Fast-Ensembles of Minimum Redundancy Feature Selection

Benjamin Schowe and **Katharina Morik**
Technische Universität Dortmund
{schowe,morik}@ls8.cs.tu-dortmund.de

Abstract

Finding relevant subspaces in very high-dimensional data is a challenging task not only for microarray data. The selection of features must be stable, but on the other hand learning performance is to be increased. Ensemble methods have succeeded in the increase of stability and classification accuracy, but their runtime prevents them from scaling up to real-world applications. We propose two methods which enhance correlation-based feature selection such that the stability of feature selection comes with little or even no extra runtime. We show the efficiency of the algorithms analytically and empirically on a wide range of datasets.

1 Introduction

The growing dimensionality of recorded data, especially in bioinformatics, demands dimension reduction methods that identify small sets of features leading to a better learning performance. Along with the high dimensionality comes a high variance, which makes it hard to find adequate feature subsets without being kept in local optima. The large number of features challenges the runtime of the selection algorithms. Hence, the main criteria for its quality are that the algorithm is *a*) **multivariate**, takes into account feature correlations, *b*) **stable**, does not vary much for unseen data of the population; *c*) **amending learning**, the learning performance is enhanced; *d*) **fast**: it scales well for very large numbers of features. Ensemble methods decrease variance and, hence, are frequently used in feature selection. However, they usually slow down the procedure. This paper presents a method that speeds up ensembles in a simple and effective way. A careful evaluation on 9 data sets investigates the quality of our new methods.

2 Related Work

Fast univariate **filter** approaches like the t-test [Fox and Dimmic, 2006] or SAM-statistics [Tusher *et al.*, 2001] compute a scoring function on the features, disregarding feature interplay. **Wrapper** approaches [Kohavi and John, 1997] better solve this problem, at the cost of much longer runtime. Each feature set evaluation demands a cross-validated training of the used learning algorithm. Some learning algorithms provide the user with an implicit feature ranking which can easily be exploited for feature selection. Such **embedded** approaches are using the weight vector of a linear SVM [Vapnik, 1998] or the frequency of feature use of a *Random Forest* (RF) [Breiman, 2001]. They are aware of feature interplay and faster than wrappers but biased towards the learning algorithm used.

A group of new algorithms has come up to bridge the gap between fast but univariate filters on the one hand, and slow but multivariate wrappers on the other hand. Their goal is to find a subset of features which is highly predictive with no or a minimum of redundant information. The *correlation based feature selection* (CFS) [Hall, 2000] performs a *sequential forward search* with a correlation measure in the evaluation step. CFS iteratively adds the feature which has the best ratio between predictive relevance of the feature and its correlation with the already selected features. Both, predictiveness and correlation, are measured by the entropy-based *symmetrical uncertainty* where the information gain IG of feature f_i w.r.t. feature f_j is divided by the sum of the entropies of f_i, f_j. Since CFS uses symmetrical uncertainty, it is only suitable for discrete values.

Ding and Peng [Ding and Peng, 2005] reinvented CFS with the capability for handling numerical variables calling it *minimum redundancy maximum relevance FS* (MRMR). For numerical features the *F-test* is used. For a continuous feature x and a nominal class variable y, both from a data set with n examples and C classes, defined as

$$F(X,Y) = \frac{(n-C)\sum_c n_c(\bar{X}_c - \bar{X})}{(C-1)\sum_c (n_c-1)\sigma_c^2} \quad (1)$$

with class-pooled variance σ_c and $i \in \{1,..,n\}$, n_c the number of examples in class c, $c \in \{1,..,C\}$. The redundancy of a numerical feature set is measured by the absolute value of *Pearson's correlation coefficient*

$$R(X,Y) = \frac{Cov(X,Y)}{\sqrt{Var(X)\cdot Var(Y)}} \quad (2)$$

with $i \in \{1,..,n\}$. It detects a linear dependency between x and y. Another possible measure for the dependency between two nominal variables used by MRMR [Ding and Peng, 2005] is the *mutual information*

$$MI(X,Y) = \sum_{i,j} P(x_i,y_j)\log_2 \frac{P(x_i,y_j)}{P(x_i)P(y_j)} \quad (3)$$

x_i and y_j are the possible values (alphabet) of X and Y.

From now on, we will use the term *correlation* and the symbol $Cor(x,y)$ as a synonym for either Pearsons's linear correlation eq. (2), F-test eq. (1) or mutual information eq. (3). The correlation measures are averaged over all combinations of the feature set. Instead of the ratio, one can also use the difference between relevance and redundancy [Ding and Peng, 2005]. In any case, the measures

are based on variance and, hence, are sensitive to outliers and a high variance of the input data, alike. This instability is not suitable, e.g., for biomedical research, where the relevance of features is in the research focus. The main algorithmic issue is the computation of the correlations. In the first step, selecting the most relevant feature takes p calculations of $Cor(x_i, y)$. The next steps in MRMR/CFS are to repeatedly add the feature which has the best ratio between relevance and redundancy to the already selected features.

$$F_{j+1} = F_j \cup \{\arg\max_{f \in \mathbb{F} \setminus F_j} \frac{Cor(f,y)}{\frac{1}{j}\sum_{g \in F_j} Cor(f,g)}\} \quad (4)$$

This takes $p - (j-1)$ correlations in each step. For the whole MRMR/CFS process of selecting k from p features

$$p + \sum_{i=1}^{k-1}(p-i) = p \cdot k - \frac{k^2 - k}{2} \quad (5)$$

correlations must be computed. Variants of the method like, e.g., [Gulgezen *et al.*, 2009] tried to improve the stability of MRMR by introducing a weighting parameter α for the ratio of relevance and redundancy. Tuning this parameter even increases the overall runtime. The same holds for the approach [Michalak and Kwaśnicka, 2006], which evaluates together those pairs of features which are higher correlated than some δ, or for *Fast Correlation-based Filter* (FCBF) [Yu and Liu, 2004], which discards all features with relevance $< \delta$. Hence, MRMR/CFS is promising but suffering from a lack of stability.

Ensemble methods

A high variance negatively effects prediction algorithms (classification & regression) as well as feature selection schemes. Ensemble methods like *Bagging* [Breiman, 1996] or *Boosting* [Freund and Schapire, 1997] reduce variance. Parallel ensembles, e.g. *Bagging*, do so by repeating the algorithm on different subsamples or bootstrapped samples of the input data. This increases the stability of the set of selected features [Saeys *et al.*, 2008; Jurman *et al.*, 2008] and - in some cases - even reduces the prediction error [Xu and Zhang, 2006]. Saeys et al. [Saeys *et al.*, 2008] showed that (bagged) ensembles of *symmetrical uncertainty weighting*, *Relief*, SVM-RFE or RF delivered more stable feature selections than the non-ensembled counterparts, but did not increase classification performance. (For RF accuracy even decreased.) The major problem with *Bagging* are the e-times repetition for ensembles of cardinality e. This increases runtime considerably.

3 Speeding up Ensembles

We now have advantages and shortcomings of the methods which we want to enhance, namely MRMR/CFS (being unstable) and ensemble methods (being too slow). The basis for our algorithm is the "split-sum trick" going back to the displacement law of statistics, the latter of which we apply first. It helps to compute the $Cor(x, y)$ in one pass for any two features. Thanks to the displacement law, cf. (6), the measures (1) to (3) can be rewritten as sums of independent terms. We use Pearson's linear correlation as an example, but the other measures can be split analogously: Since the variance of a variable X can be written as

$$Var(X) = E((X - E(X))^2) = E(X^2) - (E(X))^2 \quad (6)$$

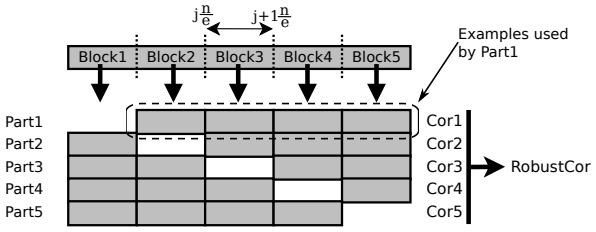


Figure 1: Inner Ensemble: Splitting correlation computation into parts: the $e = 5$ parts calculate the correlations on the data subsets marked gray. The average of Cor_1 to Cor_5 becomes $RobustCor$.

the variance- and covariance-terms now only contain sums of computationally independent terms. Pearson's linear correlation can be computed in only one pass over the data for each feature without prior computation of the mean values of each feature. Eq. (1) and (3) can similarly be split into sums of independent terms. This fact is used by our *Inner Ensemble* method and its further enhancement, the *Fast Ensemble*.

Inner Ensembles

Correlation measures like Pearson's linear correlation are very sensitive to outliers [Koh *et al.*, 2007]. This has a negative effect on the stability of the selected feature set. Our first method, the *Inner Ensembles*, increases the robustness of the correlation measure by performing a parallel *ensemble* of e correlations, instead. The correlation is calculated for e parts on subsets of the examples, each part leaving out $\frac{1}{e}$ of the data - similar to e-fold cross-validation. In contrast to cross-validation, the left-out fraction of the data is not used, at all. The average of the e correlations gives us a more robust correlation estimate. (cf. Fig. 1).

Let F_j denote the set of selected features in step $j \in \{1..k\}$ of MRMR/CFS. The MRMR/CFS algorithm first picks the feature which has the highest correlation with the label and so is most relevant. This takes p calculations of feature-label-correlation $F_1 = \{\arg\max_{f_i}(Cor(f_i, y))\}$ with $i \in [1, p]$. Usually, for ensembles of size e, this would increase runtime of MRMR/CFS by the factor e to $e \cdot p \cdot k - (k^2 - k)/2)$ correlations to compute. Now, we use that the eqs. (1), (2) and (3) can all be split into sums of sums like

$$\sum_{i=1}^{n} h_i = \sum_{i=1}^{m_1} h_i + \sum_{i=m_1+1}^{m_2} h_i + \cdots + \sum_{i=m_{e+1}}^{n} h_i \quad (7)$$

$$= \sum_{j=1}^{e} \left[\sum_{i=1+(j-1)\frac{n}{e}}^{j\frac{n}{e}} h_i \right] \quad (8)$$

for equally sized parts and arbitrary terms h_i. This allows us to compute these sums for all e intervalls $\left[(j-1)\frac{n}{e} + 1, j\frac{n}{e}\right]$ separately. The final correlation of each part $i \in [1, e]$ of the ensemble is then calculated by using all but the ith partial sums. There is virtually no extra runtime apart from e times adding up the partial sums within Algorithm 1. The memory need increases by factor e which is very small, because Pearson's linear correlation only needs five accumulator variables for the sums; F-test and mutual information only need $1+3\cdot|f_1|$ and $3\cdot|f_1|\cdot|f_2|$ variables, respectively, where $|f_1|$ and $|f_2|$ are the number of distinct values of the features f_1 and f_2. Algorithm 1 examplarily shows pseudo-code for Pearsons's linear correlation returning the correlation values for all parts in one

pass over the data. For *F-Test* (1) and *MI* (3) the procedure is similar.

Algorithm 1 Pearson Correlation Ensemble

Input: Numerical variables X, Y, ensemble size e, number of examples n
Init arrays of size e for split-sums: Cor[],e_n[],e_x[],e_xx[],e_y[],e_yy[],e_xy[]
Init overall sums: E_X=0; E_Y=0; E_XY=0; E_XX=0; E_YY=0
j=0; {index for the ensemble parts $1..e$}
for $i = 1$ to N **do** {Iterate over all examples, build overall & split-sums}
  e_x[j]+= X[i]; e_y[j]+= Y[i]; e_xy[j]+= X[i]*Y[i]; e_xx[j]+= X[i]^2; e_yy[j]+= Y[i]^2
  E_X+=X[i]; E_Y+=Y[i]; E_XY+=X[i]*Y[i]; E_XX+=X[i]^2; E_YY+=Y[i]^2
  e_n[j]++; j=(j+1) mod e;
end for
for j=1 to e **do** {Iterate over all ensemble parts: calculate correlation}
  denom=sqrt(abs((E_XX-e_xx[j]-$\frac{(E\_X-e\_x[j])^2}{n-e\_n[j]}$) · (E_YY-e_yy[j]-$\frac{(E\_Y-e\_y[j])^2}{n-e\_n[j]}$)))
  Cor[j]= (E_XY-e_xy[j]-$\frac{E\_X \cdot E\_Y}{n-e\_n[j]}$) / denom
end for
return "average" over Cor[]

Algorithm 2 Fast Ensemble of MRMR/CFS

Input: Set of all features $\mathbb{F}$, desired dimension k, size of the ensemble e, label y
for $i = 1$ to e **do** {For all parts of the ensemble}
  $F_i = \{f_{i,1} \in \mathbb{F} \mid max\ Cor(f_{i,1}, y)[i]\}$
  for j=2 to k **do** {Iteratively add best feature}
    $F_i = F_i \cup \{f_{i,j} \in \mathbb{F} \backslash F_i \mid max$ (Cor($f_{i,j}$,y)[i] / $\sum_{g \in F_i} Cor(f_{i,j}, g)[i]$)}
  end for
end for
return "average" over all F_i
Cor(a,b) first looks in cache. If measure for (a,b) or (b,a) was not yet cached, chooses appropriate measure, e.g. Alg. 1, and puts the resulting array into cache.

Fast Ensembles

Doing the calculations on diverse subsets of the data and using the split-sum trick in the *Inner Ensembles* way is extremely fast, but it increases the stability of selected feature sets only marginally, when used on each single correlation step, cf. Section 4. Our 2nd method, *Fast Ensemble*, builds an ensemble of the whole selection process, instead of stabilizing each single correlation. We dramatically reduce runtime of the full ensemble by applying the same split-sum trick.

If a correlation between two features is computed for one part of the ensemble, their correlations can be computed with practically no overhead for all other parts of the ensemble (cf. Algorithm 1). As opposed to the *Inner Ensemble*, here, the partial correlation results are not combined, but cached for later use. Just one pass over the examples is needed: For every ith part of the ensemble, the ith partial sums are left out when aggregating the sums to a correlation measure. Hence, for every two features or a feature-class combination, only one pass over the examples is needed, no matter in how many parts of the ensemble they appear. Where a full ensemble of MRMR/CFS needs $e \cdot p$ passes over the examples in order to select a first feature $f_{i,1}$ for e parts of the ensemble, our method does this just once. For every further feature $f_{i,j}$ in all parts of the ensemble, only those feature correlations need a pass over the examples which were not already considered in any other part of the ensemble.

Time-complexity directly depends on the diversity of the ensemble results. If all parts of the ensemble return the same feature set, the needed correlations have all been computed in the first part and the runtime is the same as for a single feature selection. If, in contrast, the resulting feature sets of the feature selections are disjoint, none of the feature pairs has been considered in other parts of the ensemble and thus has not been cached. These extremes are rare.

A simple example of selecting 3 out of 5 features (X_1 to X_5), label Y, illustrates *Fast Ensembles*, cf. Fig. 2. The ensemble consists of three parts. After a first calculation of the relevance of each feature, i.e. the correlation of X_i and Y, the relevance values are cached for each part. To estimate the feature with the best relevance/redundancy ratio the correlations of X_2 and the remaining four features must be computed. X_4 is chosen. In the last step the remaining features must be compared to the newly added X_4. This is repeated for all other parts. Now relevance can be computed from the split sums already computed in the first part. If X_i, X_j have been compared in an earlier part, their split-sums can be reused. The resulting sets are then combined, e.g. via majority vote. Only 15 passes are needed instead of 36 without the split-sum trick.

Our algorithms conduct search in feature subset space like wrappers do. Yet, unlike wrappers, the feature sets are not evaluated in long cross-validation runs but with a fast filter approach. Unlike filters, feature interdependencies are still considered. Hence, *Inner* and *Fast Ensembles* combine the advantages of wrapper and filter approaches. Note, that the inherent parallelism of the algorithms speeds up computation additionally. Every correlation computation between two features is independent of the other features. Thus, not only the parts of the ensemble can be computed in parallel, but also all correlation computations. Only completed calculations must be reported to some central instance which conducts the search in feature subset space.

4 Evaluation

We evaluated the performance of our algorithm with respect to **stability**, **accuracy** and **runtime** on nine publicly available datasets of a wide range of problem settings and dimensionality (Table 1). For high repeatability plugin, experiments, material and additional plots and figures are available at [SuppMat, 2010].

Stability

We analyze how the produced feature sets differ under variation of the input data. We compare our two methods *Inner Ensemble* and *Fast Ensemble* to MRMR/CFS and a full ensemble of MRMR/CFS. All ensembles consist of $e = 20$ parts. The stability of the full ensemble is only reported for completeness as it technically does the same as our *Fast Ensemble*.

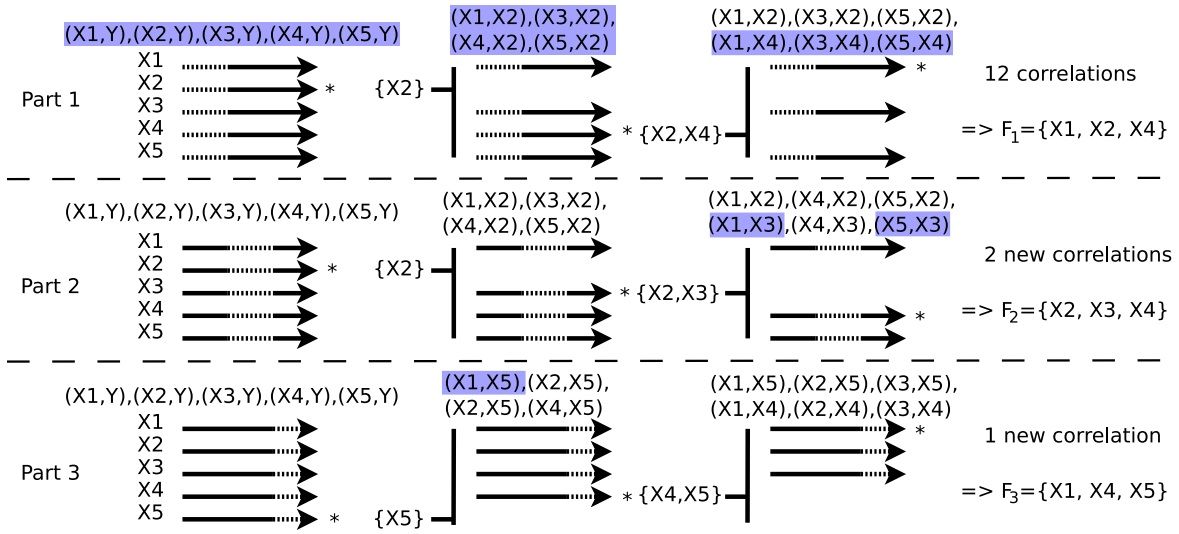


Figure 2: Fast Ensemble: A simple example with $k = 3$, $n = 5$, $e = 3$. The dashed lines in the arrows represent the subset of the examples which is left out when estimating the correlation. The correlation computations demanding a pass over the data for split-sum calculation are highlighted.

DATASET	p	n	CLASSES	DATA	$k = 10$	$k = 20$
SONAR	60	208	2	CONTINUOUS	0.0085	0.0025
IONOSPHERE	34	351	2	CONTINUOUS	0.022	0.19951
MUSK	166	476	2	CONTINUOUS	$3.1 \cdot 10^{-6}$	$1.4 \cdot 10^{-8}$
LUNG	325	73	7	NOMINAL	$4.1 \cdot 10^{-7}$	$4.0 \cdot 10^{-14}$
H.W. DIGITS	64	3823	10	CONTINUOUS	0.082	0.0058
COLON	2000	62	2	NOMINAL	$1.4 \cdot 10^{-9}$	$1.1 \cdot 10^{-6}$
LYMPHOMA	4026	96	9	NOMINAL	$1.2 \cdot 10^{-10}$	$4.4 \cdot 10^{-14}$
LEUKEMIA	7070	72	2	NOMINAL	$2.6 \cdot 10^{-11}$	$1.9 \cdot 10^{-15}$
NCI60	9712	60	9	NOMINAL	$2.4 \cdot 10^{-14}$	0.0

Table 1: Data chracteristics and significance of the difference between the stabilities achieved by *Fast Ensemble* and plain MRMR for 10 and 20 features. Values < 0.05 are highly significant.

We use the *Jaccard index* of two feature-sets

$$J(F_a, F_b) = \frac{|F_a \cap F_b|}{|F_a \cup F_b|} \tag{9}$$

to measure the stability of the feature selection. Similar to [Saeys *et al.*, 2008] we draw ten subsets from the example set like ten-fold cross-validation. On each of these subsets a feature selection is computed. The overall stability is further defined as the average of the Jaccard Indices for all combinations of those feature selections:

$$\bar{J} = \frac{2}{l^2 + l} \sum_{i=1}^{l} \sum_{j=i+1}^{l} J(F_i, F_j), \tag{10}$$

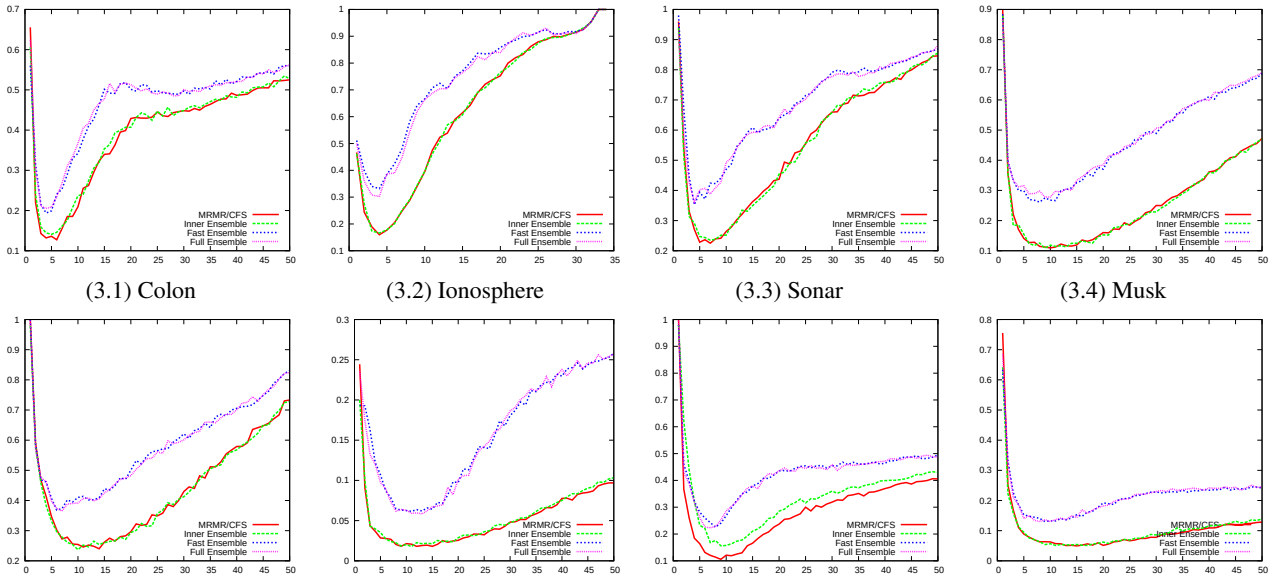
where l is the number of different feature selections in the ensemble. The average over 10 runs are reported. Fig. 3 shows exemplary results for the stability of the four feature selection methods dependent on k, the number of features to select; see [SuppMat, 2010] for complete results.

The *Fast Ensemble* version clearly outperforms the standard MRMR/CFS, whereas the *Inner Ensemble* shows nearly no visible improvement except for the *Leukemia* dataset in Fig. 3.3. For the *Leukemia* dataset the *mutual information* was used to measure relevance and redundancy. One can see that the inner ensemble only effects nominal datasets. As it increases runtime only in $O(1)$ we suggest using it for nominal datasets. As is clearly seen, our new *Fast Ensemble* achieves the same performance as a full ensemble of the same size. The benefit is the much smaller number of computations. The visible differences from 3 between methods are significant. We examplarily report the p-Values for 10 and 20 features in Table 1. The only exception in p-values is selecting 20 features from the *ionosphere* dataset, but this corresponds to the curve in Fig. 3.1. For all datasets stability increases with larger k because the (possible) overlap between subsets selected by different parts of the ensemble increases. The effect of the size e of an ensemble on the selection stability was also investigated, cf. Fig. 4 and [SuppMat, 2010]. Too small an ensemble does not increase performance and too large an ensemble degrades performance for small n. Stability increases with the number of selected features, but in general a mid-sized ensemble with $e \approx 20$ performs best for all k. We also compared our split-sum based ensembles to classical bagged ensembles of equal size. In 8 of 9 data sets *Bagging* gave less stable results.

Accuracy

We analyze if a more stable feature selection benefits classification accuracy on five different learning schemes: *Naïve Bayes*, *5-Nearest-Neighbors*, RF, a linear SVM, and *Logistic Regression* (LR) which all have different strenghts and weaknesses. We compare the standard MRMR/CFS approach (Plain) to our *Inner Ensemble* and our *Fast Ensemble* algorithm. Accuracy was averaged over 10 runs of ten-fold cross-validation. SVM and LR were only applied to two-class problems with continuous variables. Pairwise comparisons of the feature selection methods summed up over all experiments gave the following *wins/ties/losses*:

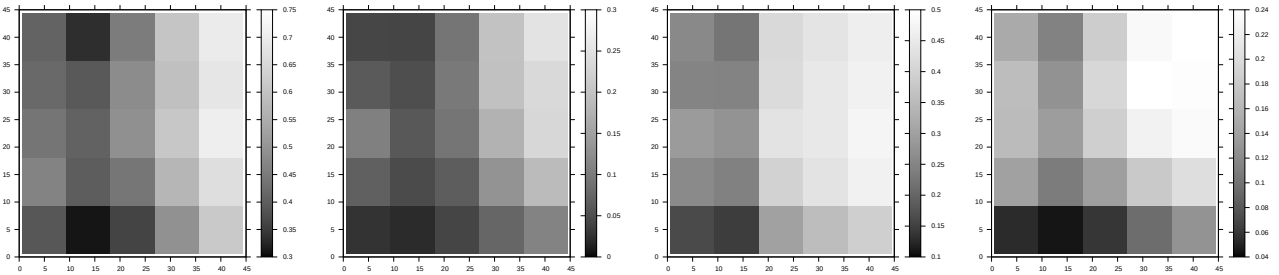
Fast Ensemble	vs	*MRMR/CFS*:	884 / 52 / 634
Inner Ensemble	vs	*MRMR/CFS*:	785 / 55 / 730
Fast Ensemble	vs	*Inner Ensemble*:	849 / 50 / 671



(3.1) Colon (3.2) Ionosphere (3.3) Sonar (3.4) Musk

(3.5) H.W. Digits (3.6) Lymphoma (3.7) Leukemia (3.8) NCI60

Figure 3: Stability of the four approaches, MRMR/CFS, *Inner Ensemble*, *Fast Ensemble*, and the full ensemble measured by the average Jaccard Index (y-axis), where k, the number of selected features is the x-axis. The *Fast Ensemble* clearly dominates the single MRMR/CFS and and delivers the same results as a full ensemble. The *Inner Ensemble* only increases stability for the *Leukemia* dataset. Plot for *Lung* can be found ind [SuppMat, 2010].



(4.1) HW Digits (4.2) Lymphoma (4.3) Leukemia (4.4) NCI60

Figure 4: Stability of the selected feature sets as a function of e (vertical axis) and k (horizontal axis). Brightness indicates stability - the brighter the better.

Table 2 shows in more detail the effect of feature selection on classification accuracy for all datasets. Feature selection was only performed on the training set and not on the test set, as this would lead to far too optimistic results. In 7 of the 9 datasets accuracy increased, when a subset of the features was used for prediction instead of all features. More detailed plots of and figures for all datasets can be found in [SuppMat, 2010]. In general, results show that *Fast Ensembles* deliver better results earlier (with less features) than MRMR/CFS, Except for the *Musk* dataset, where all feature selections degraded performance.

DATASET	5NN	NB	RF	SVM	LR
SONAR	18/13/**19**	14/**22**/13	**23**/15/11	18/**19**/12	15/**22**/12
IONOSPHERE	**18**/4/11	**17**/7/10	**19**/4/10	**20**/11/2	**18**/9/6
MUSK	4/17/**29**	2/**27**/21	**14**/6/7	**21**/13/16	18/9/**20**
LUNG	**23**/11/15	**21**/16/12	16/15/**19**		
H.W. DIGITS	**37**/7/6	**27**/8/14	17/**19**/14		
COLON	**23**/14/13	16/**20**/10	8/**22**/18		
LEUKEMIA	17/**21**/11	**20**/11/19	**21**/19/7		
LYMPHOMA	**32**/6/11	**43**/5/2	**44**/2/2		
NCI60	**20**/17/10	**31**/8/9	17/**22**/9		

Table 2: Accuracy by the number of times a method was superior to the others in the order Fast/Inner/Plain. k varied between 1 and 50. Winner is set bold.

Runtime

Let us now compare the runtime of our new *Fast Ensemble* with a standard ensemble of the same size. The overlap (number of features which two sets have in common) can be calculated based on their size k and their Jaccard index as

$$ol(\bar{J}, k) = 2 \cdot \bar{J} \cdot k/(\bar{J} + 1). \tag{11}$$

The average Jaccard index of the sets produced by the parts of the ensemble is similar to the average Jaccard index measured for Plain MRMR/CFS for inspecting stability. Considering a *Fast Ensemble* of e parts we now sum up the average amount of correlations in the parts of the ensemble. In the first part of the ensemble, all $p \cdot k - \frac{k^2-k}{2}$ correlations must be computed. In the second part, there are no correlations needed for relevance estimation, and for the redundancy check, only those features, which have not been added in the first part, must be correlated with the remaining $k - ol(\bar{J}, k)$ features. At this point, it is unclear

whether these features are added at the end or at the beginning of the selection process, i.e., whether the parts of the ensemble differ at the end or the beginning of the selection process. This determines how many features it is compared to and explains the imprecision of the estimate.

For a rough estimate assume the average probability that a feature f^i in the ith part has already been correlated to all other features in the part before is

$$P_{k,\bar{J}}(f^i) = \sum_{m=1}^{i-1} \frac{ol(\bar{J},k)}{k}(1 - P_{k,\bar{J}}(f^m)) \quad (12)$$

$$(13)$$

which reduces to

$$P_{\bar{J}}(f^i) := \frac{2\bar{J}}{\bar{J}+1} \sum_{m=1}^{i-1} (1 - P_{\bar{J}}(f^m)) \quad (14)$$

with $P_{\bar{J}}(f^1) = 0$. As seen from eq. (5), in one part, there are on average $p-(k-1)/2$ correlations to compute. When multiplied with the probability of **not** needing to compute the correlation and adding the initial relevance computation this gives a total average runtime of

$$T(p,k,e,\bar{J}) = \quad (15)$$

$$= p + (k-1)\left(p - \frac{k-1}{2}\right)\left(e - \sum_{i=1}^{e} P_{\bar{J}}(f^i)\right) \quad (16)$$

under the assumption $f_j^i = f_l^i, \forall j,l \in [1,k]$.

To give an empirical validation of this average case runtime estimation, Tables 3, 4 and 5 show the number of correlations that must be computed depending on k and e. We compare our approach with a standard ensemble of MRMR/CFS of the same size. High variance and large p can decrease the overlap between the ensemble parts, such increasing runtime as it is this overlap which speeds up runtime of our approach. It is not possible to predict in which order features are selected in different parts of the ensemble. This puts more variance to the number of needed correlations and makes it harder to predict those numbers. Nonetheless, eq. (15) seems to give a good estimate on the average case of correlation computations.

5 Conclusion

We presented two new algorithms towards selecting a maximum relevant and minimum redundant feature set. Our algorithms are more stable than the existing MRMR/CFS approach and much faster than a standard ensemble of MRMR/CFS. The speed-up is due to a faster computation of $Cor(f,f')$ based on the displacement rule and due to caching redundancy calculations from partitions. We showed that our method is well suited for feature selection on high-dimensional data as it is more robust against high variance and outliers than the single version. For the choice of $e = 20$ our algorithm is 1.4 to 19.7 times faster than a usual ensemble of MRMR/CFS.

Our methods do not rely on Mutual Information, Pearson's correlation, or the F-Test, alone. They can make use of any measure of similarity which can be split into sums. The split-sum-trick could also speed-up, e.g., Saeys' *bagged* SU [Saeys *et al.*, 2008], which builds upon MI, when replacing *Bagging* by our subset splits.

References

[Breiman, 1996] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[Breiman, 2001] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[Ding and Peng, 2005] Chris H. Q. Ding and Hanchuan Peng. Minimum redundancy feature selection from microarray gene expression data. *Journal of Bioinformatics and Computational Biology*, 3(2):185–206, 2005.

[Fox and Dimmic, 2006] Richard J. Fox and Matthew W. Dimmic. A two-sample bayesian t-test for microarray data. *BMC Bioinformatics*, 7(126), 2006.

[Freund and Schapire, 1997] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[Gulgezen *et al.*, 2009] Gokhan Gulgezen, Zehra Cataltepe, and Lei Yu. Stable and accurate feature selection. In Wray L. Buntine, Marko Grobelnik, Dunja Mladenic, and John Shawe-Taylor, editors, *ECML/PKDD (1)*, volume 5781 of *LNCS*, pages 455–468. Springer, 2009.

[Hall, 2000] Mark A. Hall. Correlation-based feature selection for discrete and numeric class machine learning. In Pat Langley, editor, *ICML*, pages 359–366. Morgan Kaufmann, 2000.

[Jurman *et al.*, 2008] Giuseppe Jurman, Stefano Merler, Annalisa Barla, Silvano Paoli, Antonio Galea, and Cesare Furlanello. Algebraic stability indicators for ranked lists in molecular profiling. *Bioinformatics*, 24(2):258–264, 2008.

[Koh *et al.*, 2007] Judice L. Y. Koh, Mong-Li Lee, Wynne Hsu, and Kai-Tak Lam. Correlation-based detection of attribute outliers. In Kotagiri Ramamohanarao, P. Radha Krishna, Mukesh K. Mohania, and Ekawit Nantajeewarawat, editors, *DASFAA*, volume 4443 of *LNCS*, pages 164–175. Springer, 2007.

[Kohavi and John, 1997] R. Kohavi and G. H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1-2):273–324, 1997.

[Michalak and Kwaśnicka, 2006] K. Michalak and H. Kwaśnicka. Correlation-based feature selection strategy in classification problems. *International Journal of Applied Mathematics and Computer Science*, 16(4):503–511, 2006.

[Saeys *et al.*, 2008] Yvan Saeys, Thomas Abeel, and Yves Van de Peer. Robust feature selection using ensemble feature selection techniques. In Walter Daelemans, Bart Goethals, and Katharina Morik, editors, *ECML/PKDD (2)*, volume 5212 of *LNCS*, pages 313–325. Springer, 2008.

[SuppMat, 2010] SuppMat. Supplemantary material for fast-ensembles of minimum redundancy feature selection, 2010. http://www-ai.cs.uni-dortmund.de/PUBDOWNLOAD/fastensembles.tar.gz.

[Tusher *et al.*, 2001] V. G. Tusher, R. Tibshirani, and G. Chu. Significance analysis of microarrays applied to the ionizing radiation response. *PNAS*, 98(9):5116–5121, April 2001.

[Vapnik, 1998] V. Vapnik. *Statistical Learning Theory*. Wiley, Chichester, GB, 1998.

Table 3: Number of computed comparisons for a standard ensemble of MRMR/CFS, our *Fast Ensemble* method, the speed gain and the estimated runtime. Figures for the *Ionosphere*, *Lung* and *Sonar* datasets. There are no figures for *Ionosphere* for more than 30 features because this dataset only contains 34 features.

		IONOSPHERE				LUNG				SONAR			
k	e	FAST	EST.	STANDARD	GAIN	FAST	EST.	STANDARD	GAIN	FAST	EST.	STANDARD	GAIN
10	5	405	504	1,475	3.64	12,505	11,578	16,025	1.28	927	1,186	2,775	2.99
20	5	567	587	2,450	4.32	20,844	17,548	31,550	1.51	1,235	1,638	5,050	4.09
30	5	594	641	2,925	4.92	26,410	19,937	46,575	1.76	1,577	1,738	6,825	4.33
40	5					29,322	22,228	61,100	2.08	1,694	1,915	8,100	4.78
50	5					32,472	23,920	75,125	2.31	1,815	1,985	8,875	4.89
10	10	424	511	2,950	6.96	16,929	17,342	32,050	1.89	1,010	1,261	5,550	5.50
20	10	550	587	4,900	8.91	24,055	20,831	63,100	2.62	1,235	1,653	10,100	8.18
30	10	595	641	5,850	9.83	29,539	21,087	93,150	3.15	1,577	1,738	13,650	8.66
40	10					31,239	22,716	122,200	3.91	1,710	1,915	16,200	9.47
50	10					33,274	24,107	150,250	4.52	1,815	1,985	17,750	9.78
10	20	424	511	5,900	13.92	22,099	21,807	64,100	2.90	884	1,267	11,100	12.56
20	20	540	587	9,800	18.15	27,550	21,576	126,200	4.58	1,269	1,653	20,200	15.92
30	20	595	641	11,700	19.66	30,820	21,158	186,300	6.04	1,577	1,738	27,300	17.31
40	20					32,269	22,727	244,400	7.57	1,725	1,915	32,400	18.78
50	20					33,669	24,109	300,500	8.93	1,815	1,985	35,500	19.56

Table 4: Number of computed comparisons for a standard ensemble of MRMR/CFS, our *Fast Ensemble* method, the speed gain and the estimated runtime. Figures for the *Musk*, *Colon* and *Handwritten Digits* datasets.

		MUSK				COLON				HW DIGITS			
k	e	FAST	EST.	STANDARD	GAIN	FAST	EST.	STANDARD	GAIN	FAST	EST.	STANDARD	GAIN
10	5	3,991	5,093	8,075	2.02	61,535	47,697	99,775	1.62	649	1,297	2,975	4.58
20	5	5,346	8,823	15,650	2.93	104,622	64,376	199,050	1.90	1,300	2,241	5,450	4.19
30	5	6,235	10,332	22,725	3.64	145,299	94,289	297,825	2.05	1,552	2,452	7,425	4.78
40	5	7,866	10,719	29,300	3.72	183,629	119,446	396,100	2.16	1,827	2,453	8,900	4.87
50	5	8,710	10,944	35,375	4.06	221,559	142,219	493,875	2.23	2,025	2,380	9,875	4.88
10	10	3,991	6,734	16,150	4.05	98,775	53,160	199,550	2.02	702	1,389	5,950	8.48
20	10	5,346	10,545	31,300	5.85	129,855	65,012	398,100	3.07	1,260	2,343	10,900	8.65
30	10	6,235	11,123	45,450	7.29	156,840	95,029	595,650	3.80	1,584	2,476	14,850	9.38
40	10	7,756	10,958	58,600	7.56	193,149	120,023	792,200	4.10	1,827	2,457	17,800	9.74
50	10	8,505	11,007	70,750	8.32	232,860	142,630	987,750	4.24	2,052	2,380	19,750	9.62
10	20	4,131	7,463	32,300	7.82	114,347	53,891	399,100	3.49	702	1,396	11,900	16.95
20	20	5,083	10,956	62,600	12.32	137,585	65,019	796,200	5.79	1,260	2,348	21,800	17.30
30	20	6,111	11,190	90,900	14.87	166,430	95,035	1,191,300	7.16	1,584	2,476	29,700	18.75
40	20	7,756	10,964	117,200	15.11	198,849	120,026	1,584,400	7.97	1,827	2,457	35,600	19.49
50	20	8,505	11,008	141,500	16.64	234,740	142,631	1,975,500	8.42	2,052	2,380	39,500	19.25

Table 5: Number of computed comparisons for a standard ensemble of MRMR/CFS, our *Fast Ensemble* method, the speed gain and the estimated runtime. Figures for the *Lymphoma*, *Leukemia* and *NCI60* datasets.

		LYMPHOMA				LEUKEMIA				NCI60			
k	e	FAST	EST.	STANDARD	GAIN	FAST	EST.	STANDARD	GAIN	FAST	EST.	STANDARD	GAIN
10	5	176,198	170,514	201,075	1.14	267,957	214,363	353,275	1.32	416,713	354,883	485,375	1.16
20	5	354,398	349,559	401,650	1.13	464,475	325,976	706,050	1.52	860,452	768,263	970,250	1.13
30	5	526,680	488,337	601,725	1.14	667,185	414,636	1,058,325	1.59	1,244,592	1,064,589	1,454,625	1.17
40	5	677,766	590,451	801,300	1.18	834,309	506,060	1,410,100	1.69	1,598,497	1,289,449	1,938,500	1.21
50	5	800,598	694,083	1,000,375	1.25	952,340	597,310	1,761,375	1.85	1,960,542	1,522,741	2,421,875	1.24
10	10	314,973	305,016	402,150	1.28	471,479	275,841	706,550	1.50	754,533	539,036	970,750	1.29
20	10	623,705	617,372	803,300	1.29	681,134	354,470	1,412,100	2.07	1,369,093	1,222,147	1,940,500	1.42
30	10	865,436	787,610	1,203,450	1.39	876,000	429,598	2,116,650	2.42	1,884,442	1,548,606	2,909,250	1.54
40	10	1,065,723	859,902	1,602,600	1.50	1,007,784	516,370	2,820,200	2.80	2,292,727	1,718,300	3,877,000	1.69
50	10	1,218,735	955,178	2,000,750	1.64	1,125,390	605,302	3,522,750	3.13	2,651,998	1,939,365	4,843,750	1.83
10	20	565,565	501,460	804,300	1.42	590,394	299,482	1,413,100	2.39	1,177,483	689,700	1,941,500	1.65
20	20	1,024,385	985,833	1,606,600	1.57	827,357	357,243	2,824,200	3.41	2,226,388	1,656,234	3,881,000	1.74
30	20	1,296,456	1,086,820	2,406,900	1.86	987,000	430,168	4,233,300	4.29	2,802,838	1,872,590	5,818,500	2.08
40	20	1,487,010	1,040,597	3,205,200	2.16	1,111,569	516,588	5,640,400	5.07	3,272,563	1,910,171	7,754,000	2.37
50	20	1,631,750	1,091,347	4,001,500	2.45	1,242,707	605,412	7,045,500	5.67	3,515,832	2,085,675	9,687,500	2.76

[Xu and Zhang, 2006] Xian Xu and Aidong Zhang. Boost feature subset selection: A new gene selection algorithm for microarray dataset. In *ICCS (2)*, pages 670–677, 2006.

[Yu and Liu, 2004] Lei Yu and Huan Liu. Efficient feature selection via analysis of relevance and redundancy. *JMLR*, 5:1205–1224, 2004.