IterReliefF: an iterative version of the ReliefF algorithm

Emilien Pilloud

CSE 527 final project report
University of Washington
December 15, 2017

Abstract

In this project, I evaluated the algorithm ReliefF as a feature selection method to classify gene expression levels in the leukemia sub-typing problem. This method had already been considered in previous works, usually in conjunction with other feature-selection techniques such as genetic algorithms. I implemented an extension of the ReliefF algorithm, called IterReliefF, which intends to suppress highly correlated features from the ReliefF output. The accuracy was not significantly improved although some slight amelioration is noticeable.

1. Introduction

With the rise of machine learning techniques those past decades and the increasingly accessible computing power, classifiers of all kind will begin to breach into our daily life. In the medical domain, it has been already in use for a long time, especially as clinical support decision systems to help physicians in their diagnosis process [1]. One decisive part of this machine learning predictive pipeline is the features selection or extraction. Indeed, when we train a model, we want to select a reasonable sized set of relevant features so that our model can achieve the best accuracy [2].
Furthermore, in a biological and/or medical genetic use, we deal with micro-arrays of gene expression levels which are vectors of length of around 20'000. It is clear that we must reduce that number of features by either choosing the most relevant ones for our purpose (selection) or combining some features together (extraction) or both. Without reduction, this high number of features would be highly impractical, would produce wrong models and lead a very slow data fitting. Even if our computational power is increasing, we want our medical prediction models to be able to run on most terrain hospitals computers and why not even on smart phones.
If we focus on the feature selection approach, there are three major kind of feature selection algorithms: filters, wrappers and embedded [3]. Filters select feature based only on the data and its intra-sec characteristics whereas wrappers rely on an underlying classifier. ReliefF is one of those filtering features selection method, which is the least computational needy kind. In this project, I try to improve the ReliefF algorithm by adding a correlation correction feature. I will test my implementation of ReliefF on a practical case: the classification of Leukemia gene expression levels. I will use some features selected by the algorithm and fit

them to a machine learning model.

2. Background

Features selection algorithm have been shown to consistently improve the accuracy of the classifier in the context of cancer gene expressions classification [2]. However, this accuracy also relies on the feature selection algorithm employed as well as on the classifier used for the testing. Unfortunately, not all the classifiers will react positively to some feature selection techniques [4].
The ReliefF algorithms family have already been used with relative success for such tasks. Yu and Liu [3] compare it to its own redundancy based filter technique and beats it whereas Hijazi and Chan [5] extend it with a genetic algorithm to reach a state-of-the-art gene selection. They also remind us that the sole use of filters and/or redundancy methods can lead to a less biological meaningful genes selection.
Researchers also extended ReliefF to suit wider purposes such as multi-label classification [6] or hierarchical classes structures [7]. Those extensions could be useful to our problem as the subtypes of Leukemia are hierarchically organized and can be seen as a multi-label problem. However, for simplicity, I decided to extend directly the ReliefF algorithm.
Finally, Wang et al. [8] implemented ReliefF for images classification purposes and explain that ReliefF is widely used especially because of it computational efficiency. They also point out interesting shortcomings of the algorithm: it lacks redundancy analysis and selecting k-nearest neighbors might add volatility to the output. They proposed solutions to remedy these problems by adding a redundancy analysis after the output of ReliefF and by inducing some randomness in the choice of the k-nearest neighbors.

3. Methods & Results

3.1. The ReliefF Algorithm

The ReliefF algorithm [9] is an algorithm of feature ranking from the filter family that has been already used with relative success in a computational biology context of classification of cancer gene expressions micro-arrays [5]. It is an extension from the original Relief [10] which could only handle binary classification problems. The main intuition behind it is to rank features accordingly to how well they differentiate a randomly chosen sample from the closest samples from others classes. That is, it penalizes features that vary a lot within a single class and values those that are the most correlated with their class.
Some of its advantages are that it is fairly noise resistant, is computationally efficient, is aware of the contextual information and do not consider the features to be independent as many filter do [11]. As aforementioned, the computational efficiency of ReliefF is appealing. In detail, the most computational part of the algorithm is to find the k-nearest hits or misses which can be done in $\mathrm{O}\big(N^2M\big)$ where N is the number of training instances and M is the number of features. As a matter of comparison, the Lasso regression could be applied to the gene selection as well. However, its running time is $\mathrm{O}\big(M^3\big)$ if $M >> N$ which is the case for

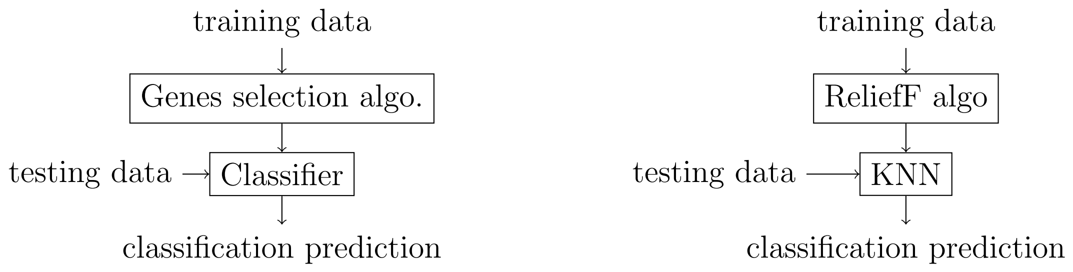
Algorithm 1 ReliefF [9]

```
procedure ReliefF($sampleSet, n, k$)
    weights[#features] $\leftarrow 0.0$
    for $i \leftarrow 1, n$ do
        $s \leftarrow$ randomSample(sampleSet)
        find k nearest hits $H_j$
        for each class C $\neq$ class(s) do
            find k nearest misses $M_j^C$
        for $a \leftarrow 1, \#features$ do
            weights[a] $\leftarrow$ weights[a] $- \sum_j^k diff(a, s, H_j)/(n \cdot k) + \sum_{C \neq class(s)} \left[ \frac{P(C)}{1-P(class(s)} \sum_j^k diff(a, s, M_j^{class(C)}) \right] /(n \cdot k)$
    return weights
```

our gene data.
It is also interesting to note that ReliefF is part of the "embarrassingly" parallel algorithms. Indeed, each iteration of the main for loop can be done independently from the others meaning that it can be easily parallelized as long as the implementation of the weights array is carefully atomic.

3.2. The basic training pipeline and my idea of improvement

A simplified but classical and functional work-flow for classifying supervised labeled data is showed on figure 1a below. We select some genes of interest and straight-forwardly fit them to the classification model which is in our case a k-nearest neighbors classifier.



(a) Simplistic Execution-flow (b) Simple case in my project

Figure 1: Simple classification work-flow

My idea of improvement would be to modify that work-flow to remedy to what appear to be the major flow of ReliefF: its lack of redundancy analysis [8]. I empirically tested the correlation of the output of the ReliefF algorithm and indeed some output features were highly correlated. To remedy to this issue, I propose an iterative version of the execution in figure 2. I suggest to run ReliefF on the training data until the stability of the first k ranked features. Then, I would apply some redundancy analysis according to the already selected

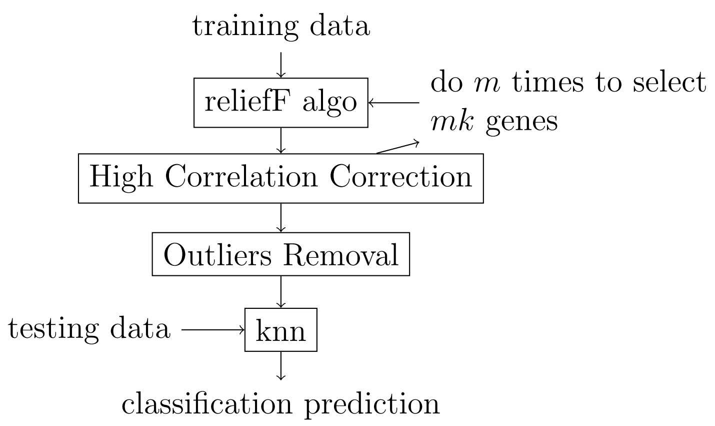

training data
reliefF algo
do m times to select mk genes
High Correlation Correction
Outliers Removal
testing data
knn
classification prediction

Figure 2: Modified Iterative Execution flow

features. For instance, one could see what genes are pairwise highly linearly correlated with the already selected genes and remove them from the candidate feature list. One could also use other relevant redundancy metric such as information gain. This procedure could be run as many times as needed to select a precise number of relevant features. It should improve the accuracy as we know that for instance the KNN classifier is sensitive to correlated features. Then, one could arguably remove a proportion of outliers form the training data depending on the sensitivity of the classifier to noise in its training dataset.

3.3. Implementation

I implemented my idea of improvement in Scala using the Spark distributed computing framework to be consistent with the parallel aspect of ReliefF. I could only test it locally however so the speed improvement behind the parallelization is not evaluated. The Scala source code is hosted at **github.com/Emilien-P/leuk-gene-analysis**.

3.4. Tests

3.4.1. Datasets

To perform my tests, I used the Mile Leukemia Dataset [12]. It consists in samples of gene expression levels classified by type and/or subtype of leukemia. For my experiments, I arbitrarily chose to focus my tests on the CML, CLL and MDS types. I kept the data size reasonable and it is more realistic for a small amount of selected features. The running times were also already fairly long as my implementation is not close to memory. The number of samples per classes were:

types	CML	CLL	MDS
number of samples	76	448	207

I preprocessed the gene expression levels by standardizing them.

3.4.2. Stability of ReliefF

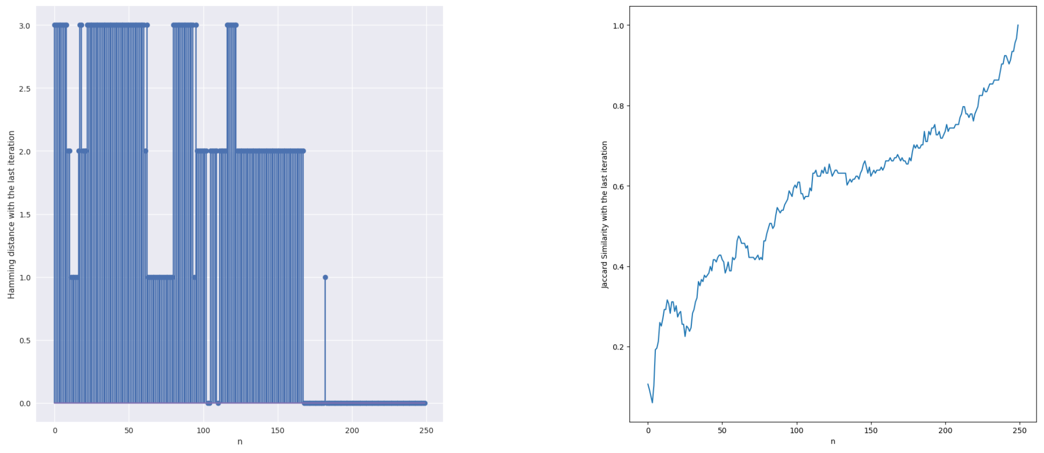
My idea of improvement relies on the fact that the ranking computed by ReliefF will converge after some number of iterations. If this is not the case, it wouldn't be grounded to remove features based on a gene that may not be selected by the final iteration. I hence computed the ranking of ReliefF for each iteration of ReliefF for a high number of sample to be sure of its stability. It is also important that the iterative extension provides reliable and stable rankings.

3.4.3. Accuracy of IterReliefF

I wanted to see weather the change in the execution flow improved the overall accuracy as I expected. I did all accuracy measure under 3-fold cross validation and selected the median score. I tested the feature selection of my algorithm against regular ReliefF and chi-square test on some selected subtypes of the data set. I used different classifiers (knn, random forest) to look for relevant differences in accuracy. I also did several different alternations of the types and number of classes tested. I set the k parameter of any ReliefF algorithm to 10 as it is a good empirical value [10].

3.5. Results & Analysis

3.5.1. Stability of ReliefF



(a) Hamming distance between the top three gene of the last iteration and the top three genes of a given iteration

(b) Jaccard similarity measure between the top 1% of genes and the top 1% of a given iteration

Figure 3: Stability measures for ReliefF on the whole Mile data set and n = 250

We can notice on figure 3a from the convergence of the top features after a certain amount of iterations of ReliefF. Hence, it seemed reasonable to me to consider an empirical value of convergence for the ReliefF algorithm and furthermore to do my high correlation filter based on the top features after a certain number of iterations. Figure 3b also hints that the ReliefF output is converging quite fast.

3.5.2. Accuracy of IterReliefF

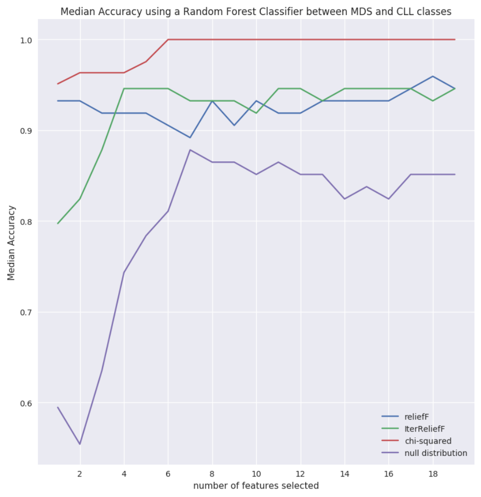


Figure 4: Accuracy under the Random Forest Classifier

Firstly, we see that the accuracy is neither significantly nor constantly improved. The iterative version of ReliefF seems to fail to rank the rights top features at fist as the slow start on all figures indicate. My hypothesis is that I took the empirical stability of ReliefF for granted and took a too small value for the expected convergence of ReliefF. It resulted in bad top selected features for IterReliefF. However, on a more sunny side, one other recurring pattern is that IterReliefF eventually does better than ReliefF after a certain number of features. I think that it happens when the output of IterReliefF eventually consists in the ReliefF good selected features without the highly correlated one.
We can see that on figure 4 between features 4 to 8, on figures 6 and 8 between 10 and 19 and on figure 7 between 11. It also seems that the IterReliefF rarely perform worst than ReliefF (figure 6 feature 19 and figure 7 feature 15) and they likely converge as the number of features increase.
It is also frustrating to see that the chi-square feature evaluation test performs better in any case. The chi-square test is univariate whereas the ReliefF family is multi-variate. It might be more suited to the gene expression features selection problem.

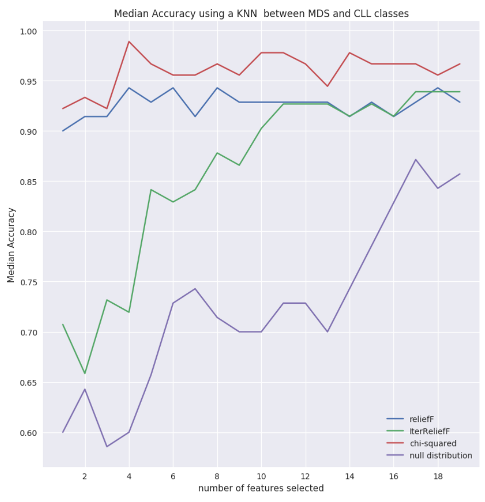


Figure 5: KNN with k = 3

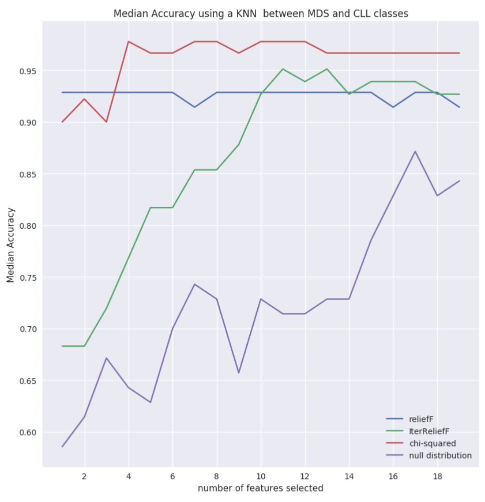


Figure 6: KNN with k = 7

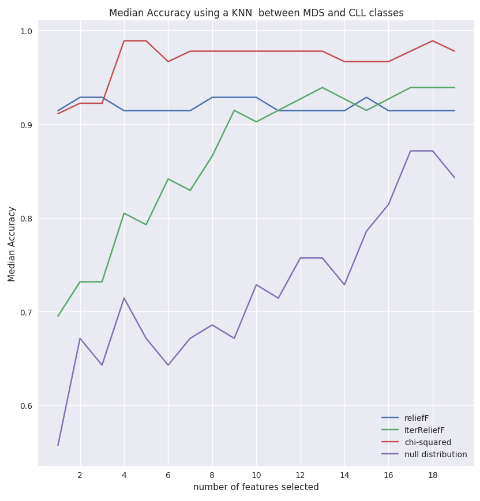


Figure 7: KNN with k = 13

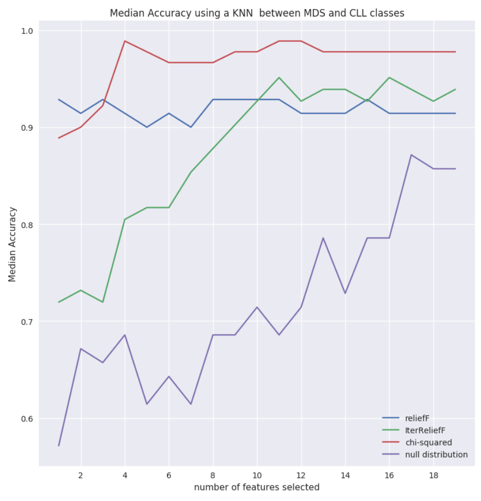


Figure 8: KNN with k = 19

Figure 9: Accuracy under KNN classifiers

4. Conclusion & Future Work

IterReliefF was not a complete success. Its accuracy was in all cases lower than the chi-square test. However, there were some intervals in the number of features where it distinctly performed better as ReliefF. It highlights that the high correlation correction can be of some use for an extended version of ReliefF. The implementation needs nonetheless some revision and correction. Maybe the iterative approach was not the most appropriate and a redundancy correction "on the fly" could yield better results.
A more in-depth analysis of the convergence of the features of ReliefF could be greatly beneficial to IterReliefF, as we've seen suffers from my stability assumption.
I also really wanted to evaluate the potential speed gain of IterReliefF in the classification process but could not because I lacked of time and had troubles with the metric test library. This would be an interesting project extension as future works. Also to using different redundancy metrics than the simple linear correlation baseline such as entropy based metrics could improve the effectiveness of IterReliefF.
In the gene expression classification problem, ReliefF might be more suited as a first feature pre-selection algorithm before forwarding the results into another features evaluation algorithm. This approach was used in Hijazi and Chan [5]. Another interesting modification could be to modify the multi-label or the hierarchical label ReliefF to implement redundancy correction [6][7].

5. Bibliography & References

[1] A. J. Aljaaf, D. Al-Jumeily, A. J. Hussain, P. Fergus, M. Al-Jumaily, N. Radi, Applied machine learning classifiers for medical applications: Clarifying the behavioural patterns using a variety of datasets, in: 2015 International Conference on Systems, Signals and Image Processing (IWSSIP), pp. 228–232.

[2] R. K. Singh, M. Sivabalakrishnan, Feature selection of gene expression data for cancer classification: A review, Procedia Computer Science 50 (2015) 52 – 57. Big Data, Cloud and Computing Challenges.

[3] L. Yu, H. Liu, Redundancy based feature selection for microarray data, in: Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '04, ACM, New York, NY, USA, 2004, pp. 737–742.

[4] H. Abusamra, A comparative study of feature selection and classification methods for gene expression data of glioma, Procedia Computer Science 23 (2013) 5 – 14. 4th International Conference on Computational Systems-Biology and Bioinformatics, CSBio2013.

[5] H. Hijazi, C. Chan, A classification framework applied to cancer gene expression profiles, J Healthc Eng 4 (2013) 10.1260/2040–2295.4.2.255. 23778014[pmid].

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

[7] I. Slavkov, J. Karcheska, D. Kocev, S. Kalajdziski, S. Džeroski, ReliefF for Hierarchical Multi-label Classification, Springer International Publishing, Cham, pp. 148–161.

[8] Z. Wang, Y. Zhang, Z. Chen, H. Yang, Y. Sun, J. Kang, Y. Yang, X. Liang, Application of relieff algorithm to selecting feature sets for classification of high resolution remote sensing image, in: 2016 IEEE International Geoscience and Remote Sensing Symposium (IGARSS), pp. 755–758.

[9] I. Kononenko, Estimating attributes: Analysis and extensions of relief 784 (1996).

[10] K. Kira, L. A. Rendell, A practical approach to feature selection, in: Proceedings of the Ninth International Workshop on Machine Learning, ML92, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1992, pp. 249–256.

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

[12] A. Kohlmann, T. J. Kipps, L. Z. Rassenti, J. R. Downing, S. A. Shurtleff, K. I. Mills, A. F. Gilkes, W.-K. Hofmann, G. Basso, M. C. Dellâ??Orto, R. Fo??, S. Chiaretti, J. De Vos, S. Rauhut, P. R. Papenhausen, J. M. Hern??ndez, E. Lumbreras, A. E. Yeoh, E. S. Koay, R. Li, W.-m. Liu, P. M. Williams, L. Wieczorek, T. Haferlach, An

international standardization programme towards the application of gene expression profiling in routine leukaemia diagnostics: the microarray innovations in leukemia study prephase, Br J Haematol 142 (2008) 802–807. 18573112[pmid].

Annexes

5.1. Number of Sample per Type

types	**number of samples**
CLL	448
AML with normal karyotype + other abnormalities	347
c-ALL/Pre-B-ALL without t(9;22)	237
MDS	207
T-ALL	174
c-ALL/Pre-B-ALL with t(9;22)	122
CML	76
Non-leukemia and healthy bone marrow	*73*
Pro-B-ALL with t(11q23)/MLL	70
ALL with t(12;21)	58
AML complex aberrant karyotype	52
AML with t(8;21)	40
ALL with hyperdiploid karyotype	40
AML with t(11q23)/MLL	38
AML with t(15;17)	37
ALL with t(1;19)	36
AML with inv(16)/t(16;16)	28
mature B-ALL with t(8;14)	13

$$\textbf{Top features correlation matrix} = \begin{bmatrix} 1 & -0.404 & 279 & 0.818 & 0.3 \\ -0.404 & 1 & -0.192 & -0.47 & 0.079 \\ 0.279 & -0.192 & 1 & 0.264 & 0.465 \\ 0.818 & -0.47 & 0.264 & 1 & 0.261 \\ 0.3 & 0.079 & 0.465 & 0.261 & 1 \end{bmatrix}$$

Figure 10: Correlation Matrix of ReliefF output with n=250 between CML and CLL classes

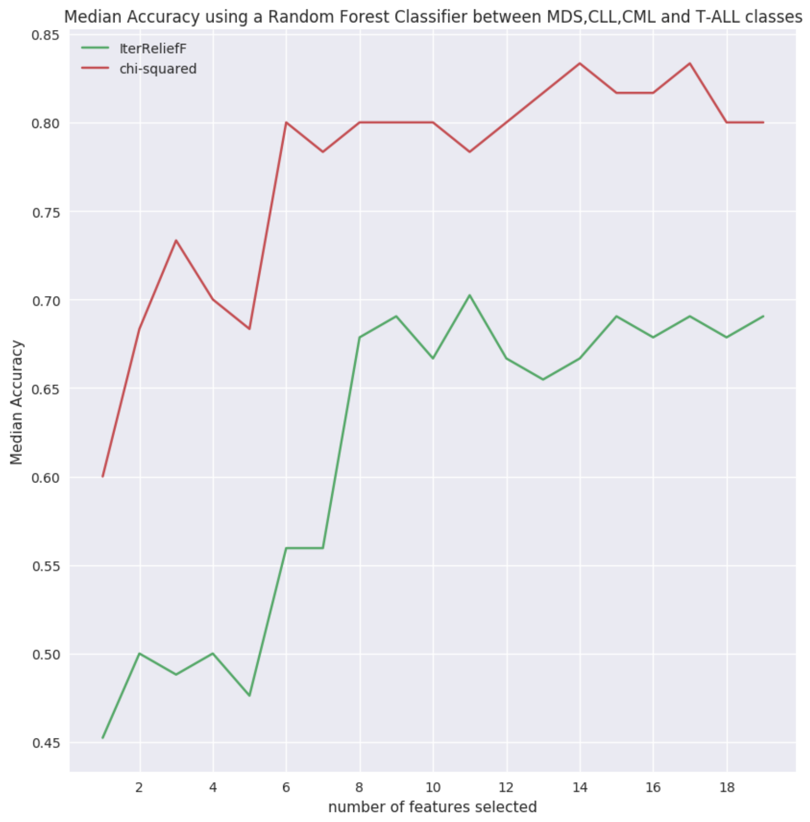


Figure 11: Accuracy between the three classes MDS, CLL, T-ALL and CML