy step always moves to a best-fit neighbor.

Termination and Output. Upon termination, SLS4FS *outputs* $\boldsymbol{b}^+$ as an approximation to $\boldsymbol{b}^*$. In this work, SLS4FS terminates after a given wallclock time limit has passed.

Dataset	ML	Feat. n	Inst. d	Time τ (s)	N_r
breast cancer (UCI)	SVM	9	700	100	1
m-of-n-3-7-10 (UCI)	SVM	10	1 324	100	1
madelon [4]	DT	500	2 000	100	50
bioresponse [8]	NB	1 776	3 751	100	178
checklists	DT	580	63 634	100	58
gas-drift (UCI)	DT	128	13 910	100	13
crime (UCI)	NB	124	2 215	100	13

Table 1: Experimental information for the 7 datasets, with varying number of features ("Feat.") and instances ("Inst.").

Algorithm	Configuration	Mean	R
RFE [5]	model = linear SVM	0.734	9
ForwardSel.	N/A	0.803	8
BackwardSel.	N/A	0.826	6
AdaptiveNoise [6]	$\phi = 0.2$, $\theta = 1/6$	0.834	5
AdaptiveSLS [13]	$P_n = 0.0$, $P_r = 0.0$, $\alpha_n, \alpha_r = 0.32$	0.844	4
SoftSLS [13]	$P_n = 0.5$, $P_r = 1/n$, $\alpha_n, \alpha_r = 0.0$	0.856	3
SLS4FS/F_{χ^2}	$P_n = 0.5$, $P_r = 0.0$, $\alpha_n, \alpha_r = 0.0$, F_{χ^2}	**0.862**	**2**
SLS4FS/$P_r = 0.1$	$P_n = 0.5$, $P_r = 0.1$, $\alpha_n, \alpha_r = 0.0$, F_{χ^2}	*0.856*	*3*
SLS4FS/F_{0s}	$P_n = 0.5$, $P_r = 0.0$, $\alpha_n, \alpha_r = 0.0$, F_{0s}	***0.878***	***1***
SLS4FS/adaptive	$P_n = 0.0$, $P_r = 0.0$, $\alpha_n, \alpha_r = 0.32$, F_{χ^2}	0.807	7

Table 2: 10 FS algorithms, their configurations, and results in experiments. These FS algorithms are well-known, from the literature, or described in this paper. SLS4FS uses soft Greedy and $N_r = n/10$. The resulting mean accuracies ("Mean"–higher is better) and ranks ("R"–lower is better) are based for the data and settings in Table 1. The 3 best algorithms are highlighted. Overall, SLS4FS (F_{0s}) is best.

3 SLS4FS: EXPERIMENTAL RESULTS

Goal. How does SLS4FS compare to other FS algorithms, including those using local search?

Method and Data. Different FS algorithms, including SLS4FS configurations, are benchmarked on several problems, see Table 1.[2] Each problem consists of a dataset, an ML model, a time limit and a neighborhood ratio.[3] The ML models considered are the classifiers Decision Tree (DT), Naive Bayes (NB), and Support Vector Machine (SVM). Each fitness evaluation is done by using in D only features $\boldsymbol{b}$, initializing L, training with L on 2/3 of the dataset, and calculating the accuracy $f(\boldsymbol{b})$ using the other 1/3. The accuracy $f(\boldsymbol{b}^+)$ of the best feature subset $\boldsymbol{b}^+$ for each algorithm is recorded.

Results and Discussion. Algorithm configurations and results are summarised in Table 2.[4] The top 3 performers, ranked by mean accuracy (in the right-most column), are all variations of SLS4FS. SLS4FS appears to be quite robust across all problems compared to existing algorithms. For example, ForwardSelection achieves the highest accuracy for 3 problems but is far behind for other problems, leading it to be ranked as one of the last overall.[5]

REFERENCES

[1] A. Bommert, X. Sun, B. Bischl, J. Rahnenführer, and M. Lang. 2019. Benchmark for filter methods for feature selection in high-dimensional classification data. *Computational Statistics & Data Analysis* 143 (2019), 1–19.

[2] J. Chen, M. Stern, M. J. Wainwright, and M. I. Jordan. 2017. Kernel feature selection via conditional covariance minimization. In *NeurIPS*. 6946–6955.

[3] I. Guyon and A. Elisseeff. 2003. An introduction to variable and feature selection. *JMLR* 3 (2003), 1157–1182.

[4] I. Guyon, S. Gunn, A. Ben-Hur, and G. Dror. 2004. Result analysis of the NIPS 2003 feature selection challenge. In *NIPS*. 545–552.

[5] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. 2002. Gene Selection for Cancer Classification Using Support Vector Machines. *Machine Learning* 46 (2002), 389–422. https://doi.org/10.1023/A:1012487302797

[6] H. H. Hoos. 2002. An Adaptive Noise Mechanism for WalkSAT. In *AAAI*. 655–660.

[7] H. H. Hoos. 2002. A mixture-model for the behaviour of SLS algorithms for SAT. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence (AAAI-02)*. Edmonton, Alberta, Canada, 661–667.

[8] Boehringer Ingelheim. 2012. Predicting a Biological Response. (2012). https://www.kaggle.com/c/bioresponse/data

[9] R. Kohavi and G. H. John. 1997. Wrappers for feature subset selection. *Artificial Intelligence* 97, 1-2 (1997), 273–324.

[10] G. Krafotias, M. Hoogendoorn, and A. E. Eiben. 2015. Parameter Control in Evolutionary Algorithms: Trends and Challenges. *IEEE Transactions on Evolutionary Computation* 19, 2 (2015), 167–187.

[11] J. Li, K. Cheng, S. Wang, F. Morstatter, R. P. Trevino, J. Tang, and H. Liu. 2017. Feature Selection: A Data Perspective. *ACM Compututing Surveys* 50, 6 (2017).

[12] O. J. Mengshoel. 2008. Understanding the Role of Noise in Stochastic Local Search: Analysis and Experiments. *Artificial Intelligence* 172, 8-9 (2008), 955–990.

[13] O. J. Mengshoel, Y. Ahres, and T. Yu. 2016. Markov Chain Analysis of Noise and Restart in Stochastic Local Search. In *IJCAI*. 639–646.

[14] O. J. Mengshoel, D. C. Wilkins, and D. Roth. 2011. Initialization and Restart in Stochastic Local Search: Computing a Most Probable Explanation in Bayesian Networks. *IEEE TKDE* 23, 2 (2011), 235–247.

[15] O. J. Mengshoel, T. Yu, and M. Zeng. 2020. Stochastic Local Search and Machine Learning: From Theory to Application and Vice Versa. In *ECAI*. 2919–2920.

[16] M. Själander, M. Jahre, G. Tufte, and N. Reissmann. 2019. EPIC: An Energy-Efficient, High-Performance GPGPU Computing Research Infrastructure. (2019). arXiv:cs.DC/1912.05848

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

[18] T. Weise, Z. Wu, and M. Wagner. 2019. An Improved Generic Bet-and-Run Strategy for Speeding Up Stochastic Local Search. In *AAAI*. 2395–2402.

[2] This includes a new dataset, checklists, which consists of inspections conducted by the Norwegian Labor Inspection Authority (NLIA). The goal of the problem is to predict the correct outcome (pass or fail) of the inspections.

[3] Algorithms are implemented in Python version 3.6 and the numpy and scikit-learn libraries are also used. Experiments are run on CPUs of the Idun cluster at NTNU under Linux CentOS [16]. Except for the checklists dataset, where a Dell XPS 15 9570 with Windows 10, an i9 8950hk processor and 32GB RAM was used for legal reasons.

[4] Hyperparameters, including $\alpha_n = \alpha_r = 0.32$ used for AdaptiveSLS and SLS4FS (adaptive), were optimized empirically, in pilot studies, using synthetic and real-world data and kept constant in these experiments.

[5] For the checklists dataset, the most important identified features were in general checklist Id, industry code, and location. Among the best performing configurations, there were differences in the number of features that were found. SLS4FS/F_{0s} found 6 times as many features as ForwardSelection, AdaptiveSLS and SoftSLS in average. By using a randomized neighborhood, SLS4FS is able to explore a larger part of the search space which yields more features with approximatly the same computational cost.