Boosted Spatially Uniform ReliefF Algorithm for Genome-Wide Genetic Analysis

Gediminas Bertasius, Delaney Granizo-Mackenzie, Ryan Urbanowicz and Jason H. Moore

Dartmouth College
1 Medical Center Dr.
Hanover, NH 03755, USA
Jason.H.Moore@Dartmouth.edu
http://www.epistasis.org

Abstract. Identifying causal factors for complex diseases is a primary goal in the field of human genetics. However, complex non-linear genotype-phenotype interactions make relevant attribute identification a very challenging problem. Relief family filtering algorithms, which utilize the nearest neighbors technique to determine relevance scores for each attribute, present an efficient and effective way to detect these genotype-phenotype interactions. Currently Multiple Threshold Spatially Uniform ReliefF (multiSURF*) is the most effective ReliefF family algorithm. In this paper, we introduce the algorithm called boostedSURF, which uses weighted distance metric to find nearest neighbors. We show that boostedSURF significantly outperforms multiSURF* in its ability to find the most predictive attributes in the simulated genetic datasets. In addition, boostedSURF also discovers new informative genes in the real datasets in comparison with the other Relief algorithms.

Keywords: Relief, Genetics, Interaction, Epistasis, Weighting, Distance, Heterogeinity

1 Introduction

With the development of new microarray technology it is now possible to measure millions of DNA sequence variations across the entire human genome [1]. Such availability of data led to an increased focus on research to develop algorithms that can successfully predict phenotype status. However, most of these studies tend to focus on linear genotype-phenotype interactions, an overly simplistic approach that results in poor predictive performance. Much of the recent research has been aimed at improving the understanding of these complex genotype-phenotype relationships [9]. One of these complex gene interactions is referred to as epistasis [10]. Epistasis describes the phenomenon, in which phenotype status is determined by the interaction between multiple genes. [10]. These complex epistatic interactions makes it difficult to analyze the effect of each gene independently because these effects are often masked by the other genes.

As a result, ordinary statistical techniques often fail to detect any meaningful patterns in the data.

Given these highly non-linear relationships, an obvious approach would be to perform an exhaustive search analyzing each possible subset of the attributes. However, an exhaustive search requires exponential running time, which would be infeasible even on datasets of moderate size. In comparison, given a dataset with n data instances and a number of attributes, ReliefF algorithms only require $O(an^2)$ time to run and yet still have the ability to successfully detect the most predictive attributes. Recently developed algorithms such as SURF* and multiSURF* have been shown to yield pretty good results at this task [2] [3].

In this paper, we introduce a new algorithm, Boosted Spatially Uniform ReliefF (boostedSURF), which further increases the power to detect the most predictive attributes in both epistatic and heterogenous environments. Unlike its predecessors, boostedSURF utilizes a weighted distance metric to find nearest/farthest neighbors, which enables boostedSURF to detect important patterns in the data with higher success rates than the previous Relief algorithms.

2 Related Work

The first ReliefF algorithm was developed in 1994 by Kononenko [5]. The basic idea behind the ReliefF algorithm is to compare data instances that are near each other. When comparing two individuals we only look for the attributes that are different in both individuals. Then, if these individuals share the same phenotype status the attribute is penalized, otherwise the attribute is rewarded. This relationship is best described by the following equation:

$$diff(a, I_1, I_2) = \begin{cases} \omega & \text{if } I_1[a] \neq I_2[a] \\ 0 & \text{otherwise} \end{cases}$$

I_1 and I_2 both denote two data instances picked for the comparison and a signifies the specific attribute at evaluation. Quantity ω has the value of 1 in the case where I_1 share the same phenotype status as I_2 and -1 otherwise:

$$\omega = \begin{cases} 1 & \text{if } I_1 == I_2 \\ -1 & \text{otherwise} \end{cases}$$

Following the first ReliefF algorithm, Kononenko worked on this topic further and developed several minor extensions to the original ReliefF algorithm, which are described in more detail in [7]. In 2007, Moore et al. improved ReliefF by adding sequential worst attribute removal to the algorithm [6]. In 2009, Greene et al. developed the SURF algorithm, which improved upon ReliefF by examining nearest neighbors that are below a certain distance threshold T rather than picking a constant number of neighbors like what its predecessor ReliefF did [4]. In 2010, Greene et al. designed SURF*, an improvement upon SURF, which introduced comparisons of the farthest as well as the nearest neighbors [3]. Later in 2012, Stokes et al. improved upon SURF* by developing SWRF* algorithm, which used sigmoid function to assign scores to the attributes [8]. The most recent improvement was Granizo-Mackenzie's et al. algorithm multiSURF* [2]. multiSURF* increased the power to detect predictive attributes and slashed the running time of SURF* by introducing unique distance thresholds for each data instance and examining only top $\approx 31\%$ nearest/farthest neighbors in the dataset [2].

3 Datasets

3.1 Epistatic Data

To evaluate the relative performance of boostedSURF, we used identical datasets to those used to test multiSURF*. These datasets contained 1000 attributes, out if which 2 were predictive, and the remaining were random noise. Heritabilities of the datasets included $0.025, 0.05, 0.1, 0.2$ and 0.4. We ran our tests on datasets containing $800, 1600$ and 3200 data instances. More detailed description of the data may be found in [2].

3.2 Real Data

FILL IN

4 Methods

4.1 Notation

Let I_i denote the i^{th} data instance from the dataset. Also, let $d(I_i, I_j)$ denote the Hamming distance between I_i and I_j, which will be computed using weighted distance metric denoted as D, where D refers to the array of distance weights for each attribute a. $S[i]$ refers to the phenotype status of an individual I_i, which can be either case or control. Furthermore, let Φ refer to the parameter that controls the frequency of updating boostedSURF's distance metric D. Finally, let $ScoreWeights$ denote the array where the score weights for each attribute a will be stored.

4.2 High Level Description of boostedSURF Algorithm

boostedSURF algorithm builds off its most recent predecessor multiSURF* [2]. Similarly to multiSURF*, we use a unique distance threshold T for each individual. At each step, we also compute σ_i, a standard deviation of all the distances between individual I_i and the rest of the individuals in the dataset. Then we use σ_i to reduce the number of nearest/farthest data instances that require comparisons, which speeds up the algorithm significantly. Just like in all of the previous Relief algorithms, when comparing two individuals we only consider cases when the value of a specific attribute between the individuals is different. Then, if the phenotype status of these two individuals is the same we reward the attribute, and penalize the attribute if the status is different. The procedure is exactly the opposite when considering the farthest neighbors: we reward the attribute if the phenotype status of two individuals is different, and penalize it if the phenotype status is the same.

The major difference between multiSURF* and boostedSURF is the distance metric used to calculate nearest/farthest neighbors. Whereas multiSURF* uses uniform distance metric to compute the neighbors, boostedSURF assigns the distance weight for each attribute based on its predictive importance at the time of the evaluation.

4.3 Maximizing the Objective Function

In this section, we explain the mathematical rationale why boostedSURF algorithm is more effective than its predecessors. In the SURF* paper, Greene et al. [4] defines an objective function, which needs to be maximized for the algorithm's optimal performance. We can design an objective function for any number of predictive attributes. However, for the sake of simplicity let's assume that there are only two predictive attributes in the dataset and the rest are random noise.

Then, if we consider random data instance I_i each of its neighbors will fall into one of the six categories: $H_{0\Delta}$,$H_{1\Delta}$,$H_{2\Delta}$,$M_{0\Delta}$,$M_{1\Delta}$,$M_{2\Delta}$ where (H, M) signify if two individuals have the same or different phenotype respectively (Hits or Misses) and the numerical subscript denotes how many predictive attributes do not match across the individuals. For example, subscript ${}_{0}\Delta$ would mean that both predictive attributes have the same values across the two individuals, while ${}_{2}\Delta$ would mean that both of the predictive attributes are different. Then, we can define our two objective functions as follows (Note: two objective functions are necessary because we are considering both nearest and farthest neighbors):

$$S_i^N = \underbrace{\frac{1}{2}(|NM_{1\Delta}| - |NH_{1\Delta}|)}_{NQ_{1\Delta}} - \underbrace{(|NH_{2\Delta}| - |NM_{2\Delta}|)}_{NQ_{2\Delta}}$$

$$S_i^F = \underbrace{-\frac{1}{2}(|FM_{1\Delta}| - |FH_{1\Delta}|)}_{FQ_{1\Delta}} + \underbrace{(|FH_{2\Delta}| - |FM_{2\Delta}|)}_{FQ_{2\Delta}}$$

From the equations above, it is clear that in both cases (nearest and farthest) we would like to maximize quantities $NQ_{1\Delta} - NQ_{2\Delta}$ and $FQ_{2\Delta} - FQ_{1\Delta}$. To make things more simplistic, let us only focus on the nearest neighbor case since the farthest neighbor case is based on identical principles. Previous Relief family algorithms rely on the fact that $NQ_{1\Delta} \geq NQ_{2\Delta}$. This is true because individuals belonging to $NQ_{1\Delta}$ will have distance that is on average 1 unit smaller than distances of the individuals in $NQ_{2\Delta}$. Therefore, when considering closest neighbors, quantity $NQ_{1\Delta}$ will likely have more individuals than $NQ_{2\Delta}$ thus, resulting in a positive value of the objective function.

boostedSURF uses this fact to make the quantity $NQ_{1\Delta} - NQ_{2\Delta}$ even higher. In the previous Relief algorithms, the average distance between individuals in $NQ_{1\Delta}$ and in $NQ_{2\Delta}$ is 1, boostedSURF employs weighted distance metric to maximize this distance and amplify the distinction between $NQ_{1\Delta}$ and $NQ_{2\Delta}$. This strategy results in an increased value of the objective function, which in turn boosts the predictive power of the algorithm.

4.4 Employing Weighted Distance Metric

It is clear that not all distance directions in the data will be equally informative. For instance, the distance direction associated with the attribute that has no predictive power will definitely be much less informative than the distance direction associated with the predictive attribute. Therefore, when calculating the distance between two individuals it would make sense to put more weight on the attributes that have some predictive power. There is clearly no way to know which attributes are relevant before running the algorithm. However, with each pass through the data, the weight scores for the attributes become more informative regarding their relevance in the dataset. Since our weighted distance metric relies on individual attributes' relevance, we propose to use these weight scores to construct weighted distance metric D. We use them in a following way:

$$D[a] = \max(ScoreWeights[a], 1)$$

Where $D[a]$ denotes the distance metric weight for an attribute a and $ScoreWeights[a]$ represents score weight of the attribute a at the time of the evaluation. The max function is introduced to ensure that no distance weight has a negative value and that each attribute is still informative to some degree (not equal to 0).

Then, distances between each individual are computed using the following scheme:

$$d(I_i, I_j) = \sum_{\forall a} \delta_a$$

where $d(I_i, I_j)$ denotes the distance between the individuals I_i, I_j, a refers to the set of all attributes, D signifies the distance metric described above, and δ_a is defined as follows:

$$\delta_a = \begin{cases} D[a] & \text{if } I_i[a] \neq I_j[a] \\ 0 & \text{otherwise} \end{cases}$$

4.5 Parameter Phi

Parameter Φ controls the frequency of recalculating the distance metric D. D will be recomputed every Φ iterations inside the algorithm. Whereas lower values of Φ will result in greater predictive power for the algorithm, higher Φ values will reduce the run time of the algorithm. Therefore, choosing the value of Φ depends on the user's preference regarding the trade-off between algorithm's predictive power and its running time. In all of our tests, we used $\Phi = 1$, which means that the distance metric D was updated after each iteration. Detailed analysis on how the choice of parameter Φ affects boostedSURF's running time is presented in 4.7

4.6 Pseudo-Code

A pseudo-code description of the algorithm can be found in Algorithm 1.

4.7 Run Time

As derived in [2], multiSURF*'s running time is estimated as $(c_1 a + 0.62 c_2 a)n^2$ where c_1, c_2 are system dependent constants, and a, n are the number of attributes and the number of data instances in the dataset respectively. Essentially, the only difference between multiSURF* and boostedSURF is that boostedSURF has to update its distance metric D at a user specified frequency rate. An update for distance metric D requires looping through each attribute, which requires $c_3 a$ time. As mentioned earlier, the frequency of updating D is determined by the parameter Φ. As a result, the procedure of updating the distance metric throughout the lifetime of an algorithm takes $c_3 \frac{n}{\Phi} a$ time. Therefore, the total running time of boostedSURF is $(c_1 a + 0.62 c_2 a)n^2 + c_3 \frac{n}{\Phi} a$. Analyzing this expression, it is clear that the only difference in the running times between multiSURF* and boostedSURF is the term $c_3 \frac{n}{\Phi} a$. Since $c_3 \frac{n}{\Phi} a$ is a lower order term in comparison to the remaining terms in the expression, using asymptotic approximation we could conclude that boostedSURF should take the same amount of time to run as multiSURF* does. In reality, though, there is a slight difference between the running times of the two algorithms. However, as our results in 5.3 illustrate, the time difference is almost negligible. Therefore, an enhanced predictive power of boostedSURF and almost identical runtime in comparison to multiSURF* makes boostedSURF an attractive alternative to the previous Relief algorithms.

4.8 Experimental Design

Epistatic Data To measure the success rate of boostedSURF on epistatic datasets we compared attribute rankings generated by boostedSURF and multiSURF*. We ran both algorithms on 500 datasets for each level of heritability, and for each dataset with different number of samples. We then counted the frequencies that predictive attributes were ranked in top $\%x$ percentile. These results are presented in section 5.1

Real Data To measure boostedSURF's performance on the real datasets we examined attribute scores returned by several Relief algorithms including boostedSURF. To illustrate differences in the performances we first produced normalized attribute scores distributions for each algorithm. In addition, we also computed correlations between the rankings produced by each algorithm. These results can be found in section 5.2

```
Data: Input dataset data, parameter $\Phi$
Result: Array of score weights $ScoreWeights$
for each individual $i$ do
    if $i \bmod \Phi == 0$ then
        for each attribute $a$ do
            $D[a] = \max(ScoreWeights[a], 1)$;
        end
    end
    compute all distances between $I_i$ and every other individual using weighted
    distance metric $D$;
    set $T_i$ to be the mean of all these distances;
    set $\sigma$ to be the standard deviation of the distances;
    for each individual $j$ do
        if $d(I_i, I_j) < T_I - \sigma/2$ then
            for each attribute $a$ do
                if $I_i[a]! = I_j[a]$ then
                    if $S[i] == S[j]$ then
                        $ScoreWeights[a] - -$;
                    else
                        $ScoreWeights[a] + +$;
                    end
                end
            end
        end
        if $d(I_i, I_j) > T_I + \sigma/2$ then
            for each attribute $a$ do
                if $I_i[a]! = I_j[a]$ then
                    if $S[i] == S[j]$ then
                        $ScoreWeights[a] + +$;
                    else
                        $ScoreWeights[a] - -$;
                    end
                end
            end
        end
    end
    return $ScoreWeights$;
end
```

Algorithm 1: boostedSURF

5 Results

5.1 Success Rate on Epistatic Datasets

The results on epistatic datasets are presented in Figure 1. The figures below prove that boostedSURF significantly outperforms multiSURF in ranking the predictive attributes in the top $0-5\%$ percentiles. This feature of boostedSURF makes the algorithm an attractive alternative to other ReliefF algorithms.

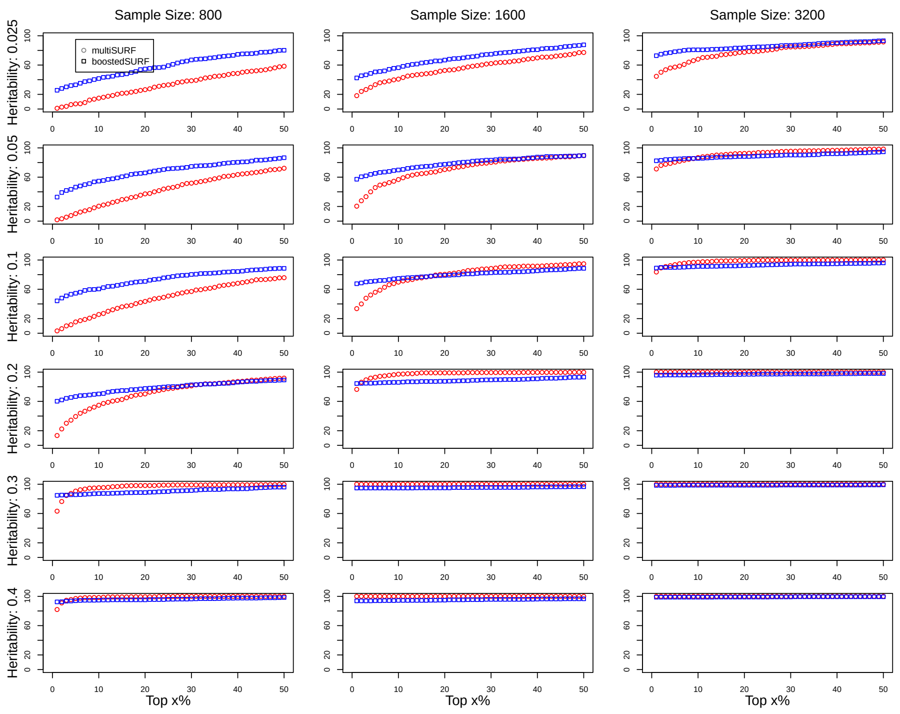

Fig. 1: Success rates of boostedSURF vs. multiSURF* on Epistatic Datasets

5.2 Results on Real Data

The results below confirm that running boostedSURF on real data produces different and more useful gene rankings in comparison to other Relief algorithms. In the previous sections, we already established that boostedSURF is more accurate than the other Relief algorithms. Therefore, given boostedSURF's enhanced predictive performance it would follow that the new information obtained from real data should be more accurate as well.

The distributions of the scores assigned to each gene are presented in Figure 2. From the distributions, it is clear that gene rankings produced by boostedSURF greatly differ from the rankings produced by other Relief algorithms. Whereas multiSURF* produces almost perfectly normal distribution of scores, boostedSURF's scores distribution is more heavily skewed towards the higher scores. This difference arises from boostedSURF's use of weighted distance metric. As explained in the earlier sections, boostedSURF's use of weighted distance metric attempts to amplify the distinction between truly informative genes and the noise. Therefore, boostedSURF tends to focus more heavily on more important genes thus, producing a skewed distribution of scores.

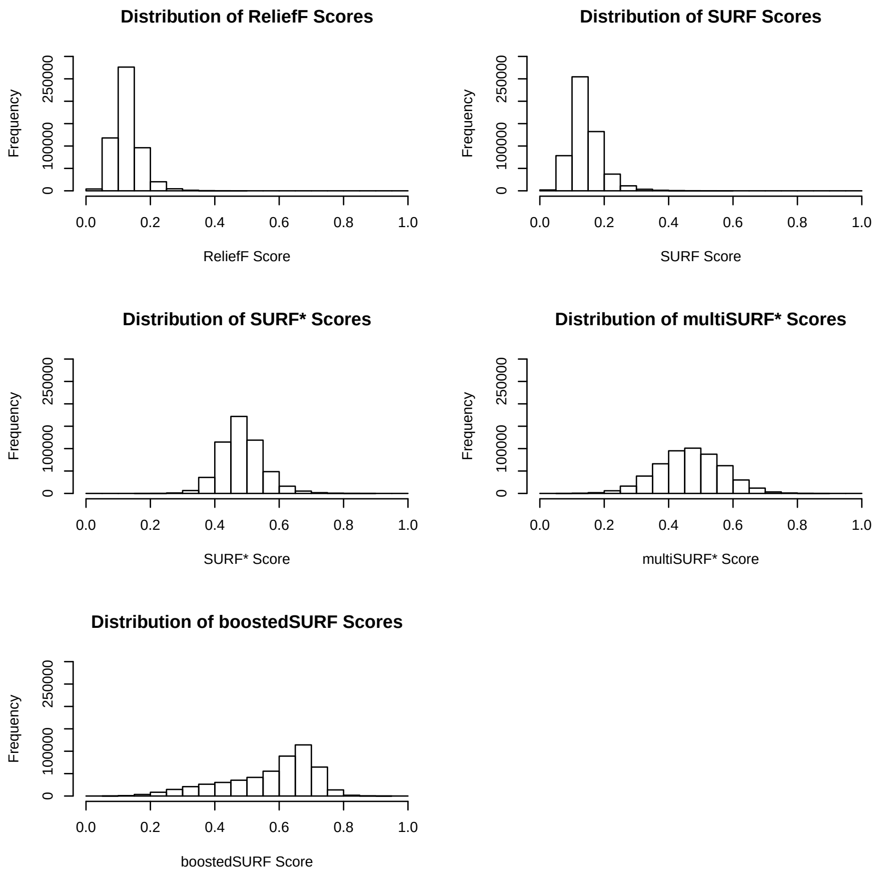

Fig. 2: Distributions of rankings of all Relief algorithms

In addition, to support our claim regarding the novelty of boostedSURF we also present results that illustrate correlations between the rankings produced by boostedSURF versus other Relief algorithms. The results in their numerical and visual form are presented in Table 1,and Figure **??** respectively. Results in the table, suggest that the correlation is highest between boostedSURF's and multiSURF*'s rankings. This makes intuitive sense because boostedSURF and multiSURF* are the two most similar algorithms in this group. However, analysis of these results on the absolute scale, makes it clear that these correlations are too low to suggest any significant similarity between the rankings produced by boostedSURF versus the rankings of other algorithms. This weak correlation, therefore, supports our previous claim that running boostedSURF on the real data provides new and more useful information about the genes importance.

	Parametric	Non-Parametric
ReliefF vs. boostedSURF	0.01577732	-0.007372329
SURF vs boostedSURF	0.009809878	0.002003963
SURF* vs boostedSURF	-0.1507351	-0.145299
multiSURF* vs boostedSURF	-0.2593137	-0.2502765

Table 1: Comparing correlations between the rankings of boostedSURF and other Relief algorithms

5.3 Runtime

The results for the runtimes of boostedSURF and multiSURF* can be found in Table 2. As illustrated in the results table, boostedSURF takes only $\approx 12\%$ more time than multiSURF* on average. The increased running time of boosted-SURF comes from the need to update the weighted distance metric D. However, our tests were done with parameter $\Phi = 1$ (slowest possible case). Therefore, boostedSURF's running time can be improved by increasing the value of parameter Φ.

We ran both algorithms on 100 datasets containing 1600 data instances and 1000 attributes. It is important to note that empirical runtime results in this paper cannot be directly compared with the results in [2]. These differences result from different implementations, different programming languages used to implement the algorithms, and different hardware used to run the tests. In this paper, both algorithms were implemented in C. Also, identical machines were used for the runs of both algorithms. Therefore, the comparison between the running times of two algorithms presented in Table 2 is fair.

6 Discussion and Conclusions

As discussed in section 4, an increase in boostedSURF's power comes mainly from using the weighted distance metric. When using the uniform distance metric to calculate the distances, an implicit assumption is made that every distance

	Mean Runtime (s)	Standard Deviation (s)
boostedSURF	57.427	3.875
multiSURF*	51.238	1.234

Table 2: Comparing the runtimes of boostedSURF and multiSURF*.

direction is equally informative. In noisy datasets with a lot of attributes, this is a completely unreasonable assumption, which hampers the predictive power of the algorithm. Our proposed scheme– to use the weighted distance metric instead of a uniform one– at least partially fixes this issue as illustrated by the enhanced performance of boostedSURF.

One of the fundamental problems with most of the filter algorithms is the lack of strictly defined objective function. Having no knowledge of what quantity needs to be maximized/minimized makes it very difficult to optimize the performance of the algorithm. This is definitely the area that should be explored more in future research. In the case of ReliefF family algorithms, the formulation of an alternative objective function may help to discover different ways on improving an algorithm's predictive power and also to deepen the mathematical intuition behind the success of ReliefF algorithms.

In the specific case of the boostedSURF algorithm, alternative ways to maximize the objective function should be explored as well. Furthermore, to achieve the optimal balance between boostedSURF's speed and predictive power, optimization techniques for the parameter Φ should also be examined. These techniques could improve boostedSURF's running time on large datasets, which would be highly beneficial.

Overall, boostedSURF is an attractive choice for genome-wide analysis. Enhanced predictive power of boostedSURF and no significant increase in running time enables more effective detection of significant patterns in the data.

References

1. Leo G. Mendoza Mark S. Chee Nat Genet Frank J. Steemers, Grace Lee. A genome-wide scalable snp genotyping assay using microarray technology. *Nat. Genet.*, 37(5):549–554, May 2005.
2. Delaney Granizo-MacKenzie and Jason H. Moore. Multiple threshold spatially uniform relieff for the genetic analysis of complex human diseases. pages 1–10, 2013.
3. C.S. Greene, D.S. Himmelstein, J. Kiralis, and J.H. Moore. The informative extremes: Using both nearest and farthest individuals can improve relief algorithms in the domain of human genetics. *LNCS*, 6023:182–193, 2010.
4. C.S. Greene, N.M. Penrod, J. Kiralis, and J.H. Moore. Spatially uniform relieff (surf) for computationally-efficient filtering of gene-gene interactions. *BioData Mining*, 2, 2009.
5. I. Kononenko. Estimating attributes: Analysis and extensions of relief. *LNCS*, 784:171–182, 1994.

6. Jason H. Moore and Bill C. White. Tuning relieff for genome-wide genetic analysis. In *Proceedings of the 5th European conference on Evolutionary computation, machine learning and data mining in bioinformatics*, EvoBIO'07, pages 166–175, Berlin, Heidelberg, 2007. Springer-Verlag.
7. M. Robnik-Sikonja and I. Kononenko. An adaptation of relief for attribute estimation in regression. *ICML*, pages 296–304, 1997.
8. MatthewE Stokes and Shyam Visweswaran. Application of a spatially-weighted relief algorithm for ranking genetic predictors of disease. *BioData Mining*, 5(1):1–11, 2012.
9. Anna L. Tyler, Folkert W. Asselbergs, Scott M. Williams, and Jason H. Moore. Shadows of complexity: what biological networks reveal about epistasis and pleiotropy. *BioEssays*, 31(2):220–227, 2009.
10. Ryan J. Urbanowicz, Ambrose Granizo-Mackenzie, and Jason H. Moore. An analysis pipeline with statistical and visualization-guided knowledge discovery for michigan-style learning classifier systems. *IEEE Comp. Int. Mag.*, 7(4):35–45, 2012.