UPPSALA
UNIVERSITET

MSc BIOINF 13 003

Using Feature Synthesis to discern non-linear interactions via composites

Nicholas Baltzer

Degree project in bioinformatics, 2013
Examensarbete i bioinformatik 30 hp till masterexamen, 2013
Biology Education Centre and Department of cell and molecular biology, Uppsala University
Supervisor: Professor Jan Komorowski

Abstract

Current standards of connecting components to outcome within any system focus primarily on one or more strong components that can be shown to have a direct impact on the outcome, or a key aspect of that outcome. This has proven effective in many situations, though not all, and generally avoids the ever-present issues of computational resources and time consumption.

Whenever a system is found with only weak correlations between component and outcome, the conclusion is that either the causation components are not part of the selected components, or the sampling of the data is insufficient for providing a clear trend. This may not be correct, as the algorithms handling the computations are designed to look for specific patterns of causation, and these patterns are not necessarily the underlying source of the causation.

The new algorithm detailed here uses feature synthesis to create composite features and measures these with an estimated Risk Ratio algorithm to find patterns of causation that may otherwise go undetected.

Using Feature Synthesis to discern non-linear interactions via composites Popular science summary

Nicholas Baltzer

September 7, 2013

Biology is very complex and problems that we hear about but do not understand very well, such as the reasons for Alzheimer's disease or cancer, depend on many factors that contribute together in a complicated way. There can be genetic factors, with variations in the DNA and there can be effects related to proteins, and there can be combinations of those. There can be further variations depending on where in the body these effects occur. These relationships have been found to be very difficult to identify using experimental methods because the amount of data is large and the interactions are not very obvious. When only one factor is important it usually sticks out in an experiment, but when several factors contribute indirectly by influencing several other factors, the analysis becomes very difficult. When this happens, and it happens very often in biology and medicine, there is a need for statistical methods that can see patterns that the experimentalist can not easily see directly. Statistical methods help us find relationships in large datasets that we can not easily find simply by looking at the data we obtained.

An important problem in the statistical analysis of large data sets is the combined effect of multiple factors that together influence outcome. To provide an example, consider a patient with an infection. Injecting this patient with antibiotics will cure the patient, but only if the infection is bacterial, and only if these bacteria are not resistant to antibiotics. Should any of these factors be different, that is, should the infection not be bacterial, or should the bacteria be resistant to antibiotics, the patient will remain sick. These factors can therefore tell us if the patient will be cured by the antibiotics, but only in some of the cases. This is what makes the statistical analysis difficult, since the factors are important in some situations but not in others. Only when all factors line up in a specific pattern do they affect the outcome, which in this case is the conclusion that the patient will be cured by an injection of antibiotics. Finding just how these factors should be lined up is difficult in biological systems, since the question is rarely so straightforward, and the factors included are often not so well-researched.

In this masters thesis, a program has been built to identify these factor patterns. If such a pattern is discovered, it is evaluated to see how strong it is, and all the factors that are included in this pattern are used to create a new factor in the dataset, called a Composite Feature, that will represent this factor pattern. This Composite Feature can then be used in the same statistical methods that find simpler relationships in large datasets since the pattern is no longer hidden in different factors. Using the previous example, with a Composite Feature built from the infection factor and the resistance factor, the type of infection and the bacterial resistance would now be considered as a single factor that can be clearly linked to the success or failure of curing the patient with antibiotics.
The program found patterns both in the synthetic datasets that were created specifically for testing the program, and in datasets of DNA methylation samples called histone modifications. The measurement of strength of the pattern is handled with a statistical method often used in pharmacology called Relative Risk. The program was built in Perl, a programming language designed for managing and manipulating text and lists of text.
This program will simplify analysis of datasets. Areas where this program will assist research efforts include pharmacology, brain research, genetics research, hospital care and patient diagnosis amongst others.

Degree project in bioinformatics, 2013
Examensarbete i bioinformatik 30hp till masterexamen
Biology Education Centre and Department of cell and molecular biology, Uppsala University
Supervisor Prof. Jan Komorowski

Contents

List of Figures

List of Tables

1 Introduction

To start with an analogy, the human body is a vast and complex system. Every peptide has a purpose, and every change in a protein can affect up to every other aspect of the system. This interlinking of the various components in the human body is also the reason for it's complexity, as few components can be succesfully analyzed in a vacuum. To fully understand the possible effects of a single protein, one must study the effect of that protein on every other component, as well as the effect of that component when combined with every other possible combination of components on the human physiology. This is a daunting task, and unlikely to prove technically feasible for a long time.

For smaller scale systems, it can however be practical. Looking at a small set of genes and environmental factors, it is computationally feasible to try them out in combinations, to see which of these have an impact on the outcome. This approach is the basis for phenotype prediction (Lewontin, 2001), and one can imagine the possible applications of such an approach on the neurobiology systems investigated by the Kobilka research group (Swaminath et al., 2004).
In order to evaluate the increased strength of these combinations, it is necessary to know the strength of the individual components, as well as the expected potency of the combination. The value is considered the difference between the expected and the achieved strength, as measured towards the outcome.
Using a measurement of combination strength called Relative Risk (Bevick et al., 2004) together with a type of feature synthesis called composite generation, combinations are measured and evaluated in a pre-process to the more typical bioinformatics evaluation algorithms. This ensures a higher level of compatibility with existing algorithms and methodology as the module can act as a supporting tool for these, since they do not need alteration in order to benefit from this type of combinatorial evaluation.

The purpose of this project was to elucidate combinations that together affect a given outcome where they cannot do so individually, and to use this knowledge for heightened efficacy of existing bioinformatical approaches as well as increased granularity in the datasets used for such tasks. The project entailed the design and construction of a new algorithm with the purpose of detecting and making visible these combinations, implemented in such a way that existing programs and pipelines can make use of the generated data without additional need for conversions. The algorithm was verified with synthetic datasets, and validated with real datasets.

1.1 Datasets

A dataset is an information system consisting of objects, features, and decisions. A feature is an event that can be measured, such as blood pressure, and an object is an observation of the features in the data set. Each object represents an instance of measurements of the features, and each decision represents the outcome of that particular object. Formally defining the dataset, $U = \{O_1, O_2, ..., O_n\}$ is the universe containing all objects, $A = \{F_1, F_2, ..., F_n\}$ is the set of all features, and d is the decision set. V_F is the value set of F, containing all values in the intersection $V_F = U \rightarrow A_F, F \in A$.

To illustrate, in Table 1 the first row lists the feature names of the data set (F_1 .. F_5), with the decision coming last. The first column lists the objects (O_1 .. O_6), and each intersection of an object row and a feature column represents the value of the feature for that particular observation. The last column, the decision d, represents the outcome for that observation.

Table 1: An example data set

	F_1	F_2	F_3	F_4	F_5	d
O_1	1	0	1	0	1	1
O_2	1	1	1	0	0	0
O_3	0	0	1	0	0	0
O_4	0	1	0	1	0	1
O_5	1	0	1	0	1	1
O_6	1	1	1	1	1	1

A dataset may contain both discrete data, and numerical data. For the example dataset, only discrete data is used, as it is easier to interpret.

1.2 Correlation

Correlation indicates a dependency between two variables, such that the outcome of one affects the outcome of the other. There are many kinds of correlation in statistics, though all revolve around the dependency of variables. The specific correlation applied in this project is the Pearson product-moment correlation (Pearson, 1900), and the values mentioned as correlation values refer to the Pearson product-moment correlation coefficient. This coefficient is measured as a linear dependency, meaning that non-linear relationships between variables do not impact the correlation. The coefficient, denoted r, is equal to 1 in the case of a perfect linear correlation, and -1 in the case of a perfect negative linear correlation. If the coefficient is 0, it means that there is no linear relationship at all between the two variables. Any value other than these specified indicate the degree and sign of linear dependence between the variables.

The Pearson correlation, with r as the correlation coefficient, X_i as a value in the sample paired with Y_i, $\bar{X}$ as the mean of the sample (with $\bar{Y}$ as the mean of the pairing sample), is calculated with

$$r = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \bar{X})^2}\sqrt{\sum_{i=1}^{n}(Y_i - \bar{Y})^2}}.$$

The simplified though equivalent expression of this formula is

$$r = \frac{1}{n-1}\sum_{i=1}^{n}(\frac{X_i - \bar{X})}{s_x})(\frac{Y_i - \bar{Y}}{s_y})$$

where $\frac{X_i - \bar{X}}{s_x}$ is the standard score, $\bar{X}$ is the sample mean, and s_x is the standard deviation.

As an example, in Table 2, when columns F_2 and d hold the same value on the same row, they are considered correlated, as their values indicate some form of dependence. F_1 has an equally strong anti-correlation, as it is the opposite of d in all but one observation. F_3 has no correlation to d.

As a more visual definition, consider the statements "it is raining" and "the streets are wet". These two events usually occur together, and as such there is a dependency between them.

Table 2: Example of correlated features F_2 and F_3, and anti-correlated feature F_1

	F_1	F_2	F_3	d
O_1	0	1	1	1
O_2	1	0	0	0
O_3	1	0	1	0
O_4	0	1	0	1
O_5	1	0	0	1
O_6	0	1	1	1

1.3 Decision Trees

A decision tree is a representation tool that uses a tree graph to represent features, feature values, and decisions as branch nodes, edges, and leaf nodes respectively. The path taken from the top node down to a leaf node represents the values required of the features along that path to determine the decision of such an object. The tree is constructed from the top down, and the addition of each feature to the graph is dependent on the information gain accorded such an inclusion.
The information gain of a feature is calculated as

$$i_{xy} = -\frac{x}{n} \cdot log_2 \frac{x}{n} - \frac{y}{n} \cdot log_2 \frac{y}{n}$$

where x and y are the number of objects from each class, and n is the total number of objects. For the feature chosen, the information values are weighted. The weighted information value is

$$\sum_{k=1}^{p} \frac{x+y}{n} \cdot i_{xy}$$

where x and y are the number of objects from each class, n is the total number of objects and p is the cardinality of the feature's value domain V_F. The information gain from choosing a feature is then the information value subtracted by this weighted information value

$$g = i_{xy} - \sum_{k=1}^{p} \frac{x+y}{n} \cdot i_{xy}$$

where g is the information gain. Since this method is biased towards features with many values, the gain is divided by s, the information value of the feature's value domain, which only takes the frequency of values into account. The formula is the same as for the information value, but instead of objects from the different classes, the frequency is used for x and y. The final value, called the information gain ratio r, is thus $r = \frac{g}{s}$. The feature with the highest r value will represent the topmost node in the decision tree.
For more details on decision tree methodology, see Safavian (1991).

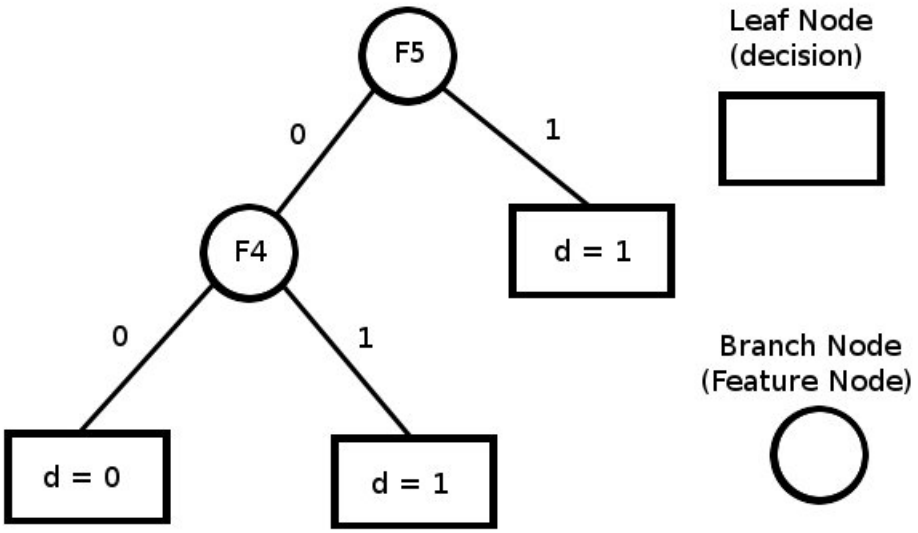

Figure 1: Example decision tree created from Table 1.4

In Figure 1.3 there are two decision nodes representing the features F_4 and F_5. The edges from these nodes represent the values these features would need in order to predict a decision. The decision tree is created from the dataset shown in Table 1.4.

Decision trees are used when speed is a concern, as a classification of this nature requires very few comparisons and relies on a single conditional structured rule rather than the multiples of a rule-based approach. It is also fast in terms of ranking features relative to each other, as the first node is the most significant feature and the features at the next level are the second most significant etc.

1.4 Reducts

A reduct is a minimal set of features that retain the same discernability towards the decision as the original dataset. This is managed via equivalence classes, where observations that use the same features to predict the decision are grouped and treated as one. To reduce the set further, the features that are universally irrelevant to estimate the decision are discarded, resulting in a smaller data set with the same level of information regarding the decision (or selected criteria).
In Table 1.4 there are six observations of five features. Each observation has a decision value, of which there are only two.

Table 3: Example reduct set (unreduced)

	F_1	F_2	F_3	F_4	F_5	d
O_1	1	0	1	0	1	1
O_2	1	1	1	0	0	0
O_3	0	0	1	0	0	0
O_4	0	1	0	1	0	1
O_5	1	0	1	0	1	1
O_6	1	1	1	1	1	1

Applying a reduct feature selection method on the data set, three reducts are $(F_1 \wedge F_2 \wedge F_5)$, $(F_1 \wedge F_4 \wedge F_5)$, and $(F_2 \wedge F_4 \wedge F_5)$. Table 1.4 is a recreation of Table 1.4 in reduced form, using minimal reduct $F_1 \wedge F_2 \wedge F_5$.

Table 4: Example reduct set (reduced)

	F_1	F_2	F_5	d
O_1	1	0	1	1
O_2	1	1	0	0
O_3	0	0	0	0
O_4	0	1	0	1
O_6	1	1	1	1

In order to create a reduct, the observations must first be categorized into equivalence classes. Each class will contain observations that need not be differentiated from each other, such as O_1 and O_5. The result is a possibly smaller set, where each observation represents an equivalence class. In standard notation, the first observation encountered for an equivalence class becomes the representative of that class. In Table 1.4 both the observations that are part of the equivalence class O_1 are listed for the sake of clarity.
Further consolidating the data set, the equivalence classes are used to construct a discernibility matrix, defining which features are needed in order to differentiate between the classes. Whenever a feature in one equivalence class differs from that of another class, it can be used to differentiate between the two classes. Table 1.4 provides all the features following this approach. Note that O_5 is not part of the matrix, since it belongs to the

Table 5: Equivalence classes

	F_1	F_2	F_3	F_4	F_5
O_1, O_5	1	0	1	0	1
O_2	1	1	1	0	0
O_3	0	0	1	0	0
O_4	0	1	0	1	0
O_6	1	1	1	1	1

class represented by O_1.

Table 6: Discernability Matrix

	O_1	O_2	O_3	O_4	O_6
O_1	$\emptyset$	$F_2 \vee F_5$	$F_1 \vee F_5$	$F_1 \vee F_2 \vee F_3 \vee F_4 \vee F_5$	$F_2 \vee F_4$
O_2		$\emptyset$	$F_1 \vee F_2$	$F_1 \vee F_3 \vee F_4$	$F_4 \vee F_5$
O_3			$\emptyset$	$F_2 \vee F_3 \vee F_4$	$F_1 \vee F_2 \vee F_4 \vee F_5$
O_4				$\emptyset$	$F_1 \vee F_3 \vee F_5$
O_6					$\emptyset$

Using Table 1.4, the full discernability is $(F_2 \vee F_5) \wedge (F_1 \vee F_5) \wedge (F_1 \vee F_2 \vee F_3 \vee F_4 \vee F_5) \wedge (F_2 \vee F_4) \wedge (F_1 \vee F_2) \wedge (F_1 \vee F_3 \vee F_4) \wedge (F_4 \vee F_5) \wedge (F_2 \vee F_3 \vee F_4) \wedge (F_1 \vee F_2 \vee F_4 \vee F_5) \wedge (F_1 \vee F_3 \vee F_5)$. A reduct must retain the discernability of the equivalence classes, and thus be constituted of at least one representative feature from each equivalence class. Furthermore, the additional condition for a minimal reduct is that the set of features chosen is as small as possible. There may be multiple possible configurations of features that fit this criteria. A minimal reduct of this formula is $F_1 \wedge F_2 \wedge F_5$, since these three features together can discern between all the equivalence classes. Constructing a reduct with only these features will provide the same accuracy as the original Table 1.4 while retaining the smaller size.
There is also the possibility to go one step further, and remove discernability between equivalence classes with the same decision value. Constructing a new matrix, only when the decision value differs are the equivalence classes discerned (Table 1.4).

The full discernability modulo decision is now $(F_2 \vee F_5) \wedge (F_1 \vee F_5) \wedge (F_1 \vee F_3 \vee F_4) \wedge (F_4 \vee F_5) \wedge (F_2 \vee F_3 \vee F_4) \wedge (F_1 \vee F_2 \vee F_4 \vee F_5)$. A minimal reduct of this formula is $F_4 \wedge F_5$. Thus, for the purpose of determining a decision value given an observation, all features except F_4 and F_5 can be ignored. These features are also the two used in constructing the decision tree in Figure 1.3.

Reducts can be approximated (Slezak, 1996) (a method that is available with both Rosetta and Dmlab), an approach that can greatly reduce the reduct size when the datasets are incomplete or contain noise, though such an approach also reduces the accuracy of the reducts.

Table 7: Discernability Matrix modulo decision

	O_1	O_2	O_3	O_4	O_6
O_1	Ø	$F_2 \vee F_5$	$F_1 \vee F_5$	Ø	Ø
O_2		Ø	Ø	$F_1 \vee F_3 \vee F_4$	$F_4 \vee F_5$
O_3			Ø	$F_2 \vee F_3 \vee F_4$	$F_1 \vee F_2 \vee F_4 \vee F_5$
O_4				Ø	Ø
O_6					Ø

1.5 Non-linear interactions

Non-linear interactions occur when one feature in combination with another one shows a stronger correlation to the decision than the expected linear increase of such a conjunction. For two features F_1 and F_2, with correlations 0.1 and 0.2 respectively, the linear increase would be the correlation interval $[0.2, 0.3]$. A conjunction that leads to a correlation outside this interval is considered a non-linear interaction. Note that in this example, a correlation of 0.31 indicates a non-linear interaction, though such a low increase is of little interest application-wise.

The linear interval for two features is bounded at the lower end by the largest correlation of the two features, and at the upper end by the sum of the two correlations, though never greater than 1.0. The interval can thus be defined as $[c(F_i), c(F_i) + c(F_j) \vee 1.0] : c(F_i) > c(F_j) | F_i, F_j \in A$ where $c(F)$ represents the correlation of feature F to the decision. Due to the nature of datasets, the linear interaction will statistically tend to the lower end of the linear interval as the correlation of the features increase, since the probability of overlapping correlations increase. Consider the feature correlations $c(F_1) = 0.60$ and $c(F_2) = 0.55$. The linear interval is $[0.60, 1.0]$. There is no possibility of avoiding overlapping correlations, with the minimum overlap being 0.15.

Non-linear interactions can also be viewed as strengthening the predictive power of other features. As such, when features F_i and F_j are part of an interactive pair, removal of F_i from the dataset will reduce the predictive power of F_j. Feature removals do not affect linear interactions in the same way, since these will contribute the same even if placed in two different subsets of the dataset.
To illustrate, view Figure 1.5. The features (circles) are directly correlated, i.e. linear. Removing one feature from the set would not impact the predictive power of the other.

When a non-linear interaction is entailed, both features must be present for the stronger correlation to become apparent. Removing one would impact the ranking of the second, reducing the correlation to the individual strength of the feature. The distinction is illustrated in Figure 1.5.
The nature of non-linear interactions inhibit their discovery to varying extents. Decision trees (1.3) are ill suited to finding them, since the increased correlation only becomes apparent once all the features have been included in the tree. As a result, larger interaction sets are harder to find than smaller sets.

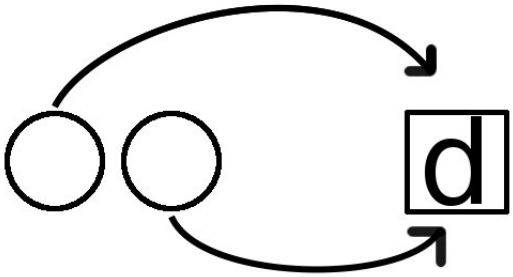

Figure 2: Example of features directly correlated to the decision

Reducts (1.4) are better equipped since the method will look for a combination of features to cover every outcome, though interaction set size is still an issue with the Random Reducts algorithm.

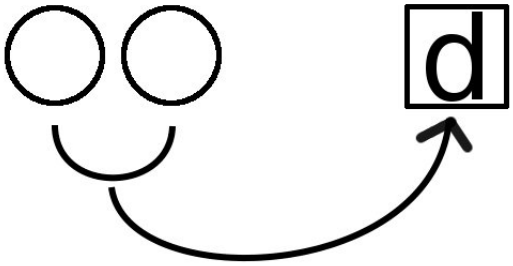

Figure 3: Example of features correlated by interaction to the decision

The creation of interactions using the algorithms presented in this study relies on a primary feature that is correlated to the decision, and a secondary feature that specifies the relation between the primary feature and the decision. The primary feature (labelled f_p in example Table 9) is randomly generated. The secondary feature (f_s) is set to 1 if the primary equals the decision, and 0 otherwise. To understand the interactive behaviour of this model, consider discerning the decision using only one of the features, then the other, and last, both of them. The effect is clearly shown in Table 8, where there is no correlation at all between features and decision. Table 9 shows the same features, though both in the same set. Using these features together, one can fully predict the decision. Comparing this to the linear interactivity of F_p and F_s, the expected interval is $[0, 0]$, as reflected by the correlation of each feature as shown in Table 8, yet the actual correlation of this conjunction is 1.0.
More information on the behaviour of interactions and the need for bioinformatic approaches to uncover them can be obtained from (Droit et al., 2005).

1.6 Feature Synthesis

Feature Synthesis is a term for the generation of features from other features or properties with the purpose to direct, tilt, or reduce the noise while still retaining the desired information and an accurate correlation to the decision. This generation results in groupings

Table 8: Example interaction set with separated features

F_s	d	F_p	d
1	1	1	1
0	0	1	0
0	1	0	1
1	0	0	0

Table 9: Example interaction set

F_s	F_p	d
1	1	1
0	1	0
0	0	1
1	0	0

of properties as they relate to the outcome, which can reduce the size of the dataset and intensify the visibility of features and properties (Rudnicki and Komorowski, 2004).
Composite feature generation is a tilt towards non-linear interactivity. This method creates a single feature from multiple in order to highlight the interactive capacity of the constituent features for the evaluation algorithms. Composite features are created by concatenating the constituent features at each observation (for an example, see Table 13). The resulting composite has a value domain cardinality in the interval $[n, m \cdot n]$ where n is the constituent with the highest value domain cardinality and m is the lowest. The increase of the value domain is important, since such a factor may impact the composite score when evaluated using existing bioinformatics algorithms. As such, the boost to the evaluation score for the composite is not linear to the increase in strength solely, but may also be inversely so to the size of the value domain.

The number of possible composites in a dataset depends on the size of the composites, since each may contain up to every feature in the dataset, and the number of features in the dataset. For n features with composites of size k, the number of possible combinations would be

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

While not all features are expected to appear in composites, all possibilities must still be evaluated, leading to a significant number of computations.

Table 10: Constituent features and the resulting composite

F_1	F_2	F_c	d
1	1	11	1
0	1	01	0
0	0	00	1
1	0	10	0

2 Algorithms

To evaluate the performance of the project module in a bioinformatical setting, two ranking algorithms are used. Monte Carlo Feature Selection (MCFS) and Random Reducts. They both rank features in a dataset depending on how important they are to determine the decision. MCFS is a proven method and has been used for several years, where as Random Reducts is more recent. They use different approaches to ranking, and each has it's benefits and drawbacks.

2.1 Monte Carlo Feature Selection

MCFS ALGORITHM(s, t, m)
1 **for** $j = 0$ **to** s
2 **do** Create subset s_j with m random features
3 **for** $j = 0$ **to** s
4 **do** Split s_j t times
5 **for** $i = 0$ **to** t
6 **do** train a classifier on the training part of t_i
7 build a decision tree from the classification
8 test the classifier on the testing part of t_i and score the features in the decision tree

The Monte Carlo Feature Selection, or MCFS (Dramiński et al., 2008), is a current standard for feature selection. The algorithm works by creating s subsets with m random features taken from the original data set. These subsets are in turn split into t pairs of training and testing sets for a decision tree (1.3) classifier. The features in the subset are then scored according to their classification performance, and these scores are used to calculate the relative importance (RI) of each feature. The algorithm pseudocode is detailed above. For a more comprehensive view, consider Figure 4.

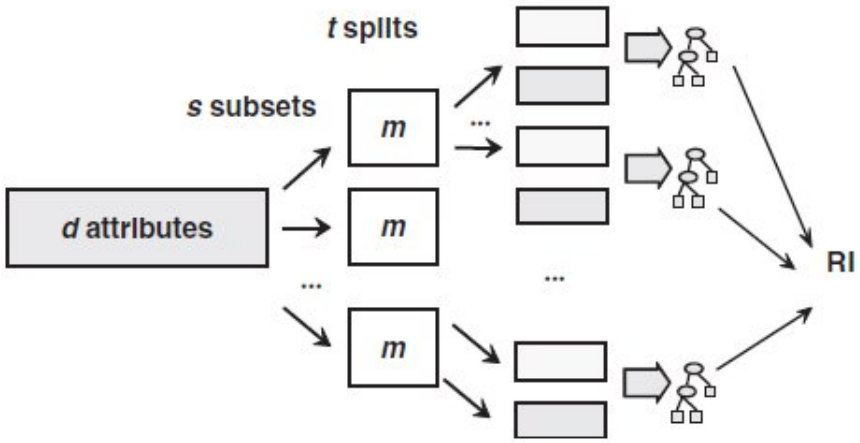

Figure 4: Visual representation of the MCFS algorithm

2.2 Random Reducts

The algorithm is provided in a modified version of command line Rosetta (Øhrn, 1996), courtesy of Marcin Kruczyk (Kruczyk et al., 2013). Apart from the addition of Random Reducts, the overall speed has also been increased via the implementation of multi-threading, an approach that has already been proven succesful (Steiß et al., 2012).
Similar to MCFS, a subset of features from the data set is selected at random. A reducer is then used to create reducts from the subset. At this point, depending on configuration, either rules are generated from the reducts and measured for accuracy and support, or the features appearing in the reducts are counted towards a feature appearance total. The final results are those features that appear often in reducts during a significant number of iterations of this process.
Currently, Random Reducts can apply four different reducers. All the experiments used Johnson, but there are also Holte 1R, Rey, and Genetic reducers available.

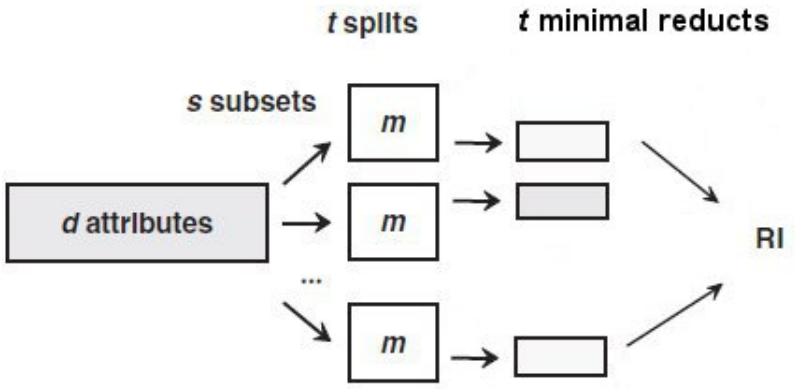

Figure 5: Visual representation of the RR algorithm

2.3 Cut-off Points

Selecting a suitable cut-off point is critical, and time-consuming. The algorithms described in 2.1 and 2.2 provide a ranking of the features in relation to each other, but no indicator to where the cut-off should be regarding significance, that is, at which point in the ranking features are no longer considered significant. Without a cut-off, features are ranked only in relative terms, since there would be no requirement of reaching a minimal significance. This aspect is handled with permutation tests.

Permutation tests revolve around permuting the decision values. This will essentially randomize the data set outcomes. Given a null hypothesis that the selected features are not better than other chosen features, multiple permutations of the outcomes and re-evaluations should result in a set of deviations from the original observation. If 5% or more of these deviations score higher than the original observation, then the null hypothesis is correct, and there is adequare chance that other features may perform better than the selected ones. If less than 5% of these deviations perform better, the null hypothesis can be rejected, and the features selected are significant (Pitman, 1937; Fisher, 1935).
Permutation testing is computationally expensive, and accounts for the majority of the time spent on ranking features. It does however provide a sound and solid cut-off point. In particular, it has been used extensively during this study due to the similarities in results provided by the two ranking algorithms when combined with this method.

During the experiments, both the normalized and basic approaches of the permutation test run method were used with the Random Reducts algorithm on each data set. The difference is a normalization during the calculations where the mean score of all features is subtracted from the score of each single feature, and the result is divided by the standard deviation in order to provide a normalized score for each feature. If a score is numbered by its feature i, the calculation is $\frac{score_i - score_{mean}}{StD}$.
Feature scores can be calculated in many ways, but there are three basic measurements

that are commonly used: rule support, rule accuracy, and rule coverage. Rule support is the number of objects where the rule predicts the correct outcome. Rule accuracy is the fraction of outcomes the rule correctly predicts, as compared to the number of objects where the rule can be applied. Defining a as the accuracy, s as the support, and d as the number of objects that can be classified with this rule, the formula for rule accuracy is $a = \frac{s}{d}$.

Rule Coverage is the fraction of rule support divided by the total number of objects that have the same outcome. It is a definition of how much of the data set is covered by a specific rule, and is given by $c = \frac{s}{|d = d_s|}$.

There is another method for selecting cut-off points with RR. It uses a method similar to the *Receiver operating characteristics* (ROC). The cut-off point is not numerical in value but rather a last feature point, where all features of lower score are discarded. This method is very fast, but suffers from certain exclusion issues. If all features in the set are of significant value, at least half will be discarded none the less. Similarily, if no features are of significant value, some of them will still be kept.

2.4 Relative Risk

Relative Risk, sometimes called Risk Ratio, is a measurement of how one factor impacts an outcome relative to another. That is, how the value of a specific feature can increase or decrease the likelihood of the outcome when compared to the same situation with a different value for that feature. The equation for determining this relative risk (RR) is

$$RR = \frac{a/(a+b)}{c/(c+d)}$$

where a is the frequency of the value that match the measured outcome (such as $smoker \wedge carcinoma$), and b is the frequency of the value that does not match the measured outcome (such as $smoker \wedge \neg carcinoma$). Opposite to these, c is the inverted frequency of b ($\neg smoker \wedge carcinoma$), and d is the opposite of a ($\neg smoker \wedge \neg carcinoma$). Extending these definitions, $a + b$ is the number of all smokers, and $c + d$ is the number of all non-smokers. These variables fall into the groupings of true positive, false negative, false positive and true negative respectively. Calculating the relative risk on a feature with binary values, the variables a, b, c, and d represent the frequency of feature to decision pairings, as shown in Table 11.

Table 11: Relative Risk as frequency of pairings

	F	d
a	1	1
b	1	0
c	0	1
d	0	0

The result of this equation will be a number in the interval $[0, \infty[$, where 1 signals no difference in the presence of the distinguished feature value. The anti-correlated interval $]0, 1[$ indicates a reduced frequency of occurences in the presence of the feature value.
Calculating the relative risk coefficient on two different features, an estimated relative risk (ERR) of the features in conjunction can be found by multiplication. Comparing this estimate with the actual value procured provides a good indicator of interactive behaviour between the two features. If the actual value is greater than the estimate in the correlated interval, or less than the estimate in the anti-correlated interval, the two features exhibit interactive behaviour with one another. This measurement of interaction can be extended to an arbitrary number of features when looking for interactive behaviour.
A more detailed explanation of relative risk is provided in Bevick et al. (2004).

3 Implementation

3.1 Synthetic dataset generator

In order to test the efficacy of the composite generator, it is necessary to provide datasets with known properties. Otherwise, the experimental results cannot be used to calibrate the algorithm, and the output cannot be verified as correct. The synthetic dataset generator has been implemented to provide sets with features of known correlation and interactive behaviour. Below are the algorithms and definitions of the synthetic dataset generator. The application is written in Perl 5.8, and runs on a 12-core unix server with 96GB of memory.

3.1.1 The basic algorithm

The implementation of the basic correlation features uses the following parameters: the correlation s of the first feature, the loss in correlation i towards the second feature, and the minimum correlation e of the last feature.
For the first feature, each value will be randomly selected as either the decision value, or a random value, where the correlation s provided is the probability that the value will be the same as the decision value. For each subsequent feature, the probability decreases by the given loss in correlation i, until the probability is lower than that of the minimum correlation specified, at which point the feature generation stops.

BASIC GENERATION ALGORITHM(s, i, e)
1 Generate a random decision vector D
2 Probability $p = s$
3 **while** $p \geq e$
4 **do for each** object o **in** feature F_p
5 **do if** $p \geq$ randomly chosen value in $[0, 1[$
6 **then** $F_p(o) = D(o)$
7 **else** $F_p(o) =$ randomly chosen value in $\{0, 1\}$
8
9 $p = p - i$

Using this method to generate features, the correlation of the feature to the decision is not exact, with a deviation dependent on the number of objects. From the results, 10,000 objects created a maximum deviation of 0.02 from the Pearson correlation. To counteract this uncertainty, correlations are calculated by the program after the data set generation so that the exact numbers are known.

3.1.2 Duplication

In some situations, tests can benefit from having two or more of the same feature included in the dataset, and this functionality has been implemented. If duplication is selected, the program will randomly duplicate features D times with probability d in the generated

dataset S, all parameters specified by the user. Only the basic set (3.1.1) is affected by duplication.

DUPLICATION ALGORITHM(D, d, S)

```
1  for each feature f in S
2     do if d > randomly chosen value in [0, 1[
3           then for i = 0 to D
4                   do duplicate f
```

3.1.3 Interactive Element Generation

The program can also create a non-linear interaction between feature pairs.
The implementation is such that the first feature has each object set randomly, and the second feature has each object set to the binary variant of $F_1 == d$, meaning that whenever feature F_1 has the same value as decision d for the same observation, the second feature $F_2 = 1$ for that observation. As a result, the two features can together discern the decision with perfect accuracy, while any one of them alone shows at best a random connection.
Scaling the pairs to below perfect correlation is handled by randomizing the second feature, much like the basic algorithm does.
Like the basic algorithm, the set of feature pairs are generated by specifying the parameters for starting correlation s, the increment i, and the ending correlation e. The decision vector D is carried over from the basic algorithm, since both subsets should rely on the same decision values. The first feature pair will have a non-linear correlation of s, and each subsequent pair will have the non-linear correlation reduced by i, until the generation stops at correlation e.

INTERACTIVE ELEMENT GENERATION ALGORITHM(s, i, e, D)

```
1   Probability p = s
2  while p ≥ e
3     do  Generate a random feature vector F_p1
4         Generate an empty feature vector F_p2
5         for each object o in feature F_p1
6            do if p ≥ randomly chosen value in [0, 1[
7                  then F_p2(o) = F_p1(o) == D(o)
8                  else F_p2(o) = randomly chosen value in {0, 1}
9
10        p = p − i
```

3.1.4 Data Naming Conventions

The features generated by the algorithms are named after the correlation they individually hold to the decision. For example, a feature named "83.90" has a correlation of 0.8390 to the decision. Features that are part of an interaction set are have a naming scheme of $i_{jk}n$, where i indicates that the feature is part of an interaction pair, j is the

interaction identifier, k is the feature identifier within that pair, and n is the individual correlation the feature holds to the decision. Example names of these interactive features are "i_{11}00.24" and "i_{12}01.65", indicating that both features belong to the first interactive pair, and their individual correlations to the decision are 0.0024 and 0.0165. Table 12 shows this notation in a ranking table format.
In the case of other data sets, the original names are kept (such as the histone modifications dataset Table 23).

Table 12: Example feature ranking with an interactive pair

MCFS	RR
69.70	i_{11}00.24
66.44	i_{12}01.65
i_{11}00.24	
55.16	
48.14	
i_{12}01.65	

3.2 Composite Feature synthesizer

The Composite Feature Synthesizer is written in Perl 5.8, and all tests have been performed with a 12-core server with 96GB of memory. The algorithms applied by the synthesizer are described below. Note that all algorithms are implemented as single-threaded solutions, impacting the final speed of the module as well dataset size limitations for the experiments. It is expected that such limitations can be eliminated should the module be rewritten on a different platform.

3.2.1 Branching

The algorithm tests each possible interaction, and as such the features are calculated for relative risk values with respect to all possible feature values. That means a feature with the value domain of $\{0, 1\}$ is calculated twice, once for each relative risk value. To account for this when generating composites, the features are copied such that each relative risk value is tied to a specific composite for the comparison. The branching is performed before relative risk calculations, both on the original features and the generated composites.

BRANCHING ALGORITHM(F)
1 **for each** value v **in** value domain V_F
2 **do** Create new feature $F_v = F$

3.2.2 Composite generation

A composite feature consists of two separate features (possibly composites themselves) that are joined together at each observation. The values at each observation are concatenated, such that an observation $F_1(o) = 1$ and $F_2(o) = 0$ would lead to the observation in the composite feature $F_c(o) = 10$. For an example of how feature values affect the composite, see table 13. The result is a feature with a possibly larger value domain, wherein each observation represents the interactive value, if any, between the two features involved. Note that as each value is concatenated, it can no longer be computed as quantitative. Thus, any discretization must be performed prior to running the composite generator.

COMPOSITE GENERATION ALGORITHM(DS)
1 Create an empty set CS
2 **for each** feature x **in** dataset DS
3 **do for each** feature y **in** vector slice $\{x + 1, DS\}$
4 **do** Create an empty feature f_{xy}
5 **for each** object o **in** feature x
6 **do** $f_{xy}(o) =$ concatenate $f_x(o)$ with $f_y(o)$
7 $CS_{xy} = f_{xy}$

Table 13: Example features and the composite generated by them

F_1	F_2	F_c
0	0	00
1	0	10
1	1	11
0	1	01

3.2.3 Relative risk calculations

The relative risk calculations implemented are divided into three steps: sorting the observations into groups, sorting the groups into relative risk variables (which together are called the risk square), and calculating the relative risk using the assembled variables.

The observations are first sorted into groups, and a category is created for each appearing combination of observation value to decision value for the specified feature, such as $\{0, 1\}$. There will be at least one category, and at most $|V_F| \times |V_d|$ categories.
The mathematical formula for counting the frequency of an observation value x to a decision value y is $\sum_{i=0}^{n} F(o) \wedge d(o) : |\{F(o) = x, d(o) = y\}| \forall o \in D$.

The algorithm counts the frequencies using feature F and decision d.

GROUP SORTING ALGORITHM(F, d)
```
1   Create empty vector of categories C
2  for each observation o in feature F
3     do if pair {F(o), d(o)} exists in C
4           then C_{F(o),d(o)} = +1
5           else  add pair {F(o), d(0)} to C
6                 C_{F(o),d(o)} = 1
```

Sorting the categories into relative risk variables, the categories vector C from the group sorting algorithm is combined with the desired feature value v_f to decision value v_d. Each category is then sorted into a, b, c and d according to these values(as detailed in 2.4).

RISKSQUARE SORTING ALGORITHM(C, v_f, v_d)
```
1   Create RiskSquare vector R
2  for each pair {f, d} in category vector C
3     do if f == v_f ∧ d == v_d
4           then R_a = +C_{o(F),d(o)}
5           else if f == v_f ∧ d! = v_d
6                  then R_b = +C_{o(F),d(o)}
7           else if f! = v_f ∧ d == v_d
8                  then R_c = +C_{o(F),d(o)}
9           else if f! = v_f ∧ d! = v_d
10                 then R_d = +C_{o(F),d(o)}
```

The last step of the implemented relative risk calculations is to calculate the relative risk value from these variables. To avoid problems with edge cases, (such as $a + b = 0$,) if either dividend or divisor equal 0, they are replaced with only a or c respectively. Furthermore, if the divisor is 0 even after this alteration (such as when $c = 0$,) it is set to 1. This method leads to a undervaluation of the relative risk, but creates no issue as the value will still be sufficient to overcome even the most stringent of boundaries. Last, should $c = 0$ and $d = 0$, the relative risk value is set to equal a. The final calculation uses the risk square vector R from the RiskSquare sorting algorithm, as it contains the values for variables a, b, c and d.

RELATIVE RISK CALCULATION ALGORITHM(R)

```
 1  if R_a + R_b == 0
 2     then dividend = R_a
 3     else  dividend = R_a/(R_a + R_b)
 4  if R_c + R_d == 0
 5     then divisor = R_c
 6     else  divisor = R_c/(R_c + R_d)
 7  if R_c == 0 and R_d! = 0
 8     then RR = dividend/(1 + R_d)
 9     else  if R_c == 0 and R_d == 0
10              then RR = R_a
11     else  RR = dividend/divisor
```

3.2.4 Feature and Value Bounding

The Composite Feature Synthesizer is very demanding of the computer system, both in terms of the number of computations and in terms of memory requirements, and as such it is beneficial to reduce the need for these two factors in as many ways as possible.
One of the ways implemented to reduce these factors is to early on remove features from the calculations that do not match the expectations of a non-linear interaction. In principle, this approach is a branch and bound algorithm as they are described in Lawler and Wood (1996). The current implementation uses two distinct value boundaries to remove features: an upper and lower limit on relative risk values, and a minimum increase boundary for composites when comparing the estimated relative risk (ERR) to the actual relative risk of the composite feature (2.4).

The relative risk value limit is intended to remove features that already show a correlation to the decision of sufficient strength, and thus remove branching calculations that would later require significant time to compute. Furthermore, these features are less likely to appear in interactions. The lower boundary is here an anti-correlation, but otherwise the same as the upper.
The minimum increase boundary (called Optimality Filtering) is a percentage signifying how much better than the estimated relative risk value a composite must be for inclusion. While this does not reduce the amount of computations as efficiently as the previous

boundary, it does sort out the weaker, and sometimes random, interactions from the resulting dataset.

The algorithm has been designed to evaluate the set of features FS with regards to the relative risk set RR of those features, and remove any features from that set if they have a correlation either greater than the upper limit $upperB$ or less than the lower limit $lowerB$.

RELATIVE RISK BOUNDING ALGORITHM($FS, RR, upperB, lowerB$)
1 **for each** item i **in** Relative risk Set RR
2 **do if** $RR_i \leq lowerB \vee RR_i \geq upperB$
3 **then** Strip feature FS_i from featureset FS

The Optimality Filtering is handled in a different algorithm, and applied at the end of the process. The boundary factor B is typically in the interval [1.1, 1.6], which represents an increase in actual relative risk value over estimated relative risk by 10% to 60%. While this interval works well, the limit parameters can be set by the user. The Optimality Filtering is only applied to the set of composites CS.

OPTIMALITY FILTERING ALGORITHM(RR, CS, B)
1 **for each** composite feature c **in** composite set CS
2 **do if** Relative Risk value $RR_c > 1$
3 **then if** $RR_c \leq RR_{f1} \cdot RR_{f2} \cdot$ boundary factor B
4 **then** Strip feature c from CS
5 **if** Relative Risk value $RR_c \leq 1$
6 **then if** $RR_c \geq RR_{f1} \cdot RR_{f2} \cdot (1 \div B)$
7 **then** Strip feature c from CS

3.2.5 Naming Conventions

Composites are named after the features they constitute of with a dash inbetween. The naming scheme is thus either n or $i_{jk}n$, where n is the correlation in percent to the decision, i indicates a feature generated as part of an interactive pair, j is the interaction the feature is part of, and k is the feature number within that interaction. The interacting feature is then listed after the first with a dash separator, such as i_{12}-00.58–i_{11}-00.14. This means that the composite consists of the features i_{12}-00.58 and i_{11}-00.14.

When data not provided by the synthetic data generator is used, the naming conventions do not apply, though the dash separator is still present in the case of a non-linear interaction. Examples of this can be seen in Table 23.

When a dataset is processed with the composite generator more than once, there is the possibility that composites from a previous iteration are included in new composites during a subsequent iteration. In this case, there will be multiple dash separators between more than two features, such as i_{12}-00.58–i_{11}-00.14–25.22. This indicates that the

first composite, i_{12}-00.58–i_{11}-00.14, showed sufficient improvement when combined with the feature 25.22 to warrant a new composite with all three features merged. There is no upper limit to how many features can be included in a composite this way. Examples of this can be seen in Table 23.

4 Experiments

The experiments have been performed in two phases. First, using the synthetic data generator, datasets with known properties and interaction pairs were generated. These datasets have several different parameters to portray a wide variety of inputs. Then, the datasets were processed with the composite generator multiple times to find the parameters that best enhanced visibility. Note that while the composite generator has been used multiple times on each dataset, the original set was used as input each time, not the composite-processed one.
Once a suitable set of parameters was matched to a dataset, both the original (non-composite) dataset and the processed set were ranked using the Monte Carlo Feature Selection and Random Reducts algorithms. A selection of these ranking tables are detailed in Section 5, together with a short description of their interesting factors.

This testing phase asserted the veracity of the composite generator, and the parameter intervals it needed to correctly discern interactive behaviour of varying strength.

The second phase was the testing of real datasets to find interactive behaviour not seen before.
Using a dataset obtained from a study or database, the composite generator processed the table with varied parameters to search for interactive behaviour. Note that the datasets used were such where interactive behaviour was expected or believed to exist. Should a dataset be too large to process in reasonable time, a subset was taken to represent it.
Since the real data sets may contain interactive behaviour in pairs greater than two, the output from the composite generator was processed again with a varying set of parameters, though these were higher than during the initial process to increase the requirements of strength for interactions. Once this was complete, the original dataset, as well as any output of interest, was ranked with MCFS and Random Reducts. The results are contrasted in Section 5.

The second phase showed the practical application of the composite generator module.

5 Results

5.1 Ranking tables and dataset conventions

Results from the Monte Carlo Feature Selection (MCFS) and Random Reducts (RR) algorithms are listed side by side, with a column inbetween signifying the difference between results in number of rankings apart with respect to the MCFS ranking. Should features appear in the same location in both rankings, the difference is zero, listed as a dash in the variation column. Should a feature present in the MCFS ranking not be present in the RR ranking, the variation column will mark this with XX.
Names for datasets from the synthetic data generator are covered in 3.1.4, and composite names are covered in 3.2.5.

5.2 Results

Table 14: $i_1 = 100\%$ $i_2 = 80\%$ without composites

MCFS	Variation	RR
69.78	XX	i_{12}-00.58
60.39	XX	i_{11}-00.14
i_{11}-00.14	1	
51.87	XX	
45.10	XX	
40.38	XX	
i_{12}-00.58	6	
i_{21}-00.61	XX	
28.94	XX	
i_{22}01.51	XX	

In Table 14, MCFS is picking up the singles, with i_{11} third. This reflects the scoring measurement, where i_{11} and i_{12} are rated as sufficient if they both appear in it, but weak if only one does. Random Reducts finds the interactive features i_{11} and i_{12} sufficient, discarding all other features.
Using normalization with Random Reducts does not produce a different result with this dataset, as it only serves to enhance the score of the interactive pair.

Table 15: $i_1 = 100\%$ $i_2 = 80\%$ with composites

MCFS	Variation	RR
69.78	–	69.78
i_{12}-00.58–i_{11}-00.14	–	i_{12}-00.58–i_{11}-00.14
60.39	XX	
51.87	XX	
45.10	XX	
i_{22}-01.51–i_{21}-00.61	XX	
40.38	XX	
28.94	XX	

In Table 15, the composite features are scored higher than their constituents in MCFS as compared to Table 14, but not at the top, where one might expect to find them. In similar experiments, MCFS values the perfectly correlated composite somewhere between 60% and 70% as compared to single features. Random Reducts in this case follows the same pattern as MCFS when applied without further measurement methods.

Table 16: $i_1 = 100\%$ $i_2 = 80\%$ with composites, normalized

MCFS	Variation	RR
69.78	XX	i_{12}-00.58–i_{11}-00.14
i_{12}-00.58–i_{11}-00.14	1	
60.39	XX	
51.87	XX	
45.10	XX	
i_{22}-01.51–i_{21}-00.61	XX	
40.38	XX	
28.94	XX	

Applying normalization to Table 15, the resultant Table 16 shows Random Reducts reducing the ranking to the composite solely. Of note is that Random Reducts finds the composite sufficient, ranking it higher than MCFS even in composite form.

Table 17: $i_1 = 100\%$ $i_2 = 80\%$ $i_3 = 60\%$ $i_7 - i_8 = 40\%$ without composites

MCFS	Variation	RR
58.51	–	58.51
51.40	–	51.40
45.99	–	45.99
36.90	–	36.90
i_{11}-02.38	5	23.34
28.13	XX	i_{32}-02.53
23.34	XX	i_{22}-02.51
i_{12}-03.87	1	i_{21}-03.21
i_{22}-02.51	-2	i_{12}-03.87
i_{21}-03.21	-2	i_{11}-02.38
i_{31}-02.73	XX	
i_{32}-02.53	XX	

Looking at Table 17, the interactive elements are speckled, due to the scoring methods applied by the two ranking algorithms. In this case the frequency of the interactive pairs is insufficient to top the Random Reducts ranking.

Table 18: $i_1 = 100\%$ $i_2 = 80\%$ $i_3 = 60\%$ $i_4 = 40\%$ without composites, normalized

MCFS	Variation	RR
58.51	–	i_{12}-03.87
51.40	–	i_{11}-02.38
45.99	XX	
36.90	XX	
i_{11}-02.38	-3	
28.13	XX	
23.34	XX	
i_{12}-03.87	-7	
i_{22}-02.51	XX	
i_{21}-03.21	XX	
i_{31}-02.73	XX	
i_{32}-02.53	XX	

Again, with normalization to account for the appearance frequency, in Table 18 Random Reducts picks up on the interaction pairing i_1, without the need for composites.

Table 19: $i_1 = 100\%$ $i_2 = 80\%$ $i_3 = 60\%$ $i_4 = 40\%$ with composites

MCFS	Variation	RR
i_{12}-03.87–i_{11}-02.38	–	i_{12}-03.87–i_{11}-02.38
51.40–45.99	XX	
58.51	XX	
36.90	XX	
28.13	XX	
23.34	XX	
i_{22}-02.51	XX	
i_{21}-03.21	XX	
i_{31}-02.73	XX	
i_{32}-02.53	XX	

Using a harsh optimality filter, there are two composites in Table 19, while the remaining interaction pairs are left as single features. Note that the second composite consists of two features that were not created as an interaction pair.

Table 20: Histone modification subset, first five rows only.

H3K27me3	H3K27me3.prec	H3K27me3.succ	H3K4me2	H3K4me2.prec	H3K4me2.succ	H3K79me3	H3K79me3.prec	RRH3K79me3.succ	H3R2me1	H3R2me1.prec	H3R2me1.succ	EXPR
0	0	0	0	0	0	0	0	0	0	0	0	1
0	1	0	0	0	0	0	1	1	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	0	1	1	0

Table 20 shows a subset of a histone modification dataset. The twelve features included were chosen as they were likely candidates for interactive behaviour.

Table 21: Histone modification subset after two iterations (1.2 and 1.5)

H3K4me2.succ	H3R2me1.succ	H3K27me3.succ–H3R2me1.prec	H3K4me2–H3K27me3–H3R2me1.prec	H3K79me3–H3K27me3–H3R2me1.prec	H3K79me3.prec–H3K27me3–H3R2me1.prec	H3K79me3.succ–H3K27me3–H3R2me1.prec	H3R2me1–H3K27me3–H3R2me1.prec	H3K27me3–H3R2me1.prec–H3K27me3.prec–H3R2me1.prec	H3K27me3–H3R2me1.prec–H3K27me3.prec–H3K4me2.prec	EXPR
0	0	00	000	000	000	000	000	0000	0000	1
0	0	00	000	000	100	100	000	0010	0010	1
0	0	00	000	000	000	000	000	0000	0000	1
0	0	00	000	000	000	000	000	0000	0000	1
0	1	11	001	001	001	001	001	0101	0100	0

Table 21 shows the first rows of Table 20 after being processed with the composite generator twice, first with a minimum increase boundary of 20% (1.2 multiplier) and after that once more with a minimum increase boundary of 50% (1.5 multiplier). The original twelve features have combined into ten, signifying that some features take part in many interactions.

Table 22: Histone modification subset after two iterations (1.2 and 1.6)

H3K4me2.succ	H3K79me3	H3K79me3.prec	H3K79me3.succ	H3R2me1	H3R2me1.succ	H3K27me3.prec–H3R2me1.prec	H3K27me3.succ–H3R2me1.prec	H3K4me2–H3K27me3–H3R2me1.prec	H3K27me3–H3R2me1.prec–H3K27me3.prec–H3K4me2.prec	EXPR
0	0	0	0	0	0	00	00	000	0000	1
0	0	1	1	0	0	10	00	000	0010	1
0	0	0	0	0	0	00	00	000	0000	1
0	0	0	0	0	0	00	00	000	0000	1
0	0	0	0	0	1	01	11	001	0100	0

Table 22 shows the first rows of the same subset as Table 20, but after two iterations of the composite generator, with the second iteration running with a minimum increase boundary of 60%. When compared to Table 21, the differences are significant.

Table 23: Ranking of Table 21

MCFS	Variation	RR
H3K27me3–H3R2me1.prec–H3K27me3.prec–H3K4me2.prec	XX	
H3K4me2–H3K27me3–H3R2me1.prec	XX	
H3K79me3–H3K27me3–H3R2me1.prec	XX	
H3K27me3–H3R2me1.prec–H3K27me3.prec–H3R2me1.prec	XX	
H3K27me3.succ–H3R2me1.prec	XX	
H3K79me3.succ–H3K27me3–H3R2me1.prec	XX	
H3K79me3.prec–H3K27me3–H3R2me1.prec	XX	
H3K4me2.succ	XX	
H3R2me1.succ	XX	
H3R2me1–H3K27me3–H3R2me1.prec	XX	

Table 23 shows the results of the histone modification subset from Table 21 when ranked with MCFS and Random Reducts. Random Reducts did not find any features above the cutoff point with this dataset.

6 Discussion

The ranking of composites with MCFS, and the definition of decision trees, show that decision trees do not score features entire, but rather a particular value of a feature's value domain. This may not always be a desired trait as evidenced by the results, where the composites would be expected to come out first, given their superior correlation. As such, there may be better options for ranking datasets with a mix of large and small value domain features, to avoid this tilting. Random Reducts, while finding interactive pairs with greater success, also showed inconsistent behaviour, sometimes listing one or more features of lower correlation above the interactive pairs. While this disappeared with the use of normalization, the behaviour of the algorithm should be as consistent and dependable as possible.

The boundaries and optimality filter are good ways to reduce the computational requirements of the module, but the process can still be prohibitively large. As such, further restraints might prove useful, though on what parameters they would apply is yet to be discovered. Potentially, an early ranking of composites might serve as a dynamic boundary for single features, requiring they be uncorrelated enough that an interaction truly matter. This would reduce the interaction finding capacity overall, but the gains in speed might be worth the trade. Furthermore, such an approach would not impact the results of a Random Reducts ranking.

A dynamic scan for interactive behaviour of arbitrary size would greatly improve the useability of the module. However, the method of scanning must be efficient, since a dataset can generate up to $\frac{n^2}{2}$ combinations just with two-feature composites. Such a development must however wait until the module can be rewritten onto a platform more suitable for high-resource computations. The traditional approaches of stopping after a certain amount of time would work, but leaves a level of uncertainty best avoided.

The results from Table 23 suggest that composite generation may, independently of ranking algorithms, make visible key components that participate in multiple interactive sets. This area of interactive behaviour has many uses in biological applications, such as the impact of pleiotropic pathway factors (Chavali et al., 2010). This is an aspect of the generator that would benefit from deeper study and analysis, and the systematic integration of this quality into the program would add another spectrum of functionality.

7 Conclusions

The composite generator shows that as a proof-of-concept, Relative Risk estimations can be used to predict likely interactive capacity. Furthermore, effective boundaries and filters can reduce the computation time required for these predictions. While not a tool for anyone to use, a proper understanding of the mechanics employed will allow for practical and feasible results. There are also multiple possible extensions to further the functionality and useability of the module.

Perl has proven an unsuitable platform to build upon for this particular project, and it's impedance has hindered the application of the composite generator on larger scale datasets. Thus, in order to take full advantage of the algorithms described herein, a rewrite is recommended onto a more suitable platform, such as C. Given the amplitude of independent data, threading and shared memory reuse would significantly improve performance speed while reducing the memory footprint. This is the logical step to converting the module from a proof-of-concept to a production band application.

The results indicate an increased visibility of composites to the human eye in both ranking algorithms. In MCFS, the ranking also becomes more stable, as only one feature is required in the split to be measured. Note however that the valuation of the composite is not consistent with the correlation specified in the synthetic dataset generator, as it's value domain has increased in size.

Independently of the ranking algorithms, the composite generator shows which features take part in multiple interactions. While not describing the strength of these, knowing which feature is the cornerstone in multiple properties or processes is a valuable tool in biological systems.

When used in conjunction with Monte Carlo Feature Selection, the composite generator can explain the reasons for weak features appearing in the higher rankings, where previously these interactive features may have been interpreted as individually strong. This is likely to apply to all ranking algorithms that rely on decision trees for feature scoring without compensating for interactive behaviour.

8 Acknowledgements

I would like to thank Marcin Kruzcyk for his assistance, both with Rosetta and the general complexities of Bioinformatics. My gratitude also extends to Susanne Bornelöv for her patience and help with all the practical details of Relative Risk, Perl, Unix, and lunch. And last, my appreciation of Prof. Jan Komorowski for giving me this opportunity to have fun while experiencing the academic life.

References

V. Bevick, L. Cheek, and J. Ball. Statistics review 11: Assessing risk. *Critical Care*, 8 (4):287–291, 2004. doi:10.1186/cc2908.

S. Chavali, F. Barrenäs, K. Kanduri, and M. Benson. Network properties of human disease genes with pleiotropic effects. *BMC Systems Biology*, 4(1):78, 2010. ISSN 1752-0509. doi: 10.1186/1752-0509-4-78.

M. Dramiński, A. Rada-Iglesias, S. Enroth, C Wadelius, J. Koronacki, and J. Komorowski. Monte carlo feature selection for supervised classification. *Bioinformatics*, 24(1):110117, 2008. doi:10.1093/bioinformatics/btm486.

A. Droit, G. G. Poirier, and J. M. Hunter. Experimental and bioinformatic approaches for interrogating proteinprotein interactions to determine protein function. *Journal of Molecular Endocrinology*, 34:263–280, 2005. doi: 10.1677/jme.1.01693.

R. A. Fisher. The design of experiments. 1935.

M. Kruczyk, N. Baltzer, J. Mieczkowski, J. Koronacki, and J. Komorowski. Random reducts: A monte carlo rough set-based method for feature selection in large datasets. *Fundamenta Informaticae*, 127:273 – 288, 2013. doi: 10.3233/FI-2013-909.

E. L. Lawler and D. E. Wood. Branch-and-bound methods: A survey. *Operations Research*, 14(4):699 – 719, 1996. doi: 10.1287/opre.14.4.699.

R.C. Lewontin. Genotype and phenotype. In Editors in Chief: Neil J. Smelser and Paul B. Baltes, editors, *International Encyclopedia of the Social & Behavioral Sciences*, pages 6159 – 6162. Pergamon, Oxford, 2001. ISBN 978-0-08-043076-8. doi: 10.1016/B0-08-043076-7/03077-1.

A. Øhrn. Rosetta: A collection of classes and routines for empirical modelling and data mining. `http://rosetta.sourceforge.net/`, 1996.

K. Pearson. On the criterion that a given system of deviations from the probable in the case of correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling. *Philosophical Magazine*, 50(302):157 – 175, 1900. doi: 10.1080/14786440009463897.

E. J. G. Pitman. Significance tests which may be applied to samples from any populations. *Supplement to the Journal of the Royal Statistical Society*, 4(1):119 – 130, 1937.

W. R. Rudnicki and J. Komorowski. Feature synthesis and extraction for the construction of generalized properties of amino acids. 3066:786–791, 2004. doi: 10.1007/978-3-540-25929-9_100. URL `http://dx.doi.org/10.1007/978-3-540-25929-9_100`.

S. R. Safavian. A survey of decision tree classifier methodology. *Systems, Man and Cybernetics, IEEE Transactions on*, 21(3):660 – 674, 1991. doi: 10.1109/21.97458.

D. Slezak. Approximate reducts in decision tables, 1996.

V. Steiß, T. Letschert, Helmut Schäfer, and R. Pahl. Permory-mpi: A program for high-speed parallel permutation testing in genome-wide association studies. *Bioinformatics*, 28(8):1168–1169, 2012. doi: 10.1093/bioinformatics/bts086.

G. Swaminath, Y. Xiang, T. W. Lee, J. Steenhuis, C. Parnot, and B. K. Kobilka. Sequential binding of agonists to the beta2 adrenoceptor. kinetic evidence for intermediate conformational states. *Journal of Biological Chemistry*, 279:686 – 691, 2004. doi: 10.1074/jbc.M310888200.