
SWAG: A Wrapper Method for Sparse Learning

Roberto Molinari

Department of Mathematics and Statistics
Auburn University

Gaetan Bakalli

Geneva School of Economics and Management
University of Geneva

Stéphane Guerrier

Geneva School of Economics and Management and Faculty of Science
University of Geneva

Cesare Miglioli

Geneva School of Economics and Management
University of Geneva

Samuel Orso

Geneva School of Economics and Management
University of Geneva

Olivier Scaillet

University of Geneva (Geneva Finance Research Institute (GFRI)) and
Swiss Finance Institute (SFI)

Abstract

Predictive power has always been the main research focus of learning algorithms with the goal of minimizing the test error for supervised classification and regression problems. While the general approach for these algorithms is to consider all possible attributes in a dataset to best predict the response of interest, an important branch of research is focused on sparse learning in order to avoid overfitting which can greatly affect the accuracy of out-of-sample prediction. However, in many practical settings we believe that only an extremely small combination of different attributes affect the response whereas even sparse-learning methods can still preserve a high number of attributes in high-dimensional settings and possibly deliver inconsistent prediction performance. As a consequence, the latter methods can also be hard to interpret for researchers and practitioners, a problem which is even more relevant for the “black-box”-type mechanisms of many learning approaches. Finally, aside from needing to quantify prediction uncertainty, there is often a problem of replicability since not all data-collection procedures measure (or observe) the same attributes and therefore cannot make use of proposed learners for testing purposes. To address all the previous issues, we propose to study a procedure that combines screening and wrapper methods and aims to find a library of extremely low-dimensional attribute combinations (with consequent low data collection and storage costs) in order to (i) match or improve the predictive performance of any

particular learning method which uses all attributes as an input (including sparse learners); (ii) provide a low-dimensional network of attributes easily interpretable by researchers and practitioners; and (iii) increase the potential replicability of results due to a diversity of attribute combinations defining strong learners with equivalent predictive power. We call this algorithm “Sparse Wrapper Algorithm” (SWAG).

Keywords: interpretable machine learning, big data, wrapper, sparse learning, meta learning, ensemble learning, greedy algorithm, feature selection, variable importance network.

JEL Codes: C45, C51, C52, C53, C55, C87.

1 Motivation

We typically judge the efficacy of a given supervised learning mechanism with its prediction capability. Indeed, once we have trained a learner, we generally measure its performance on a test set and eventually on a validation set. Following this procedure, different learners can be compared in order to understand which one appears to provide the best performance for a given test set and, given possible ties, choices are made based on other criteria such as, for example, computational efficiency. In the latter sense, an important area of research is that of *sparse* learning (see e.g. [7, 49]) which, in the context of this work, refers to the use of learners that select and make use only of a reduced number of attributes in a dataset (as opposed to all of them). Indeed, aside from guaranteeing reduced computational complexity for prediction when employing fewer attributes and therefore reducing costs for data collection and storage, sparse learners are effective in addressing the common problem of overfitting by excluding attributes that are possibly not informative and that can increase prediction variability. In addition, by selecting and making use of a small set of attributes, sparse learners can also lend themselves to being more easily interpreted and can consequently be used as a basis to further investigate certain phenomena (for an overview see [7]).

While a variety of sparse learning mechanisms have been proposed over the years, each of them is subject to certain limitations. For example, “screening (filtering) methods” are a common and computationally (reasonably) efficient approach even for datasets with an extremely large number of attributes where a screening of the attributes is made in a first-step using various information measures, and successively the screened attributes are used within a learning mechanism (see e.g. [7, 10]). However, the initial screening is often disconnected from the actual prediction capability of the chosen learning mechanism thereby running the risk of not including potentially important attributes within the learner. Another general approach are so-called “embedded methods” where the optimization of the learner and the selection of attributes is performed simultaneously. The main example of this approach are the regularized empirical risk minimization techniques such as Ridge Regression [20], Lasso [37] and Elastic-Net [50] and their non-linear adaptations (see e.g. [19] for a review). While, under certain assumptions, these approaches are ideal since they integrate the learning problem with the attribute-selection problem, embedded methods can run into different practical issues and, in extremely high dimensional settings ($p \gg n$) can still select many attributes (despite important penalization) thereby hindering interpretation (see e.g. [29]). Compared to the mentioned approaches, this paper puts forward a heuristic procedure that, while making use of screening procedures, falls within the “wrapper methods”. These methods make use of the loss-functions used to optimize learning mechanisms and use heuristic algorithms to search the attribute space while measuring the predictive performance of the explored attribute subsets (see e.g. [22]). Examples of such methods are step-wise forward selection or backward elimination approaches (see e.g. [16, 25, 48]) where the choice of each added or removed attribute is based on the predictive performance of the previous step. The latter procedures belong to the general class of *greedy algorithms* (see e.g. [4, 36]) since the direction in which they explore the attribute space is strictly dependent on the results obtained at their previous step. This can lead to sub-optimal solutions (e.g. convergence to local-minima) and, in large dimensions, can be computationally expensive also depending on the complexity of the learning mechanisms.

Despite their many advantages, all the current sparse learning mechanisms select a single learner (with corresponding unique attributes) and, in high-dimensional settings, nevertheless tend to select many attributes especially in a highly correlated environment (see e.g. [31, 42]). These features can have

important practical impacts in different settings where, for example, replicability and interpretability of the learners is of relevance (see e.g. [27]). Indeed, from genomics (e.g. [47]) to online prediction (e.g. [3]), there are many tasks that require a degree of flexibility in the use of a multitude of (small) subsets of attributes while preserving high predictive performance (as hinted also in [46]). For example, (i) in medical studies machines collect different measurements (attributes) for a specific problem (e.g. [11]); (ii) for online search algorithms every subject provides different attributes (according to their preferences or willingness to disclose information) to determine suggestions or matches (e.g. [43]); (iii) in pattern recognition, images are collected at different resolutions and therefore a single learner may not be flexible enough to adapt to different image features (e.g. [45]). The idea of having a library of learners (and possibly of attributes) was considered, for example, in [5] where libraries are created by forward-selection based on ensemble prediction. In this direction, this work puts forward a new greedy version of the algorithm that was proposed for gene selection problems in [15] which overcomes some of its limitations when exploring the attribute space and aims at creating a library of strong low-dimensional learners which can either be used individually or in an ensemble approach. We call this development the "Sparse Wrapper Algorithm" (SWAG for short).

The main goal of the proposed algorithm is to explore the attribute space in an "informed" manner to improve the prediction performance of any learning mechanism or, at least, preserve its predictive performance while using a considerably smaller set of attributes compared to those used if the mechanism were applied to the entire dataset (with consequent gains in terms of data collection and storage). In addition, the algorithm aims at delivering a library of strong sparse learners each containing a small number of (diverse) attributes which not only can be used in settings where only certain attributes are collected (replicability) with no loss in predictive power but can also be interpreted, on their own or collectively, in order to better understand phenomena (interpretability) and provide possible future directions of research in different domains.

2 Sparse Wrapper Algorithm

As mentioned above, the SWAG combines screening approaches within a general wrapper method. In order to find a library (set) of learners that take a small number of attributes as inputs and that have high predictive power, the algorithm proceeds in a forward-step manner. More specifically, it builds and tests learners starting from very few attributes until it includes a maximal number of attributes by increasing the number of attributes at each step. Hence, for each fixed number of attributes, the algorithm tests various (randomly selected) learners and picks those with the best performance in terms of training error. Throughout, the algorithm uses the information coming from the best learners at the previous step to build and test learners in the following step. In the end, it outputs a set of strong low-dimensional learners.

Given the above intuitive explanation, we now provide a more formal description and introduce basic notations. Let $\mathbf{y} \in \mathbb{R}^n$ denote the response and $\mathbf{X} \in \mathbb{R}^{n \times p}$ denote an attribute matrix with n instances and p attributes, the latter being indexed by a set $\mathcal{S} := \{1, \dots, p\}$. In addition, we denote a generic learning mechanism as $\mathcal{L} := \mathcal{L}(\mathbf{y}, \mathbf{X})$ with l denoting a general learner which we build by using (i) the learning mechanism \mathcal{L} and (ii) a subset of attributes in \mathbf{X} . Finally, we let $\mathcal{P}(\mathcal{A})$ and $|\mathcal{A}|$ denote respectively the power set and cardinality of a set \mathcal{A} . In the following paragraphs, we describe the algorithm and introduce meta-parameters with interpretation and selection discussed in Sec. 2.1.

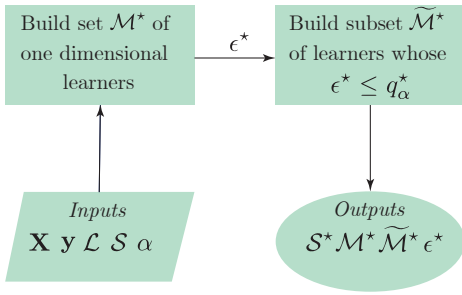
The first choice to make for the algorithm is to determine the maximum dimension of attributes that the user wants as input in a learner. We denote this parameter as $p_{\max} < p$. Based on this parameter, SWAG aims at exploring the space of attributes in order to find sets of learners using \hat{p} attributes ($1 \leq \hat{p} \leq p_{\max}$) with extremely high predictive power. To do so, the algorithm makes use of the step-wise screening procedure described in the following paragraphs.

First Step The first screening step starts by using one *distinct* attribute at a time to create p learners. Once these learners are built, a set of learners \mathcal{M}^* is now available which is indexed by the ordered index set $\mathcal{I} := \{1, \dots, p\}$ (i.e. each learner $l \in \mathcal{M}^*$ is indexed by a unique element $i \in \mathcal{I}$). Having chosen a measure of predictive error (e.g. a loss function such as the misclassification rate for classification problems), one can then apply r -repeated k -fold cross-validation to determine the training error of each learner $l \in \mathcal{M}^*$. We denote the vector containing the cross-validation error as $\epsilon^* \in \mathbb{R}^p$ which is also indexed by the set \mathcal{I} (i.e. we associate each learner $l \in \mathcal{M}^*$ with an element

in the training error vector ϵ^*). Given this, it is now possible to select a performance quantile q_α^* with $\alpha \in (0, 1)$ set by the user. We define this quantile such that the following expression holds:

$$\frac{1}{p} \sum_{i=1}^p \mathbb{I}_{\{\epsilon_i^* \leq q_\alpha^*\}} = \alpha,$$

where \mathbb{I}_A is the indicator function that takes the value 1 if A is true and 0 otherwise, while ϵ_i^* denotes the i^{th} element of the vector ϵ^* . The smaller the value of α , the smaller the selected training errors. The procedure then selects all the learners whose training error is smaller or equal to q_α^* and includes these in a new learner set $\widetilde{\mathcal{M}}^*$. The set $\widetilde{\mathcal{M}}^*$ collects one-dimensional attribute learners with high predictive power and therefore also contains a subset of attributes $\mathcal{S}^* \subset \mathcal{S}$. We can assume that subset to be highly informative with respect to the response of interest \mathbf{y} . We describe this procedure in Algo. 1.



Algorithm 1 First Screening Algorithm

INPUTS: A response $\mathbf{y} \in \mathbb{R}^n$ and attributes $\mathbf{X} \in \mathbb{R}^{n \times p}$; An attribute index set $\mathcal{S} := \{1, \dots, p\}$; A learning mechanism \mathcal{L} ; A performance percentile $\alpha \in (0, 1)$; the number of repetitions r and the number of folds k to compute the cross-validation training error.

- 1: Using the mechanism \mathcal{L} , build p learners by using all attributes in the set \mathcal{S}
- 2: Create a learner set \mathcal{M}^* (with $|\mathcal{M}^*| = p$)
- 3: Build a r -repeated k -fold cross-validation error vector $\epsilon^* \in \mathbb{R}^p$ and identify the α -quantile q_α^* of this vector
- 4: Create new learner set $\widetilde{\mathcal{M}}^*$ with learners whose cross-validation error is smaller or equal to q_α^*
- 5: Create attribute index set \mathcal{S}^* with attributes included in the learners in the set $\widetilde{\mathcal{M}}^*$.

OUTPUTS: \mathcal{S}^* ; $\widetilde{\mathcal{M}}^*$; \mathcal{M}^* ; ϵ^* .

General Step Fixing a given attribute dimension \hat{p} such that $2 \leq \hat{p} \leq p_{\max}$, the general screening step builds a maximum number m of *distinct* learners, all included in a learner set $\mathcal{M}^{\hat{p}}$, that each take combinations of \hat{p} *distinct* attributes. In order to build the m learners, the general step takes the attribute index set \mathcal{S}^* from Algo. 1 and a set of learners $\widehat{\mathcal{M}}$ in which each learner is of dimension $\hat{p} - 1$ (i.e. each learner in $\widehat{\mathcal{M}}$ takes $\hat{p} - 1$ attributes as an input). We let $s_l \in \widetilde{\mathcal{S}} := \{s \in \mathcal{P}(\mathcal{S}^*) \mid |s| = \hat{p} - 1\}$ denote the attribute indices for a specific learner $l \in \widehat{\mathcal{M}}$.

In order to build the m learners, the procedure first verifies whether we can use all possible attribute combinations. Indeed, if we denote $p^* := |\mathcal{S}^*|$, then the total number of learners that we can build is $\tilde{m} := \binom{p^*}{\hat{p}}$ and, if $m \geq \tilde{m}$, then the learner set $\mathcal{M}^{\hat{p}}$ contains all possible learners. However, if $m < \tilde{m}$, then this step randomly selects a learner $l \in \widehat{\mathcal{M}}$ and it builds a new learner by using the attributes indexed by s_l and a randomly selected distinct attribute from the attribute index set \mathcal{S}^*/s_l . We repeat this step until we obtain m distinct learners and include them in the learner set $\mathcal{M}^{\hat{p}}$.

Once the candidate learner set $\mathcal{M}^{\hat{p}}$ is built, the general step of the algorithm closely follows Algo. 1. Specifically, we build an r -repeated k -fold cross-validation error $\epsilon^{\hat{p}}$ and compute its performance quantile $q_\alpha^{\hat{p}}$. The main output of this step is then a new learner set $\widetilde{\mathcal{M}}^{\hat{p}}$ which includes all learners whose training error in $\epsilon^{\hat{p}}$ is smaller or equal to $q_\alpha^{\hat{p}}$ (with both $\epsilon^{\hat{p}}$ and $\mathcal{M}^{\hat{p}}$ ordered by the same index set $\mathcal{I}^{\hat{p}} := \{1, \dots, m\}$). We describe this general step in Algo. 2.

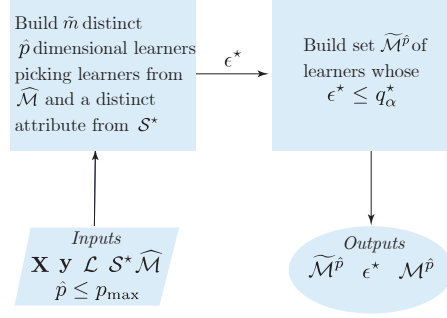
As *per* its name, the above described algorithms provide a straightforward manner of selecting a set of predictive learners for a given attribute dimension \hat{p} . However, as the attribute dimension increases, the number of possible distinct attribute combinations increases exponentially fast. Therefore, as the attribute dimension grows, there is an increased risk of inefficiently exploring the attribute space if we simply randomly picks m attribute combinations. For this reason, SWAG performs a *greedy* procedure which uses the information from Algo. 1 to obtain the set of best attributes \mathcal{S}^* which is the easiest to explore completely. The next step of the algorithm then takes the set \mathcal{S}^* and the set of best learners $\widetilde{\mathcal{M}}^*$ as the input $\widehat{\mathcal{M}}$ for Algo. 2 for attribute dimension $\hat{p} = 2$. At each of the following

Algorithm 2 General Screening Algorithm

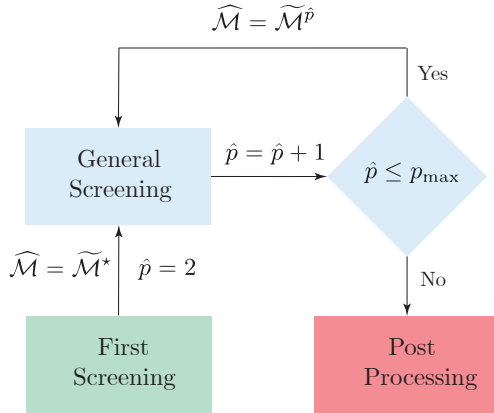
INPUTS: A response $\mathbf{y} \in \mathbb{R}^n$ and attributes $\mathbf{X} \in \mathbb{R}^{n \times p}$; An attribute index set $S^* \subset \{1, \dots, p\}$ from Algo. 1; A number of attributes $\hat{p} \leq p_{\max}$; A learner set $\widehat{\mathcal{M}}$; A learning mechanism \mathcal{L} ; A maximum number of learners m ; A performance percentile $\alpha \in (0, 1)$; the number of repetitions r and the number of folds k to compute the cross-validation training error.

- 1: Define $\tilde{m} := \binom{p^*}{\hat{p}}$
- 2: **if** $\tilde{m} \leq m$ **then**
- 3: Using the mechanism \mathcal{L} , build all possible \tilde{m} learners with \hat{p} attribute inputs to create learner set $\mathcal{M}^{\hat{p}}$
- 4: **else**
- 5: Using the mechanism \mathcal{L} , build m learners with \hat{p} attribute inputs by extracting s_l from randomly sampled learners $l \in \widehat{\mathcal{M}}$ and adding an attribute from S^* / s_l to create learner set $\mathcal{M}^{\hat{p}}$
- 6: **end if**
- 7: Build an r -repeated k -fold cross-validation error vector $\epsilon^{\hat{p}}$ and identify the α -quantile $q_{\alpha}^{\hat{p}}$ of this vector
- 8: Create new learner set $\widetilde{\mathcal{M}}^{\hat{p}}$ with learners whose cross-validation error is smaller or equal to $q_{\alpha}^{\hat{p}}$

OUTPUTS: $\widetilde{\mathcal{M}}^{\hat{p}}; \mathcal{M}^{\hat{p}}; \epsilon^{\hat{p}}$.



steps $\hat{p} > 2$, the algorithm defines $\widehat{\mathcal{M}} := \widetilde{\mathcal{M}}^{\hat{p}-1}$. Therefore, when increasing the attribute dimension, the algorithm only considers attribute combinations based on “informative” learners and attributes from the previous dimension. We repeat this procedure for all attribute dimensions until we reach the maximal dimension p_{\max} . Throughout the procedure, the algorithm saves the strong learner sets (whose training error is below the computed quantile) and training error vectors for each dimension \hat{p} . This procedure defines the SWAG outlined in Algo. 3.



Algorithm 3 SWAG

INPUTS: A response $\mathbf{y} \in \mathbb{R}^n$ and attributes $\mathbf{X} \in \mathbb{R}^{n \times p}$; An attribute index set $S := \{1, \dots, p\}$; A learning mechanism \mathcal{L} ; A maximum number of attributes $p_{\max} (< p)$; A maximum number of learners m for each step of the algorithm; A performance percentile $\alpha \in (0, 1)$; the number of repetitions r and the number of folds k to compute the cross-validation training error.

- 1: Run Algo. 1 using inputs $\mathbf{y}, \mathbf{X}, S, \mathcal{L}, m, \alpha$ and obtain S^* and $\widehat{\mathcal{M}}^*$
- 2: $\widehat{\mathcal{M}} \leftarrow \widehat{\mathcal{M}}^*$
- 3: $\hat{p} \leftarrow 2$
- 4: **while** $\hat{p} \leq p_{\max}$ **do**
- 5: Run Algo. 2 using inputs $\mathbf{y}, \mathbf{X}, S^*, \widehat{\mathcal{M}}, \mathcal{L}, m, \alpha$ and obtain $\widetilde{\mathcal{M}}^{\hat{p}}$
- 6: $\widehat{\mathcal{M}} \leftarrow \widetilde{\mathcal{M}}^{\hat{p}}$
- 7: $\hat{p} \leftarrow \hat{p} + 1$
- 8: **end while**
- 9: Create
 - a set of strong learner sets $\widetilde{\mathcal{M}} := \{\widetilde{\mathcal{M}}^1, \widetilde{\mathcal{M}}^2, \dots, \widetilde{\mathcal{M}}^{p_{\max}}\}$
 - a set of training error vectors $\tilde{\epsilon} := \{\epsilon^1, \epsilon^2, \dots, \epsilon^{p_{\max}}\}$

OUTPUTS: $\widetilde{\mathcal{M}}; \tilde{\epsilon}$

Post-Processing The output of SWAG is a set of learners with high predictive power for each dimension of attribute combinations smaller or equal to p_{\max} . The user could choose to directly make use of this set to perform predictions based on different sets of attributes or arrange attributes into networks for interpretation and exploration. However, these sets of learners could undergo an additional screening procedure according to the needs of the user. For example, an approach that we use for the applications in Sec. 3 is to compute the median training error for each vector in the set $\tilde{\epsilon}$ (as defined in Algo. 3) and select the quantile \tilde{q}_{δ} corresponding to the dimension whose median is the lowest, where $\delta \in (0, 1)$ can differ from α (a possible choice is 0.01). Having identified this quantile,

the user can then select the learners (with the desired dimensions) whose training error is smaller or equal to this quantile.

Computational Complexity SWAG is a computationally intense procedure since it builds at most $p + m(p_{\max} - 1)$ learners. This implies that the order of computation for any learner is multiplied by this factor. In addition, the r -repeated k -fold cross-validation will add complexity proportionally to the values of r and k . Nevertheless, the learners are built on an extremely small number of attributes. This implies computational efficiency for those mechanisms whose complexity highly depends on the number of attributes. Besides, we can perform each fitting and cross-validation in parallel. We leave a more formal study of the complexity of SWAG and its possible improvement for further research.

2.1 Meta-Learning Options

For the application of SWAG, we assume the user has chosen a learning mechanism \mathcal{L} for which he would like to improve upon its predictions and/or would like to obtain more interpretable and replicable learners without losing predictive power. Based on this, the user has to define the meta-parameters of the algorithm and eventually choose an ensemble approach to aggregate the strong learners that are output.

Meta-Parameters The main user-defined parameters for SWAG are (i) the maximum attribute dimension p_{\max} ; (ii) the maximum number of learners m to build within each step and (iii) the percentile probability level α . Ideally, with unlimited computing power, the first two parameters would be as large as possible, i.e. $p_{\max} = p$ and $m = \binom{p}{\lceil p/2 \rceil}$. However, this defeats the purpose of the algorithm and therefore we must set these parameters according to interpretability/replicability requirements as well as available computing power and time. Below we list several rules-of-thumbs for the choice of these parameters:

- p_{\max} : For given available computing power and efficiency of the learning mechanism \mathcal{L} , this parameter depends on the total dimension of the problem p . Indeed, the goal of SWAG is to find extremely small dimensional learners. Therefore, even with very large p , one could always fix this parameter within a range of 5-20 (or smaller) for interpretability and/or replicability purposes. In addition, if an embedded method is computationally efficient to compute on the entire dataset, this parameter could be the number of selected attributes through this method (given computational constraints). Another criterion, when working with binary classification problems, is to use the *Event Per Variable* (EPV) rule presented in [44] (see Sec. 3 for example). In future work, this parameter can be implicitly determined by the algorithm based on the training error quantile (or other metric) thereby defining p_{\max} as the attribute dimension where the training error curve stops decreasing significantly similarly to the scree plot in factor or principal component analysis (see e.g. [6]).
- m : For given available computing power and efficiency of the learning mechanism \mathcal{L} , this parameter determines the proportion of attribute space explored by the algorithm. We know that it depends on the size of the problem p since we necessarily have $m \geq p$ for the screening step of Algo. 1. In addition, we need to choose this parameter considering the percentile α : if m is small and α is small, then the number of strong learners that we select could be extremely low (possibly zero). In general, we would want a large m (so that α can eventually be chosen to be very small) and, remembering that p^* is the number of attributes released from Algo. 1, a rule-of-thumb is to set $m = \binom{p^*}{2}$ (or close to it) in order to explore the entire (or most of the) subspace of two-dimensional learners generated by p^* .
- α : as discussed in the previous point, this parameter is related to the maximum number of learners m . As α gets larger, we explore a greater portion of the attribute space. Ideally, we want to choose a small α since we would want to select strong learners (with extremely low training error) and this is possible if m is large enough. Generally good values for α are 0.01 or 0.05, implying that we select (roughly) 1% or 5% of the m learners at each step.

Ensemble Learning The main output of SWAG is a set $\widetilde{\mathcal{M}}$ of strong low-dimensional learners. While we can use them individually to obtain accurate predictions in different settings (where, for example, different attributes are available) or collectively to generate interpretable networks, we can also use them together in an ensemble approach. For example we can include them in Bagging

[1], Boosting [35] or other model-averaging approaches (see e.g. [34]) to increase stability and prediction accuracy. In this sense, Sec. 3 includes examples of a “majority-rule” averaging approach for classification problems using the set $\widetilde{\mathcal{M}}$.

3 Empirical Results

We study the empirical performance of SWAG with different learning mechanisms and on different datasets taken from the UCI Machine Learning Repository (see [12]) and ArrayExpress (see [23]). However, we choose to focus on a specific dataset in order to discuss the algorithm in more detail and then summarize the results on the other datasets in the appendix 5. The chosen dataset is the Meter A data analyzed in [17] and collecting measurements on ultrasonic flowmeter diagnostics. Achieving good diagnostics regarding the health of a flowmeter is of extreme importance for condition-based maintenance in many industrial sectors such as the oil and gas industry. Incorrect measurement can entail considerable economic and material losses (see e.g. [40]). In this data there are $n = 87$ instances and $p = 36$ attributes to classify the health of a meter into two classes: “Healthy” (Class 1) or “Installation effects” (Class 2). Given that the attributes are measurements of physical nature, we decide to consider all first-order interactions which finally delivers a total attribute size of $p = 666$ (36 original attributes plus $\binom{36}{2} = 630$ interactions).

We choose to apply SWAG using four different learning mechanisms, namely: (i) Lasso (logistic) ([13]); (ii) Linear-Kernel Support Vector Machine (SVM) (see e.g. [41]); (iii) Radial-Kernel SVM ([8]); (iv) Random Forest ([2]). To ensure a fair comparison, we run all the analyses with the `caret` R package (see e.g. [24] for a review). We set all the hyper-parameters specific to each learning mechanism at their common/default choices (see the discussion in appendix 5). For all these mechanisms, we apply SWAG using the following meta-parameters. Since $p = 666$, we choose to set $\alpha = 0.05$ as a good compromise between fixing an α as small as possible (in order to select strong learners) and the possibility of exploring a considerable portion of the attribute space (see Sec. 2.1). With m depending on the computing time and power available, we choose $m = 4 \cdot 10^4$ in order to explore the entire subspace of three-dimensional learners for all the learning mechanisms. We fix $p_{\max} = 6$ following the EPV rule discussed in [44]. For this specific dataset, we train on 80% of the dataset delivering 28 instances for one class and 41 for the other and, aiming for an EPV roughly between 4 and 7 (see [44]), we find that the chosen value of p_{\max} implies an EPV of 4.67. Finally, we choose to apply 10-repeated 10-fold cross-validation (i.e. $r = 10$ and $k = 10$) within SWAG, and $\delta = 0.01$ for the post-processing.

Tab. 1 collects the SWAG results in terms of training error (ϵ_{train}) and test error (ϵ_{test}), together with a plot of their distribution (i.e. a histogram of the errors for of all SWAG learners). We see that the range of training errors for SWAG learners is consistently smaller than that of their original version. An exception is the Linear-Kernel SVM where it appears to be the opposite. However, we can see the true advantage of SWAG learners occurs in terms of their test error. It highlights how they appear to generalize better than their original counterparts. Indeed, in all cases SWAG learners’ test error is consistently and, in most cases, considerably lower than their original version. In addition, if we use a simple learner-averaging approach for SWAG learners (majority-rule in this case) we can see that the test error is almost always zero or, in any case, is consistently smaller than test errors (and often training errors) of the original versions. More importantly, we must consider all these results in light of the extremely small number of attributes used within SWAG learners: in all cases these results are achieved using between 2 to 6 attributes while, at best, the original counterparts select 12 attributes (Lasso) or all the attributes (666).

In relation to the latter result on learner dimension, SWAG has two additional advantages consisting in *interpretability* and *replicability* of its outputs. The replicability feature simply consists in the availability of a library of learners from which researchers and practitioners can select the information (attributes) of interest. We highlight the interpretability feature, for the Meter A dataset, in Fig. 1. Using the results from Lasso-based SWAG, we can see how we can arrange the attributes most frequently included in the selected learners (Table on the right of Fig. 1) into an informative network. Its edges represent the most common connections between these attributes (left of Fig. 1). Therefore, in order to understand the mechanics and perform diagnostics for this ultrasonic flowmeter, among the 666 attributes, a researcher could for example focus his attention on the interaction (i) between flatness ratio and gain as well as (ii) between the speed of sound and gain at the first end (of the fifth path). As a final note, given that cost-based maintenance can have asymmetric costs according to the

Table 1: Results for the Meter A dataset with the four considered learning mechanisms. For each learning mechanism \mathcal{L} , we have the range of training errors for SWAG learners and the single error for the original mechanism (first row) as well as those of the testing errors (second row). For each row, we plot the distribution of these errors for SWAG learners (third column for each mechanism). The third row for each mechanism shows the Majority-Rule procedure (MR) applied to SWAG learners (no available results for the original version since there is only a single learner). The last row collects the number of attributes (dimension) included in each set of learners.

	Lasso-logistic			Linear-Kernel SVM		
	SWAG	Original		SWAG	Original	
ϵ_{train}	[0.013, 0.019]	0.116		[0.256, 0.259]	0.097	
ϵ_{test}	[0, 0]	0.333		[0, 0.055]	0.722	
MR	0	-		0	-	
$ s_l $	[4, 6]	12		[2, 6]	666	

	Radial-Kernel SVM			Random Forest		
	SWAG	Original		SWAG	Original	
ϵ_{train}	[0.007, 0.016]	0.101		[0.044, 0.050]	0.079	
ϵ_{test}	[0, 0.055]	0.388		[0, 0.278]	0.278	
MR	0	-		0.111	-	
$ s_l $	[4, 6]	666		[4, 6]	666	

decision taken on the flowmeter, SWAG would allow to select learners based on the corresponding (non-convex) cost-function instead of the symmetric kind of loss (i.e. each type of error is weighted equally) typically used to design learning mechanisms (see e.g. [9, 28]).

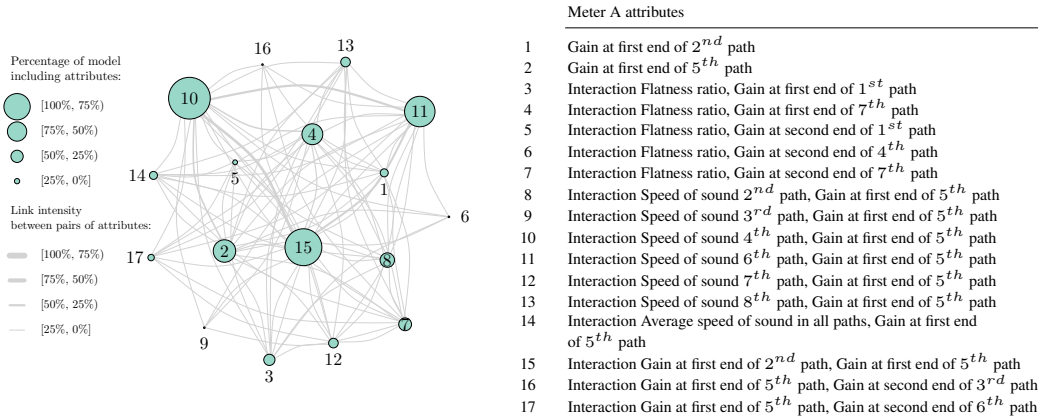


Figure 1: Network of attribute importance and pairwise link-intensity in the set of learners $\tilde{\mathcal{M}}$ for the Meter A dataset using the Lasso-based SWAG results.

We expect the choice of the meta-parameters to impact SWAG results. Given that p_{max} depends on the user's preference in terms of sparsity (or is guided by rules such as the EPV), the main parameters to study are α and m . Tab. 2 gathers the results of a sensitivity analysis on the Meter A data when using the Linear-Kernel SVM within SWAG with different values of these parameters. Those robustness checks show that the choice of m does not appear to greatly affect the results while the choice of α does. Indeed, as the value of α decreases, so does the testing error (and the learner dimension). We expect this effect since a smaller value of α selects stronger learners at each screening step.

Table 2: Sensitivity analysis of meta-parameters m and α for the Meter A dataset with Linear-Kernel SVM.

	$m = 10,000$			$m = 40,000$		
	$\alpha = 0.2$	$\alpha = 0.1$	$\alpha = 0.05$	$\alpha = 0.2$	$\alpha = 0.1$	$\alpha = 0.05$
ϵ_{test}	[0.056, 0.500]	[0, 0.111]	[0, 0.056]	[0.056, 0.722]	[0, 0.667]	[0, 0.056]
MR	0	0	0	0	0	0
$ s_l $	[5, 6]	[2, 6]	[2, 6]	[4, 6]	[2, 6]	[2, 6]

As mentioned earlier, we have run SWAG on other datasets (i.e. LSVT speech recognition and AHUS breast cancer data) using the same learning mechanisms considered in this section (see Appendix 5). For these datasets, SWAG learners have consistently better training errors than their original versions, and their test error intervals always include those of the original versions. As a direct consequence, there are SWAG learners with same or better test error and with considerably fewer attributes: between 3 and 6 for LSVT data (10 or 312 for original versions) and between 5 and 8 for AHUS data (17 or 15739 for original versions). SWAG favours parsimony in all our three empirical applications, without sacrificing prediction accuracy.

4 Broader Impact

SWAG does not have any further negative or positive outcomes, in terms of ethical aspects or future societal consequences, beyond those already characterizing the inner learning mechanisms. Indeed, being a screening-based wrapper method, it consists in an algorithm that aims at finding attribute subspaces that can greatly enhance the performance of any given learning mechanism. Hence, it does not necessarily modify the inherent characteristics of the latter mechanisms. A feature that nevertheless can modify these characteristics, is the higher probability for SWAG to select learners with fewer redundant attributes (e.g. correlated attributes). In fact, based on its screening procedures, SWAG selects learners that improve predictive performance. As such, at each increasing step, it reduces the probability of selecting learners that do not improve their performance by adding redundant attributes. The latter attributes will not add further information than that already provided by the attributes included in the learner sets from the previous steps. Consequently, we often avoid the unnecessary selection of new “redundant” learners to achieve parsimony during the screening procedure.

In terms of practical consequences, an immediately noticeable impact is that SWAG, aside from considerably improving (or matching) the performance of a chosen learner using fewer attributes (with savings in data collection and storage), provides a reasonable approach to assess the prediction uncertainty of a given learner. In addition, SWAG enables to compare different learners based on the distribution of their training and/or test errors. Indeed, if we do not require a low-dimensional strong learner, then we can always use the SWAG error distribution to understand whether the chosen learning mechanism appears to be better or at least comparable. In this sense, the SWAG error distribution can provide a “validation metric”, in the direction outlined by [33], to justify the use of a particular learning mechanism for a given dataset (in the same way a goodness-of-fit measure is used in statistical inference).

A more general impact of SWAG lies in its library of strong low-dimensional learners. Indeed, the algorithm can integrate the advantages of mechanisms such as Random Forest. For this specific choice, we can retrieve the importance of attributes thanks to the Mean Decrease Impurity (MDI) or the Mean Decrease Accuracy (MDA) of each predictor in the forest (see e.g. [21, 26]). On the other hand, SWAG delivers a set of strong learners that we can arrange into low-dimensional and interpretable networks. Similarly to stability selection algorithms [30], we can consider the most common attributes of the network as the main hubs. Moreover each attribute, connected to these main hubs, can be either a secondary hub or “synonym” of the other attributes (i.e. the predictive power is equivalent when using main-hub attribute A together with secondary-hub attribute B or C). Therefore, SWAG can make interpretable even learning mechanisms classified as of the “black-box”-type. Moreover, based on the concept of “synonyms”, the multiplicity of strong learners delivered by SWAG can allow to adequately respond to predictive needs even in cases where many attributes are not collected or stored. Indeed, when dealing with questionnaires or biological testing for example,

not all the responses or measurements are at disposal. If we get access to a handy library of learners providing accurate prediction, we can pick the learner(s) that best suit(s) the available information. The latter is the “replicability” advantage of SWAG which, along with its “interpretability” feature, can be of great impact in many fields such as the natural sciences and medical research. In the latter fields, it is often the case that machines and materials used to collect measurements during experiments or tests can only store or process certain types of information. For example we may not be able to exploit results from other research to adequately predict responses (e.g. presence or absence of tumors). SWAG not only allows to interpret biological information (through the creation of gene networks for example) but also to increase the probability of making correct diagnoses even if a limited amount of genetic or other information (attributes) is accessible.

Finally, the inherent nature of SWAG lends to the selection of strong learners based on non-convex loss functions that can be different from those used to optimize the learners themselves. In many practical settings the goal of a learning procedure may not necessarily be to minimize (training and/or testing) errors in general, but would ideally be to minimize certain “types” of error. For example, in the medical setting, the goal of a prediction would be to decide whether or not to provide a patient with a certain treatment. In this case, providing a treatment when it is *not* needed (type-I error or false positive) may be less “costly” than *not* providing a treatment when it *is* needed (type-II error or false negative). Depending on the “cost” of each decision, we could evaluate the cross-validation error within SWAG based on such an asymmetric loss (cost) function and finally select learners that perform best in terms of this metric (see e.g. [38]). We believe that this can have profound impacts in many sectors, from machine-maintenance to patient treatments, where we often characterize decision-making by asymmetric costs and the availability of little or very specific sets of information.

Acknowledgments and Disclosure of Funding

This work was supported in part by the SNSF Professorships Grant 176843, in part by the Innosuisse-Boomerang Grant 37308.1 IP-ENG, in part by the NSF under Grants SES-1534433 and SES-1853209 and SES-182582 and in part by the NCATS-NIH under Grant UL1 TR002014. We performed all the computations at University of Geneva on the Baobab cluster.

5 Appendix

5.1 Hyper-Parameter Choice

We set all the hyper-parameters specific to each learning mechanism based on common/default choices:

1. **Lasso-logic**: we use the same grid of penalty terms (λ) for the Lasso (logistic) based on 100 values within the range $\lambda \in [0, 0.30]$ ¹.
2. **Linear- and Radial-Kernel SVM**: in both cases we pre-process the data (i.e. centered and rescaled). We keep the default cost parameter $c = 1$ for the Linear-Kernel SVM while for Radial-Kernel SVM we select the cost from a grid of 10 values (i.e. `tuneLength = 10`);
3. **Random Forest**: we keep the default value for the number of trees (i.e. `ntree = 500`) while we make the common choice for the number of randomly sampled attributes at each split (i.e. `mtry = \sqrt{p}` for the original version and `mtry = $\sqrt{\hat{p}}$` for SWAG, see [14] for a detailed discussion).

5.2 Additional Empirical Results

In this section we present the SWAG results for two other datasets: the LSVT voice rehabilitation dataset from the UCI Machine Learning Repository (see [12]) and the Breast Cancer dataset from the ArrayExpress repository (see [23]). We use the same four learning mechanisms as in Sec. 3.

5.2.1 LSVT speech signal processing

The voice rehabilitation dataset was analyzed in [39] in order to assess the effectiveness of a computer program called “Lee Silverman voice treatment (LSVT) Companion”. It allows patients with Parkinson’s disease to independently progress through a rehabilitative treatment session. Taking data on 126 samples from 14 patients who followed the latter treatment, 310 dysphonia measures were taken on each of them (plus information on sex and age of the patients) and used to understand if they could correctly predict whether the patients’ voices were “acceptable” or “unacceptable” after this treatment. Their analysis relies on a robust feature selection to choose 8 attributes (based on the first eight attributes classified by the feature selection method) and subsequently Radial-Kernel SVM was tested (along with Random Forest) to obtain around 90% accuracy in classifying patients’ progress.

We train on 80% of the original data (with 33 “acceptable” and 67 “unacceptable” classes) and test on the remaining 20% (9 “acceptable” and 17 “unacceptable” classes) preserving the ratio of classes. Also in this case, for SWAG we choose $p_{\max} = 6$ which gives an EPV of 5.5. We keep the same value of m used for the Meter A data in Sec. 3 in order to widely explore learners up to dimension 3 (i.e. $m = 4 \cdot 10^4$). We opt for a different strategy for the α parameter. for Algo. 1, given that the total number of attributes is not large ($p = 312$), we decide to set $\alpha = 0.1$ as this choice widens the search on the candidate “informative” attributes. For Algo. 2, we set $\alpha = 0.05$ in order to select strong learners based on these attributes. Finally, we keep the usual value for the post-processing parameter (i.e. $\delta = 0.01$). Tab. 3 collects the SWAG results and the output of the original learning mechanisms.

On one hand, SWAG learners beat the original versions in terms of training error in all cases. On the other hand, the performance on test errors is not uniform and varies according to the learning mechanism. For example, the Lasso or the Radial-Kernel SVM, show how the original version either outperforms SWAG learners or has the same test error of the best SWAG learners. However, we must emphasise that we obtain SWAG learner errors using only roughly 50% of the attributes used in the Lasso (i.e. 4 versus 10) and 1% of the attributes for Radial-Kernel SVM (i.e. 4 versus 312). The interval of SWAG learners, in the case of Linear-Kernel SVM and Random Forest, includes the test errors of the original versions. This indicates that there are learners that obtain better test errors with only 3 to 6 attributes as opposed to 312 for the original versions.

There is also scientific interest in determining the attributes (and combinations thereof) that most contribute to the definition, in this case, of a Parkinson’s speech treatment as being acceptable or not. We can arrange SWAG learners (in this case based on the Radial-Kernel SVM) into an informative

¹In this particular case, we implement Algo. 1 with the `caret` package using classical (non-penalized) logistic regression

Table 3: Results for the LSVT dataset with the four considered learning mechanisms. For each learning mechanism \mathcal{L} , we have the range of training errors for SWAG learners and the single error for the original mechanism (first row) as well as those of the testing errors (second row). For each row, we plot the distribution of these errors for SWAG learners (third column for each mechanism). The third row for each mechanism shows the majority-rule procedure (MR) applied to SWAG learners (no available results for the original version since there is only a single learner). The last row collects the number of attributes (dimension) included in the learners.

	Lasso-logistic			Linear-Kernel SVM		
	SWAG	Original		SWAG	Original	
ϵ_{train}	[0.058, 0.067]	0.138		[0.049, 0.073]	0.186	
ϵ_{test}	[0.115, 0.231]	0.115		[0.115, 0.269]	0.192	
MR	0.192	-		0.154	-	
$ s_l $	[4, 6]	10		[5, 6]	312	

	Radial-Kernel SVM			Random Forest		
	SWAG	Original		SWAG	Original	
ϵ_{train}	[0.039, 0.05]	0.130		[0.087, 0.108]	0.169	
ϵ_{test}	[0.077, 0.231]	0.077		[0.115, 0.269]	0.154	
MR	0.077	-		0.154	-	
$ s_l $	[4, 6]	312		[3, 6]	312	

network amenable to interpretation as in the left part of Fig. 2. Its right part lists the attributes that most contribute to the network. Based on the SWAG network, researchers interested in improving speech treatment should focus on the 2nd and 3rd Mel-Frequency Cepstral coefficients and on the entropy with base-4 logarithmic coefficients (as well as the interactions between these three attributes as highlighted by their frequent presence in the same SWAG learners).

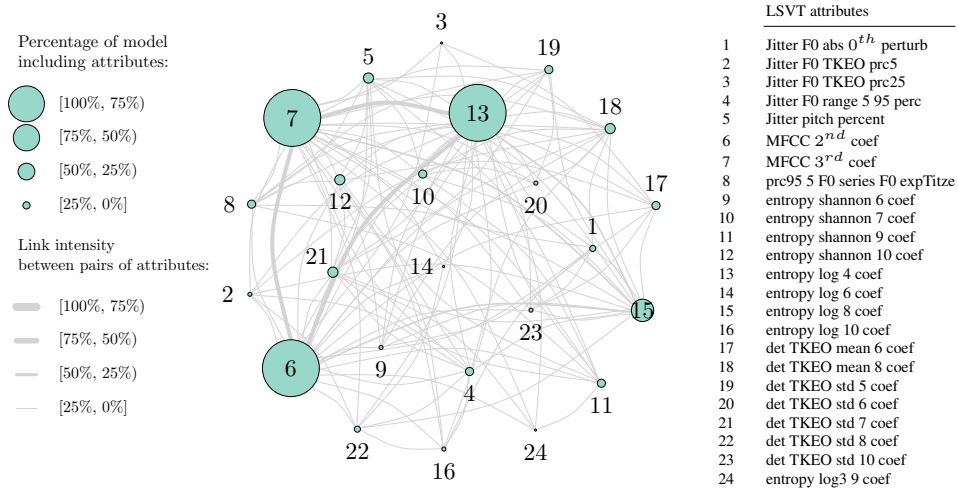

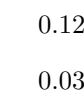
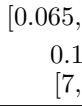



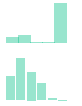
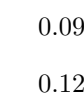
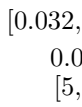

Figure 2: Network of attribute importance and pairwise link-intensity in the set of learners $\tilde{\mathcal{M}}$ for the LSVT dataset using the Radial-Kernel SVM SWAG results.

5.2.2 AHUS Gene Selection

The AHUS dataset was analyzed in [18] and collected 15739 miRNA expressions for 156 patients (women) who are either “healthy” or are diagnosed with “breast cancer”. Among others, the goal of analyzing this data is to identify expressions/biomarkers that we can target in order to more effectively diagnose and treat breast cancer. It also opens up further avenues of research regarding certain gene functions so as to better understand their possible relations to breast cancer. Again, we split the data into a training set using 80% of the original data while preserving the ratio of classes (i.e. 56 “healthy” and 69 “breast cancer” classes) and a test set (14 “healthy” and 17 “breast cancer” classes). Here, we specify $p_{\max} = 8$ based on the EPV rule which, using this parameter, was equal to 7. This value is at the upper acceptable bound in order to better explore the attribute space given the total dimension of 15739. As for the m parameter, exploring all learners of dimension 3 would require $m = 7 \cdot 10^5$ therefore requiring too much computational power. Therefore, we limit ourselves to $m = 4 \cdot 10^4$ in order to ensure the exploration of at least all learners of dimension 2. Given the large number of attributes ($p = 15739$), we decide to fix $\alpha = 0.01$. This choice allows to select a reasonable number of attributes in Algo. 1 and very strong learners in Algo. 2. We summarize the SWAG results and output of the original learning mechanisms in Tab. 4.

Table 4: Results for the AHUS dataset with the four considered learning mechanisms. For each learning mechanism \mathcal{L} , we have the range of training errors for SWAG learners and the single error for the original mechanism (first row) as well as those of the testing errors (second row). For each row, we plot the distribution of these errors for SWAG learners (third column for each mechanism). The third row for each mechanism shows the majority-rule procedure (MR) applied to SWAG learners (no available results for the original version since there is only a single learner). The last row collects the number of attributes (dimension) included in the learners.

	Lasso-logic			Linear-Kernel SVM		
	SWAG	Original		SWAG	Original	
ϵ_{train}	[0.038, 0.054]	0.136		[0.031, 0.049]	0.121	
ϵ_{test}	[0.065, 0.226]	0.065		[0.065, 0.194]	0.032	
MR	0.129	-		0.129	-	
$ s_{\mathcal{L}} $	[6, 8]	17		[7, 8]	15739	

	Radial-Kernel SVM			Random Forest		
	SWAG	Original		SWAG	Original	
ϵ_{train}	[0.031, 0.039]	0.111		[0.031, 0.040]	0.096	
ϵ_{test}	[0.032, 0.194]	0.194		[0.032, 0.194]	0.129	
MR	0.032	-		0.097	-	
$ s_{\mathcal{L}} $	[5, 8]	15739		[5, 8]	15739	

In a similar way to the LSVT data in Sec. 5.2.1, SWAG learners greatly outperform their original versions in terms of training error but have varying performance with regards to test error. In the worst case, SWAG learners either perform worse (Linear-Kernel SVM) or at most as well (Lasso) as the original versions. Again, we stress that this performance-gap is not large even if SWAG exploits an extremely lower subset of attributes than the original versions. Besides, SWAG learner test errors include that of the original Random Forest, indicating that we can obtain an improved performance with as few as 5 attributes instead of 15739, whereas all SWAG test errors are lower or equal to the original version of the Radial-Kernel SVM. We preserve the advantage of the extremely lower attribute dimension without sacrificing prediction accuracy.

Especially in this dataset, the two key features “interpretability” and “replicability” of SWAG are of the utmost importance. Indeed, we can both construct informative low-dimensional networks and perform accurate diagnoses when collecting different gene expressions as highlighted in the preliminary work in [32].

5.3 R Package

We make SWAG available as an R package at <https://github.com/SMAC-Group/SWAG-R-Package>. The name of the package is `swag`.

References

- [1] L. Breiman. Bagging predictors. Machine Learning, 24(2):123–140, 1996.
- [2] L. Breiman. Random forests. Machine Learning, 45(1):5–32, 2001.
- [3] J. M. Carmona-Cejudo, M. Baena-García, J. Campo-Avila, R. Morales-Bueno, J. Gama, and A. Bifet. Using gnu-mail to compare data stream mining methods for on-line email classification. In Proceedings of the Second Workshop on Applications of Pattern Analysis, pages 12–18, 2011.
- [4] R. Caruana and D. Freitag. Greedy attribute selection. In Machine Learning Proceedings 1994, pages 28–36. Elsevier, 1994.
- [5] R. Caruana, A. Niculescu-Mizil, G. Crew, and A. Ksikes. Ensemble selection from libraries of models. In Proceedings of the 21st International Conference on Machine Learning, page 18, 2004.
- [6] R. B. Cattell. The scree test for the number of factors. Multivariate Behavioral Research, 1(2): 245–276, 1966.
- [7] G. Chandrashekar and F. Sahin. A survey on feature selection methods. Computers & Electrical Engineering, 40(1):16–28, 2014.
- [8] C. Cortes and V. Vapnik. Support vector machine. Machine Learning, 20(3):273–297, 1995.
- [9] S. F. Crone. Training artificial neural networks for time series prediction using asymmetric cost functions. In Proceedings of the 9th International Conference on Neural Information Processing, 2002. ICONIP'02., volume 5, pages 2374–2380. IEEE, 2002.
- [10] S. Das. Filters, wrappers and a boosting-based hybrid for feature selection. In Icml, volume 1, pages 74–81, 2001.
- [11] S. Draghici, P. Khatri, A. C. Eklund, and Z. Szallasi. Reliability and reproducibility issues in dna microarray measurements. TRENDS in Genetics, 22(2):101–109, 2006.
- [12] D. Dua and C. Graff. UCI machine learning repository, 2017. URL <http://archive.ics.uci.edu/ml>.
- [13] J. Friedman, T. Hastie, and R. Tibshirani. Regularization paths for generalized linear models via coordinate descent. Journal of Statistical Software, 33(1):1, 2010.
- [14] R. Genuer, J.-M. Poggi, and C. Tuleau-Malot. Variable selection using random forests. Pattern Recognition Letters, 31(14):2225–2236, 2010.
- [15] S. Guerrier, N. Mili, R. Molinari, S. Orso, M. Avella-Medina, and Y. Ma. A predictive based regression algorithm for gene network selection. Frontiers in Genetics, 7:97, 2016.
- [16] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. Machine Learning, 46(1-3):389–422, 2002.
- [17] K. S. Gyamfi, J. Brusey, A. Hunt, and E. Gaura. Linear dimensionality reduction for classification via a sequential bayes error minimisation with an application to flow meter diagnostics. Expert Systems with Applications, 91:252–262, 2018.
- [18] V. D. Haakensen, V. Nygaard, L. Greger, M. R. Aure, B. Fromm, I. R.K. Bukholm, T. Lüders, S.-F. Chin, A. Git, C. Caldas, et al. Subtype-specific micro-rna expression signatures in breast cancer progression. International Journal of Cancer, 139(5):1117–1128, 2016.
- [19] T. Hastie, R. Tibshirani, and M. Wainwright. Statistical learning with sparsity: the lasso and generalizations. CRC press, 2015.
- [20] A. E. Hoerl and R. W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. Technometrics, 12(1):55–67, 1970.

- [21] H. Ishwaran. Variable importance in binary regression trees and forests. Electronic Journal of Statistics, 1:519–537, 2007.
- [22] R. Kohavi and G. H. John. Wrappers for feature subset selection. Artificial Intelligence, 97(1-2):273–324, 1997.
- [23] N. Kolesnikov, E. Hastings, M. Keays, O. Melnichuk, Y. A. Tang, E. Williams, M. Dylag, N. Kurbatova, M. Brandizi, and T. Burdett. Arrayexpress update—simplifying data submissions. Nucleic Acids Research, 43(D1):D1113–D1116, 2015.
- [24] M. Kuhn. Building predictive models in r using the caret package. Journal of Statistical Software, 28(5):1–26, 2008.
- [25] H. Liu and X. Chen. Nonparametric greedy algorithms for the sparse learning problem. In Advances in Neural Information Processing Systems, pages 1141–1149, 2009.
- [26] G. Louppe, L. Wehenkel, A. Suter, and P. Geurts. Understanding variable importances in forests of randomized trees. In Advances in Neural Information Processing Systems, pages 431–439, 2013.
- [27] U. M. Marigorta, J. A. Rodríguez, G. Gibson, and A. Navarro. Replicability and prediction: lessons and challenges from gwas. Trends in Genetics, 34(7):504–517, 2018.
- [28] H. Masnadi-Shirazi and N. Vasconcelos. Asymmetric boosting. In Proceedings of the 24th International Conference on Machine Learning, pages 609–619, 2007.
- [29] N. Meinshausen. Relaxed lasso. Computational Statistics & Data Analysis, 52(1):374–393, 2007.
- [30] N. Meinshausen and P. Bühlmann. Stability selection. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 72(4):417–473, 2010.
- [31] N. Meinshausen and B. Yu. Lasso-type recovery of sparse representations for high-dimensional data. The Annals of Statistics, 37(1):246–270, 2009.
- [32] C. Miglioli, G. Bakalli, S. Guerrier, S. Orso, R. Molinari, and N. Mili. Non-coding chameleon micro-rnas in breast cancer: the elusive function of genomic variables with high predictive power. Working Paper, 2020.
- [33] C. Nadeau and Y. Bengio. Inference for the generalization error. In Advances in Neural Information Processing Systems, pages 307–313, 2000.
- [34] A. E. Raftery, D. Madigan, and J. A. Hoeting. Bayesian model averaging for linear regression models. Journal of the American Statistical Association, 92(437):179–191, 1997.
- [35] R. E. Schapire. The strength of weak learnability. Machine Learning, 5(2):197–227, 1990.
- [36] A. Tewari, P. K. Ravikumar, and I. S. Dhillon. Greedy algorithms for structurally constrained high dimensional problems. In Advances in Neural Information Processing Systems, pages 882–890, 2011.
- [37] R. Tibshirani. Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 58(1):267–288, 1996.
- [38] K. M. Ting. A comparative study of cost-sensitive boosting algorithms. In Proceedings of the 17th International Conference on Machine Learning. Citeseer, 2000.
- [39] Athanasios Tsanas, Max A Little, Cynthia Fox, and Lorraine O Ramig. Objective automatic assessment of rehabilitative speech treatment in parkinson’s disease. IEEE Transactions on Neural Systems and Rehabilitation Engineering, 22(1):181–190, 2013.
- [40] TUV-NEL. Testing the diagnostic capabilities of liquid ultrasonic flow meters. National Measurement System, 2012.

- [41] V. Vapnik. The Nature of Statistical Learning Theory. Springer science & business media, 2013.
- [42] D. Vats and R. Baraniuk. When in doubt, swap: High-dimensional sparse recovery from correlated measurements. In Advances in Neural Information Processing Systems, pages 989–997, 2013.
- [43] L. Vaughan and Y. Chen. Data mining from web search queries: A comparison of google trends and baidu index. Journal of the Association for Information Science and Technology, 66(1): 13–22, 2015.
- [44] E. Vittinghoff and C. E. McCulloch. Relaxing the rule of ten events per variable in logistic and cox regression. American Journal of Epidemiology, 165(6):710–718, 2007.
- [45] G. Wang, W. Li, M. A Zuluaga, R. Pratt, P. A. Patel, M. Aertsen, T. Doel, A. L. David, J. Deprest, S. Ourselin, et al. Interactive medical image segmentation using deep learning with image-specific fine tuning. IEEE Transactions on Medical Imaging, 37(7):1562–1573, 2018.
- [46] M. J. Whittingham, Philip A. Stephens, R. B. Bradbury, and R. P. Freckleton. Why do we still use stepwise modelling in ecology and behaviour? Journal of Animal Ecology, 75(5): 1182–1189, 2006.
- [47] M. Xiong, X. Fang, and J. Zhao. Biomarker identification by feature wrappers. Genome Research, 11(11):1878–1887, 2001.
- [48] T. Zhang. Adaptive forward-backward greedy algorithm for sparse learning with linear models. In Advances in Neural Information Processing Systems, pages 1921–1928, 2009.
- [49] Z. Zhang, Y. Xu, J. Yang, X. Li, and D. Zhang. A survey of sparse representation: algorithms and applications. IEEE Access, 3:490–530, 2015.
- [50] H. Zou and T. Hastie. Regularization and variable selection via the elastic net. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 67(2):301–320, 2005.