Searching for Interacting Features

Zheng Zhao and Huan Liu
Department of Computer Science and Engineering
Arizona State University
{zheng.zhao, huan.liu}@asu.edu

Abstract

Feature interaction presents a challenge to feature selection for classification. A feature by itself may have little correlation with the target concept, but when it is combined with some other features, they can be strongly correlated with the target concept. Unintentional removal of these features can result in poor classification performance. Handling feature interaction can be computationally intractable. Recognizing the presence of feature interaction, we propose to efficiently handle feature interaction to achieve efficient feature selection and present extensive experimental results of evaluation.

1 Introduction

The high dimensionality of data poses a challenge to learning tasks such as classification. In the presence of many irrelevant features, classification algorithms tend to overfit training data [Guyon and Elisseeff, 2003; Dash and Liu, 1997]. Many features can be removed without performance deterioration [Gilad-Bachrach *et al.*, 2004]. Feature selection is one effective means to remove irrelevant features [Blum and Langley, 1997]. Optimal feature selection requires an exponentially large search space ($O(2^N)$, where N is the number of features) [Almuallim and Dietterich, 1994]. Researchers often resort to various approximations to determine relevant features (e.g., relevance is determined by correlation between individual features and the class) [Hall, 2000; Yu and Liu, 2003]. However, a single feature can be considered irrelevant based on its correlation with the class; but when combined with other features, it becomes very relevant. Unintentional removal of these features can result in the loss of useful information and thus may cause poor classification performance. This is studied in [Jakulin and Bratko, 2003] as attribute interaction. For example, MONK1 is a data set involving feature interaction. There are six features in MONK1 and the target concept of MONK1 is: $(A_1 = A_2)$ or $(A_5 = 1)$. Here A_1 and A_2 are two interacting features. Considered individually, the correlation between A_1 and the class C (similarly for A_2 and C) is zero, measured by mutual information. Hence, A_1 or A_2 is irrelevant when each is individually evaluated. However, if we combine A_1 with A_2, they are strongly relevant in defining the target concept. An intrinsic character of feature interaction is its irreducibility [Jakulin and Bratko, 2004], i.e., a feature could lose its relevance due to the absence of its interacting feature(s).

Existing efficient feature selection algorithms usually assume feature independence [Dash and Liu, 1997; Hall, 2000]. Because of the irreducible nature of feature interactions, these algorithms cannot select interacting features such as A_1 and A_2 in MONK1. Others attempt to explicitly address feature interactions by finding some low-order interactions (2- or 3-way). In [Jakulin and Bratko, 2003], the authors suggest to use interaction gain as a practical heuristic for detecting attribute interaction. Using interaction gain, their algorithms can detect if datasets have 2-way (one feature and the class) and 3-way (two features and the class) interactions. They further provide in [Jakulin and Bratko, 2004] a justification of the interaction information, and replace the notion of 'high' and 'low' in [Jakulin and Bratko, 2003] with statistical significance and illustrate the significant interactions in the form of interaction graph. Below we apply four feature selection algorithms to synthetic data with known interaction and observe how they fare: FCBF [Yu and Liu, 2003], CFS [Hall, 2000], ReliefF [Kononenko, 1994], and FOCUS [Almuallim and Dietterich, 1994], all available in WEKA [Witten and Frank, 2005].

Motivating examples: Synthetic data with known feature interaction. Four synthetic data sets are used to examine how various algorithms deal with known feature interactions in feature selection. The first data set is Corral [John *et al.*, 1994], having six boolean features A_0, A_1, B_0, B_1, I, R. The class Y is defined by $Y = (A_0 \wedge A_1) \vee (B_0 \wedge B_1)$ and features A_0, A_1, B_0, B_1 are independent of each other. Feature I is irrelevant to Y and its values have a uniform random distribution; and feature R is correlated with Y 75% of the time and is redundant. The other three training data sets are MONKs data. Their target Concepts are: (1) MONK1: $(A_1 = A2)$ or $(A_5 = 1)$; (2) MONK2: Exactly two of $A_1 = 1, A_2 = 1, A_3 = 1, A_4 = 1, A_5 = 1, A_6 = 1$; and (3) MONK3: ($A_5 = 3$ and $A_4 = 1$) or ($A_5 \neq 4$ and $A_2 \neq 3$) (5% class noise added to the training data).

Results are presented in Table 1. For Corral, all four algorithms remove the irrelevant feature I, but only FOCUS removes the redundant feature R. Features A_0, A_1 and B_0, B_1 interact with each other to determine the class label of an instance. CFS, FCBF and ReliefF cannot remove R because R

Table 1: Features selected by each algorithm on artificial data, and the '_' indicates a missing relevant feature.

	Relevant Features	FCBF	CFS	ReliefF	FOCUS
Corral	A_0, A_1, B_0, B_1	$A_0, A_1, B_0, B_1, \mathbf{R}$	$A_0, \_, \_, \_, \mathbf{R}$	$A_0, A_1, B_0, B_1, \mathbf{R}$	A_0, A_1, B_0, B_1
Monk1	A_1, A_2, A_5	$A_1, \_, \mathbf{A_3}, \mathbf{A_4}, A_5$	$\_, \_, A_5$	A_1, A_2, A_5	A_1, A_2, A_5
Monk2	$A_1, A_2, A_3, A_4, A_5, A_6$	$\_, \_, \_, A_4, A_5, A_6$	$\_, \_, \_, A_4, A_5, A_6$	$A_1, A_2, A_3, A_4, A_5, A_6$	$A_1, A_2, A_3, A_4, A_5, A_6$
Monk3	A_2, A_4, A_5	$A_2, \_, A_5, \mathbf{A_6}$	$A_2, \_, A_5$	A_2, A_4, A_5	$\mathbf{A_1}, A_2, A_4, A_5$

is strongly correlated (75%) with Y. For all the three MONKs data sets, ReliefF can find true relevant features[1] as seen in Table 1. Both FCBF and CFS perform similarly, and FCBF finds more features. FOCUS can handle feature interaction when selecting features. However, as an exhaustive search algorithm, FOCUS finds irrelevant feature A_1 for MONK3 due to 5% noise in the data. In a sense, it overfits the training data. FOCUS is also impractical because finding moderately high-order interactions can be too expensive, as $\sum_{i=1}^{m} \binom{N}{i}$ can be too large, when dimensionality N is large.

In this work, we design and implement an efficient approach to deal with feature interactions. Feature interactions can be implicitly handled by a carefully designed feature evaluation metric and a search strategy with a specially designed data structure, which together take into account interactions among features when performing feature selection.

2 Interaction and Data Consistency

One goal of feature selection is to remove all *irrelevant* features. First, we define feature relevance as in [John *et al.*, 1994]. Let $\mathbf{F}$ be the full set of features, F_i be a feature, $S_i = \mathbf{F} - \{F_i\}$ and P denote the conditional probability of class C given a feature set.

Definition 1 (Feature Relevance) *A feature F_i is relevant iff*

$$\exists S'_i \subseteq S_i, \ such\ that\ P(C|F_i, S'_i) \neq P(C|S'_i).$$

Otherwise, feature F_i is said to be irrelevant.

Definition 1 suggests that a feature can be relevant only when its removal from a feature set will reduce the prediction power. From Definition 1, it can be shown that a feature is relevant due to two reasons: (1) it is strongly correlated with the target concept; or (2) it forms a feature subset with other features and the subset is strongly correlated with the target concept. If a feature is relevant because of the second reason, there exists feature interaction. Feature interaction is characterized by its irreducibility [Jakulin and Bratko, 2004]. A kth feature interaction can be formalized as:

Definition 2 (kth order Feature Interaction) ***F** is a feature subset with k features $F_1, F_2, \ldots, F_k$. Let $\mathscr{C}$ denote a metric that measures the relevance of the class label with a feature or a feature subset. Features $F_1, F_2, \ldots, F_k$ are said to interact with each other iff: for an arbitrary partition $\mathscr{F} = \{\mathscr{F}_1, \mathscr{F}_2, \mathscr{F}_3, \ldots, \mathscr{F}_l\}$ of **F**, where $l \geq 2$ and $\mathscr{F}_i \neq \phi$, we have*

$$\forall i \in [1, l], \quad \mathscr{C}(\boldsymbol{F}) > \mathscr{C}(\mathscr{F}_i)$$

Identifying either relevant features or a kth order feature interaction requires exponential time. Definitions 1 and 2 cannot be directly applied to identify relevant or interacting features when the dimensionality of a data set is high. Many efficient feature selection algorithms identify relevant features based on the evaluation of the correlation between the class and a feature (or a selected feature subset). Representative measures used for evaluating relevance include: distance measures [Kononenko, 1994; Robnik-Sikonja and Kononenko, 2003], information measures [Fleuret, 2004], and consistency measures [Almuallim and Dietterich, 1994], to name a few. Using these measures, feature selection algorithms usually start with an empty set and successively add "good" features to the selected feature subset, the so-called sequential forward selection (SFS) framework. Under this framework, features are deemed relevant mainly based on their individually high correlations with the class, and relevant interacting features of high order may be removed [Hall, 2000; Bell and Wang, 2000], because the irreducible nature of feature interaction cannot be attained by SFS.

Recall that finding high-order feature interaction using relevance (Definitions 1 and 2) entails exhaustive search of all feature subsets. In order to avoid exponential time complexity, we derive a feature scoring metric based on the consistency hypothesis proposed in [Almuallim and Dietterich, 1994] to approximate the relevance measure as in Definition 1. With this metric, we will design a fast filter algorithm that can deal with feature interaction in subset selection.

Let D be a data set of m instances, $D = \{d_1, d_2, \ldots, d_m\}$, and $\mathbf{F}$ be the feature space of D with n features, $\mathbf{F} = \{F_1, F_2, \ldots, F_n\}$, we have the following:

Definition 3 (Inconsistent Instances) *If two instances d_i and d_j in D have the same values except for their class labels, d_i and d_j are inconsistent instances or the two matching instances are inconsistent.*

Definition 4 (Inconsistent-Instances Set) *$\mathscr{D} \subseteq D$, $\mathscr{D}$ is an inconsistent-instances set iff $\forall\ d_i, d_j \in \mathscr{D}$, $i \neq j$, either d_i and d_j are inconsistent or they are duplicate. $\mathscr{D}$ is a **maximal** inconsistent-instances set, iff $\forall d \in D$ and $d \notin \mathscr{D}$, $\mathscr{D} \cup \{d\}$ is not an inconsistent-instances set.*

Definition 5 (Inconsistency Count) *Let $\mathscr{D}$ be an inconsistent-instances set with **k** elements $d_1, d_2, \ldots, d_k$, and $c_1, c_2, \ldots, c_t$ are the class labels of $\mathscr{D}$, we partition $\mathscr{D}$ into t subsets $S_1, S_2, \ldots, S_t$ by the class labels, where $S_i = \{d_j |\ d_j$ has label $c_i\}$. The inconsistency count of $\mathscr{D}$ is:*

$$inconsistencyCount(\mathscr{D}) = k - \max_{1 \leq i \leq t} \|S_i\|$$

Definition 6 (Inconsistency Rate) *Let $\mathscr{D}_1, \mathscr{D}_2, \ldots, \mathscr{D}_p$ denote all maximal inconsistent-instances sets of D, the inconsistency rate (ICR) of D is:*

$$ICR(D) = \frac{\sum_{1 \leq i \leq p} inconsistencyCount(\mathscr{D}_i)}{m}$$

[1] Interestingly, using the full data sets ReliefF missed A_1, A_5 for MONK2, A_1 for MONK3.

Definition 7 (Consistency Contribution) ***or c-contribution*** *Let π denote the projection operator which retrieves a subset of columns from D according to the feature subset, the c-contribution (CC) of feature F_i for $\mathbf{F}$ is defined as:*

$$CC(F_i, \mathbf{F}) = ICR(\pi_{F-\{F_i\}}(D)) - ICR(\pi_F(D))$$

It is easy to verify that the inconsistency rate is monotonic in terms of the number of features, i.e., $\forall S_i$, S_j, $S_i \subseteq S_j \Rightarrow ICR(\pi_{S_i}(D)) \geq ICR(\pi_{S_j}(D))$. Hence, c-contribution of a feature is always a non-negative number with the zero meaning no contribution. C-contribution of a feature F_i is a function of $\mathbf{F} - \{F_i\}$, where $\mathbf{F}$ is the set of features for D. C-contribution of a feature is an indicator about how significantly the elimination of that feature will affect consistency. C-contribution of an irrelevant feature is zero. C-contribution can be considered as an approximation of the metric in Definition 1 by using inconsistency rate as an approximation of P, the conditional probability of class C given a feature set.

The monotonic property of inconsistency rate suggests that the backward elimination search strategy fits c-contribution best in feature selection. That is, one can start with the full feature set and successively eliminating features one at a time based on their c-contributions. Backward elimination allows every feature to be evaluated with the features it may interact with. Hence, backward elimination with c-contribution should find interacting features. However, backward elimination using inconsistency rate or c-contribution has two problems. The first problem is that it is *very costly* as it needs to calculate inconsistency rate for each potentially removable feature. As in the work of FOCUS [Almuallim and Dietterich, 1994], FOCUS relies on exhaustive search. It is impractical to do so when the dimensionality is reasonably large, which separates this work from FOCUS. We will design a specific data structure next to achieve efficient calculation of c-contribution for our algorithm INTERACT. The second problem is that c-contribution measure is sensitive to which feature is selected to compute first, the so-called *the feature order problem*. This is because features evaluated first for their consistency are more likely to be eliminated first.

Solutions to the two problems will enable c-contribution to be used in building an efficient algorithm of backward elimination. We present our solutions and the algorithm next.

3 Eliminating Irrelevant Features

We first present our solutions that form two pillar components for the algorithm INTERACT and then discuss its details.

3.1 Efficient update of c-contribution

C-contribution relies on the calculation of inconsistency rate. With the monotonicity of inconsistency rate, the following two properties are true after a feature f_i is eliminated from a set $\{f_1, .., f_i, ..., f_n\}$ where $i = 1, 2, ..., n$: (I) all *inconsistent instances sets* are still inconsistent; and (II) each maximal *inconsistent instances set* will be either of equal size or bigger. Based on these two properties, we implement a hashing mechanism to efficiently calculate c-contribution: Each instance is inserted into the hash table using its values of those features in S_{list} as the hash key, where S_{list} contains the ranked features that are not yet eliminated (S_{list} initialized with the full set of features). Instances with the same hash key will be insert into the same entry in the hash table. And the information about the labels is recorded. Thus each entry in the hash table corresponds to a maximal inconsistency set of $\pi_{S_{list}}(D)$. Hence, the inconsistency rate of $\pi_{S_{list}}(D)$ can be obtained by scanning the hash table. Property (I) says that in order to generate an entry in the hash table for a new S_{list} (after eliminating a feature), it is not necessary to scan the data again, but only the current hash table. Property (II) suggests that after each iteration of elimination, the number of entries of the hash table for new S_{list} should decrease. Therefore, the hashing data structure allows for efficient update of c-contribution iterative feature elimination.

3.2 Dealing with the feature order problem

We now consider the feature order problem in applying c-contribution. If we can keep removing the current, most irrelevant feature, we will likely retain the most relevant ones in the remaining subset of selected features. Assuming that a set of features can be divided into subset $S1$ including relevant features, and subset $S2$ containing irrelevant ones. By considering to remove features in $S2$ first, features in $S1$ are more likely to remain in the final set of selected features. Therefore, we apply a heuristic to rank individual features using *symmetrical uncertainty* (SU) in an descending order such that the (heuristically) most relevant feature is positioned at the beginning of the list. SU is often used as a fast correlation measure to evaluate the relevance of individual features [Hall, 2000; Yu and Liu, 2003]. This heuristic attempts to increase the chance for a strongly relevant feature to remain in the selected subset. Let $H(X)$ and $H(X, C)$ denote entropy and joint entropy respectively, and $M(X, C) = H(C) + H(X) - H(X, C)$ the mutual information measuring the common information shared between the two variables. SU between the class label C and a feature F_i is:

$$SU(F_i, C) = 2\left[\frac{M(F_i, C)}{H(F_i) + H(C)}\right]$$

This ranking heuristic cannot guarantee that the interacting features are ranked high. For MONK1, for example, $SU(A_1, C) = SU(A_6, C) = 0$. Either one can be evaluated first for its c-contribution. Since $CC(A_1, \mathbf{F}) > CC(A_6, \mathbf{F})$, A_6 is eliminated. We will experimentally examine the ranking effect in Section 4.2.

3.3 INTERACT - An algorithm

The above solutions pave the way for c-contribution to be used in feature selection. We present an algorithm, INTERACT, that searches for interacting features. It is a filter algorithm that employs backward elimination to remove those features with no or low c-contribution. The details are shown in Figure 1: Given a full set with N features and a class attribute C, it finds a feature subset S_{best} for the class concept. The algorithm consists of two major parts. In the first part (lines 1-6), the features are ranked in descending order based on their SU values. In the second part (lines 7-16), features are evaluated one by one starting from the end of the ranked feature list. Function $getLastElement()$ returns the feature

input:
  F: the full feature set with features F_1, $F_2, \ldots, F_N$
  C: the class label
  δ: a predefined threshold
output:
  S_{best}: the best subset

1 S_{list} = NULL
2 **for** i=1 **to** N **do**
3 calculate $SU_{F_i,c}$ for F_i
4 append F_i to S_{list}
5 **end**
6 order S_{list} in descending values of $SU_{i,c}$
7 $F = getLastElement(S_{list})$
8 **repeat**
9 **if** $F <> NULL$ **then**
10 $p = CC(F, S_{list})$ // c-contribution
11 **if** $p \leq \delta$ **then**
12 remove F from S_{list}
13 **end**
14 **end**
15 $F = getNextElement(S_{list}, F)$
16 **until** $F == NULL$
17 $S_{best} = S_{list}$
18 **return** S_{best}

Figure 1: Algorithm INTERACT

in the end of the list, S_{list}. If c-contribution of a feature is less than δ, the feature is removed, otherwise it is selected. Function $getNextElement(S_{list}, F)$ returns the next unchecked feature just preceding F in the ranked feature list (line 15). The algorithm repeats until all features in the list are checked. δ is a predefined threshold ($0 < \delta < 1$). Features with their c-contribution $< \delta$ are considered immaterial and removed. A large δ is associated with a high probability of removing relevant features. Relevance is defined by c-contribution: the higher value of $CC(F, S_{list})$ indicates that F is more relevant. $\delta = 0.0001$ if not otherwise mentioned. The parameter can also be tuned using the standard cross validation.

Time complexity of INTERACT: The first part of the algorithm has a linear time complexity of $O(NM)$, where N is the number of features and M is the number of instances of a given data set. For the second part of the algorithm, the calculation of a feature's c-contribution using a hash table takes also $O(NM)$. For N features, the time complexity of INTERACT is $O(N^2M)$. This is the worst case analysis. Its average time complexity is less because (1) we only use the hash table of current S_{list} in the calculation of c-contribution, and (2) the number of the entries in the hash table decreases after each iteration. If we assume it decreases to a percentage of the initial size, where $0 < a < 1$, then in N iterations, the overall time complexity of INTERACT is $O(NM(1 - a^{N+1})/(1 - a))$ (the proof is omitted). In other words, INTERACT is expected to be comparable with heuristic algorithms such as FCBF [Yu and Liu, 2003].

4 Empirical Study

We empirically evaluate the performance of INTERACT in search of interacting features. For the four synthetic data sets with known feature interaction (Table 1), INTERACT finds only the relevant features ($\delta = 0.05$). Now we evaluate INTERACT using benchmark data sets in comparison with some representative feature selection algorithms with the following aspects: (1) number of selected features, (2) predictive accuracy with feature selection, and (3) run time. We also examine how effective the two solutions (given in Sections 3.1 and 3.2) are through a lesion study by removing one of them at a time from INTERACT.

Table 2: Summary of the benchmark data sets. F: number of features; I: number of instances and C: number of classes.

Data Set	F	I	C	Data Set	F	I	C
lung-cancer	56	32	3	vehicle	18	846	4
zoo	16	101	7	kr-vs-kp	36	3196	2
wine	13	178	3				
soy-large	35	306	19	internet-ads	1558	3278	2
cmc	9	1473	3	45×4026+2C	4026	45	2

4.1 Experiment setup

In our experiments, we choose four representative feature selection algorithms for comparison. They are FCBF [Yu and Liu, 2003], CFS [Hall, 2000], ReliefF [Kononenko, 1994], and FOCUS [Almuallim and Dietterich, 1994]. All are available in the WEKA environment [Witten and Frank, 2005]. INTERACT is implemented in the WEKA's framework. It will be made available upon request. All the experiments were conducted in the WEKA environment.

From 28 data sets, [Jakulin, 2005] identified 10 data sets having feature interactions without selecting interacting features. Here we focus on our discussion on the 10 data sets[2]. The datasets are from the UCI ML Repository [Blake and Merz, 1998]. We also include another two datasets in the experiment: the 'internet-ads' data from the UCI ML Repository, and the '45×4026+2C' data from [Alizadeh and et al., 2000]. The information about the 9 data sets (without 3 MONKS data sets) is summarized in Table 2. For each data set, we run all 5 feature selection algorithms and obtain selected feature subsets of each algorithm. For data sets containing features with continuous values, if needed, we apply the MDL discretization method (available in WEKA). We remove all index features if any. In order to evaluate whether the selected features are indeed good, we apply two effective classification algorithms C4.5 and a linear Support Vector Machine (SVM)[3] (both available from Weka) before and after feature selection and obtain prediction accuracy by 10-fold cross-validation (CV). Although C4.5 itself evaluates features (one at a time), its performance is sensitive to the given sets of features (e.g., its accuracy rates on MONK1 are 88.88% and 75.69% for (A_1, A_2, and A_5) and for all 6 features, respectively). Because of FOCUS's exponential time complexity, we only provide those results of FOCUS obtained in 3 hours of dedicated use of the PC. INTERACT, FCBF, CFS and Re-

[2]In which 3 data sets are MONKs data. We consider them synthetic data and discussed them earlier separately.

[3]Since Naive Bayes Classifier (NBC) assumes conditional independence between features [Irina Rish, 2001], selecting interacting features or not has limited impact on it. Our experimental results of NBC conform to the analysis and will be presented elsewhere due to the space limit.

Table 3: Number of selected features for each algorithm (IN: INTERACT, FC: FCBF, RE: ReliefF, FO: FOCUS, FS: Full Set). NA denotes not available.

Data Set	IN	FC	CFS	RE	FO	FS
lung-cancer	6	6	11	22	4	56
zoo	5	8	9	14	5	16
wine	5	10	8	11	5	13
soy-large	13	13	21	32	NA	35
cmc	9	2	3	5	9	9
vehicle	18	4	11	7	18	18
kr-vs-kp	29	7	3	8	NA	36
internet-ads	49	38	11	2	NA	1558
45×4026+2C	3	64	42	15	NA	4026
average	15.22	16.89	13.22	12.89	8.20	640.78

liefF all complete their runs in seconds. This is consistent with our understanding and expectation of these algorithms.

4.2 Results and discussion

Number of selected features. Table 3 presents the numbers of features selected by the five algorithms. All algorithms significantly reduced the number of features in many cases (e.g., from 56 to as few as 6). The average numbers of selected features for the 9 data sets are 15.22 (INTERACT), 16.89 (FCBF), 13.22 (CFS), 12.89 (ReliefF), and 640.78 (Full Set). For four data sets (indicated by NA in the table), FOCUS did not finish after 3 hours. For the 4 synthetic data sets with known relevant features (Table 1), INTERACT selected only the relevant ones. Next we examine the effect of this reduction on accuracy.

Predictive accuracy before and after feature selection. For the 9 data sets with known interactions, we obtained predictive accuracy rates by 10-fold cross validation using C4.5 and SVM. The results are shown in the two sub-tables of Table 5. For both classifiers, the reduction of features by INTERACT obtains results comparable with *using all features*: average accuracy 83.58% (INTERACT) vs. 79.74% (Full Set) for C4.5, and 82.88% (INTERACT) vs. 81.04% (Full Set) for SVM. Comparing INTERACT with other feature selection algorithms, INTERACT performs consistently better for C4.5 with better average accuracy. For SVM, INTERACT is comparable with other feature selection algorithms. One exception is the 'soy-large' data for the result of SVM. We notice that the data set has 35 features, 306 instances, and 19 classes (Table 2); INTERACT identifies 13 features (Table 3) - the smallest number of selected features (FCBF also selected 13 features). We surmise that it may be too easy for the inconsistency measure to be satisfied with a small feature subset when each class has a small number of instances. In sum, for both classifiers, INTERACT can help achieve better or similar accuracy, and hence, INTERACT is *effective in search of interacting features*.

The effect of feature ranking. INTERACT ranks features before backward elimination of features begins. As a part of the lesion study, we remove the ranking component from INTERACT to form a version INTERACT$_{\setminus R}$. We summarize the results here due to the space limit: INTERACT is always better than or equivalent to INTERACT$_{\setminus R}$; the average 10-fold CV accuracy for C4.5 and SVM for (INTERACT, INTERACT$_{\setminus R}$) are (83.58,78.69) and (82.88 78.54), respectively. Noticing that INTERACT ranks features using SU and

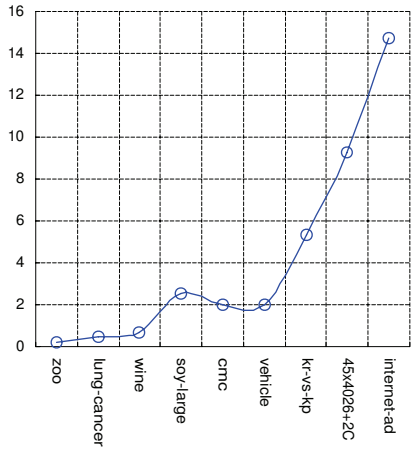

Figure 2: $T_{INTERACT_{\setminus D}}/T_{INTERACT}$

Table 4: Run time (in second) for each algorithm.

Data Set	IN	FC	CFS	Re	FO
lung-cancer	0.09	0.02	0.05	0.02	2.31
zoo	0.08	0.01	0.01	0.02	0.81
wine	0.09	0.02	0.02	0.02	0.33
soy-large	0.14	0.06	0.05	0.09	NA
cmc	0.14	0.02	0.03	0.11	2.81
vehicle	0.20	0.05	0.06	0.11	976.58
kr-vs-kp	1.31	0.27	0.31	0.61	NA
internet-ads	150.86	60.484	54.344	31.359	NA
45×4026+2C	20.30	1.02	323.25	1.50	NA
average	19.25	6.88	42.01	3.76	196.57

FCBF employs SU, we observe in Table 5 that FCBF does not perform as well as INTERACT in selecting interacting features. The results suggest that the combination of SU and c-contribution helps INTERACT to achieve its design goal. Clearly, using SU is a heuristic. There could be other alternatives to rank the features. Studying the effects of different ranking algorithms is one line of our on-going research.

The effect of the data structure. We devised a hashing data structure to speed up the time consuming calculation of c-contribution during feature selection. Here we examine how effective the data structure is by comparing the run time of INTERACT with that of INTERACT$_{\setminus D}$ which does not employ the hashing data structure. Since data size is a determining factor for run time, we first reorganize the 9 data sets according to their sizes (approximated by $N * M$ - number of features multiplied by number of instances without considering feature types such as nominal or continuous). Figure 2 shows the ratios using time of INTERACT as the base: T$_{INTERACT_{\setminus D}}$ divided by T$_{INTERACT}$. It shows that the run time difference between INTERACT and INTERACT$_{\setminus D}$ is more pronounced for larger data sets.

Run time comparison. Table 4 records the run time for each feature selection algorithm. Except for FOCUS, all algorithms finished their runs within the given time. The algorithms are ordered as ReliefF, FCBF, INTERACT, CFS and FOCUS: ReliefF is the fastest, and FOCUS is the slowest if it finishes the run within 3 hours.

Table 5: Accuracy of C4.5 and SVM on selected features: 'Acc' denotes 10-fold CV accuracy(%) and p-Val obtained from a two-tailed t-Test. The symbols "+" and "-" respectively identify statistically significant (at 0.05 level) if INTERACT wins over or loses to the compared algorithm. NA denotes the result is not available.

Data Set	INTERACT	FCBF		CFS		ReliefF		FOCUS		WholeDS	
	Acc	Acc	p-val	Acc	p-val	Acc	p-val	Acc	p-val	Acc	p-val
C4.5											
lung-cancer	75.00	50	0.01+	43.75	0.00+	65.63	0.06	37.5	0.01+	46.88	0.02+
zoo	91.09	92.08	0.72	91.09	1.00	91.09	1.00	91.09	1.00	92.08	0.78
wine	96.63	93.82	0.05+	93.82	0.05+	83.15	0.00+	93.26	0.05+	93.82	0.05+
soy-large	82.03	84.31	0.25	84.31	0.17	84.31	0.31	NA	NA	85.95	0.11
cmc	52.14	51.93	0.89	54.11	0.08	50.17	0.10	52.27	0.17	52.27	0.17
vehicle	73.05	57.68	0.00+	69.62	0.00+	67.73	0.00+	73.05	1.00	73.05	1.00
kr-vs-kp	99.19	94.02	0.00+	90.43	0.00+	90.43	0.00+	NA	NA	99.44	0.31
internet-ads	96.46	95.85	0.08	95.76	0.03+	91.52	0.00+	NA	NA	96.4	0.79
45×4026+2C	86.67	80.00	0.23	77.78	0.14	86.67	1.00	NA	NA	77.78	0.19
Average	83.58	77.74		77.85		78.97		69.43		79.74	
Win/Loss		4/0		5/0		4/0		2/0		2/0	
SVM											
lung-cancer	62.50	53.13	0.32	62.50	1.00	62.50	1.00	31.25	0.01+	37.50	0.03+
zoo	93.07	92.08	0.72	96.04	0.19	96.04	0.19	93.07	1.00	96.04	0.19
wine	96.63	98.31	0.08	96.63	1.00	84.83	0.01+	89.89	0.04+	97.75	0.17
soy-large	83.33	87.91	0.01-	92.16	0.00-	92.16	0.00-	NA	NA	91.18	0.00-
cmc	48.74	44.87	0.05+	49.97	0.37	44.33	0.02+	48.68	0.34	48.68	0.34
vehicle	73.88	42.08	0.00+	57.21	0.00+	50.35	0.00+	73.88	1.00	73.88	1.00
kr-vs-kp	95.90	94.06	0.00+	90.43	0.00+	90.43	0.00+	NA	NA	95.93	0.94
internet-ads	96.34	95.97	0.08	95.94	0.02+	90.57	0.00+	NA	NA	97.28	0.01-
45×4026+2C	95.56	100.00	0.12	100.00	0.12	95.56	1.00	NA	NA	91.11	0.39
Average	82.88	78.71		82.32		78.53		67.35		81.04	
Win/Loss		3/1		3/1		5/1		2/0		1/2	

5 Conclusions

We recognize the need to study feature interaction in subset selection using some synthetic data sets, and propose to search for interacting features employing c-contribution to measure feature relevance. We investigated the key issues hindering the use of consistency measures and developed INTERACT that handles feature interaction and efficiently selects relevant features. It ranks features to overcome the feature order problem. INTERACT implements a special hashing data structure to avoid repeated scanning of a data set by taking advantage of the intrinsic properties of the c-contribution measure, which results in a significant speed-up of the algorithm. The efficiency and effectiveness of INTERACT are demonstrated through theoretical analysis as well as extensive experiments comparing with representative feature selection algorithms using the synthetic and real-world data sets with *known feature interactions*. Experimental results show that INTERACT can effectively reduce the number of features, and maintain or improve predictive accuracy in dealing with interacting features. This work presents promising initial efforts on searching efficiently for interacting features.

References

[Alizadeh and et al., 2000] A. Alizadeh and et al. Distinct types of diffuse large B-cell lymphoma identified by gene expression profiling. *Nature*, 403:503–511, 2000.

[Almuallim and Dietterich, 1994] H. Almuallim and T. G. Dietterich. Learning boolean concepts in the presence of many irrelevant features. *Artificial Intelligence*, 69(1-2):279–305, 1994.

[Bell and Wang, 2000] D. A. Bell and H. Wang. A formalism for relevance and its application in feature subset selection. *Machine Learning*, 41(2):175–195, 2000.

[Blake and Merz, 1998] C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998.

[Blum and Langley, 1997] A. L. Blum and P. Langley. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 97:245–271, 1997.

[Dash and Liu, 1997] M. Dash and H. Liu. Feature selection for classification. *Intelligent Data Analysis: An International Journal*, 1(3):131–156, 1997.

[Fleuret, 2004] Franois Fleuret. Fast binary feature selection with conditional mutual information. *J. Mach. Learn. Res.*, 5:1531–1555, 2004.

[Gilad-Bachrach *et al.*, 2004] Ran Gilad-Bachrach, Amir Navot, and Naftali Tishby. Margin based feature selection - theory and algorithms. In *ICML* 2004.

[Guyon and Elisseeff, 2003] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *J. Mach. Learn. Res.*, 3:1157–1182, 2003.

[Hall, 2000] M. A. Hall. Correlation-based feature selection for discrete and numeric class machine learning. In *ICML*, 2000.

[Irina Rish, 2001] Jayram Thathachar Irina Rish, Joseph L. Hellerstein. An analysis of data characteristics that affect naive bayes performance. Technical report, IBM T.J. Watson, 2001.

[Jakulin and Bratko, 2003] Aleks Jakulin and Ivan Bratko. Analyzing attribute dependencies. In *PKDD*, 2003.

[Jakulin and Bratko, 2004] Aleks Jakulin and Ivan Bratko. Testing the significance of attribute interactions. In *ICML*, 2004.

[Jakulin, 2005] Aleks Jakulin. *Machine Learning Based on Attribute Interactions*. PhD thesis, University of Ljubljana, 2005.

[John *et al.*, 1994] G. H. John, R. Kohavi, and K. Pfleger. Irrelevant feature and the subset selection problem. In *ICML*, 1994.

[Kononenko, 1994] I. Kononenko. Estimating attributes : Analysis and extension of RELIEF. In *ECML*, 1994.

[Robnik-Sikonja and Kononenko, 2003] M. Robnik-Sikonja and I. Kononenko. Theoretical and empirical analysis of Relief and ReliefF. *Machine Learning*, 53:23–69, 2003.

[Witten and Frank, 2005] I. H. Witten and E. Frank. *Data Mining - Pracitcal Machine Learning Tools and Techniques with JAVA Implementations*. Morgan Kaufmann Publishers, 2 edition, 2005.

[Yu and Liu, 2003] L. Yu and H. Liu. Feature selection for high-dimensional data: a fast correlation-based filter solution. In *ICML*, 2003.