2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems

Efficient Randomized Feature Selection Algorithms

Zigeng Wang, Sanguthevar Rajasekaran
Computer Science and Engineering, University of Connecticut
371 Fairfield Road, Storrs, CT 06269, USA
Email: {zigeng.wang,sanguthevar.rajasekaran}@uconn.edu

***Abstract*—Feature selection is a core problem in machine learning. It plays an important role in making efficient and explainable machine-driven decisions. Embedded feature selection methods, such as decision trees and LASSO, suffer from learner dependency and cannot be applied well to many popular learners. Wrapper methods, which fit arbitrary learning models, are receiving growing interests in many scientific fields. In order to effectively search relevant features in wrapper methods, many randomized schemes have been proposed. In this paper, we present efficient randomized feature selection algorithms empowered by automatic breadth searching and attention searching adjustments. Our schemes are generic and highly parallelizable in nature and can be easily applied to many related algorithms. Theoretical analysis proves the efficiency of our algorithms. Extensive experiments on synthetic and real dataset show that our techniques achieve significant improvements in the selected features' quality and selection time.**

***Index Terms*—Feature Selection, Randomized Algorithms, Efficient Selection**

I. Introduction

In the Information Age, we see a data explosion not only in data volume but also in data dimensions. High-dimensional data are often notorious to tackle. They consume a lot of computational resources and storage space, and also make learning models vulnerable to overfiting. Feature selection is a useful tool to reduce data dimensions and to make machine-driven decisions concise and explainable.

With respect to selection strategies, feature selection methods can be mainly categorized into three groups: filter, wrapper and embedded methods. Filter methods are unsupervised feature selection schemes, in which features are ranked based on certain statistical evaluation criteria. Filter methods are usually with high efficiency [1], but due to their unsupervised nature, the selected features can be far from optimal for a given learner [2]. Compared to filter methods, wrapper methods directly select features based on their prediction performance with respect to arbitrary predefined learning models. Different search algorithms are employed to identify the optimal subset of features with the best predicting performance. But conventional wrapper methods can take a long time to converge. Besides filter and wrapper methods, embedded methods are a hybrid type of methods which perform feature selection during the learner's training process. Common examples include decision tree and LASSO based models, which can simultaneously select features while training tree based models and generalized linear models. But embedded methods strongly depend on machine learning algorithms, which makes it difficult to be applied to many popular non-parametric models, such as KNNs [3]. Moreover, the LASSO based approaches are trained with the whole feature set and cannot scale well in terms of memory, given ultra-high-dimensional data [4].

TABLE I
Adavantages and Disadvantages of Feature Selection Methods

Category	Advantage	Disadvantage
Filter	Computationally Efficient	Low Accuracy
Wrapper	High Accuracy	Computationally Inefficient
Embedded	Less Extra Cost	Learner Dependent

In modern scientific applications, different non-parametric machine learning models have shown great potentials. For example, Gaussian process has been applied to polymer discovery and achieves accurate bandgap predictions based on hierarchical material descriptors [5], [6]. KNNs based approaches become state-of-the-art algorithms for metagenomic classification based on short DNA subsequences called K-mers [7], [8]. For these applications, the total number of feature dimensions can be from hundreds to millions, giving rise to an urgent need to select important features from them. In view of the discussion above, wrapper methods become the most suitable feature selection schemes.

Conventional wrapper methods either use exponential branch and bound [9], [10] or sequential greedy searching strategies [11], [12] to locate important feature set. But these approaches usually suffer from local optima or the non-monotonicity of the features. In order to overcome these issues, different randomized population based optimization schemes, such as genetic and particle swarm algorithms [13], [14], have been applied to feature selection. These approaches can achieve a better accuracy but with a comparatively high computation overhead. Thus, concise randomized approaches currently receive a great research attention. Different randomized feature elimination and selection algorithms [15]–[17] have been developed and achieve a promising performance.

In this paper, we propose efficient randomized feature selection algorithms embedding three speed-up techniques to further enhance the selection process with respect to accuracy. The first **Semi-Randomized Selection** technique accelerates the feature candidate discovery by automatically adjusting the local searching breadth in each iteration based on the quality of the discovered important features. The second **Warm Start** technique warmly initializes a feature subset and saves a lot of computation when the algorithm starts. The third **Cool Down** technique optimizes the searching attention to highly potential features according to their cooling down probabilities based on

978-1-7281-2058-4/19/$31.00 ©2019 IEEE
DOI 10.1109/HPCC/SmartCity/DSS.2019.00117

feature evaluation history. Experiments show that our proposed algorithms achieve significant improvements in the quality of the features selected and total running time.

The contributions of this paper are as the following: 1) the proposed speed-up techniques can be easily applied to different randomized feature selection algorithms for arbitrary predefined learning models; 2) our algorithms naturally support parallelization and can achieve an asymptotically optimal speedup; 3) our algorithms' memory usage is independent of the given feature dimension and is highly scalable for high-dimensional data; and 4) we provide a detailed convergence proof for the proposed feature selection algorithms.

II. Related Works

In this section, we introduce the basics of wrapper methods. Then, we discuss some fundamental deterministic wrapper algorithms and recent randomized approaches.

A. Wrapper Feature Selection Strategy

Wrapper methods select important feature sets based on their predictive performance. As shown in Fig. 1, a generic wrapper method includes two main steps: (1) search for a subset of features as a candidate feature set; (2) evaluate the candidate set with the predefined learner. These two steps are repeated multiple times until the stopping condition is met.

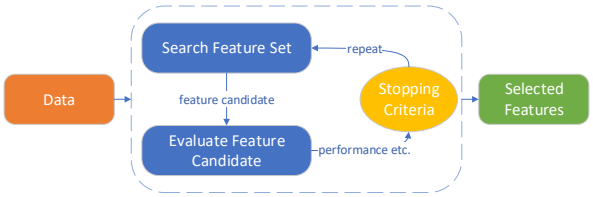


Fig. 1. An Illustrative Framework of Wrapper Feature Selection Methods

B. Selection of Candidate Subsets

Candidate subset selection is a focal component in wrapper methods. Given a dataset with n dimensions, the total number of possible feature subsets can be up to 2^n. An exhaustive search becomes impractical when n is very large. Thus, many heuristic searching algorithms have been proposed for probabilistic optimal feature subset discovery. With respect to searching strategies, we can classify wrapper methods into two categories, deterministic and randomized feature selections.

1) Deterministic feature selection: Deterministic methods employ deterministic searching schemes and generate fixed outputs. Branch and bound is a common approach used in feature selection [9], [10]. However, this approach suffers from exponential time complexity and assumes monotonicity.

Sequential feature selections form another main trend. Traditional sequential forward search (SFS) and sequential backward search (SBS) employ greedy hill climbing strategies and slightly modify the current subset in each move [12]. Inspired by SFS and SBS, sequential floating search algorithm [18] and *replacing-the-weak-feature* strategy [11] have been proposed to achieve a faster convergence, but these greedy approaches usually suffer from getting stuck in local optima.

2) Randomized feature selection: In order to avoid getting stuck in local optima and to speedup the selection process, different randomized approaches have been developed.

One type of these approaches is by employing existing conventional population based genetic or particle swarm algorithms [13], [14], in which random subsets of features are generated and refreshed iteratively based on the optimization algorithms. These approaches can usually locate a better feature subset [19]. But for some light-weight applications, these types of approaches are not perfectly suitable, due to their high computation overhead of the embedded optimization schemes. As a result, many simplified randomized approaches designed specially for feature selection have been developed.

Stracuzzi et. al proposed a randomized variable elimination algorithm (RVE) [15]. RVE searches backwards with eliminating one or more random features in each iteration. The search through a sparse sampling of the feature space results fewer subset evaluations. RVE can be very effective when there exist many irrelevant or redundant features, but can be memory intensive when the initial data dimension is high.

Saha et al. developed a novel randomized feature selection algorithm (RFS) [16]. The RFS algorithm starts with a randomly picked feature subset and moves to a neighboring feature subset by adding and/or deleting one feature based on the outcomes of coin flips and the neighbor's prediction accuracy. RFS selects features with a fine granularity on single feature operation. Comparing to RVE, RFS is specially suitable for high dimensional data due to its memory efficiency by evaluating only a subset of features.

III. Our Algorithm

Based on the existing randomized wrapper algorithms [15], [16], we propose our efficient randomized feature selection algorithms empowered by three novel speedup and accuracy enhancement techniques. The techniques include

- **Semi-Randomized Selection** which locates feature candidates with high accuracy improvement potentials by ranking a subset of neighbors of the current feature set in order to construct a more informational feature subset and to accelerate the convergence;
- **Warm Start** which warmly initializes the selection feature subset instead of directly starting with a random subset (e.g. in RFS [16]). Implanted with feature elimination technique, warm start enables a fast initializing speed for datasets with different characteristics;
- **Cool Down** which adaptively adjusts the searching attention and optimizes the computational cost on evaluating temporarily weak/strong features by setting a cooling down probability among all the features based on their related performance evaluation history.

A. Semi-Randomized Selection

In feature selection, a better gradient results in more meaningful features. In RFS [16], the current feature subset F'

moves to its neighboring subset F'' if F'' has a better accuracy. However, this accuracy improvement can be very tiny and many less important features with little accuracy contribution are selected. As a result, this $pure\text{-}randomized$ single feature operation slows down the selection process. Especially when the learner is non-deterministic or the features are noisy, irrelevant features can interact rapidly in each searching iteration. This phenomenon can be more severe at the beginning of the feature search, which will be detailed in Subsection III-B.

In order to efficiently locate a informational subset of features, we design a novel $semi\text{-}randomized$ feature selection algorithm (Semi-RFS). Instead of picking one random feature in each iteration, Semi-RFS picks a group of features and selects the locally optimal combination as the new candidate set F''. With a high probability, the locally best feature contributes to an acuter angle to descend in the optimization process and the selected features contribute more to the accuracy, so that a more concise feature set can be efficiently constructed.

According to different applications, we design two types of Semi-RFS algorithms, namely static Semi-RFS and adaptive Semi-RFS (shown in Algorithm 1). Static Semi-RFS selects the optimal features within a randomly picked feature group (F_g and $F_{g'}$) of static size g and g', where g and g' denote the group sizes for features picked from the residual feature set $F - F'$ and feature set F'. The adaptive selection is by selecting from feature groups of sizes calculated by an adaptive function F_{semi}. F_{semi} adjusts the random feature group size based on the searching stages. At the initialization stage, the picked feature group size should be comparatively large in order to avoid adding less significant features into set F'. Because during initialization, set F' is pre-mature and many weak residual features are with high probability to be included. While in the final stage, a smaller feature group is recommended since the searching randomness will be beneficial for a potential further descend. A convergence proof and theoretical comparison are given in Section IV-A.

$$\begin{aligned} g &= |F - F'| \cdot \frac{1}{\beta + e^{\alpha \cdot N_{ni}}} \\ g' &= |F'| \cdot \frac{1}{\beta + e^{\alpha \cdot N_{ni}}} \end{aligned} \tag{1}$$

In order to adaptively adjust the group size, function F_{semi} is carefully designed based on searching stage estimation. At the beginning stage, a feature candidate set with a higher accuracy can be easily discovered in most iterations. While there may not be accuracy improvements in a number of consecutive iterations when the algorithm is approaching an end, since a better feature set candidate can be difficult to find due to the close-to-mature feature set F'. So, here we can have a coarse estimation of the searching stage based on the number of consecutive iterations with no accuracy improvements N_{ni}. A smaller or close to zero N_{ni} indicates the feature selection is at the beginning stage, while a larger N_{ni} indicates the search is close to the end stage. We calculate the size of the feature groups with respect to N_{ni} in Eqs. 1. Generalized sigmoid functions are used to compute the percentage of the features to be included in each feature group. β is used to control the initial group size and α is used to control the influence of this stage estimation. In Section V-B, we show through experiments that the sensitivities of β and α are very low.

Algorithm 1: Semi-RFS

Input: F' (initial feature subset), F (whole feature set) and L (learner)

Compute the accuracy A generated by L with feature subset F' ;
repeat
  *For adaptive case: compute group size g and g';
  Compute new accuracy A' of local best feature subset F'' by adding and/or deleting feature from feature group F_g and/or $F_{g'}$;
  if $A' > A$ **then**
    Update A and F' with A' and F'', then search from F';
  else
    With probability $1 - u$, update A and F' and search from F';
  end
until Stopping condition C_{stop} is satisfied;
Return F' ;

For the calculation of the number of consecutive no-improvement iterations N_{ni}, in order to maintain the stability of the system influenced by the impulse of a better feature subset discovery, we borrow the ideas from adaptive filters and compute N_{ni} based on its previous values.

$$N_{ni}(k) = \begin{cases} 0, & k = 0 \\ w \cdot n_{ni}(k) + (1 - w) \cdot N_{ni}(k-1), & k > 0 \end{cases} \tag{2}$$

In Eqs. 2, k denotes the index of the iteration and $n_{ni}(k)$ denotes the current number of consecutive no-improvement iterations up to the kth iteration. We can see that the value of $N_{ni}(k)$ is a weighted combination of the current $n_{ni}(k)$ and its previous value $N_{ni}(k-1)$. In this way, the group size calculation function will be more robust to the unpredictable discovery of the better feature sets. For simplicity, we omit the index k for majority of the cases for N_{ni}.

B. Warm Start

In wrapper methods, a nice initialized feature subset can save a significant amount of searching overhead. A pure random initial feature subset F' (i.e. in [16]) can be with many irrelevant features and an arbitrary neighbor of F' will be a high probability to improve the accuracy. In this way, F' can be frequently updated with tiny accuracy upgrades. From the optimization point of view, the starting point is far from optimal with a high probability which will make the search process difficult to converge due to the plateau. Thus, it is of importance to smartly locate the starting point of the selection with a warmly initialized preliminary feature subset F_{pre}.

The preliminary initial feature subset F_{pre} would be from experts, from low weight computational methods, or even from low weight feature selection algorithms. The initial feature subset possesses its general informative power to nicely represent and to describe the data characteristics. But more precisely, a perfect feature subset for a specific learner may not be perfect for another because the learnability and the learning strategy can vary from learner to learner. Thus, in the second step, the predefined learner should be involved to tune the preliminary initial feature set for a quick fit. We employ adaptive Semi-RFS on the preliminary features with a fine-grained feature group size calculation function F_{warm} shown in Eqs. 3 to efficiently fit the model.

$$\begin{aligned} g &= |F - F'| \cdot \frac{I_{warm}}{\beta \cdot I_{warm} + e^{\alpha \cdot N_{ni}}} \\ g' &= |F'| \cdot \frac{I_{warm}}{\beta \cdot I_{warm} + e^{\alpha \cdot N_{ni}}} \end{aligned} \tag{3}$$

Compared to Eqs. 1, we introduce another index I_{warm} to finely estimate the searching stages during the initialization to speed up the warm start. At the beginning stage, the number of consecutive no-improvement iterations N_{ni} can be fairly small or even close to zero and only using N_{ni} may not give a fine estimation of the searching stage. So, here we involve another parameter PCT_{imp} (the percentage of feature sets with accuracy improvement constructed with the previous feature group) for stage estimation. In Semi-RFS, a group of features (F_g and/or $F_{g'}$) are randomly picked in each iteration and feature candidate sets are constructed with each feature in the feature group for evaluation. A larger percentage of candidates that can improve the current accuracy indicates a higher probability that the warm start is at its beginning due to the weakness of the preliminary feature set F'. So, we formulate the positive correlation between I_{warm} and PCT_{imp} as the following,

$$I_{warm}(k) = \begin{cases} 1, & k = 0. \\ \frac{1}{1 - PCT_{imp}(k)}, & v' \in k > 0. \end{cases} \tag{4}$$

From Eqs. 3 and 4, we know the positive correlation among group size g (and g'), I_{warm} and PCT_{imp}. Similar to calculating N_{ni}, we also employ adaptive design to compute PCT_{imp}. $PCT_{imp}(k)$, the percentage at iteration step k, will be the weighted combination of the $exact$ percentage $pct_{imp}(k-1)$ (in step $k-1$) and its previous value $PCT_{imp}(k-1)$.

$$PCT_{imp}(k) = \begin{cases} 0, & k = 0. \\ pct_{imp}(k-1), & v' \in k = 1. \\ (1 - w') \cdot PCT_{imp}(k-1) & \\ \quad + w' \cdot pct_{imp}(k-1), & k > 1. \end{cases} \tag{5}$$

The two-step warm start initialization process is shown in Algorithm 2. It begins with a preliminary feature set. Then, the predefined learner will be involved together with the finely adaptive Semi-RFS for feature set tuning and initializing.

Besides the warm start with a subset of preliminary features, RVE [15] related feature elimination algorithm can also be empowered by warm start. The feature selection can start with the complete feature set and follow the same Semi-RFS structure. In order to enhance the feature elimination speed, biased coin tosses will be made to remove the features with a higher probability. It is worth to mention that, with the nature of using the whole feature space, RVE [15] should be carefully used for ultra-high dimensional data due to their potentially high memory usage to fit the entire feature space.

Algorithm 2: Warm Start Initial Feature Subset Selection

Input: F_{pre} (preliminary feature set), F (whole feature set) and L (learner)
Output: A subset F' of features ($F' \subset$F)
begin
  Compute the accuracy A generated by L with F_{pre};
  Set group size overriding function F_{warm};
  Call Semi-RFS with F_{warm}, F_{pre}, F and L;
  Return the return value F' of Semi-RFS;
end

C. Cool Down

In feature selection, weak features can waste the searching effort drastically. It is important to identify the temporarily weak features and adaptively adjust the searching attention. In [16] and [15], the generation of feature set candidates (by adding and/or removing features) is purely random so that all the features in the residual feature subset $F - F'$ are likely to be added into F' with equal probability and all the features in F' are equally likely to be deleted. The pure randomness makes the searching process memory-less and the history prediction accuracy information in the feature candidate evaluation process has not been fully taken into consideration when generating new feature candidates.

Taking pure feature addition as an example, we construct feature candidates by adding a random feature f into F' to get candidate F''. If F'' (equivalently, $F' + f$) performs no better than feature subset F', for the next round of addition, feature f is still equally likely to be selected with the other features in $F - F'$. However, the weakness of feature f has been proved in the previous evaluation and it cannot contribute to the accuracy with current subset F' given a deterministic learner. In this way, computational power is wasted on repeatedly evaluating temporarily weak features. It is desirable to develop a technique that nicely allocates the computation attention by fully utilizing the history evaluation results.

In order to fully leverage history evaluations, we design our cool down technique to efficiently adjust the attention to promising features in the addition and deletion operations. Instead of picking features with equal probabilities, the less-promising features will be assigned with a large cool-down factor to decrease their probability of being chosen. For the feature addition operation, if a feature f in $F - F'$ has been evaluated to be a temporarily weak feature ($f + F'$ results in a large accuracy decrease), a large value will be assigned

to the cool-down factor related to f. The same holds for the feature deletion operation, if a feature f' in F' is proved to be a temporarily strong feature (the deletion of f' from F' results in a large accuracy decrease), a large value will be assigned to the cool-down factor of f'. Eqs. 6 describe the relation of the cool-down factors and the probability of being chosen.

$$P(i) = \begin{cases} \frac{1/fac(i)}{\sum_{i=1,f_i \in F'}^{|F|} 1/fac(i)}, & \text{if } f_i \in F'. \\ \frac{1/fac(i)}{\sum_{i=1,f_i \in F-F'}^{|F|} 1/fac(i)}, & \text{if } f_i \in F - F'. \end{cases} \quad (6)$$

In Eqs. 6, we use f_i to denote the feature with index i in F and $fac(i)$ to denote the cool-down factor of feature f_i. $P(i)$ represents unified inverse relation between the probability that features f_i will be chosen into the feature group for local selection and their cool-down factor values.

Knowing that the whole construction of the selected feature set F' can change dramatically after several iterations, a temporarily weak/strong feature may not permanently be weak/strong. Thus, we need to gradually warm up their corresponding factors in order to increase their probabilities of being picked. So, we design the strategy shown in Algorithm 3.

Algorithm 3: Semi-RFS with Cool Down

Input: F' (initial feature subset), F (whole feature set) and L (learner)

Compute the accuracy A generated by L with feature subset F';
Initialize the cool down factors facs for all features in F;
repeat
  Compute feature group sizes g and g';
  Compute selection probabilities $P_{F'}$ and $P_{F-F'}$;
  Compute new accuracy A' of local best feature subset F'' generated with feature group F_g and/or $F_{g'}$ selected with probability $P_{F'}$ and $P_{F-F'}$;
  Decrease the all the facs greater than 1 by 1;
  Update facs for all previously evaluated features;
  Update A and F' based on A' and search from F';
until Stopping condition C_{stop} is satisfied;
Return F';

For illustration purpose, we combine the cool down together with the adaptive Semi-RFS. Since the warm start calculates probability solely based on the accuracy comparison, it can be also easily incorporated with the cool down technique.

IV. Analysis

A. Convergence Proof for Semi-RFS

We formulate Semi-RFS as a semi-random walk in a directed graph $G(V, E)$. Each node in V represents a subset of n features and there are in total 2^n nodes. For an arbitrary node v, edges are connecting to its neighbors v's. Semi-RFS starts from v in G and moves to a neighboring node v'. Since the next node visited depends only on the current node, we model the algorithm as a time-homogeneous Markov chain. For node v, let $A(v)$ be the accuracy of the feature set of node v, $N(v)$ be the set of neighbors of node v and N be the size of $N(v)$. We are using the same state transition probabilities as in RFS [16], $P_{vv'}$ from node v to v':

$$P_{vv'} = \begin{cases} 0, & v' \notin N(v). \\ min(1, c \cdot exp(A(v') - A(v))), & v' \in N(v). \end{cases} \quad (7)$$

where c is a constant for state transition adjustment.

RFS picks the next node v' purely randomly. And in Semi-RFS, the node v' is selected within a small feature group F_g. Let v be any node. For every neighbor v' of v, we assign a *rank* as follows. The rank of v' will be 1 if v' has the least accuracy improvement from among all the neighbors of v. The rank of v' will be 2 if it has the second least accuracy improvement over all the neighbors of v; and so on. We first investigate the rank of node v' with respect to the group size g and neighbor size N. We find that the random sampling lemma from randomized sorting [20] nicely fits our approach.

Lemma 4.1: Let F_g be a subset of features from the whole neighbor set $N(v)$. Let v'_{ar} be an arbitrary neighboring node of rank j in F_g. Let r_j be the rank of v'_{ar} in $N(v)$. Let α be a constant. Then,

$$Prob.\left(|r_j - j\frac{N}{g}| > \sqrt{4\alpha}\frac{N}{\sqrt{g}}\sqrt{\log N}\right) < N^{-\alpha} \quad (8)$$

We can see that the global rank r_j of an arbitrary node v'_{ar} with rank j in F_g is bounded. In Semi-RFS, we pick the node v' with the greatest accuracy improvement whose ranking is the value of group size g, so that

$$Prob.\left(N - r_g > \sqrt{4\alpha}\frac{N}{\sqrt{g}}\sqrt{\log N}\right) < N^{-\alpha} \quad (9)$$

From Eq. 9, the number of neighboring nodes which can achieve greater accuracy improvement than v' is tightly bounded given a comparatively large N. By only using a group size of $k \cdot logN$, we can further derive the bound as,

$$Prob.\left(N - r_g > \frac{N}{\sqrt{k/4\alpha}}\right) < N^{-\alpha} \quad (10)$$

where k is a constant for static Semi-RFS and a generally decreasing parameter based on Eq. 1 for adaptive Semi-RFS.

We can see, in RFS, the worst gradient that can be picked in each iteration is

$$\Delta = min_{v' \in N(v)} A(v') - A(v). \quad (11)$$

Comparatively in Semi-RFS, the gradient becomes

$$\Delta' = min_{v' \in N(v), and\ rank(v') > r_g - \frac{N}{\sqrt{k/4\alpha}}} A(v') - A(v) \quad (12)$$

with a high probability in each iteration. As a result, a steeper descent can be achieved in each iteration.

We follow the assumption in [16] and [21] that an algorithm is said to have converged if the underlying Markov chain had been in a globally optimal state at least once. Let the diameter of $G(V, E)$ be D and clearly the expectation of D is $\Theta(n)$ with

in total n features. And there must be a path from the starting node v_{start} to the optimal node v_{opt} of length $\leq D$.

Based on Eq. 7, if v' is a neighbor of v, the probability of a move from v to v' is normalized to be $\frac{1}{N}P_{vv'} \geq \frac{1}{N}exp(c\Delta')$. So that for any node v_{start}, the expected number of steps before v_{opt} is visited is

$$E(steps) \leq N^D \cdot exp(-c\sum_{i=1}^{D}\Delta_i') = [N \cdot exp(-c\overline{\Delta'})]^D. \quad (13)$$

Given an arbitrary integer m, we can easily prove by induction that Semi-RFS converges within $\leq 2m[N \cdot exp(-c\overline{\Delta'})]^D$ moves, with a probability of $\geq (1-2^{-m})$. Therefore, with a feature group size $k \cdot logN$, the total number of evaluations is $\leq 2m \cdot k \cdot logN[N \cdot exp(-c\overline{\Delta'})]^D$ with probability $\geq (1-2^{-m})$. Compared to the bound $2m[N \cdot exp(-c\overline{\Delta})]^D$ of RFS, Semi-RFS can be more efficient given the following condition:

$$\overline{\Delta'} - \overline{\Delta} \geq \frac{\log k + \log\log(N)}{c \cdot D}. \quad (14)$$

In Semi-RFS and RFS, a neighboring feature subset F'' is generated by adding one feature from $F - F'$ and/or deleting one feature of F'. Given a constant size or even a linear size of F' with respect to n, there can be at most $N = O(n^2)$ such neighbors. Since $D = \Theta(n)$, the condition above can easily be satisfied given a comparatively large n.

B. Parallelization of Semi-RFS

In the previous section, we have theoretically proved that Semi-RFS achieves a high sequential computational efficiency. In this subsection, we show the parallelization potentials of Semi-RFS to further boost the feature selection efficiency by using many processors.

To illustrate our approach, we employ the Parallel Random Access Machine (PRAM) model as an example. PRAM is a collection of RAMs working in synchrony where communication takes place with the help of a common block of shared memory [22]. Concurrent Read and Concurrent Write (CRCW) PRAM is a common type of it. We illustrate a paralleled version of Semi-RFS based on CRCW PRAM in Algorithm 4.

In wrapper methods, feature subset candidate evaluation consumes a majority of the overall runtime. Fortunately, the candidate evaluations in Semi-RFS naturally support parallelism. Each feature subset in the candidate groups can be evaluated by each individual processor in parallel. Given the group size $O(k \cdot \log N)$ and at most $N = O(n^2)$ in subsection IV-A, we can complete one group of evaluations in constant steps using $\log\ n$ CRCW PRAM processors. And selecting locally best feature subset can be done in $O(\log\log\log\ n)$ time using the parallel divide-and-conquer algorithm given $\log\ n$ CRCW PRAM processors in [23]. As a result, Parallel Semi-RFS is asymptotically work optimal, which reduces the sequential evaluation steps from $2m \cdot k \cdot \log N[N \cdot exp(-c\overline{\Delta'})]^D$ to $2m \cdot [N \cdot exp(-c\overline{\Delta'})]^D$ given $\log\ n$ CRCW PRAM processors.

Algorithm 4: Parallel Semi-RFS

Input: F' (initial feature subset), F (whole feature set) and L (learner)

```
Compute the accuracy A of feature set F' with learner L;
repeat
    Compute group size g and g';
    Compute accuracies A's for each feature subset candidates in parallel;
    Select the locally best feature subset candidate with respect to accuracy in parallel;
    Update A and F';
until Stopping condition C_stop is satisfied;
Return F' ;
```

V. Experimental Evaluation

In this section, we apply our approaches to NIPS 2003 feature selection challenge synthetic dataset [24] and compare with existing randomized feature selection algorithm [16]. Then, we conduct hyper-parameter sensitivity empirical study for adaptive Semi-RFS. Moreover, we apply our algorithm to real material descriptor discovery for polymer property prediction [6] and compare the feature selection results.

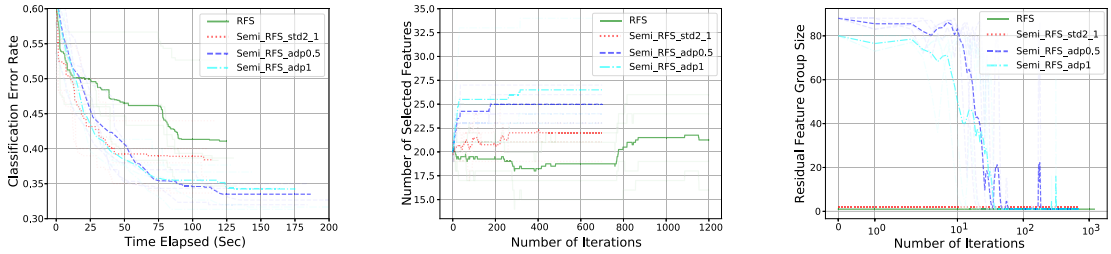
All the algorithms are evaluated on a Dell Precision Workstation T7910 running RHEL 7.0 on two sockets each containing 8 Dual Intel Xeon Processors E5-2667 (8C 16HT, 20MB Cache, 3.2GHz) and 256GB RAM.

A. Synthetic Data Analysis

Firstly, we employ our algorithms on NIPS 2003 feature selection challenge synthetic dataset generated by Scikit-Learn Toolbox [25] adapted from [24]. The dataset includes 300 examples and 500 features, in which 30 features are informative and 4 of the 30 informative features are redundant. There are in total 3 classes and there is 1 cluster for each class. We use Gradient Boosting Trees as the learner, and run the feature selection algorithms for 4 times with 5-fold cross-validation and calculate the average results.

We first compare static and adaptive Semi-RFS algorithms with RFS. For static Semi-RFS, we set the group size g from residual feature set $F - F'$ as 2; for adaptive Semi-RFS, we set α as 0.5, 1 and β as 5. We further use a universal group size 1 from F' for a fair comparison. Due to the imbalanced workloads to each iteration, we use elapsed time to be the x-axis for error rate comparison. In Fig. 2(a), leveraging better gradients, our Semi-RFS algorithm converges much faster. Due to the adaptive function on searching stage estimation, adaptive Semi-RFS algorithms automatically adjust the candidate group size (shown in Fig. 2(c)) and perform a faster descent. We can also infer from Fig. 2(b) that our Semi-RFS approaches locate more meaningful features that can contribute to the accuracy.

The warm start technique is then tested. In the figure legends, we use $warm$ to mark the algorithms embedded with warm start technique. Low-depth Extra Trees Classifier is used as the preliminary feature selection algorithm and

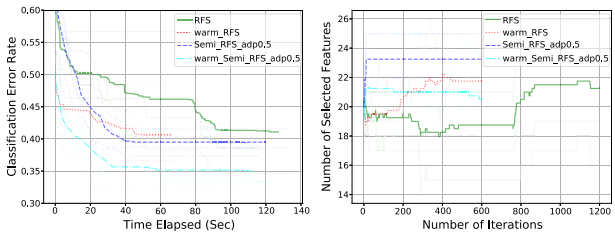


(a) Error Rate (b) Number of Selected Features (c) Group Size

Fig. 2. Comparison of Semi-RFS and RFS

(The highly transparent lines show the records for each run and the bold lines show the average. The same to all the following plots.)

locates a preliminary set of 20 features. Fig. 3(a) shows that the warm start greatly contributes to the initialization step of the randomized selection and accelerates the convergence for both Semi-RFS and RFS. We also compare the number of selected features in each iteration in Fig. 3(b). We can see that warm start not only contributes to the accuracy but also helps to well generalize the model. With fewer or similar number of selected features, algorithms empowered with warm start achieve a much higher prediction accuracy.



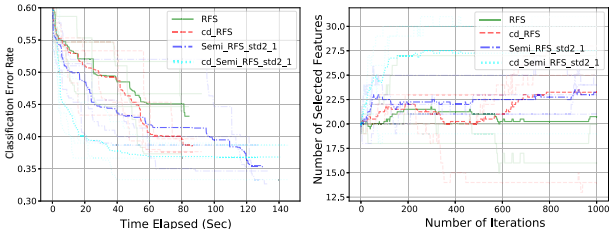
(a) Error Rate (b) Number of Selected Features

Fig. 3. Comparison of Warm Start, Semi-RFS and RFS

We thirdly evaluate the performance of the cool down approach. In the figure legends, we use cd to mark the algorithms embedded with cool down technique. For static Semi-RFS, we set the group size g from residual feature set $F-F'$ with 2 and set group size 1 to g' from F'. As we can see in Fig. 4(a), the cool down approach dramatically contributes to the convergence speed of both RFS and Semi-RFS. Cool down technique nicely adjusts the searching attention to high potential features and results in a higher accuracy and more selected meaningful features as shown in Fig. 4(b).

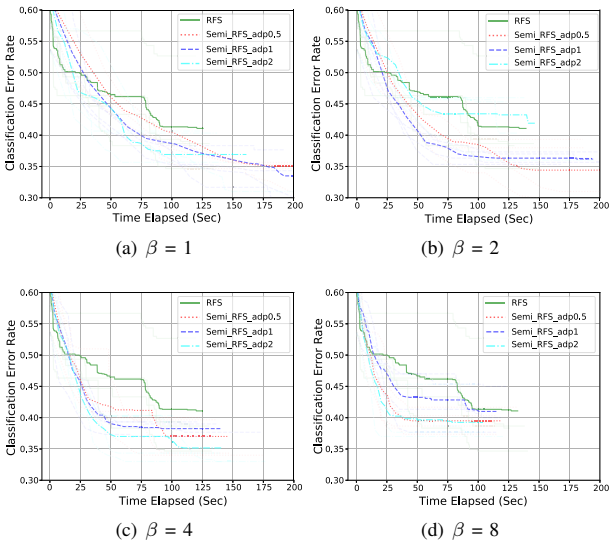
B. Hyper-Parameter Sensitivity Study in Adaptive Semi-RFS

In adaptive Semi-RFS, there are mainly two hyper-parameters β and α. β is used to control the initial group size and α is used to control the influence of this stage estimation. Tuning hyper-parameters is usually a tedious task. We conduct an empirical study on these hyper-parameters' sensitivity. We use the same experimental settings as in subsection V-A. We test different combinations of β and α and evaluate their convergence speed with respect to prediction performance.



(a) Error Rate (b) Number of Selected Features

Fig. 4. Comparison of Cool Down, Semi-RFS and RFS



(a) β = 1 (b) β = 2

(c) β = 4 (d) β = 8

Fig. 5. Classification Error Rate Comparison with Different α and β Values

As we can see in Fig. 4, we compare the performance with α values = 0.5, 1, 2 and with β values = 1, 2, 4 and 8. We can see that α is not value sensitive and there are no obvious differences with β = 1 or 2. But when β continues increasing, the performance may be degraded because more randomness is involved in the initialization stage and more weak features are selected. In general, both α and β are not sensitive, and even assigning $1s$ to both parameters can give a nice result.

TABLE II
RESULT COMPARISON WITH MATERIAL FINGERPRINTING DATA

Metric	No. of Features			RMSE			Running Time(Sec)	
Algorithms	FE [6]	RFS [16]	Semi-RFS	FE [6]	RFS [16]	Semi-RFS	RFS [16]	Semi-RFS
bandgap	88	**33.1**	33.3	0.47	0.466	**0.464**	2456	**1204**
dielectric	35	28.4	**26.7**	0.48	**0.471**	0.472	1942	**1015**

C. Materials Property Prediction

Predicting the properties of unknown materials is a central problem in Materials Genomics. One of the widely used approaches is to build a prediction model using the properties of known materials. Specifically, we have tested our algorithm for polymer property prediction. The whole dataset includes 242 features spanning hierarchical levels of descriptors of atomic-scale, quantitative structure-property relationship (QSPR) and morphological descriptors [6].

We compare our algorithm with RFS [16] and the state-of-the-art feature elimination (FE) approach designed for polymer property prediction [6]. We use the same Gaussian Process regressor as the learning model together with the same learning environment setup in [6]. We calculate the average results of 10 runs with 5-fold cross validation. For the FE, we directly use the results shown in [6] for a generous comparison.

From Table II, compared with FE, both RFS and Semi-RFS can locate a more concise set of important features with higher accuracy for both bandgap and dielectric constant predictions. Especially for the bandgap prediction, our algorithm locates a subset of 33 descriptors, which is less than half the size of the descriptors selected by FE with a slightly better accuracy. In terms of the running time, our Semi-RFS algorithm is much more efficient, only using approximately half of the running time of RFS. The efficiently discovered concise set of features enhance the material descriptors' explainability, which will also provide material scientists more insights into the connection between material properties and its descriptors.

VI. CONCLUSIONS

In this paper, we have presented computational and memory efficient randomized feature selection algorithms based on three speedup techniques. Our schemes are generic in nature and can be easily applied to different randomized feature selection algorithms for arbitrary predefined learning models. Our algorithms naturally support parallelization and come with a solid convergence proof. Experiments on benchmark datasets show that our algorithms achieve convincing improvements with respect to the prediction accuracy and the running time.

ACKNOWLEDGEMENTS

This work has been supported in part by the following NSF grants: 1447711, 1514357, 1743418, and 1843025.

REFERENCES

[1] Peng Xiao, Zigeng Wang, and Sanguthevar Rajasekaran. Novel speedup techniques for parallel singular value decomposition. In *2018 IEEE 20th International Conference on High Performance Computing and Communications (HPCC)*, pages 188–195. IEEE, 2018.

[2] Girish Chandrashekar and Ferat Sahin. A survey on feature selection methods. *Computers & Electrical Engineering*, 40(1):16–28, 2014.

[3] Valeria Fonti and Eduard Belitser. Feature selection using lasso. *VU Amsterdam Research Paper in Business Analytics*, 2017.

[4] John Duchi, Shai Shalev-Shwartz, Yoram Singer, and Tushar Chandra. Efficient projections onto the l 1-ball for learning in high dimensions. In *Proceedings of the 25th international conference on Machine learning*, pages 272–279. ACM, 2008.

[5] Cheng Li, David Rubín de Celis Leal, Santu Rana, Sunil Gupta, Alessandra Sutti, Stewart Greenhill, Teo Slezak, Murray Height, and Svetha Venkatesh. Rapid bayesian optimisation for synthesis of short polymer fiber materials. *Scientific reports*, 7(1):5683, 2017.

[6] Chiho Kim, Anand Chandrasekaran, Tran Doan Huan, Deya Das, and Rampi Ramprasad. Polymer genome: A data-powered polymer informatics platform for property predictions. *The Journal of Physical Chemistry C*, 2018.

[7] Rachid Ounit, Steve Wanamaker, Timothy J Close, and Stefano Lonardi. Clark: fast and accurate classification of metagenomic and genomic sequences using discriminative k-mers. *BMC genomics*, 16(1):236, 2015.

[8] Peter Menzel, Kim Lee Ng, and Anders Krogh. Fast and sensitive taxonomic classification for metagenomics with kaiju. *Nature communications*, 7:11257, 2016.

[9] Patrenahalli M. Narendra and Keinosuke Fukunaga. A branch and bound algorithm for feature subset selection. *IEEE Transactions on computers*, C-26(9):917–922, 1977.

[10] Justin Doak. An evaluation of feature selection methods and their application to computer security. *Technical Report CSE-92-18 UC Davis*, 1992.

[11] Songyot Nakariyakul and David P Casasent. An improvement on floating search algorithms for feature subset selection. *Pattern Recognition*, 42(9):1932–1940, 2009.

[12] Stuart J Russell and Peter Norvig. *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,, 2016.

[13] Bing Xue, Mengjie Zhang, Will N Browne, and Xin Yao. A survey on evolutionary computation approaches to feature selection. *IEEE Transactions on Evolutionary Computation*, 20(4):606–626, 2016.

[14] Pedram Ghamisi and Jon Atli Benediktsson. Feature selection based on hybridization of genetic algorithm and particle swarm optimization. *IEEE Geoscience and remote sensing letters*, 12(2):309–313, 2015.

[15] David J Stracuzzi and Paul E Utgoff. Randomized variable elimination. *Journal of Machine Learning Research*, 5(Oct):1331–1362, 2004.

[16] Subrata Saha, Sanguthevar Rajasekaran, and Rampi Ramprasad. Novel randomized feature selection algorithms. *International Journal of Foundations of Computer Science*, 26(03):321–341, 2015.

[17] Aida Brankovic, Alessandro Falsone, Maria Prandini, and Luigi Piroddi. A feature selection and classification algorithm based on randomized extraction of model populations. *IEEE transactions on cybernetics*, 48(4):1151–1162, 2018.

[18] Pavel Pudil, Jana Novovičová, and Josef Kittler. Floating search methods in feature selection. *Pattern recognition letters*, 15(11):1119–1125, 1994.

[19] David J Stracuzzi. Randomized feature selection. In *Computational Methods of Feature Selection*, pages 53–74. Chapman and Hall/CRC, 2007.

[20] Sanguthevar Rajasekaran and John H Reif. Derivation of randomized sorting and selection algorithms. In *Parallel Algorithm Derivation and Program Transformation*, pages 187–205. Springer, 1993.

[21] Sanguthevar Rajasekaran. On simulated annealing and nested annealing. *Journal of Global Optimization*, 16(1):43–56, 2000.

[22] Joseph JáJá. *An introduction to parallel algorithms*, volume 17. Addison-Wesley Reading, 1992.

[23] Ellis Horowitz, Sartaj Sahni, and Sanguthevar Rajasekaran. *Computer algorithms C++: C++ and pseudocode versions*. Macmillan, 1997.

[24] Isabelle Guyon, Steve Gunn, Asa Ben Hur, and Gideon Dror. Design and analysis of the nips2003 challenge. In *Feature Extraction*, pages 237–263. Springer, 2006.

[25] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.