Copyright © 2015 American Scientific Publishers
All rights reserved
Printed in the United States of America

Advanced Science Letters
Vol. XXXXXXXXX

MeLiF: Filter Ensemble Learning Algorithm for Gene Selection

Ivan Smetannikov, Andrey Filchenkov
Computer Technologies Lab, ITMO University, St. Petersburg 197101, Russia

Feature filtering algorithms are commonly used for prepossessing high-dimensional datasets such as DNA microarrays. Feature selection ensemble learning is a developing field, where new efficient and robust feature selection algorithms are produced by combining existing ones. We propose a new algorithm called MeLiF which is based on ranking filters ensemble via learning linear form. We compare this algorithm on nine DNA-microarray datasets with basic feature filtering algorithms and RelifF. MeLiF has shown the best AUC score and competitive stability in comparison with other methods.

Keywords: Ensemble Learning; Feature Selection; Ranking Filter; Stability; Univariate Filter; Big Data Preprocessing; DNA Microarray; Dimension Reduction; Coordinate Descent.

1. INTRODUCTION

Feature selection (FS) remains one of the important domains in machine learning. Nowadays, the application of FS algorithms is a necessary step for preprocessing high-dimensional data[1]. Extreme growth of data retrieved in many fields such as text mining, sensor data processing and bioinformatics has been taking place during the last decade, which shifted attention to simple but efficient algorithms that can be applied for datasets with thousands of features.

Gene analysis is one of the domains where a lot of new high-dimensional data is generated. An example of such data is gene expression profiles, usually referred to as DNA microarrays, which represent measurements of simultaneous gene expression[2]. The analysis of such datasets leads to understanding gene functionality and their reaction to certain conditions such as environment, drug, or disease influence. The main problem is that such datasets contain only a few objects and many features, which can easily lead to classifier overfitting. Therefore FS is de-facto standard for DNA microarrays preprocessing[3].

*Email Address: afilchenkov@niuitmo.ru

Three traditional approaches in FS are[2]: filters, wrappers and embedded methods. Despite wrappers and embedded methods being powerful tools that could show very high results, they consume relatively much time comparing to filters. That is why filtering methods are widely applicable for preprocessing DNA microarrays. Filters are also subdivided into univariative and multivariative algorithms. Algorithms in the first class ignore dependencies between features; nevertheless, they are still widespread in the domain because of their speed, which allows them to be integrated into multilevel optimization tasks.

Nowadays, novel methods in FS tend to be based on combinations or compositions of different algorithms and approaches. This ensemble learning paradigm has shown excellent results for classifiers such as AdaBoost and Random Forest[4]. This paradigm also seems to be promising for FS algorithms. One of the inspiring examples is the method proposed by Chuang et al[5]. A detailed review of novel ensemble methods in feature selection was made by Bolon-Canedo et al[6].

In this work we present a new approach, which is based on learning a linear combination of univariative filters.

2. BACKGROUND

FS can be represented as an optimization problem: given feature set $F = \{f_1, \dots, f_{|F|}\}$, target vector Y and quality measure Q defined on the set 2^F of all the subsets of F, find such $F^* \in 2^F$ that

$$F^* = \arg\max_{F' \subseteq F} Q(F').$$

The question of choosing a proper quality measure is still discussable, but usually $Q(F')$ is defined as $Q_c(C(F', Y))$, where C is a classifier and Q_c is one of classification quality measures. Also auxiliary quality measures are used: the number of selected features (the smaller the better) and runtime (the smaller the better). Another auxiliary quality measure, which is commonly used in the gene selection problem, is stability[7]. The stability measure is high for methods which are insensitive to the size of sample. This property is essential for gene selection because the number of objects is usually extremely small.

The strategy which is used in filters exploits some inner qualities of features that determine their applicability for solving the given problem. This strategy does not involve any classifier, which makes these algorithms relatively universally applicable. An inner quality measure I_o usually describes how strongly a subset F' is connected with the target vector Y and how similar the features in F' are. The choice of inner quality is another open question, and it seems that no universal measure exists which suits for all the cases. Nevertheless, any inner quality measure estimation for a feature subset consumes much less time than a single execution of a classifier on this subset, therefore the most time consuming part of these algorithms is search space.

Univariate, or ranking, filters are a subclass of filters which are known to be the fastest algorithms for feature selection. They do not estimate each subset, but use single feature measure $m_o : F \to \mathbb{R}$. A subset measure is estimated as the sum of measures of its elements:

$$m_o(F') = \sum_{f \in F'} m_o(f).$$

With a fixed number of features to be chosen, the optimization problem is solved greedily: top most features are chosen. The general scheme of a ranking filter is to estimate $m_o(f)$ for each feature f, sort the features with respect to m_o and select top-ranked features.

We use the following notation for ranking filters: it applies a certain selection rule κ to feature set F, ordered with respect to feature measure m, and its output is feature subset $F' = \kappa(m, F)$. In these terms the optimization problem

$$F^* = \arg\max_{F' \subseteq F} Q(F').$$

is reduced to the optimization problem

$$m^* = \arg\max_{m' : F \to \mathbb{R}} Q\big(\kappa(m', F)\big).$$

Nowadays, three different approaches to learn feature ensembles exist. The first ensemble learning approach combines output subsets of FS algorithms, or, in case of ranking filters, aggregates ranked feature lists[8,9]. The second approach combines classifiers which are learnt on different FS algorithm outputs[10]. The last ensemble learning approach combines feature measures to obtain a new, better feature measure[11].

3. LEARNING LINEAR COMBINATION OF FEATURE MEASURES

Assume we have a fixed set of different feature measures $M = \{m_1, \dots, m_{|M|}\}$, where $m_i : F \to \mathbb{R}$, which are considered to be good approximations for the optimal measure m^*. Also assume we have a fixed FS rule κ.

The approach we propose is to find the solution of the optimization problem within the linear space of the features measures:

$$m^* = \arg\max_{m' \in \mathbf{V}(M)} Q\big(\kappa(m', F)\big),$$

where $\mathbf{V}(M)$ is a vector space built on the basis M, i.e. $\forall m' \in \mathbf{V}(M) \quad \exists A = (\alpha_1, \dots, \alpha_{|M|}) \in \mathbb{R}^{|M|}$ such that $m' = \sum_{i=1}^{|M|} \alpha_i m_i$.

The optimization problem is reduced to the following one: find a vector $A^* = (\alpha_1, \dots, \alpha_{|M|})$, such that

$$A^* = \arg\max_{A' \in \mathbb{R}^{|M|}} Q\left(\kappa\left(\sum_{i=1}^{|M|} \alpha_i m_i, F\right)\right) = \arg\max_{A' \in \mathbb{R}^{|M|}} Q_A(A').$$

In general, it is impossible to retrieve the derivative of Q_A, therefore analytical methods are not applicable.

In our approach we use a variation of the coordinate descent method. When the method is in the point $A = (a_1, a_2, \dots, a_M)$ it estimates the value of Q_{A^+} for the point $A^+ = (a_1, \dots, a_{i-1}, a_i + \delta, a_{i+1}, \dots, a_{|M|})$. If the value of Q_{A^+} is lower than Q_A, then the method tries the point $A^- = (a_1, \dots, a_{i-1}, a_i - \delta, a_{i+1}, \dots, a_{|M|})$ in the same way. If the value of Q_{A^-} is also lower than Q_A then the method tries to improve result by varying the $(i+1)$-th coordinate. But if at least one of them have shown higher result, the method moves to the corresponding point and assigns $i = 1$. Once all the possible coordinates are tried and no better result is found, the method stops.

Points like (1,2,3,4) and (2,4,6,8) will show identical results, as the proportion between aggregated metrics will remain the same. Therefore there is no need in changing the descent step, because it would be equivalent to vector rescaling. In the case of a fixed descent step, the search space can be described as a search grid, with grid size being equal to the optimization step size. Keeping this in mind we cached all the points which have already been visited, and could efficiently reuse this information so we never considered the same point more than once for the same data.

The initial points of search are the following: $(1, 0, \dots, 0)$, $(0, 1, \dots, 0)$, $\dots$, $(0, 0, \dots, 1)$, and $(1, 1, \dots, 1)$.

4. MELIF ALGORITHM

The algorithm we propose is called MeLiF (**me**asure **li**near **f**orm). The algorithm uses function

EVALOFQCACH(A) returning the value of $Q_A(A)$. This function evaluates this value for each A only once and caches it in order to return the saved value with each subsequent function call, The algorithm has the following input: F is feature set, Y is target vector, $M = \{m_1, \dots, m_{|M|}\}$ is set of feature measures, κ is feature selection rule, Q is solution quality measure. The only output of the algorithm is feature subset F'. The algorithm is parametrized with δ, the length of the descent step.

```
MeLiF(F, Y, N, κ, Q;δ)
BEGIN
  FOR (i = 1 TO |M|)
  BEGIN
    Initial[i] ← (0_(1), ..., 1_(i), ..., 0_(|M|))
  END
  Initial[|M + 1|] ← (1_(1), ..., 1_(|M|))
  q* ← 0
  FOR (p = 1 TO |M| + 1)
  BEGIN
    A ← Initial[p]
    q ← EVALOFQCACH (A)
    isImprovable ← TRUE
    c ← 1
    noImprove ← 0
    WHILE (isImprovable)
    BEGIN
      noImprove ← noImprove + 1
      A+ ← A + (0_(1), ..., δ_(c), ..., 0_(|M|))
      A− ← A + (0_(1), ..., −δ_(c), ..., 0_(|M|))
      q+ ← EVALOFQCACH(A+)
      IF (q+ > q)
      BEGIN
        q ← q+
        A ← A+
        noImprove ← 0
      ELSE
        q− ← EVALOFQCACH(A−)
        IF (q− > q)
        BEGIN
          q ← q−
          A ← A−
          noImprove ← 0
        END
      END
      IF (noImprove = |M|)
      BEGIN
        isImprovable ← FALSE
      END
    END
    IF (q > q*)
    BEGIN
      q* ← q
      A* ← A
    END
  END
  M* ← M · A
  F′ ← M · A
  F′ ← κ(M*, F)
  RETURN F′
END
```

5. EXPERIMENT SETUP

We ran two series of computational experiments. The first series was used to investigate the performance of MeLiF and tune its parameters. We chose a strategy of space search. For this purpose we used 132 different datasets from different fields, each one with less than 1000 features. Some results retrieved from these experiments are presented in the Discussion.

The second series of experiments was performed to compare our algorithm with other ones. We organized the experiments in the following way. As the basic feature measures we used Spearman correlation, Symmetrical uncertainty, Value difference metric and Fit criterion.

We compare MeLiF with five algorithms. Four of them are filters with the described feature measures. We will refer to them as "basic filters". The fifth algorithm is well-known ReliefF algorithm which is one of the most effective algorithm for feature selection[12,13]. We use ReliefF with 1, 3 and 5 neighbors.

We ran the algorithm on six DNA microarrays, which were not used during the first series of experiments. Two of them were taken from Cancer Program Data Sets, from the Broad Institute[14]: Breast cancer A dataset (98 instances × 1213 features,) and the Leukemia dataset (30×15061). The other four datasets were taken from the Bioinformatics Research Group of Universidad Pablo de Olavide Dataset Repository[15]: Leukemia dataset (38 × 7129), the Lymphoma dataset (96 × 4026), the Central neural system tumors dataset (60×7129) and the Global cancer dataset (144×16063). Five of the chosen datasets are multi-labeled. We splited them into several derivative binary datasets, the number of which equaled the number of labels the original dataset contains. Then we dropped some of these datasets, as some of them contained too few instances of one of the classes.

We used three algorithm performance measures: AUC, stability, and runtime. AUC is the well-known Area Under the Curve measure estimating classifier performance. As the classifier on the resulting feature subset we use SVM[16].

We measure the stability of a feature selection algorithm by taking three random subsamples X_1, X_2, X_3, each containing 60% of objects in the initial dataset, and then applying this feature selection algorithm, thus receiving three feature subsets F_1, F_2, and F_3. The stability is defined as $\big(T_d(F_1, F_2) + T_d(F_1, F_3) + T_d(F_2, F_3)\big)/3$, where T_d is the Tanimoto distance[17]:

6. EXPERIMENT RESULTS

To compress the experiment result representation, we aggregated the results of all the algorithms on each dataset for the derivative binary one-versus-rest dataset. We provide links to the source where complete tables with result can be found.

The comparison of the AUC measure is presented in table 1. The complete table can be found online[18].

Table.1. Average AUC scores for each dataset trough all derivative binary datasets.

METHO	B_CAN	G_CAN	LEU_UP	LEU_BR	LYMPH	TUMOR
FC	0.965	0.858	0.967	1	0.947	0.677
Sp	0.989	0.83	0.935	1	0.943	0.679
SU	0.993	0.849	0.968	1	0.945	0.647
VDM	0.974	0.838	0.951	1	0.954	0.689
Equal	0.992	0.858	0.932	1	0.954	0.69
Rel-1	0.968	0.839	0.941	1	0.964	0.639
Rel-3	0.966	0.836	0.952	1	0.96	0.632
Rel-5	0.98	0.844	0.949	1	0.957	0.624
MeLiF	**0.996**	**0.923**	**0.984**	1	**0.996**	**0.723**

We can see that the MeLiF algorithm outperforms all other algorithms on all the datasets. This statement is almost true for all derivative binary datasets: ReliefF slightly outperforms MeLiF only in a single case.

The stability measure is presented in Table 2. The complete table can be found online[19].

Table.2. Average stability scores for each dataset trough all derivative binary datasets.

METHO	B_CAN	G_CAN	LEU_UP	LEU_BR	LYMPH	TUMOR
FC	0.417	**0.46**	0.225	0.315	0.392	0.157
Sp	**0.533**	0.414	**0.336**	0.448	**0.497**	**0.271**
SU	0.446	0.353	0.266	0.349	0.402	0.21
VDM	0.149	0.175	0.193	0.241	0.325	0.114
Equal	0.372	0.292	0.266	0.367	0.377	0.16
Rel-1	0.137	0.172	0.093	**0.945**	0.237	0.117
Rel-3	0.197	0.217	0.138	**0.945**	0.258	0.157
Rel-5	0.272	0.249	0.168	**0.945**	0.281	0.158
MeLiF	0.192	0.339	0.174	0.945	0.303	0.13

According to the table, MeLiF shows competitive stability on all the datasets except the Broad Institute leukemia dataset. As it can be seen, all the methods achieved the highest AUC scores on this dataset. It is true for all the derivative binary datasets as well. Therefore we assume that the function we optimize has many local maxima, and the same points are found by ReliefF, while other algorithms find different points on each run .

Basic filters runtime is very low, therefore in Table 3 only runtime for ReliefF and MeLiF is presented. The complete table can be found online[20].

Table.3. Average time consumed in minutes and starting amount of features for each dataset for all derivative binary datasets.

METHO	B_CAN	G_CAN	LEU_UP	LEU_BR	LYMPH	TUMOR
ReliefF	1	2	1	1	1	1
MeLiF	1	9	5	2	3	1
Features	1213	16063	7129	15061	4026	7129

We can see that the MeLiF algorithm is slower than RelieF. However, the runtime differs only is 5 times as maximum.

7. DISCUSSION

The MeLiF algorithm which we proposed in this paper has shown excellent efficacy, competitive stability and time cost comparing to filters with the basic feature measure and ReliefF algorithms.

Such high results achieved by MeLiF can be partly explained by the fact that it was adjusting the quality measure performing as a wrapper. MeLiF showed limited success in producing stable results, but there is no algorithm which has shown better results.

Time consumption is of most interest in this case. MeLiF was the slowest algorithm for all the datasets comparing with ReliefF and the basic filters. We see no clear dependency on the number of features according to Table 3. In particular, on G_CAN dataset with the maximum number of features, ReliefF was almost 5 times faster. But on the second biggest dataset, LEU_BR, this proportion is only 2. It is worth to note that the difference in resulting AUCs on the first dataset is very high.

Feature selection algorithm time complexity depends on the time complexity of space search and the time complexity of measure evaluation. Let T_{MeLiF} denote MeLiF time complexity.

PROPOSITION 1. $T_{MeLiF} = O(|F|)$..

The correctness of the first statement is explained by the fact that each MeLiF step has complexity $O(T_{rf}(|F|)$, which is $O(|F|)$, and the number of steps does not depend on $|F|$, since the search is performed in a space whose dimensionality is independent of $|F|$.

COROLLARY 1**.** The following statements are true:

1) $T_{MeLiF}(|F|) = O(T_{rf}|F|)$,
2) $T_{MeLiF}(|F|) = o(T_{ssf}|F|)$,
3) $T_{MeLiF}(|F|) = o(T_w|F|)$,

where T_{rf} denotes a ranking filter complexity, T_{ssf} denotes a space search filter complexity, and T_w denotes a wrapper complexity.

During our research we have tried different maximization methods on 132 different datasets from various areas. Because its single goal was exploration, we had no certain pipeline or framework for this research, therefore we do not include the description of this

experiment series in the paper. Nevertheless, we found two notable observations

1. On some of the datasets all the initial points like $(0,\dots,0,1,0,\dots,0)$ were local optima, and the optimization method was stuck in these points for many steps unable to improve result. We have never registered this algorithm behavior for point $(1,1,1,1)$.

2. All the initial points were fruitful to produce the best results of the algorithm, and there was no initial point to end in best result for most of the runs.

8. FUTURE WORK

We tested the approach only on the four feature measures. Despite the result being high, we expect to improve it by considering other feature measures like Hilbert-Smidt Independence Criterion[21].

Another direction of the future work is improving the algorithm stability. This can be achieved by solving the optimization problem of maximizing quality of the auxiliary criterion to maximize stability. This problem is related to the bias-variance problem, therefore the solution is a tradeoff between stability and efficacy.

It also may be fruitful to use swarm optimization methods, such as artificial bee colony optimization[22], because these methods exploit the interaction of different solutions found by different species to find the global optimum. This suits to the problem of finding optimal linear combination, because we use several different initial points to start, and points in the search space can be easily combined with each other.

ACKNOWLEDGMENTS

Authors would like to thank Vladislav Dolganov, Igor Buzhinsky, and Veronika Minina for useful comments. This work was financially supported by the Government of Russian Federation, Grant 074-U01.

REFERENCES

[1] J. Fan, R. Samworth, and Y. Wu, "Ultrahigh dimensional feature selection: beyond the linear model," The Journal of Machine Learning Research, vol. 10, pp. 2013–2038, 2009.

[2] Y. Saeys, I. Inza, and P. Larranaga, "A review of feature selection techniques in bioinformatics," bioinformatics, vol. 23, no. 19, pp. 2507–2517, 2007.

[3] L. Yu, "Feature selection for genomic data analysis," Computational methods of feature selection, pp. 337–353, 2008.

[4] T. G. Dietterich, "Ensemble methods in machine learning," in Multiple classifier systems. Springer, 2000, pp. 1–15.

[5] L.-Y. Chuang, C.-H. Yang, K.-C. Wu, and C.-H. Yang, "A hybrid feature selection method for dna microarray data," Computers in biology and medicine, vol. 41, no. 4, pp. 228–237, 2011.

[6] V. Bolon-Canedo, N. S ´ anchez-Maro ´ no, A. Alonso-Betanzos, J. Benıtez, and F. Herrera, "A review of microarray datasets and applied feature selection methods," Information Sciences, vol. 282, pp. 111–135, 2014.

[7] H.-M. Lai, A. Albrecht, and K. Steinhofel, "Gene selection guided by feature interdependence," World Academy of Science, Engineering and Technology (WASET), no. 79, pp. 1432–1438, 2013.

[8] A.-C. Haury, P. Gestraud, and J.-P. Vert, "The influence of feature selection methods on accuracy, stability and interpretability of molecular signatures," PloS one, vol. 6, no. 12, p. e28210, 2011.

[9] T. Abeel, T. Helleputte, Y. Van de Peer, P. Dupont, and Y. Saeys, "Robust biomarker identification for cancer diagnosis with ensemble feature selection methods," Bioinformatics, vol. 26, no. 3, pp. 392–398, 2010.

[10] V. Bolon-Canedo, N. Sanchez-Marono, and A. Alonso-Betanzos, "An ensemble of filters and classifiers for microarray data classification," Pattern Recognition, vol. 45, no. 1, pp. 531–539, 2012.

[11] A. Filchenkov, V. Dolganov, and I. Smetannikov, "PCA-based algorithm for constructing ensembles of feature ranking filters," in ESANN. ESANN, 2015, pp. 201– 206.

[12] I. Kononenko, "Estimating attributes: analysis and extensions of relief," in Machine Learning: ECML-94. Springer, 1994, pp. 171–182.

[13] Y. Sun, "Feature weighting through local learning," Computational Methods of Feature Selection, p. 233, 2008.

[14] http://www.broadinstitute.org/cgi-bin/cancer/datasets.cgi

[15] http://eps.upo.es/bigs/datasets.html

[16] C. J. Burges, "A tutorial on support vector machines for pattern recognition," Data mining and knowledge discovery, vol. 2, no. 2, pp. 121–167, 1998.

[17] A. H. Lipkus, "A proof of the triangle inequality for the Tanimoto distance," Journal of Mathematical Chemistry, vol. 26, no. 1-3, pp. 263–265, 1999.

[18] http://genome.ifmo.ru/files/papers_files/ICMLA2015/MeLiF-AUC.pdf

[19] http://genome.ifmo.ru/files/papers_files/ICMLA2015/MeLiF-Steps.pdf

[20] http://genome.ifmo.ru/files/papers_files/ICMLA2015/MeLiF-Time.pdf

[21] L. Song, A. Smola, A. Gretton, J. Bedo, and K. Borgwardt, "Feature selection via dependence maximization," The Journal of Machine Learning Research, vol. 13, no. 1, pp. 1393–1434, 2012.

[22] D. T. Pham and M. Castellani, "The bees algorithm: modelling foraging behaviour to solve continuous optimization problems," Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science, vol. 223, no. 12, pp. 2919–2938, 2009