MDFS: MultiDimensional Feature Selection in R

by Radosław Piliszek, Krzysztof Mnich, Szymon Migacz, Paweł Tabaszewski, Andrzej Sułecki, Aneta Polewko-Klim, and Witold Rudnicki

Abstract Identification of informative variables in an information system is often performed using simple one-dimensional filtering procedures that discard information about interactions between variables. Such an approach may result in removing some relevant variables from consideration. Here we present an R package **MDFS** (MultiDimensional Feature Selection) that performs identification of informative variables taking into account synergistic interactions between multiple descriptors and the decision variable. **MDFS** is an implementation of an algorithm based on information theory (Mnich and Rudnicki, 2017). The computational kernel of the package is implemented in C++. A high-performance version implemented in CUDA C is also available. The application of **MDFS** is demonstrated using the well-known Madelon dataset, in which a decision variable is generated from synergistic interactions between descriptor variables. It is shown that the application of multidimensional analysis results in better sensitivity and ranking of importance.

Introduction

Identification of variables that are related to the decision variable is often the most important step in dataset analysis. In particular, it becomes really important when the number of variables describing the phenomena under scrutiny is large.

Methods of feature selection fall into three main categories (Guyon and Elisseeff, 2003):

- filters, where the identification of informative variables is performed before data modelling and analysis,
- wrappers, where the identification of informative variables is achieved by analysis of the models,
- embedded methods, which evaluate utility of variables in the model and select the most useful variables.

Filters are designed to provide a quick answer and therefore are the fastest. On the other hand, their simplicity is also the source of their errors. The rigorous univariate methods, such as *t*-test, do not detect interactions between variables. Heuristical methods that avoid this trap, such as Relief-f algorithm (Kononenko, 1994), may be biased towards weak and correlated variables (Robnik-Šikonja and Kononenko, 2003). Interesting heuristical filter based on decision trees – Monte Carlo Feature Selection (MCFS) (Dramiński et al., 2007; Dramiński and Koronacki, 2018) – avoids this pitfall. However, it may fail to find purely synergistic variables. Several filtering methods are designed to return only the non-redundant subset of variables (Zhao and Liu, 2007; Peng et al., 2005; Wang et al., 2013). While such methods may lead to very efficient models, their selection may be far from the best when one is interested in deeper understanding of the phenomena under scrutiny.

The wrapper algorithms are designed around machine learning algorithms such as SVM (Cortes and Vapnik, 1995), as in the SVM-RFE algorithm (Guyon et al., 2002), or random forest (Breiman, 2001), as in the Boruta algorithm (Kursa et al., 2010). They can identify variables involved in non-linear interactions. Unfortunately, for systems with tens of thousands of variables they are slow. For example, the Boruta algorithm first expands the system with randomised copies of variables and then requires numerous runs of the random forest algorithm.

The embedded methods are mostly limited to linear approximations and are part of a modelling approach where the selection is directed towards the utility of the model (**??**). Therefore, variables that are relevant for understanding the phenomena under scrutiny may be omitted and replaced by variables more suitable for building a particular model.

Here we introduce an R package implementing a filter based on information theory. The algorithm can identify synergistic relevant variables by performing an exhaustive search of low-dimensional combinations of variables.

Theory

Kohavi and John proposed that a variable $x_i \in X$, where X is a set of all descriptive variables, is weakly relevant if there exists a subset of variables $X_{sub} \subset X : x_i \notin X_{sub}$ that one can increase information on

the decision variable y by extending this subset with the variable x_i (Kohavi and John, 1997). Mnich and Rudnicki introduced the notion of k-weak relevance, that restricts the original definition by Kohavi and John to $(k-1)$-element subsets X_{sub} (Mnich and Rudnicki, 2017).

The algorithm implements the definition of k-weak relevance directly by exploring all possible k-tuples of variables $x_i \cup \{x_{m_1}, x_{m_2}, \ldots, x_{m_{k-1}}\}$ for k-dimensional analysis. For example, in 2 dimensions we explore a set of all possible pairs of variables. For each variable x_i we check whether adding it to another variable x_k adds information to the system. If there exists such x_k, then we declare x_i as 2-weakly relevant.

The maximum decrease in conditional information entropy upon adding x_i to description, normalized to sample size, is used as the measure of x_i's relevance:

$$IG^k_{max}(y; x_i) = N \max_m \left(H\left(y|x_{m_1}, x_{m_2}, \ldots, x_{m_{k-1}}\right) - H\left(y|x_i, x_{m_1}, x_{m_2}, \ldots, x_{m_{k-1}}\right) \right), \qquad (1)$$

where H is (conditional) information entropy and N is the number of observations. Difference in (conditional) information entropy is known as (conditional) mutual information. It is multiplied by N to obtain the proper null-hypothesis distribution. To name this value we reused the term information gain (IG) which is commonly used in information-theoretic context to denote different values related to mutual information.

To declare a variable k-weakly relevant it is required that its $IG^k_{max}(y; x_i)$ is statistically significant. This can be established via a comparison:

$$IG^k_{max}(y; x_i) \geq IG_{lim}, \qquad (2)$$

where IG_{lim} is computed using a procedure of fitting the theoretical distribution to the data.

For a sufficiently large sample, the value of IG for a non-informative variable, with respect to a single k-tuple, follows a χ^2 distribution. $IG^k_{max}(y; x_i)$, which is the maximum value of IG among many trials, follows an extreme value distribution. This distribution has one free parameter corresponding to the number of independent tests which is generally unknown and smaller than the total number of tests. The parameter is thus computed empirically by fitting the distribution to the irrelevant part of the data (Mnich and Rudnicki, 2017). This allows to convert the IG^k_{max} statistic to its p-value and then to establish IG_{lim} as a function of significance level α. Since many variables are investigated, the p-value should be adjusted using well-known FWER (Holm, 1979) or FDR (**?**) control technique. Due to unknown dependencies between tests, for best results we recommend using Benjamini-Hochberg-Yekutieli method (Benjamini and Yekutieli, 2001)[1] when performing FDR control.

In one dimension ($k = 1$) Equation 1 reduces to:

$$IG^1_{max}(y; x_i) = N\left(H(y) - H(y|x_i)\right), \qquad (3)$$

which is a well-known G-test statistic (**?**).

All variables that are weakly relevant in one-dimensional test should also be discovered in higher-dimensional tests, nevertheless their relative importance may be significantly influenced by interactions with other variables. Often the criterium for inclusion to further steps of data analysis and model building is simply taking the top n variables, therefore the ordering of variables due to importance matters as well.

Algorithm and implementation

The **MDFS** package (Piliszek et al., 2018) consists of two main parts. One is an R interface to two computational engines. These engines utilise either CPU or NVIDIA GPU and are implemented in standard C++ and in CUDA C, respectively. Either computational engine returns the IG^k_{max} distribution for a given dataset plus requested details which may pose an interesting insight into data. The other part is a toolkit to analyse results. It is written entirely in R. The version of **MDFS** used and described here is 1.0.3. The term 'MDFS' (MultiDimensional Feature Selection) is used to denote the analysis, method and algorithm presented in this article as well.

The IG^k_{max} for each variable is computed using a straightforward algorithm based on Equation 1. Information entropy (H) is computed using discretised descriptive variables. Discretisation is performed using customisable randomised rank-based approach. To control the discretisation process we use a concept of range. Range is a real number between 0 and 1 affecting the share each discretised variable class has in the dataset. Each share is sampled from a uniform distribution on the interval $(1 - \text{range}, 1 + \text{range})$. Hence, range $= 0$ results in an equipotent split, range $= 1$ equals a completely

---

[1] Method "BY" for `p.adjust` function.

random split. Let us assume that there are N objects in the system and we want to discretise a variable to c classes. To this end, $(c-1)$ distinct integers from the range $(2, N)$ are obtained using computed shares. Then, the variable is sorted and values at positions indexed by these integers are used to discretise the variable into separate classes. In most applications of the algorithm there is no default best discretisation of descriptive variables, hence multiple random discretisations are performed. The IG_{max}^k is computed for each discretisation, then the maximum IG_{max}^k over all discretizations is returned. Hence, the final returned IG_{max}^k is a maximum over both tuples and discretisations.

The problem of selecting the right amount of classes (the right value of c) is similar to bias–variance tradeoff but more subtle. The statistic is better when there are less classes (binary being the best case) but the shape ought to be better when there are more classes as it improves the resolution. When the right split is known (as we show later with Madelon), it is best to use it. Otherwise we recommend to try different numbers of classes and do several random discretizations for each.

Conditional information entropy is obtained from the experimental probabilities of a decision class using the following formula:

$$H\left(y|x_1, \ldots, x_k\right) = -\sum_{d=0,1} \sum_{i_1=1:c} \cdots \sum_{i_k=1:c} p_{i_1,\ldots,i_k}^d \log\left(p_{i_1,\ldots,i_k}^d\right), \quad (4)$$

where $p_{i_1,\ldots,i_k}^d$ denotes the conditional probability of class d in a k-dimensional voxel with coordinates i_j. Note that the number of voxels in k dimensions is c^k, where c is the number of classes of discretised descriptive variables. To this end, one needs to compute the number of instances of each class in each voxel. The conditional probability of class d in a voxel is then computed as

$$p_{i_1,\ldots,i_k}^d = \frac{N_{i_1,\ldots,i_k}^d + \beta^d}{N_{i_1,\ldots,i_k}^0 + \beta^0 + N_{i_1,\ldots,i_k}^1 + \beta^1}, \quad (5)$$

where $N_{i_1,\ldots,i_k}^d$ is the count of class d in a k-dimensional voxel with coordinates i_j and β^d is a pseudo-count corresponding to class d:

$$\beta^d = \xi \frac{N^d}{\min\left(N^0, N^1\right)}, \quad (6)$$

where $\xi > 0$ can be supplied by the user. The default value is set to 0.25. It was obtained in an experimental process to achieve the best fit to χ^2 distribution. Usual usage should not mandate the need to change ξ.

The implementation of the algorithm is currently limited to binary decision variables. The analysis for information systems that have more than two categories can be performed either by executing all possible pairwise comparisons or one-vs-rest. Then all variables that are relevant in the context of a single pairwise comparison should be considered relevant. In the case of continuous decision variable one must discretise it before performing analysis. In the current implementation all variables are discretised into an equal number of classes. This constraint is introduced for increased efficiency of computations, in particular on a GPU.

Another limitation is the maximum number of dimensions set to 5. This is due to several reasons. Firstly, the computational cost of the algorithm is proportional to number of variables to power equal the dimension of the analysis, and it becomes prohibitively expensive for powers larger than 5 even for systems described with a hundred of variables. Secondly, analysis in higher dimensions requires a substantial number of objects to fill the voxels sufficiently for the algorithm to detect real synergies. Finally, it is also related to the simplicity of efficient implementation of the algorithm in CUDA.

The most time-consuming part of the algorithm is computing the counters for all voxels. Fortunately, this part of the computations is relatively easy to parallelise, as the exhaustive search is very well suited for GPU. Therefore, a GPU version of the algorithm was developed in CUDA C for NVIDIA GPGPUs and is targeted towards problems with a very large number of features. The CPU version is also parallelised to utilise all cores available on a single node. The 1D analysis is available only in the CPU version since there is no benefit in running this kind of analysis on GPU.

Package functions introduction

There are three functions in the package which are to be run directly with the input dataset: `MDFS`, `ComputeMaxInfoGains`, and `ComputeInterestingTuples`. The first one, `MDFS`, is our recommended function for new users, since it hides internal details and provides an easy to use interface for basic end-to-end analysis for current users of other statistical tests (e.g., `t.test`) so that the user can straightforwardly get the statistic values, p-values, and adjusted p-values for variables from input. The other two functions are interfaces to the IG-calculating lower-level C++ and CUDA C++ code.

`ComputeMaxInfoGains` returns the max IGs, as described in the theory section. It can optionally provide information about the tuple in which this max IG was observed. On the other hand, one might be interested in tuples where certain IG threshold has been achieved. The `ComputeInterestingTuples` function performs this type of analysis and reports which variable in which tuple achieved the corresponding IG value.

The `ComputePValue` function performs fitting of IGs to respective statistical distributions as described in the theory section. The goodness of fit is tested using Kolmogorov-Smirnov one-sample test and a warning is emitted if the threshold is exceeded. `ComputePValue` returns an object of the `"MDFS"` class which contains, in particular, *p*-values for variables. This class implements various methods for handling output of statistical analysis. In particular they can plot details of IG distribution, output *p*-values of all variables, and output relevant variables. `ComputePValue` is implemented in a general way, extending beyond limitations of the current implementation of `ComputeMaxInfoGains`. In particular, it can handle multi-class problems and different number of divisions for each variable.

The `AddContrastVariables` is an utility function used to construct contrast variables (Stoppiglia et al., 2003; Kursa et al., 2010). Contrast variables are used solely for improving reliability of the fit of statistical distribution. In the case of fitting distribution to contrast variables we know exactly how many irrelevant variables there are in the system. The contrast variables are not tested for relevance and hence not used when adjusting *p*-values to not decrease the sensitivity without reason.

Canonical package usage

As mentioned earlier, the recommended way to use the package is to use the `MDFS` function. It uses the other packaged functions to achieve its goal in the standard and thoroughly tested way, so it may be considered the canonical package usage pattern. The `MDFS` function is general in terms of contrast variables being optional, hence let us examine a simplified version of it assuming the contrast variables are actually being used. We also neglect the setting of seed but we recommend it to be set so that the result is reproducible. The `MDFS` wrapper does accept a seed and saves it with the result.

The first step is to build the contrast variables:

```
contrast <- AddContrastVariables(data, n.contrast)
```

In the next step, the compute-intensive computation of IGs is executed:

```
MIG.Result <- ComputeMaxInfoGains(contrast$x, decision,
  dimensions = dimensions, divisions = divisions,
  discretizations = discretizations, range = range, pseudo.count = pseudo.count)
```

The first two positional parameters are respectively the feature data (plus contrast variables) and the decision. The other parameters decide on the type of computed IGs: `dimensions` controls dimensionality, `divisions` controls the number of classes in the discretisation (it is equal to `divisions`+1), `discretizations` controls the number of discretisations, `range` controls how random the discretisation splits are, and `pseudo.count` controls the regularization parameter ζ for pseudocounts.

Finally, the computed IGs are analysed and a statistical result is computed:

```
fs <- ComputePValue(MIG.Result$IG,
  dimensions = dimensions, divisions = divisions,
  contrast.mask = contrast$mask,
  one.dim.mode = ifelse (discretizations==1, "raw",
                      ifelse(divisions*discretizations<12, "lin", "exp")))

statistic <- MIG.Result$IG[!contrast$mask]
p.value <- fs$p.value[!contrast$mask]
adjusted.p.value <- p.adjust(p.value, method = p.adjust.method)
relevant.variables <- which(adjusted.p.value < level)
```

The `one.dim.mode` parameter controls the expected distribution in 1D. The rule states that as long as we have 1 discretisation the resulting distribution is chi-squared, otherwise, depending on the product of `discretizations` and `divisions`, the resulting distribution might be closer to a linear or exponential, as in higher dimensions, function of chi-squared distributions. This is heuristic and might need to be tuned. Features with adjusted *p*-values below some set level are considered to be relevant.

Testing the Madelon dataset

For demonstration of the **MDFS** package we used the training subset of the well-known Madelon dataset (Guyon et al., 2007). It is an artificial set with 2000 objects and 500 variables. The decision was generated using a 5-dimensional random parity function based on variables drawn from normal distribution. The remaining variables were generated in the following way. Fifteen variables were obtained as linear combinations of the 5 input variables and remaining 480 variables were drawn randomly from the normal distribution. The data set can be accessed from the UCI Machine Learning Repository (Dheeru and Karra Taniskidou, 2017) and it is included in **MDFS** package as well.

We conducted the analysis in all possible dimensionalities using both CPU and GPU versions of the code. Additionally, a standard t-test was performed for reference. We examined computational efficiency of the algorithm and compared the results obtained by performing analysis in varied dimensionalities.

In the first attempt we utilised the given information on the properties of the dataset under scrutiny. We knew in advance that Madelon was constructed as a random parity problem and that each base variable was constructed from a distinct distribution. Therefore, we could use one discretisation into 2 equipotent classes. In the second attempt the recommended 'blind' approach in 2D was followed, which utilises several randomized discretisations.

For brevity, in the following examples the set of Madelon independent variables is named x and its decision is named y:

```
x <- madelon$data
y <- madelon$decision
```

For easier comparison we introduce a helper function to obtain, from MDFS analysis, the relevant features' indices in decreasing relevance (increasing p-value) order:

```
GetOrderedRelevant <- function (result) {
  result$relevant.variables[order(result$p.value[result$relevant.variables])]
}
```

One can now obtain p-values from t-test, adjust them using Holm correction (one of FWER corrections, the default in the p.adjust function), take relevant with level 0.05, and order them:

```
> tt <- ttests(x, ina = y+1)[,2] # we only use $p$-values (2nd column)
> tt.adjusted <- p.adjust(tt, method = "holm")
> tt.relevant <- which(tt.adjusted < 0.05)
> tt.relevant.ordered <- tt.relevant[order(tt.adjusted[tt.relevant])]
> tt.relevant.ordered
[1] 476 242 337  65 129 106 339  49 379 443 473 454 494
```

A FWER correction is used because we expect strong separation between relevant and irrelevant features in this artificial dataset. We used the ttests function from the **Rfast** (Papadakis et al., 2018) package as it is a version of t-test optimized for this purpose.

To achieve the same with **MDFS** for 1, 2, and 3 dimensions one can use the wrapper MDFS function:

```
> d1 <- MDFS(x, y, n.contrast = 0, dimensions = 1, divisions = 1, range = 0)
> d1.relevant.ordered <- GetOrderedRelevant(d1)
> d1.relevant.ordered
[1] 476 242 339 337  65 129 106  49 379 454 494 443 473

> d2 <- MDFS(x, y, n.contrast = 0, dimensions = 2, divisions = 1, range = 0)
> d2.relevant.ordered <- GetOrderedRelevant(d2)
> d2.relevant.ordered
[1] 476 242  49 379 154 282 434 339 494 454 452  29 319 443 129 473 106 337  65

> d3 <- MDFS(x, y, n.contrast = 0, dimensions = 3, divisions = 1, range = 0)
> d3.relevant.ordered <- GetOrderedRelevant(d3)
> d3.relevant.ordered
[1] 154 434 282  49 379 476 242 319  29 452 494 106 454 129 473 443 339 337  65 456
```

The changes in the relevant variables set can be examined with simple setdiff comparisons:

```
> setdiff(tt.relevant.ordered, d1.relevant.ordered)
integer(0)
> setdiff(d1.relevant.ordered, tt.relevant.ordered)
integer(0)
```

```
> setdiff(d1.relevant.ordered, d2.relevant.ordered)
integer(0)
> setdiff(d2.relevant.ordered, d1.relevant.ordered)
[1] 154 282 434 452  29 319
> setdiff(d2.relevant.ordered, d3.relevant.ordered)
integer(0)
> setdiff(d3.relevant.ordered, d2.relevant.ordered)
[1] 456
```

One may notice that ordering by importance leads to different results for these 4 tests.

In the above the knowledge about properties of the Madelon dataset was used: that there are many random variables, hence no need to add contrast variables, and that the problem is best resolved by splitting features in half, hence one could use 1 discretisation and set range to zero.

However, one is usually not equipped with such knowledge and then may need to use multiple random discretisations. Below an example run of 'blind' 2D analysis of Madelon is presented:

```
> d2b <- MDFS(x, y, dimensions = 2, divisions = 1, discretizations = 30, seed = 118912)
> d2b.relevant.ordered <- GetOrderedRelevant(d2b)
> d2b.relevant.ordered
[1] 476 242 379  49 154 434 106 282 473 339 443 452  29 454 494 319  65 337 129
> setdiff(d2b.relevant.ordered, d2.relevant.ordered)
integer(0)
> setdiff(d2.relevant.ordered, d2b.relevant.ordered)
integer(0)
```

This demonstrates that the same variables are discovered, yet with a different order.

Performance

The performance of the CPU version of the algorithm was measured on a computer with two Intel Xeon E5-2650 v2 processors, running at 2.6 GHz. Each processor has eight physical cores. Hyperthreading was disabled.

The GPU version was tested on a computer with a single NVIDIA Tesla K80 accelerator. The K80 is equipped with two GK210 chips and is therefore visible to the system as two separate GPGPUs. Both were utilised in the tests.

The Madelon dataset has moderate dimensionality for modern standards, hence it is amenable to high-dimensional analysis. The CPU version of the code handles analysis up to four dimensions in a reasonable time, see Table 1.

The performance gap between CPU and GPU versions is much higher than suggested by a simple comparison of hardware capabilities. This is due to two factors. Firstly, the GPU version has been highly optimised towards increased efficiency of memory usage. The representation of the data by bit-vectors and direct exploitation of the data locality allows for much higher data throughput. What is more, the bit-vector representation allows for using very efficient *popcnt* instruction for counter updates. On the other hand the CPU version has been written mainly as a reference version using a straightforward implementation of the algorithm and has not been strongly optimised.
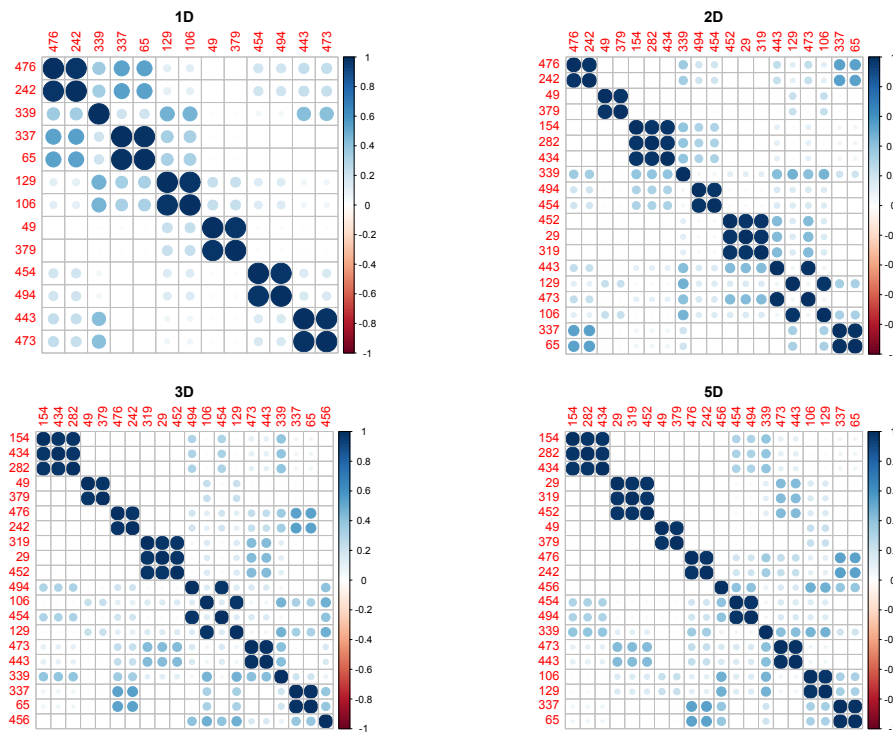
	t-test	1D	2D	3D	4D	5D
CPU	0.01s	0.01s	0.44s	42s	1h:58m	249h
GPU	-	-	0.23s	0.2s	9.8s	59m:37s

Table 1: Execution times for the Madelon dataset.

	1D	2D	3D	4D	5D
CPU	0.35s	5.8s	37m:11s	92h	-
GPU	-	2.9s	3.3s	7m:36s	42h

Table 2: Execution times for the Madelon dataset with 30 random discretisations.

Structure of the Madelon dataset revealed by MDFS analysis
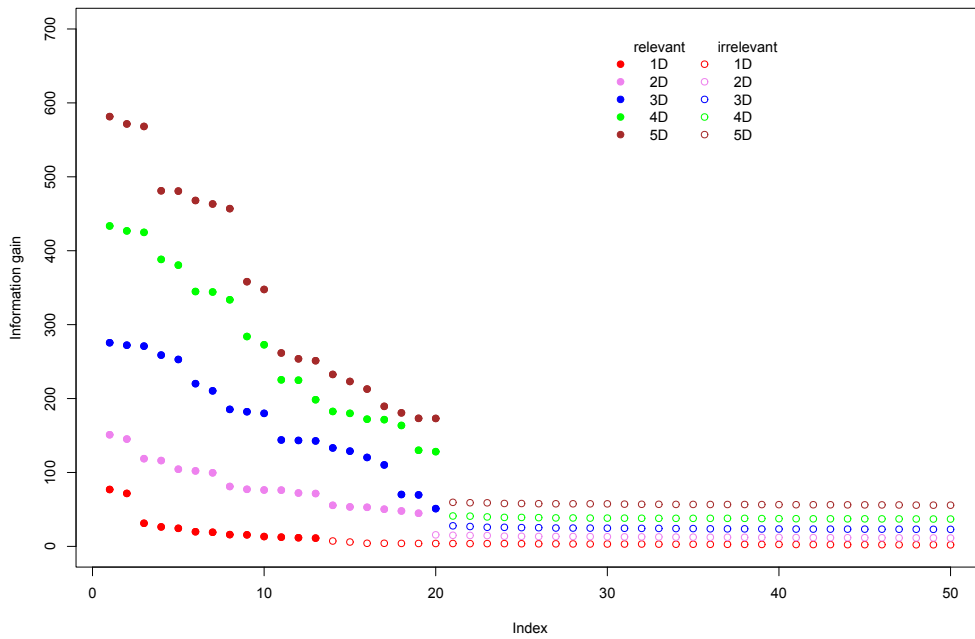

Figure 1: Correlation plots for relevant variables discovered in 1-, 2-, 3-, and 5-dimensional analysis of the Madelon dataset with one deterministic discretisation with division in the middle. The variables are ordered by IG.

Cluster	Members
154	154, 282, 434
29	29, 319, 452
49	49, 379
242	476, 242
456	456
454	454, 494
339	339
443	473, 443
106	106, 129
65	337, 65

Table 3: Discovered variable clusters (as seen in correlation plots) ordered by descending maximum relevance (measured with 5D IG), identified by the variable with the lowest number.

The twenty relevant variables in Madelon can be easily identified by analysis of histograms of variables, their correlation structure and by a priori knowledge of the method of construction of the dataset. In particular, base variables, i.e. these variables that are directly connected to a decision variable, have the unique distribution that has two distinct peaks. All other variables have smooth unimodal distribution, hence identification of base variables is trivial. What is more, we know that remaining informative variables are constructed as linear combinations of base variables, hence they should display significant correlations with base variables. Finally, the nuisance variables are completely random, hence they should not be correlated neither with base variables nor with their

Figure 2: Information gain obtained by the MDFS algorithm using 1-, 2-, 3-, 4-, and 5-dimensional variants of the algorithm for the Madelon dataset with one deterministic discretisation with division in the middle. Full circles represent variables deemed relevant. All variables are sorted by IG. Margin between irrelevant and relevant features grows with dimensionality.
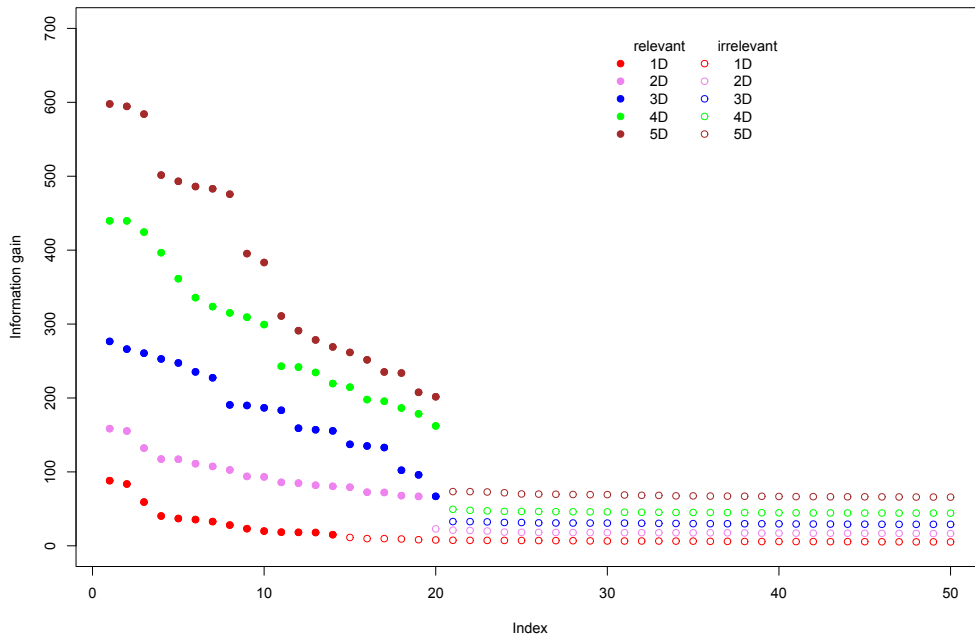
Figure 3: Information gain obtained by the MDFS algorithm using 1-, 2-, 3-, 4-, and 5-dimensional variants of the algorithm for the Madelon dataset with 30 random discretisations. Full circles represent variables deemed relevant. All variables are sorted by IG. The margin between irrelevant and relevant features grows with dimensionality.

linear combinations. The analysis of correlations between variables reveals also that there are several groups of very highly correlated ($r > 0.99$) variables, see Figure 1. All variables in such a group can be considered as a single variable, reducing the number of independent variables to ten. The entire group

Table 4: Summary of results for the Madelon dataset with one deterministic discretisation with division in the middle. The variable clusters are ordered by descending IG. The numbers of base variable clusters are highlighted in boldface. Clusters represented by 65 and 106, displayed in italic font, are deemed highly relevant in 1D analyses and the least relevant in 5D analysis.

	t-test	1D	2D	3D	4D	5D
1.	**242**	**242**	**242**	**154**	**154**	**154**
2.	*65*	339	**49**	**49**	**49**	**29**
3.	*106*	65	**154**	242	**29**	**49**
4.	339	*106*	339	**29**	**242**	**242**
5.	**49**	**49**	454	454	454	**456**
6.	443	454	**29**	*106*	339	454
7.	454	443	443	443	*106*	339
8.	-	-	*106*	339	**456**	443
9.	-	-	*65*	*65*	443	*106*
10.	-	-	-	**456**	*65*	*65*

Table 5: Summary of results for the Madelon dataset with 30 random discretisations. The variable clusters are ordered by descending IG. The numbers of base variable clusters are highlighted in boldface. Similar behaviour with 65 and 106 is observed as in the single discretisation case. Note the irrelevant variable 205 (underlined) discovered in 1D as relevant due to small margin between relevant and irrelevant features.

	t-test	1D	2D	3D	4D	5D
1.	**242**	**242**	**242**	**154**	**154**	**154**
2.	*65*	339	**49**	**49**	**49**	**29**
3.	*106*	65	**154**	242	**29**	**49**
4.	339	443	*106*	**29**	**242**	**242**
5.	**49**	*106*	443	*106*	454	454
6.	443	454	339	454	*106*	**456**
7.	454	**49**	**29**	443	339	443
8.	-	<u>205</u>	454	339	443	339
9.	-	-	*65*	*65*	**456**	*106*
10.	-	-	-	**456**	*65*	*65*

is further represented by the variable with the lowest number. The clusters are presented in Table 3.

This clear structure of the dataset creates an opportunity to confront results of the MDFS analysis with the ground truth and observe how the increasing precision of the analysis helps to discover this structure without using the a priori knowledge on the structure of the dataset.

One-dimensional analysis reveals 13 really relevant variables (7 independent ones), both by means of the *t*-test and using the information gain measure, see Table 4. Three-dimensional and higher-dimensional analyses find all 20 relevant variables. Additionally, with the exception of one-dimensional case, in all cases there is a clear separation between IG obtained for relevant and irrelevant variables, see Figure 2. This translates into a significant drop of *p*-value for the relevant variables.

Five variables, namely $\{29, 49, 154, 242, 456\}$, are clearly orthogonal to each other, hence they are the base variables used to generate the decision variable. Five other variables are correlated with base variables and with each other, and hence they are linear combinations of base variables. The

one-dimensional analyses, both *t*-test and mutual-information approach, find only two base variables, see Table 4. What is more, while one of them is regarded as highly important (lowest *p*-value), the second one is considered only the 5th most important out of 7. Two-dimensional analysis finds 4 or 5 base variables, depending on the method used. Moreover, the relative ranking of variables is closer to intuition, with three base variables on top. The relative ranking of importance improves with increasing dimensionality of the analysis. In 5-dimensional analysis all five base variables are scored higher than any linear combination. In particular, the variable 456, which is identified by 3D analysis as the least important, rises to the eighth place in 4D analysis and to the fifth in 5D. Interestingly, the variable 65, which is the least important in 5D analysis is the second most important variable in *t*-test and the third most important variable in 1D.

Conclusion

We have introduced a new package for identification of informative variables in multidimensional information systems which takes into account interactions between variables. The implemented method is significantly more sensitive than the standard *t*-test when interactions between variables are present in the system. When applied to the well-known five-dimensional problem—Madelon—the method not only discovered all relevant variables but also produced the correct estimate of their relative relevance.

Acknowledgments

Bibliography

Y. Benjamini and D. Yekutieli. The control of the false discovery rate in multiple testing under dependency. *The Annals of Statistics*, 29(4):1165–1188, 2001. [p2]

L. Breiman. Random Forests. *Machine Learning*, 45:5–32, 2001. URL https://doi.org/10.1023/a:1010933404324. [p1]

C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995. URL https://doi.org/10.1007/bf00994018. [p1]

D. Dheeru and E. Karra Taniskidou. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml. [p5]

M. Dramiński and J. Koronacki. Rmcfs: An r package for monte carlo feature selection and interdependency discovery. *Journal of Statistical Software, Articles*, 85(12):1–28, 2018. ISSN 1548-7660. URL https://doi.org/10.18637/jss.v085.i12. [p1]

M. Dramiński, A. Rada-Iglesias, S. Enroth, C. Wadelius, J. Koronacki, and J. Komorowski. Monte carlo feature selection for supervised classification. *Bioinformatics*, 24(1):110–117, 2007. ISSN 1367-4803. [p1]

I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3(Mar):1157–1182, 2003. URL https://doi.org/10.1162/153244303322753616. [p1]

I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Machine learning*, 46(1-3):389–422, 2002. URL https://doi.org/10.1023/a:1012487302797. [p1]

I. Guyon, J. Li, T. Mader, P. A. Pletscher, G. Schneider, and M. Uhr. Competitive Baseline Methods Set New Standards for the NIPS 2003 Feature Selection Benchmark. *Pattern Recognition Letters*, 28(12):1438–1444, 2007. URL https://doi.org/10.1016/j.patrec.2007.02.014. [p5]

S. Holm. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6(2):65–70, 1979. ISSN 03036898, 14679469. URL https://doi.org/10.2307/4615733. [p2]

R. Kohavi and G. H. John. Wrappers for feature subset selection. *Artif. Intell.*, 97(1-2):273–324, 1997. ISSN 0004-3702. URL https://doi.org/10.1016/s0004-3702(97)00043-x. [p2]

I. Kononenko. Estimating attributes: Analysis and extensions of relief. In *European Conference on Machine Learning*, pages 171–182, 1994. URL https://doi.org/10.1007/3-540-57868-4_57. [p1]

M. B. Kursa, A. Jankowski, and W. R. Rudnicki. Boruta – a system for feature selection. *Fundamenta Informaticae*, 101(4):271–285, 2010. URL https://doi.org/10.3233/fi-2010-288. [p1, 4]

K. Mnich and W. R. Rudnicki. All-relevant feature selection using multidimensional filters with exhaustive search. *CoRR*, abs/1705.05756, 2017. URL http://arxiv.org/abs/1705.05756. [p1, 2]

M. Papadakis, M. Tsagris, M. Dimitriadis, S. Fafalios, I. Tsamardinos, M. Fasiolo, G. Borboudakis, J. Burkardt, C. Zou, K. Lakiotaki, and C. Chatzipantsiou. *Rfast: A Collection of Efficient and Extremely Fast R Functions*, 2018. URL https://CRAN.R-project.org/package=Rfast. R package version 1.9.2. [p5]

H. Peng, F. Long, and C. Ding. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8):1226–1238, 2005. URL https://doi.org/10.1109/tpami.2005.159. [p1]

R. Piliszek, K. Mnich, P. Tabaszewski, S. Migacz, A. Sułecki, and W. R. Rudnicki. *MDFS: MultiDimensional Feature Selection*, 2018. URL https://CRAN.R-project.org/package=MDFS. R package version 1.0.3. [p2]

M. Robnik-Šikonja and I. Kononenko. Theoretical and empirical analysis of ReliefF and RReliefF. *Machine Learning*, 53(1-2):23–69, 2003. URL https://doi.org/10.1023/a:1025667309714. [p1]

H. Stoppiglia, G. Dreyfus, R. Dubois, and Y. Oussar. Ranking a random feature for variable and feature selection. *Journal of Machine Learning Research*, 3(7-8):1399–1414, 2003. URL https://doi.org/10.1162/153244303322753733. [p4]

G. Wang, Q. Song, B. Xu, and Y. Zhou. Selecting feature subset for high dimensional data via the propositional foil rules. *Pattern Recognition*, 46(1):199–214, 2013. URL https://doi.org/10.1016/j.patcog.2012.07.028. [p1]

Z. Zhao and H. Liu. Searching for interacting features. In *IJCAI*, volume 7, pages 1156–1161, 2007. [p1]

Radosław Piliszek
Computational Centre, University of Bialystok
Konstantego Ciolkowskiego 1M, 15-245 Bialystok
Poland
0000-0003-0729-9167
r.piliszek@uwb.edu.pl

Krzysztof Mnich
Computational Centre, University of Bialystok
Konstantego Ciolkowskiego 1M, 15-245 Bialystok
Poland
0000-0002-6226-981X
k.mnich@uwb.edu.pl

Szymon Migacz
Interdisciplinary Centre for Mathematical and Computational Modelling, University of Warsaw
Pawińskiego 5A, 02-106 Warsaw
Poland

Paweł Tabaszewski
Interdisciplinary Centre for Mathematical and Computational Modelling, University of Warsaw
Pawińskiego 5A, 02-106 Warsaw
Poland

Andrzej Sułecki
Interdisciplinary Centre for Mathematical and Computational Modelling, University of Warsaw
Pawińskiego 5A, 02-106 Warsaw
Poland

Aneta Polewko-Klim
Institute of Informatics, University of Bialystok
Konstantego Ciolkowskiego 1M, 15-245 Bialystok

Poland
0000-0003-1987-7374
anetapol@uwb.edu.pl

Witold Rudnicki
Institute of Informatics, University of Bialystok
Konstantego Ciolkowskiego 1M, 15-245 Bialystok
Poland
and
Interdisciplinary Centre for Mathematical and Computational Modelling, University of Warsaw
Pawińskiego 5A, 02-106 Warsaw
Poland
0000-0002-7928-4944
w.rudnicki@uwb.edu.pl