information

Article

sCwc/sLcc: Highly Scalable Feature Selection Algorithms

Kilho Shin [1,2,*,†], Tetsuji Kuboyama [3,†], Takako Hashimoto [4] and Dave Shepard [5]

1 Graduate School of Applied Informatics, University of Hyogo, Kobe 651-2197, Japan
2 Information Networking Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA
3 Computer Centre, Gakushuin University, Tokyo 171-0031, Japan; ori-mdpi2017@tk.cc.gakushuin.ac.jp
4 Institude of Economic Research, Chiba University of Commerce, Chiba 272-8512, Japan; takako@cuc.ac.jp
5 Center for Digital Humanities, University of California Las Angeles; Los Angeles, CA 90095, USA; shepard.david@gmail.com
* Correspondence: yshin@ai.u-hyogo.ac.jp or kilhoshin314@gmail.com or kilhos@andrew.cmu.edu; Tel.: +81-78-303-1901
† These authors contributed equally to this work.

Received: 31 October 2017; Accepted: 2 December 2017; Published: 6 December 2017

Abstract: Feature selection is a useful tool for identifying which features, or attributes, of a dataset cause or explain the phenomena that the dataset describes, and improving the efficiency and accuracy of learning algorithms for discovering such phenomena. Consequently, feature selection has been studied intensively in machine learning research. However, while feature selection algorithms that exhibit excellent accuracy have been developed, they are seldom used for analysis of high-dimensional data because high-dimensional data usually include too many instances and features, which make traditional feature selection algorithms inefficient. To eliminate this limitation, we tried to improve the run-time performance of two of the most accurate feature selection algorithms known in the literature. The result is two accurate and fast algorithms, namely sCwc and sLcc. Multiple experiments with real social media datasets have demonstrated that our algorithms improve the performance of their original algorithms remarkably. For example, we have two datasets, one with 15,568 instances and 15,741 features, and another with 200,569 instances and 99,672 features. sCwc performed feature selection on these datasets in 1.4 seconds and in 405 seconds, respectively. In addition, sLcc has turned out to be as fast as sCwc on average. This is a remarkable improvement because it is estimated that the original algorithms would need several hours to dozens of days to process the same datasets. In addition, we introduce a fast implementation of our algorithms: sCwc does not require any adjusting parameter, while sLcc requires a threshold parameter, which we can use to control the number of features that the algorithm selects.

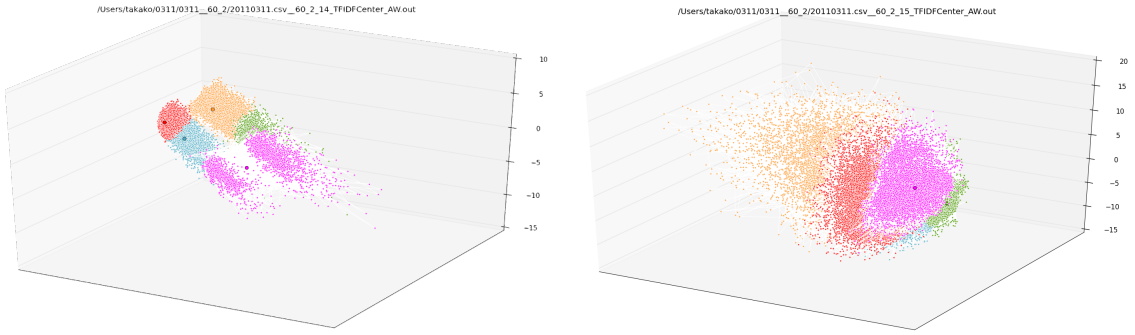
Keywords: feature selection; consistency; high-dimensional data; scalability

1. Introduction

Accurate and fast feature selection is a useful tool of data analysis. In particular, feature selection on categorical data is important in real world applications. Features, or attributes, and, in particular, features to specify class labels, which represent the phenomena to explain, and/or the targets to predict are often categorical. In this paper, we propose two new feature selection algorithms that are as accurate as, and drastically faster than, any other methods represented in the literature. In fact, our algorithms are the first accurate feature selection algorithms that scale well to big data.

The importance of feature selection can be demonstrated with an example. Figure 1 depicts the result of clustering tweets posted to Twitter during two different one-hour windows on the day of the Great East Japan Earthquake, which hit Japan at 2:46 p.m. on 11 March 2011 and inflicted catastrophic

damage. Figure 1a plots 97,977 authors who posted 351,491 tweets in total between 2:00 p.m. and 3:00 p.m. on the day of the quake (the quake occurred in the midst of this period of time), while Figure 1b plots 161,853 authors who posted 978,155 tweets between 3:00 p.m. and 4:00 p.m. To plot, we used word-count-based distances between authors and a multidimensional scaling algorithm. Moreover, we grouped the authors into different groups using the *k*-means clustering algorithm based on the same distances. Dot colors visualize that clustering. We observe a big change in clustering between the hour during which the quake occurred, and the hour after the quake.



(**a**) (**b**)

Figure 1. Clustering of twitter data. (**a**) tweets between 2:00 p.m. and 3:00 p.m. of 11 March. The quake hit Japan at 2:46 p.m., and 97,977 authors who posted 351,491 tweets in total are plotted; (**b**) tweets between 3:00 p.m. and 4:00 p.m. of 11 March. Furthermore, 161,853 authors who posted 978,155 tweets in total are plotted.

Two questions naturally arise: first, what do the clusters mean? Second, what causes the change from Figure 1a to Figure 1b? Answering these questions requires a method for selecting words that best characterize each cluster; in other words, a method for feature selection.

To illustrate, we construct two datasets, one for the timeframe represented in Figure 1a and one for the time-frame represented in Figure 1b, called dataset A and dataset B, respectively. Each dataset consists of a word count vector for each author that reflects all words in all of their tweets. Dataset A has 73,543 unique words, and dataset B has 71,345 unique words, so datasets A and B have 73,543 and 71,345 features, respectively. In addition, each author was given a class label reflecting the category he or she was assigned to from the *k*-means clustering process.

It was our goal to select a relatively small number of features (words) that were relevant to class labels. We say that a set of features is relevant to class labels, if the values of the features uniquely determine class labels with high likelihood. Table 1 depicts an example of a dataset for explanation. $F_1, \ldots, F_5$ are features, and the symbol C denotes a variable to represent class labels. The feature F_5, for example, is totally *irrelevant* to class labels. In fact, we have four instances with $F_5 = 0$, and a half of them have the class label 0, while the other half have the class label 1. The same holds true for the case of $F_5 = 1$. Therefore, F_5 cannot explain class labels at all and is useless to predict class labels. In fact, predicting class labels based on F_5 has the same success probability as guessing them by tossing a fair coin (the Bayesian risk of F_5 to C is 0.5, which is the theoretical worst). On the other hand, F_1 is more relevant than F_5 because the values of F_1 explain 75% of the class labels, and, in other words, the prediction based on F_1 will be right with a probability of 0.75 (that is, the Bayesian risk is 0.25).

The relevance of individual features can be estimated using statistical measures such as mutual information, symmetrical uncertainty, Bayesian risk and Matthew's correlation coefficients. For example, at the bottom row of Table 1, the mutual information score $I(F_i, C)$ of each feature F_i to class labels is described. We see that F_1 is more relevant than F_5, since $I(F_1, C) > I(F_5, C)$.

To our knowledge, the most common method deployed in big data analysis to select features that characterize class labels is to select features that show higher relevance in some statistical measure. For example, in the example of Table 1, F_1 and F_2 will be selected to explain class labels.

However, when we look into the dataset of Table 1 more closely, we understand that F_1 and F_2 cannot determine class labels uniquely. In fact, we have two instances with $F_1 = F_2 = 1$, whose class labels are 0 and 1. On the other hand, F_4 and F_5 as a combination uniquely determine the class labels by the formula of $C = F_1 \oplus F_2$, where $\oplus$ denotes the addition modulo two. Therefore, the traditional method based on relevance scores of individual features misses the right answer.

Table 1. An example dataset.

F_1	F_2	F_3	F_4	F_5	C
1	0	1	1	1	0
1	1	0	0	0	0
0	0	0	1	1	0
1	0	1	0	0	0
1	1	1	1	0	1
0	1	0	1	0	1
0	1	0	0	1	1
0	0	0	0	1	1
0.189	0.189	0.049	0.000	0.000	$I(F_i, C)$

This problem is well known as the problem of feature interaction in feature selection research. Feature selection has been intensively studied in machine learning research. The literature describes a class of feature selection algorithms that can solve this problem, referred to as consistency-based feature selection (for example, [1–5]).

Figure 2 shows the result of feature selection using one of the consistency-based algorithms, namely, CWC (Combination of Weakest Components) [5]. The dataset used was one generated in the aforementioned way from the tweets of the day when the quake hit Japan and includes 161,425 instances (authors) and 38,822 features (words). The figure shows not only the 40 words selected but also their scores and ranks measured by the symmetrical uncertainty (in parentheses).

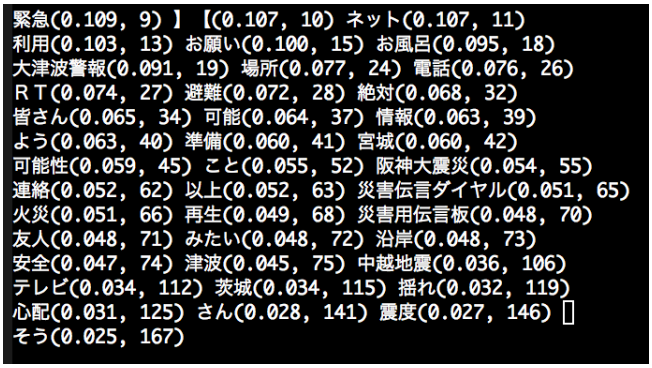


Figure 2. An example result of feature selection by CWC: Word (Score, Rank). Scores and ranks are measured by the symmetrical uncertainty. The Japanese words in this figure are translated as "emergency" (9), "networks" (11), "utilize" (13", "favor" (15), "bath" (18), "great tsunami warning" (19), "place" (24), "phone" (26), "evacuation" (28), "absolute" (32), "all" (34), "possible" (37), "information" (39), "like" (40), "preparation" (41), "Miyagi" (42), "possibility" (45), "thing" (52), "Hanshin Great Quake" (55), "notification" (62), "over" (63), "disaster mail telephone" (65), "friend" (71), "as if" (72), "coast" (73), "safety" (74), "tsunami" (75), "Chu-Etsu Quake" (106), "television" (112), "Ibaraki" (115), "shock of earthquake" (119), "worry" (125), "Mr.", "Mrs." or "Ms." (141), "earthquake intensity" (146) and "seem" (167). The numbers within parentheses indicate the ranks of the words.

This result contains two interesting findings. First, the word ranked 141th is translated as "Mr.", "Mrs.", or "Ms.", which is a polite form of address in Japanese. This form of address is common in

Japanese writing, so it seems odd that the word would identify a cluster of authors well. In fact, the relevance of the word is as low as 0.028. However, if we understand the nature of CWC, we can guess that the word must interact with other features to determine which cluster the author falls inside. In fact, it turns out that the word interacts with the 125th-ranked word, "worry". Hence, we realize that a portion of those tweets must have been asking about the safety of someone who was not an author's family member—in other words, someone who the author would have addressed with the polite form of their name.

The second interesting finding is that the words with the highest relevance to class labels have not been selected. For example, the word that means "quake" was ranked at the top but not selected. This is because the word was likely to be used in the tweets with other selected words such as words that translated to "tsunami alert" (ranked 19th), "the Hanshin quake" (55th), "fire" (66th), "tsunami" (75th) and "the Chu-Etsu quake" (106th), so that CWC judged the word "quake" to be redundant once the co-occurring words had been selected. Our interpretation is that these co-occurring words represent the *contexts* in which the word "quake" was used, and selecting these words gave us more information than selecting "quake", which is too general in this case.

Thus, the consistency-based algorithms do not simply select features with higher relevance; instead, they give us knowledge that we cannot obtain from selection based on the relevance of individual features. In spite of these advantages, however, consistency-based feature selection is seldom used in big data analysis. Consistency-based algorithms require heavy computation and the amount of computation increases as the size of data increases so greatly as to make application to large data sets unfeasible.

This paper's contribution is to improve the run-time performance of two consistency-based algorithms that are known the most accurate, namely CWC and LCC (Linear Consistency Constrained feature selection) [4]. We introduce two algorithms that perform well on big data: SCWC and SLCC. They always select the same features as CWC and LCC, respectively, and, therefore, perform with the same accuracy. SLCC accepts a threshold parameter to control the number of features to select and has turned out to be as fast as SCWC on average in our experiments.

2. Feature Selection on Categorical Data in Machine Learning Research

In this section, we give a brief review of feature selection research focusing on categorical data. The literature describes three broad approaches: *filter*, *wrapper* and *embedded*. Filter approaches aim to select features based on the intrinsic properties of datasets leveraging statistics and information theory, while wrapper and embedded approaches aim to optimize the performance of particular classification algorithms. We are interested in the filter approach in this paper. We first introduce a legacy feature selection framework and identify two problems in that framework. Then, we introduce the consistency-based approach to solve these problems. For convenience, we will describe a feature or a feature set that is relevant to class labels simply as *relevant*.

2.1. The Legacy Framework: Sum of Relevance (SR)

In the legacy and fundamental framework of feature selection, which underlies most of the known practical feature selection algorithms, we use *sum-of-relevance* (SR) functions to evaluate collective relevance of feature sets.

Sum of relevance
Computing the sum of relevance of individual features is an efficient method for estimating the collective relevance.

For example, let $I(F, C)$ denote the mutual information of an individual feature F and the class variable C. To be specific, $I(F, C)$ is defined by

$$I(F,C) = \sum_{x,y} \Pr[F = x, C = y] \log \frac{\Pr[F = x, C = y]}{\Pr[F = x] \Pr[C = y]}.$$

The values of x and y are selected from the sample spaces of F and C. It is well known that, the larger $I(F,C)$ is, the more tightly F and C correlate with each other. If we do not know the population distribution Pr, we use the empirical distribution derived from a dataset. The sum-of-relevance for a feature set $\{F_1, \ldots, F_n\}$ based on I is determined by

$$\mathrm{SR}(F_1, \ldots, F_n) = \sum_{i=1}^{n} I(F_i, C),$$

and estimates the collective relevance of $\{F_1, \ldots, F_n\}$.

The principle of *SR-based feature selection* is to find a good balance to the trade-off between the SR value of and the number of features to select. This can be achieved efficiently by computing the relevance of individual features and sorting the features with respect to the computed relevance scores. For example, Table 1 shows a dataset, and we see the relevance of each feature measured by the mutual information at the bottom row. Since $I(F_1, C) = I(F_2, C) \approx 0.13$, $I(F_3, C) \approx 0.03$ and $I(F_4, C) = I(F_5, C) = 0$ hold, if the requirement is to select two features that maximize the SR value, the best choice is definitely F_1 and F_2. If the requirement is to select the smallest feature set whose SR value is no smaller than 0.25, the answer should be F_1 and F_2 as well. As a substitute for mutual information, we can use the Bayesian risk, the symmetrical uncertainty and Matthews correlation coefficients, for example.

RELIEF-F [6] is a well-known example of a feature selection algorithm that relies only on SR functions. For the underlying relevance function, RELIEF-F uses a distance-based randomized function. Since computing this distance-based relevance function requires relatively heavy computation, RELIEF-F is not very fast, but, in general, the simple SR-based feature selection scales and can be applied to high-dimensional data.

2.2. The Problem of Redundancy

The simple SR-based feature selection has, however, two important problems that will harm the collective relevance of selected features. One of them is the problem caused by *internal correlation*, which is also known as the problem of *redundancy*. The problem is described as follows.

Problem of redundancy
Feature selection by SR may select features that are highly mutually correlated, and such high internal correlation definitely decreases the collective relevance of the features.

The dataset of Table 2 is obtained by adding the feature F_6 to the dataset of Table 1. Eventually, F_6 is a copy of F_1, and, hence, $I(F_6, C) = I(F_1, C) \approx 1.3$ holds. To select two features that maximize the SR value, we have three answer candidates this time, that is, $\{F_1, F_2\}$, $\{F_1, F_6\}$ and $\{F_2, F_6\}$. Among the candidates, $\{F_1, F_6\}$ is clearly a wrong answer, since its joint relevance has no gain over the individual relevance of F_1 and F_6.

This thought experiment inspires us to pay attention to the internal correlation among features. If the internal correlation among features is greater, the features include more redundancy when they determine classes. For example, if we use mutual information to evaluate internal correlation, the internal correlation of $\{F_1, F_2\}$ is computed to be $I(F_1, F_2) = 0$, that is, F_1 and F_2 are independent of each other. On the other hand, the internal correlation of $\{F_1, F_6\}$ is $I(F_1, F_6) = H(F_1) \approx 0.68$. Therefore, the set of $\{F_1, F_6\}$ includes more redundancy than $\{F_1, F_2\}$, and, hence, we should select $\{F_1, F_2\}$ rather than $\{F_1, F_6\}$. The principle of *minimizing redundancy* (MR) is to design feature selection algorithms so that they avoid selecting features that have high internal correlation.

The algorithm of mRMR (Minimum Redundancy and Maximum Relevance) [7] is a well-known greedy forward selection algorithm that maximizes the sum of relevance (SR) with respect to the mutual information and minimizes the internal redundancy determined by

$$\mathrm{IC}(F_1, \ldots, F_n) = \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} I(F_i, F_j).$$

FCBF (Fast Correlation-Based Filter) [8] and CFS (Correlation-based Feature Selection) [9] are also known to be based on the principle of minimizing redundancy.

Although the principle of minimizing redundancy definitely improves the actual collective relevance of features to select, it cannot solve the other problem of the SR framework, which we state next.

Table 2. An example dataset.

F_1	F_2	F_3	F_4	F_5	F_6	C
1	0	1	1	1	**1**	0
1	1	0	0	0	**1**	0
0	0	0	1	1	**0**	0
1	0	1	0	0	**1**	0
1	1	1	1	0	**1**	1
0	1	0	1	0	**0**	1
0	1	0	0	1	**0**	1
0	0	0	0	1	**0**	1
0.189	0.189	0.049	0.000	0.000	**0.189**	$I(F_i, C)$

2.3. The Problem of Feature Interaction

We start with describing the problem.

Problem of feature interaction
Feature selection by SR may miss features if their individual relevance is low but they show high collective relevance by interacting one another.

For the datasets of Tables 1 and 2, F_4 and F_5 determine the class C by the formula of $C = F_4 \oplus F_5$, where $\oplus$ denotes the addition modulo two, and, hence, their collective relevance is the highest. Nevertheless, the sum of relevance for $\{F_4, F_5\}$ is zero, and, hence, the feature selection algorithms that we saw in Sections 2.1 and 2.2 have no chance to select $\{F_4, F_5\}$.

This problem is explained by *interaction among features*: when more than one features that individually show only low relevance exhibit high collective relevance, we say that the features *interact* with each other. As shown in the example above, neither the SR principle nor the MR principle can incorporate feature interaction into the results of feature selection.

The literature provides two approaches to solve this problem: *rule-based*, and *consistency-based*. We will define these approaches here.

FRFS (FOIL Rule based Feature subset Selection) [10] is a characteristic example of a rule-based feature selection algorithm. The algorithm first extracts events of feature interaction from a dataset as frequent rules. Each rule is in the form of $(f_1, \ldots, f_k) \Rightarrow c$ such that the antecedent $f_1, \ldots, f_k$ are feature values and the consequent c is a class label. FRFS can perform the rule extraction efficiently by leveraging First Order Inductive Learner (FOIL) [11]. However, it is still too slow to apply to big data analysis.

2.4. Consistency-Based Feature Selection

The consistency-based approach solves the problem of feature interaction by leveraging *consistency measures*. A consistency measure is a function that takes sets of features as input rather than individual features. Furthermore, a consistency measure is a function that represents collective *irrelevance* of the feature set input, and, hence, the smaller a value of a consistency measure is, the more relevant the input feature set is.

Moreover, a consistency measure is required to have the *determinacy* property: its measurement is zero, if, and only if, the input feature set uniquely determines classes.

Definition 1. *A feature set of a dataset is* consistent, *if, and only if, it uniquely determines classes, that is, any two instances of the dataset that are identical with respective to the values of the features of the feature set have the identical class label as well.*

Hence, a consistency measure function returns the value zero, if, and only if, its input is a consistent feature set. An important example of the consistency measure is the Bayesian risk, also known as the *inconsistency rate* [2]:

$$\mathrm{Br}(F_1, \ldots, F_n) = 1 - \sum_{x_1, \ldots, x_n} \max_y \Pr[F_1 = x_1, \ldots, F_n = x_n, C = y].$$

The variable x_i moves in the sample space of F_i, while the variable y moves in the sample space of C. It is evident that the Bayesian risk is non-negative, and determinacy follows from

$$\sum_{x_1, \ldots, x_n} \max_y \Pr[F_1 = x_1, \ldots, F_n = x_n, C = y] \leq \sum_{x_1, \ldots, x_n} \Pr[F_1 = x_1, \ldots, F_n = x_n] = 1.$$

Another important example of the consistency measure is the *binary consistency measure*, defined as follows:

$$\mathrm{Bn}(F_1, \ldots, F_n) = \begin{cases} 0, & \text{if } \{F_1, \ldots, F_n\} \text{ is consistent;} \\ 1, & \text{otherwise.} \end{cases}$$

FOCUS [1], the first consistency-based algorithms in the literature, performs an exhaustive search to find the smallest feature set $\{F_1, \ldots, F_n\}$ with $\mathrm{Bn}(F_1, \ldots, F_n) = 0$.

Apparently, FOCUS cannot be practically fast. In general, consistency-based feature selection has problems in time-efficiency because of the broadness of the search space. In fact, the search space should be the power set of the entire set of features, and its size is an exponential function of the number of features.

Problem of consistency measures
When N features describe a dataset, the number of the possible input to a consistency measure is as large as 2^N.

The *monotonicity property* of consistency measures helps to solve this problem. The Bayesian risk, for example, has this property: if $\mathcal{F} \subseteq \mathcal{G}$, $\mathrm{Br}(\mathcal{F}) \geq \mathrm{Br}(\mathcal{G})$ holds, where $\mathcal{F}$ and $\mathcal{G}$ are feature subsets of a dataset. Almost all of the known consistency measures such as the binary consistency measure and the conditional entropy $H(C \mid \mathcal{F}) = H(C) - I(\mathcal{F}, C)$ have this property as well. In [5], the consistency measure is formally defined as a non-negative function that has the determinacy and monotonicity properties.

Although some of the algorithms in the literature such as ABB (Automatic Branch and Bound) [2] took advantage of the monotonicity property to narrow their search space, the real breakthrough was yielded by Zhao and Liu in their algorithm INTERACT [3]. INTERACT uses the combination of the sum-of-relevance function based on the symmetrical uncertainty and the Bayesian risk.

The symmetrical uncertainty is a harmonic mean of the ratios of $I(F,C)/H(F)$ and $I(F,C)/H(C)$ and hence turns out to be

$$\mathrm{SU}(F;C) = \frac{2I(F,C)}{H(F)+H(C)}.$$

The basic idea of INTERACT is to narrow down the search space boldly and to take SR values of the symmetrical uncertainty into account to cover the caused decrease of relevance. Although the search space of INTERACT is very narrow, the combination of the SR function and the consistency measure keeps the accuracy performance good. LCC [4] improves INTERACT and can exhibit better accuracy. Although INTERACT and LCC are much faster than previous consistency-based algorithms described in the literature, they are not fast enough to apply to large datasets with thousands of instances and features.

CWC [5] is a further improvement and replaces the Bayesian risk with the binary consistency measure, which can be computed faster. CWC is reported to be about 50 times faster than INTERACT and LCC on average. In fact, CWC performs feature selection for a dataset with 800 instances and 100,000 features in 544 s, while LCC does it in 13,906 s. Although the improvement was remarkable, CWC is not fast enough to apply to big data analysis.

2.5. Summary

Figure 3 summarizes the progress of feature selection in the literature. The legacy framework of sum-of-relevance (SR) has the problems of redundancy and feature interaction. The principle of minimizing redundancy (MR) in combination with SR solves the problem of redundancy and provides practical algorithms such as mRMR. Furthermore, using consistency measures (CM) solves the problem of feature interaction, but is time-consuming because complete (exhaustive) search (CS) is necessary. On the other hand, the combination of SR and CM allows linear search (LS) and improves the low time-efficiency of the consistency-based feature selection dramatically. In particular, CWC is the fastest and the most accurate consistency-based algorithm and is comparable with FRFS, which is rule-based. Nevertheless, CWC or FRFS does not scale well for big data analysis.

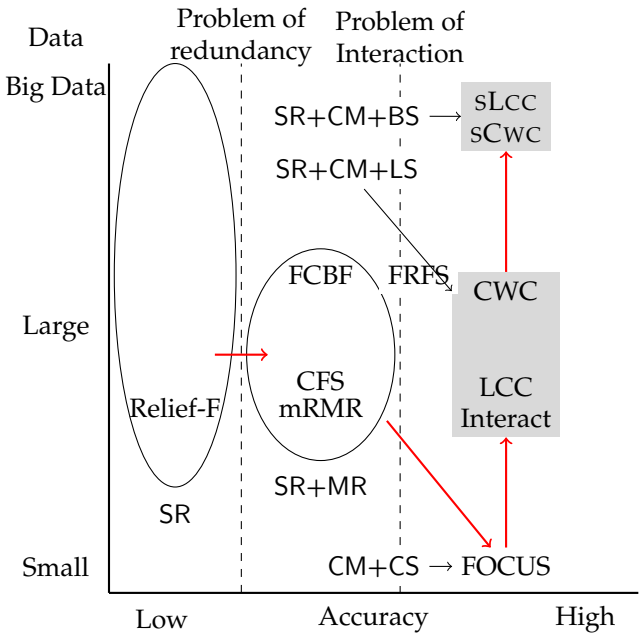


Figure 3. Progress of feature selection.

3. sCwc and sLcc

sCwc and sLcc improve the time efficiency of Cwc and Lcc significantly. The letter "s" of sCwc and sLcc represents "scalable", "swift" and "superb".

3.1. The Algorithms

We start by explaining the Cwc algorithm. Figure 1 depicts the algorithm. Given a dataset described by a feature set $\{F_1, \ldots, F_N\}$, Cwc aims to output a *minimal consistent subset* $S \subseteq \{F_1, \ldots, F_N\}$.

Definition 2. *A minimal consistent subset S satisfies* $\mathrm{Bn}(S) = 0$ *and* $\mathrm{Bn}(T) > 0$ *for any proper subset* $T \subset S$.

Achieving this goal is, however, impossible if $\mathrm{Bn}(F_1, \ldots, F_N) > 0$ holds, since $\mathrm{Bn}(S) \geq \mathrm{Bn}(F_1, \ldots, F_N) > 0$ always holds by the monotonicity property of Bn. Therefore, the preliminary step of Cwc is to remove the cause of $\mathrm{Bn}(F_1, \ldots, F_N) > 0$. To be specific, if $\mathrm{Bn}(F_1, \ldots, F_N) > 0$, there exists at least one *inconsistent pair* of instances, which are identical with respect to the feature values but with different class labels. The process of denoising is thus to modify the original dataset so that it includes no inconsistent pairs. To denoise, we have two approaches as follows:

1. We can add a *dummy* feature $\widehat{F}$ to $\{F_1, \ldots, F_N\}$ and can assign a value of $\widehat{F}$ to an instance so that, if the instance is not included in any inconsistent pair, the assigned value is zero; otherwise, the assigned value is determined depending on the class label of the instance.
2. We can eliminate at least a part of the instances that are included in inconsistent pairs.

Although both of the approaches can result in $\mathrm{Bn}(F_1, \ldots, F_N) = 0$, the former seems better because useful information may be lost by eliminating instances. Fortunately, high-dimensional data usually have the property of $\mathrm{Bn}(F_1, \ldots, F_N) = 0$ from the beginning since N is very large. When $\mathrm{Bn}(F_1, \ldots, F_N) = 0$, denoising is benign and does nothing.

On the other hand, to incorporate sum-of-relevance into consistency-based feature selection, we sort features in the incremental order of their symmetrical uncertainty scores, that is, we renumber F_is so that $\mathrm{SU}(F_i) \leq \mathrm{SU}(F_j)$ if $i < j$. The symmetrical uncertainty, however, is not the mandatory choice, and we can use any measure to evaluate relevance of an individual feature so that, the greater a value of the measure is, the more relevant the feature is. For example, we can replace the symmetrical uncertainty with the mutual information $I(F, C)$.

Cwc deploys a backward elimination approach: it first sets a variable S to the entire set $\{F_1, \ldots, F_N\}$ and then investigates whether each F_i can be eliminated from S without violating the condition of $\mathrm{Bn}(S) = 0$. That is, S is updated by $S = S \setminus \{F_i\}$, if, and only if, $\mathrm{Bn}(S \setminus \{F_i\}) = 0$. Hence, Cwc continues to eliminate features until S becomes a minimal consistent subset. Algorithm 1 describes the algorithm of Cwc.

The order of investigating F_i is the incremental order of i, and, hence, the incremental order of $\mathrm{SU}(F_i)$. Since F_i is more likely to be eliminated than F_j with $i < j$, we see that Cwc stochastically outputs minimal consistent subsets with higher sum-of-relevance scores.

Algorithm 1 The algorithm of Cwc [5]

Require: A dataset described by $\{F_1, \ldots, F_N\}$ with $\mathrm{Bn}(F_1, \ldots, F_N) = 0$.
Ensure: A minimal consistent subset $S \subseteq \{F_1, \ldots, F_N\}$.

```
Sort F_1, ..., F_N in the incremental order of SU(F_i;C).
Let S = {F_1, ..., F_N}.
for i = 1, ..., N do
    if Bn(S \ {F_i}) = 0 then
        update S by S = S \ {F_i}.
    end if
end for
```

To improve the time efficiency of CWC, we restate the algorithm of CWC as follows. To illustrate, let S_0 be a snapshot of S immediately after CWC has selected F_k. In the next step, CWC investigates whether $\mathrm{Bn}(S_0 \setminus \{F_{k+1}\}) = 0$ holds. If so, CWC investigates whether $\mathrm{Bn}(S_0 \setminus \{F_{k+1}, F_{k+2}\}) = 0$ holds. CWC continues the same procedure, until it finds F_ℓ with $\mathrm{Bn}(S_0 \setminus \{F_{k+1}, F_{k+2}, \ldots, F_\ell\}) > 0$. This time, CWC does not eliminate F_ℓ. S is set to $S_0 \setminus \{F_{k+1}, F_{k+2}, \ldots, F_{\ell-1}\}$, and CWC investigate $F_{\ell+1}$ next. Thus, to find F_ℓ to be selected, CWC solves the problem to find ℓ such that

$$\ell = \min\{i \mid i \in \{k+1, \ldots, N\}, \mathrm{Bn}(S \setminus \{F_{k+1}, \ldots, F_i\}) > 0\}.$$

To solve the problem, CWC relies on linear search.

On the other hand, the idea of improving CWC is obtained by looking at the same problem from a different direction. By the monotonicity property of Bn, $\mathrm{Bn}(S_0 \setminus \{F_{k+1}, F_{k+2}, \ldots, F_i\}) \geq \mathrm{Bn}(S_0 \setminus \{F_{k+1}, F_{k+2}, \ldots, F_\ell\}) > 0$ holds for any $i \geq \ell$, and, therefore, the formula

$$\ell - 1 = \max\{i \mid i \in \{k+1, \ldots, N\}, \mathrm{Bn}(S \setminus \{F_{k+1}, \ldots, F_i\}) = 0\}$$

also characterizes ℓ.

This characterization of ℓ indicates that we can take advantage of binary search instead of linear search to find ℓ (Algorithm 2). Since the average time complexity of the binary search is $O(\log(N-k))$, we can expect significant improvement compared with the time complexity of $O(N-k)$ of the linear search used in CWC. Algorithm 3 depicts our improved algorithm, sCWC.

Algorithm 2 Binary search to find ℓ

Require: $S \subseteq \{F_1, \ldots, F_N\}$ and $k \in \{1, \ldots, N-1\}$ such that $S \supseteq \{F_k, \ldots, F_N\}$ and $\mathrm{Bn}(S) = 0$.
Ensure: $\ell \in \{k+1, \ldots, N\}$ such that $\ell = \arg\min\{\mathrm{Bn}(S \setminus \{F_{k+1}, \ldots, F_i\}) > 0 \mid i = k+1, \ldots, N\}$.

```
if Bn(S \ {F_{k+1}, ..., F_N}) = 0 then
    ℓ = None.                                   ▷ No such ℓ exists.
else
    Let low, high, mid = k, N, ⌈(low+high)/2⌉.
    repeat
        if Bn(S \ {F_{k+1}, ..., F_mid}) > 0 then
            Let high = mid.
        else
            Let low = mid.
        end if
        Let mid = ⌈(low+high)/2⌉.
    until mid = high holds.
    ℓ = high.
end if
```

In addition, Algorithm 4 depicts the algorithm of LCC [4]. In contrast to CWC, LCC accepts a threshold parameter $\delta \geq 0$. The parameter determines the strictness of its elimination criteria. The greater δ is, the looser the criteria is, and, therefore, the smaller features LCC selects.

There are two major differences between LCC and CWC: first, LCC does not require that the entire feature set $\{F_1, \ldots, F_N\}$ is consistent. Therefore, denoization to make $\mathrm{Bn}(F_1, \ldots, F_N) = 0$ is not necessary. Secondly, the elimination criteria of $\mathrm{Bn}(S \setminus \{F_i\}) = 0$ of CWC is replaced with $\mathrm{Br}(S \setminus \{F_i\}) \leq \delta$. By the determinacy property, $\mathrm{Br}(S \setminus \{F_i\}) = 0$, if, and only if, $\mathrm{Bn}(S \setminus \{F_i\}) = 0$. Hence, with $\delta = \mathrm{Br}(F_1, \ldots, F_N) = 0$, LCC selects the same features as CWC does.

Algorithm 3 The algorithm of SCWC

Require: A dataset described by $\{F_1, \ldots, F_N\}$ with $\mathrm{Bn}(F_1, \ldots, F_N) = 0$.
Ensure: A minimal consistent subset $S \subseteq \{F_1, \ldots, F_N\}$.
1: Sort $F_1, \ldots, F_N$ in the incremental order of $\mathrm{SU}(F_i; C)$.
2: Let $S = \{F_1, \ldots, F_N\}$.
3: Let $k = 0$.
4: **repeat**
5: Find $\ell = \arg\min\{\mathrm{Bn}(S \setminus \{F_{k+1}, \ldots, F_i\}) > 0 \mid i = k+1, \ldots, N\}$ by binary search.
6: **if** ℓ does not exist **then**
7: Let $\ell = N + 1$.
8: **end if**
9: Update S by $S = S \setminus \{F_{k+1} \ldots F_{\ell-1}\}$.
10: Let $k = \ell$.
11: **until** $k \geq N$ holds.

Algorithm 4 The algorithm of LCC [4]

Require: A dataset described by $\{F_1, \ldots, F_N\}$ and a non-negative threshold δ.
Ensure: A minimal δ-consistent subset $S \subseteq \{F_1, \ldots, F_N\}$.
1: Sort $F_1, \ldots, F_N$ in the incremental order of $\mathrm{SU}(F_i; C)$.
2: Let $S = \{F_1, \ldots, F_N\}$.
3: **for** $i = 1, \ldots, N$ **do**
4: **if** $\mathrm{Br}(S \setminus \{F_i\}) \leq \delta$ **then**
5: update S by $S = S \setminus \{F_i\}$.
6: **end if**
7: **end for**

Since the Bayesian risk has the monotonicity property as well, $b_i = \mathrm{Br}(S \setminus \{F_{k+1}, \ldots, F_i\})$ compose an increasing progression, and, hence, we can find ℓ such that

$$\begin{aligned} \ell &= \min\{i \mid i \in \{k+1, \ldots, N\}, \mathrm{Br}(S \setminus \{F_{k+1}, \ldots, F_i\}) > \delta\} \\ &= 1 + \max\{i \mid i \in \{k+1, \ldots, N\}, \mathrm{Br}(S \setminus \{F_{k+1}, \ldots, F_i\}) \leq \delta\} \end{aligned}$$

very efficiently by means of binary search. Algorithm 5 describes the improved algorithm of SLCC based on binary search.

Algorithm 5 The algorithm of SLCC

Require: A finite dataset and a non-negative threshold δ.
Ensure: A minimal δ-consistent subset $S \subseteq \{F_1, \ldots, F_N\}$.
1: Sort $F_1, \ldots, F_N$ in the incremental order of $\mathrm{SU}(F_i; C)$.
2: Let $S = \{F_1, \ldots, F_N\}$.
3: Let $k = 0$.
4: **repeat**
5: Find $\ell = \arg\min\{\mathrm{Br}(S \setminus \{F_{k+1}, \ldots, F_i\}) > \delta \mid i = k+1, \ldots, N\}$ by binary search.
6: **if** ℓ does not exist **then**
7: Let $\ell = N + 1$.
8: **end if**
9: Update S by $S = S \setminus \{F_{k+1} \ldots F_{\ell-1}\}$. Let $k = \ell$.
10: **until** $k \geq N$ holds.

3.2. Complexity Analysis

When N_F and N_I denote the number of features and instances of a dataset, the average time complexity of CWC is estimated by $O(N_F N_I(N_F + \log N_I))$ [12]. The first term $O(N_F^2 N_I)$ represents the feature selection computation, while the second term $O(N_F N_I \log N_I)$ shows the instance sorting process. In [12], it is shown that sorting instances at the initial stage of the algorithm is highly effective for investigating $\mathrm{Bn}(S \setminus \{F_i\}) = 0$ efficiently (see also Section 6.1). Since SCWC improves the time-efficiency of selecting features by replacing the linear search of CWC with binary search, we can estimate its time complexity by $O(N_F N_I(\log N_F + \log N_I))$. By the same analysis, we can conclude that the same estimate of time complexity applies to SLCC.

We have verified this estimate of time complexity through experiments using high-dimensional datasets described in Table 3, whose dimensions vary from 15,741 to 38,822. Figure 4 plots the experimental results: the x-axis represents $N_F N_I(\log N_F + \log N_I)$, while the y-axis represents run-time of SCWC in milliseconds. We observe that the plots are approximately aligned along a straight line, and, hence, can conclude that the aforementioned estimate is right.

Table 3. Run-time of SCWC when applied to real high-dimensional data.

# of Instances	# of Features	Run-Time (ms)	# of Instances	# of Features	Run-Time (ms)
15,568	15,741	2529	93,862	97,261	189,018
16,319	17,221	3682	103,063	103,063	233,562
22,540	21,667	2064	108,715	106,808	247,292
22,540	21,684	3547	142,811	102,083	310,531
23,036	19,723	7378	150,402	37,610	49,303
26,319	17,221	2576	150,517	37,601	54,149
34,125	29,367	12,189	155,244	37,659	47,497
37,057	26,938	12,062	161,425	38,822	72,298
44,471	30,828	55,581	179,765	99,930	367,125
44,812	32,721	21,191	183,978	100,622	401,111
45,284	34,123	7500	184,108	99,588	458,469
48,348	35,056	8873	185,325	100,466	391,873
52,400	39,570	26,770	187,929	98,562	403,179
64,193	48,810	7017	195,736	99,339	461,130
71,814	49,974	41,089	195,887	99,419	394,033
90,797	94,707	162,052	200,569	99,672	405,803

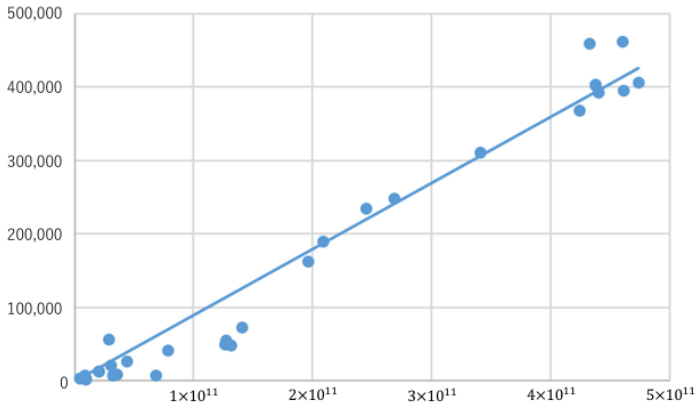


Figure 4. A relationship between $N_F N_I(\log N_F + \log N_I)$ (the x-axis) and run-time of SCWC (the y-axis).

4. Comparison of Feature Selection Algorithms

We compare SCWC and SLCC with four benchmark algorithms, namely, FRFS, CFS, RELIEF-F and FCBF, with respect to the accuracy, the run-time and the number of features selected. FRFS [10]

is a rule-based algorithm, while CFS [9], RELIEF-F [13] and FCBF [8] are sum-of-relevance-based algorithms. In addition, CFS and RELIEF-F are designed to avoid redundant selection of features.

4.1. Datasets to Use

For the comparison, we use 15 relatively large datasets, since the benchmark algorithms are not fast enough to apply to really high-dimensional datasets such as those described in Table 3. Table 4 describes the datasets, and Figure 5 plots the number of features N_F (x-axis) and the number of instances N_I (y-axis) of each dataset.

To make the comparison fair, ten of the datasets are chosen from the feature selection challenges of Neural Information Processing Systems (NIPS) 2003 [14] and World Congress on Computational Intelligence (WCCI) 2006 [15]. The datasets of NIPS 2003 emphasize the largeness of the feature number N_F, while those of WCCI 2006 do the instance number N_I. The remaining five datasets are retrieved from the University of California, Irvine (UCI) repository of machine learning databases [16].

Table 4. Attributes of the 15 datasets used in the experiment for comparison of accuracy.

#	Dataset	# of Features	# of Instances	Reference
1	ADA	48	4147	[15]
2	ADS	1558	3279	[16]
3	ARCENE	10,000	100	[14]
4	CYLINDER	40	512	[16]
5	DEXTER	20,000	300	[14]
6	DOROTHEA	100,000	800	[14]
7	GINA	970	3153	[15]
8	GISETTE	5000	6000	[14]
9	HIVA	1617	3845	[15]
10	KR-VS-KP	36	3196	[16]
11	MADELON	500	2000	[14]
12	MUSHROOM	22	8124	[16]
13	NOVA	16,969	1754	[15]
14	SPLICE	60	3192	[16]
15	SYLVA	216	13,086	[15]

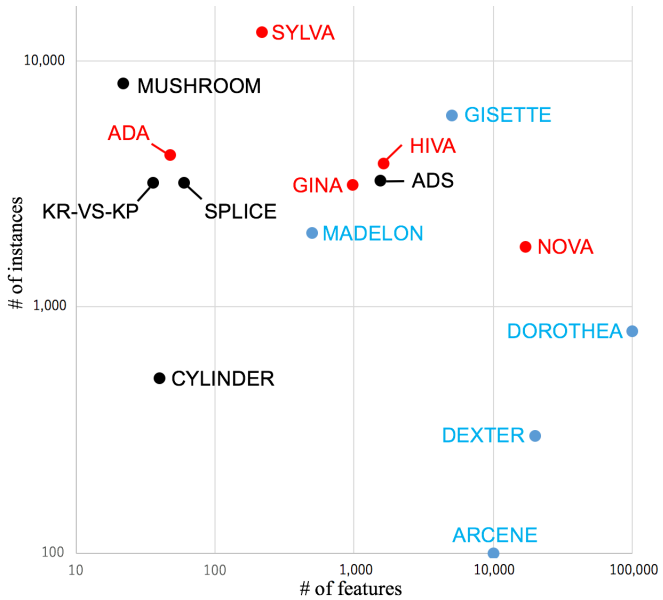


Figure 5. The fifteen datasets used in the experiment. The blue plots (•) represent the five datasets used in the feature selection challenge of NIPS 2003 [14], while the red plots (•) do those used in the challenge of WCCI 2006 [15]. The other five are retrieved from the USI repository [16].

4.2. Comparison of Accuracy

When the input dataset is described by a consistent feature set, that is, when $\mathrm{Bn}(\{F_1, \ldots, F_N\}) = 0$ holds, sLCC with $\delta = 0$ selects the same features as sCWC does. Hence, we compare the best the area under an receiver operating characteristic curve (AUC-ROC) scores of sLCC when the parameter δ varies from 0 to 0.02 at interval of 0.002 with the AUC-ROC scores of the other benchmark algorithms. In addition, since many of the benchmark algorithms cannot finish feature selection for the dataset DOROTHEA within a reasonable time allowance, we do not use the dataset for the purpose of comparison in accuracy.

4.2.1. Method

We generate 10 pairs of training and test data subsets from each dataset of Figure 5 by distributing the instances to test and training data subsets at random with a ratio of 4:1, and perform the following for each pair:

(1) We run the feature selection algorithms on the training data subset and then reduce the training dataset so that the selected features describe the *reduced training data subset*.
(2) We reduce the test data subset so that the selected features describe the *reduced test data subset*.
(3) We train three classifiers with the reduced training data subset. The classifiers to use are C Support Vector Machine with Radial Base Function (RBF-Kernel-C-SVM), Naïve Bayes and C4.5. Optimal values for the γ and C parameters of the RBF-kernel-C-SVM and the confidence factor of C4.5 are chosen through grid search with ten-fold cross validation on the reduced training data subset.
(4) We make the trained classifiers predict class labels for all of the instances of the reduced test data subset and compute scores of the accuracy measures of AUC-ROC (Area Under Curve of ROC curve) and F-measure by comparing the obtained prediction and the true class labels.

For sLCC, we run experiments with sLCC changing δ from 0 to 0.02 at interval of 0.002 and select the best scores.

4.2.2. Results and Analysis

Tables 5–10 describe the result of the comparison. For each combination of a classifier and an accuracy measure, we see the raw scores of the six feature selection algorithms in the upper rows and their rankings in the lower rows. Figures 6 and 7 also depict the same information, where sLCC, FRFS, CFS, RELIEF-F and FCBF are displayed in the colors of blue, orange, gray, yellow and light blue, respectively.

Remarkably, for all the combinations of classifiers and accuracy measures, sLCC and FRFS monopolize the first and second places with respect to both of the averaged raw scores and ranks. Furthermore, sLCC outperforms the others except for the combination of Naïve Bayes and AUC-ROC with respect to the averaged raw scores. With respect to the averaged ranks, sLCC is ranked top for the combinations of SVM and AUC-ROC, SVM and F-Score and Naïve Bayes and F-Score, while FRFS outperforms sLCC for the other three combinations.

Table 11 shows for each feature selection algorithm its averaged scores of AUC-ROC score and F-measure across the three classifiers and the 15 datasets. We see that sLCC outperforms the other algorithms for both AUC-ROC and F-Scores. Table 11 also shows the averaged ranks of the feature selection algorithms across the two accuracy measures, three classifiers and the 14 datasets. sLCC and FRFS turn out to have the same averaged rank, and they are evidently superior to the other three benchmark algorithms.

To verify the observed superiority of sLCC and FRFS, we conduct non-parametric multiple comparison tests following the recommendation by Demšar [17]. To be specific, we have performed the Friedman test and then the Hommel test. To avoid the type II error of a multiple comparison test, the tests are conducted only once based on the averaged ranks displayed in Table 11, which are

computed across all of the combinations of a classifier, an accuracy measures and a dataset. The results of the tests are described in the left column of Table 11. For the Freedman test, the observed p-value is extremely small and displayed as 0.00, and, therefore, we reject the null hypothesis and conclude that there exists a statistically significant difference among the feature selection algorithms. In the Hommel test, which follows the Friedman test, we use sLCC as a control. The calculated p-values are negligibly small for CFS, RELIEF-F and FCBF, and, hence, we can conclude that the observed superiority of sLCC to CFS, RELIEF-F and FCBF is statistically significant. On the other hand, the results of the Hommel test indicates that sLCC and FRFS are compatible with each other, since the corresponding p-value is as great as 0.981, which is very close to 1.0.

As a conclusion, to obtain high accuracy, we can recommend to use sLCC and FRFS for feature selection. To emphasize the difference between these two algorithms, sLCC will perform better when used with SVM, while FRFS will perform better when used with C4.5.

Table 5. Support Vector Machine (SVM) and the Area Under an Receiver Operating Characteristic Curve (AUC-ROC). **Av.** denotes averaged values.

Dataset	1	2	3	4	5	7	8	9	10	11	12	13	14	15	Av.
RAW SCORES															
LCC	0.748	0.910	0.773	0.701	0.862	0.879	0.952	0.652	0.992	0.835	1.00	0.869	0.964	0.973	0.865
FRFS	0.763	0.839	0.604	0.643	0.793	0.956	0.980	0.688	0.941	0.605	1.00	0.894	0.973	0.989	0.833
CFS	0.745	0.895	0.650	0.559	0.855	0.863	0.912	0.600	0.937	0.749	0.990	0.863	0.947	0.888	0.818
RELIEF	0.743	0.873	0.500	0.681	0.706	0.621	0.548	0.641	0.929	0.565	1.00	0.540	0.956	0.952	0.733
FCBF	0.745	0.884	0.582	0.637	0.741	0.817	0.929	0.596	0.937	0.602	0.990	0.816	0.948	0.903	0.795
RANKING															
LCC	2.0	1.0	1.0	1.0	1.0	2.0	2.0	2.0	1.0	1.0	2.0	2.0	2.0	2.0	1.57
FRFS	1.0	5.0	3.0	3.0	3.0	1.0	1.0	1.0	2.0	3.0	2.0	1.0	1.0	1.0	2.00
CFS	3.5	2.0	2.0	5.0	2.0	3.0	4.0	4.0	3.5	2.0	4.5	3.0	5.0	5.0	3.46
RELIEF	5.0	4.0	5.0	2.0	5.0	5.0	5.0	3.0	5.0	5.0	2.0	5.0	3.0	3.0	4.07
FCBF	3.5	3.0	4.0	4.0	4.0	4.0	3.0	5.0	3.5	4.0	4.5	4.0	4.0	4.0	3.89

Table 6. SVM and F-Score.

Dataset	1	2	3	4	5	7	8	9	10	11	12	13	14	15	Av.
RAW SCORES															
LCC	0.826	0.965	0.756	0.713	0.860	0.879	0.952	0.961	0.992	0.836	10.00	0.902	0.963	0.992	0.900
FRFS	0.848	0.942	0.578	0.651	0.789	0.956	0.980	0.970	0.943	0.605	10.00	0.927	0.973	0.996	0.868
CFS	0.833	0.962	0.624	0.511	0.851	0.863	0.912	0.960	0.939	0.749	0.990	0.900	0.946	0.983	0.859
RELIEF	0.827	0.950	0.373	0.689	0.704	0.545	0.402	0.961	0.927	0.504	10.00	0.651	0.955	0.990	0.748
FCBF	0.836	0.959	0.532	0.643	0.727	0.817	0.929	0.961	0.939	0.602	0.990	0.871	0.947	0.985	0.838
RANKING															
LCC	5.0	1.0	1.0	1.0	1.0	2.0	2.0	3.0	1.0	1.0	2.0	2.0	2.0	2.0	1.85
FRFS	1.0	5.0	3.0	3.0	3.0	1.0	1.0	1.0	2.0	3.0	2.0	1.0	1.0	1.0	2.00
CFS	3.0	2.0	2.0	5.0	2.0	3.0	4.0	5.0	3.5	2.0	4.5	3.0	5.0	5.0	3.50
RELIEF	4.0	4.0	5.0	2.0	5.0	5.0	5.0	3.0	5.0	5.0	2.0	5.0	3.0	3.0	4.00
FCBF	2.0	3.0	4.0	4.0	4.0	4.0	3.0	3.0	3.5	4.0	4.5	4.0	4.0	4.0	3.64

Table 7. Naïve Bayes and AUC-ROC.

Dataset	1	2	3	4	5	7	8	9	10	11	12	13	14	15	Av.
RAW SCORES															
LCC	0.891	0.933	0.755	0.776	0.910	0.901	0.954	0.769	0.945	0.660	0.999	0.922	0.973	0.997	0.885
FRFS	0.887	0.954	0.640	0.919	0.903	0.906	0.955	0.803	0.965	0.634	0.999	0.930	0.978	0.998	0.891
CFS	0.878	0.942	0.682	0.732	0.952	0.897	0.964	0.732	0.956	0.654	0.992	0.928	0.968	0.989	0.876
RELIEF	0.870	0.851	0.768	0.715	0.762	0.901	0.938	0.688	0.975	0.572	0.998	0.542	0.979	0.997	0.825
FCBF	0.892	0.937	0.641	0.586	0.858	0.892	0.966	0.717	0.956	0.618	0.992	0.893	0.968	0.989	0.850
RANKING															
LCC	2.0	4.0	2.0	2.0	2.0	2.5	4.0	2.0	5.0	1.0	1.5	3.0	3.0	2.5	2.61
FRFS	3.0	1.0	5.0	1.0	3.0	1.0	3.0	1.0	2.0	3.0	1.5	1.0	2.0	1.0	2.00
CFS	4.0	2.0	3.0	3.0	1.0	4.0	2.0	3.0	3.5	2.0	4.5	2.0	4.5	4.5	3.07
RELIEF	5.0	5.0	1.0	4.0	5.0	2.5	5.0	5.0	1.0	5.0	3.0	5.0	1.0	2.5	3.57
FCBF	1.0	3.0	4.0	5.0	4.0	5.0	1.0	4.0	3.5	4.0	4.5	4.0	4.5	4.5	3.71

Table 8. Naïve Bayes and F-Score.

Dataset	1	2	3	4	5	7	8	9	10	11	12	13	14	15	Av.
							Raw Scores								
Lcc	0.832	0.945	0.681	0.588	0.835	0.824	0.888	0.946	0.899	0.629	0.990	0.889	0.919	0.987	0.857
Frfs	0.831	0.804	0.597	0.851	0.776	0.822	0.885	0.964	0.917	0.605	0.977	0.899	0.924	0.979	0.845
Cfs	0.789	0.952	0.585	0.622	0.849	0.818	0.901	0.946	0.927	0.612	0.986	0.889	0.912	0.979	0.841
Relief	0.772	0.897	0.737	0.660	0.659	0.823	0.873	0.894	0.922	0.501	0.955	0.638	0.927	0.982	0.803
Fcbf	0.837	0.948	0.554	0.544	0.724	0.817	0.908	0.951	0.927	0.599	0.986	0.857	0.912	0.981	0.825
							Ranking								
Lcc	2.0	3.0	2.0	4.0	2.0	1.0	3.0	3.5	5.0	1.0	1.0	2.5	3.0	1.0	2.43
Frfs	3.0	5.0	3.0	1.0	3.0	3.0	4.0	1.0	4.0	3.0	4.0	1.0	2.0	4.5	2.96
Cfs	4.0	1.0	4.0	3.0	1.0	4.0	2.0	3.5	1.5	2.0	2.5	2.5	4.5	4.5	2.86
Relief	5.0	4.0	1.0	2.0	5.0	2.0	5.0	5.0	3.0	5.0	5.0	5.0	1.0	2.0	3.57
Fcbf	1.0	2.0	5.0	5.0	4.0	5.0	1.0	2.0	1.5	4.0	2.5	4.0	4.5	3.0	3.18

Table 9. C4.5 and AUC-ROC.

Dataset	1	2	3	4	5	7	8	9	10	11	12	13	14	15	Av.
							Raw Scores								
Lcc	0.849	0.915	0.642	0.5	0.832	0.841	0.93	0.685	0.997	0.758	10.00	0.825	0.966	0.985	0.838
Frfs	0.896	0.879	0.617	0.5	0.785	0.901	0.956	0.641	0.979	0.623	10.00	0.886	0.98	0.997	0.831
Cfs	0.864	0.923	0.555	0.557	0.783	0.846	0.932	0.633	0.963	0.752	0.993	0.823	0.963	0.944	0.824
Relief	0.84	0.895	0.63	0.684	0.74	0.828	0.937	0.664	0.976	0.574	10.00	0.516	0.965	0.986	0.803
Fcbf	0.861	0.891	0.612	0.52	0.712	0.834	0.919	0.584	0.963	0.616	0.993	0.767	0.963	0.945	0.787
							Ranking								
Lcc	4.0	2.0	1.0	4.5	1.0	3.0	4.0	1.0	1.0	1.0	2.0	2.0	2.0	3.0	2.25
Frfs	1.0	5.0	3.0	4.5	2.0	1.0	1.0	3.0	2.0	3.0	2.0	1.0	1.0	1.0	2.18
Cfs	2.0	1.0	5.0	2.0	3.0	2.0	3.0	4.0	4.5	2.0	4.5	3.0	4.5	5.0	3.25
Relief	5.0	3.0	2.0	1.0	4.0	5.0	2.0	2.0	3.0	5.0	2.0	5.0	3.0	2.0	3.14
Fcbf	3.0	4.0	4.0	3.0	5.0	4.0	5.0	5.0	4.5	4.0	4.5	4.0	4.5	4.0	4.18

Table 10. C4.5 and F-Score.

Dataset	1	2	3	4	5	7	8	9	10	11	12	13	14	15	Av.
							Raw Scores								
Lcc	0.831	0.966	0.584	0.414	0.774	0.808	0.908	0.961	0.990	0.712	10.00	0.894	0.956	0.990	0.842
Frfs	0.851	0.955	0.566	0.414	0.748	0.860	0.931	0.965	0.944	0.604	10.00	0.911	0.971	0.993	0.837
Cfs	0.837	0.965	0.539	0.433	0.735	0.816	0.915	0.957	0.940	0.710	0.990	0.859	0.950	0.983	0.831
Relief	0.818	0.949	0.594	0.587	0.699	0.821	0.922	0.954	0.927	0.501	10.00	0.623	0.956	0.991	0.810
Fcbf	0.831	0.954	0.518	0.421	0.666	0.795	0.895	0.953	0.940	0.599	0.990	0.832	0.950	0.986	0.809
							Ranking								
Lcc	3.5	1.0	2.0	4.5	1.0	4.0	4.0	2.0	1.0	1.0	2.0	2.0	2.5	3.0	2.39
Frfs	1.0	3.0	3.0	4.5	2.0	1.0	1.0	1.0	2.0	3.0	2.0	1.0	1.0	1.0	1.89
Cfs	2.0	2.0	4.0	2.0	3.0	3.0	3.0	3.0	3.5	2.0	4.5	3.0	4.5	5.0	3.18
Relief	5.0	5.0	1.0	1.0	4.0	2.0	2.0	4.0	5.0	5.0	2.0	5.0	2.5	2.0	3.25
Fcbf	3.5	4.0	5.0	3.0	5.0	5.0	5.0	5.0	3.5	4.0	4.5	4.0	4.5	4.0	4.29

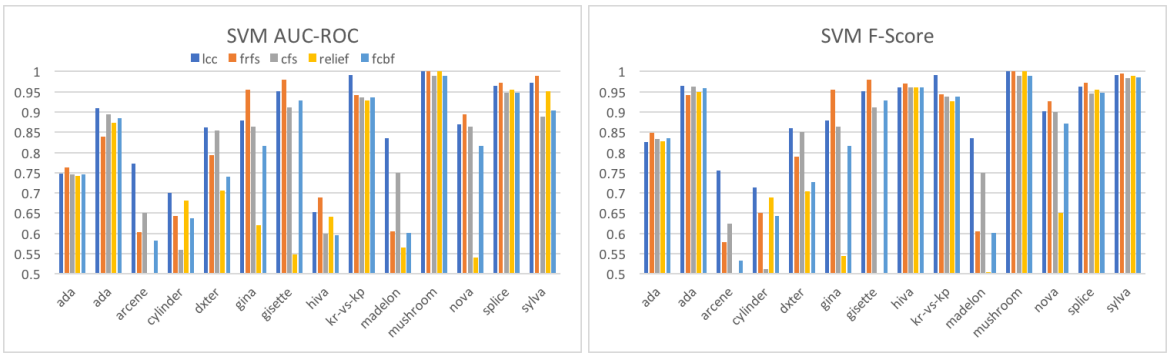


Figure 6. *Cont.*

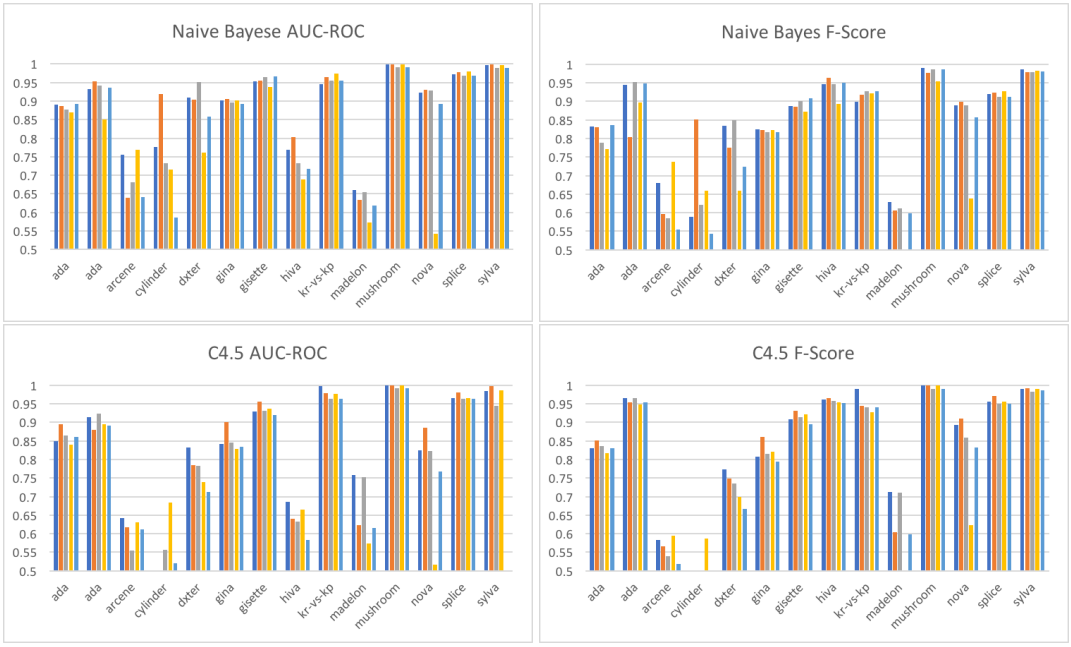


Figure 6. Comparison in accuracy.

Figure 7. Ranking.

Table 11. Overall comparison of the feature selection algorithms. The scores of the Area Under an Receiver Operating Characteristic Curve (AUC-ROC) and F-Score are the averaged values across the three classifiers and the 14 datasets. On the other hand, the averaged ranks are computed across all of the combinations of classifiers, accuracy measures and datasets. The Friedman and Hommel tests are conducted based on the averaged ranks computed here.

	Average			*p*-Value	
	AUC-ROC	**F-Score**	**Rank**	**Friedman**	**Hommel**
SLCC	0.862	0.863	20.18		CTRL
FRFS	0.852	0.850	20.18		90.81×10^{-1}
CFS	0.839	0.843	30.22	0.000	40.37×10^{-5}
RELIEF-F	0.787	0.787	30.60		10.91×10^{-8}
FCBF	0.811	0.824	30.82		90.24×10^{-11}

4.2.3. Accuracy of SLCC for Various δ

Figure 8 shows the results of experiments of performing SLCC changing the value of δ from 0 to 0.02 at interval of 0002: the AUC-ROC scores computed based on the results when using the RBF-kernel-C-SVM as a classifier for different values of δ are displayed in orange per dataset.

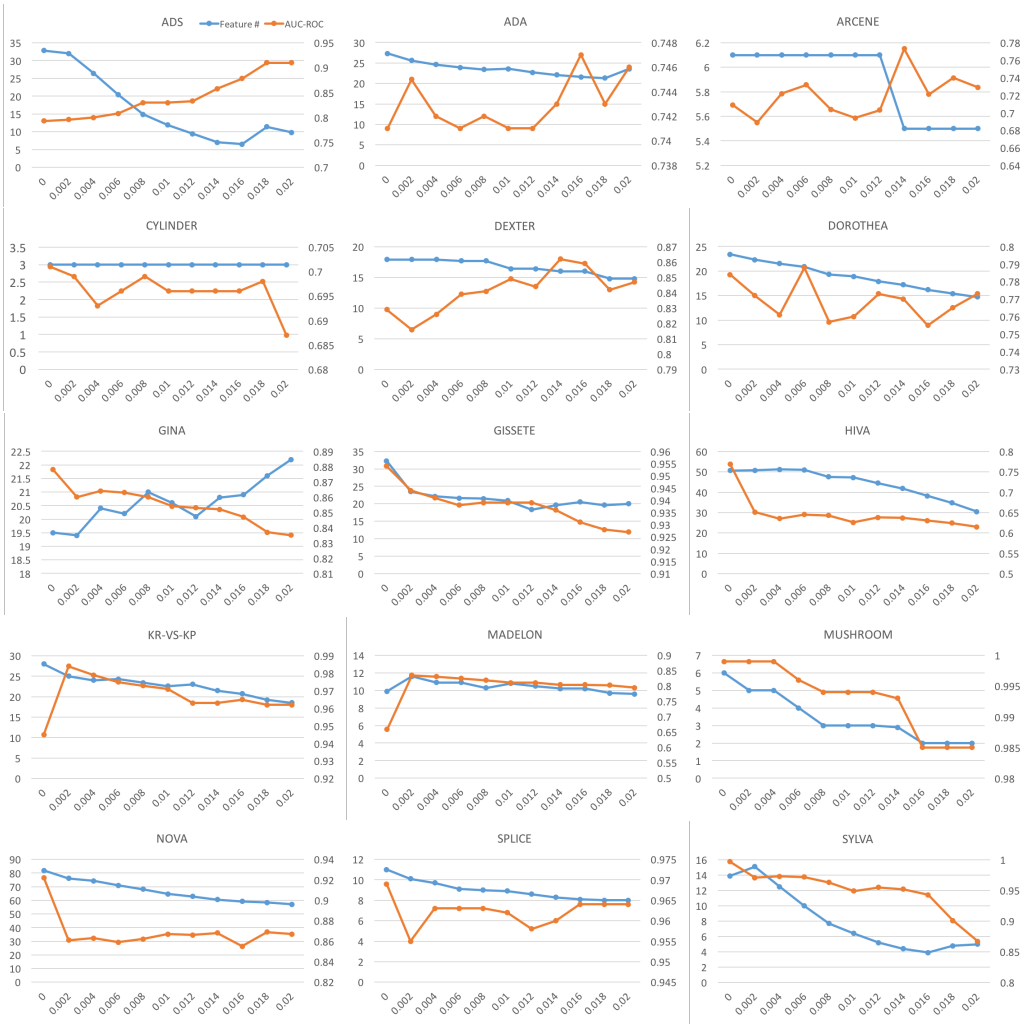


Figure 8. The Area Under an Receiver Operating Characteristic Curve (AUC-ROC) scores and numbers of features selected by SLCC changing δ from 0.0 to 0.02 at interval of 0.002. The lines and plots in blue represent feature numbers, while those in orange do AUC-ROC scores.

For the seven datasets of GINA, GISETTE, HIVA, MUSHROOM, NOVA and SILVA, that is, for almost half of the datasets investigated, the AUC-ROC score reaches the maximum when $\delta = 0$. On the other hand, we should note that the results for KR-VS-KP and MADELON show steep drop-offs of the AUC-ROC score at $\delta = 0$. Since SLCC outputs the same results as SCWC does in most of cases when $\delta = 0$, these results imply a clever way to use SCWC and SLCC: we can try SCWC first and then apply SLCC if good results are not obtained from SCWC.

Figure 8 also includes plots of the numbers of features selected (plots and lines in blue). In theory, running SLCC with a greater δ will results in selection of a smaller number of features. The results of the experiments support this only except for GINA.

4.3. Time-Efficiency

Table 12 describes the experimental results of the run-time performance of SLCC, SCWC, FRFS, CFS, RELIEF-F, FCBF, LCC, CWC and INTERACT. The feature selection algorithm of INTERACT was not used for the comparison in accuracy, since it was shown in [12] that LCC always outperform INTERACT with respect to accuracy. In the experiment, we use six datasets out of the 15 datasets described in Table 4. These six datasets need a relatively long time to perform feature selection and are appropriate for the purpose of comparing the time-efficiency of feature selection algorithms. Furthermore, we use a Mac Book Pro (2016, Apple Inc., Cupertino, CA, USA) with Quad Core i7 2.5 GHz processor and 8 GB memory. The threshold parameter δ for SLCC and LCC is set to 0.01.

From the result, we see that SLCC and SCWC outperform the others, and they have greatly improved the performance of CWC and LCC. In particular, the extent of the improvement of SLCC from LCC is remarkably greater than that of SCWC from CWC. In fact, from Table 12, we see that, even though there is a significant difference in run-time between LCC and CWC, the run-time performance of SLCC and SCWC appears comparable with each other. This can be explained as follows: with a greater δ, SLCC/LCC eliminates more features; in other words, the intervals between adjacent selected features become wider; this implies that the number of features investigated by binary search decreases, while the number of features investigated by linear search remains the same; thus, as δ increases, the extent of improvement by SLCC over LCC becomes more significant.

Table 12. Comparison of run-time (seconds) with relatively large datasets.

Dataset	SLCC	SCWC	FRFS	CFS	RELIEF-F	FCBF	LCC	CWC	INTERACT
DEXTER	0.063	0.12	0.40	35.5	1591.8	2.33	183	54.6	193
DOROTHEA	0.31	0.68	2.3	–	–	–	13,906	544	14,102
GISETTE	0.75	0.86	11.0	3,978	351	25.2	203	5.21	219
HIVA	0.76	0.42	2.16	932	3.10	3.11	14.9	1.12	15.5
NOVA	0.32	0.33	3.44	–	502	15.5	705	155	749
SYLVA	0.25	0.53	5.56	11.9	11.6	1.95	2.92	0.497	3.25
Results of the Hommel Test									
Averaged Rank	1.0	–	2.7	–	–	–	–	2.5	3.8
p-Values	CTRL.	–	0.025	–	–	–	–	0.044	0.000

The experimental results depicted by Figure 9 supports this discussion. Since LCC relies on linear search, every feature is evaluated exactly one time regardless of the value of δ. Hence, the run-time of LCC remains the same even if δ changes. By contrast, as Figure 9 shows, the run-time of SLCC decreases as δ increases. This is because the number of features selected decreases as δ increases, and, consequently, SLCC investigates a fewer number of features. When looking at Figure 8 from this viewpoint, we realize that it is a basic tendency that the number of features selected is a decreasing function of δ (the dataset of GINA is the only exception).

Table 12 also shows the results of the Hommel test to compare SLCC with FRFS, CWC and INTERACT, selected from the feature selection algorithms that can finish feature selection within

a reasonable time allowance for all of the six datasets: FRFS is the fastest of the benchmark algorithms; CWC is included to show the degree of improvement by SLCC; INTERACT is included because it is well known as the first consistency-based feature selection algorithm that is practically efficient. The displayed p-values indicate that the observed superiority of SLCC is statistically significant for the significance level of 5%. These datasets are, however, significantly smaller in size than the data to which we intend to apply SLCC and SCWC. We further investigate the efficiency performance of SLCC and SCWC in the next section.

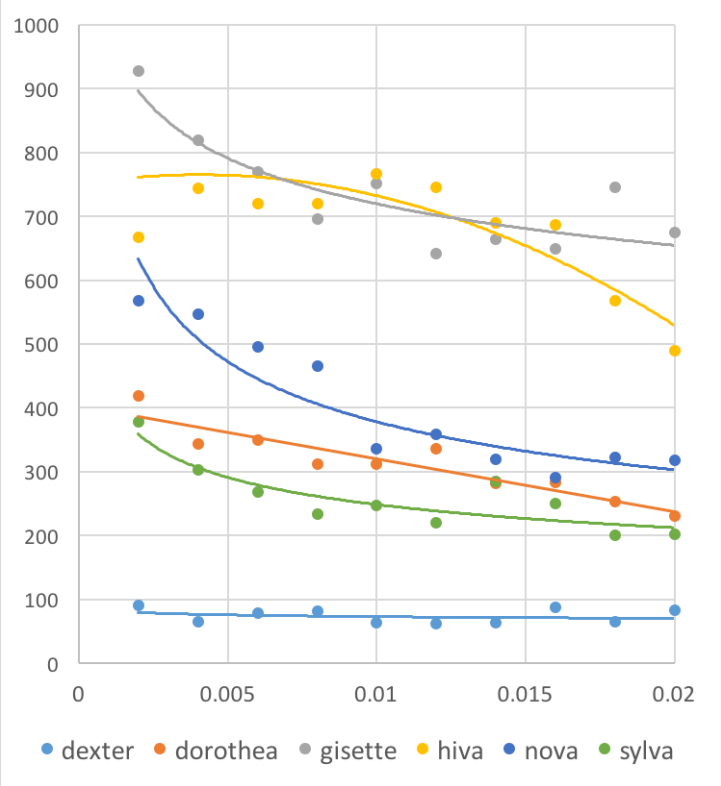


Figure 9. Relation between the run-time of SLCC and the value of δ. The x-axis represents the value of δ, while the y-axis does the run-time of SLCC in milliseconds.

5. Performance of SLCC and SCWC for High-Dimensional Data

In this section, we look into both of the accuracy and time-efficiency performance of SLCC and SCWC when applied to high-dimensional data. We use 26 real datasets studied in social network analysis, which were described in Section 1 as well. These datasets were generated from the large volume of tweets sent to Twitter on the day of the Great East Japan Earthquake, which hit Japan at 2:46 p.m. on 11 March 2011 and inflicted catastrophic damage. Each dataset was generated from a collection of tweets posted during a particular time window of an hour in length and consists of a word count vector for each author of Twitter that reflects all words in all they sent during that time window. In addition, each author was given a class label reflecting the category he or she was assigned from the k-means clustering process. We expect that this annotation represents the extent to which authors are related to the Great East Japan Earthquake.

Table 13 shows the AUC-ROC scores of C-SVM that run on the features selected by SCWC. We measured the scores using the method described in Section 4.2.1. Given time constraints, we use only 18 datasets out of the 26 datasets prepared. We see that the scores are significantly high, and the features selected well characterize the classes.

From Table 3, we observe that the run-times of SCWC on the aforementioned 26 datasets range from 1.397 s to 461.130 s, and the average is 170.017 s. Thus, this experiment shows that the time-efficiency of SCWC is satisfactory enough to apply it to high-dimensional data analysis.

Table 13. AUC-ROC of SCWC when applied to real high-dimensional data.

# of Instances	# of Features	AUC-ROC	# of Instances	# of Features	AUC-ROC
71,814	49,974	0.993	44,471	30,828	0.986
52,400	39,570	0.976	44,812	32,721	0.985
34,125	29,367	0.991	45,284	34,123	0.955
22,540	21,684	0.986	48,348	35,056	0.995
16,319	17,221	0.938	161,425	38,822	0.988
15,568	15,741	0.976	155,244	37,659	0.937
23,036	19,723	0.963	150,517	37,601	0.990
37,057	26,938	0.971	150,402	37,610	0.988
	AVERAGES		67,085	31,540	0.976

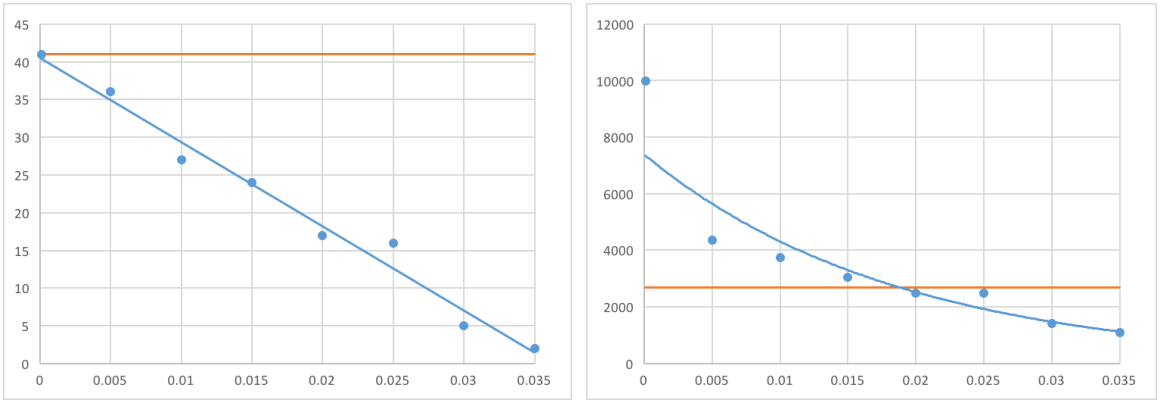
For a more precise measurement, we compare SCWC with FRFS. Table 14 shows the results. Since running FRFS takes much longer, we test only three datasets and use a more powerful computer with CentOS release 5.11 (The CentOS Project), Intel Xeon X5690 6-Cores 3.47 GHz processor and 192 GB memory (Santa Clara, CA, USA). Although we tested only a few datasets, the superiority of SCWC to FRFS is evident: SCWC is more than twenty times faster than FRFS.

Table 14. Run-time of SLCC and FRFS (CentOS release 5.11, Intel Xeon X5690 6-Cores 3.47 GHz, 198 GB memory).

	# of Instances	# of Features	SCWC (s)	FRFS (s)	Ratio
1	90,797	94,707	121.6	2849.4	23.4
2	83,862	97,261	143.5	3249.8	22.6
3	108,715	104,808	215.4	5891.6	27.3

In addition, we can conclude that SCWC remarkably improves the time-efficiency of CWC. Running CWC on the smallest dataset with 15,567 instances and 15,741 features in Table 3 requires several hours to finish feature selection. Based on this, we estimate that it will take up to ten days to process the largest dataset with 200,569 instances and 99,672 features because we know the time complexity of CWC is $O(N_F N_I (N_F + \log N_I))$. Surprisingly, SCWC has finished feature selection of this dataset in only 405 s.

Lastly, we investigate how the parameter δ can affect the performance of SLCC. As described in Section 4.3, with greater δ, SLCC will eliminate more features, and, consequently, the run-time will decrease. To verify this, we run an experiment with SLCC with the dataset with 161,425 instances and 38,822 features. Figure 10 exhibits plots of the results. In fact, the number of features selected by and the run-time of SLCC decrease as the threshold δ increases. In addition, we see that, although SLCC selects the same features as SCWC when $\delta = 0$, the run-time is greater than SCWC. This is because computing the Bayesian risk (Br) is computationally heavier than computing the binary consistency measure (Bn). It is also interesting to note that SLCC becomes faster than SCWC for greater thresholds, and their averaged run-time performance appears comparable.



(**a**) Numbers of feature selected by sLCC with different values for the parameter δ.

(**b**) Run-time of sLCC in millisecond with different values for the parameter δ.

Figure 10. The effects of different values upon the threshold parameter δ. The x-axis represents the value of δ, while the y-axis represents (**a**) the number of feature selected by and (**b**) the run-time of sLCC. The orange lines indicate the corresponding values by sCWC.

6. An Implementation

In this section, we show an implementation of our algorithms, which is also used in the experiments stated above. The data structure deployed by the implementation is a secret ingredient that makes the implementation fast. In addition, parallel computation will be possible thanks to the data structure.

6.1. The Data Structure

We consider the moment when sCWC selected $F_{k_1}, F_{k_2}, \ldots, F_{k_{\ell-1}}$ in this order and has just decided to select the feature F_k. We let $F_{k_\ell} = F_k$ and call the sequence $(F_{k_\ell}, F_{k_{\ell-1}}, \ldots, F_{k_1})$ a *prefix*. Note that $k_1 < k_2 < \cdots < k_\ell = k$ holds. In the next round, $F_{k+1}, F_{k+2}, \ldots, F_N$ are *targets* of investigation. Hence, sCWC selects one of $F_{k+1}, F_{k+2}, \ldots, F_N$, denoted by $F_{k_{\ell+1}}$, and eliminates all of $F_{k+1}, F_{k+2}, \ldots, F_{k_{\ell+1}-1}$.

In our implementation of sLCC and sCWC, at this moment, every instance of a dataset is represented as a vector of values for the sequence of features $(F_{k_\ell}, F_{k_{\ell-1}}, \ldots, F_{k_1}, F_N, F_{N-1}, \ldots, F_{k+2}, F_{k+1})$, a concatenation of the prefix and the target features, and all of the instances are aligned in the lexicographical order of the feature values.

Figure 11 shows an example. In the example, the prefix is (F_5, F_3, F_2), and the targets in the next round are F_9, F_8, F_7, F_6. For simplicity, we assume that all of the features and the class are binary variables, which take either 0 or 1 as values. The COUNT column shows the number of instances that are identical in all of the remaining features ($F_2, F_3, F_5, F_6, F_7, F_8$ and F_9) and the class.

The following are advantages of this data structure. To illustrate, we let

$$S = \{F_{k_\ell}, F_{k_{\ell-1}}, \ldots, F_{k_1}, F_N, F_{N-1}, \ldots, F_{k+1}\}.$$

1. To find inconsistent pairs of instances with respect to S, we only have to compare adjacent vectors (rows) in the data structure. We say that two instances compose an inconsistent pair with respect to S, if, and only if, the instances have the same value for every feature in S but are different in class labels.
2. To evaluate $\mathrm{Br}(S \setminus \{F_{k+1}, \ldots, F_i\})$ and $\mathrm{Bn}(S \setminus \{F_{k+1}, \ldots, F_i\})$, we only have to evaluate the measures in the reduced data structure obtained by simply eliminating the columns that correspond to $F_{k+1}, \ldots, F_i$. In the reduced data structure, instances are still aligned in the lexicographical order of values with respect to $(F_{k_\ell}, F_{k_{\ell-1}}, \ldots, F_{k_1}, F_N, F_{N-1}, \ldots, F_{i+2}, F_{i+1})$, and, hence, by investing adjacent vectors (rows), we can evaluate the measures.

3. Assume that the algorithm selects $F_{k_{\ell+1}}$ and eliminates the features $F_{k+1}, F_{k+2}, \ldots, F_{k_{\ell+1}-1}$. The necessary update of the data structure can be carried out in time linear to the number of instances: first, we simply eliminate the columns corresponding to $F_{k+1}, F_{k+2}, \ldots, F_{k_{\ell+1}-1}$; in the reduced data structure, instances are aligned in the lexicographical order of values with respect to $(F_{k_\ell}, F_{k_{\ell-1}}, \ldots, F_{k_1}, F_N, F_{N-1}, \ldots, F_{k_{\ell+1}})$; to update the prefix from $(F_{k_\ell}, F_{k_{\ell-1}}, \ldots, F_{k_1})$ to $(F_{k_{\ell+1}}, F_{k_\ell}, \ldots, F_{k_1})$, we only have to apply the bucket sort with respect to the value of $F_{k_{\ell+1}}$.

For the example of Figure 11, $\mathrm{Bn}(F_5, F_3, F_2, F_9, F_8, F_7, F_6) = 0$ is derived, since no adjacent vectors are congruent with respect to the features $F_5, F_3, F_2, F_9, F_8, F_7, F_6$.

Prefix			Targets					
F_5	F_3	F_2	F_9	F_8	F_7	F_6	Class	Count
0	0	0	0	0	0	0	0	1
0	0	0	0	0	1	0	1	4
0	0	0	0	0	1	1	1	2
0	0	0	1	0	0	0	0	1
0	0	0	1	1	0	1	1	1
0	0	1	0	0	0	0	0	4
0	0	1	0	1	0	0	1	2
1	1	0	0	0	1	1	0	5
1	1	0	0	1	0	1	1	1

Figure 11. An example of data structure.

Next, to evaluate $\mathrm{Bn}(F_5, F_3, F_2, F_9, F_8)$, we temporarily eliminate the columns of F_6 and F_7. Figure 12 shows the resulting data structure. By investigating adjacent vectors (rows), we see that the first and second instances are inconsistent with each other with respect to the features F_5, F_3, F_2, F_9, F_8. Hence, we have $\mathrm{Bn}(F_5, F_3, F_2, F_9, F_8) = 1$.

Prefix			Targets					
F_5	F_3	F_2	F_9	F_8	F_7	F_6	Class	Count
0	0	0	0	0			0	1
0	0	0	0	0			1	4
0	0	0	0	0			1	2
0	0	0	1	0			0	1
0	0	0	1	1			1	1
0	0	1	0	0			0	4
0	0	1	0	1			1	2
1	1	0	0	0			0	5
1	1	0	0	1			1	1

Figure 12. Evaluating $\mathrm{Bn}(F_5, F_3, F_2, F_9, F_8)$.

Since we can verify $\mathrm{Bn}(F_5, F_3, F_2, F_9, F_8, F_7) = 0$ by the same means, sCwc selects F_7 and eliminates F_6. The left chart of Figure 13 shows the resulting data structure after eliminating F_6 (F_7 is moved to the top of the prefix), while the right chart exhibits the result of applying bucket sort according to the value of F_7. We should note that the vectors are aligned in the lexicographical order of the values of $(F_7, F_5, F_3, F_2, F_9, F_8)$, and the data structure is ready to go to the next round of selection.

Prefix				Trgt.			
F_7	F_5	F_3	F_2	F_9	F_8	Cl.	Cnt.
0	0	0	0	0	0	0	1
1	0	0	0	0	0	1	4
1	0	0	0	0	0	1	2
0	0	0	0	1	0	0	1
0	0	0	0	1	1	1	1
0	0	0	1	0	0	0	4
0	0	0	1	0	1	1	2
1	1	1	0	0	0	0	5
0	1	1	0	0	1	1	1

⇒

Prefix				Trgt.			
F_7	F_5	F_3	F_2	F_9	F_8	Cl.	Cnt.
0	0	0	0	0	0	0	1
0	0	0	0	1	0	0	1
0	0	0	0	1	1	1	1
0	0	0	1	0	0	0	4
0	0	0	1	0	1	1	2
0	1	1	0	0	1	1	1
1	0	0	0	0	0	1	4
1	0	0	0	0	0	1	2
1	1	1	0	0	0	0	5

Figure 13. Eliminating F_6 and sorting with respect to the value of F_7.

6.2. The Program `spcwc.jar`

Instructions for using this program, namely `spcwc.jar`, are given in Table 15. The program contains implementation of both sCwc and sLcc. Using sLcc requires specifying a threshold value, which should be between $\mathrm{Br}(F_1, \ldots, F_N)$ and $\mathrm{Br}(\emptyset)$, where $F_1, \ldots, F_N$ are the entire features of a dataset. $\mathrm{Br}(\emptyset)$ is given by

$$\mathrm{Br}(\emptyset) = 1 - \max_y \Pr[C = y].$$

sLcc returns the entire set $\{F_1, \ldots, F_N\}$ when $\delta < \mathrm{Br}(F_1, \ldots, F_N)$, while it returns the empty set when $\delta \geq \mathrm{Br}(\emptyset)$.

In addition, we may change the measure used to sort features in the first step of sCwc and sLcc. Different feature selection results can be obtained with different measures. The measure can be either symmetrical uncertainty (default), mutual information, Bayesian risk, or Matthews correlation coefficient.

The program also outputs a log file with the extension `.log`. This log file contains the run-time record of the program, the features selected, the numbers of instances and features of the dataset input, and the measurements of individual features in the symmetrical uncertainty, the mutual information, the Bayesian risk and Matthews correlation coefficient.

The recommended method for using the program is to run it first with only the `i` option. Features will be sorted according to their symmetrical uncertainty scores, and, then, sCwc will select features. If we are not satisfied with the program's result, we can try other options; for example, we can try sLcc with optimized threshold values. To obtain optimized threshold, we can take advantage of any methods for hyper-parameter optimization such as the grid search and the Bayesian optimization [18].

6.3. Parallelization

Another important advantage of the data structure described in Section 6.1 is its suitability for parallel computing. Since evaluation of the Bayesian risk and the binary consistency measure can be performed only by investigating whether adjacent instances are inconsistent with each other, we can partition the entire data structure into multiple partitions and can investigate them in parallel. The data structure must be partitioned so that two adjacent instances that belong to different partitions are not congruent with respect to values of the current features because such adjacent instances cannot be inconsistent.

For example, to evaluate $\mathrm{Bn}(F_5, F_3, F_2, F_9, F_8)$ in Figure 12, we can partition the data structure of Figure 12 as Figure 14 depicts and investigate the partitions in parallel: $\mathrm{Bn}(F_5, F_3, F_2, F_9, F_8) = 0$ holds, if, and only if, any of the partitions includes no adjacent instances that are mutually inconsistent. For example, the first three instances of Figure 12 must belong to the same partition because they have identical values for the features F_5, F_3, F_2, F_9, F_8.

Prefix			Targets				Class	Count	
F_5	F_3	F_2	F_9	F_8	F_7	F_6			
0	0	0	0	0			0	1	
0	0	0	0	0			1	4	Partition 1
0	0	0	0	0			1	2	
0	0	0	1	0			0	1	
0	0	0	1	1			1	1	Partition 2
0	0	1	0	0			0	4	
0	0	1	0	1			1	2	
1	1	0	0	0			0	5	Partition 3
1	1	0	0	1			1	1	

Figure 14. An example of data structure.

Table 15. Usage of `SPCWC.JAR`.

Option	Values	Description
`-i`	`<path>`	Path to the input attribute-relation file format (arff) file.
`-a`	`<algorithm>`	The feature selection algorithm to use.
	`cwc`	Run SCWC (default).
	`lcc`	Run SLCC.
`-t`	`<number>`	A threshold value for SLCC.
		The value should be in the interval [0,1).
		When the value 0 is specified, SCWC will run, even if `-a lcc` is specified.
`-s`	`<measure>`	A statistical measure to use when sorting features.
	`su`	The symmetrical uncertainty will be used (default).
	`mi`	The mutual information will be used.
	`br`	The Bayesian risk will be used.
	`mc`	Matthews correlation coefficient will be used.

7. Conclusions

Feature selection is a useful tool for data analysis, and, in particular, is useful to interpret phenomena that you find in data. Consequently, feature selection has been studied intensively in machine learning research, and multiple algorithms that exhibit excellent accuracy have been developed. Nevertheless, such algorithms are seldom used for analyzing huge data because the algorithms usually take too much time. In this paper, we have introduced two new feature selection algorithms, namely SCWC and SLCC, that scale well to huge data. They are based on the algorithms that exhibited excellent accuracy in the literature and do not harm the accuracy of the original algorithms. We have also introduced an implementation of our new algorithms and have described a recommended usage of the implementation.

Acknowledgments: This work was partially supported by the Grant-in-Aid for Scientific Research (KAKENHI Grant Number 16K12491, 17H00762 and 26280090) from the Japan Society for the Promotion of Science.

Author Contributions: This research started when Kuboyama discovered the fact that CWC can run faster when it deploys a forward selection approach instead of a backward elimination approach. This discovery was supported by the observation that CWC tends to eliminate many features before it selects the first feature. After an intensive discussion between Shin and Kuboyama, they reached the algorithm of SCWC, which deploys binary search, and as a result, has a significantly improved time complexity. Furthermore, Shin extended the result to LCC and developed the algorithm of SLCC, which is as fast as SCWC and shows better accuracy performance. Shin and Kuboyama collaborated to implement the algorithms. Hashimoto generated the SNS datasets to use in their experiments and conducted the experiments collaboratively with Shin. Shepard applied the algorithms to real problems that involved large scale data and provided valuable comments, which contributed to the improvement of the algorithms. Shin wrote this paper.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Almuallim, H.; Dietterich, T.G. Learning boolean concepts in the presence of many irrelevant features. *Artif. Intell.* **1994**, *69*, 279–305
2. Liu, H.; Motoda, H.; Dash, M. A monotonic measure for optimal feature selection. In Proceedings of the European Conference on Machine Learning, Chemnitz, Germany, 21–23 April 1998.
3. Zhao, Z.; Liu, H. Searching for Interacting Features. In Proceedings of the International Joint Conference on Artificial Intelligence, Hyderabad, India, 6–12 January 2007; pp. 1156–1161.
4. Shin, K.; Xu, X. Consistency-based feature selection. In Proceedings of the 13th International Conferece on Knowledge-Based and Intelligent Information & Engineering System, Santiago, Chile, 28–30 September 2009.
5. Shin, K.; Fernandes, D.; Miyazaki, S. Consistency Measures for Feature Selection: A Formal Definition, Relative Sensitivity Comparison, and a Fast Algorithm. In Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Spain, 16–22 July 2011; pp. 1491–1497.
6. Kononenko, I. *Estimating Attributes: Analysis and Extension of RELIEF*; Springer: Berlin/Heidelberg, Germany, 1994.
7. Peng, H.; Long, F.; Ding, C. Feature selection based on mutual information: Criteria of max-dependency, max-relevance and min-redundancy. *IEEE Trans. Pattern Anal. Mach. Intell.* **2005**, *27*, 1226–1238.
8. Yu, L.; Liu, H. Feature selection for high-dimensional data: A fast correlation-based filter solution. In Proceedings of the Twentieth International Conference on Machine Learning, Washington, DC, USA, 21–24 August 2003.
9. Hall, M.A. Correlation-based feature selection for discrete and numeric class machine learning. In Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000), Stanford, CA, USA, 29 June–2 July 2000.
10. Wang, G.; Song, Q.; Xu, B.; Zhou, Y. Selecting feature subset for high dimensional data via the propositional FOIL rules. *Pattern Recognit.* **2013**, *46*, 199–214.
11. Quinlan, J.; Cameron-Jones, R. FOIL: A midterm report. In Proceedings of the European Conference on Machine Learning, Vienna, Austria, 5–7 April 1993; Springer: Berlin/Heidelberg, Germany, 1993; pp. 1–20.
12. Shin, K.; Miyazaki, S. A Fast and Accurate Feature Selection Algorithm based on Binary Consistency Measure. *Comput. Intell.* **2016**, *32*, 646–667.
13. Kira, K.; Rendell, L. A practical approach to feature selection. In Proceedings of the 9th International Workshop on Machine Learning, Aberdeen, UK, 1–3 July 1992; pp. 249–256.
14. Neural Information Processing Systems (NIPS). *Neural Information Processing Systems Conference 2003: Feature Selection Challenge*; NIPS: Grenada, Spain, 2003.
15. World Congress on Computational Intelligence (WCCI). In Proceedings of the IEEE World Congress on Computational Intelligence 2006: Performance Prediction Challenge, Vancouver, BC, Canada, 16–21 July 2006.
16. Blake, C.S.; Merz, C.J. UCI Repository of Machine Learning Databases; Technical Report; University of California: Irvine, CA, USA, 1998.
17. Demšar, J. Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Theory* **2006**, *7*, 1–30.
18. Snoek, J.; Larochelle, H.; Adams, R.P. Practical Bayesian Optimization of Machine Learning Algorithms. *arXiv* **2012**, arXiv:1206.2944.