Multiple Threshold Spatially Uniform ReliefF for the Genetic Analysis of Complex Human Diseases

Delaney Granizo-Mackenzie and Jason H. Moore

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

Abstract. Detecting genetic interactions without running an exhaustive search is a difficult problem. We present a new heuristic, multiSURF*, which can detect these interactions with high accuracy and in time linear in the number of genes. Our algorithm is an improvement over the SURF* algorithm, which detects genetic signals by comparing individuals close to, and far from, one another and noticing whether differences correlate with different disease statuses. Our improvement consistently outperforms SURF* while providing a large runtime decrease by examining only individuals very near and very far from one another. Additionally we perform an analysis on real data and show that our method provides new information. We conclude that multiSURF* is a better alternative to SURF* in both power and runtime.

Keywords: Relief, Genetics, Interaction, Epistasis, Heuristic, Weighting, Real Data

1 Introduction

A perpetual problem in modern genomics is dealing with massive quantities of data. Many geneticists are trying to develop phenotype prediction algorithms for use in clinical practice. However, few such tests have been successfully found. The inherent problem is that most Genome Wide Association Studies (GWAS) so far have focused on linear genotype-to-phenotype association. Whereas there are some phenotypes with strong associations to single gene main effects, the vast majority of possible associations are between multiple genes and one phenotype. Given a phenotype there are only n possible main effects, one for each gene; there are, however, $2^n - n$ possible higher-order signals that are comprised of a set of genes non-linearly correlated with the phenotype. There is an increasing amount of evidence that most genetic associations are in fact high-order and non-additive. [8,1,7].

Given that we would like to find these associations, it then seems that we must venture into combinatorial space. One common way to search for interacting genes is to use the exhaustive search known as Multifactor Dimensionality

Reduction (MDR) [6]. Multifactor Dimensionality Reduction works by exhaustively analyzing all possible models of up to n attributes – commonly Single Nucleotide Polymorphisms (SNPs).[1] Searching for two-way or pair-wise interactions takes at least $O(a^2n)$ time in a dataset with a attribute SNPs and n individuals. This is completely unfeasible on a realistically sized genetic dataset with $a >> 10^5$, given modern computing technology and pace of development. Therefore any polynomial-time heuristic that can help find gene interactions is very valuable. One standard approach for finding interacting genes is known as 'two-pass' and involves filtering the dataset by eliminating genes that are considered unimportant by a heuristic. Once the filtering has occurred, the second pass does an exhaustive all-subsets analysis on the much smaller and more manageable new dataset [10]. One of the most successful filters is the SURF* algorithm, developed by Greene et al. in 2010 [2]. We introduce a new improvement, Multiple Threshold SURF (multiSURF*) that further increases the power and vastly decreases the runtime. This will be very valuable for genetics studies, as our new algorithm can process much more data and find more interesting genes in the same time as would be previous necessary for SURF*. This best of both worlds improvement is highly desirable in computer science; SURF* improved upon SURF, but takes much more time. Often amplification algorithms can be used to provide better results given more time, but we provide better results in less time.

1.1 Related Work

In 1994, Kononenko developed ReliefF [5]. Based on previous work by Kira et al. [4], ReliefF proved a novel way of approaching signal detection in binary outcome datasets. The principle underlying the ReliefF algorithm is simple, pick individuals similar to one another and then check if the disease status changes. If it does, reward the attributes that are different and if not reward the ones that are the same. Greene et al. improved upon ReliefF with Spatially Uniform ReliefF (SURF) in 2009 and then again with SURF* in 2010 [3,2]. SURF differs from ReliefF by using all neighbors below a threshold and SURF* uses all neighbors, but splits them into near and far. The attraction of SURF* is that it detects higher-order genetic interactions with no main effects much more often than SURF, even in large datasets. SURF* also takes only $O(an^2)$ time. These two properties combined enable the filtering of huge genetic datasets into ones small enough for an exhaustive search. [2]. We introduce an algorithm that can outperform even SURF*'s power, and also slashes SURF*'s running time. This enhances SURF*'s two attractive properties and makes our alternative better suited for use as a filter in two-pass genetic studies.

[1] SNPs are single letter differences in DNA, each version – or mutation – being called an allele. Since there are two alleles, the original, and the polymorphism with one nucleotide different, there are 4 possible combinations in diploid DNA. Two of these combinations are identical as order is unimportant and we are left with three possibilities for any SNP measurement.

2 Methods

2.1 Simulated Data

Our data were identical to those used to benchmark SURF* and SURF; hence we were able to provide a fair comparison [2].

We used simulated data with each dataset containing one two-way epistatic model and 998 random noise SNPs [11]. Our datasets were balanced and each individual had 1000 SNP measurements with only two as part of a non-linear association with the binary disease status. There was no association between either model SNP and the disease status, but instead a non-linear association involving both SNPs together. Each SNP had a minor and major allele with occurrence frequencies of 0.6 and 0.4, respectively. To generate a dataset we need a penetrance function to determine genotype phenotype relationship. We used 30 penetrance functions that met the requirements of having no main effects and the proper minor allele frequencies[2]. Each dataset also had a heritability. This is the probability that a genetic effect will carry through to a phenotype. A heritability of 1.0 means a perfect association between genotype and phenotype, whereas 0.0 is no association. Our datasets were generated with heritabilities of 0.025, 0.05, 0.1, 0.2, 0.3, and 0.4; we used 5 penetrance functions for each. For each of these 30 models we used 100 random variants of the dataset generated using the same penetrance function and heritability. Finally, datasets with duplicate genetic models were generated with 800, 1600, and 3200 individuals.

2.2 Real Data

This study population consists of nationally available genetic data from 2,286 men of European-descent (488 non-aggressive and 687 aggressive cases, 1,111 controls) collected through the Prostate, Lung, Colon, and Ovarian (PLCO) Cancer Screening Trial., a randomized, well-designed, multi-center investigation sponsored and coordinated by the National Cancer Institute (NCI) and their Cancer Genetic Markers of Susceptibility (CGEMS) program. We focused here on prostate cancer aggressiveness as the endpoint. Between 1993 and 2001, the PLCO Trial recruited men ages 55-74 years to evaluate the effect of screening on disease specific mortality, relative to standard care. All participants signed informed consent documents approved by both the NCI and local institutional review boards. Access to clinical and background data collected through examinations and questionnaires was approved for use by the PLCO. Men were included in the current analysis if they had a baseline PSA measurement before October 1, 2003, completed a baseline questionnaire, returned at least one Annual Study Update (ASU), and had available SNP profile data through the CGEMS data portal (`http://cgems.cancer.gov/`). We used a biological filter to reduce the set of genes to just those involved in apoptosis (programmed cell

[2] This and the experimental design section are synopses of our experiment, which was identical to that from both the SURF and SURF* papers. For another description of the experiment and further explanation of the design, please see references [3,2]

death), DNA repair and antioxidation/carcinogen metabolism. These biological processes are hypothesized to play an important role in prostate cancer. A total of 219 SNPs in these genes were studied here.

2.3 multiSURF* Implementation

Notation We denote the i^{th} individual from the dataset as I_i; $d(I_i, I_j)$ refers to the hamming distance between two individuals I_i and I_j. The hamming distance is calculated by counting the number of unequal SNP measurements between the two individuals. For instance, an individual with measurements 01001 will have a hamming distance of 2 from an individual with 01010. $s[i]$ refers to the disease status – case or control – of individual I_i. We denote the number of individuals with n and the number of SNPs – or attributes – with a. We denote the standard deviation of a set/vector X as σ_X.

Algorithm The SURF* algorithm functions by making comparisons between individuals. We first define a weight for each SNP which is initialized to 0. The type of comparison depends on whether an individual is near or far from another. We start by computing the mean of all distances, which we call the threshold, T. Then for each possible non-ordered pairing of individuals I_i, I_j we check if $d(I_i, I_j) < T$. If so we treat individual I_j as near to I_i, and in the opposite case, $d(I_i, I_j) > T$, we treat it as far from I_i. If the two individuals are near to one another we check all SNPs that have different values across both individuals. For each of these we increment the weight if the status is different and decrement if the status is the same. If the two individuals are far from one another we check all SNPs that have the same value across both individuals. For each of these we increment the weight if the status is different and decrement if the status is the same.

The authors note that the original SURF* paper described the algorithm as examining SNPs that were different across far individuals. We implement the algorithm by examining SNPs that are the same and reversing the signs on increment and decrement. This does not affect the monotonicity of the final scores of the SNPs, and is slightly faster.

We change the SURF* algorithm in two distinct ways. First of all we give each individual its own threshold rather than having one global threshold. Second we more strictly define near and far. For more reference to the original SURF* algorithm please see the corresponding paper [2].

Multiple Thresholds In the SURF* algorithm one global distance threshold T is initially computed. Any individual I_j closer than T to individual I_i is then classified as near and any individual I_j further than T from individual I_i is classified as far. Instead of this global threshold, T, we compute one threshold, T_i, for each individual I_i. In SURF*, T is computed as the mean distance between all $(n^2 - n)/2$ individual pairings I_i, I_j. In multiSURF* we compute each T_i to be the mean of all distances $d(I_i, I_j)$ between the fixed individual I_i and all other individuals I_j. Formally:

$$T_i = \frac{\sum_j d(I_i, I_j)}{n-1}$$

.

Near and Far In the SURF* algorithm each individual I_j is treated as being either near to, or far from, another individual I_i. This is determined by whether $d(I_i, I_j) < T$. This means that a comparison is made between every pair of individuals. In multiSURF* we have stricter definitions of near and far and therefore make fewer comparisons. While computing T_i we compute the distances between I_i and every other individual and save this as a vector, V. We take the standard deviation of these distances, σ_V and set a new dead-band variable D_i equal to $\sigma_V/2$. We then say that an individual I_j is near to I_i if and only if $d(I_i, I_j) < T_i - D_i$. Similarly, an individual is far if and only if $d(I_i, I_j) > T_i + D_i$. If we assume a normal distribution of distances this means that $\Phi(\sigma/2) \approx 31\%$ of individuals I_j will be classified as near and likewise for far. This reduces the runtime by a large factor, as multiSURF* makes approximately 62% of the comparisons that SURF* does.

Pseudo-Code A pseudo-code description of the algorithm can be found in Algorithm 1. We implemented the algorithm in Java. For full source code, please contact the authors at the given email.

Running Time SURF* first computes all distances, this takes $c_1 n^2 a$ time for some system-dependent c_1. The next round of computation to update the weights runs over all pairs of instances and takes $c_2 n^2 a$ time for another system-dependent constant. SURF therefore takes a total of $(c_1 + c_2)n^2 a$ time. multiSURF* computes all distances, so again we take $c_1 n^2 a$ time. In addition we compute a D_i for each individual I_i. The required variance computation is linear and takes $c_3 n^2$ in total. Next we perform an instance comparison if and only if the distance between the two satisfies our dead-band requirement. We assume a normal distribution of distances between a given instance I_i and all others. To find the number of instances that will be near or far is the same as computing a two-tailed p-value. Using the cumulative distribution function we compute

$$2(1 - \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{1/2} e^{\frac{-x^2}{2}} \, \mathrm{d}x) \approx 0.62$$

Therefore we will perform about $0.62n^2$ comparisons, each taking the same time $c_2 a$. This yields a total runtime of

$$c_1 n^2 a + c_3 n^2 + 0.62 c_2 n^2 a =$$

$$(c_1 a + 0.62 c_2 a + c_3) n^2 =$$

```
Data: Input dataset data
Result: Array of weights w
set $d[i][j] = d(I_i, I_j)$ for all i and all j;
for each $i$ do
    compute all distances between $I_i$ and every other indiviudal;
    set $T_i$ to be the mean of all these distances;
    set $\sigma$ to be the standard deviation of the distances;
    set $D_i$ to be $\sigma/2$;
end
for each $i$ do
    for each $j$ do
        if $d(I_i, I_j) < T_I - D_i$ then
            for each $a$ do
                if $I_i[a]! = I_j[a]$ then
                    if $s[i] == s[j]$ then
                        $w[a] - -$;
                    else
                        $w[a] + +$;
                    end
                end
            end
        end
        if $d(I_i, I_j) > T_I + D_i$ then
            for each $a$ do
                if $I_i[a] == I_j[a]$ then
                    if $s[i] == s[j]$ then
                        $w[a] - -$;
                    else
                        $w[a] + +$;
                    end
                end
            end
        end
    end
    return w;
end
```

Algorithm 1: multiSURF*

For large values of a we can obtain the very good approximation

$$(c_1 a + 0.62 c_2 a + c_3)n^2 < (c_1 a + 0.62 c_2 (a + c_4))n^2 \approx$$
$$(c_1 a + 0.62 c_2 a)n^2$$

Compared to SURF*'s

$$(c_1 a + c_2 a)n^2$$

Therefore multiSURF* will outperform SURF* by some system dependent constant for large values of a. To test how the two performed on a modern computer we ran multiSURF* on a sampling of the previously defined datasets. The machine was an Intel Core 2 Duo P8600 2.4GHz with 4GB of RAM. We arbitrarily selected 10 datasets with 1600 instances and a heritability of 0.05 and ran 10 times on each dataset for a total of 100 runs. The average runtime and standard deviation for each method is detailed in Table 1.

2.4 Experimental Design

To measure success we examined the rankings returned by multiSURF* and SURF* for each dataset. For each ranking we chose the minimum score of the two model SNPs and found the percentage of SNPs ranked higher. We then counted the number of times that the model SNPs were ranked in the top $x\%$ over the $\frac{\text{30 genetic models}}{\text{6 heritabilities}} \times 100\ \text{variants} = 500$ datasets for each heritability and number of individuals. These results are displayed in Figure 1.

To test if the observed differences were significant we counted the number of times that multiSURF* ranked the model SNPs higher than 95% of all other SNPs over all 9000 datasets. Doing the same for SURF* gave us a contingency table. We used a Fisher's exact test on these counts to obtain a probability that the observed difference was due to chance; the p-value can be found in the results section along with a discussion of the graphical results [9].

2.5 Real Data Analysis

To further validate multiSURF* we performed a comparative analysis on the dataset described in Section 2.2.

We ran several Relief-family methods on the dataset: ReliefF with 350 neighbors, SURF, SURF*, and multiSURF*. We then ranked SNPs by the weight assigned to them. We wished to determine which genes each algorithm treats as significantly more important than others, so we performed a Fisher's exact test. We found the corresponding gene for each SNP and then counted the number of times that gene was represented by a SNP ranked in the top 10%. These counts gave us a p-value for each SNP via the Fisher's exact test. It should be noted that the p-value is not corresponding to any actual model for the dataset, simply whether the genes were being ranked higher by the algorithm by chance. We then selected all genes that had achieved a significant p-value for any Relief method and listed them in a table. The table can be found in Table 2 and shows which genes are being considered 'important' by which algorithm.

3 Results

3.1 Success Rate

The results are presented in Figure 1. The difference between the two methods was found to be highly significant ($p < 10^{-15}$). We found that multiSURF* outperformed SURF* on all sample sizes and heritabilities. The exceptions are the cases in which SURF* already achieves a 100% success rate on all percentiles. The improvements were consistent across all the sample sizes and heritabilities.

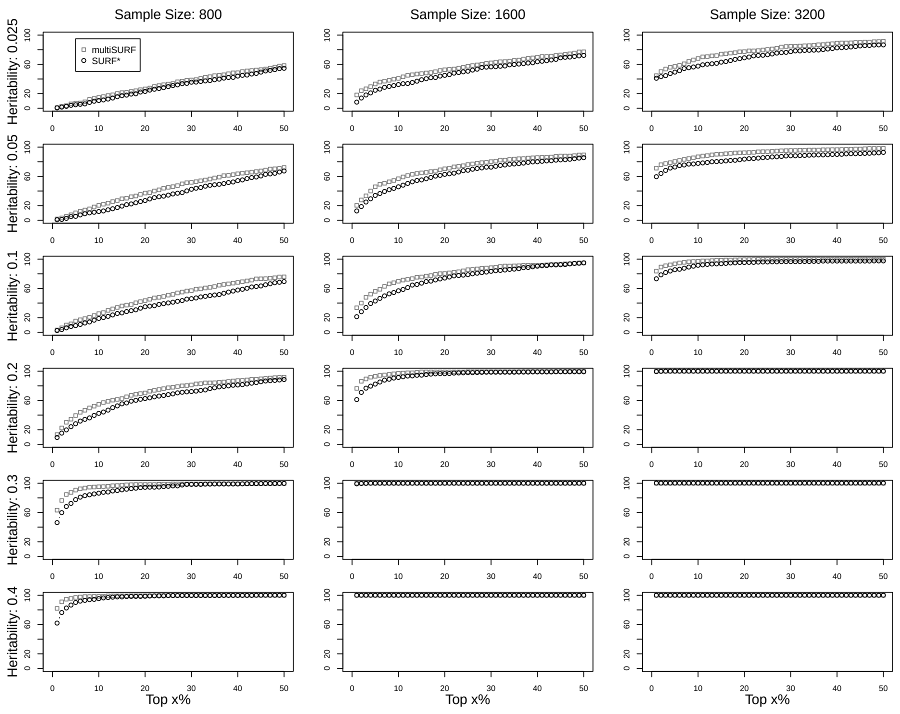

Fig. 1. Success rate of multiSURF* vs. SURF*

3.2 Runtime

The runtimes can be found in Table 1. On average, multiSURF* runs in only 67.8% of the time of SURF*. The code implementation was in Java and not optimized, we only wanted to find a comparison between the two methods, implemented identically. A faster implementation using threading could vastly cut down on execution time for both methods.

	Mean Runtime (s)	Standard Deviation (s)
multiSURF*	8.323	0.369
SURF*	12.276	1.275

Table 1. Comparing the runtimes of multiSURF* and SURF*.

3.3 Real Data Results

The analysis results are shown in Table 2. The symbol $*$ indicates statistical significance ($p < 0.05$), $**$ indicates high statistical significance ($p < 0.01$). We note that multiSURF* found the same genes as the other algorithms, with the exception of BCL2L11, which was found only by SURF*. In addition, multi-SURF* found a new gene, ATK3. Several of the p-values were identical because they were based on discrete counts and a Fisher's Exact test.

	ReliefF	SURF	SURF*	multiSURF*
NAT2	0.0000106**	0.0005274**	1	0.0000106**
PRKCQ	0.001464**	0.001464**	0.001024**	0.001464**
RAF1	0.0001462**	0.301	1	0.01504*
ATK3	0.106	0.106	1	0.0000196**
BCL2L11	0.2953	0.2953	0.02725*	1

Table 2. The probability that each gene received higher rankings by chance.

4 Discussion and Conclusions

The increase in power stems largely from the disregarding of the individuals neither near nor far, which is enabled by the individual thresholds. Since the distribution of zygosity measurements $\{0, 1, 2\}$ is likely not uniform for any given SNP and in our case is highly non-uniform, a given individual I_i may have common or uncommon values. Take for example the case in which 0 is common for all SNPs, the individual with measurements $0, 0, \ldots, 0$ will be much closer, on average, to any other individual. An individual with complementary measurements such as $2, 2, \ldots, 2$ will, on the other hand, be much further from other individuals. Therefore in a large random population there will be individuals for which a global threshold is ill-fitting; instead of splitting other individuals evenly with roughly half being classified near and half far, the majority of individuals will fall into one class or the other. This is effectively a dilution of potentially useful information contained in near and far individuals. If we assume a normal distribution of distances from I_i to all other individuals, then having individual thresholds, T_i, set to the mean of the differences will split the instances evenly into near and far. With this even split we can then apply the dead-band and disregard individuals that are neither very near nor very far. By looking only at the extremes of the distribution we can provide a better result for effectively the same reasons that SURF* works in the first place. A rigorous mathematical proof of this has yet to be presented and is a potential topic of future research. The authors examined several arbitrarily chosen values for the deadband multiplier, namely the α for $\alpha \times \sigma$. $\alpha = 1/2$ worked the best for all tested data, but an argument could be made for α being a parameter of the algorithm. In fact, every dataset may have an optimal α for finding important SNPs in that dataset, but a way to quickly determine the optimal α for a given dataset is not currently known.

Overall multiSURF* is a desirable alternative to SURF*; it takes less time to run and will find an underlying genetic model more frequently. This enables geneticists to filter even larger datasets and search for higher-order models with both greater power and efficiency.

Acknowledgments. The authors would like to thank Jeff Kiralis for general assistance and Nadia Penrod and Jeff Kiralis for reviewing the manuscript. This work was funded by NIH grants AI59694, LM009012, LM010098.

References

1. J.H Cordell. Detecting genegene interactions that underlie human diseases. *Nature Reviews Genetics*, 489:392–404, 2009.
2. 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.
3. 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.
4. K. Kira and L.A. Rendell. A practical approach to feature selection. *Machine Learning*, pages 249–256, 1992.
5. I. Kononenko. Estimating attributes: Analysis and extensions of relief. *LNCS*, 784:171–182, 1994.
6. J.H. Moore. *Knowledge Discovery and Data Mining: Challenges and Realities with Real World Data*, chapter Genome-wide analysis of epistasis using multifactor dimensionality reduction: feature selection and construction in the domain of human genetics. 2007.
7. J.H. Moore and M.D. Ritchie. The challenges of whole-genome approaches to common diseases. *JAMA*, 291:1642–1643, 2004.
8. J.H. Moore and S.M. Williams. Epistasis and its implications for personal genetics. *AJHG*, 85:309–320, 2009.
9. R.R. Sokal and F.J. Rohlf. *Biometry: the principles and practice of statistics in biological research.* 3 edition.
10. D. Thomas. Gene-environment-wide association studies. *Nat. Rev. Genetics*, 11:259–272, 2010.
11. D.R. Velez, B.C. White, A.A. Motsinger, W.S. Bush, M.D. Ritchie, S.M. Williams, and J.H. Moore. A balanced accuracy function for epistasis modeling in imbalanced datasets using multifactor dimensionality reduction. *Genetic Epidemiology*, 31:306–315, 2007.