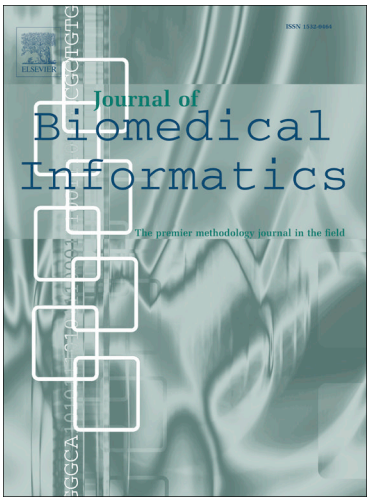

Relief-Based Feature Selection: Introduction and Review

Ryan J. Urbanowicz, Melissa Meeker, William La Cava, Randal S. Olson, Jason H. Moore

PII: S1532-0464(18)30140-0
DOI: https://doi.org/10.1016/j.jbi.2018.07.014
Reference: YJBIN 3019

To appear in: *Journal of Biomedical Informatics*

Accepted Date: 14 July 2018

Please cite this article as: Urbanowicz, R.J., Meeker, M., Cava, W.L., Olson, R.S., Moore, J.H., Relief-Based Feature Selection: Introduction and Review, *Journal of Biomedical Informatics* (2018), doi: https://doi.org/10.1016/j.jbi.2018.07.014

Relief-Based Feature Selection: Introduction and Review

Ryan J. Urbanowicz[a,*], Melissa Meeker[b], William La Cava[a], Randal S. Olson[a], Jason H. Moore[a]

[a]*Institute for Biomedical Informatics, University of Pennsylvania, Philadelphia, PA 19104, USA*
[b]*Ursinus College, Collegeville, PA, 19426, USA*

Abstract

Feature selection plays a critical role in biomedical data mining, driven by increasing feature dimensionality in target problems and growing interest in advanced but computationally expensive methodologies able to model complex associations. Specifically, there is a need for feature selection methods that are computationally efficient, yet sensitive to complex patterns of association, e.g. interactions, so that informative features are not mistakenly eliminated prior to downstream modeling. This paper focuses on Relief-based algorithms (RBAs), a unique family of filter-style feature selection algorithms that have gained appeal by striking an effective balance between these objectives while flexibly adapting to various data characteristics, e.g. classification vs. regression. First, this work broadly examines types of feature selection and defines RBAs within that context. Next, we introduce the original Relief algorithm and associated concepts, emphasizing the intuition behind how it works, how feature weights generated by the algorithm can be interpreted, and why it is sensitive to feature interactions without evaluating combinations of features. Lastly, we include an expansive review of RBA methodological research beyond Relief and its popular descendant, ReliefF. In particular, we characterize branches of RBA research, and provide comparative summaries of RBA algorithms including contributions, strategies, functionality, time complexity, adaptation to key data characteristics, and software availability.

Keywords: Feature Selection, Feature Interaction, Feature Weighting, Filter, ReliefF, Epistasis

*Corresponding Author
Email addresses: ryanurb@upenn.edu (Ryan J. Urbanowicz), memeeker@ursinus.edu (Melissa

1. Background

The fundamental challenge of almost any data mining or modeling task is to identify and characterize relationships between one or more features in the data (also known as predictors or attributes) and some endpoint (also known as the dependent variable, class, outcome, phenotype, or concept). In most datasets, only a subset of available features are *relevant features*, i.e. informative in determining the endpoint value. The remaining *irrelevant features*, which are rarely distinguishable *a priori* in real world problems, are not informative yet contribute to the overall dimensionality of the problem space. This increases the difficulty and computational burden placed on modeling methods. *Feature selection* could generically be defined as the process of identifying relevant features and discarding irrelevant ones.

Figure 1 illustrates the typical stages of a data mining analysis pipeline. Specifically, raw data is preprocessed in preparation for analysis. This typically includes some type of cross validation where the data is split into training, validation, and testing subsets to avoid overfitting and assess the generalizability of the final model. Next, different feature processing approaches can be employed to remove irrelevant features or construct better relevant ones. Modeling then takes place on this preprocessed data. Model performance could then feed back into another round of feature processing (dotted line). This is the case for *wrapper* feature selection methods, reviewed below. The final model is ultimately assessed and interpreted in a post analysis stage that ideally leads to the discovery of useful knowledge. Feature selection is an important part of a successful data mining pipeline, particularly in problems with very large feature spaces. Poorly performed feature selection can have significant downstream consequences on data mining, particularly when relevant features have been mistaken as irrelevant and removed from consideration.

Meeker), lacava@upenn.edu (William La Cava), olsonran@upenn.edu (Randal S. Olson), jhmoore@upenn.edu (Jason H. Moore)

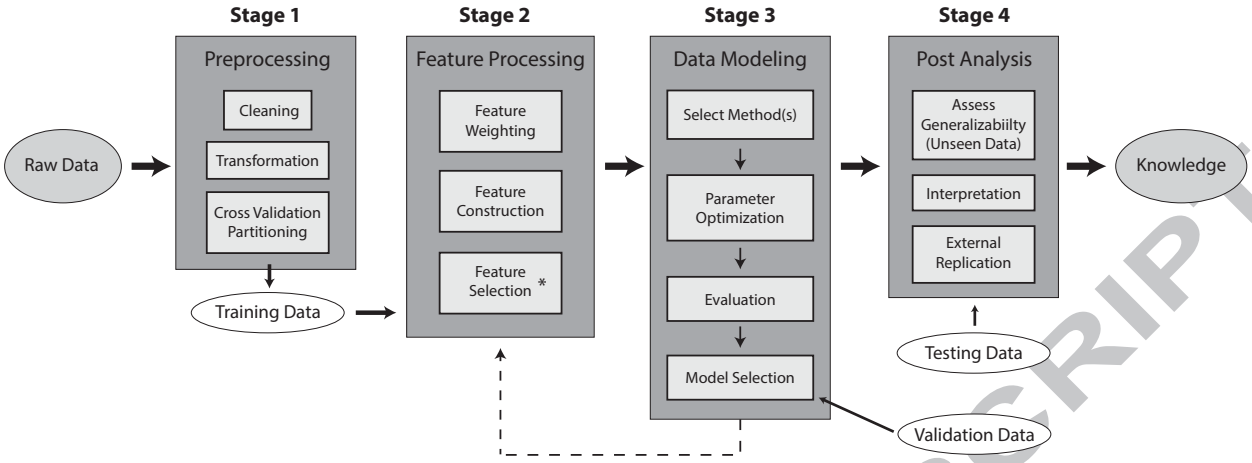

Figure 1: Typical stages of a data mining analysis pipeline. Feature selection is starred as it is the focus of this review. The dotted line indicates how model performance can be fed back into feature processing, iteratively removing irrelevant features or seeking to construct relevant ones.

1.1. Types of Feature Selection

A large variety of feature selection methodologies have been proposed and research continues to support the claim that there is no universal "best" method for all tasks [12]. In order to navigate methodological options and assist in selecting a suitable method for a given task it is useful to start by characterizing and categorizing different feature selection methods [23, 79, 12, 51]. One such characterization is with regards to the feature selection objective.

1. *Idealized*: find the minimally sized feature subset that is necessary and sufficient to describe the target concept [53].
2. *Target Feature Count*: select a subset of m features from a total set of n features, $m < n$, such that the value of a criterion function is optimized over all subsets of size m [78].
3. *Prediction Accuracy Improvement*: choose a subset of features that best increases prediction accuracy or decreases model complexity without significantly decreasing the prediction accuracy [48].
4. *Approximate Original Class Prediction Probability Distribution*: for classification prob-

lems, select a feature subset that yields a class prediction probability distribution that is as close as possible to the class prediction probability distribution given all features. In contrast with prediction accuracy this perspective seeks to preserve additional information regarding probabilities of class predictions [56].

5. *Rank and Define Cutoff*: first rank all features using some surrogate measure of feature 'value', then define the feature subset by applying an ad-hoc cutoff. This cutoff may be determined by statistical or subjective likelihood of relevance or simply a desired number of features in the subset [52].

This list is an updated version of one regularly used in the literature [23, 75, 107, 36]. Alternatively, feature selection methods can be distinguished based on their relationship with the construction of the model (i.e. induction) [95, 12, 18, 107, 51, 75].

1. *Filter Methods*: use a 'proxy measure' calculated from the general characteristics of the training data to score features or feature subsets as a processing step prior to modeling. Filters are generally much faster and function independently of the induction algorithm, meaning that selected features can then be passed to any modeling algorithm. Filter methods can be roughly classified further by the filtering measures they employ, i.e. information, distance, dependence, consistency, similarity, and statistical measures [23, 12, 51]. Examples include information gain [46], chi-square [50], and Relief [53].
2. *Wrapper Methods*: employ any stand-alone modeling algorithm to train a predictive model using a candidate feature subset. The testing performance on a hold-out set is typically used to score the feature set. Alternatively in a modeling algorithm like a random forest, estimated feature importance scores can be applied to select a feature subset [72]. In any wrapper method, a new model must be trained to test any subsequent feature subset, therefore wrapper methods are typically iterative and computationally intensive, but can identify the best performing features set for that specific modeling algorithm [42, 12, 51]. Each iteration of the wrapper, the feature subset is generated based on the selected search strategy, e.g. forward or backward selection

[54, 63] or a heuristic feature subset selection [114, 44]. Examples include wrappers for Naïve Bayes [22], Support Vector Machines (SVM) [13], and most any modeling algorithm combined with a feature subset generation approach. Thus a wrapper method is defined by both the selected induction algorithm as well as the feature subset search strategy. However, due to the computational complexity of wrappers, only the simplest modeling methods can be used efficiently.

3. *Embedded Methods*: perform feature selection as a part of the modeling algorithm's execution. These methods tend to be more computationally efficient than wrappers because they simultaneously integrate modeling with feature selection. This can be done, for instance, by optimizing a two-part objective function with (1) a goodness-of-fit term and (2) a penalty for a larger number of features. As with wrappers, the features selected by embedded methods are induction algorithm dependent [42, 12, 51]. Examples include Lasso [108], Elastic Net [125], and various decision tree based algorithms, e.g. CART [14], C4.5 [85], and most recently, XGBoost [20].

Many hybrid methods have also been proposed that seek to combine the advantages of wrappers and filters [51].

Lastly, feature selection approaches have also been broadly categorized as relying on either *individual evaluation* or *subset evaluation* [119, 12]. Individual evaluation, (i.e. feature weighting/ranking) assesses individual features and assigns them weights/scores according to their degrees of relevance [11, 119]. Subset evaluation instead assesses candidate feature subsets that are selected based on a given search strategy [12]. Filter, wrapper, or embedded methods can be either subset or individual evaluation methods.

The remainder of this paper will focus on the family of *Relief-based* feature selection methods referred to here as Relief-Based Algorithms (RBAs) that can be characterized as *individual evaluation filter methods*. For reviews of features selection methods in general, we refer readers to [63, 23, 42, 5, 12, 107, 18, 51, 75].

1.2. Why Focus on Relief-based Feature Selection?

One advantage of certain wrapper or embedded methods is that by relying on subset evaluation they have the potential to capture feature dependencies in predicting the endpoint, i.e. interactions [12]. In contrast, very few filter methods, besides for example, FOCUS [3] and INTERACT [123] that are notably subset evaluation filters, claim to be able to handle feature interactions. The most reliable but naive approach for identifying feature interactions is to exhaustively search over all subsets of the given feature set, e.g. FOCUS. This quickly becomes computationally intractable in problems with larger feature spaces. The inefficiency of these approaches is that they must explicitly search through combinations of features. Alternatively, the Relief algorithm and its derivatives are, to the best of our knowledge, the only individual evaluation filter algorithms capable of detecting feature dependencies. These algorithms do not search through feature combinations, but rather use the concept of nearest neighbors to derive feature statistics that indirectly account for interactions. Furthermore, RBAs retain the generalized advantages of filter algorithms, i.e. they are relatively fast (with an asymptotic time complexity of $\mathcal{O}(instances^2 \cdot features)$), and the selected features are not induction algorithm dependent. The ability to confidently utilize selected features with different induction algorithms may save further downstream computational effort when applying more than a single modeling technique. This is important in the context of (1) the 'no-free lunch' theorem proposed by Wolpert and Macready [116] that suggests that no single modeling algorithm can be optimal for all problems, and (2) the widely acknowledged value of ensemble methods reviewed by Rokach [93] that combine input from multiple statistical or machine learning induction methods to make the best informed predictions. Lastly, individual evaluation approaches, including RBAs, offer a greater flexibility of use. Specifically, individual feature weights may be applied not only to select 'top' features, but can also be applied as expert knowledge to guide stochastic machine learning algorithms such as evolutionary algorithms [111]. Furthermore, when selecting features, feature sets of different sizes can be selected based on whatever criteria is desired for feature inclusion from a ranked feature list.

The following two subsections address important considerations related to our assertion

that RBAs deserve particular attention. A closer look at the strengths and weaknesses of the Relief algorithm is given in Section 2.1.1.

1.2.1. Feature Construction

An alternative or supplemental approach to facilitate the detection and modeling of interactions is to apply feature construction (see Figure 1), also known as constructive induction or feature extraction. Feature construction methods, e.g. principle component analysis or linear discriminant analysis [68], define new features as a function of two or more other features [74]. This subset of constructed features can be added to the original feature space, or analyzed in its place (achieving dimensionality reduction). A common side effect of most any feature construction method is that the original features are no longer recognizable, leading to challenges in downstream model interpretability.

One feature construction method geared specifically towards capturing feature interactions is multifactor dimensionality reduction (MDR) [87]. Another more general example is polynomial feature construction that is able to detect multiplicative interactions [106]. These approaches attempt to combine individual features that may be interacting and construct a single feature that can be more easily identified as relevant using any simple feature selection or induction method. There are many possible feature construction approaches to chose from and some can be quite computationally expensive. Notably, applying feature construction does not necessarily preclude the need for feature selection. Thus, assuming that a feature selection and modeling approach has been chosen that is sensitive to a target interaction dimensionality (e.g. 2-way or 3-way), it may be most efficient to skip feature construction, particularly if downstream model interpretation is critical. While feature construction certainly has its own utility, further discussion is outside the scope of this review.

1.2.2. Redundancy

Relevant features can be more restrictively defined as any feature that is neither irrelevant nor redundant to the target concept [56, 23]. Feature redundancy is explored further by Yu and Liu [119]. Some feature selection methods seek to remove redundant features while others do not. Caution should be used when removing presumably redundant features,

because unless two features are perfectly correlated (i.e. truly redundant) there may still be information to be gained from including them both [42]. One repeatedly noted drawback of RBAs is that they do not remove feature redundancies, i.e. they seek to select all features relevant to the endpoint regardless of whether some features are strongly correlated with others [53, 5, 35]. However, except for features that are perfectly correlated, it is not always clear whether useful information is being lost when 'redundant' features are removed. For example, it has been suggested that preserving redundant features can be a benefit, as it "may point to meaningful clusters of correlated phenotypes" [109]. If removing redundancy is clearly important to success in a given problem domain, many effective methods are available that can be applied before, after, or integrated with RBA feature selection to remove feature redundancies [8, 35, 117, 102, 17, 1, 67, 40].

1.3. Paper Summary

In the text that follows, we (1) introduce RBAs from the perspective of the original Relief algorithm noting key concepts and intuitions, (2) examine the contributions of the landmark ReliefF algorithm, (3) differentiate thematically distinct branches of RBA research, (4) review methodological expansions and advancements introduced by derivative members of the RBA family in the wake of Relief and ReliefF, (5) consider RBA evaluations, and (6) summarize software availability. This review was prepared to complement a comprehensive research comparison of 'core' RBAs presented by Urbanowicz et al. [113].

2. Introduction to Relief

In this section we provide algorithmic and conceptual descriptions of the original Relief algorithm relevant to understanding all members of the RBA family.

2.1. Relief

Kira and Rendell [53, 52] formulated the original Relief algorithm inspired by instance-based learning [2, 16]. As an individual evaluation filtering feature selection method, Relief calculates a proxy statistic for each feature that can be used to estimate feature 'quality' or

'relevance' to the target concept (i.e. predicting endpoint value). These feature statistics are referred to as feature weights ($W[A]$ = weight of feature 'A'), or more casually as feature 'scores' that can range from -1 (worst) to $+1$ (best). Notably, the original Relief algorithm was limited to binary classification problems, and had no mechanism to handle missing data. Strategies to extend Relief to multi-class or continuous endpoint problems are not detailed here, but are described in the respective works cited in the review section of this paper.

Algorithm 1 Pseudo-code for the original Relief algorithm

Require: for each training instance a vector of feature values and the class value
  $n \leftarrow$ number of training instances
  $a \leftarrow$ number of features (i.e. attributes)
  Parameter: $m \leftarrow$ number of random training instances out of n used to update W

  initialize all feature weights $W[A] := 0.0$
  for i:=1 **to** m **do**
    randomly select a 'target' instance R_i
    find a nearest hit 'H' and nearest miss 'M' (instances)
    for A:= 1 **to** a **do**
      $W[A] := W[A] - \mathit{diff}(A, R_i, H)/m + \mathit{diff}(A, R_i, M)/m$
    end for
  end for
  return the vector W of feature scores that estimate the quality of features

As summarized by the pseudo-code in Algorithm 1, the Relief algorithm cycles through m random training instances (R_i), selected without replacement, where m is a user-defined parameter. Each cycle, R_i is the 'target' instance and the feature score vector W is updated based on feature value differences observed between the target and neighboring instances. Therefore each cycle, the distance between the 'target' instance and all other instances is calculated. Relief identifies two nearest neighbor instances of the target; one with the same class, called the *nearest hit* (H) and the other with the opposite class, called the *nearest miss* (M). The last step of the cycle updates the weight of a feature A in W if the feature value differs between the target instance R_i and either the nearest hit H or the nearest miss M (see Figure 2). Features that have a different value between R_i and M support the hypothesis that they are informative of outcome, so the quality estimation $W[A]$ is increased.

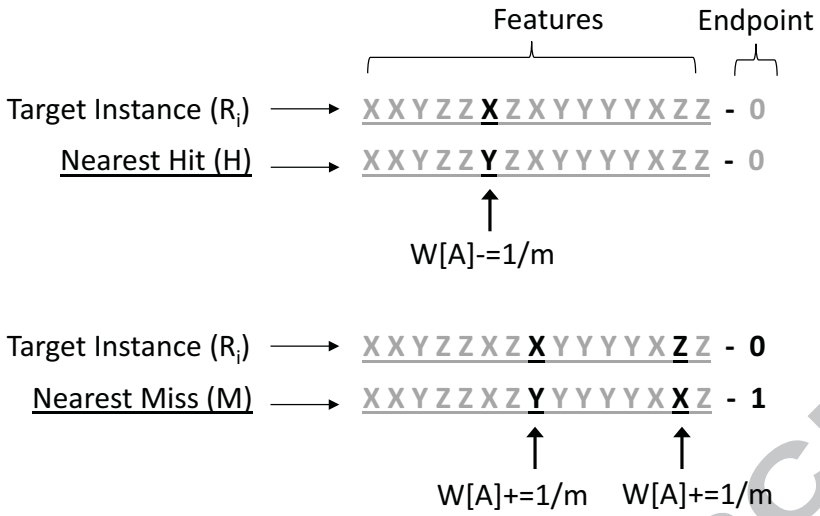

Figure 2: Relief updating $W[A]$ for a given target instance when it is compared to its nearest miss and hit. In this example, features are discrete with possible values of X, Y, or Z, and endpoint is binary with a value of 0 or 1. Notice that when the value of a feature is different, the corresponding feature weight increases by $1/m$ for the nearest miss, and reduces by $1/m$ for the nearest hit.

Conversely, features with differences between R_i and H provide evidence to the contrary, so the quality estimation $W[A]$ is decreased. The *diff* function in Algorithm 1 calculates the difference in value of feature A between two instances I_1 and I_2, where $I_1 = R_i$ and I_2 is either H or M, when performing weight updates [90]. For discrete (e.g. categorical or nominal) features, *diff* is defined as:

$$diff(A, I_1, I_2) = \begin{cases} 0 & \text{if } value(A, I_1) = value(A, I_2) \\ 1 & \text{if } otherwise \end{cases} \tag{1}$$

and for continuous (e.g. ordinal or numerical) features, *diff* is defined as:

$$diff(A, I_1, I_2) = \frac{|value(A, I_1) - value(A, I_2)|}{max(A) - min(A)} \tag{2}$$

The maximum and minimum values of A are determined over the entire set of instances. This normalization ensures that weight updates fall between 0 and 1 for both discrete and continuous features. Additionally, in updating $W[A]$, dividing the output of *diff* by m guarantees that all final weights will be normalized within the interval $[-1, 1]$.

The *diff* function is also used to calculate the distance between instances when finding nearest neighbors. The total distance is simply the sum of *diff* distances over all attributes (i.e. Manhattan distance). Technically, the original Relief algorithm used Euclidian distance rather than Manhattan distance i.e. the *diff* terms were squared during instance distance measurements and feature weighting. However, experiments by [62] indicated no significant difference between results using *diff* or squared *diff*, thus the simplified description of the Relief algorithm has become standard. It has also been suggested that any valid distance metric could be used by Relief [109]. Thus, determining the best distance metric remains an open research question. While the above *diff* function performs well when features are either uniformly discrete or continuous, it has been noted that given a dataset with a mix of discrete and continuous features, this *diff* function can underestimate the quality of the continuous features [61]. One proposed solution to this problem is a *ramp function* that naively assigns a full *diff* of 0 or 1 if continuous feature values are some user defined minimum or maximum value apart from one another, respectively, and a function of the distance from these boundaries otherwise [45, 91, 61]. However since this approach adds two additional user-defined parameters requiring problem dependent optimization, it may be challenging to apply in practice.

2.1.1. Strengths and Limitations

Regarding strengths, Relief has been presented as being both non-myopic [61], i.e. it estimates the quality of a given feature in the context of other features, and non-parametric [109], i.e. it makes no assumptions regarding the population distribution or sample size. The efficiency of the algorithm has been attributed to the fact that it doesn't explicitly explore feature subsets and because it does not bother trying to identify an optimal minimum feature subset size [53]. Instead, Relief was originally "intended as a *screener* to identify a subset of features that may not be the smallest and may still include some irrelevant and redundant features, but that is small enough to use with more refined approaches in a detailed anlaysis" [109]. Consider that an exhaustive search for interactions between all feature pairs alone would have a time complexity of $\mathcal{O}(2^a)$, while Relief boasts a time complexity of $\mathcal{O}(a \cdot m \cdot n)$,

or $\mathcal{O}(a \cdot n)$ whenever $m < n$. Furthermore, it has been suggested that Relief could be viewed as an *anytime algorithm*, i.e. one that can be stopped and yield results at any time, but it is presumed that with more time or data it will improve the results [91].

Regarding limitations, the original Relief analysis suggests that the algorithm can be fooled by insufficient training cycles (i.e. not a large enough m). The original paper also suggests that Relief is fairly noise-tolerant and unaffected by feature interactions. However, later work identified that Relief was susceptible to noise interfering with the selection of nearest neighbors [58]. Further, research into RBAs has, until recently, been limited to considering 2-way feature interactions only. Therefore, it was unclear if RBAs could detect feature interactions with a dimensionality beyond 2 features. Research paired with this review suggests that only specific RBAs have the ability to detect higher order interactions (e.g. 3-way, 4-way, and 5-way), thus RBAs are only universally reliable in detecting 2-way interactions [113]. Relief has also been noted to have a reduced power to identify relevant non-monotonic features (e.g. features with a Gaussian distribution) [9]. Most importantly, it has been repeatedly demonstrated empirically and theoretically that core RBA performance deteriorates as the number of irrelevant features becomes 'large' [91, 77, 33, 109]. This deterioration of performance in identifying interacting features is primarily due to the fact that Relief's computation of neighbors and weights becomes increasingly random as the number of features increases. This is an example of the curse of dimensionality. The iterative RBAs reviewed in Section 3.4 have been demonstrated to improve RBA performance in these types of large feature spaces. Differently, deteriorating performance in detecting main effects in very large feature spaces is primarily due to feature scores being based on feature value differences between a subset of neighboring instances, rather than differences from all instances. Thus the main effect signal in RBA scores is not expected to be as pronounced. Given a very large feature space, this less pronounced main effect feature score is less likely to stand out. In general, it is likely that myopic feature selection algorithms that compute scores by comparing the feature values of all training instances will have the most power to detect simple main effects [71]. In addition to an iterative RBA approach, another way to address lost main effect performance could involve running both an RBA and a myopic

feature selection algorithm, selecting the top non-redundant set of features combined from both algorithms [115]. Unfortunately, as of yet, there are no clear guidelines regarding the size of the feature space where (1) myopic methods would be expected to outperform RBAs in detecting main effects, or (2) interactions effects can't be distinguished from random background noise. Simulations studies such as those reviewed in Section 3.6 offer some insight, however in real-world applications many factors are expected impact success (e.g. number of training instances, type of signal, signal strength, feature distributions, feature type, etc.).

Lastly, it's notable that Relief scores do not reflect the nature of an association. For example Relief does not tell you which attributes might be interaction partners, or whether a score is high due to a linear effect or an interaction. This is left to downstream modeling. Furthermore, there is no established way to assess how many of the high scoring selected features may be false discoveries. It is possible this issue could be addressed through permutation testing as suggested by McKinney et al. [71].

2.2. Feature Subset Selection

The original description of Relief specified an automated strategy for feature subset selection [53]. Specifically, a *relevance threshold* (τ) was defined such that any feature with a relevance weight $W[A] \geq \tau$ would be selected. Kira and Rendell demonstrated that "statistically, the relevance level of a relevant feature is expected to be larger than zero and that of an irrelevant one is expected to be zero (or negative)". Therefore generally the threshold should be selected such that $0 < \tau < 1$. More precisely they proposed the bounds i.e. $0 < \tau \leq \frac{1}{\sqrt{\alpha m}}$, based on Chebyshev's inequality, where α is the probability of accepting an irrelevant feature as relevant (i.e. making a Type I error). If τ is set too high, there is an increased chance that one or more relevant features will fail to be selected. Alternatively if τ is set too low, it is expected that an increased number of irrelevant features will be selected. Like any significance threshold the choice of τ is somewhat arbitrary. Not all features with a weight above the selected threshold will necessarily be relevant because it is expected that some irrelevant features will have a positive weight by chance.

In practice, rather than choosing a value of τ, it is often more practical to choose some number of features to be selected *a priori* based on the functional, computational, or run time limitations of the downstream modeling algorithms that will be applied. Ultimately the goal is to provide the best chance that all relevant features are included in the selected set for modeling, but at the same time, remove as many of the irrelevant features as possible to facilitate modeling, reduce over-fitting, and make the task of induction tractable.

2.3. Intuition, Interpretation, and Interactions

Relief algorithms often appear simple at first glance, but understanding how to interpret feature weights and gaining the intuition as to how feature dependencies (i.e. interactions) can be gleaned without explicitly considering feature subsets is not always apparent. The key idea behind Relief is to estimate feature relevance according to how well feature values distinguish concept (endpoint) values among instances that are similar (near) to each other. Two complementary interpretations of Relief feature weights have been derived and presented: (1) a probabilistic interpretation [58, 60, 62, 90, 91] and (2) an interpretation as the portion of the explained concept changes [90, 91]. Next, we summarize these interpretations and why they explain Relief's ability to detect interactions.

2.3.1. Probabilistic Interpretation

The first interpretation is that the Relief weight estimate $W[A]$ of feature A is an approximation of the following difference of probabilities:

$$\begin{aligned} W[A] = {} & P(\text{different value of } A \mid \text{nearest instance from different class}) \\ & - P(\text{different value of } A \mid \text{nearest instance from same class}) \end{aligned} \tag{3}$$

Consider that as the number of nearest neighbors used in scoring increases from 1 and approaches n this effectively eliminates the condition that instances used in scoring be 'near'. Notably if we were to eliminate the 'near' requirement from 3, the formula becomes:

$$W[A] = P(\text{different value of } A \mid \text{different class}) - P(\text{different value of } A \mid \text{same class}) \quad (4)$$

As derived by Robnik-Šikonja and Kononenko [91] it can be shown that, without the near condition, Relief weights would be strongly correlated with impurity functions such as the Gini-index gain. Impurity functions including information gain [46], gain ratio [85], gini-index [14], distance measure [26], j-measure [97], and MDL [59] have often been used as myopic filter feature selection algorithms that assume features to be conditionally independent given the class.

Thus, it is the '*nearest instance*' condition in Equation 3 and the resulting fact that Relief weights are averaged over local estimates in smaller parts of the instance subspace (rather than globally over all instances) that enables Relief algorithms to take into account the context of other features and detect interactions [62, 61]. It has been demonstrated by Robnik-Šikonja and Kononenko [91], that as the number of neighbors used in scoring approaches n the ability of Relief to detect feature dependencies disappears, since scoring is no longer limited to 'near' instances.

2.3.2. Concept Change Interpretation

The second interpretation of Relief weights has been argued as being the more comprehensible/communicable than the probabilistic one [90]. The authors demonstrate that Relief relevance weights $W[A]$ can be interpreted as the ratio between the number of explained changes in the concept and the number of examined instances. If a particular change can be explained in multiple ways, all ways share credit for it in the quality estimate. Also if several features are involved in one way of the explanation, all of them get the credit in their quality estimate [91]. To illustrate this idea, Table 1 presents a simple Boolean problem where *Class* is determined by the expression $(A_1 \wedge A_2) \vee (A_1 \wedge A_3)$, such that all three features (A_1, A_2, and A_3) are relevant.

In the first instance of Table 1 it can be said that A_1 is responsible for class assignment

Table 1: Tabular dataset description of Boolean problem $Class = (A_1 \wedge A_2) \vee (A_1 \wedge A_3)$ including the responsibility of each feature for yielding an expected class change. Adapted from Robnik-Šikonja and Kononenko [91].

	Feature Values					Score Change		
Instances	A_1	A_2	A_3	$Class$	Responsible Features	A_1	A_2	A_3
R_1	1	1	1	1	A_1	1/8	0	0
R_2	1	1	0	1	A_1 or A_2	0.5/8	0.5/8	0
R_3	1	0	1	1	A_1 or A_3	0.5/8	0	0.5/8
R_4	1	0	0	0	A_2 or A_3	0	0.5/8	0.5/8
R_5	0	1	1	0	A_1	1/8	0	0
R_6	0	1	0	0	A_1	1/8	0	0
R_7	0	0	1	0	A_1	1/8	0	0
R_8	0	0	0	0	(A_1 and A_2) or (A_1 and A_3)	1/8	0.5/8	0.5/8
					Total	0.75	0.1875	0.1875

because changing its value would be the only feature value change necessary to make $Class = 0$. In the second instance, changing either A_1 or A_2 would make $Class = 0$, thus they share the responsibility. Similar responsibility assignments can be made for instances 3 to 7, while in instance 8, changing only one feature value isn't enough for $Class$ to change, however there are two pairs of feature value changes that can. As detailed by Robnik-Šikonja and Kononenko [91], adding up the responsibility for each feature results in a score estimate of 0.75 for A_1, and an estimate of 0.1875 for both A_2 and A_3. This result makes sense given that A_1 clearly has a stronger linear association with $Class$ (i.e. a main effect), but both A_2 and A_3 contribute to a lesser extent, interacting with A_1 in a subset of instances. This conceptual example was validated empirically by Robnik-Šikonja and Kononenko [91], finding in a dataset with these three relevant features along with five random binary features, that the Relief relevance estimate for A_1 converges near 0.75, and estimates for A_2 and A_3 converge near 0.1875 with an increasing number of training instances.

2.3.3. Breaking Down Interaction Detection

To further clarify how Relief detects interactions, Table 2 offers a simple example dataset that we will use to walk through Relief scoring. In this example, A_1 and A_2 are relevant features. When they have different values, the $Class = 1$ otherwise the $Class = 0$. This

is an example of a 'pure' interaction, where no individual feature has an association with endpoint. A_3 is an irrelevant feature.

Table 2: Example dataset with interaction between A_1 and A_2. A_3 is irrelevant. Adapted from [62].

Instances	A_1	A_2	A_3	$Class$
R_1	1	0	1	1
R_2	1	0	0	1
R_3	0	1	1	1
R_4	0	1	0	1
R_5	0	0	1	0
R_6	0	0	0	0
R_7	1	1	1	0
R_8	1	1	0	0

Table 3 breaks down how scoring would proceed over 8 cycles with each instance getting to be the respective target. For each target, we see what instance is the nearest hit and miss, as well as what feature has a different value between the instances (given in parentheses), and thus is relevant to scoring. If there is a tie for nearest neighbor, both instances are listed with their respective different valued feature. For example when R_1 is the target, its nearest hit is R_2. The only feature with a different value between these two instances is A_3. The nearest miss for R_1 is a tie between R_5 and R_7 that have feature value differences at A_1 and A_2, respectively.

Table 3: Breakdown of Relief nearest neighbors (i.e. hits and misses) and corresponding feature value differences given in parentheses when a given instance from Table 2 is the target.

	Nearest	
Target	Hit	Miss
R_1	$R_2(A_3)$	$R_5(A_1)$,$R_7(A_2)$
R_2	$R_1(A_3)$	$R_6(A_1)$,$R_8(A_2)$
R_3	$R_4(A_3)$	$R_5(A_2)$,$R_7(A_1)$
R_4	$R_3(A_3)$	$R_6(A_2)$,$R_8(A_1)$
R_5	$R_6(A_3)$	$R_1(A_1)$,$R_3(A_2)$
R_6	$R_5(A_3)$	$R_2(A_1)$,$R_4(A_2)$
R_7	$R_8(A_3)$	$R_1(A_2)$,$R_3(A_1)$
R_8	$R_7(A_3)$	$R_2(A_2)$,$R_4(A_1)$

Table 4 summarizes the resulting number of nearest hit and miss score contributions from

the Table 3. When there is a tie between instances for nearest neighbor, we give each feature difference half credit since only one can contribute at a time. We can see from Table 3 that among nearest hits we observe no feature value differences for A_1 or A_2, but a total of 8 of them for A_3 across all 8 cycles. Among nearest misses, we observe 8 feature value differences for both A_1 and A_2. However since it would be only one or the other each scoring iteration they each receive a total of 4. Lastly, the Relief scoring scheme applies negative scoring to nearest hits, and positive scoring to nearest misses. As seen in this simple example, Relief easily differentiate between the relevant interacting features A_1 and A_2 (each with a final score of 4) and the irrelevant feature A_3 (with a final score of -8).

Table 4: Summary of score contributions in 2-way epistasis problem yielding Relief scores.

	A_1	A_2	A_3	Relief Scoring
Nearest Hit	0	0	8	- 1
Nearest Miss	4	4	0	+1
Relief Score Total	4	4	-8	

3. A Review of Relief-based Algorithms

In this section, we offer a comprehensive review of the RBAs inspired by Relief. We highlight major research themes, advancements, concepts, and adaptations.

3.1. ReliefF: The Best Known Variant

The original Relief algorithm [53] is rarely applied in practice anymore and has been supplanted by *ReliefF* [58] as the best known and most utilized RBA to date. Notably, the 'F' in ReliefF refers to the sixth algorithm variation (from A to F) proposed by Kononenko [58]. The ReliefF algorithm has been detailed in a number of other publications [60, 62, 91]. Here we highlight four key ways that ReliefF differs from Relief. First, ReliefF relies on a 'number of neighbors' user parameter k that specifies the use of k nearest hits and k nearest misses in the scoring update for each target instance (rather than a single hit and miss). This change increased weight estimate reliability, particularly in noisy problems. A k of 10

was suggested based on preliminary empirical testing and has been widely adopted as the default setting. This algorithm variation was originally proposed under the name *ReliefA*.

Second, three different strategies were proposed to handle incomplete data (i.e. missing data values). These strategies were proposed under the names Relief(B-D). When encountering a missing value, the 'best' approach (*ReliefD*), sets the *diff* function equal to the class-conditional probability that two instances have different values for the given feature. This is implicitly an interpolation approach.

Third, two different strategies were proposed to handle multi-class endpoints. These strategies were proposed under the names ReliefE and ReliefF. *ReliefF*, which inherited the changes proposed in ReliefA and ReliefD, was selected as the 'best' approach. During scoring in multi-class problems, ReliefF finds k nearest misses from *each* 'other' class, and averages the weight update based on the prior probability of each class. Conceptually, this encourages the algorithm to estimate the ability of features to separate all pairs of classes regardless of which two classes are closest to one another. Lastly, since it is expected that as the parameter m approaches the total number of instances n, the quality of the weight estimates becomes more reliable, Kononenko [58] proposed the simplifying assumption that $m = n$. In other words, every instance in the dataset gets to be the target instance one time (i.e instances are selected without replacement). We adopt this assumption in deriving the time complexity of RBAs below. This is why the asymptotic time complexity of 'core' Relief algorithms are given as $\mathcal{O}(n^2 \cdot a)$, rather than $\mathcal{O}(m \cdot n \cdot a)$.

3.2. Organizing RBA Research

Following ReliefF, a number of variations and improvements have been proposed. Table 5 chronologically organizes summary information on key RBAs dealing with fundamental feature selection problems. Brief descriptions of the algorithms and their contributions are given along with our designation of the closest parent algorithm in parentheses. Parent algorithms may deviate from what was described in the respective publication(s). This is due to inconsistent nomenclature (e.g. ReliefA implementations being more generically referred to as ReliefF even if they did not include extensions for missing or multi-class data handling)

Table 5: Summary of key Relief-based algorithms.

Algorithm, Reference(s), & Description/Contribution (*Closest Parent Algorithm*)	**Time Complexity** Asymptotic ($\mathcal{O}$) / Complete (Approx.)	**Focus**	**Continuous F.**	**Multi-class**	**Regression**	**Missing Data**
Relief [53, 52] The first 'filter' feature selection algorithm sensitive to feature dependencies.	$\mathcal{O}(n^2 \cdot a)$ $\chi + c_3 2na$	C	X			
ReliefA [58] Introduced k nearest neighbor scoring to address noisy data. (*Relief*)	$\mathcal{O}(n^2 \cdot a)$ $\chi + c_3 2kna$	C	X			
Relief(B-D) [58] Strategies for handling incomplete data. ReliefD selected as 'best'. (*ReliefA*).	$\mathcal{O}(n^2 \cdot a)$ $\chi + c_3 2kna$	D	X			X
Relief(E-F) [58] Strategies for multi-class endpoint. ReliefF became the standard. (*ReliefD*)	$\mathcal{O}(n^2 \cdot a)$ $\chi + c_3 2kna$	D	X	X		X
RReliefF [60] Adapting to regression problems. [89] Adopts exponential instance weighting by distance from target. (*ReliefF*)	$\mathcal{O}(n^2 \cdot a)$ $\chi + c_3 na + c_y(1 + 2a)$ $+c_{y1} 0.5n^2 + c_{y2} kn + c_{y3} 2kna$	D C	X		X	X
Relieved-F [55] Deterministic neighbor selection and incomplete data handling. (*ReliefF*)	$\mathcal{O}(n^2 \cdot a)$ $\chi + c_3 2kna$	C D	X	X		X
Iterative Relief [31] Address bias against non-monotonic features. The first iterative approach. Neighbors uniquely determined by radius (r) and instances weighted by distance from target. (*Relief*)	$\mathcal{O}(\mathcal{I} \cdot n^2 \cdot a)$ $\mathcal{I}(\chi + c_3 0.5n^2 a + c_{y1} a)$ $+c_{y2} a$	C I	X			
I-RELIEF [103, 102] All instances sigmoidally weighted by distance from target, i.e. no defined neighbors. Proposed online learning variant. [104] Local-learning updates between iterations for improved convergence. (*Iterative Relief*)	$\mathcal{O}(\mathcal{I} \cdot n^2 \cdot a)$ $\mathcal{I}(\chi + c_3 n^2 a + c_{y1} a)$ $+c_{y2} a$	C I	X	X		
TuRF [77] Address noise and large feature spaces with iterative removal of fixed percent of lowest scoring features. (*ReliefA*)	$\mathcal{O}(\mathcal{I} \cdot n^2 \cdot a)$ $\sum_{i=0}^{\mathcal{I}} [f(\text{ReliefA}) +$ $c_y a_i \log(a_i)]$	I				
Evaporative Cooling ReliefF [70] Address noise and large feature spaces with iterative 'evaporative' removal of x lowest quality features via (*ReliefA*) and mutual information (or random forest scores) [69]. Privacy protection variant [64].	$\mathcal{O}(\mathcal{I} \cdot n^2 \cdot a)$ $\sum_{i=0}^{\mathcal{I}} [f(\text{ReliefA}) +$ $c_{y1} 6an +$ $c_{y2} a_i \log(a_i)]$	I	X			
EReliefF [81] Seeks to address issues related to incomplete and/or multi-class data. (*ReliefF*)	$\mathcal{O}(n^2 \cdot a)$ $\chi + c_3((p \log p + q \log q) + na)$	D	X	X		X
VLSReliefF [33, 65] Efficient interaction detection in large feature spaces by scoring random feature subsets. (*ReliefA*)	$\mathcal{O}(S \cdot n^2 \cdot a_s)$ $S(\chi_s + c_3 2kna_s + c_{y1}) + c_{y2} a_s$	E				
***i*VLSReliefF** [33] Iterative, TuRF-like extension of (*VLSReliefF & TuRF*).	$\mathcal{O}(\mathcal{I} \cdot S \cdot n^2 \cdot a_s)$ $\sum_{i=0}^{\mathcal{I}} [f(\text{VLSReliefF}) + c_y a_i \log(a_i)]$	I E				
ReliefMMS [21] Feature weight relative to average feature *diff* between instance pairs. (*ReliefA*)	$\mathcal{O}(n^2 \cdot a)$ $\chi + c_3 2kna$	C				
SURF [39] Threshold-based nearest neighbors for scoring. (*ReliefA & Iterative Relief*)	$\mathcal{O}(n^2 \cdot a)$ $\chi + c_3 0.5n^2 a$	C				
SURF* [38] Introduces 'far' scoring to improve detection of epistatic interactions. (*SURF*)	$\mathcal{O}(n^2 \cdot a)$ $\chi + c_3 n^2 a$	C				
SWRF* [101] Extends (*SURF**) with sigmoid weighting taking distance from threshold into account. Introduces modular framework for Relief development (MoRF).	$\mathcal{O}(n^2 \cdot a)$ $\chi + c_3 n^2 a + c_y 0.25n^2$	C				
LH-RELIEF [15] Feature weighting by measuring the margin between the sample and its hyperplane. (*I-RELIEF*)	$\mathcal{O}(\mathcal{I} \cdot n^2 \cdot a)$ $\chi + c_3 2na + c_y na$	C I	X	X		
MultiSURF* [37] Target instance defined neighbor threshold and dead-band no-score zone. (*SURF**)	$\mathcal{O}(n^2 \cdot a)$ $\chi + c_3 0.62n^2 a + c_y n^2$	C				
ReliefSeq [71] Feature-wise adaptive k. Choice of k impacts differential detection of main effect vs. interaction. (*ReliefA*)	$\mathcal{O}(n^2 \cdot a)$ $\chi + k_{max}(c_3 2kna) + c_y k_{max} \log k_{max}$	C	X	X	X	
MultiSURF [113] Removed 'far' scoring from (*MultiSURF**) to recover main effects. Added strategies to address data types as part of ReBATE. RBAs succeed with heterogeneity.	$\mathcal{O}(n^2 \cdot a)$ $\chi + c_3 0.31n^2 a + c_y n^2$	C D	X	X	X	X

- $\chi = c_0 a + c_1 0.5n^2 a + c_2 n \log n$ *(The universal terms of RBA complete time complexities are given by χ)*
- *n is the number of training instances, a is the number of features, and c_y represents unique constant terms.*
- k is the user specified number of nearest neighbors; p and q specify the number of hits and misses, respectively.
- $\mathcal{I}$ is the number of iterations (determined by user parameters)
- f(ReliefA) and f(VLSReliefF) are the complete time complexities of ReliefF and VLSReliefF, respectively)
- a_i is the remaining number of features at iteration i
- $a_i = a(1 - p)^i$ (for TuRF or iVLSReliefF), and $a_i < a$ (for Evap. Cool. ReliefF)
- a_s is the number of features in each subset, S is the number of subsets, and χ_s is χ where a is replaced by a_s
- k_{max} is the maximum k considered by ReliefSeq

and some previously missed citations of relevant work. In the sections following this review we will adopt the name 'ReliefF' for any Relief algorithm that uses k nearest neighbors, and makes the $m = n$ assumption (regardless of any data type handeling implementations), as

has become common in the literature [76, 109].

Table 5 also provides asymptotic time complexities $\mathcal{O}$ (highlighted in yellow), to easily compare run time order of magnitude. Additionally, based on algorithmic descriptions in the respective publications, we provide approximations of complete algorithm time complexities (highlighted in blue). These equations provide computational insight into algorithmic differences and reveal more subtle run time differences. Variables are defined below the table. Constants in the table are numbered (e.g. c_3) according to which additive term in the equation that it corresponds to (e.g. $c_3 2na$). The term with c_0 initializes the feature weights to 0.0. The term with c_1 calculates all unique pairwise distances between instances. Given the assumption of $m = n$, it is computationally more efficient to pre-compute all pairwise distances rather than on a target by target basis as proposed in the original Relief algorithm. The term wit c_2 corresponds to finding the nearest instances (or separates nearest from furthest in the case of SURF* and MultiSURF*). The term with c_3 corresponds with updating the feature weights. Algorithms that require additional terms label the corresponding constants generically with c_y or a numbered variation. Note that complete time complexity terms that are universal to all RBAs are represented by χ within the table.

Todorov [109] suggested that there are two primary directions of RBA development: (1) strategies for selecting and/or weighting neighbors in scoring (i.e. what we call 'core algorithm' developments), and (2) strategies for moving beyond a single pass over the data to 'iterative' implementations. In Table 5, the column labeled 'Focus' identifies the respective research direction(s) of the corresponding algorithm, going beyond the two suggested by Todorov. These include; (1) 'C' for *core algorithm*, i.e. variants impacting a single run through the training data such as variations in neighbor selection or scoring, (2) 'I' for *iterative approach*, i.e. variants designed to iteratively apply a core Relief algorithm for multiple cycles through the training data, (3) 'E' for *efficiency*, i.e. variants seeking to improve computational efficiency, and (4) 'D' for *data type handling*, i.e. variants that seek to address the challenges of different data types including continuous feature values, multi-class endpoints, continuous endpoints (i.e. regression), or missing data values. The remaining columns of Table 5 indicate (with 'X's') whether the corresponding algorithm

explicitly considered or implemented algorithm extensions to handle any of the four data types above, beyond discrete features and binary classes.

We can make some basic observations from this table. First, relatively little attention has been paid to adapting RBAs to regression problems. Second, the majority of proposed variations have focused on data with discrete-valued features and a binary endpoint. Notably many of these works have been application driven, focusing on feature selection in genomics problems with single nucleotide polymorphisms (SNPs) as features that can have one of three discrete values (0, 1, or 2) and a binary endpoint representing sick vs. healthy subjects [77, 70, 33, 39, 69, 38, 101, 37]. RBAs are particularly appealing in this domain since the number of features (a) in respective datasets is typically much larger than the number of available training instances (n), and RBAs have a linear time complexity with respect to a, but a quadratic time complexity with respect to n. However, core RBAs aimed at SNP analysis, such as SURF [39], SURF* [38], SWRF* [101], and MuliSURF* [37] were not originally extended to handle other basic data types. Table 5 concludes with our recently proposed core algorithm named *MultiSURF* [113]. MultiSURF performed most consistently across a variety of problem types (e.g. 2-way and 3-way interactions as well as heterogeneous associations) in comparison with ReliefF, SURF, SURF*, MultiSURF* and a handful of other non-RBA features selection methods. The work by Urbanowicz et al. [113] also extended MultiSURF along with ReliefF, SURF, SURF*, and MultiSURF* to handle a variety of different data type issues under a unified implementation framework called ReBATE.

The following subsections go into greater depth describing notable RBAs that fall into our 'core', 'iterative', 'efficiency' or 'data type' categories, as well as peripheral RBA research directions not included in this table.

3.3. Neighbor Selection and Instance Weighting

This section references algorithms in Table 5 with a core focus (C). How do we select nearest hits and misses? What number of neighboring instances should be used in feature scoring? Is there information to be gained from considering 'far' instance pairs? How

should the scoring contribution of those neighboring instances be weighted; also referred to as observation weighting by Todorov [109]? These are the primary questions that have been asked in the context of core RBAs. Note that *instance weighting* refers to the weight placed on an instance during the scoring update. By default, most RBAs (including ReliefF) assign neighboring instances a weight of 1, and all others a weight of 0. Figure 3 illustrates how a variety of RBAs (arranged chronologically) differ with respect to neighbor selection and instance weighting. For every RBA, we assume that each instance in n gets the opportunity to be the target instance during feature scoring. Note that for ReliefF in Figure 3, a k of 3 is chosen for illustration simplicity, but a k of 10 is most common. Figure 3 includes RBAs that adopt a 'distance-from-target' instance weighting scheme, i.e. Iterative Relief, I-RELIEF, and SWRF*, where instance weight ranges from 0 to 1. For all other RBAs in the table, instances that are identified as either near or far, have a full weight of 1, while all others have a zero instance weight in feature scoring. Three of the RBAs (i.e. I-RELIEF, SURF*, and SWRF*) are unique in giving all instances, besides the target, some weight each scoring cycle.

The original Relief algorithm used two nearest neighbors (i.e. one nearest hit and miss), each with an equal instance weighting [53] . ReliefA through ReliefF used k nearest neighbors with equal instance weighting [58]. Iterative Relief was the first to specify a radius r around the target instance that would define the cutoff for which instances would be considered neighbors [31]. Additionally, while RRelief [60] was the first to suggest differentially weighting instances based on their distance from the target instance in regression, Iterative Relief was the first to suggest this for discrete class problems [31]. The effect there was that the closest neighbors had a greater impact on feature weighting than those out towards the edge of the radius. I-RELIEF proposed forgoing the determination of neighbors entirely, instead using an instance weighting function over the entire set of hit and miss instances, again so that the closest neighbors had the greatest impact on feature weighting [103, 102, 104]. Similar to Iterative Relief, SURF employed a distance threshold T to define instances as neighbors (where T was equal to the average distance between all instance pairs in the data) [39]. However, in contrast with Iterative Relief, SURF utilizes equal instance weights for all

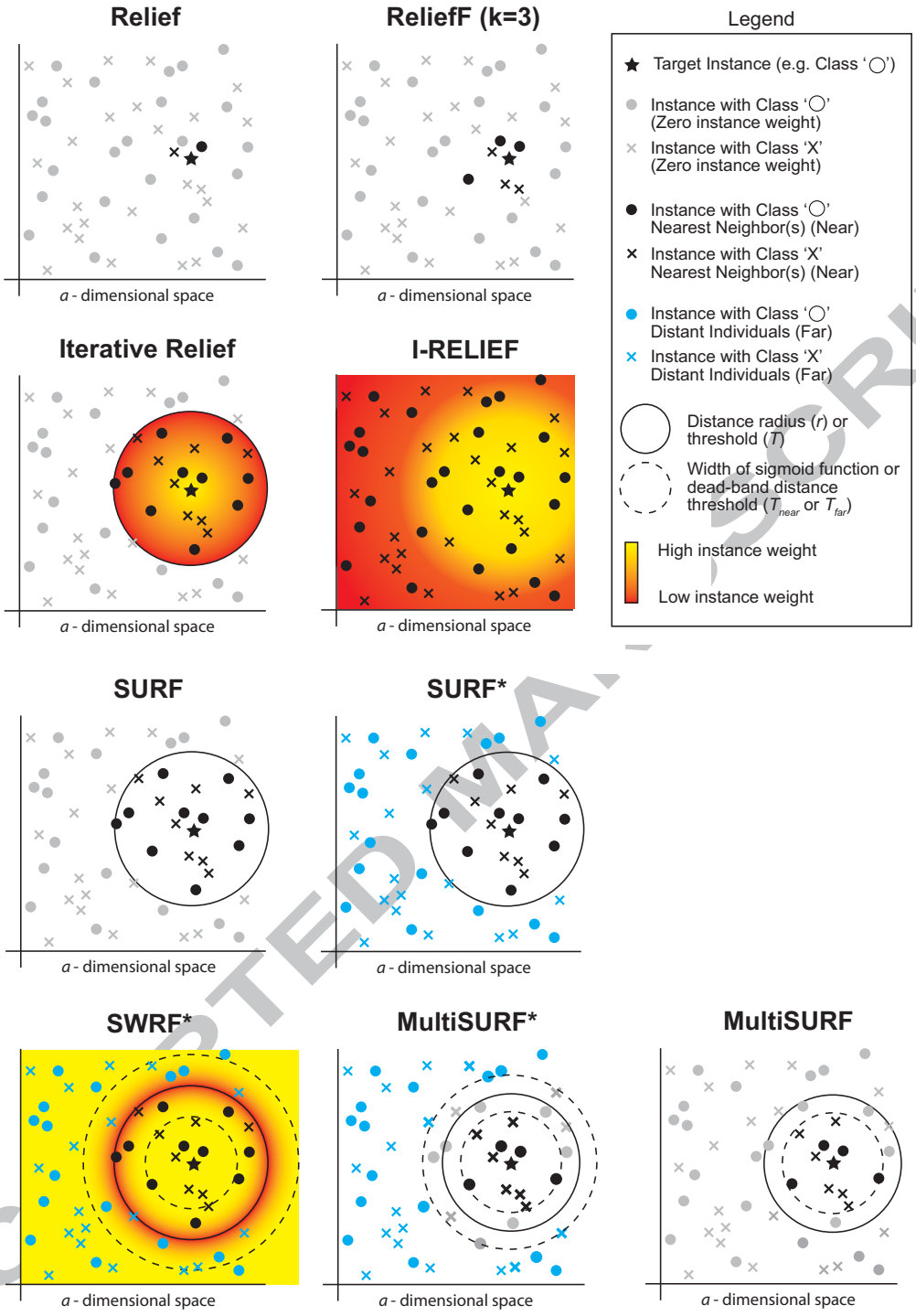

Figure 3: Illustrations of RBA neighbor selection and/or instance weighting schemes. Methods with a red/yellow gradient adopt an instance weighting scheme while other methods identify instances as 'near' or 'far' which then contribute fully to feature weight updates. These illustrations are conceptual and are not drawn to scale.

instances defined as neighbors.

The SURF* expansion introduced the concept of instances that were near vs. far from the target instance [38] (see Figure 3). Applying the same T from SURF, any instance within the threshold was considered near, and those outside were far. SURF* was similar to I-RELIEF in that all other instances besides the target contributed to scoring. This is reflected in the complete time complexity of the two algorithms where the feature scoring term is $c_3 n^2 a$ for both. However SURF* weights all 'near' instances equally, and all 'far' instances in a similarly equal, but opposite way. Specifically, for far instances, feature value differences in hits receive a $+1$ while feature value differences in misses receive a -1, i.e. the opposite scoring strategy than what is presented in Figure 2. Note that in mathematics the '*' indicates opposite, therefore RBAs that utilize 'far' scoring have been given this affix. Some publications have instead have used the affix 'STAR' (e.g. SURFSTAR).

SWRF* integrated concepts from SURF* and I-RELIEF, preserving the definition of near and far established in SURF*, but adopting a sigmoid instance weighting function from I-RELIEF so that the nearest of neighbors have the greatest *standard* scoring weight, while the farthest 'far' instances have the greatest *opposite* scoring weight. Instances near T have the smallest scoring weights. The width of the SWRF* sigmoid function is proportional to the standard deviation σ of all pairwise instance distances. In contrast with SWRF*, MultiSURF* took an alternate approach to discounting instances near T by introducing a dead-band zone on both the near and far side of T (i.e. T_{near} or T_{far}) [37]. Any instances that fell within this 'middle' distance zone were excluded from scoring (i.e. neither near or far). Another major difference is that MultiSURF* defined T as the mean pairwise distance between the target instance and all others, as opposed to the mean of all instance pairs in the data. This adapts the definition of near/far to a given part of the feature space. Similarly, the width of the dead-band zone is the standard deviation σ of pairwise distances between the target instance and all others. One final difference between MultiSURF* and SURF* is that the 'far' scoring logic was inverted to save computational time. Specifically in SURF*, *differences* in feature values in hits yielded a reduction in feature score, and an increase in misses. Since different feature values are expected to be more frequent in far individuals, in

MultiSURF*, *same* feature values in hits yielded an increase in feature score, and a decrease in misses. Also recognizing the importance of neighbor selection, ReliefSeq proposed the concept of an adaptive k for all features [71]. ReliefSeq effectively examines all possible values of k up to a k_{max} and for each feature, picking the k that yields the largest feature weight in the final scoring. While more computationally intensive, the authors claim that varying k on a feature by feature basis provides greater flexibiliy in detecting either main or interaction effects. Notably, ReliefSeq was applied to the analysis of RNA-Seq expression data. Most recently, MultiSURF was proposed, preserving most aspects of MultiSURF* but eliminating the 'far' scoring [113]. This was due to the fact that while 'far' scoring improved the detection of 2-way interactions, it also greatly deteriorated the ability of RBAs to detect simple main effect associations. MultiSURF is claimed to balance performance with respect to its (1) ability to detect main or interaction effects, (2) computational efficiency, (3) ease of use (i.e. no parameters to set), and (4) applicability to a variety of data types.

3.4. Iterative and Efficiency Approaches

This section references algorithms in Table 5 with an iterative (I) or efficiency (E) focus. As noted earlier, core RBA performance is understood to degrade as the number of irrelevant features becomes 'large' particularly with respect to noisy problems. This has been observed or noted in a number of works [91, 103, 102, 77, 33, 39, 109]. As pointed out by Sun and Li [103], this is because a core RBA defines nearest neighbors in the original feature space, which are highly unlikely to be the same in weighted space (i.e. the space where we have assigned low weights to features least likely to be relevant). To deal with this issue, iterative and efficiency approaches have been proposed that are wrapped around or integrated into core RBAs.

Iterative Relief introduced the idea of running the core RBA more than once, each time using the feature weights W from the previous iteration to update pairwise distance calculations such that a low scoring feature from the previous iteration has less influence on instance distance in the current iteration [31] (see Figure 4). These 'temporary' feature weights were referred to as *parameters* by Todorov [109] and designated by the variable

ϕ. Iteratively updating the distance weights can cause certain samples to enter and leave neighborhoods of other samples. To reduce discontinuities in the feature weight estimates that arise from changing neighborhoods, Iterative Relief also introduced a radius to define neighborhoods rather than a set number of instances, as illustrated in Figure 3. Iterations continued until the weights W converge, or until some maximum number of iterations is reached. It is important to be aware of stop criteria since iterative approaches can become quite computationally expensive.

Sun and Li introduced another iterative Relief method known as I-RELIEF [103]. I-RELIEF adopts an iterative approach similar to Iterative Relief, but mathematically derived Relief as an online algorithm that solves a convex optimization problem with a margin-based objective function [103, 102]. As such, I-RELIEF has been described as an outlier removal scheme since the margin averaging is sensitive to large variations [15]. Later, Local Learning I-RELIEF (our name for the unnamed algorithm) applied the concept of local learning to improve iterative convergence by promoting sparse feature weighting [104]. 'Sparse' refers to there being a minimal number of converged feature weights with a value greater than zero. This was achieved by introducing the ℓ_1 norm penalty (as in *lasso*) into optimization of I-RELIEF.

TuRF presents a much simpler iterative approach that can easily be wrapped around any other core RBA despite the fact that it was originally designed to be used with ReliefF [77] (see Algorithm 2). TuRF is essentially a recursive feature elimination approach. Each iteration, the lowest scoring features are eliminated from further consideration with respect to both distance calculations and feature weight updates (see Figure 4). However selecting the number of iterations (p) is not trivial. Evaporative Cooling ReliefF offers another novel approach that employs simulated annealing to iteratively remove lowest relevance features, where relevance is a function of both ReliefF and (myopic) mutual information scores [70], or instead ReliefF and transformed random forest importance scores [69]. Most recently, the evaporative cooling concept was adapted to the challenge of patient privacy preservation, and was extended for continuous feature analysis (i.e. fMRI network data) [64].

It was noted by Todorov [109], that for TuRF, or any of the other iterative approaches

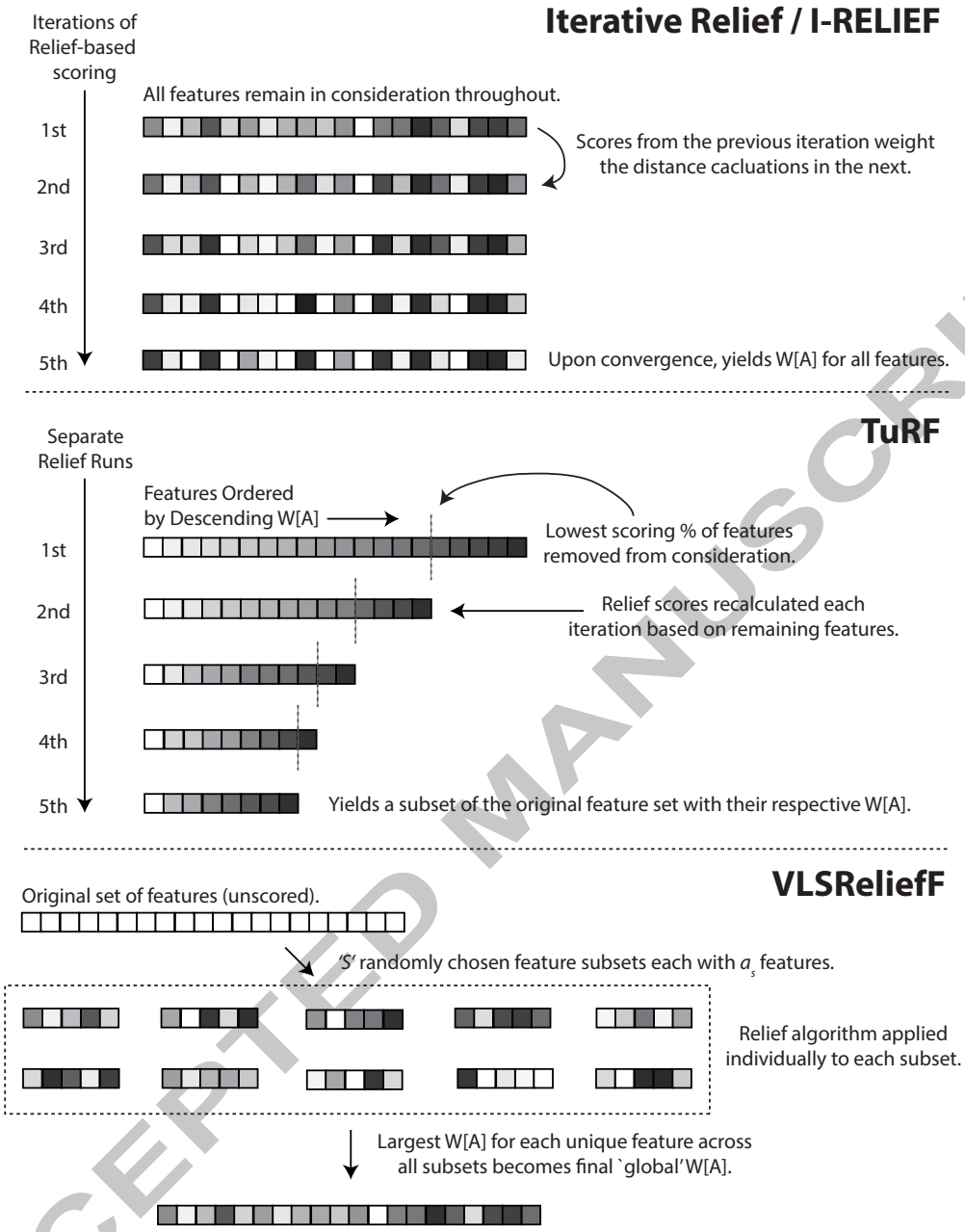

Figure 4: Illustrations of the basic concepts behind key iterative and efficiency approaches including TuRF, Iterative Relief/I-RELIEF, and VLSReliefF. Features are represented as squares, where darker shading indicates a lower feature weight/score.

that could 'remove' features from consideration by giving them a ϕ of 0 in the distance calculation, it is still possible to estimate a relevance score $W[A]$ for it (thus perhaps giving the feature the opportunity to be reintroduced as relevant later). It was warned that this

Algorithm 2 Pseudo-code of TuRF algorithm

```
a ← number of attributes (i.e. features)
Parameter: p ← number of iterations

for i:=1 to p do
   run ReliefF and estimate feature weights (W)
   sort features by weight
   remove p/a of remaining features with lowest weights
end for
return last ReliefF weight estimates for remaining features
```

could lead to undesirable oscillatory behavior and poor convergence of scoring. It should also be mentioned that any of the iterative strategies for updating parameters (e.g. I-RELIEF) could also be combined with a specific core RBA besides the one it was originally implemented with (e.g. the iterative component of Iterative Relief could be wrapped around the core SWRF* approach).

Despite the fact that core Relief algorithms are relatively fast, they can still be slow in very large feature spaces (common to bioinformatics), or more importantly, when large training sets are available (because RBAs scale quadratically with the number of instances). One of the more unique RBA proposals focuses on improving algorithm efficiency with regards to both run time and performance. Specifically, VLSReliefF targets the detection of feature interactions in very large feature spaces [33] (see Figure 4). The principle behind VLSReliefF is simply that weights estimated by ReliefF are more accurate when applied to smaller feature sets. Therefore it individually applies ReliefF to some number of randomly selected feature subsets (S), each of size $a_s < a$ with the expectation that at least one subset in the population will contain all interacting features that are associated with endpoint (and will thus have elevated weights for those features). The partial ReliefF results are integrated by setting the 'global' feature weight $W[A]$ to the maximum 'local' weight for a given feature across all S ReliefF runs. With regards to detecting feature interactions, the risk of this approach is that if all relevant interacting features don't appear together in at least one of the S random feature subsets, then the interaction will likely be missed. That is why properly setting the S and a_s is critical to maximizing the probability of success. Furthermore,

knowing the desired order of interaction to be sought (e.g. 2-way, 3-way) is needed to calculate S. The VLSReliefF concept was inspired by work proposing a Random Chemistry ReliefF algorithm, an iterative approach that ran ReliefF on random feature subsets [34]. The VLSReliefF concept could be integrated with other core RBAs. An iterative TuRF-like version called iVLSReliefF has also been proposed [33].

3.5. Other Relief-based methods

This section references algorithms in Table 5 with a data type focus (D). In the interest of breadth, it also summarizes ancillary Relief expansions not included in Table 5. In a number of studies, emphasis has been placed on the handling of different data types beyond discrete features and binary classes. Beyond RReliefF [60, 89] little attention has been paid to handling continuous endpoints (i.e. regression) other than examples like FARelief [7], or RM-RELIEF [66]. Alternative methods of handling continuous features beyond those originally introduced in Relief were described by Demšar [27] and Blessie and Karthikeyan [10]. An alternative method for handling a multi-class endpoint was described by Ye et al. [118]. Little else has been proposed for handling multi-class endpoints or missing data beyond adaptations of those from the original ReliefF algorithm [58].

The Relief concept has also been adapted to a variety of specific data problems. The most common problem is the removal of redundant features as discussed earlier. Previously, a handful of stand-alone RBAs, or some combination of an existing RBA with a redundancy removal heuristic have been proposed to deal with feature redundancy [8, 35, 40, 41, 117, 19, 73, 122, 67, 17, 1]. Another popular area of investigation is the adaptation of Relief to multi-label learning, i.e. where instances can each have more than one class label assigned to it [57, 99, 100, 96, 83, 86]. Other problems to which RBAs have been adapted include: multiple instance learning, i.e. bags of not clearly labeled instances [120, 121], dealing with non-monotonic relationships [9, 31], dealing with survival data (i.e. data exploring the duration of time until one or more events happen) [6], dealing with imbalanced data [88, 49], clustering [24], and feature extraction [105].

Other notable Relief methodological variations include approaches for feature set evalu-

ation [4], instance selection [25], and ensemble learning [94, 124]. Verma et al. [115] recently introduced the concept of *collective feature selection*, which selects the union of features from top performing algorithms (including RBAs), highlighting the benefit of selecting top features identified by algorithms with different strengths and weaknesses.

Attempts at parallelizing RBAs for run time efficiency have been proposed by Lee et al. [65] and Eiras-Franco et al. [32]. Many other works applying RBAs or drawing inspiration from them exist in the literature but are beyond the scope of this methodological review. Earlier reviews in the form of book chapters include (1) Kononenko and Sikonja's focused examination of their own ReliefF and RReliefF contributions, (2) Moore's brief review of ReliefF and select RBAs in the context of epistasis analysis, and (3) Todorov's more recent summarial overview of target RBAs and advancements in the context of detecting gene-environment interactions [61, 76, 109].

3.6. RBA Evaluations

The datasets chosen to test, evaluate, and compare RBAs in previous studies have often focused on (1) a small sample of simulated or toy benchmark datasets [53, 21], (2) a set of real-world benchmarks (e.g. from the UCI repository) [8, 35, 84, 98, 36], (3) some real data analysis that is new or yet to be established as a benchmark [29], or (4) some combination of these three [58, 91, 103, 102, 70, 104, 15, 1, 30, 115].

Some RBAs have been compared across a spectrum of simulated datasets capturing a greater breadth of problem scenarios. This was true for TuRF, SURF, SURF*, SWRF*, and MulitSURF* each developed with the bioinformatic detection of epistastic interactions in mind [77, 39, 38, 101, 37]. In each of these studies, RBAs were evaluated on datasets with purely epistatic 2-way interactions (i.e. no main effects) with varying numbers of training instances (e.g. 200 to 3200) as well as different heritabilities (e.g. 0.01 to 0.4). Heritability is a genetics term that indicates how much endpoint variation is due to the genetic features. In the present context heritability can be viewed as the signal magnitude, where a heritability of 1 is a 'clean' dataset (i.e. with the correct model, endpoint values will always be correctly predicted based on feature values), and a heritability of 0 would be a completely

noisy dataset with no meaningful endpoint associations. All features were simulated as single nucleotide polymorphisms (SNP) that could have have a discrete value of (0, 1, or 2) representing possible genotypes. In each dataset, two features were predictive (i.e. relevant) of a binary class while the remaining 998 features were randomly generated, based on genetic guidelines of expected genotype frequencies, yielding a total of 1000 features. Similarly, VLSRelief explored SNP simulations and 2-way epistasis varying heritability similar to the other studies, but fixing datasets to 1600 instances and simulating datasets with either 5000 or 100,000 total features [33]. It should be noted that most of these studies sought to compare core RBAs to respective iterative TuRF expansions, which is why larger feature spaces were simulated.

Another recent investigation compared ReliefF, TuRF, SURF, chi-square, logistic regression, and odds ratio, in their ability to rank features in SNP data simulated to include 15 epistatic feature pairs each contributing additively to class determination [30]. RBAs again performed best both in this simulation, and in identifying interacting SNPs from a real world genome-wide association study (GWAS), confirmed by exhaustive calculation of information gain.

Beyond the simulated genetic analyses described above, there are only a couple examples of comparative evaluations of RBAs over a reasonably diverse set of synthetic datasets including one of Relief [5] and another of ReliefF [12] in comparison with other feature selection approaches. Notably in both studies, the selected RBA stood out as the more reliable and successful feature selection algorithm, except when dealing with removing feature redundancy. Most recently, a much wider comparison of core RBA algorithms was completed over a broad spectrum of simulated datasets with various properties and underlying patterns; including main effects, interactions, and patterns of genetic heterogeneity [113]. That study (1) confirmed the utility of RBA methods over chi-square, ANOVA, mutual information, and random forest based approaches for feature selection, (2) illustrated performance differences between a number of core RBAs (i.e. ReliefF, SURF, SURF*, MultiSURF*), and (3) introduced MultiSURF and novel implementations of ReliefF.

Clearly, the ultimate goal in developing feature selection methods is to apply them to

real world problems and ideally facilitating the modeling of previously unknown patterns of association in that data. However, as similarly argued by Robnik-Šikonja and Kononenko [91], Belanche and González [5], Bolón-Canedo et al. [12], and Olson et al. [80]: to properly evaluate and compare methodologies, diverse simulation studies should first be designed and applied. This is because: (1) Uniquely, a simulation study can be designed by systematically varying key experimental conditions, e.g. varying noise, the number of irrelevant features, or the underlying pattern of association in the data. This allows us to explicitly identify generalizable strengths and weakness of the methods and to draw more useful conclusions; (2) The ground-truth of the dataset is known, e.g. we know which features are relevant vs. irrelevant, we know the pattern of association between relevant features and endpoint, and we know how much signal is in the dataset (i.e. so we know what testing accuracy should be achievable in downstream modeling). This knowledge of ground truth allows us to perform power analyses over simulated dataset replicates to directly evaluate the success rate of our methodologies.

3.7. Software Availability

ReliefF [58] and its counterpart for dealing with regression data, i.e. RReliefF [60], are currently the most widely implemented RBAs. They can be found in the following freely available data mining software packages: CORElearn [92] (in C++), Weka [43] (in Java), Orange [28], and R [47] (within the dprep and CORElearn packages). A C++ version of ReliefF is available as part of Evaporative Cooling ReliefF[1].

Separately, implementations of ReliefF [58], SURF [39], SURF* [38], and MultiSURF* [37] as well as the iterative TuRF algorithm [77] were made available in the open source Multifactor Dimensionality Reduction (MDR) [87] software package[2]. These Java implementations are computationally efficient, but can only handle 'complete data' (i.e. no missing values) with discrete features and a binary endpoint. Python 2.7 versions of these algorithms were later implemented and made available within the open source Extended Su-

[1] https://github.com/insilico/EC/blob/master/src/library/ReliefF.cpp
[2] http://sourceforge.net/projects/mdr

pervised Tracking and Classifying System (ExSTraCS)[3] [110, 112]. These implementations were less computationally efficient, but extended each algorithm to handle different data types, including continuous features, multi-class endpoints, regression, and missing data. Other C# implementations of ReliefF, SURF*, and SWRF* were made available as part of the modular framework for Relief development (MoRF)[4] [101]. A C# implementation of ReliefSeq[5] is also available [71]. Most recently, ReliefF, SURF, SURF*, MultiSURF*, MultiSURF, and TuRF were all implemented within the Relief-Based Algorithm Training Environment (ReBATE). These ReBATE implementations were coded more efficiently in Python (2 and 3) and similarly extended to handle the aforementioned data types. Stand-alone, Cython-optimized ReBATE software[6] and a scikit-learn [82] compatible format[7] were both made available with thorough documentation.

4. Conclusion

In this work we have placed Relief-based algorithms (RBAs) in the context of other feature selection methods, provided an in-depth introduction to the Relief-algorithm concept, described four general branches of RBA research, and reviewed key methodological differences within these branches. This work highlights a number of conclusions that can be made about RBAs, including (1) they are generally proficient at detecting not only univariate effects but 2-way interactions as well, (2) they scale linearly well with the number of features, but quadratically with the number of training instances, (3) iterative and efficiency approaches offer a solution to scaling RBAs up very large feature spaces, (4) RBAs are 'anytime' algorithms, (5) the choice of instance neighbors is a critical aspect of RBA success, setting these methods apart from other feature selection approaches, (6) the individual feature weights output by an RBA can be used to probabilistically guide downstream machine learning methods (i.e. feature weighting), (7) RBAs have already been flexibly adapted to

[3] https://github.com/ryanurbs/ExSTraCS_2.0
[4] https://github.com/mattstokes42/MoRF
[5] http://insilico.utulsa.edu/index.php/reliefseq
[6] https://github.com/EpistasisLab/ReBATE
[7] https://github.com/EpistasisLab/scikit-rebate

an array of data types and specific application domains, and (8) implementations of a variety of RBAs are available.

This promising area of feature selection will likely benefit from future research focusing on: (1) the most effective and reliable instance weighting approach (e.g. classic 'full' instance weighting, or 'distance-from-target-based instance weighting) (2) optimizing the number of neighbors and neighbor selection to improve RBA performance in a problem-dependent manner (3) improved strategies (e.g. iterative or efficacy) for scaling RBAs to large-scale data (i.e. many features and/or many instances), (4) adapting to other new problem domains (e.g. temporal data), (5) limiting or eliminating user defined RBA run parameters (to make them easier to apply, and require less prior knowledge about the problem domain to set correctly), (6) new strategies for collective or ensemble feature selection, and (7) more rigorous (e.g. statistical) approaches for determining a selection cutoff threshold.

RBAs represent a powerful family of feature selection approaches that strike a key balance between ability to detect complex patterns, flexibility to handle different data types, and computational efficiency. While ReliefF has been the staple go-to algorithm of the family for many years, many advancements have since been made. Understanding these advancements is key to selecting the best approach for application as well as in guiding the development of even better feature selection approaches.

Acknowledgements

We thank the reviewers for their thoughtful comments. Special thanks to Brian Cole for his constructive feedback. This work was supported by National Institutes of Health grants AI116794, DK112217, ES013508, EY022300, HL134015, LM009012, LM010098, LM011360, TR001263, and the Warren Center for Network and Data Science.

References

[1] Agre, G., Dzhondzhorov, A., 2016. A weighted feature selection method for instance-based classification. In: International Conference on Artificial Intelligence: Methodology, Systems, and Applications. Springer, pp. 14–25.

[2] Aha, D. W., Kibler, D., Albert, M. K., 1991. Instance-based learning algorithms. Machine learning 6 (1), 37–66.

[3] Almuallim, H., Dietterich, T. G., 1991. Learning with many irrelevant features. In: AAAI. Vol. 91. pp. 547–552.

[4] Arauzo-Azofra, A., Benitez, J. M., Castro, J. L., 2004. A feature set measure based on relief. In: Proceedings of the fifth international conference on Recent Advances in Soft Computing. pp. 104–109.

[5] Belanche, L. A., González, F. F., 2011. Review and evaluation of feature selection algorithms in synthetic problems. arXiv preprint arXiv:1101.2320.

[6] Beretta, L., Santaniello, A., 2011. Implementing relieff filters to extract meaningful features from genetic lifetime datasets. Journal of biomedical informatics 44 (2), 361–369.

[7] Bins, J., 2000. Feature selection of huge feature sets in the context of computer vision. Computer Science. Fort Collins, CO, Colorado State University 156.

[8] Bins, J., Draper, B. A., 2001. Feature selection from huge feature sets. In: Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on. Vol. 2. IEEE, pp. 159–165.

[9] Bins, J., Draper, B. A., 2002. Evaluating feature relevance: Reducing bias in relief. In: JCIS. pp. 757–760.

[10] Blessie, E. C., Karthikeyan, E., 2011. Relief-disc: An extended relief algorithm using discretization approach for continuous features. In: Emerging Applications of Information Technology (EAIT), 2011 Second International Conference on. IEEE, pp. 161–164.

[11] Blum, A. L., Langley, P., 1997. Selection of relevant features and examples in machine learning. Artificial intelligence 97 (1), 245–271.

[12] Bolón-Canedo, V., Sánchez-Maroño, N., Alonso-Betanzos, A., 2013. A review of feature selection methods on synthetic data. Knowledge and information systems 34 (3), 483–519.

[13] Bradley, P. S., Mangasarian, O. L., 1998. Feature selection via concave minimization and support vector machines. In: ICML. Vol. 98. pp. 82–90.

[14] Breiman, L., Friedman, J., Stone, C. J., Olshen, R. A., 1984. Classification and regression trees. CRC press.

[15] Cai, H., Ng, M., 2012. Feature weighting by relief based on local hyperplane approximation. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining. Springer, pp. 335–346.

[16] Callan, J. P., Fawcett, T., Rissland, E. L., 1991. Cabot: An adaptive approach to case-based search. In: IJCAI. Vol. 12. pp. 803–808.

[17] Challita, N., Khalil, M., Beauseroy, P., 2015. New technique for feature selection: Combination between elastic net and relief. In: Technological Advances in Electrical, Electronics and Computer Engineering (TAEECE), 2015 Third International Conference on. IEEE, pp. 262–267.

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

[19] Chang, C.-C., 2010. Generalized iterative relief for supervised distance metric learning. Pattern Recognition 43 (8), 2971–2981.

[20] Chen, T., Guestrin, C., 2016. Xgboost: A scalable tree boosting system. In: Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining. ACM, pp. 785–794.

[21] Chikhi, S., Benhammada, S., 2009. Reliefmss: a variation on a feature ranking relieff algorithm. International Journal of Business Intelligence and Data Mining 4 (3-4), 375–390.

[22] Cortizo, J. C., Giraldez, I., 2006. Multi criteria wrapper improvements to naive bayes learning. In: IDEAL. Vol. 4224. Springer, pp. 419–427.

[23] Dash, M., Liu, H., 1997. Feature selection for classification. Intelligent data analysis 1 (1-4), 131–156.

[24] Dash, M., Ong, Y.-S., 2011. Relief-c: Efficient feature selection for clustering over noisy data. In: Tools with Artificial Intelligence (ICTAI), 2011 23rd IEEE International Conference on. IEEE, pp. 869–872.

[25] Dash, M., Yee, O. C., 2007. extrarelief: improving relief by efficient selection of instances. Lecture Notes in Computer Science 4830, 305.

[26] De Mántaras, R. L., 1991. A distance-based attribute selection measure for decision tree induction. Machine learning 6 (1), 81–92.

[27] Demšar, J., 2010. Algorithms for subsetting attribute values with relief. Machine learning 78 (3), 421–428.

[28] Demšar, J., Curk, T., Erjavec, A., Gorup, Č., Hočevar, T., Milutinovič, M., Možina, M., Polajnar, M., Toplak, M., Starič, A., et al., 2013. Orange: data mining toolbox in python. The Journal of Machine Learning Research 14 (1), 2349–2353.

[29] Dessì, N., Pascariello, E., Pes, B., 2013. A comparative analysis of biomarker selection techniques. BioMed research international 2013.

[30] Dorani, F., Hu, T., 2018. Feature selection for detecting gene-gene interactions in genome-wide association studies. In: International Conference on the Applications of Evolutionary Computation. Springer, pp. 33–46.

[31] Draper, B., Kaito, C., Bins, J., 2003. Iterative relief. In: Computer Vision and Pattern Recognition Workshop, 2003. CVPRW'03. Conference on. Vol. 6. IEEE, pp. 62–62.

[32] Eiras-Franco, C., Bolón-Canedo, V., Ramos, S., González-Domínguez, J., Alonso-Betanzos, A., Touriño, J., 2016. Multithreaded and spark parallelization of feature selection filters. Journal of Computational Science 17, 609–619.

[33] Eppstein, M. J., Haake, P., 2008. Very large scale relieff for genome-wide association analysis. In:

Computational Intelligence in Bioinformatics and Computational Biology, 2008. CIBCB'08. IEEE Symposium on. IEEE, pp. 112–119.

[34] Eppstein, M. J., Payne, J. L., White, B. C., Moore, J. H., 2007. Genomic mining for complex disease traits with random chemistry. Genetic Programming and Evolvable Machines 8 (4), 395–411.

[35] Flórez-López, R., 2002. Reviewing relief and its extensions: a new approach for estimating attributes considering high-correlated features. In: Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on. IEEE, pp. 605–608.

[36] Gore, S., Govindaraju, V., 2016. Feature selection using cooperative game theory and relief algorithm. In: Knowledge, Information and Creativity Support Systems: Recent Trends, Advances and Solutions. Springer, pp. 401–412.

[37] Granizo-Mackenzie, D., Moore, J. H., 2013. Multiple threshold spatially uniform relieff for the genetic analysis of complex human diseases. In: European Conference on Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics. Springer, pp. 1–10.

[38] Greene, C. S., Himmelstein, D. S., Kiralis, J., Moore, J. H., 2010. The informative extremes: using both nearest and farthest individuals can improve relief algorithms in the domain of human genetics. In: European Conference on Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics. Springer, pp. 182–193.

[39] Greene, C. S., Penrod, N. M., Kiralis, J., Moore, J. H., 2009. Spatially uniform relieff (surf) for computationally-efficient filtering of gene-gene interactions. BioData mining 2 (1), 5.

[40] Guyon, I., Bitter, H.-M., Ahmed, Z., Brown, M., Heller, J., 2003. Multivariate non-linear feature selection with kernel multiplicative updates and gram-schmidt relief. In: BISC Flint-CIBI 2003 Workshop. Berkeley. pp. 1–11.

[41] Guyon, I., Bitter, H.-M., Ahmed, Z., Brown, M., Heller, J., 2005. Multivariate non-linear feature selection with kernel methods. In: Soft computing for information processing and analysis. Springer, pp. 313–326.

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

[43] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I. H., 2009. The weka data mining software: an update. ACM SIGKDD explorations newsletter 11 (1), 10–18.

[44] Holland, J. H., 1992. Adaptation in natural and artificial systems. 1975. Ann Arbor, MI: University of Michigan Press and.

[45] Hong, S. J., 1997. Use of contextual information for feature ranking and discretization. IEEE transactions on knowledge and data engineering 9 (5), 718–730.

[46] Hunt, E. B., Marin, J., Stone, P. J., 1966. Experiments in induction. Academic Press.

[47] Ihaka, R., Gentleman, R., 1996. R: a language for data analysis and graphics. Journal of computational and graphical statistics 5 (3), 299–314.

[48] Inza, I., Larrañaga, P., Etxeberria, R., Sierra, B., 2000. Feature subset selection by bayesian network-based optimization. Artificial intelligence 123 (1-2), 157–184.

[49] Jedrzejowicz, J., Kostrzewski, R., Neumann, J., Zakrzewska, M., 2018. Imbalanced data classification using mapreduce and relief. Journal of Information and Telecommunication 2 (2), 217–230.

[50] Jin, X., Xu, A., Bie, R., Guo, P., 2006. Machine learning techniques and chi-square feature selection for cancer classification using sage gene expression profiles. In: International Workshop on Data Mining for Biomedical Applications. Springer, pp. 106–115.

[51] Jović, A., Brkić, K., Bogunović, N., 2015. A review of feature selection methods with applications. In: Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2015 38th International Convention on. IEEE, pp. 1200–1205.

[52] Kira, K., Rendell, L. A., 1992. The feature selection problem: Traditional methods and a new algorithm. In: AAAI. Vol. 2. pp. 129–134.

[53] Kira, K., Rendell, L. A., 1992. A practical approach to feature selection. In: Proceedings of the ninth international workshop on Machine learning. pp. 249–256.

[54] Kittler, J., 1978. Feature set search algorithms. Pattern recognition and signal processing.

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

[56] Koller, D., Sahami, M., 1996. Toward optimal feature selection. Tech. rep., Stanford InfoLab.

[57] Kong, D., Ding, C., Huang, H., Zhao, H., 2012. Multi-label relieff and f-statistic feature selections for image annotation. In: Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on. IEEE, pp. 2352–2359.

[58] Kononenko, I., 1994. Estimating attributes: analysis and extensions of relief. In: European conference on machine learning. Springer, pp. 171–182.

[59] Kononenko, I., 1995. On biases in estimating multi-valued attributes. In: Ijcai. Vol. 95. pp. 1034–1040.

[60] Kononenko, I., Robnik-Sikonja, M., Pompe, U., 1996. Relieff for estimation and discretization of attributes in classification, regression, and ilp problems. Artificial intelligence: methodology, systems, applications, 31–40.

[61] Kononenko, I., Šikonja, M. R., 2008. Non-myopic feature quality evaluation with (r) relieff. Computational Methods of Feature Selection, 169–191.

[62] Kononenko, I., Šimec, E., Robnik-Šikonja, M., 1997. Overcoming the myopia of inductive learning algorithms with relieff. Applied Intelligence 7 (1), 39–55.

[63] Langley, P., 1994. Selection of relevant features in machine learning. In: Proceedings of the AAAI Fall

symposium on relevance. Vol. 184. pp. 245–271.

[64] Le, T. T., Simmons, W. K., Misaki, M., Bodurka, J., White, B. C., Savitz, J., McKinney, B. A., 2017. Differential privacy-based evaporative cooling feature selection and classification with relief-f and random forests. Bioinformatics 33 (18), 2906–2913.

[65] Lee, K.-Y., Liu, P., Leung, K.-S., Wong, M.-H., 2015. Very large scale relieff algorithm on gpu for genome-wide association study. In: Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA). The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), p. 78.

[66] Li, L., 2014. Relief for regression with missing data in variable selection. Ph.D. thesis, Citeseer.

[67] Liu, X., Wang, X., Su, Q., 2015. Feature selection of medical data sets based on rs-relieff. In: Service Systems and Service Management (ICSSSM), 2015 12th International Conference on. IEEE, pp. 1–5.

[68] Martínez, A. M., Kak, A. C., 2001. Pca versus lda. IEEE transactions on pattern analysis and machine intelligence 23 (2), 228–233.

[69] McKinney, B. A., Crowe Jr, J. E., Guo, J., Tian, D., 2009. Capturing the spectrum of interaction effects in genetic association studies by simulated evaporative cooling network analysis. PLoS genetics 5 (3), e1000432.

[70] McKinney, B. A., Reif, D. M., White, B. C., Crowe, J., Moore, J. H., 2007. Evaporative cooling feature selection for genotypic data involving interactions. Bioinformatics 23 (16), 2113–2120.

[71] McKinney, B. A., White, B. C., Grill, D. E., Li, P. W., Kennedy, R. B., Poland, G. A., Oberg, A. L., 2013. Reliefseq: a gene-wise adaptive-k nearest-neighbor feature selection tool for finding gene-gene interactions and main effects in mrna-seq gene expression data. PloS one 8 (12), e81527.

[72] Menze, B. H., Kelm, B. M., Masuch, R., Himmelreich, U., Bachert, P., Petrich, W., Hamprecht, F. A., 2009. A comparison of random forest and its gini importance with standard chemometric methods for the feature selection and classification of spectral data. BMC bioinformatics 10 (1), 213.

[73] Mhamdi, F., Mhamdi, H., 2013. A new algorithm relief hybrid (hrelief) for biological motifs selection. In: Bioinformatics and Bioengineering (BIBE), 2013 IEEE 13th International Conference on. IEEE, pp. 1–5.

[74] Michalski, R. S., 1983. A theory and methodology of inductive learning. Artificial intelligence 20 (2), 111–161.

[75] Mlambo, N., Cheruiyot, W. K., Kimwele, M. W., 2016. A survey and comparative study of filter and wrapper feature selection techniques. International Journal of Engineering and Science (IJES) 5 (8), 57–67.

[76] Moore, J. H., 2015. Epistasis analysis using ReliefF. Springer.

[77] Moore, J. H., White, B. C., 2007. Tuning relieff for genome-wide genetic analysis. In: European

Conference on Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics. Springer, pp. 166–175.

[78] Narendra, P. M., Fukunaga, K., 1977. A branch and bound algorithm for feature subset selection. IEEE Transactions on Computers 9 (C-26), 917–922.

[79] Ni, W., 2012. A review and comparative study on univariate feature selection techniques. Ph.D. thesis, University of Cincinnati.

[80] Olson, R. S., La Cava, W., Orzechowski, P., Urbanowicz, R. J., Moore, J. H., 2017. Pmlb: A large benchmark suite for machine learning evaluation and comparison. arXiv preprint arXiv:1703.00512.

[81] Park, H., Kwon, H.-C., 2007. Extended relief algorithms in instance-based feature filtering. In: Advanced Language Processing and Web Information Technology, 2007. ALPIT 2007. Sixth International Conference on. IEEE, pp. 123–128.

[82] 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.

[83] Pupo, O. G. R., Morell, C., Soto, S. V., 2013. Relieff-ml: an extension of relieff algorithm to multi-label learning. In: Iberoamerican Congress on Pattern Recognition. Springer, pp. 528–535.

[84] Qamar, A. M., Gaussier, E., 2012. Relief algorithm and similarity learning for k-nn. International Journal of Computer Information Systems and Industrial Management Applications (IJCISIM) 4, 445–458.

[85] Quinlan, J. R., 1993. C4. 5: Programming for machine learning. Morgan Kauffmann 38.

[86] Reyes, O., Morell, C., Ventura, S., 2015. Scalable extensions of the relieff algorithm for weighting and selecting features on the multi-label learning context. Neurocomputing 161, 168–182.

[87] Ritchie, M. D., Hahn, L. W., Roodi, N., Bailey, L. R., Dupont, W. D., Parl, F. F., Moore, J. H., 2001. Multifactor-dimensionality reduction reveals high-order interactions among estrogen-metabolism genes in sporadic breast cancer. The American Journal of Human Genetics 69 (1), 138–147.

[88] Robnik-Šikonja, M., 2003. Experiments with cost-sensitive feature evaluation. In: European Conference on Machine Learning. Springer, pp. 325–336.

[89] Robnik-Šikonja, M., Kononenko, I., 1997. An adaptation of relief for attribute estimation in regression. In: Machine Learning: Proceedings of the Fourteenth International Conference (ICML97). pp. 296–304.

[90] Robnik-Šikonja, M., Kononenko, I., 2001. Comprehensible interpretation of reliefs estimates. In: Machine Learning: Proceedings of the Eighteenth International Conference on Machine Learning (ICML2001), Williamstown, MA, USA. San Francisco: Morgan Kaufmann. pp. 433–40.

[91] Robnik-Šikonja, M., Kononenko, I., 2003. Theoretical and empirical analysis of relieff and rrelieff. Machine learning 53 (1-2), 23–69.

[92] Robnik-Sikonja, M., Savicky, P., 2012. Corelearnclassification, regression, feature evaluation and ordinal evaluation. The R Project for Statistical Computing.

[93] Rokach, L., 2010. Ensemble-based classifiers. Artificial Intelligence Review 33 (1), 1–39.

[94] Saeys, Y., Abeel, T., Van de Peer, Y., 2008. Robust feature selection using ensemble feature selection techniques. Machine learning and knowledge discovery in databases, 313–325.

[95] Saeys, Y., Inza, I., Larrañaga, P., 2007. A review of feature selection techniques in bioinformatics. bioinformatics 23 (19), 2507–2517.

[96] Slavkov, I., Karcheska, J., Kocev, D., Kalajdziski, S., Dzeroski, S., 2013. Extending relieff for hierarchical multi-label classification. machine learning 4, 13.

[97] Smyth, P., Goodman, R. M., 1992. An information theoretic approach to rule induction from databases. IEEE transactions on Knowledge and data engineering 4 (4), 301–316.

[98] Song, Q., Ni, J., Wang, G., 2013. A fast clustering-based feature subset selection algorithm for high-dimensional data. IEEE transactions on knowledge and data engineering 25 (1), 1–14.

[99] Spolaôr, N., Cherman, E. A., Monard, M. C., Lee, H. D., 2012. Filter approach feature selection methods to support multi-label learning based on relieff and information gain. In: Advances in Artificial Intelligence-SBIA 2012. Springer, pp. 72–81.

[100] Spolaôr, N., Cherman, E. A., Monard, M. C., Lee, H. D., 2013. Relieff for multi-label feature selection. In: Intelligent Systems (BRACIS), 2013 Brazilian Conference on. IEEE, pp. 6–11.

[101] Stokes, M. E., Visweswaran, S., 2012. Application of a spatially-weighted relief algorithm for ranking genetic predictors of disease. BioData mining 5 (1), 20.

[102] Sun, Y., 2007. Iterative relief for feature weighting: algorithms, theories, and applications. IEEE transactions on pattern analysis and machine intelligence 29 (6).

[103] Sun, Y., Li, J., 2006. Iterative relief for feature weighting. In: Proceedings of the 23rd international conference on Machine learning. ACM, pp. 913–920.

[104] Sun, Y., Todorovic, S., Goodison, S., 2010. Local-learning-based feature selection for high-dimensional data analysis. IEEE transactions on pattern analysis and machine intelligence 32 (9), 1610–1626.

[105] Sun, Y., Wu, D., 2008. A relief based feature extraction algorithm. In: Proceedings of the 2008 SIAM International Conference on Data Mining. SIAM, pp. 188–195.

[106] Sutton, R. S., Matheus, C. J., 1991. Learning polynomial functions by feature construction. In: ML. pp. 208–212.

[107] Tang, J., Alelyani, S., Liu, H., 2014. Feature selection for classification: A review. Data Classification: Algorithms and Applications, 37.

[108] Tibshirani, R., 1996. Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society. Series B (Methodological), 267–288.

[109] Todorov, A., 2016. Statistical Approaches to Gene X Environment Interactions for Complex Phenotypes. MIT Press, Ch. An Overview of the RELIEF Algorithm and Advancements, pp. 95–116.

[110] Urbanowicz, R. J., Bertasius, G., Moore, J. H., 2014. An extended michigan-style learning classifier system for flexible supervised learning, classification, and data mining. In: International Conference on Parallel Problem Solving from Nature. Springer, pp. 211–221.

[111] Urbanowicz, R. J., Granizo-Mackenzie, D., Moore, J. H., 2012. Using expert knowledge to guide covering and mutation in a michigan style learning classifier system to detect epistasis and heterogeneity. In: International Conference on Parallel Problem Solving from Nature. Springer, pp. 266–275.

[112] Urbanowicz, R. J., Moore, J. H., 2015. Exstracs 2.0: description and evaluation of a scalable learning classifier system. Evolutionary intelligence 8 (2-3), 89–116.

[113] Urbanowicz, R. J., Olson, R. S., Schmitt, P., Meeker, M., Moore, J. H., 2018. Benchmarking relief-based feature selection methods for bioinformatics data mining. Journal of biomedical informatics.

[114] Van Laarhoven, P. J., Aarts, E. H., 1987. Simulated annealing. In: Simulated annealing: Theory and applications. Springer, pp. 7–15.

[115] Verma, S. S., Lucas, A., Zhang, X., Veturi, Y., Dudek, S., Li, B., Li, R., Urbanowicz, R., Moore, J. H., Kim, D., et al., 2018. Collective feature selection to identify crucial epistatic variants. BioData mining 11 (1), 5.

[116] Wolpert, D. H., Macready, W. G., 1997. No free lunch theorems for optimization. IEEE transactions on evolutionary computation 1 (1), 67–82.

[117] Yang, J., Li, Y.-P., 2006. Orthogonal relief algorithm for feature selection. In: International Conference on Intelligent Computing. Springer, pp. 227–234.

[118] Ye, K., Feenstra, K. A., Heringa, J., IJzerman, A. P., Marchiori, E., 2008. Multi-relief: a method to recognize specificity determining residues from multiple sequence alignments using a machine-learning approach for feature weighting. Bioinformatics 24 (1), 18–25.

[119] Yu, L., Liu, H., 2004. Efficient feature selection via analysis of relevance and redundancy. Journal of machine learning research 5 (Oct), 1205–1224.

[120] Zafra, A., Pechenizkiy, M., Ventura, S., 2010. Feature selection is the relieff for multiple instance learning. In: Intelligent Systems Design and Applications (ISDA), 2010 10th International Conference on. IEEE, pp. 525–532.

[121] Zafra, A., Pechenizkiy, M., Ventura, S., 2012. Relieff-mi: An extension of relieff to multiple instance learning. Neurocomputing 75 (1), 210–218.

[122] Zeng, X., Wang, Q., Zhang, C., Cai, H., 2013. Feature selection based on relieff and pca for underwater

sound classification. In: Computer Science and Network Technology (ICCSNT), 2013 3rd International Conference on. IEEE, pp. 442–445.

[123] Zhao, Z., Liu, H., 2009. Searching for interacting features in subset selection. Intelligent Data Analysis 13 (2), 207–228.

[124] Zhou, Q., Ding, J., Ning, Y., Luo, L., Li, T., 2014. Stable feature selection with ensembles of multi-relieff. In: Natural Computation (ICNC), 2014 10th International Conference on. IEEE, pp. 742–747.

[125] Zou, H., Hastie, T., 2005. Regularization and variable selection via the elastic net. Journal of the Royal Statistical Society: Series B (Statistical Methodology) 67 (2), 301–320.

- Relief-based feature selection methods (RBAs) are reviewed in detailed context
- RBAs can detect interactions without examining pairwise combinations
- Iterative RBAs have been developed to scale them up to very large feature spaces
- Research focused on core algorithms, iterative scaling, and data type flexibility