Wrapper Feature Selection with Partially Labeled Data

Vasilii Feofanov Emilie Devijver
Massih-Reza Amini
Univ. Grenoble Alpes, CNRS, Grenoble INP, LIG, 38000 Grenoble, France
firstname.lastname@univ-grenoble-alpes.fr

Published in Applied Intelligence *

Abstract

In this paper, we propose a new feature selection approach with partially labeled training examples in the multi-class classification setting. It is based on a new modification of the genetic algorithm that creates and evaluates candidate feature subsets during an evolutionary process, taking into account feature weights and recursively eliminating irrelevant features. To increase the variety of data, unlabeled observations are employed in the feature selection process, namely by pseudo-labeling them using a self-learning algorithm with a recently proposed transductive policy. Empirical results on different data sets show the effectiveness of our method compared to several state-of-the-art semi-supervised feature selection approaches.

1 Introduction

We consider semi-supervised classification problems where observations are described by a large number of characteristics. In this case, the original set of features may contain irrelevant or redundant characteristics to the output, which with the lack of labeled information leads to inefficient learning models. In practice, the removal of such features has been shown to provide important keys for the interpretability of results and yield better prediction performance (Guyon and Elisseeff, 2003; Chandrashekar and Sahin, 2014).

Feature selection techniques have been widely developed and, depending on the availability of class labels, can be supervised, unsupervised or semi-supervised. Being agnostic to the target variable, unsupervised approaches generally ignore the discriminative power of features, so their use may lead

*https://link.springer.com/article/10.1007/s10489-021-03076-w

to poor performance. In contrast, supervised feature selection algorithms benefit from abundant labeled examples, so they effectively select the subset of relevant characteristics. When there is no access to a large number of observations, performance of supervised approaches degrades, so selection of relevant features becomes an intricate issue. In semi-supervised learning (Sheikhpour et al., 2017), in addition to the scarce labeled set, a large collection of unlabeled data is assumed available, so the aim is to perform feature selection by exploiting both available labeled and unlabeled examples in order to provide a solution that preserves important structures of data, reducing significantly the original dimension and leading to high performance in accuracy. To take benefit from the large number of available unlabeled observations and compensate the scarcity of labeled examples in the learning process, a common approach is to increase the diversity of labeled training data by assigning pseudo-labels to some unlabeled observations using either self-learning or co-training approaches (Blum and Mitchell, 1998; Tür et al., 2005; Amini et al., 2008). However, pseudo-labels are prone to error, and their use may degrade performance as compared to a fully supervised model trained on the initial labeled examples only.

As searching of the optimal feature subset by exhaustive search would be computationally infeasible, many classical methods are based on sequential search algorithms, such as forward or backward selection. However, such methods are also computationally heavy for large-scale applications. Heuristic search algorithms, like the genetic algorithm (Goldberg and Holland, 1988), significantly reduce the computational time (Siedlecki and Sklansky, 1993; Xue et al., 2015). The genetic algorithm is an evolutionary optimization algorithm inspired by the natural selection process, where a *fitness function* is optimized by evolving iteratively a *population* of *candidates* (possible feature subsets). Starting from a randomly drawn population, at every *generation* a new population is produced by preserving *parents* from the last generation and creating *offspring* from the parents using operation of *crossover* and *mutation*. After a predefined number of generations the algorithm is stopped, and a candidate with the best fitness score in the last population is returned. The approach can be very effective when the number of features is very large. However, it has two main drawbacks: it may have a high variability on large-dimensional data sets, and the solution that the algorithm outputs is generally not as sparse as it could be, because any information like feature importance is ignoring during the crossover.

In this paper, we propose a new semi-supervised feature selection method, denoted by `TSLA-FSGA`, that first pseudo-labels unlabeled data using the self-learning algorithm with the transductive criterion derived by Feofanov et al. (2019). This allows to minimize the number of label errors produced by self-learning and to use a diverse augmented data for the feature subset search, which is performed by a new modification of the genetic algorithm, called *Feature Selection Genetic Algorithm* (`FSGA`) that reduces the variance in selection and ensures a high level of sparsity. This is achieved by considering feature weights during the optimization phase and iteratively removing features found to be

irrelevant. We guide FSGA by the out-of-bag score of the Random Forest classifier (Breiman, 2001), and we empirically show that the proposed approach is fast, accurate on large-scale benchmarks, and it outperforms the supervised baseline.

The remainder of this paper is organized as follows. Section 2 describes related work. Section 3 introduces the semi-supervised feature selection approach we propose, which is based on a modification of the genetic algorithm. Section 4 presents the experiments conducted on several data sets. Lastly, Section 5 concludes the paper and discusses the future work.

2 Related Work

Feature selection methods can be classified into three main families. Filter methods score features following a criterion and perform selection before the construction of a learning model (Yang et al., 2010; Zhao et al., 2008). Embedded techniques perform model-based feature selection in order to infer the importance of features during the training process (Chen et al., 2017). Finally, wrapper approaches use a learner to effectively find a subset of features that are discriminatively powerful together (Kohavi and John, 1997; Ren et al., 2008). In the specific case of semi-supervised learning, some works have been initiated recently according to these three directions.

Most of the semi-supervised feature selection algorithms are extensions of popular supervised or unsupervised filters. The Semi-Fisher Score (SFS, Yang et al. (2010)) extends the supervised Fisher score by embedding the graph Laplacian computed on labeled and unlabeled data. The Semi-supervised Laplacian Score (SSLS, Zhao et al. (2008)) is a graph-based approach that uses unlabeled examples to identify which features are able to preserve the local structure of the data, and labeled examples to maximize distance between observations from different classes. In the binary classification case, Sechidis and Brown (2018) proposed to use a supervised filter for partially-labeled data by using a surrogate target variable that is either fully positive or negative for unlabeled data. The main disadvantage of the filter approaches is that feature importance is evaluated individually, so there is a risk to discard features that are strong only in combination with others (Guyon and Elisseeff, 2003).

Recently, embedded approaches have become actively studied. They output feature weights during the training process, but, compared with wrappers, are inflexible to the objective. The Rescaled Linear Square Regression (RLSR, Chen et al. (2017)) is one of the approaches that ranks features by learning a projection matrix using the least square regression with a sparse regularization (Chen et al., 2017; Wu et al., 2021) and scaling the regression coefficients with a set of scale factors. Jiang et al. (2019) use the Bayesian approach to learn weights both for features and unlabeled examples that could be potentially irrelevant. In supervised feature selection, a popular embedded approach is the recursive feature elimination (RFE, Guyon et al. (2002)) that recursively relearns a learning algorithm removing a portion of features with the lowest fea-

ture weights at each iteration. Although this approach effectively removes irrelevant features, it may also remove some informative variables that are weak individually (Darst et al., 2018). In Section 4, we illustrate the empirical performance of its extension to semi-supervised learning.

Ren et al. (2008) proposed a semi-supervised wrapper approach (`CoT-FSS`), which incorporates unlabeled data to the training set by means of co-training, and find the best feature subset using forward sequential search. The approach performs co-training inside the wrapper, which makes it computationally expensive. In Han et al. (2011), it was shown that the complexity may be reduced by pseudo-labeling the unlabeled examples just once and then performing the wrapper feature selection on the augmented the data set. As the structure of wrapper methods is more flexible, the choice of the criterion is not necessarily limited to the accuracy score, and other learning-based metrics can be used (Song et al., 2007). This is particularly attractive for semi-supervised learning where the criterion may be evaluated using both labeled and unlabeled data.

However, sequential search approaches (Ren et al., 2008; Han et al., 2011) are time consuming for high-dimensional data, so in recent years attention has returned to evolutionary algorithms (Xue et al., 2015). The use of the genetic algorithm for wrapper selection has shown success for different tasks such as brain imaging data (Szenkovits et al., 2018) or classification of electroencephalograph signals (Buza, 2020), and the approach has been recently applied for feature selection in the semi-supervised multi-target regression case (Syed et al., 2021). However, in the high-dimensional setting, the performance of the classical genetic algorithm may decrease due to the large search space. In this work, we focus on the semi-supervised multi-class classification framework and propose a new modification of the genetic algorithm that allows to reduce variance in output and increase feature selection quality when dimension is large.

3 Semi-supervised Wrapper Algorithm: `TSLA-FSGA`

We consider the multi-class classification framework with the output space $\mathcal{Y} = \{1, \dots, K\}$, $K \geq 2$, and the input space $\mathcal{X} \subset \mathbb{R}^d$, where d is the total number of features. We suppose given a set of labeled examples $Z_{\mathcal{L}} = \{\mathbf{x}_i, y_i\}_{1 \leq i \leq l} \in (\mathcal{X} \times \mathcal{Y})^l$ and a set of unlabeled examples $X_{\mathcal{U}} = \{\mathbf{x}_i\}_{l+1 \leq i \leq l+u}$, $X_{\mathcal{U}} \in \mathcal{X}^u$. Given a level of sparsity $d' \ll d$, the goal is to find a feature subset $S^* \subseteq \{1, \dots, d\}$, $|S^*| = d'$ based on $Z_{\mathcal{L}} \cup X_{\mathcal{U}}$ that leads to the highest classification performance among all possible feature subsets of size d'.

Below, we present our semi-supervised framework for wrapper feature selection using both labeled and unlabeled data. Based on random forest as the base learning algorithm,described in Section 3.1, the approach consists of two phases: first, we increase variety of the training data by pseudo-labeling the unlabeled examples using a self-learning algorithm described in Section 3.2; on this new data set, we perform the feature selection in a wrapper fashion by a proposed genetic algorithm named Feature Selection Genetic Algorithm,

described in Section 3.3.

3.1 Base Classifier

In the literature of wrapper feature selection approaches (Guyon and Elisseeff, 2003; Xue et al., 2015), most of the well known classification methods have been used as the base classifier, which evaluates the strength of a feature subset. In this paper, we have chosen a random forest (Breiman, 2001, denoted by `RF`) of decision trees due to its ability to output feature weights, versatility for different tasks and high accuracy when the labeled set is scarce (Biau and Scornet, 2016).

The main principle of learning decision trees (Breiman et al., 1984) consists in top-down recursive binary partitioning of the feature space into disjoint regions. At every node, a feature and a split value are chosen from maximization of the Gini impurity decrease criterion. After growing a tree t, a feature weight ω_j^t is derived for a feature j by computing the total impurity decrease the feature yields. The main advantage of the decision tree is its versatility, since it can be applied for both numerical and categorical features being a scale-invariant non-linear learning algorithm.

As a single decision tree is prone to overfit, we consider the random forest classifier denoted by H, a majority vote ensemble of decision trees, where each tree $h_t,\ t \in \{1, \ldots, T\}$ is trained on a bootstrap sample B_t (Efron, 1992) of size l drawn with replacement from the training set $\mathrm{Z}_{\mathcal{L}}$. `RF` has been proposed along with the out-of-bag error, which has been shown to be an unbiased estimator of the generalization error (Breiman, 2001). For every training example $\mathbf{x} \in \mathrm{Z}_{\mathcal{L}}$ and a class $c \in \mathcal{Y}$, the out-of-bag vote is evaluated as the proportion of trees that did not contain the example $\mathbf{x}$ in their respective bootstrap sample and that output for $\mathbf{x}$ the class c :

$$v_{\mathcal{L}}(\mathbf{x}, c) = \frac{1}{|\{t : \mathbf{x} \notin \mathrm{B}_t\}|} \sum_{t:\ \mathbf{x} \notin \mathrm{B}_t} \mathbb{I}(h_t(\mathbf{x}) = c).$$

Then, the out-of-bag error is defined as follows:

$$F_{\mathcal{L}} = \frac{1}{l} \sum_{i=1}^{l} \mathbb{I}(y_i \neq \operatorname*{argmax}_{c \in \mathcal{Y}} v_{\mathcal{L}}(\mathbf{x}_i, c)). \tag{1}$$

In addition, the random forest outputs feature weights by averaging them over the trees: for a feature j,

$$w_j = \frac{1}{T} \sum_{t=1}^{T} \omega_j^t.$$

3.2 Pseudo-labeling via Self-learning Algorithm

In semi-supervised learning, the self-learning algorithm (`SLA`) is one of the most common approaches to benefit from unlabeled data. The main idea is

to iteratively re-learn a supervised base classifier (e.g., the random forest) by expanding the labeled set. At each iteration, it assigns pseudo-labels (i.e., predicted labels) to those unlabeled examples that have prediction confidence above a threshold. The pseudo-labeled examples are then included in the training set, and the classifier is retrained. The process is repeated until no examples for pseudo-labeling are left. At the end, a new augmented data set with an increased diversity of training examples is hence obtained. The algorithm is illustrated in Figure 1.

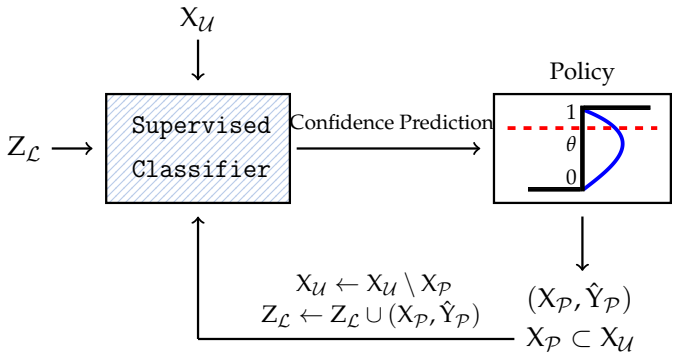


Figure 1: Self-Learning Algorithm `SLA`.

A common approach is to set the pseudo-labeling threshold to a fixed value (Tür et al., 2005; Han et al., 2011), which practically may degrade the classification performance comparing to the base classifier learned on the labeled examples only. To overcome this issue, Feofanov et al. (2019) have proposed to find this threshold dynamically in the context of multi-class majority vote classifiers (including random forests) based on an upper-bound of the majority vote's error in the transductive case (Vapnik, 1998). The class votes for the unlabeled examples, defined for the `RF` classifier H as

$$v(\mathbf{x}, c) = \frac{1}{T} \sum_{t=1}^{T} \mathbb{I}(h_t(\mathbf{x}) = c), \ \forall x \in X_{\mathcal{U}}, c \in \mathcal{Y},$$

are considered as indicators of prediction confidence. Thus, the self-learning algorithm with transductive policy, further called `TSLA`, considers a majority vote as the base classifier and dynamically learns at each iteration a threshold vector $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_K)$, one threshold per class, by minimizing the following criterion:

$$\min_{\boldsymbol{\theta} \in (0,1]^K} \frac{\sum_{\mathbf{x} \in X_{\mathcal{U}}} \mathbb{I}(y \neq H(\mathbf{x})) \mathbb{I}(v(\mathbf{x}, H(\mathbf{x})) \geq \theta_{H(\mathbf{x})})}{\sum_{\mathbf{x} \in X_{\mathcal{U}}} \mathbb{I}(v(\mathbf{x}, H(\mathbf{x})) \geq \theta_{H(\mathbf{x})})}. \tag{2}$$

Following this criterion, we choose a threshold as a trade-off between the proportion of examples going to be pseudo-labeled (numerator) and the error they

induce (denominator). The numerator is upper bounded using the transductive bound proposed by Feofanov et al. (2019).

Among the existing approaches for pseudo-labeling of unlabeled data, we have chosen TSLA due to the following reasons. Firstly, the approach does not require fixing any hyperparameter, since the confidence threshold is found automatically. Secondly, Feofanov et al. (2019) has experimentally shown that the performance of TSLA does not practically degrade compared to the supervised baseline thereby providing a safe solution. In Section 4.2, we experimentally show that the quality of pseudo-labels can indeed influence the feature selection quality by comparing TSLA with another version of self-learning.

In contrast to Ren et al. (2008) and Syed et al. (2021), who used a semi-supervised base classifier to evaluate the strength of feature subsets, we assign pseudo-labels to unlabeled examples prior to the feature selection step and then perform a subset search on the expanded training set, which drastically reduce the algorithm's complexity.

3.3 FSGA: Feature Selection Genetic Algorithm

After obtaining an augmented data set via TSLA, we perform a heuristic search using a genetic algorithm (Goldberg and Holland, 1988). The classical genetic algorithm, CGA, ignores during the crossover any information like feature importance, since a child inherits features from its parents at random. Moreover, the larger is the number of features, the larger is the search space, so the algorithm becomes highly variable which affects the performance (Xue et al., 2015).

In this connection, we propose a new genetic algorithm for feature selection that tackles these two problems: 1) the algorithm takes into account the importance of features during the generation of a new population by using a weighted crossover, 2) it iteratively removes variables that are found to be irrelevant, which accelerates the convergence and reduces the search space. The main steps of this algorithm are summarized in Fig. 2 and are described as follows.

Initialization: the population $\mathcal{P}_0$ is initialized by randomly generating feature subsets of a fixed length d'. Each candidate $S \in \mathcal{P}_0$ corresponds to a feature subset.

Evaluation of Fitness and Feature Weights: for the generation $g \geq 0$ and for each candidate $S \in \mathcal{P}_g$, a random forest model is learned on the labeled and the pseudo-labeled examples. Feature weights $\{w_j^S\}_{j \in S}$ are derived from the Random Forest classifier restricted to the feature subset S. To evaluate the strength of the subset S, the out-of-bag score OOB is considered as a fitness score. It evaluates the generalization error without subsampling (compared to the cross-validation for example), reducing the computational cost. The introduced fitness criterion is computed on labeled and pseudo-labeled data with

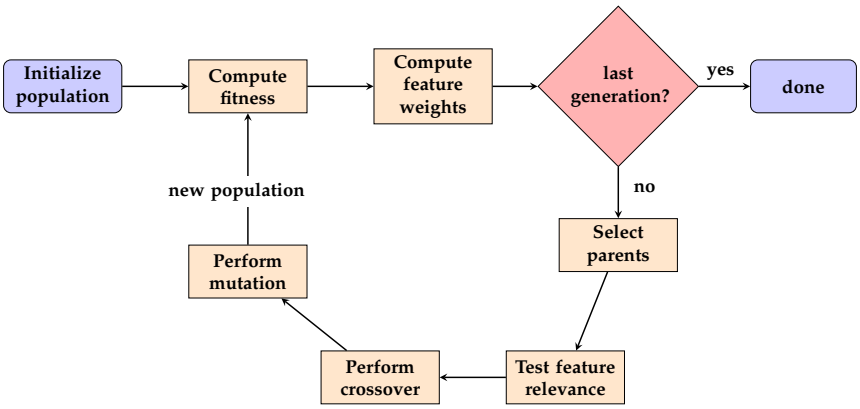


Figure 2: Feature Selection Genetic Algorithm (FSGA) in a nutshell.

projection on a feature subset S as follows:

$$F_{\mathcal{L}\cup\hat{\mathcal{L}}}(S) := \frac{1}{l+\tilde{l}} \sum_{i=1}^{l+\tilde{l}} \mathbb{I}(\hat{y}_i \neq \operatorname*{argmax}_{c\in\mathcal{Y}} v_{\mathcal{L}}(\mathbf{x}_i^{[S]}, c)), \tag{3}$$

where $\tilde{l} \leq u$ corresponds to the number of examples that have been pseudo-labeled, $\hat{y}_i = y_i$ for $i = \{1,\ldots,l\}$, $\hat{y}_i$ corresponds to the pseudo-labels for $i = \{l+1,\ldots,l+\tilde{l}\}$, and $\mathbf{x}_i^{[S]}$ denotes projection of $\mathbf{x}_i$ on the set of features S. Note that this fitness criterion considers pseudo-labels that might be erroneous, but Dietterich (2000) has empirically shown that the underlying random forest bagging procedure is robust in the presence of noise in the output, which additionally validates our choice of the fitness criterion.

Parent Selection: among the population $\mathcal{P}_g$, p candidates with the best fitness scores are selected, used for the next population $\mathcal{P}_{g+1}$ and for producing a new offspring. There exists various policies in the literature of genetic algorithms to select parents (Goldberg and Deb, 1991) such as tournament selection or proportionate reproduction. However, we have experimentally observed no benefit from such additional randomness, so we stick to the simplest policy of choosing candidates with the best fitness score (which corresponds to the tournament of maximal size).

Relevance Test: a test is performed to eliminate irrelevant to response variables. We embed an approach of Tuv et al. (2009) to compare variables with their copies using randomly permuted values. At first, we consider the features that appear at least in one of the candidates: $S_g = \{j \in \{1,\ldots,d\} : \exists S \in \mathcal{P}_g \text{ s.t. } j \in S\}$. We compute their average weights as:

$$\bar{w}_j = \frac{\sum_S \mathbb{I}[S\in\mathcal{P}_g]\wedge[j\in S]\, w_j^S}{\sum_{S\in\mathcal{P}_g} \mathbb{I}(j \in S)}, \quad j \in S_g.$$

We define the set of suspicious features S_{out} as a percentage[1] of features that have the smallest average weight. The principle of the test is to learn a classifier on a new data set, composed by: the features detected by the best parent, suspicious irrelevant features and their randomly permuted copies. We construct R times the data set with permuted copies and learn the classifier. For each $r \in \{1, \ldots, R\}$, we look at the feature weights of the noisy counterparts and evaluate a high percentile from their distribution, denoted by τ_r, which serves as a threshold to distinguish informative variables from noisy ones. Thus, for each feature $j \in S_{\text{out}}$ we have a sample of R feature weights retrieved from the R classifiers. This sample is compared with the sample of thresholds $(\tau_r)_{r=1}^{R}$ using the one-sided Mann-Whitney U test with a suitably small p-value. The hypothesis rejection for a feature j implies that its feature weight is statistically close to feature weights of the noisy counterparts, so this feature is called irrelevant, removed and will not be further considered by the algorithm. Since the size of selected parents may be less than the initial d' after this test, we perform backfilling by randomly including (all but irrelevant) features into the subset to reach the size d'. We set the weights of these features to be 10^{-10} so they have little chance to participate in the next crossover.
The test requires to set such parameters as the percentile and the p-value. The higher percentile is taken, the higher thresholds $(\tau_r)_{r=1}^{R}$ are set, while high p-values suggest a more drastic assessment of irrelevance. We have noticed that the relevance test becomes more qualitative with the increase of classes[2] as in this case the difference between informative and irrelevant variables become more evident, so the test is not very sensitive to the choice of the percentile and the p-value, and the number of experiments R can be even reduced. Originally, this test has been proposed for supervised feature selection, and it was performed just once on the whole feature set (Tuv et al., 2009). In semi-supervised learning, the number of labeled examples is often much smaller than the number of features, so the features weights may be biased leading to not correct relevance estimation. Although incorporation of the pseudo-labeled unlabeled examples may help to reduce the bias, we have observed that using the relevance test iteratively at each generation improves the performance results (see more details in Appendix, Section A).

Crossover and Mutation: A new child is generated by mating two parents. In contrast to `CGA`, we inherit variables according to their weights: for each parent, its features are sorted by their weights in the decreasing order. The crossover point that characterizes the proportion of features inherited from the first parent is taken randomly, and we fill the child by its sorted features until we reach the quota. The rest of the features are taken from the second parent, ensuring no repetitions. This type of crossover suggests to increase exploitation of informative features with large feature weights.
To increase the diversity of candidates and prevent "deadlocks", mutation is used: for each child, a random number of features (not greater than a parame-

[1] In our experiments, we fix it to 30%.
[2] Assuming we are not dealing with a high class-imbalance.

ter $\text{mut}_{\max}$) from the subset S are replaced by the same number of features out of S. Since the proposed weighted crossover operator is highly exploitative, an aggressive mutation (i.e., large values of $\text{mut}_{\max}$) is a reasonable choice to balance exploration.

Those steps are repeated to generate new populations for several generations, and a candidate with the best fitness in the final population is output.

3.4 Time Complexity

In this section, we discuss the time complexity of our method and compare it with the other semi-supervised feature selection approaches discussed in Section 2. The conclusion of this section is empirically illustrated in Section 4.4.

Selecting $d' = \sqrt{d}$ variables, for each candidate feature subset, `FSGA` evaluates the fitness based on the random forest classifier that has the average time complexity $O(\sqrt{d}Tn\log^2 n)$, where $n \approx 0.632 \cdot (l + \tilde{l})$ corresponds to the number of labeled and pseudo-labeled examples that are used in a bootstrap sample, T is the number of trees (Louppe, 2014). Then, the complexity of `FSGA` is $O(\sqrt{d}TN_g(N_c + 2R)n\log^2 n)$, where N_g is the number of generations, N_c is the population size, R is the number of experiments in the relevance test. In our experimental setup, we have set fixed T, N_g, N_c, R, so the complexity can be written as $O(\sqrt{d}n\log^2 n)$, which indicates a good scalability of the algorithm. Note also that the trees of `RF` as well as the learning models for fitness evaluation are naturally parallelized, which can significantly speed up the algorithm.

Compared to `FSGA`, the wrapper algorithms like Ren et al. (2008); Han et al. (2011) are time consuming for high-dimensional data, because they are based on sequential feature subset searching, which yields a complexity cubic in the dimension and quasilinear in the sample size [3]. The complexity of semi-supervised filter approaches like `SFS` (Yang et al., 2010) and `SSLS` (Zhao et al., 2008) are linear with respect to the dimension, but quadratic with respect to the sample size because of the Laplacian matrix's evaluation. In a large-scale setting, the complexity of `RLSR` (Chen et al., 2017) is high, being cubic in the dimension (or linear in the dimension and quadratic in the sample size when the sample size is large).

4 Experimental Results

The benefit of our approach is illustrated on a simulated data set as well as 10 publicly available data sets (Chang and Lin, 2011; Guyon, 2003; Xiao et al., 2017; LeCun et al., 1998; Li et al., 2018; Dua and Graff, 2017).

The synthetic data is generated based on the algorithm[4] that was used to create the `Madelon` data set (Guyon, 2003). The size of training labeled and

[3] In the case of using a decision-tree based classifier inside the wrapper.

[4] We use the implementation of Pedregosa et al. (2011).

training unlabeled sets are set respectively to 100 and 900. We fixed the number of classes to 3; the number of features to 20 wherein 8 features (No. 1-8) are informative, 6 redundant features (No. 9-14) are exact copies of the first informative variable, and 6 features (No. 15-20) are irrelevant to the target. We have observed that the first informative variable is individually strong, whereas the second one is weak, so the redundant features may be more attractive for selection rather than the second variable.

The characteristics of 10 benchmark data sets are summarized in Table 1. The associated applications are text classification with `PCMAC`, `Relathe` and `Basehock` databases; image classification with `Fashion`, `MNIST`, `Coil20` and `Gisette` data sets; bioinformatics with `Protein` data set; feature selection with `Madelon`; and speech recognition with `Isolet`. `MNIST` and `Fashion` data sets have been restricted to a subset of 10000 observations. To imitate the semi-supervised setting, we do not use the train / test splits that are proposed by data sources, but we use our own splits such that $l \ll u$.

Table 1: Characteristics of data sets used in our experiments ordered by dimension d.

Data set	# of lab. examples, l	# of unlab. examples, u	Dimension, d	# of classes, K
`Protein`	108	972	77	8
`Madelon`	260	2340	500	2
`Isolet`	156	1404	617	26
`Fashion`	100	9900	784	10
`MNIST`	100	9900	784	10
`Coil20`	144	1296	1024	20
`PCMAC`	195	1748	3289	2
`Relathe`	143	1284	4322	2
`Basehock`	200	1793	4862	2
`Gisette`	70	6930	5000	2

We use the scikit-learn implementation of the random forest with 100 trees of maximal depth (Pedregosa et al., 2011). The latter is used as the majority vote classifier for `TSLA`, whose implementation is provided by Feofanov et al. (2019). The code we have developed is available at `https://github.com/vfeofanov/TSLA-FSGA`.

To perform comparison with state-of-the-art approaches, for all feature selection algorithms, we first find a feature subset using a feature selection method, where the number of selected features d' is fixed to $\lfloor \sqrt{d} \rfloor$. Then, we train `TSLA` on the selected features and compute its performance, the classification accuracy on the unlabeled set (`ACC-U`).

For all experimental results, we perform 20 random labeled / unlabeled splits of the initial collection and report the average classification accuracy over the 20 trials on the unlabeled training set. We set a time limit to 1 hour per split and terminate an algorithm if the limit is exceeded. These cases are indicated

as NA. All experiments were performed on a cluster with an Intel(R) Xeon(R) CPU E5-2640 v3 at 2.60GHz, 32 cores, 256GB of RAM, the Debian 4.9.110-3 x86_64 OS.

The experiments are organized as follows. First, we validate the Feature Selection Genetic Algorithm (FSGA) through an ablation study showing benefit of each step of the algorithm for finding an optimal feature subset. Then, we study how the choice of a learning model and the use of pseudo-labels have an impact on the selection criterion's strength. Finally, a comparison of the full method TSLA-FSGA with state-of-the-art methods is performed.

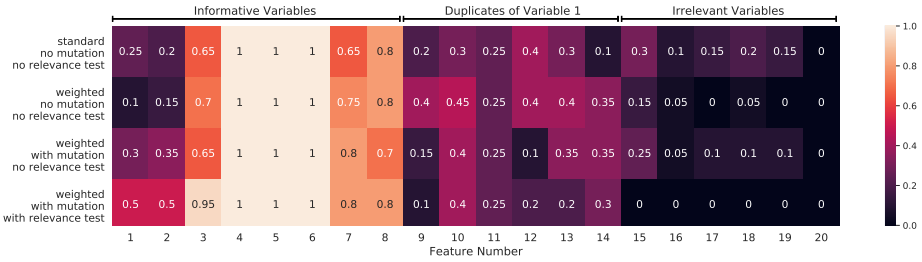
4.1 Validation of the Feature Selection Genetic Algorithm

In this section, we want to demonstrate the benefit of the proposed FSGA by showing that the weighted crossover and the relevance test provides clear improvement over the classical genetic algorithm (CGA). All versions of the genetic algorithm considered in this section use TSLA for pseudo-labeling and the pseudo-supervised out-of-bag score given in Eq. (3) as the fitness criterion.

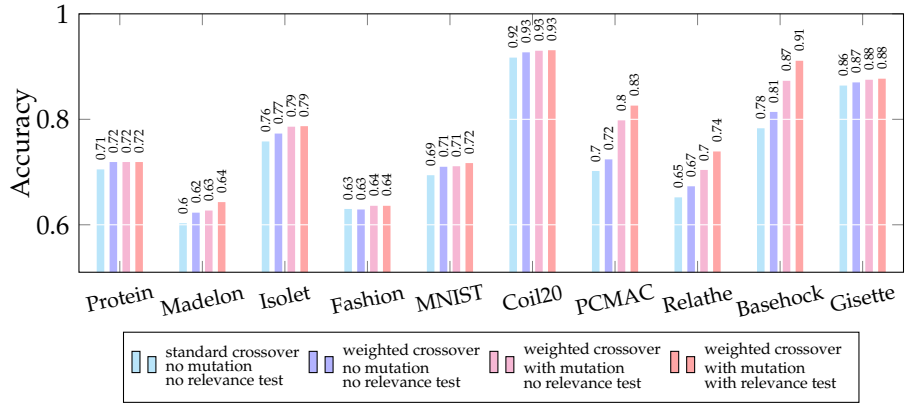
In the implementation of the genetic algorithms, the number of generations is fixed to 25, the population size to 40 and the number of parents to 8. The maximum number of mutations is set to $\lfloor \sqrt{d}/2 \rfloor$. In the relevance test, we consider the 30% worst variables as suspicious, learn 10 classifiers with randomly permuted values, use the 95-th percentile to find the threshold, and set the p-value for the hypothesis test to 0.05. To highlight the benefit of the weighted crossover, we first run the genetic algorithm without mutation. Then, we refine the algorithm by successively adding mutation and relevance test, which finally leads to FSGA.

First, the algorithms are compared on the synthetic data set, and the results are illustrated in Fig. 3a. By looking at the irrelevant variables (15-20), we can observe that the weighted crossover is less prone to select these variables compared to the standard one. At the same time, when the weighted crossover is used alone, the subset search has little exploration concentrating on the individually strong features, as illustrated on variables 2 (weak feature) or 9 to 14 (strong features). This is solved by activating the mutation step, which helps to generate more diverse subsets. However, with mutation, the output variance increases, so the irrelevant variables are again become selected more often. Then, this variance is reduced by activating the relevance test, which removes from consideration features found to be irrelevant to the target. Thus, we also reduce the search space, so the selection quality is generally improved (variables 2, 3 and 8 are selected more often, whereas variables 9-14 less often).

Then, the algorithms are compared on the benchmark data sets, and the experimental results are depicted in Fig. 3b. For most of data sets, more refinements yield better performance, which particularly leads to a high difference between the first and the last columns on PCMAC, Relathe and Basehock with an improvement of around 10%. According to the Mann and Whitney U test on level 0.01, the weighted crossover significantly outperforms the standard one



(a) Results on the synthetic data set. The features are sorted in the following order: 8 informative features, 6 redundant features, 6 irrelevant ones. On the graph, each cell represents the number of times when a feature was chosen by a feature selection method divided by the number of experiments (20).



(b) Comparison on the benchmark data sets described in Table 1. The accuracy on the unlabeled set (`ACC-U`) of a classifier trained on the final feature subset is illustrated.

Figure 3: Comparison of 4 different versions of the genetic algorithm: the standard crossover comparing with the weighted crossover, adding to the latter successively mutation operator and then the relevance test. Note that the proposed approach, `FSGA`, corresponds to the last column in red.

on `Madelon`, `MNIST` and `Basehock` data sets. Adding the mutation step leads to a significant improvement on `PCMAC` and Basehock. Finally, the relevance test is a useful procedure to reduce the search space, and a significant improvement is observed on 3 data sets (`PCMAC, Relathe, Basehock`).

4.2 Improvement from Pseudo-labeling Unlabeled Data

In this section, we set the search scheme to `FSGA` and investigate how the choice of the learning algorithm and the fitness criterion impact the feature selection quality and what contribution unlabeled data can make.

At first, we evaluate two supervised baselines: when a single decision tree

is used as a learning algorithm with the 5-fold cross-validation score as the fitness criterion (`Sup-Tree`), and when the random forest is learned with the out-of-bag score as the criterion (`Sup-RF`). In both cases, only the labeled examples are used for learning and fitness evaluation, and by comparing these two criteria we study whether the choice of a more sophisticated learning approach improves the results.

Then, we analyze the utility of unlabeled data for feature selection by considering three semi-supervised approaches. At the beginning, we use a self-learning algorithm to pseudo-label the unlabeled examples. Then, the labeled and the pseudo-labeled examples are used for training the genetic algorithm, where the random forest is set as a learning algorithm. To study how the quality of pseudo-labels influence the performance of the feature subset search, we compare two self-learning policies: 1) when at each step randomly picked 10% of the unlabeled examples are added with their corresponding predictions (`RSLA`), 2) when we add at each step unlabeled examples with prediction vote higher than a threshold empirically learned from minimization of the transductive criterion (`TSLA`, Section 3.2). Note that the former policy is similar to a mechanism used by Ren et al. (2008) for co-training. In the both cases, the strength of a feature subset S is evaluated by the pseudo-supervised out-of-bag score $F_{\mathcal{L}\cup\hat{\mathcal{L}}}(S)$ defined by Eq. (3). Finally, to see the impact of pseudo-labels on the fitness criterion alone, we introduce a third approach, where the pseudo-labels are acquired by `TSLA` and used inside the genetic algorithm, but the feature strength is evaluated on validation sets consisting only of the purely labeled examples. In other words, for each subset S, the learning algorithm is trained on both the labeled and the pseudo-labeled examples, and the out-of-bag score on the labeled examples is used as the fitness criterion:

$$F_{\mathcal{L}}(S) := \frac{1}{l}\sum_{i=1}^{l} \mathbb{I}(y_i \neq \operatorname*{argmax}_{c\in\mathcal{Y}} v_{\mathcal{L}}(\mathbf{x}_i^{[S]}, c)), \tag{4}$$

where $\mathbf{x}_i^{[S]}$ denotes projection of $\mathbf{x}_i$ on the set of features S.

The performance results are summarized in Table 2. At first, we can see that the random forest provides always more qualitative selection compared to the single tree (higher accuracy and smaller variance). Hence, the use of ensemble methods improves the selection quality, which would be connected with their robustness to overfitting.

Then, we observe better quality selection when the pseudo-labeled examples are used in the algorithm, so all the three semi-supervised approaches generally outperform its supervised baseline `Sup-RF`. `TSLA` with $F_{\mathcal{L}\cup\hat{\mathcal{L}}}$ as the criterion benefits the most from unlabeled data, and it significantly outperforms `Sup-RF` on 4 data sets, `RSLA` on `Relathe` and `Basehock`, `TSLA` with $F_{\mathcal{L}}$ on `Isolet` and `MNIST`.

By comparing `RSLA` and `TSLA`, we can see that the more careful pseudo-labeling based on the transductive guarantees leads to the highest performance, and the larger portions of noisy pseudo-labels generally lead to worse results.

Table 2: The classification performance (accuracy) of different approaches to evaluate the fitness score in FSGA: two supervised baselines Sup-Tree and Sup-RF, three semi-supervised approaches, where TSLA and RSLA denote which self-learning policy is used for pseudo-labeling, while $F_{\mathcal{L}\cup\hat{\mathcal{L}}}$ and $F_{\mathcal{L}}$ denote which fitness criterion is taken. In addition, a % of wrong pseudo-labeled unlabeled examples (%N) is provided for the approaches that use $F_{\mathcal{L}\cup\hat{\mathcal{L}}}$ as the criterion. $^{\downarrow}$ indicates statistically significantly worse performance than the best result (shown in bold), according to the Mann-Whitney U test ($p < 0.01$).

Data set	Sup-Tree	Sup-RF	RSLA: $F_{\mathcal{L}\cup\hat{\mathcal{L}}}$		TSLA: $F_{\mathcal{L}\cup\hat{\mathcal{L}}}$		TSLA: $F_{\mathcal{L}}$
	ACC-U	ACC-U	ACC-U	%N	ACC-U	%N	ACC-U
Protein	$.707 \pm .043$	$.719 \pm .037$	**.725** $\pm .045$	25.5%	$.719 \pm .04$	21.4%	$.702 \pm .04$
Madelon	$.606^{\downarrow} \pm .04$	$.617^{\downarrow} \pm .023$	$.632 \pm .02$	39.6%	$.643 \pm .031$	42%	**.651** $\pm .034$
Isolet	$.744^{\downarrow} \pm .026$	$.751^{\downarrow} \pm .027$	$.77 \pm .025$	18.3%	**.787** $\pm .02$	14.1%	$.766^{\downarrow} \pm .022$
Fashion	$.604^{\downarrow} \pm .027$	$.627 \pm .021$	$.631 \pm .021$	32.5%	**.636** $\pm .023$	30.6%	$.631 \pm .014$
MNIST	$.639^{\downarrow} \pm .029$	$.676^{\downarrow} \pm .024$	$.714 \pm .02$	19.5%	**.717** $\pm .021$	17.4%	$.694^{\downarrow} \pm .021$
Coil20	$.903^{\downarrow} \pm .022$	$.922 \pm .018$	$.922 \pm .017$	6.8%	**.931** $\pm .016$	5.7%	$.917 \pm .022$
PCMAC	$.783^{\downarrow} \pm .022$	$.822 \pm .014$	$.824 \pm .021$	17.4%	**.826** $\pm .019$	15.9%	**.826** $\pm .018$
Relathe	$.684^{\downarrow} \pm .033$	$.735 \pm .024$	$.708^{\downarrow} \pm .042$	28.6%	**.739** $\pm .027$	23.1%	$.738 \pm .028$
Basehock	$.84^{\downarrow} \pm .032$	$.902^{\downarrow} \pm .008$	$.9^{\downarrow} \pm .009$	8.5%	**.911** $\pm .01$	7.8%	$.902 \pm .018$
Gisette	$.849^{\downarrow} \pm .024$	**.88** $\pm .01$	**.88** $\pm .012$	11.7%	$.877 \pm .01$	12%	$.874 \pm .015$

However, both for RSLA and TSLA, the performance is not degraded with respect to the baseline Sup-RF on most of data sets, which validates our choice of the out-of-bag score inside the criterion. Thus, we conclude that the selection becomes more qualitative when unlabeled data are explored using the self-learning algorithm.

When we compare the two criteria based on TSLA, $F_{\mathcal{L}\cup\hat{\mathcal{L}}}$ and $F_{\mathcal{L}}$, we deduce that the use of pseudo-labeled data for evaluation of feature strength is actually helpful. This may be connected with the fact that the few labeled examples bias the fitness score, and trusting pseudo-labels would give more benefit than harm. This is coherent with a general observation that the traditional supervised model selection based on validation is not effective in the semi-supervised setting (Madani et al., 2005). Note that TSLA with $F_{\mathcal{L}}$ nevertheless outperforms the baseline Sup-RF, since the pseudo-labels are still used to compute the feature weights.

4.3 Comparison with the State-of-the-Art

Finally, we validate the proposed approach referred as TSLA-FSGA, by comparing its performance with the state-of-the-art. It is compared with RLSR, SFS, SSLS and CoT-FSS (with decision tree as the learning algorithm inside the wrapper, the co-training for pseudo-labeling and the 5-fold cross-validation

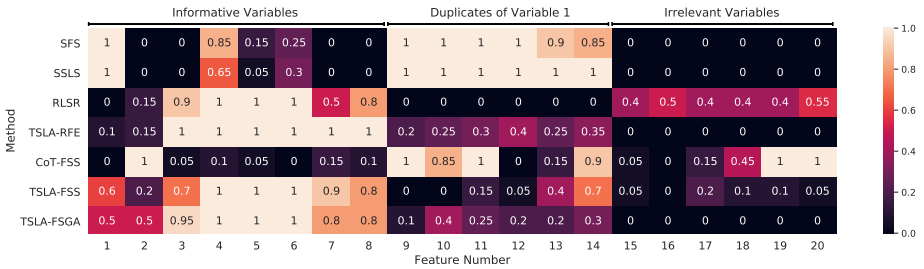
score as the selection criterion), all introduced in Section 2. In order to additionally validate FSGA, we also compare with two more selection algorithms, for which the unlabeled examples are pseudo-labeled before the feature selection step using TSLA for a fair comparison. Then, the first approach (denoted by TSLA-FSS), similarly to Han et al. (2011), performs the forward sequential search by minimizing the pseudo-supervised out-of-bag score of a random forest given by Eq. (3), whereas the second approach (denoted by TSLA-RFE) applies the recursive feature elimination based on the random forest.

Due to the lack of labeled training examples, the hyperparameters of all methods are set to their default values. Namely, for RLSR, the regularization parameter γ is set to 0.1; for SSLS and SFS, the number of nearest neighbors is set to 20, and the bandwidth for constructing the graph Laplacian is determined using the median distance heuristic (Schölkopf, 1997). For TSLA-FSS, at each step we add 10% best features into the model, and for TSLA-RFE, at each step we remove 10% worst ones.

First, we compare the considered approaches on the synthetic data set. In Fig. 4a the feature selection results averaged over 20 trials are reported. One can observe that the filter approaches, SFS and SSLS, perfectly detect and discard the irrelevant features. However, since the importance of features is evaluated independently, only individually strong informative variables are selected. Thus, some informative features are rarely or never selected (2-3, 5-8), and redundant features (9-14), which bring no new information, are preferred. In contrast, RLSR is able to find redundancies but does not succeed to eliminate the irrelevant variables (15-20). From the results, CoT-FSS has the most difficulties to select relevant variables without a clear selection pattern. All TSLA-RFE, TSLA-FSS and TSLA-FSGA perform quite well. FSS is less effective in eliminating the irrelevant variables than RFE and FSGA that discard them as perfect as the filter methods. Although RFE is slightly better in selecting variables 7 and 8, FSGA outperforms it in selecting variable 2. As it was mentioned before, variable 2 is an individually weak variable, so RFE often discards it preferring to keep one of the duplicates of variable 1. In turn, FSGA evaluates features *jointly* focusing explicitly on how the selected features are combined.

Figure 4b summarizes the performance results on the 10 benchmark data sets. On 4 data sets, Isolet, Fashion, MNIST and Coil20, our approach significantly outperforms all the other methods, while it is never significantly worse in cases when TSLA-FSGA is not the best. Compared to our approach, performances of other wrapper-based methods (CoT-FSS and TSLA-FSS) are significantly worst in most of the cases, which indicates the superiority of a genetic algorithm over a sequential search as the search scheme. We can also see that the performance of RLSR fluctuates from one data set to another. This could be connected with its sensitivity to the value of its regularization parameter γ, which is difficult to tune with few labeled examples. The filter methods, SFS and SSLS, are significantly worst in all situations. One can conclude that the filters are more suitable as a pre-processing step rather than as a complete feature selection process.

Finally, the recursive feature elimination, TSLA-RFE, has the best perfor-



(a) Results on the synthetic data set. The features are sorted in the following order: 8 informative features, 6 redundant features, 6 irrelevant ones. On the graph, each cell represents the number of times when a feature was chosen by a feature selection method divided by the number of experiments (20).

(b) Performance results on benchmark data sets. The classification accuracy is computed on the unlabeled set. $^{\downarrow}$ indicates statistically significantly worse performance than the best result (shown in bold), according to the Mann-Whitney U test ($p < 0.01$).

Data set	Filters		Embedded		Wrappers		
	SFS	SSLS	RLSR	TSLA-RFE	CoT-FSS	TSLA-FSS	TSLA-FSGA
Protein	$.676^{\downarrow} \pm .035$	$.634^{\downarrow} \pm .037$	$.695 \pm .034$	$\mathbf{.719} \pm .039$	$.583^{\downarrow} \pm .09$	$\mathbf{.719} \pm .033$	$\mathbf{.719} \pm .04$
Madelon	$.589^{\downarrow} \pm .035$	$.526^{\downarrow} \pm .038$	$.516^{\downarrow} \pm .015$	$\mathbf{.658} \pm .027$	$.514^{\downarrow} \pm .032$	$.649 \pm .032$	$.643 \pm .031$
Isolet	$.56^{\downarrow} \pm .037$	$.549^{\downarrow} \pm .03$	$.638^{\downarrow} \pm .05$	$.755^{\downarrow} \pm .03$	$.438^{\downarrow} \pm .058$	$.656^{\downarrow} \pm .053$	$\mathbf{.787} \pm .02$
Fashion	$.348^{\downarrow} \pm .03$	$.35^{\downarrow} \pm .033$	$.419^{\downarrow} \pm .048$	$.615^{\downarrow} \pm .025$	$.462^{\downarrow} \pm .043$	$.444^{\downarrow} \pm .041$	$\mathbf{.636} \pm .023$
MNIST	$.112^{\downarrow} \pm .0$	$.176^{\downarrow} \pm .029$	$.129^{\downarrow} \pm .013$	$.671^{\downarrow} \pm .029$	$.394^{\downarrow} \pm .068$	$.514^{\downarrow} \pm .027$	$\mathbf{.717} \pm .021$
Coil20	$.748^{\downarrow} \pm .046$	$.743^{\downarrow} \pm .045$	$.858^{\downarrow} \pm .029$	$.902^{\downarrow} \pm .02$	$.817^{\downarrow} \pm .038$	$.817^{\downarrow} \pm .025$	$\mathbf{.931} \pm .016$
PCMAC	$.613^{\downarrow} \pm .056$	$.545^{\downarrow} \pm .027$	$.773^{\downarrow} \pm .033$	$\mathbf{.832} \pm .019$	NA	$.813 \pm .042$	$.826 \pm .019$
Relathe	$.613^{\downarrow} \pm .025$	$.584^{\downarrow} \pm .017$	$.719 \pm .034$	$\mathbf{.746} \pm .028$	NA	$.694 \pm .032$	$.739 \pm .027$
Basehock	$.733^{\downarrow} \pm .051$	$.582^{\downarrow} \pm .081$	$.875^{\downarrow} \pm .02$	$\mathbf{.913} \pm .008$	NA	NA	$.911 \pm .01$
Gisette	$.851^{\downarrow} \pm .019$	$.521^{\downarrow} \pm .01$	$.51^{\downarrow} \pm .01$	$.871 \pm .017$	NA	NA	$\mathbf{.877} \pm .01$

Figure 4: Comparison of our method with the state-of-the-art approaches to select relevant features in semi-supervised learning.

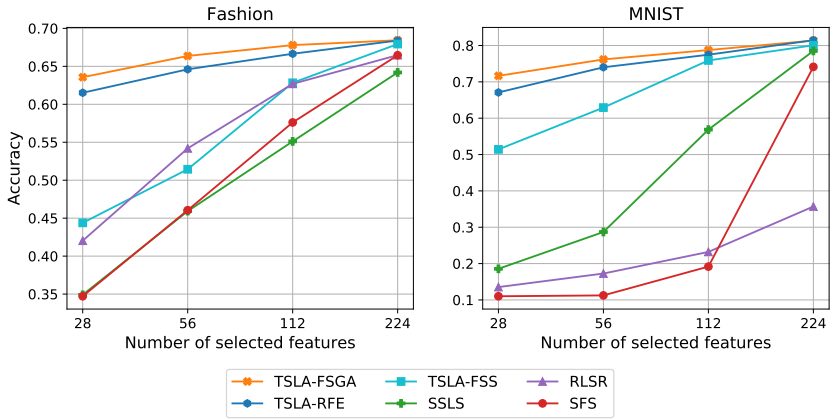


Figure 5: Performance of the proposed method and the state-of-the-art approaches on the `Fashion` and `MNIST` data sets when the number of selected features varies. `CoT-FSS` is omitted due to the large computational time.

mance on the data sets with many irrelevant features such as `Madelon`, `PCMAC`, `Relathe` and `Basehock`. However, when there are plenty of different informative features (`Isolet`, `Fashion`, `MNIST`, `Coil20`), `RFE` tends to underselect some weak but important for classification variables, so it becomes significantly worse than `TSLA-FSGA`. In addition, all these data sets are multi-class, which may arise additional difficulties for the feature selection task. In contrast, `TSLA-FSGA` successfully outputs feature subsets that are sparse and highly discriminative at the same time. Also, with the help of the relevance test, our approach detects and explicitly eliminates irrelevant variables, so the algorithm performs very well on `Madelon`, `PCMAC`, `Relathe` and `Basehock` as well.

In addition, we have looked at the performance of the algorithms depending on the number of selected features d'. In Figure 5, we demonstrate the performance results on the `Fashion` and `MNIST` data sets, taking $d' \in \{\sqrt{d}, 2\sqrt{d}, 4\sqrt{d}, 8\sqrt{d}\}$. One can observe that our approach `TSLA-FSGA` outperforms the other algorithms on different sparsity levels. However, we can see that the difference in accuracy between all the algorithms becomes less apparent as we relax the sparsity requirement. This is explained by the fact that the search space is significantly smaller for larger d', and the performance is less sensitive to incorrect choices of features. This observation has motivated us to fix the number of selected features as $d' = \lfloor \sqrt{d} \rfloor$ in all other experiments.

4.4 Run-time

We present in Table 3 the run-time of all the algorithms, to illustrate theoretical complexities introduced in Section 3.4.

A particular attention should be taken on `MNIST, Fashion, Gisette` with

Table 3: The average run-time of the feature selection algorithms under consideration on the benchmark data sets. *s* stands for seconds, *m* for minutes and *h* for hours.

Data set	SFS	SSLS	RLSR	TSLA-RFE	CoT-FSS	TSLA-FSS	TSLA-FSGA
Protein	1 s	1 s	10 s	9 s	22 s	26 s	2 m
Madelon	15 s	14 s	1 m	9 s	5 m	4 m	4 m
Isolet	5 s	1 s	2 m	13 s	9 m	6 m	4 m
Fashion	4 m	4 m	3 m	24 s	16 m	13 m	7 m
MNIST	5 m	4 m	3 m	24 s	18 m	14 m	7 m
Coil20	8 s	1 s	3 m	13 s	16 m	7 m	3 m
PCMAC	45 s	31 s	23 m	13 s	>1 h	26 m	3 m
Relathe	29 s	22 s	38 m	13 s	>1 h	29 m	4 m
Basehock	1 m	51 s	44 m	16 s	>1 h	>1 h	4 m
Gisette	17 m	17 m	21 m	40 s	>1 h	>1 h	6 m

respect to the large sample size as well as on Relathe, Basehock, Gisette with respect to the large dimension. Although the genetic algorithm FSGA is slower than RFE, it still passes the scale well both with respect to the sample size and the dimension. Being very fast on small data sets, the filter methods SFS and SSLS significantly slow down with the increase of sample size. In turn, when the dimension is large, RLSR becomes expensive too. On large data sets, CoT-FSS and TSLA-FSS are computationally infeasible. In general, the results clearly illustrate the complexity discussion standing in Section 2.

5 Conclusion and Future Work

In this paper, we proposed a new framework for semi-supervised wrapper feature selection. To increase the diversity of labeled data, unlabeled examples are pseudo-labeled using a self-learning algorithm. To produce a sparse solution, we proposed a modification of the genetic algorithm by taking into account feature weights during its evolutionary process and eliminating variables irrelevant to the target. The proposed model was empirically validated through an ablation study and a comparison with several feature selection approaches.

As a future work, it would be interesting to detect automatically the level of sparsity, set to $\lfloor \sqrt{d} \rfloor$ in our paper. In this case, simple criteria like the out-of-bag score trivially lead to the situation when a large number of selected features is chosen. One solution would be to add the regularization term to impose the sparsity level, for example, as proposed by (Frohlich et al., 2003; Da Silva et al., 2011).

Another direction would be to improve evaluation of the feature strength by developing a fitness criterion aware of possible errors in pseudo-labels. For instance, this can be achieved by introducing an additional assumption to determine data regions of low confidence, where the pseudo-labels are prone to error.

6 Acknowledgments

We thank 3 anonymous reviewers for their valuable comments. This work was partly funded by the IDEX project IRS (France).

7 Declarations

Funding This work was partly funded by the IDEX project IRS (France).
Conflicts of interest/Competing interests Not applicable.
Availability of data and material All the data sets are publicly available in Chang and Lin (2011); Guyon (2003); Xiao et al. (2017); LeCun et al. (1998); Li et al. (2018); Dua and Graff (2017).
Code availability The code we have developed is available at `https://github.com/vfeofanov/TSLA-FSGA`.

References

Amini M, Laviolette F, Usunier N (2008) A transductive bound for the voted classifier with an application to semi-supervised learning. In: Advances in Neural Information Processing Systems, pp 65–72

Biau G, Scornet E (2016) A random forest guided tour. Test 25(2):197–227

Blum A, Mitchell T (1998) Combining labeled and unlabeled data with co-training. In: Proceedings of the eleventh annual conference on Computational learning theory (COLT), pp 92–100

Breiman L (2001) Random forests. Machine Learning 45(1):5–32

Breiman L, Friedman J, Stone CJ, Olshen RA (1984) Classification and regression trees. CRC press

Buza K (2020) Asterics: Projection-based classification of eeg with asymmetric loss linear regression and genetic algorithm. In: 2020 IEEE 14th International Symposium on Applied Computational Intelligence and Informatics (SACI), pp 35–40, DOI 10.1109/SACI49304.2020.9118837

Chandrashekar G, Sahin F (2014) A survey on feature selection methods. Computers & Electrical Engineering 40(1):16–28

Chang CC, Lin CJ (2011) LIBSVM: A Library for Support Vector Machines. ACM Transactions on Intelligent Systems and Technology 2(3):27:1–27:27

Chen X, Yuan G, Nie F, Huang JZ (2017) Semi-supervised feature selection via rescaled linear regression. In: Proceedings of the 26th International Joint Conference on Artificial Intelligence, vol 2017, pp 1525–1531

Da Silva SF, Ribeiro MX, Neto JdEB, Traina-Jr C, Traina AJ (2011) Improving the ranking quality of medical image retrieval using a genetic feature selection method. Decision support systems 51(4):810–820

Darst BF, Malecki KC, Engelman CD (2018) Using recursive feature elimination in random forest to account for correlated variables in high dimensional data. BMC genetics 19(1):1–6

Dietterich TG (2000) An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. Machine learning 40(2):139–157

Dua D, Graff C (2017) UCI machine learning repository. URL `https://archive.ics.uci.edu/ml/index.php`

Efron B (1992) Bootstrap methods: another look at the jackknife. In: Breakthroughs in statistics, Springer, pp 569–593

Feofanov V, Devijver E, Amini MR (2019) Transductive bounds for the multiclass majority vote classifier. Proceedings of the AAAI Conference on Artificial Intelligence 33:3566–3573

Frohlich H, Chapelle O, Scholkopf B (2003) Feature selection for support vector machines by means of genetic algorithm. In: Proceedings. 15th IEEE International Conference on Tools with Artificial Intelligence, IEEE, pp 142–148

Goldberg DE, Deb K (1991) A comparative analysis of selection schemes used in genetic algorithms. In: Foundations of genetic algorithms, vol 1, Elsevier, pp 69–93

Goldberg DE, Holland JH (1988) Genetic algorithms and machine learning. Machine learning 3(2):95–99

Guyon I (2003) Design of experiments of the nips 2003 variable selection benchmark. In: NIPS 2003 workshop on feature extraction and feature selection

Guyon I, Elisseeff A (2003) An introduction to variable and feature selection. Journal of machine learning research 3(Mar):1157–1182

Guyon I, Weston J, Barnhill S, Vapnik V (2002) Gene selection for cancer classification using support vector machines. Machine learning 46(1):389–422

Han Y, Park K, Lee YK (2011) Confident wrapper-type semi-supervised feature selection using an ensemble classifier. In: 2011 2nd International Conference on Artificial Intelligence, Management Science and Electronic Commerce (AIMSEC), IEEE, pp 4581–4586

Jiang B, Wu X, Yu K, Chen H (2019) Joint semi-supervised feature selection and classification through bayesian approach. In: Proceedings of the AAAI conference on artificial intelligence, vol 33, pp 3983–3990

Kohavi R, John GH (1997) Wrappers for feature subset selection. Artificial intelligence 97(1-2):273–324

LeCun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. Proceedings of the IEEE 86(11):2278–2324

Li J, Cheng K, Wang S, Morstatter F, Trevino RP, Tang J, Liu H (2018) Feature selection: A data perspective. ACM Computing Surveys (CSUR) 50(6):94

Louppe G (2014) Understanding random forests: From theory to practice. 1407.7502

Madani O, Pennock DM, Flake GW (2005) Co-validation: Using model disagreement on unlabeled data to validate classification algorithms. In: Advances in neural information processing systems, pp 873–880

Mann HB, Whitney DR (1947) On a test of whether one of two random variables is stochastically larger than the other. The Annals of Mathematical Statistics 18(1):50–60

Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E (2011) Scikit-learn: Machine learning in Python. Journal of Machine Learning Research 12:2825–2830

Ren J, Qiu Z, Fan W, Cheng H, Yu PS (2008) Forward semi-supervised feature selection. In: Washio T, Suzuki E, Ting KM, Inokuchi A (eds) Advances in Knowledge Discovery and Data Mining, Springer Berlin Heidelberg, Berlin, Heidelberg, pp 970–976

Schölkopf B (1997) Support vector learning. PhD thesis, Oldenbourg München, Germany

Sechidis K, Brown G (2018) Simple strategies for semi-supervised feature selection. Machine Learning 107(2):357–395

Sheikhpour R, Sarram MA, Gharaghani S, Chahooki MAZ (2017) A survey on semi-supervised feature selection methods. Pattern Recognition 64(C):141–158

Siedlecki W, Sklansky J (1993) A note on genetic algorithms for large-scale feature selection. In: Handbook of pattern recognition and computer vision, World Scientific, pp 88–107

Song L, Smola A, Gretton A, Borgwardt KM, Bedo J (2007) Supervised feature selection via dependence estimation. In: Proceedings of the 24th international conference on Machine learning, pp 823–830

Syed FH, Tahir MA, Rafi M, Shahab MD (2021) Feature selection for semi-supervised multi-target regression using genetic algorithm. Applied Intelligence pp 1–24, DOI 10.1007/s10489-021-02291-9

Szenkovits A, Meszlényi R, Buza K, Gaskó N, Lung RI, Suciu M (2018) Feature selection with a genetic algorithm for classification of brain imaging data. In: Advances in feature selection for data and pattern recognition, Springer, pp 185–202

Tür G, Hakkani-Tür DZ, Schapire RE (2005) Combining active and semi-supervised learning for spoken language understanding. Speech Communication 45:171–186

Tuv E, Borisov A, Runger G, Torkkola K (2009) Feature selection with ensembles, artificial variables, and redundancy elimination. Journal of Machine Learning Research 10:1341–1366

Vapnik VN (1998) Statistical Learning Theory. Wiley-Interscience

Wu X, Chen H, Li T, Wan J (2021) Semi-supervised feature selection with minimal redundancy based on local adaptive. Applied Intelligence pp 1–22

Xiao H, Rasul K, Vollgraf R (2017) Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. `cs.LG/1708.07747`

Xue B, Zhang M, Browne WN, Yao X (2015) A survey on evolutionary computation approaches to feature selection. IEEE Transactions on Evolutionary Computation 20(4):606–626

Yang M, Chen YJ, Ji GL (2010) Semi_fisher score: A semi-supervised method for feature selection. In: 2010 International Conference on Machine Learning and Cybernetics, IEEE, vol 1, pp 527–532

Zhao J, Lu K, He X (2008) Locality sensitive semi-supervised feature selection. Neurocomputing 71(10-12):1842–1849

A Additional Information on Relevance Test

In Section 3.3, we proposed to use the relevance test at every generation by testing just a portion of features, whereas it was originally proposed to be performed once on the whole feature set (Tuv et al., 2009). To validate our choice,

we compare two versions of the genetic algorithm: 1) at first, the relevance test is performed on the whole feature set, features found to be irrelevant are removed, and then FSGA is run without the relevance test step; 2) relevance test is used at every generation of FSGA as described in Section 3.3.

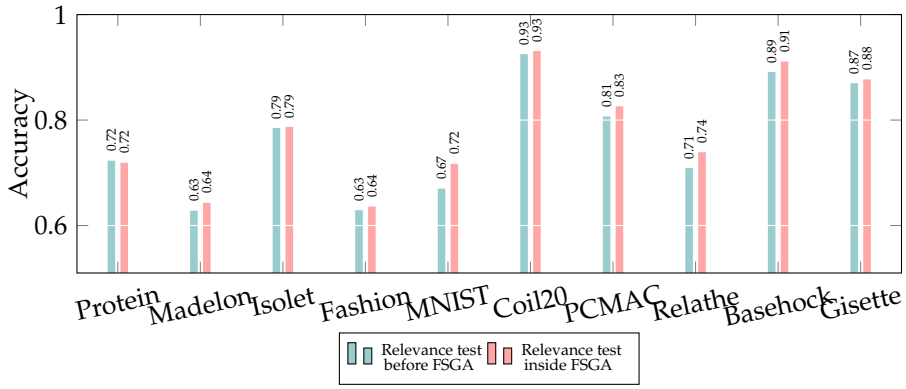


Figure 6: Comparison of 2 versions of FSGA on the benchmark data sets: when the relevance test performed just once before the genetic algorithm, and when it is performed at every generation as it is described in Section 3.3. The accuracy on the unlabeled set (ACC-U) of a classifier trained on the final feature subset is illustrated.

The performance results on the benchmarks are illustrated in Figure 6. As one can see, when the relevance test is iteratively used, the classification accuracy is noticeably higher on most of data sets, particularly on those with a high number of irrelevant features (Madelon, PCMAC, Relathe, Basehock). This suggests that the model learned on the whole feature set suffers from the curse of dimensionality, so some irrelevant variables are not detected based on derived feature weights.

In addition, we demonstrate that integration of the relevance test also significantly improves the classical genetic algorithm (CGA), where the standard crossover is used. Figure 7 illustrates the performance of the standard crossover when successively the mutation step and the relevance test are added, comparing with the FSGA, where the proposed weighted crossover is used. One can see that on the data sets with a large number of irrelevant features (PCMAC, Relathe, Basehock) the performance of CGA is drastically improved. Nevertheless, the proposed weighted crossover tends to outperform the standard crossover, which additionally validates our contribution.

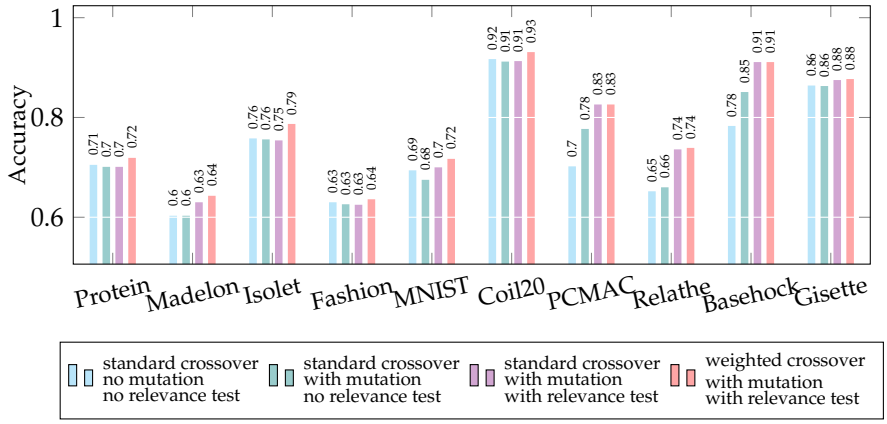


Figure 7: Comparison of 4 different versions of the genetic algorithm on the benchmark data sets: the standard crossover, to which the mutation operator and the relevance test are successively added, and the `FSGA`. Note that the second column corresponds to the classical genetic algorithm (`CGA`). The accuracy on the unlabeled set (`ACC-U`) of a classifier trained on the final feature subset is illustrated.