Speeding Up Feature Subset Selection Through Mutual Information Relevance Filtering

Gert Van Dijck and Marc M. Van Hulle

Katholieke Universiteit Leuven, Computational Neuroscience Research Group, bus 1021, B-3000 Leuven, Belgium
Gert.VanDijck@mtm.kuleuven.be,
Marc.VanHulle@med.kuleuven.be

Abstract. A relevance filter is proposed which removes features based on the mutual information between class labels and features. It is proven that both feature independence and class conditional feature independence are required for the filter to be statistically optimal. This could be shown by establishing a relationship with the conditional relative entropy framework for feature selection. Removing features at various significance levels as a preprocessing step to sequential forward search leads to a huge increase in speed, without a decrease in classification accuracy. These results are shown based on experiments with 5 high-dimensional publicly available gene expression data sets.

1 Introduction

With the ever increasing feature dimensionality in pattern recognition problems such as in gene expression data [1], text categorization problems or image classification problems, it is important to develop algorithms which increase the speed of the search for optimal feature subsets. Nowadays, feature dimensionalities often reach 10000 to 100000 features.

Besides the speed of a feature selection approach it is important to prove under which conditions optimality will be achieved. In this paper we prove the optimality of an information theoretic (mutual information-based) relevance filter. It can be proven under which conditions it will be optimal by establishing a formal relationship with the *conditional relative entropy* framework from [2]. This relationship will be derived in section 2. Special care in this article is taken towards statistical significance of feature relevance by means of permutation testing, this is explored in section 2.

Experiments in section 4 show that an increase in speed in the wrapper search [3] is obtained if it is preceded by the relevance filter. Although optimality of the filter requires conditions, such as feature independence and class conditional feature independence, the experiments show that the filter does not decrease the classification performance of the feature sets found in the wrapper search.

2 Relevance Filtering

Several other authors have been proposing mutual information as a feature selection criterion: see e.g. MIFS [4], for classification problems and [5] for regression

J.N. Kok et al. (Eds.): PKDD 2007, LNAI 4702, pp. 277–287, 2007.

problems. Here, we show that a mutual information criterion for classification problems can be derived from the conditional relative entropy framework for feature selection [2]. This relationship between conditional relative entropy framework for feature selection and mutual information for feature selection has not been established before.

2.1 Optimality of Marginal Feature Relevance

Following lemma establishes a link between the conditional relative entropy framework and marginal mutual information.

Lemma 1: If features are both independent and class conditional independent, then the *conditional relative entropy* (a special case of the Kullback-Leibler divergence, KL) between 2 posterior probabilities is equivalent to a sum of mutual information (MI) contributions of the class label with the omitted features individually.
More formally this can be written as follows.

$$\textit{if independence}: \forall F_1, F_2, ..., F_n : p(F_1, F_2, ..., F_n) = \prod_{i=1}^{n} p(F_i)$$

$$\textit{and class conditional independence}: \forall F_1, F_2, ..., F_n, C : p(F_1, F_2, ..., F_n \mid C) = \prod_{i=1}^{n} p(F_i \mid C)$$

$$\textit{then } KL(P(C \mid F_1, F_2, ..., F_n) \,\|\, P(C \mid F_{l_1}, F_{l_2}, ..., F_{l_{n1}})) = \sum_{m_i} MI(C; F_{m_i})$$

$$\textit{with } \left\{F_{l_1}, F_{l_2}, ..., F_{l_{nl}}\right\} \subset \left\{F_1, F_2, ..., F_n\right\} \textit{ and}$$

$$\left\{F_{m_1}, F_{m_2}, ..., F_{m_{n2}}\right\} = \left\{F_1, F_2, ..., F_n\right\} \setminus \left\{F_{l_1}, F_{l_2}, ..., F_{l_{nl}}\right\}$$

Where, F_1, …F_n refer to the full set of features, indices l_1, …l_{n1} refer to the included features and indices m_1, …m_{n2} refer to the omitted features. Note that the full feature set consists of the union of the set of included and omitted features and hence n = n1 + n2. Firstly, we remark that feature independence (condition 1) is not a sufficient condition, also class conditional independence (condition 2) is required. This should not come as a surprise, because the Naïve Bayes (NB) classifier also requires class conditional independence. Despite these assumptions, experiments have shown that the NB outperforms many other classifiers [6].

Proof
Starting from the definition of the conditional relative entropy between 2 posterior probabilities $P(C \mid F_1, F_2, ..., F_n)$ and $P(C \mid F_{l_1}, F_{l_2}, ..., F_{l_{n1}})$, and using the convention that 0ln(0) = 0:

$$\sum_{c} \iiint_{f} P(c, f_1, f_2, ..., f_n) \ln\left[\frac{P(c \mid f_1, f_2, ..., f_n)}{P(c \mid f_{l_1}, f_{l_2}, ..., f_{l_{n1}})}\right] df_1 df_2 ... df_n \tag{1}$$

Using Bayes' theorem and dividing both the numerator and the denominator by P(c) within the logarithm this is equivalent to:

$$= \sum_{c} \iiint_{f} P(c, f_1, f_2, ..., f_n) \ln \left[\frac{\frac{P(f_1, f_2, ..., f_n \mid c)P(c)}{p(f_1, f_2, ..., f_n)P(c)}}{\frac{P(f_{l_1}, f_{l_2}, ..., f_{l_{n1}} \mid c)P(c)}{p(f_{l_1}, f_{l_2}, ..., f_{l_{n1}})P(c)}} \right] df_1 df_2 ... df_n \tag{2}$$

Using the assumption of both independence and class conditional independence this can be further written as:

$$\begin{aligned} &= \sum_{c} \iiint_{f} P(c, f_1, f_2, ..., f_n) \ln \frac{\prod_{i=1}^{n} P(f_i \mid c)P(c)}{\prod_{i=1}^{n} P(f_i)P(c)} df_1 df_2 ... df_n \\ &- \sum_{c} \iiint_{f} P(c, f_1, f_2, ..., f_n) \ln \frac{\prod_{i=1}^{n_1} P(f_{l_i} \mid c)P(c)}{\prod_{i=1}^{n_1} P(f_{l_i})P(c)} df_1 df_2 ... df_n \end{aligned} \tag{3}$$

Using the definition of conditional probabilities this is equal to:

$$\begin{aligned} &= \sum_{i=1}^{n} \sum_{c} \iiint_{f} P(c, f_1, f_2, ..., f_n) \ln \frac{P(f_i, c)}{P(f_i)P(c)} df_1 df_2 ... df_n \\ &- \sum_{i=1}^{n_1} \sum_{c} \iiint_{f} P(c, f_1, f_2, ..., f_n) \ln \frac{P(f_{l_i}, c)}{P(f_{l_i})P(c)} df_1 df_2 ... df_n \end{aligned} \tag{4}$$

Integrating out variables not appearing within the logarithms and applying the definition of mutual information: $MI(F_i;C) = \sum_{c} \int_{F_i} P(c, f_i) \ln \frac{P(f_i, c)}{P(f_i)P(c)} df_i$, we finally obtain:

$$= \sum_{i=1}^{n} MI(F_i;C) - \sum_{i=1}^{n_1} MI(F_{l_i};C) \tag{5}$$

Which is equivalent to:

$$= \sum_{i=1}^{n_2} MI(F_{m_i};C) \tag{6}$$

This ends the proof. ❑

This means that in going from the full set $\{F_1, F_2, ..., F_n\}$ to a subset $\{F_{l_1}, F_{l_2}, ..., F_{l_{n1}}\} \subset \{F_1, F_2, ..., F_n\}$, the distance between the full set and the subset, i.e.

$KL(P(C \mid F_1, F_2, ..., F_n) \| P(C \mid F_{l_1}, F_{l_2}, ..., F_{l_{n1}}))$, consists of the individual mutual information contributions between the omitted features and the class variable. As shown in the derivation, this holds under the assumption of both feature independence and class conditional feature independence. Knowing that the mutual information is always larger than or equal to 0, see [7], we can remove the features for which $MI(F_i;C) = 0$. Hence, features for which the mutual information with the class variable C is 0 can be omitted, without an increase in distance to full posterior probability. The effect of the filter on the performance of feature subset selection will be investigated in the experimental section of this article. However, there are 2 very important reasons for making the assumptions: accuracy and speed. As the result of the theorem shows, we can limit ourselves to 1 dimensional estimators of the mutual information. In general, the accuracy of mutual information estimators decreases with increasing dimensionality and fixed number of data points. Secondly, the mutual information between the class label and each feature can be computed independently from every other feature. Hence, there is a large potential in the increase in speed, while all computations can be scheduled in parallel.

2.2 Permutation Testing

As shown in lemma 1, under appropriate conditions, features for which the mutual information with the class variable, $MI(F_i;C)$, equals 0 can be removed. However, testing whether $MI(F_i;C) = 0$, is not straightforward. Mutual information [7] is defined based on the true underlying density $P(F_i,C)$ and this density is in general inaccessible. Therefore, one needs to rely on finite sample estimators: $\widehat{MI}(F_i;C)$ of the mutual information. Estimators of the mutual information based on histograms are biased [8], due to the discretization in intervals and thus need to be avoided. In this article the mutual information is estimated by a recent k-nearest neighbor approach to entropy estimation which does not need a discretization of the features [9]:

$$\hat{H}(F_i) = -\psi(k) + \psi(N) + \log c_d + \frac{d}{N}\sum_{i=1}^{N} \log \varepsilon(i) \tag{7}$$

Here $\psi(.)$ is the psi-function, N the number of data points, $\varepsilon(i)$ is twice the distance from the i'th data point to its k'th neighbor, d the dimensionality and c_d the volume of the d-dimensional unit ball. For the maximum norm one has $c_d = 1$, for the Euclidean norm $c_d = \pi^{d/2}/\Gamma(1 + d/2)$, with $\Gamma(.)$ the gamma-function. Because features are considered individually, d is equal to 1.
The mutual information can then be computed from the entropy by means of:

$$\widehat{MI}(F_i;C) = \hat{H}(F_i) - \sum_{j=1}^{k} \hat{H}(F_i \mid c_j) P(c_j) \tag{8}$$

Here $\hat{H}(F_i \mid c_j)$ is the entropy of the feature F_i conditioned on the class c_j, $P(c_j)$ is the probability for the j'th class. We take 'k' equal to 6 in the experiments.

An intelligent approach to test whether the data is likely to be a realization of a certain null-hypothesis can be obtained by manipulating the data such that the

null-hypothesis is fulfilled and computing the test-statistic under these manipulations. Hereto, class labels are randomly permuted; this causes the class variable to be independent of the feature F_i. Any statistical dependence that is left is contributed to coincidence. This procedure is known as permutation testing. We restrict ourselves to 1000 permutations.

3 Feature Subset Selection

Here, we propose the wrapper subset selection that will be used in the experiments. The wrapper selection consists of following components: search procedure, induction algorithm and the criterion function to evaluate a particular subset.

3.1 Search Procedure

Many feature selection algorithms have been proposed and explored in the literature. The most well-known search procedures are: sequential search algorithms [10], Branch and bound [11], Genetic algorithms [12], [13]. Sequential forward search (SFS) is the algorithm being used here. We discuss it in more detail here, because it will allow us to explain some of the observations made in the experiments.

Suppose we have a set **Y** of D available features: $\mathbf{Y} = \{F_1, F_2, \ldots F_D\}$; denote the set of k selected features by $\mathbf{X}_k = \{x_i: 1 \leq i \leq k, x_i \in Y\}$. Suppose that we also dispose of a feature selection criterion function J(.). The particular choice of J(.) will be addressed in 3.2. The following pseudo-code defines the SFS algorithm. We implicitly assume in the input that we have feature realizations $\mathbf{f}^j = [f_1^j, f_2^j, \ldots f_D^j]$ and class labels c^j to our disposal, where index 'j' refers to a particular data point. This is written in shorthand notation by the variable notation $F_1, F_2, \ldots F_D$ and C in the input. In the initialization, we set the performance of the empty set $J(\mathbf{X}_{opt} = \varnothing)$ equal to 0. It is assumed that 0 is the minimum value of J(.). In each iteration of the body of the pseudo-code, the most significant feature is searched for, until some predefined dimension d.

Input: features $F_1, F_2, \ldots F_D$; class labels C; dimension d
Initialize: $\mathbf{Y} = \{F_1, F_2, \ldots F_D\}$; $\mathbf{X}_0 = \varnothing$; $\mathbf{X}_{opt} = \varnothing$; $J(\mathbf{X}_{opt}) = 0$
For k = 0 to d-1
$F_i' = argmax_{Fi \in Y} J(\mathbf{X}_k \cup \{F_i\})$;determine the most significant feature F_i' in **Y**.
$\mathbf{X}_{k+1} = \mathbf{X}_k \cup F_i'$;update the selected feature set.
$Y = Y - \{F_i'\}$;update the set of available features.
if $(J(\mathbf{X}_{k+1}) > J(\mathbf{X}_{opt}))$; update the optimal feature set.
$\mathbf{X}_{opt} = \mathbf{X}_{k+1}$
end
end
Output: $\mathbf{X}_{opt}$

Here it is assumed that $d \leq D$. The most significant feature F'_i is added to the existing subset ($\mathbf{X}_k$) to form a new subset of features ($\mathbf{X}_{k+1}$). Subsequently this most significant feature is removed from the set of available features $\mathbf{Y}$. Only when the performance of the new subset is better than the optimal set found so far ($\mathbf{X}_{opt}$), the optimal subset is updated. Finally, the feature set with the highest performance is given as an output. If this performance is obtained for several subsets, the smallest subset will be returned, this is due to the strict inequality constraint in the update of $\mathbf{X}_{opt}$.

The SFS is among the feature selection algorithms with the lowest time complexity: $O(D^2)$, see [14], with 'D' the number of features in the full feature set. BAB, SFFS, SBFS have time complexities in the order of $O(2^D)$.

A second motivation for the use of SFS can be found in table 3 of [12]. Here, we observe that the performances of SFS on different data sets are often only a few percent smaller than SFFS and the hybrid genetic algorithms (HGA's). Moreover, the reported results are obtained with leave-one-out cross-validation (LOO-CV) without considering separate training and test sets. It may be that, with the use of separate training sets and test sets, the advantage of the more complex search algorithms, when tested on independent test sets and when taking statistical significance into account, will vanish.

We work with independent test and training sets. In this case the problem of model selection (feature subset selection) and model evaluation is performed on different data sets and this allows the assessing of the overfitting behavior of feature subset selection. Moreover, when using several test sets, the statistical significance of the combined hybrid filter/wrapper approach can be compared to the stand-alone wrapper approach. The creation of several training and test sets is further explained in the experimental section of this paper.

3.2 Performance Estimation

As the performance measure J(.), we use the leave-one out cross-validation (LOO-CV) criterion. Note, that this measure is in general non-monotonic: adding a feature to an existing set $\mathbf{X}_k$, can either increase ($J(\mathbf{X}_{k+1}) \geq J(\mathbf{X}_k)$) or decrease ($J(\mathbf{X}_{k+1}) \leq J(\mathbf{X}_k)$) the LOO-CV performance. If LOO-CV is used as a criterion the lowest value is 0 and hence the initialization ($J(\mathbf{X}_{opt} = \varnothing) = 0$) in the pseudo-code is correct. If the LOO-CV performance is used, the maximum value that can be achieved is equal to 1. Once this optimal performance is achieved, the loop in the pseudo-code does not need to run until d -1, because no subset with more features can achieve a higher-performance. As the induction algorithm we use the k nearest neighbor approach (k-NN), with k equal to 1. The use of this induction algorithm is motivated by: the ease of use and important feature subset selection articles [12], [14] have been using k-NN as well.

4 Experiments

The data sets used in the experiments are 5 gene-expression data sets: ALL-AML leukemia [15], central nervous system (CNS) [16], colon tumor [17], diffuse large B-cell lymphoma (DLBCL) [18] and MLL leukemia [19]. These data sets are

summarized in table 1. These are challenging data sets due to the number of dimensions and the limited number of samples available. Some preprocessing steps have been applied.

Firstly, features are normalized by subtracting the mean and dividing by the standard deviation. Secondly, if missing values exist, these are replaced by sampling from a normal distribution with the same mean and standard deviation as the samples from the same class for which values are available. If no separate training sets and tests were available, approximately two-third of the samples of each class was assigned to the training set and the rest to the test set. The number of samples assigned to the training ad test sets for each data set are summarized in table 1. Note that the first 4 data sets form a binary classification problem, and the fifth a ternary one. For the sake of repeatability and comparability of our results, we do not consider the performance on a single training and test set, but also on reshuffled versions.

Table 1. Summary of properties of the data sets: TR, training set; TE, test set; C1, class 1; C2, class 2; C3, class 3; N, number of features

Data Set	TR	TR C1	TR C2	TE	TE C1	TE C2	N
ALL	38	27	11	34	20	14	7129
CNS	39	25	14	21	14	7	7129
Colon	42	27	15	20	13	7	2000
DLBCL	32	16	16	15	8	7	4026
		20	17		4	3	
MLL		C3			C3		
	57	20		15	8		12582

Each data set is reshuffled 20 times and has the same number of samples from each class assigned to the training and test set, as shown in table 1. Where possible, these reshufflings are stratified: they contain the same number of examples of each class in the training and the test set as in the original data set. The reshuffling strategy implies that a feature selection procedure can be run 20 times on a different training set of the same data set. The outcome of each run can than be used to compute the test result on the associated test set. This also allows one to assess the variability of the test result and to test the statistical significance of the test results.

4.1 Discussion

In table 2, the results on the 5 data sets are shown for different p-values of the relevance filter. For p = 0, no features are removed and thus the sequential forward search is run on the full dimensionality of the data sets. Hence, this allows to compare the proposed hybrid approach to the stand-alone wrapper approach. For p = 0.1, 0.2 and so on, only those features are retained which exceed the [p*1000] order statistic of the mutual information. The testing performances are clearly much lower and change only slightly with changing p. However, due to the large standard deviations, these changes cannot be shown to be significant with changing 'p'. The first row for each data set in table 2 contains the training performances averaged over the 20 different training sets.

The second row contains the average test performances averaged over the 20 different test sets. The standard deviations of the performances for the training and test sets are shown as well.

The third row of each data set contains the speed-up factor. This has been obtained by summing the time needed to run the SFS algorithm on the 20 training sets for a particular p-value, divided by the same sum without filtering (p = 0). This explains the '1' values in the p = 0 column. The time needed for the filter is ignored: the computations of the MI for the 1000 permutations can be run in parallel; moreover the MI computations for all features can be run in parallel as well, because feature dependencies are ignored. This causes the time of the filter not more than tens of seconds to a few seconds.

From the training performances in table 2, we observe that the wrapper search finds in many cases subsets for which the LOO-CV performances on the training sets is 100% or very close to 100%. The filter has no effect on these training performances, except for the CNS and the colon data set filtered at the p = 1 level. Here considerable lower training performances are obtained: 82.2% and 89.0% respectively.

The average test performances are considerable lower than their corresponding training performances. This shows that it is very important to consider separate training and test sets in these small sample size settings. The SFS procedure tends to overfit heavily. In order to assess whether the filter improves the test performance compared to the p = 0 (unfiltered) test performance, hypothesis testing has been applied. This is indicated by the symbols '0' and '+' beneath the testing performances. A '0' indicates no change and the '+' indicates an improvement. The first symbol is the result of applying a t-test at the $\alpha = 0.05$ significance level, this tests whether the mean of the p = 0 level is different from the mean at the other levels. The second symbol is the result of applying the Wilcoxon rank-sum test at the $\alpha = 0.05$ level. The t-test assumes Gaussianity of the average performances and is therefore sensitive to outliers in the test performance.

An interesting observation is that by filtering at subsequent p-levels, e.g. p = 0, 0.1 and 0.2, sometimes exactly the same subsets are obtained in SFS. This can be explained as follows. Suppose that 5 features were sufficient to obtain a 100% training performance. If at the next higher p-level none of these features are removed, then they will be selected again as the optimal subset. This is due to the deterministic

Table 2. Result of the sequential forward search with various levels of relevance filtering. Average recognition rate results with standard deviations are shown for training and test sets. The speed-up factors for the filtered data sets are shown as well.

Data Set	P = 0.0	P = 0.1	P = 0.2	P = 0.4	P = 0.6	P = 0.8	P = 0.9	P = 1.0
ALL train	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0
ALL test	84.0 ± 10.0	82.9 ± 10.0 0 0	83.8 ± 10.6 0 0	82.9 ± 10.2 0 0	83.1 ± 10.9 0 0	85.1 ± 9.6 0 0	86.0 ± 8.4 0 0	86.9 ± 5.7 0 0
ALL speed up	1	1.42	1.54	1.84	2.39	3.72	4.17	63.69
CNS train	99.5 ± 1.1	99.5 ± 1.3	99.7 ± 0.8	99.6 ± 0.9	99.4 ± 1.1	99.1 ± 1.7	99.2 ± 1.2	82.2 ± 4.9
CNS test	55.2 ± 10.1	56.7 ± 8.9 0 0	54.8 ± 8.5 0 0	54.8 ± 10.3 0 0	57.1 ± 10.1 0 0	57.6 ± 8.7 0 0	59.3 ± 9.9 0 0	66.2 ± 9.0 + +
CNS speed up	1	1.15	2.12	2.62	2.61	4.67	10.94	2352
Colon train	99.9 ± 0.5	99.9 ± 0.5	100.0 ± 0	99.8 ± 0.7	99.8 ± 0.7	99.8 ± 0.7	99.5 ± 1.0	89.0 ± 4.3
Colon test	69.5 ± 9.6	72.7 ± 7.7 0 0	73.8 ± 9.4 0 0	73.0 ± 7.5 0 0	74.8 ±8.5 0 0	70.3 ± 9.1 0 0	75.5 ± 10.5 0 0	76.0 ± 10.6 + 0
Colon speed up	1	1.07	1.91	1.38	1.59	3.63	4.88	309.7
DLBCL train	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0
DLBCL test	75.7 ± 13.2	75.0 ± 12.6 0 0	75.0 ± 12.6 0 0	75.7 ± 12.5 0 0	78.3 ± 10.3 0 0	81.3 ± 7.7 0 0	82.3 ± 7.9 0 0	80.3 ± 11.5 0 0
DLBCL speed up	1	1.40	1.50	1.38	2.44	3.0	4.36	46.05
MLL train	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0
MLL test	84.0 ± 11.7	84.0 ± 11.7 0 0	84.0 ± 11.7 0 0	82.7 ± 10.9 0 0	82.7 ± 10.9 0 0	84.7 ± 8.9 0 0	81.0 ± 9.7 0 0	88.0 ± 9.3 0 0
MLL speed up	1	1.03	1.05	1.37	1.67	1.83	1.76	6.70

behavior of the SFS procedure. This explains the occurrence of the same test performances for subsequent p-values: this can be observed in the MLL test performances (for p = 0, 0.1 and 0.2) and to a smaller extent in the DLBCL data set (for p = 0.1 and 0.2).

From table 2, it can be observed that in almost all cases no significant change in the test performance is obtained after filtering and that both the t-test and Wilcoxon rank-sum test agree on this. A significant improvement is obtained for the CNS data set and the colon data set both filtered at the p = 1 level, although in the latter data set the t-test and the Wilcoxon rank-sum test disagree on this. It is also for these cases that the LOO-CV training performance is lower. Hence, it seems that filtering at a high significance level reduces somewhat the overfitting behavior.

The most interesting results are obtained with the increase of the speed of the SFS algorithm on the filtered data sets. Especially the gain in speed for the p = 0.9 and p = 1 are large. For p = 1 the speed up factors are respectively: 63.69, 2352, 309.7, 46.05, 6.70. The large differences in some of these gains can be explained by the number of features left compared to the full dimensionality. These are for the different data sets respectively: 169 out of 7129, 10 out of 7129, 10 out of 2000, 85 out of 4026 and 2011 out of 12582. In some cases, it is possible that the gain in speed does not increase with a small increase in p. This can be explained by the fact that when initially in the search a few good genes are found with 100% LOO-CV performance, the SFS can be stopped. However, if a data set is filtered at a higher p-level, some of the genes may be removed. The SFS procedure might then need to search a longer time to find a 100% LOO-CV subset, or even worse it needs to execute all iterations of the loop. This can be observed in the DLBCL data set (when p is changed from p = 0.2 to 0.4) and in the colon data set (when p is changed from p = 0.2 to 0.4 and 0.6).

5 Conclusions

It is proven that features, for which the mutual information is equal to 0, can be removed without loss of information, if features are both independent and class conditional independent. This result has been obtained by establishing the link with the conditional relative entropy framework for feature subset selection. This allowed motivating mutual information as an optimal criterion in feature selection, rather than a heuristic one. Removing features at various significance levels of the mutual information statistic prior to sequential forward search (SFS) with the 1-NN method does not decrease the average performance of the test sets. A huge increase in speed can be obtained by first filtering features at the 0.9 or 1.0 level.

Acknowledgements. The first author is supported by the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT Vlaanderen).

The second author is supported by the Excellence Financing program of the K.U. Leuven (EF 2005), the Belgian Fund for Scientific Research Flanders (G.0248.03, G.0234.04), the Flemish Regional Ministry of Education (Belgium) (GOA 2000/11), the Belgian Science Policy (IUAP P5/04), and the European Commission (NEST-2003-012963, STREP-2002-016276, and IST-2004-027017). This work made use of the HPC (High Performance Computing) infrastructure of the K. U. Leuven.

References

1. Quackenbush, J.: Microarray Analysis and Tumor Classification. The New England Journal of Medicine 354, 2463–2472 (2006)
2. Koller, D., Sahami, M.: Toward Optimal Feature Selection. In: Proceedings of the Thirteenth International Conference on Machine Learning, pp. 284–292. Morgan Kaufmann, San Francisco (1996)
3. Kohavi, R., John, G.H.: Wrappers for Feature Subset Selection. Artificial Intelligence 97, 273–324 (1997)
4. Battiti, R.: Using Mutual Information for Selecting Features in Supervised Neural Net Learning. IEEE Transactions on Neural Networks 5, 537–550 (1994)
5. Van Dijck, G., Van Hulle, M.: Speeding-up the Wrapper Feature Subset Selection in Regression by Mutual Information Relevance and Redundancy Analysis. In: Proceedings of the 16th International Conference on Artificial Neural Networks, pp. 31–40 (2006)
6. Domingos, P., Pazzani, M.: On the Optimality of the Simple Bayesian Classifier under Zero-one Loss. Machine Learning 29, 103–130 (1997)
7. Cover, T.M., Thomas, J.A.: Elements of Information Theory, 2nd edn. John Wiley & Sons, Hoboken New Jersey (2006)
8. Paninski, L.: Estimation of Entropy and Mutual Information. Neural Computation 15, 1191–1253 (2003)
9. Kraskov, A., Stögbauer, H., Grassberger, P.: Estimating Mutual Information. Physical Review E 69, 066138-1 – 066138-16 (2004)
10. Pudil, P., Novovičová, J., Kittler, J.: Floating Search Methods in Feature Selection. Pattern Recognition Letters 15, 1119–1125 (1994)
11. Narendra, P.M., Fukunaga, K.: A Branch and Bound Algorithm for Feature Subset Selection. IEEE Transactions on Computers C- 26, 917–922 (1977)
12. Oh, I.-S., Lee, J.-S., Moon, B.-R.: Hybrid Genetic Algorithms for Feature Selection. IEEE Transactions on Pattern Analysis and Machine Intelligence 26, 1424–1437 (2004)
13. Siedlecki, W., Sklansky, J.: A Note on Genetic Algorithms for Large-scale Feature Selection. Pattern Recognition Letters 10, 335–347 (1989)
14. Kudo, M., Sklansky, J.: Comparison of Algorithms that Select Features for Pattern Classifiers. Pattern Recognition 33, 25–41 (2000)
15. Golub, T.R., et al.: Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression Monitoring. Science 286, 531–537 (1999)
16. Pomeroy, S.L., et al.: Prediction of Central Nervous System Embryonal Tumour Outcome Based on Gene Expression. Nature 415, 436–442 (2002)
17. Alon, U., et al.: Broad Patterns of Gene Expression Revealed by Clustering Analysis of Tumor and Normal Colon Tissues Probed by Oligonucleotide Arrays. In: Proceedings of the National Academy of Sciences of the United States of America vol. 96, pp. 6745–6750 (1999)
18. Alizadeh, A.A., et al.: Distinct Types of Diffuse Large B-cell Lymphoma Identified by Gene Expression Profiling. Nature 403, 503–511 (2000)
19. Armstrong, S.A., et al.: MLL Translocations Specify a Distinct Gene Expression Profile that Distinguishes a Unique Leukemia. Nature Genetics 30, 41–47 (2002)