Online Feature Selection with Streaming Features

Xindong Wu[1,2], Kui Yu[1], Wei Ding[3], Hao Wang[1], and Xingquan Zhu[4*]

Abstract—**We propose a new online feature selection framework for applications with streaming features where the knowledge of the full feature space is unknown in advance. We define streaming features as features that flow in one by one over time whereas the number of training examples remains fixed. This is in contrast with traditional online learning methods that only deal with sequentially added observations, with little attention being paid to streaming features. The critical challenges for online streaming feature selection include (1) the continuous growth of feature volumes over time; (2) a large feature space, possibly of unknown or infinite size; and (3) the unavailability of the entire feature set before learning starts.**

In the paper, we present a novel Online Streaming Feature Selection (OSFS) method to select strongly relevant and non-redundant features on the fly. An efficient Fast-OSFS algorithm is proposed to improve feature selection performance. The proposed algorithms are evaluated extensively on high-dimensional datasets and also with a real-world case study on impact crater detection. Experimental results demonstrate that the algorithms achieve better compactness and higher prediction accuracy than existing streaming feature selection algorithms.

***Index Terms*— Feature selection, streaming features, supervised learning**

I. INTRODUCTION

Feature selection in predictive modeling has received considerable attention in statistics and machine learning [14-15, 26, 43] during the last three decades. A variety of feature selection algorithms have been developed and proven to be effective in improving prediction accuracy for classification [2, 20, 33]. Traditional feature selection methods assume that all features are pre-computed and presented to a learner before feature selection takes place. This assumption, however, is often violated in many real-world applications where not all features can be present in advance. For example, many image processing processes involve a search of potential features for machine learning algorithms to fulfill the pattern recognition goal, but image features are often expensive to generate and store and therefore may exist in a streaming format [13, 22, 42**].** More specially, Mars crater detection from high resolution planetary images is an important task in planetary research because it provides an effective solution for measuring the relative ages of planetary surfaces. While texture features have proven to be effective in crater detection [11**]**, tens of thousands of texture-based features, in different scales and different resolutions, can potentially be generated for high resolution planetary images. It is infeasible to pre-generate texture features from planetary images that have a near global coverage of the Martian surface.

An intriguing question is that if we need a high level of computational effort to generate those features up front, should we develop a new way of integrating new features as they arrive and carry out the computation, or should we wait a long time for all features to become available and then carry out the learning process? This presents an interesting research question on how to design an effective mechanism to deal with feature selection without the knowledge of the full feature space. At the same time, when the potential feature space is enormous, an exhaustive search over the entire feature space becomes very costly or simply infeasible. Under such circumstances, we need an effective design to ensure that feature selection is properly and effectively carried out, even though smoothing through the entire feature space is simply not an option.

Indeed, many existing feature selection algorithms are effective in selecting a subset of predictive features for various classification problems, but their scope is essentially limited to the problem settings that all features are given before the learning begins and they therefore cannot deal with the above challenges [9, 22, 34].

Motivated by these observations, we formulate dynamic features as streaming features, whereby features are no longer static but flow in one by one, and each new feature is processed upon its arrival. Based on the newly formulated problem, we present a novel framework for selecting features from streaming features, which is inspired by feature relevance and feature redundancy. This framework involves two key components: (1) the utilization of feature relevance to select features on the fly, and (2) the removal of redundant features from the selected candidates thus far, based on feature redundancy. Two new algorithms, Online Streaming Feature Selection (OSFS) and Fast-OSFS, are proposed to validate the effectiveness of the proposed framework.

In summary, the unique contributions that distinguish the proposed work from existing approaches are threefold: (1) our work advances the relevance- and redundancy-based feature selection one step further for handling streaming features; (2) a novel framework based on feature selection is proposed to manage streaming features; and (3) two new online streaming feature selection algorithms are proposed with extensive comparisons and experimental studies.

The remainder of the paper is organized as follows. Section 2 discusses related work. Section 3 presents the proposed framework for streaming feature selection. Section 4 describes two algorithmic solutions to the streaming feature selection problem. Section 5 reports experimental results and a case

[1] School of Computer Science and Information Engineering, Hefei University of Technology, China

[2] Department of Computer Science, University of Vermont, USA

[3] Department of Computer Science, University of Massachusetts Boston, USA

[4] QCIS Centre, Faculty of Engineering & Information Technology, University of Technology, Sydney, Australia (* contact author)

xwu@cs.uvm.edu, ykui713@gmail.com,ding@cs.umb.edu, jsjxwangh@hfut.edu.cn, xqzhu@it.uts.edu.au

study on streaming features for impact crater detection, and Section 6 concludes the paper.

II. RELATED WORK

For many years, feature selection as an effective means for handling data with high dimensionality has been generally viewed as being the problem of searching for an optimal subset of features. Feature selection can be broadly classified into three categories: wrapper, filter, and embedded methods. A wrapper method performs a heuristic search in the space of all possible feature subsets, using a classifier of choice to assess each subset. Although this method has high accuracy, the exponential number of possible subsets makes the method computationally expensive in general [7, 18].

The filter methods, independent of any classifiers, apply evaluation measures such as distance, information, dependency, and consistency to select features and then build a classifier using selected features [5, 10, 23, 27]. Because of their simplicity and fast computational performance, many filter methods have been proposed to solve the feature selection problem [24, 30]. In recent studies especially, causal filter methods have attracted much attention [3, 6].

The embedded methods attempt to simultaneously maximize classification performance and minimize the number of features used. These types of methods are typically more efficient than the wrapper methods because a filter algorithm is built with a classifier to guide the feature selection procedure. A variety of embedded feature selection methods have been introduced, including using classification or regression as an optimization problem with specified loss and penalty functions [16, 29, 37-39].

The work discussed above shares one common assumption, which is that all candidate features are available from the very beginning, because all features are examined at each iteration to select the best feature. In the context of streaming features, feature dimensions continuously increase and not all features are presented in advance. Consequently, this poses great challenges to traditional feature selection methods.

Several research efforts have been made to address the streaming feature challenge. Perkins and Theiler considered an online feature selection problem and proposed the Grafting algorithm based on a stagewise gradient descent approach for online feature selection [22]. Grafting treats the selection of suitable features as an integral part of learning a predictor in a regularized learning framework. It optimizes the L1-regularized maximum likelihood using two iterative steps: optimizing over all the free parameters and selecting new features. Grafting operates in an incremental iterative fashion, gradually building up a feature set while training a predictor model using gradient descent. In each iteration, a fast gradient-based heuristic is used to identify a feature that is most likely to improve the existing model, with the model being incrementally optimized using gradient descent. Glocer et al. extended this algorithm to solve the edge detection problem in grayscale imagery [13]. While the Grafting algorithm is able to handle streaming features, it needs to select the value of a regularization parameter—λ in advance to determine which feature is most likely to be selected for the model at each iteration. Choosing a suitable regularization parameter λ inevitably requires information about the global feature set. Therefore, Grafting is ineffective in dealing with streaming features with an unknown feature size.

Ungar et al. and Zhou et al. studied streamwise feature selection and proposed two novel algorithms based on streamwise regression, information-investing, and Alpha-investing [31, 40-41]. Dhillon et al. extended the Alpha-investing method and proposed a multiple streamwise feature selection algorithm to address the case of multiple feature classes [12]. The Alpha-investing method sequentially considers new features for addition to a predictive model by modeling the candidate feature set as a dynamically generated stream. Alpha-investing can handle candidate feature sets of unknown or even infinite sizes. It uses linear and logistic regression to dynamically adjust the threshold of error reduction required for evaluating a new feature for inclusion by the predictive model so far.

One inherent deficiency of Alpha-investing is that it only considers adding new features but never evaluates the redundancy of selected features after new features have been added. Because the information-investing uses a very similar approach to Alpha-investing, we adopt Alpha-investing in this paper for comparative studies. Alpha-investing requires some prior knowledge about the structure of the feature space in order to heuristically control the choice of candidate feature selection. In real-world applications, obtaining sufficient prior information about the structure of the feature space is not always feasible. Our proposed framework, by comparison, makes an additional effort to manage the real-world feature selection problem in streaming features without any prerequisite for (or prior knowledge of) the feature space structure.

III. A FRAMEWORK FOR FEATURE SELECTION WITH STREAMING FEATURES

In this section, we formally define streaming features and discuss relevant special characteristics. Based on the new definition, we review notations of feature relevance and make two propositions to deal with feature redundancy in streaming features.

Definition 1 Streaming features: Streaming features involve a feature vector that flows in one by one over time while the number of training examples remains fixed.

The uniqueness of feature selection in streaming features, compared to traditional feature selection, is as follows.

- **The dynamic and uncertain nature of the feature space**. Feature dimensions may grow over time and may even extend to an infinite size.
- **The streaming nature of the feature space**. Features flow in one at a time and each feature is required to be processed online upon its arrival.

Due to the inapplicability of traditional feature selection methods for handling applications involving streaming features, we will review some notations of feature relevance and then propose two methods to handle feature redundancy in streaming features. To characterize feature relevance, an input feature can be in one of three disjoint categories, namely, strongly relevant, weakly relevant or irrelevant [18-19]. In the

definitions below, F represents a full set of features and C denotes the class attribute (note that the full feature set F does not include the class attribute C). Assuming X_i denotes the i^{th} input feature, $F - \{X_i\}$ represents the feature subset excluding feature X_i.

Definition 2 Conditional Independence [19]**:** Two distinct features $X_i \in F$ and $X_k \in F$ are conditionally independent on a subset $S \subseteq F$, iff $P(X_i|X_k, S) = P(X_i|S)$ or $P(X_k|X_i,S) = P(X_k|S)$.

Definition 3 Strong relevance: A feature X_i is strongly relevant to C iff $\forall S \subseteq F - \{X_i\}\ s.t.\ P(C|S) \neq P(C|S, X_i)$.

Definition 4 Weak relevance: A feature X_i is weakly relevant to C iff it is not strongly relevant, and

$$\exists S \subset F - \{X_i\}\ s.t.\ P(C|S) \neq P(C|S, X_i) \quad (1)$$

Definition 5 Irrelevance: A feature X_i is irrelevant to C iff it is neither strongly nor weakly relevant, and

$$\forall S \subseteq F - \{X_i\}\ s.t.\ P(C|S, X_i) = P(C|S) \quad (2)$$

Weakly relevant features can be further divided into redundant features and non-redundant features [36] based on a Markov blanket criterion [19].

Definition 6 Markov blanket: Denoting $M_i \subset F$ a subset of features, if for the given M_i the following property

$$\forall Y \in F - M_i\ s.t.\ P(C|M_i, Y) = P(C|M_i) \quad (3)$$

holds, then M_i is a Markov blanket for C (MB(C) for short).

Definition 7 Redundant features: A feature X_i is redundant to the class attribute C, if and only if it is weakly relevant to C and has a Markov blanket, $MB(X_i)$, that is a subset of the Markov blanket of MB(C).

The Markov blanket of a feature X_i subsumes the information that X_i has about C while the Markov blanket of the class attribute C carries information that all of the other features have about C. In other words, the Markov blanket of the class attribute C is the optimal feature subset which contains all the weakly relevant but non-redundant features and strongly relevant features, as shown in Figure 1.

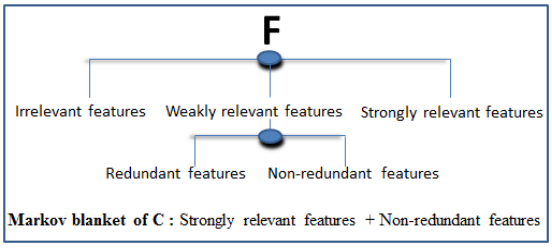

Figure 1: Feature relevance and Markov blanket of C

While irrelevant features can be easily removed, removing redundant features is the key task for an optimal feature selection process. Existing methods for removing redundant features are based on the Markov blanket criterion: let G be the current set of features (G contains the whole set of features at the beginning, *i.e.* G=F); at any phase, if there exists a Markov blanket for a feature X within the current G, X is removed from G [19]. When the full feature space is large, finding a Markov blanket for a feature is very difficult. Moreover, it is nontrivial to convert existing learning methods to deal with redundant features in streaming features because the prior knowledge about the whole feature space is unknown.

Within the context of streaming features, we set the MB(C) as an empty set initially and gradually build the MB(C) over time. We then use it to identify and remove redundant features from the streaming features. If an incoming feature is relevant to the class attribute C and is added into the current feature set, we use Proposition 1 to determine which of the selected features observed so far may become redundant as time passes.

Proposition 1 As the features flow in one by one, a current Markov blanket of the class attribute C at time t is denoted as $CMB(C)_t$. At time t+1, a new feature X_i which is relevant to C is added to $CMB(C)_{t+1}$. If for any existing feature Y in $CMB(C)_{t+1}$, $\exists S \subseteq \{CMB(C)_{t+1} - Y\}\ s.t.\ P(C|Y, S) = P(C|S)$, then Y is redundant and can be removed from $CMB(C)_{t+1}$.

Theorem 1 [19] Assume G is our current set of selected features, and a previously removed feature $X_i \notin G$ has a Markov blanket within G. When $Y \in MB(X_i)$ is removed based on a Markov blanket within G, then X_i also has a Markov blanket within G-{Y}.

Theorem 1 shows that a redundant feature removed earlier remains redundant during the rest of the process when some features within its Markov blanket are later removed. We use Corollary 1 to guarantee that Proposition 1 satisfies this desirable property.

Corollary 1 A feature X_i removed earlier by Proposition 1 remains redundant when some features within $MB(X_i)$ are removed during the rest of the streaming feature selection process.

Proof: Assuming that at time t feature X_i is discarded because its Markov blanket $MB(X_i) \subseteq CMB(C)$ by Proposition 1. Assume $\exists Y \in MB(X_i)$ is removed later. In this case, Theorem 1 guarantees that X_i also has a Markov blanket within $\{MB(X_i) - Y\} \cup \{MB(Y)\}$. Thus, in streaming features, if at time $t + \varepsilon\ (\varepsilon > 0)$, Y is removed, X_i remains redundant because it still has a Markov blanket in $CMB(C) - \{Y\}$. As a result, a feature removed earlier by Proposition 1 remains redundant during the rest of the process. □

Corollary 1 therefore guarantees that the strongly relevant features and non-redundant features can be selected as the features stream one by one over time. By Proposition 1 and Corollary 1, we propose a framework for feature selection with streaming features that contains two major steps: (1) online relevance analysis that discards irrelevant features and retains relevant ones; and (2) online redundancy analysis which eliminates redundant features from the features selected so far (see Figure 2).

1. **Initialization**
   Best candidate feature set BCF={}, the class attribute C
2. **Online Relevance Analysis**
   (1) Stream in a new feature X
   (2) Determine whether X is relevant to C.
   a. If X is irrelevant to C, then X is discarded
   b. Otherwise, X is added to BCF
3. **Online Redundancy Analysis**
   Online identify redundant features and remove them from BCF by Proposition 1.
4. Alternate steps 2 and 3 until a predefined prediction accuracy or the maximum number of iterations is reached.
5. Output selected features contained in BCF.

Figure 2: The framework for Feature Selection with Streaming Features

IV. ONLINE STREAMING FEATURE SELECTION ALGORITHMS

In this section, we propose two new algorithms to implement the framework for feature selection with streaming features: Online Streaming Feature Selection (OSFS) and an accelerated OSFS (Fast-OSFS), for streaming feature selection.

A. An Online Streaming Feature Selection Algorithm

OSFS Algorithm: The pseudo-code of the Online Streaming Feature Selection (OSFS) algorithm is shown in Figure 3, where $Ind(C, X|S)$ denotes conditional independence test between a feature X and the class attribute C given a subset S, $Dep(C, X|S)$ represents conditional dependence test, and BCF stands for the set of best candidate features so far.

The OSFS algorithm

```
1.  BCF={};                                   12. /*Online redundancy analysis*/
2.  repeat                                    13. if (added)
3.    added=0;                                14. for each feature Y ∈ BCF
4.  /* Stream in a new feature*/               15. if ∃S ⊆ BCF-Y s.t. Ind(C,Y|S)
5.    X←get_new_feature()                     16. /*Remove Y from BCF */
6.  /*Online relevance analysis*/             17.       BCF = BCF-Y;
7.    if Dep(C,X|∅)                           18.    endif
8.       added=1;                             19.  endfor
9.  /*Add X to BCF */                         20. endif
10.     BCF = BCF ∪ X;                        21. until a predefined accuracy satisfied
11. endif                                     22  output BCF
```

Figure 3: The OSFS algorithm

OSFS employs a two-phase optimal subset discovery scheme: online relevance analysis (from steps 6-11) and online redundancy analysis (from steps 12-20). In the relevance analysis phase, OSFS discovers strongly and weakly relevant features and adds them into BCF. When a new feature arrives, OSFS assesses its relevance to the class attribute C and decides to either discard the new feature or add it to BCF, according to its relevance.

Once a new feature is included into BCF, the redundancy analysis phase is triggered. In this phase, using Proposition 1, OSFS dynamically identifies and eliminates redundant features within BCF. If a subset exists within BCF to make any existing feature in BCF and the class attribute C conditionally independent, the previously selected feature Y, Y∈BCF, becomes redundant and is removed from BCF. OSFS alternates the above two phases till one of the following stopping criteria is satisfied: (1) a predefined prediction accuracy is satisfied, or (2) the maximum number of iterations is reached, or (3) no further features are available.

The Ind and Dep functions in OSFS: In Figure 3, OSFS uses the notations $Ind(C, X|S)$ and $Dep(C, X|S)$ to denote the conditional independence/dependence tests to identify irrelevant and redundant features. The tests can be implemented using the G^2 test that is an alternative to the χ^2 test [28, 21] (refer to reference [21], pp. 611-615 and reference [28], pp. 148-151for detailed explanations about the G^2 test).

We briefly explain the G^2 test using one example. With three features, X_i, X_j and X_k, we set S_{ijk}^{abc} to be the number of counts satisfying $X_i = a, X_j = b$ and $X_k = c$ in a dataset. S_{ij}^{ab}, S_{jk}^{bc} and S_k^c are defined in a similar way. If X_i and X_j are conditionally independent given X_k, the G^2 statistic is defined as

$$G^2 = 2\sum_{a,b,c} S_{ijk}^{abc} \ln \frac{S_{ijk}^{abc} S_k^c}{S_{ik}^{ab} S_{jk}^{bc}} \quad (4)$$

This formula can be easily extended to the case in which X_i and X_j are conditionally independent given a subset of S. The G^2 statistic is asymptotically distributed as χ^2 with appropriate degrees of freedom. In general, when we check the conditional independence of X_i and X_j given S, the number of degrees of freedom df used in the test is calculated as

$$df = (r_i - 1)(r_j - 1)\prod_{r_k \in S} r_k \quad (5)$$

where r_i is the domain (number of distinct values) of X_i. As a heuristic, Spirtes et al. reduced the number of degrees of freedom by one for each count that is equal to zero [28].

A significance level of α is used to measure the probability of rejecting a conditional independency hypothesis, so we can conclude the conditional independence at α level (often significance levels of 0.01 or 0.05 are used).

To measure the functions $\mathrm{Ind}(C, X|S)$ and $\mathrm{Dep}(C, X|S)$, OSFS uses a p-value returned by the G^2 test to measure these two functions. Under the null hypothesis of the conditional independence between a feature X and the class attribute C given the subset S, assuming ρ is the p-value returned by the G^2 test and α is a given significance level, the functions $\mathrm{Ind}(C, X|S)$ and $\mathrm{Dep}(C, X|S)$ in OSFS are defined as follows.

Definition 8 $\mathbf{Dep(C, X|S)}$: The function $\mathrm{Dep}(C, X|S)$ defines that X and C are conditionally dependent given S. This function holds if and only if $\rho \leq \alpha$, which concludes that the null hypothesis is rejected.

Definition 9 $\mathbf{Ind(C, X|S)}$: The function $\mathrm{Ind}(C, X|S)$ defines that X and C are conditionally independent given S. This function holds if and only if $\rho > \alpha$, which concludes that the null hypothesis is accepted.

With the above two definitions and Proposition 1, we conclude that for the current BCF, X is redundant to C and should be discarded if the function $Ind(C, X|S)$ holds conditioned on a subset S. Otherwise, we conclude that X is strongly relevant or non-redundant to C for the time being, and then add X into BCF if the function $Dep(C, X|S)$ holds conditioned on all subsets within the current BCF.

Reliability of the Ind and Dep: According to the work of [1, 28] by performing a reliable conditional independence test between X_i and X_j given X_k, the average number of instances per cell of the contingency table of $\{X_i, X_j\} \cup X_k$ must be at least φ, i.e., $N/((r_i - 1) \times (r_{j-1}) \times r_k) \geq \varphi$ (N is the total number of instances, r_i is the number of distinct values of X_i, and φ is often set to 5 or 10). With the G^2 test, the calculation of S_{ijk}^{abc} requires the counting of the number of occurrences of all different possible values for features X_i, X_j and X_k, This implies that the number of training instances required to accurately count these values is exponential to the size of the conditioning set X_k. Hence in the OSFS algorithm, we assume $\mathrm{Ind}(X_i, X_j|X_k)$ holds unless there are at least five training instances ($\varphi = 5$) for each set of different possible values of X_i, X_j and X_k.

Reliability of OSFS: As illustrated by the algorithm in Figure 3, conditional tests on line 7 and line 15 control the

reliability of OSFS because these two lines could make false positive and false negative errors that correspond to traditional type I and type II errors. A type I error in statistics is the error of rejecting the null hypothesis when it is true while a type II error is the error of accepting the null hypothesis when it is false.

The false negative errors denote that OSFS may discard strongly relevant or non-redundant features incorrectly. The dependence test on line 7 always performs a reliable test, since it is an unconditional independence test. OSFS might conduct unreliable tests on line 15 when the available instances are insufficient to check the tests conditioning on all subsets of BCF, and false negative errors may then occur. For example, an incoming feature X is strongly relevant to C. On line 15, assume BCF contains two features Y and Z at this time, thus, all of the tests $\mathrm{Dep}(C, X|Y)$, $\mathrm{Dep}(C, X|Z)$, and $\mathrm{Dep}(C, X|Y, Z)$ must hold so that X can be added into BCF. When the available instances suffice to only check the tests $\mathrm{Dep}(C, X|Y)$ and $\mathrm{Dep}(C, X|Z)$, OSFS will assume that $\mathrm{Ind}(C, X|Y, Z)$ holds and return a type II error. Therefore, to control the type II error in this case, OSFS can limit the size of the maximum conditioning subset of BCF according to the sizes of the available sample and BCF, instead of using an exhaustive search over all subsets within BCF on line 15.

To control the false positives, OSFS classifies them into irrelevant and redundant features. For an irrelevant feature, if an incoming feature X is irrelevant to C, X is discarded at the line 7 test.

For a redundant feature, two situations to be considered include (1) the size of the streaming feature set is finite; and (2) the size of the streaming feature set is infinite. When the feature size is finite, OSFS searches a subset S on line 15, based on Proposition 1, for each feature within BCF to evaluate its redundancy with respect to C. For example, a redundant feature X is added into BCF as a relevant feature on line 7. If training instances are sufficient and most of the features are irrelevant and redundant features, OSFS can keep the size of BCF reasonably small so that conditioning on all subsets within BCF is computationally feasible. As features flow in one by one over time, the subset S within BCF must be found to make X redundant and remove it from BCF at the test on line 15, but if we limit the size of the maximum conditioning subset of BCF to k, OSFS might return a type I error (false positives will enter BCF) when k is not big enough to make X and C conditionally independent.

When the size of the streaming feature set is infinite, OSFS may also fail to remove X during the independence tests on line 15. In this case, false positive errors may occur which results in redundant features to enter into BCF. Because a feature could arrive randomly, OSFS does not know in advance when the subset S for X can be found within BCF; therefore, the actual time for OSFS to remove X is unknown. Assuming that all tests are reliable, if X is really a redundant feature, a subset S must exist to satisfy the term $\mathrm{Ind}(C, X|S)$ as features flow in one by one over time.

In summary, on the assumption that all independence tests are reliable, OSFS can control the false positive and false negative errors well**.** If the size of the maximum conditioning subset is limited to k, the selection of the k value is crucial. Our empirical studies show that setting k to 3 yields satisfactory results. Since the G^2 test focuses on the independence tests of discrete distribution, Fisher's z-test is an alternative and reliable test [28] to assess the conditional independence tests of continuous data (refer to reference [21], pp. 611-615 for detailed explanations about Fisher's z-test).

Time complexity of OSFS: The time complexity of OSFS depends on the number of tests. Assuming |M| is the number of features that have arrived so far, the worst-case complexity is $O(|M||BCF|k^{|BCF|})$ where k is the maximum size to which a conditioning set may grow and $k^{|BCF|}$ denotes the total number of subsets needs to be checked in BCF (i.e., all subsets whose sizes do not exceed k). Assuming |SF| contains the number of all relevant features in |M|, the average time complexity is $O(|SF||BCF|k^{|BCF|})$. It is clear that the time complexity of OSFS is determined by the number of features within BCF, and is independent of the total number of features and training instances. Thus, OSFS is very time-efficient if only a small number of features in a large feature space is predictive and relevant to C, which is the case in many real-world applications. Meanwhile, OSFS is also memory-efficient, because it only needs to store a small number of relevant features at any time by adding and discarding features online.

B. Fast-OSFS Algorithm

As we have discussed, the most time-consuming part of OSFS is the redundancy analysis phase. When OSFS includes an incoming feature to BCF, its redundancy analysis phase will re-examine the redundancy of each feature of BCF. If the size of BCF is large, this process will significantly reduce the runtime performance of OSFS.

To improve selection efficiency, the process of handling redundant features can be divided into two steps: (1) determining whether an incoming new feature is redundant, and (2) identifying which of the selected features observed so far may become redundant once the inclusion of the new feature occurs. Based on Definition 7, we propose to use Proposition 2 to identify whether a newly arrived feature is redundant.

Proposition 2 As features flow in one by one over time, a current Markov blanket of C at time t is denoted by $\mathrm{CMB}(C)_t$. At time t+1, a new feature X_i streams in and is relevant to C. If $\exists S \subseteq \mathrm{CMB}(C)_t \ s.t.\ P(C|X_i, S) = P(C|S)$, then X_i is redundant can be discarded.

To test which features in $\mathrm{CMB}(C)_t$ might become redundant due to the inclusion of X_i, we propose Proposition 3 as follows.

Proposition 3 As the features continuously flow in, for any given time point t with $\mathrm{CMB}(C)_t$, when a new feature X_i arrives at time t+1, if there is no $\mathrm{MB}(X_i)$ within $\mathrm{CMB}(C)_t$ and the following condition applies, $Y \in \mathrm{CMB}(C)_t$, $\exists S \subseteq \{\mathrm{CMB}(C)_t \cup X_i\} - \{Y\} \ s.t. \ P(C|Y, S) = P(C|S)$, then Y can be removed from $\mathrm{CMB}(C)_t$.

Proposition 3 explains that as an incoming feature is added into BCF by Proposition 2, only the subsets created by the inclusion of this new feature need to be tested to check the redundancy of the other features in BCF. Clearly, propositions 2 and 3 also satisfy the property of Corollary 1. Therefore, with the above two Propositions, an accelerated OSFS algorithm named Fast-OSFS is proposed in Figure 4.

Fast-OSFS accelerates OSFS by dividing the online redundancy analysis in OSFS into two parts. The first part (lines 10-18) is a redundancy analysis which aims to remove a new relevant but redundant feature, say X_i, from inclusion in BCF. If X_i is successfully removed, Fast-OSFS directly deals with the next incoming feature. Otherwise, if X_i is not eliminated in the first part, the second part (lines 19-28) is triggered, which adds the new feature X_i into BCF, validates each originally existing feature in BCF, and checks whether any of these features has become redundant after the inclusion of new feature X_i into BCF. In the second part, Fast-OSFS reduces the computational cost of conditional independence tests by only considering the subsets within BCF that contain the newly added feature instead of all subsets within BCF.

The Fast-OSFS algorithm

```
1.  BCF ={};                                   16.  endif
2.  repeat                                     17.  /*Add X to BCF */
3.     added=0;                                18.  BCF = BCF ∪ X;
4.  /*Stream in a new feature*/                19.  /*Redundancy analysis 2: */
5.    X←get_new_feature()                      20.  /*for each feature within BCF*/
6.  /*online relevance analysis */            21.  for each feature Y ∈ BCF-X
7.     if Dep(C, X|∅)                          22.  /*Find S ⊆ BCF containing X*/
8.           added=1;                          23.  if ∃S ⊆ BCF-Y s.t. Ind(C, Y|S)
9.     endif                                   24.  /*Remove Y from BCF */
10. /*Redundancy analysis 1:*/                 25.       BCF = BCF-Y;
11. /* for X */                                26.    endif
12. if (added)                                 27.  endfor
13.    if ∃S ⊆ BCF s.t. Ind(C, X|S)            28.  endif
14. /*Discard X */                            29.  until a predefined accuracy satisfied
15.     go  to Step 2                          30.  output BCF
```

Figure 4: The Fast-OSFS algorithm

The time complexity of Fast-OSFS is as follows. Assuming |M| is the number of features that have arrived so far and |SF| contains the number of all relevant features in |M|, the best time complexity of Fast-OSFS is $O(|SF|k^{|BCF|})$ if Fast-OSFS does not perform the second part for all $|SF|$ arrived features. If the second part is performed for $|SF_1|$ features in SF, the time complexity is $O(|SF - SF_1|k^{|BCF|} + |SF_1||BCF|k^{|BCF|^*})$ where $k^{|BCF|^*}$only considers those subsets in BCF that contain the newly added feature, and if the sizes of the subsets do not exceed k. $k^{|BCF|}$ denotes all subsets in BCF whose sizes are less than or equal to k. If the second part is performed for all features in SF, the worst-case time complexity is $O(|SF||BCF|k^{|BCF|^*})$. When compared with OSFS, it is easy to conclude that for all three conditions above, Fast-OSFS is faster than OSFS.

V. EXPERIMENTAL STUDIES

To compare the performance of the proposed OSFS and Fast-OSFS algorithms with existing streaming feature selection methods, we use high dimensional datasets as our test-beds, by observing features one by one to simulate the situation of streaming features. Table 1 summarizes the 16 high-dimensional datasets used in our experiments.

In Table 1, the ovarian-cancer and breast-cancer datasets are biomedical datasets [8, 32]. The *lymphoma* and *sido0* datasets are from [25] and the WCCI 2008 Performance Prediction Challenges. The *madelon*, *arcene*, *dexter*, and *dorothea* datasets are from the NIPS 2003 feature selection challenge. The *colon*, *prostate*, *leukemia*, and *lung-cancer* datasets are four frequently studied public microarray datasets [35]. The *ohsumed* and *apcj-etiology* are two massive high-dimensional text datasets [17, 4].

Table 1: Summary of the benchmark datasets.
#: the number of features, SIZE: the number of instances

Dataset	#	SIZE		Dataset	#	SIZE
bankruptcy	147	7063		leukemia	7129	72
sylva	216	14374		prostate	6033	102
madelon	500	2000		lung-cancer	12533	181
arcene	10000	100		breast-cancer	17816	286
dexter	20000	300		ovarian-cancer	2190	216
dorothea	100000	800		sido0	4932	12678
lymphoma	7399	227		ohsumed	14373	5000
colon	2000	62		apcj-etiology	28228	15779

For the four NIPS 2003 challenge datasets, we use their original training and validation sets, and for the remaining twelve datasets we use 10-fold cross-validation in our experiments. Two measurements for evaluating our algorithms with Grafting and Alpha-investing are compactness (the proportion or number of selected features) and prediction accuracy (the percentage of the correctly classified test instances which were previously unseen). We use two classifiers, *K*nn and J48 in Spider toolbox, and report the average prediction accuracy in the experiments. The results were collected on a DELL workstation with Windows 7, 2.9GHz CPU and 12GB memory. Grafting and Alpha-investing were performed using their original implementations. The tuning parameter λ for Grafting was selected using cross-validation and the parameters of Alpha-investing were set using its default settings, W_0=0.5 and α_Δ=0.5. For all 16 datasets in Table 1, the independence tests are G^2 tests with the statistical significance level, α, being set to 0.05. For the impact crater case study dataset, we use Fisher's z-tests.

A. Comparing OSFS to Grafting and Alpha-investing

Figure 5 reports the performance of OSFS with Grafting with respect to the prediction accuracy (the y-axis to the left) and the compactness in terms of the number of selected features (the y-axis to the right). From the top two curves in Figure 5, we can see that the prediction accuracy of OSFS is superior to Grafting on 13 out of 16 datasets. For the remaining 3 datasets, *arcene*, *ovarian-cancer*, and *leukemia* (corresponding labels in the x-axis are 1, 5, and 9), the accuracy of OSFS is inferior to Grafting, whereas the two curves at the bottom show that Grafting selects more features than OSFS on those three datasets. For most datasets, we can conclude that OSFS selects fewer features than Grafting. Although for the *dexter*, *madelon*, and the last three datasets, *ohsumd*, *bankruptcy*, and *sylva*, Grafting selects fewer features than OSFS, the accuracy of Grafting on those five datasets is lower than OSFS. It is worth noting that Grafting uses a gradient-based heuristic, which is computationally inefficient, to identify whether a new feature is predictive in each iteration. The large number of features in *dorothea* make the runtime of Grafting increase dramatically and eventually fail in our experiments.

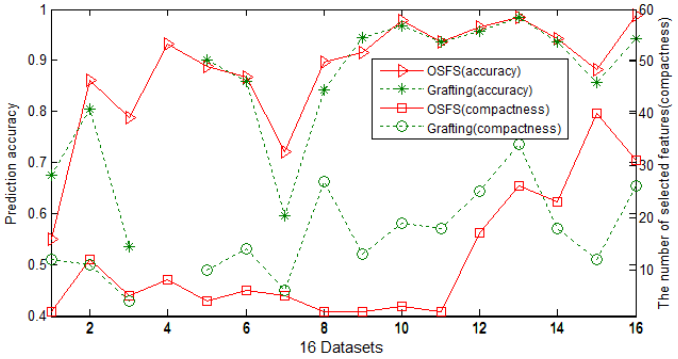

Figure 5: The prediction accuracy (top two figures) and number of selected features (bottom two figures) of OSFS vs. Grafting. (The labels of the x-axis from 1 to 16 denote the datasets: 1:arcene; 2:dexter; 3:madelon; 4:dorohthea; 5:ovarian-cancer; 6:breast-cancer; 7: lymphoma; 8: colon; 9:leukemia; 10:lung-cancer; 11:prostate; 12:sido0; 13: apcj-etiology; 14:ohsumed; 15:bankruptcy; 16:sylva.)

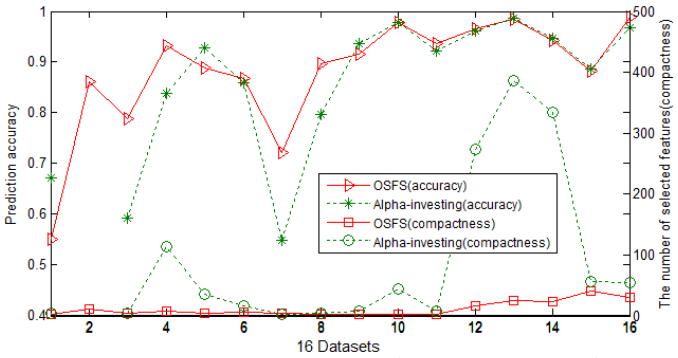

Figure 6: The prediction accuracy (top two figures) and number of selected features (bottom two figures) of OSFS and Alpha-investing. (The labels of the x-axis are the same as the labels of the x-axis in Figure 5.)

In Figure 6, we report the comparisons between OSFS and Alpha-investing, which show that OSFS wins 10 times and ties 3 times on prediction accuracy compared to Alpha-investing. The results show that OSFS selects far fewer features than Alpha-investing on most of datasets datasets (see the bottom two figures). On the third dataset, *madelon*, and the seventh dataset, *lymphoma*, Alpha-investing selects fewer features than OSFS, the accuracy of Alpha-investing on those two datasets is significantly lower than OSFS. Moreover, Alpha-investing fails to select any features on the *dexter* dataset because it is a very sparse dataset.

B. Comparing Fast-OSFS to Grafting and Alpha-investing

Figure 7 reports the prediction accuracy (the two curves at the top) and compactness (the two curves at the bottom) comparisons between Fast-OSFS and Grafting. The results show that Fast-OSFS wins over Grafting 14 times and only loses twice against Grafting on prediction accuracy. Meanwhile, in the bottom two figures, Fast-OSFS loses 8 times on compactness compared to Grafting, but Fast-OSFS has higher accuracy than Grafting on those eight datasets.

In Figure 8, in comparison with Alpha-investing, Fast-OSFS also only loses on two datasets with respect to prediction accuracy. Although on the third dataset, *madelon*, and the seventh dataset, *lymphoma*, Alpha-investing selects fewer features than Fast-OSFS, the accuracy of Alpha-investing on those two datasets is significantly lower than Fast-OSFS. Moreover, on the massive and high-dimensional datasets, such as *sido0*, *apcj-etiology*, and *ohsumed*, Fast-OSFS selects far fewer features than Alpha-investing, but it achieves the same, or higher prediction accuracy than Alpha-investing.

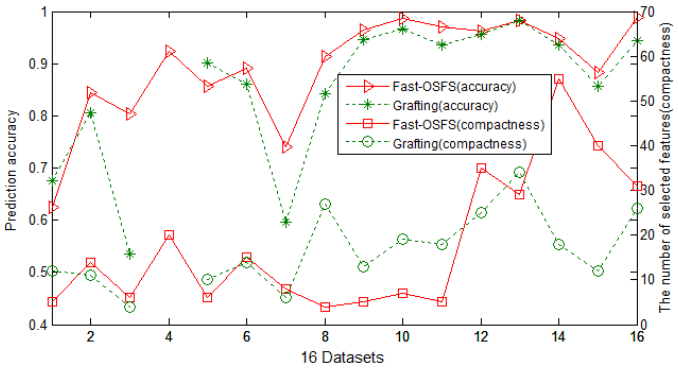

Figure 7: The prediction accuracy (top two figures) and number of selected features (bottom two figures) of Fast-OSFS and Grafting. (The labels of the x-axis are the same as the labels of the x-axis in Figure 5.)

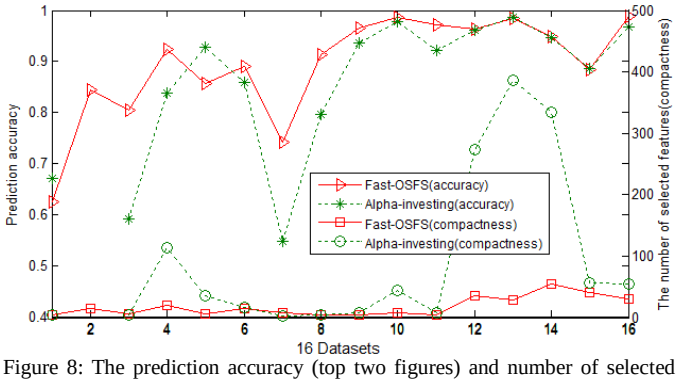

Figure 8: The prediction accuracy (top two figures) and number of selected features (bottom two figures) of Fast-OSFS and Alpha-investing. (The labels of the x-axis are the same as the labels of the x-axis in Figure 5.)

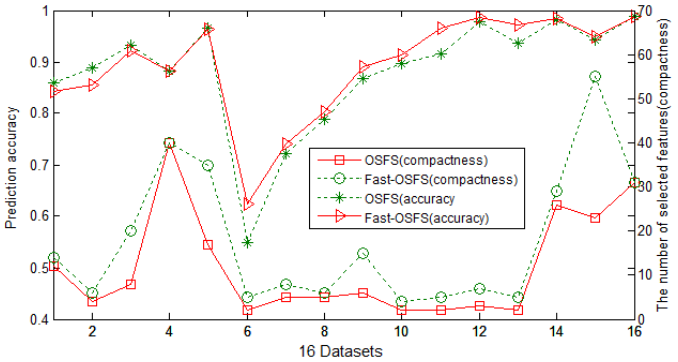

Figure 9: The prediction accuracy (top two figures) and number of selected features (bottom two figures) of Fast-OSFS and OSFS. (The labels of the x-axis from 1 to 16 denote the datasets: 1:dexter; 2:ovarian-cancer; 3:dorothea; 4:bankruptcy; 5:sido0; 6:arcene; 7:lymphoma; 8:madelon; 9:breast-cancer; 10:colon; 11:leukemia; 12:lung-cancer; 13:prostate; 14: apcj-etiology; 15: ohsumed; 16:sylva.)

In our experiments, we find that Alpha-investing, similar to OSFS, also shows slightly better accuracy than Fast-OSFS on the *arcene* and *ovarian-cancer* datasets. A possible explanation is that for datasets with a very small sample-to-variable ratio, this could aggravate the number of unreliable tests if the number of features within BCF increases over time.

C. Comparisons between OSFS and Fast-OSFS

The comparisons between Fast-OSFS and OSFS in Figure 9 show that Fast-OSFS has higher accuracy than OSFS on 13 out of the 16 datasets, although Fast-OSFS selects more features than OSFS, which demonstrates that Fast-OSFS significantly improves the performance of OSFS. Both OSFS and Fast-OSFS perform multiple statistical comparisons to assess whether a feature is redundant. In the redundancy analysis phase, OSFS needs to evaluate each feature within BCF without first considering whether a newly added feature is redundant. It significantly increases the system runtime when the number of features within BCF is large which, in turn, reduces the test of statistical power. Fast-OSFS, on the other hand, significantly reduces the total number of tests because it first examines the redundancy of a new feature and checks only the subsets of BCF containing the feature newly added into BCF.

Table 2 reports the accelerate ratio which is the number of tests performed by OSFS divided by the number of tests performed by Fast-OSFS on the same dataset (the alpha value is fixed at 0.05). An accelerate ratio value greater than one indicates that Fast-OSFS performs fewer tests than OSFS. In Table 2, Fast-OSFS involves far fewer tests than OSFS, thus, Fast-OSFS has stronger statistical power than OSFS.

Table 2: The ratio of conditional independence tests (OSFS/Fast-OSFS)

Dataset	Accelerate Ratio	Dataset	Accelerate Ratio
dexter	31.82	lymphoma	3.41
dorothea	35.96	breast-cancer	10.67
arcene	3.50	ovarian-cancer	15.50
madelon	3.34	sylva	13.79
colon	2.08	bankruptcy	14.17
prostate	3.26	sido0	23.81
lung-cancer	4.88	apcj-etiology	159.91
leukemia	2.54	ohsumed	107.50

D. Runtime Analysis

D.1 A summary of runtime of OSFS and Fast-OSFS

In addition to the theoretical analysis of the time complexity for OSFS and Fast-OSFS, a summary of the runtime performance for OSFS and Fast-OSFS is reported in Tables 3 and 4. Because the runtime of OSFS and Fast-OSFS is significantly influenced by the size of BCF, we report the runtime of both algorithms with the alpha value up to 0.01 (Table 3) and up to 0.05 (Table 4). The results in Tables 3 and 4 show that Fast-OSFS is much faster than OSFS on all datasets. When only a small number of features are predictive, the proposed OSFS and Fast-OSFS algorithms are very efficient, even if a dataset has hundreds of thousands of features, as is the case for the *lung-cancer* and *dexter* datasets. For a dataset with a large number of predictive features, such as *sido0*, *apcj-etiology*, and *ohsumed* (the number of predictive features is shown in the bottom figures of Figure 9), OSFS is time-consuming, even for the *bankruptcy* and *sylva* datasets which contain no more than 300 features. In addition, we observe that the runtime of OSFS is linear to the number of total features, but exponential to the number of features which are predictive.

As we have analyzed in Section IV.B, Fast-OSFS alleviates this problem by significantly reducing the number of tests (as shown in Table 2). For example, from the bottom figures of Figure 9, we can see that there is a large number of predictive features on *bankruptcy*, *sido0*, *apcj-etiology*, and *ohsumed*, but Fast-OSFS is still quite efficient, as shown in Tables 3 and 4.

Table 3: Runtime performance (in seconds) of OSFS and Fast-OSFS (alpha=0.01). (A/B in the second column denotes the runtime of OSFS, i.e., A, vs. the runtime of Fast-OSFS, i.e., B)

Dataset	Runtime		Dataset	Runtime
dexter	4/**1**		lymphoma	0/0
dorothea	64/**34**		breast-cancer	20/**4**
arcene	0/0		ovarian-cancer	1/0
madelon	0/0		sylva	1892/**170**
colon	0/0		bankruptcy	1272/**127**
prostate	0/0		sido0	10085/**410**
lung-cancer	6/**1**		apcj-etiology	11141/**139**
leukemia	0/0		ohsumed	2851/**66**

Table 4: Runtime performance (in seconds) of OSFS and Fast-OSFS (alpha=0.05). (A/B in the second column denotes the runtime of OSFS, i.e., A, vs. the runtime of Fast-OSFS, i.e., B)

Dataset	Runtime		Dataset	Runtime
dexter	38/**2**		lymphoma	2/**1**
dorothea	1988/**78**		breast-cancer	97/**9**
arcene	1/0		ovarian-cancer	4/0
madelon	0/0		sylva	4807/**348**
colon	0/0		bankruptcy	3645/**261**
prostate	1/0		sido0	42789/**2014**
lung-cancer	10/**2**		apcj-etiology	118329/**676**
leukemia	0/0		ohsumed	156271/**1103**

D.2 Runtime Analysis for Grafting and Alpha-investing

Since the Grafting and Alpha-investing algorithms used in the experiments are both implemented in *MATLAB* by the authors and our algorithms are written in C language, a direct time comparison between the baselines and our algorithms is inappropriate. Instead, we investigate an analysis as follows. Grafting recasts feature subset selection as optimizing the L1-regularized maximum likelihood estimation by iteratively performing two steps: optimizing over all the free parameters and selecting a new feature. Grafting needs a regularization parameter—λ in advance to determine which feature is most likely to be selected to the model at each iteration. It is understandable that choosing a suitable regularization parameter requires the entire feature set information in advance, but the full feature set information is unknown in advance in streaming features. Moreover, finding an optimum value for a free parameter at each step is usually a computationally expensive step. Because Alpha-investing and our proposed two algorithms do not need to determine any prior parameters in advance, we focus on the comparison between Alpha-investing and Fast-OSFS.

The Alpha-investing algorithm uses linear and logistic regression to dynamically adjust the threshold on the error reduction required for evaluating a new feature that is added to the predictive model. Alpha-investing only considers whether to add a new feature and never considers the discard of the selected features or adding discarded features again. Therefore, when a new feature X arrives, even if X is irrelevant, Alpha-inverting adds it into the current model, that is, $\{model \cup X\}$, and then uses linear regression to compute

the error reduction between $\{model \cup X\}$ and $\{model\}$ to decide whether to drop X. At the arrival of each new feature, the time complexity of Alpha-investing is $O(|V||model|^2)$ where |V| is the total number of features arrived so far and |model| is the number of features selected from V in the current model.

Table 5: Runtime performance (in seconds) of Alpha-investing and Fast-OSFS (A/B in the second and last columns denotes the runtime of Alpha-investing, i.e., A, vs. the runtime of Fast-OSFS, i.e., B).

alpha=0.01			alpha=0.05	
Dataset	**Runtime**		**Dataset**	**Runtime**
dorothea	993/34		dorothea	993/78
sido0	929/410		sido0	929/2014
apcj-etiology	7305/139		apcj-etiology	7305/676
ohsumed	794/66		ohsumed	794/1103

When the feature set size is not large and the number of features in the current model is small, Alpha-investing is very time efficient. However, when the size of the streaming feature set is huge and the number of features within the current model is large, Alpha-investing is not computationally efficient. Although Alpha-investing is implemented in *MATLAB*, we still show the runtime of Alpha-inverting and Fast-OSFS in Table 5. We can see that when the alpha value goes up to 0.01, Fast-OSFS is computationally efficient whereas Alpha-investing is not. When the alpha value is around 0.05, Fast-OSFS is still very fast on the *dorothea* and *apcj-etiology* datasets.

E. Handling datasets with an unknown feature space

In the above experiments, the streaming features are simulated using datasets with known feature sizes. In this subsection, we study the performance of OSFS and Fast-OSFS under the situation where the entire feature set of a dataset is unknown in advance. To demonstrate the performance of the algorithms, we use prediction accuracy as a criterion to explore the streaming feature selection process; four NIPS2003 feature selection challenge datasets, *madelon*, *arcene*, *dexter*, and *dorothea*, and four gene datasets, *colon*, *prostate*, *leukemia*, and *lung-cancer* in Table 1 are used for these evaluation studies.

We use the original training and validation sets for the NIPS 2003 challenge datasets. Because the colon and leukemia gene expression datasets have a small number of instances, we randomly select 10 instances as the test instances (5 positive and 5 negative instances) and the rest of the instances are used for training. For the remaining two gene expression datasets, we randomly select, using cross-validation, the first 2/3 instances for training and the remaining 1/3 instances are used for testing. *K*nn is used as a baseline classifier on the training and testing sets with all features. With this baseline, our two algorithms and Alpha-investing use *K*nn to dynamically evaluate streaming feature selection on the testing instances.

In Figure 10, we report the change of the prediction accuracies of three algorithms on *K*nn with respect to the features continuously arriving over time. The red horizontal line in each figure denotes the prediction accuracy of the baseline *K*nn classifier trained using all features. Because Alpha-investing fails to select any features on *dexter*, its accuracy is omitted in the figure.

The results in Figure 10 demonstrate that a comparison of Fast-OSFS with OSFS and Alpha-investing shows that Fast-OSFS is more stable and has better prediction accuracy.

Among the four NIPS 2003 datasets, the prediction accuracies of all of three algorithms on the *arcene* dataset fall below the accuracy of the baseline classifier. Alpha-investing stops selecting any features with the percentage of features up to 10%, while Fast-OSFS continues the feature selection till the feature percentage increases to 60% and then stops (denoted by the straight line after the selected features are 60% or higher). For the *dexter* dataset, Fast-OSFS and OSFS reach the baseline accuracy as the feature percentage increases to 70%. For the *madelon* and *dorothea* datasets, both OSFS and Fast-OSFS exceed the baseline accuracy as features stream in, while Alpha-investing falls behind. Among the four gene expression datasets, Fast-OSFS is able to reach or outperform the baseline without an exhaustive search over the entire feature set, with the exception of the leukemia dataset.

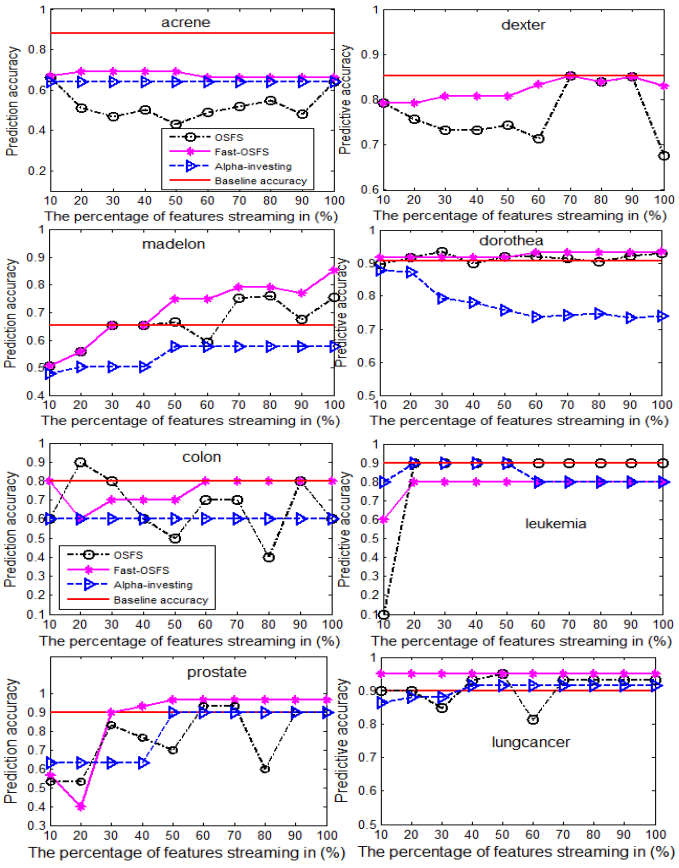

Figure 10: The prediction accuracy changes with respect to the number of features streaming in

The above observations conclude that when the underlying feature space is unknown or significantly large, it is unnecessary to exhaustively search over the entire feature space. Compared to Alpha-investing and OSFS, Fast-OSFS achieves better and more stable performance in terms of the prediction accuracy of the models trained from selected streaming features.

F. Conclusions of the Experimental Results

The above experiments and comparisons conclude that OSFS *and Fast-OSFS* outperform both Grafting and Alpha-investing on most of the datasets (in terms of the prediction accuracy and the compactness of the selected features). When handling streaming features, the main drawback of Grafting is that it needs to tune parameter λ in advance while Alpha-investing cannot properly handle the original features without any prior information about the feature structure. Since Alpha-investing only considers each feature once, its running speed is very fast, but this also causes Alpha-investing to select more features than other methods (including our algorithms). In addition, Alpha-investing cannot discard redundant features from the current model to deal with the situation that some features may have been useful in the past but have become redundant or irrelevant to the target concept as time goes by.

In personalized news filtering, for example, users' interests constantly change so that new words may become useful whereas previously selected words may become outdated and redundant. Grafting and the proposed OSFS and Fast-OSFS algorithms can effectively handle this problem.

With the prior knowledge about the structure of the feature space, Alpha-investing is fast and achieves good performance since the prior knowledge helps the algorithm to heuristically control the selection of candidate features. With prior knowledge, our framework also performs well; with domain knowledge, for example, the corresponding redundant features can be removed earlier because it is easier for our algorithms to find strongly relevant and non-redundant features if informative features are placed earlier in the streaming features. For a strongly relevant feature, say Y, and its copies, say Y_1 and Y_2, which carry exactly the same predictive information about the class attribute C, the incoming order of these features does not matter. This is because both our algorithms and Alpha-investing can select any one feature from Y and its copies Y_1 and Y_2. As soon as one of the strongly relevant features, Y, Y_1, or Y_2, is selected, the remaining features will be excluded from the succeeding streaming feature selection process.

In reality, features are rarely identical but may be strongly correlated; thus, given a feature A which is relevant to the class attribute C, the order of features A and B might matter if feature B is redundant (but not identical) to A. Under such circumstances, our algorithms and Alpha-investing might select a different set of features depending on the actual order of the features. To evaluate the impact of feature order on the algorithm's performance, we generate a number of trials in which each trial represents a random ordering of features as a feature stream. We apply different algorithms (OSFS, Fast-OSFS, and Alpha-investing) to each randomized trial and report the results in Figures 11-13, where the x-axis represents each of the randomized trials and the y-axis represents the number of selected features (Figure 11) and the prediction accuracies from the corresponding trial.

The results in Figures 11-13 (the *madelon* dataset) confirm that varying the order of the incoming features does impact on the final outcomes. Overall, the results demonstrate that Fast-OSFS is the most stable method and Alpha-investing appears to be highly unstable.

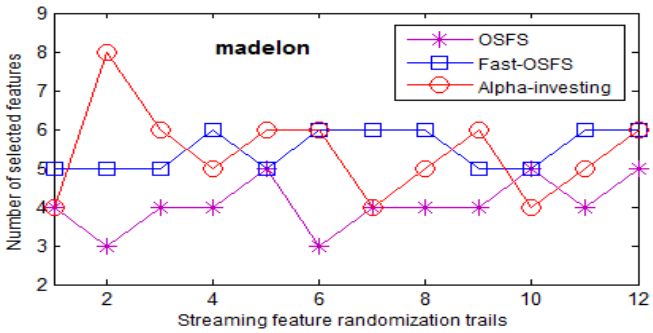

Figure 11: Numbers of selected features from 12 randomized trials (each trial represents a random ordering of features as a feature stream)

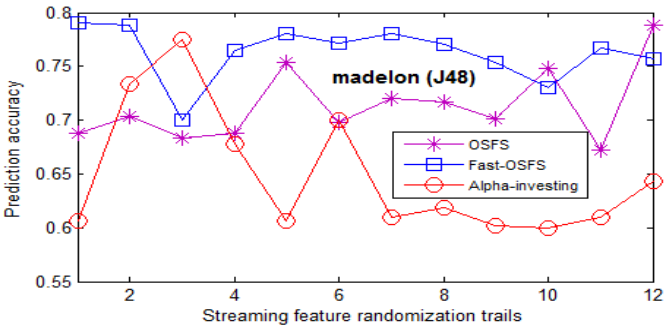

Figure 12: Prediction accuracies from 12 randomized trials using decision tree learning algorithms (J48)

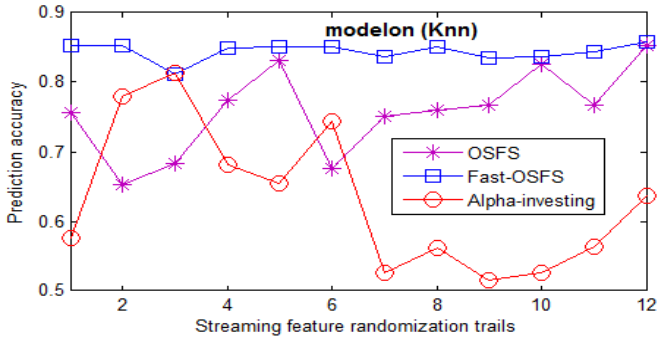

Figure 13: Prediction accuracies from 12 randomized trials using Knn learning algorithms

As for the proposed OSFS and Fast-OSFS algorithms, Fast-OSFS significantly accelerates OSFS by employing a new redundancy analysis strategy. The accelerated feature selection strategy in Fast-OSFS might introduce additional false positive features into BCF, which explains why OSFS always selects fewer features than Fast-OSFS (as shown in Figure 11). With sufficient instances and a small size of BCF, OSFS achieves almost the same prediction accuracy and runtime as Fast-OSFS, but it results in a smaller number of selected features. On the other hand, with sufficient instances and a large size of BCF, Fast-OSFS is much faster than OSFS. With an insufficient number of instances and a large size BCF, Fast-OSFS is superior to OSFS because Fast-OSFS significantly mitigates the false negative errors by reducing the number of tests involved in checking feature redundancy.

G. A Case Study on Automatic Impact Crater Detection

In addition to the validation on the publicly available benchmark datasets, we also use a real-world impact crater dataset to evaluate our streaming feature selection algorithms. Impact craters, the structures formed by the collisions of meteoroids on planetary surfaces, are among the most studied geomorphic features in the solar system because they yield information about past and present geological processes. Surveying craters provides the only tool for remotely measuring the relative ages of geologic formations.

Planetary probes deliver ever-increasing volumes of high resolution images; however, the scientific utilization of these images in ever-higher spatial resolution is hampered by the lack of tools for their effective automated analysis. Texture features have proven to be effective for crater detection. Tens of thousands of texture-based features in different scales and resolutions can be generated for crater detection on remotely sensed images which provide an extensive near-global coverage of a remote planet, such as Mars. While rich texture features provide a tremendous source of potential features for use in crater detection tasks, they are expensive to generate and store. The reality calls for efficient feature selection to develop a processing pipeline for fast and accurate surveys of craters from high resolution images and make possible the assembly of global "million crater" catalogs of craters, not only on Mars, but also on Mercury, the Moon, and other planets. Consequently, this makes an ideal case study for validating our streaming selection framework, compared to traditional feature selection approaches.

In this case study, our work is based on the crater detection framework proposed by Ding et al. (Figure 14). There are three steps in the crater-detection framework [11].

(1) Crater candidates are the regions of an image that may potentially contain craters and the image can be collected using remote sensing techniques. A key insight to constructing crater candidates is that a sub-kilometer crater can be recognized as a pair of crescent-like highlight and shadow regions in an image (see Figure 15 [11]). Crescent-like shadow and highlight regions in an image are identified from images using a shape detection method based on mathematical morphology, and those highlight and shadow regions are matched so that each pair will be used to construct crater candidates, that is, the locations where craters are likely to reside.

(2) Image texture features are extracted from crater candidates using square kernels.

(3) Craters are identified using supervised learning algorithms.

The experiments in crater detection are evaluated on Mars because it is at the center of NASA exploration efforts. There is a very extensive, near-global coverage of the Martian surface with high resolution planetary images. A portion of the High Resolution Stereo Camera (HRSC) nadir panchromatic image h0905 is selected, taken by the Mars Express spacecraft, to serve as the test set [11]. The selected image has a resolution of 12.5 meters/pixel and a size of 3,000 by 4,500 pixels (37,500×56,250m^2). The image represents a significant challenge to automatic crater detection algorithms because it covers a terrain that has spatially variable morphology and because its contrast is rather poor (mostly noticeable when the image is inspected at a small spatial scale).

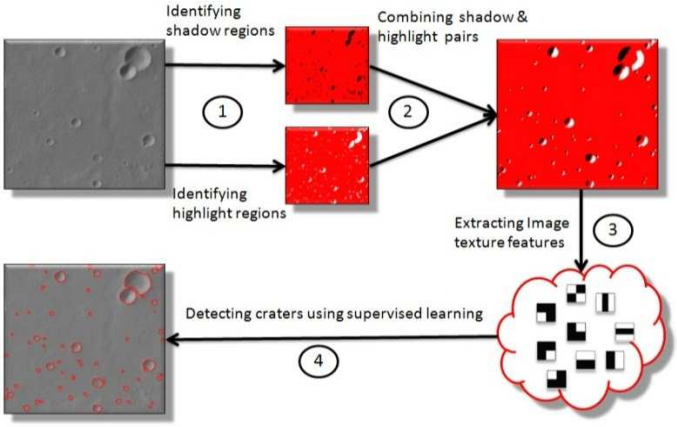

Figure 14: The crater-detection framework proposed by Ding et al. [11]

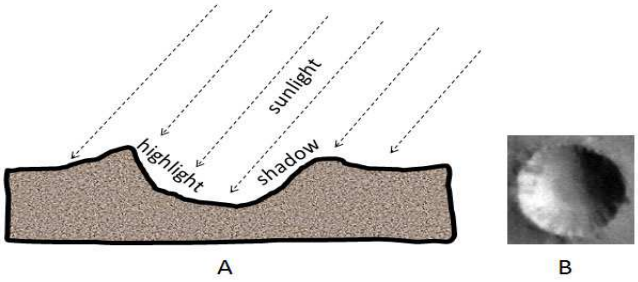

Figure 15: (A) an illustration explaining why an image of a sub-kilometer crater consists of crescent-like highlight and shadow regions. (B) An image of an actual 1 *km* crater showing the highlight and shadow regions.

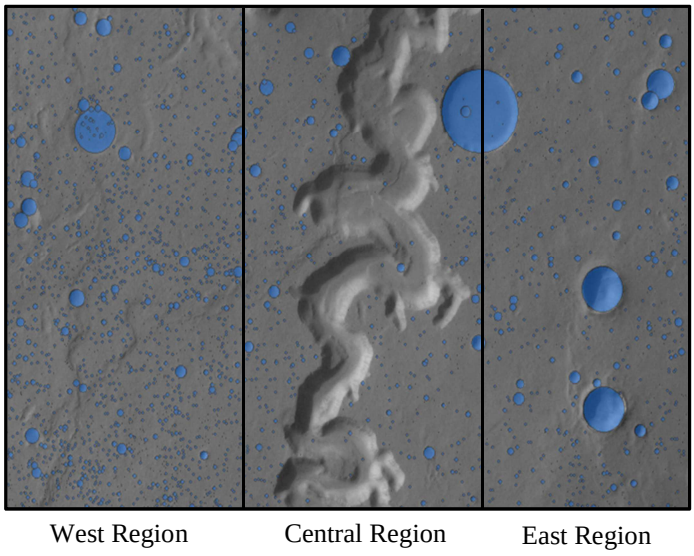

Figure 16: Impact craters in a 37,500×56,250 m^2 test image from Mars.

The image is divided into three sections denoted as the west region, the central region, and the east region (see Figure 16 [11]) for the test sets summarized in Table 6. The central region is characterized by surface morphology that is distinct from the rest of the image. The west and east regions have similar morphology but the west region is much more heavily

cratered than the east region. 1,089 image texture features are constructed. The training set consists of 204 true craters and 292 non-crater examples selected randomly from crater candidates located in the northern half of the east region. A streaming feature selection framework for the crater detection is given in Figure 17.

Table 6: Summary of crater datasets

	#samples (crater candidates)	#features
West region	6,708	1,089
Central region	2,935	1,089
East region	2,026	1,089

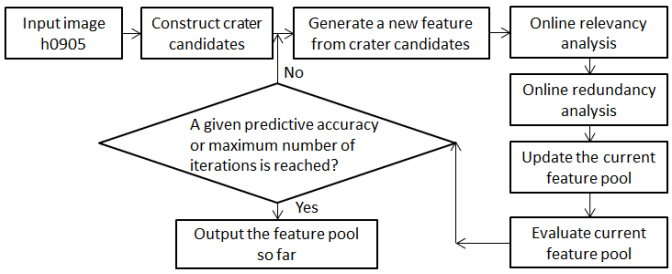

Figure 17: A framework of streaming feature selection for crater detection

In the following sections, we compare our algorithms (OSFS and Fast-OSFS) with Alpha-investing and other state-of-the-art feature selection algorithms. *K*nn is used to train classifiers from each selected feature set, through which we can compare the prediction accuracies of different methods. In order to thoroughly demonstrate the behaviors of our algorithms in the case study, we report the number of selected features and the prediction accuracy with respect to two alpha values (0.01 and 0.05). The best results are bold-faced in the tables.

G.1 Comparisons with Alpha-investing

With the value of alpha up to 0.01, Table 7 reports the prediction accuracy on three regions using our algorithms and Alpha-investing. On the three regions, both our algorithms select the same four features from the training dataset and result in the same prediction accuracy. From Table 7, we can see that our algorithms select fewer features and have higher accuracy than Alpha-investing on the west and central regions, and the accuracies of our algorithms on the east region are also comparable to Alpha-investing.

Table 7: The prediction accuracy on three regions (alpha=0.01)

	#Selected features	West region	Central region	East region
OSFS	4	**0.7753**	**0.7826**	0.7725
Fast-OSFS	4	**0.7753**	**0.7826**	0.7725
Alpha-investing	16	0.7589	0.7666	**0.7730**

With the value of alpha up to 0.05, as shown in Table 8, our algorithms also select fewer features and have higher accuracy than Alpha-investing on all three test regions. On the west region and central region, OSFS has the highest prediction accuracy, while Fast-OSFS has the highest accuracy on the east region.

Table 8: The prediction accuracy on the three regions (alpha=0.05)

	# Selected features	West region	Central region	East region
OSFS	5	**0.7809**	**0.7874**	**0.7828**
Fast-OSFS	5	**0.7809**	**0.7874**	**0.7828**
Alpha-investing	16	0.7589	0.7666	0.7730

Figures 18 and 19 report the performance of OSFS and Alpha-investing with the percentage of the features streaming in (Fast-OSFS has the same performance as OSFS, so its performance is omitted from the figures). Figure 18 shows that the number of selected features changes as more features stream in. We can see that our two algorithms select far fewer features than Alpha-investing at any stage. When the percentage of the features increases to 50%, the number of selected features remains stable for Alpha-investing and OSFS.

Figure 19 illustrates that the test errors of both OSFS and Alpha-investing change over time as the features flow in continuously. The overall results confirm that OSFS is superior to Alpha-investing. The test errors of both algorithms remain stable when the percentage of the total features increases to 50%. This observation validates the rationality of stream feature selection and confirms that instead of trying to smooth across all potential features, we can use a small number of features to train a much stronger model.

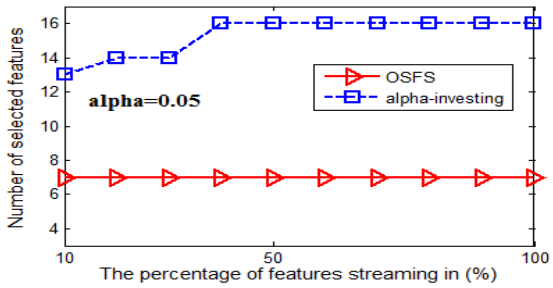

Figure 18: Number of selected features changes as the percentage of the features increases over time

G.2 Comparisons with Traditional Feature Selection Algorithms

In this section, we compare our algorithms with the state-of-the-art non-streaming fashion feature selection algorithms, a causal feature selection algorithm, the LARS (Least Angle Regression) algorithm, a Naïve boosting algorithm and an algorithm without feature selection. Causal feature selection has recently been proposed as an emerging successful filtering approach in feature selection and has shown that it dominates most feature selection methods in prediction accuracy and compactness [3]. The HITON_PC algorithm is selected to instantiate a causal feature selection approach [3]. The LARS algorithm is an embedded feature selection method recently introduced to handle classification or regression problems by using optimization with specified loss and penalty functions. The Naïve boosting algorithm was proposed by Ding et al. [11]; it integrates the boosting algorithm and greedy feature selection algorithms for crater detection.

In Tables 9 and 10, we report the prediction accuracies of all methods on the three regions. With the value of alpha up to

0.01, as shown in Table 9, we can see that our algorithms win on the west region and are very competitive with the LARS and Naïve boosting algorithms on the central and east regions. As shown in Table 10, both our algorithms outperform the other four algorithms on the west and east regions. Although our algorithms lose on the central region, they are also very close to the Naïve boosting algorithm. In summary, compared to the traditional non-streaming fashion feature selection algorithms, our new algorithms select far fewer features and result in higher or at least comparable accuracies as other methods. Most importantly, our algorithms provide a new processing pipeline for streaming based feature selection with fast and accurate surveys of craters from high resolution images.

Table 9: The prediction accuracy on three regions (alpha=0.01)

	#Selected features	West region	Central region	East region
OSFS	4	**0.7753**	0.7826	0.7725
Fast-OSFS	4	**0.7753**	0.7826	0.7725
HITON_PC	4	0.7722	0.7853	0.7636
LARS	6	0.7740	0.7881	**0.7799**
Naïve Boosting	150	0.7661	**0.7888**	0.7749
No feature selection	1089	0.7303	0.7499	0.7710

Table 10: The prediction accuracy on three regions (alpha=0.05)

	#Selected features	West region	Central region	East region
OSFS	7	**0.7809**	0.7874	**0.7828**
Fast-OSFS	7	**0.7809**	0.7874	**0.7828**
HITON_PC	6	0.7749	0.7792	0.7813
LARS	6	0.7740	0.7881	0.7799
Naïve Boost	150	0.7661	**0.7888**	0.7749
No feature selection	1089	0.7303	0.7499	0.7710

Interestingly, the results in Tables 9 and 10 demonstrate that although a river-shaped region (the Nanedi Valles on Mars) runs through the central image, which makes it morphologically different from the original training set, the five feature selection algorithms in Tables 9 and 10 result in slightly better prediction accuracy on the central region than other regions. The reason for this is that we use crater candidates, which are the regions of an image that may potentially contain craters, for crater detection and calculation of prediction accuracy instead of an inefficient, exhaustive search of the entire image. While the river-shaped region appears to make crater detection more difficult, the distinct texture features generated by the crater candidates make them fairly easy to recognize compared to the small crater regions to the east and west. Moreover, although the west and east regions have similar morphology, the west region is much more heavily dense with small craters than the east region. Thus, the prediction accuracy on the west region is slightly lower than the accuracy on the east region.

VI. CONCLUSIONS

In this paper, we have proposed two new algorithms for streaming feature selection. Compared to the two state-of-the-art algorithms, Grafting and Alpha-investing, the proposed algorithms OSFS and Fast-OSFS have demonstrated high efficiency and effectiveness for applications containing many irrelevant and/or redundant features.

In the experiments, our study has shown that in most cases for applications involving streaming or an infinite size of features, a small number of features can be selected to train a much stronger model, rather than trying to smooth across all potential features. We have also applied online streaming feature selection to a real-world Mars impact crater dataset and compared our algorithms with Alpha-investing and other state-of-the-art traditional feature selection algorithms. The experiments have demonstrated that the proposed algorithms select far fewer features than other methods, and their prediction accuracy is mostly higher than, or at least as good as, other methods.

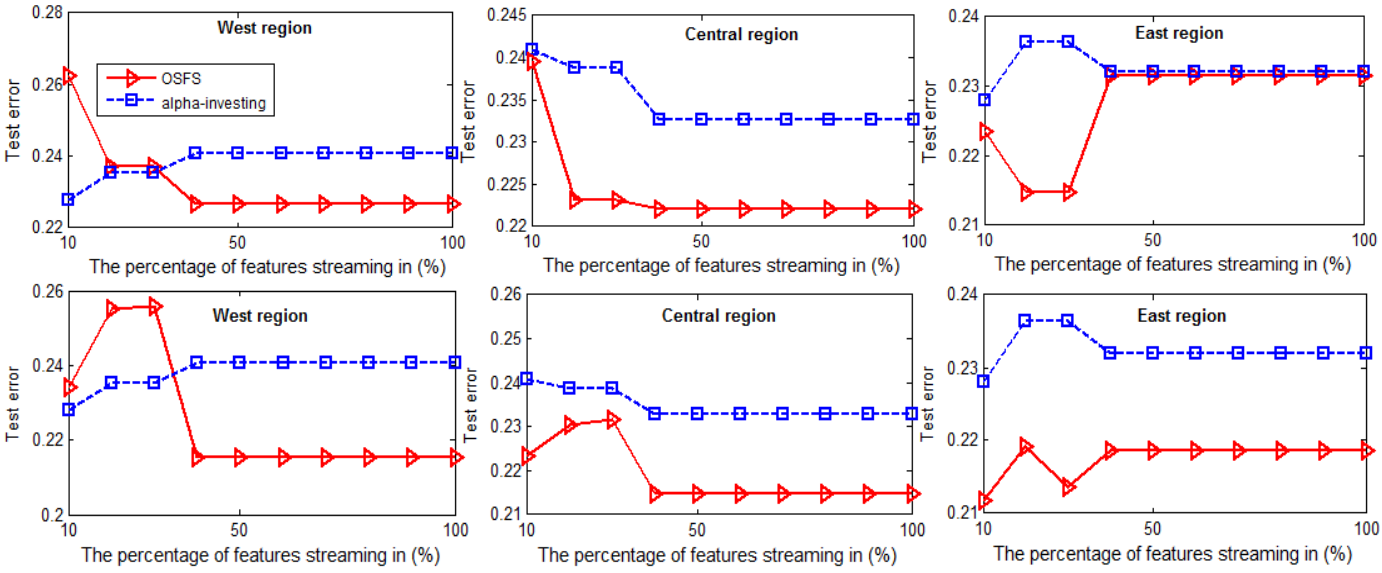

Figure 19: The test errors of three algorithms with respect to the increase of the percentage of features in three regions (In the top figures OSFS using alpha=0.01 and in the bottom three figures OSFS using alpha=0.05).

ACKNOWLEDGEMENTS

This work is supported by the National 863 Program of China (2012AA011005), the National Natural Science Foundation of China (61229301, 61070131, 61175051 and 61005007), the US National Science Foundation (CCF-0905337), and the US NASA Research Award (NNX09AK86G). X. Zhu is sponsored by Australian Research Council (ARC) Future Fellowship (FT100100971). The authors would like to thank the anonymous reviewers for their valuable and constructive comments on improving the paper.

References

[1] A. Agresti. *Categorical Data Analysis*. (1990) New York: John Wiley and Sons.

[2] A. Akadi, A. Amine, A. Ouardighi and D. Aboutajdine. (2011) A two-stage gene selection scheme utilizing MRMR filter and GA wrapper. *Knowledge and Information Systems*, 26(3), 487-500.

[3] C. F. Aliferis, A.Statnikov, I. Tsamardinos , S. Mani and X. Koutsoukos. (2010) Local causal and Markov blanket induction for causal discovery and feature selection for classification, Part I: Algorithms and empirical evaluation. *Journal of Machine Learning Research*, 11:171-234.

[4] Y. Aphinyanaphongs, A. Statnikov and C. F. Aliferis. (2006) A comparison of citation metrics to machine learning filters for the identification of high quality medline documents. *J. Am. Med. Inform. Assoc.*, 13(4):446–455.

[5] G. Brown, A. Pocock, M. Zhao and M. Luj´an. (2012) Conditional likelihood maximisation: a unifying framework for information theoretic feature selection. *Journal of Machine Learning Research*, 13:27-66.

[6] G. Bontempi and P. E. Meyer. (2010) Causal filter selection in microarray data. *ICML*'10, 95-102.

[7] N. Bouguila and D. Ziou. (2011) A countably infinite mixture model for clustering and feature selection. *Knowledge and Information Systems*, 21:1-20.

[8] T. P. Conrads et al. (2004) High-resolution serum proteomic features for ovarian cancer detection. *Endocr. Relat Cancer*, 11:163-178.

[9] A. Cuzzocrea. (2011) Data warehousing and knowledge discovery from sensors and streams. *Knowledge and Information Systems*, 28:491-493.

[10] M. Dash and H. Liu. (2003) Consistency-based search in feature selection. *Artificial Intelligence*, 151(1-2), 155-176.

[11] W. Ding, T. Stepinski, Y. Mu, L. Bandeira, R. Vilalta, Y. Wu, Z. Lu, T. Cao and X. Wu. (2011) Sub-kilometer crater discovery with boosting and transfer learning. *ACM Transactions on Intelligent Systems and Technology*, 2(4), 1-22.

[12] P. S. Dhillon, D. Foster and L. Ungar. (2010) Feature selection using multiple streams. *AISTATS*'10, 153-160.

[13] K. Glocer, D. Eads and J. Theiler. (2005) Online feature selection for pixel classification. *ICML*'05, 249 - 256.

[14] I. Guyon, C.F. Aliferis and A. Elisseeff. (2008) Causal feature selection. In: *Computational methods of feature selection*. H. Liu and H. Motoda Eds. Boca Raton, FL: Chapman and Hall.

[15] I. Guyon and A. Elisseeff. (2003) An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157-1182.

[16] X. He, M. Ji, C. Zhang and H. Bao. (2011) A variance minimization criterion to feature selection using laplacian regularization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(10), 2013-2025.

[17] T. Joachims. (2002) *Learning to classify text using support vector machines*. Boston: Kluwer Academic.

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

[19] D. Koller and M. Sahami. (1996) Toward optimal feature selection. *ICML*'96, 284-292.

[20] H. Malik, D. Fradkin and F. Moerchen. (2011) Single pass text classification by direct feature weighting. *Knowledge and Information Systems*, 28(1), 79-98.

[21] R. Neapolitan. (2003) *Learning Bayesian networks*. Upper Saddle River, NJ: Prentice Hall.

[22] S. Perkins and J. Theiler. (2003) Online feature selection using grafting. *ICML*'03, 592-599.

[23] H. Peng, F. Long and C. Ding. (2005) Feature selection based on mutual information: Criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8):1226–1238.

[24] I. Rodriguez-Lujan, R. Huerta, C. Elkan and C. Santa-Cruz. (2010) Quadratic programming feature selection. *Journal of Machine Learning Research*, 11:1491-1516.

[25] A. Rosenwald et al. (2002) The use of molecular profiling to predict survival after chemotherapy for diffuse large-B-cell lymphoma. *N. Engl. J Med.*, 346, 1937-1947.

[26] M. Shah, M. Marchand, J. Corbeil. (2012) Feature selection with conjunctions of decision stumps and learning from microarray data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(1), 174-186.

[27] L. Song, A. Smola, A. Gretton, J. Bedo and K. Borgwardt. (2012) Feature selection via dependence maximization. *Journal of Machine Learning Research*, 13:1393−1434.

[28] P. Spirtes, C. Glymour and R. Scheines. (2000) *Causation, prediction, and search*, 2nd edition. Cambridge, MA: MIT Press.

[29] R. Tibshirani. (1996) Regression shrinkage and selection via the Lasso. Journal of Royal. Statist. Soc. B. 58, 267–288.

[30] E. Tuv, A. Borisov, G. C. Runger and K.Torkkola. (2009) Feature selection with ensembles, artificial variables, and redundancy Elimination. *Journal of Machine Learning Research*, 10:1341-1366.

[31] L. Ungar, J. Zhou, D. Foster and B. Stine. (2005) Streaming feature selection using IIC. *AI&Statistics*'05, 384-393.

[32] Y. Wang et al. (2005) Gene-expression profiles to predict distant metastasis of lymph-node negative primary breast cancer. *Lancet*, 365, 671-679.

[33] R. Wang, S. Shan, X. Chen, J. Chen and W. Gao. (2011) Maximal linear embedding for dimensionality reduction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(9), 1776–1792.

[34] K. Yu, H. Wang and W. Ding. (2010) Online streaming feature selection. *ICML*'10, 1159-1166.

[35] L. Yu, C. Ding and S. Loscalzo. (2008) Stable feature selection via dense feature groups. *KDD*'08, 803-811.

[36] L. Yu and H. Liu. (2004) Efficient feature selection via analysis of relevance and redundancy. *Journal of Machine Learning Research*, 5: 1205-1224.

[37] P. Zhao and B. Yu. (2006) On model selection consistency of Lasso. *Journal of Machine Learning Research*, 7:2541–2567.

[38] T. Zhang. (2009) On the consistency of feature selection using greedy least squares regression. *Journal of Machine Learning Research*, 10: 555-568.

[39] Z. Zhang and N. Ye. (2011) Locality preserving multimodal discriminative learning for supervised feature selection. *Knowledge and Information Systems*, 27(3), 473-490.

[40] J. Zhou, D. P. Foster, R. Stine and L.H. Ungar. (2005) Streaming feature selection using Alpha-investing. *KDD*'05, 384 -393.

[41] J. Zhou, D. Foster, R.A. Stine and L.H. Ungar. (2006) Streamwise feature selection. *Journal of Machine Learning Research*, 7:1861-1885.

[42] X. Zhu, W. Ding, P. S. Yu and C. Zhang. (2011) One-class learning and concept summarization for data streams. *Knowledge and Information Systems*, 28(3), 523-553.

[43] H. Zeng and Y. Cheung. (2011) Feature selection and kernel learning for local learning-based clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(8), 1532-1547.

Xindong Wu is a Yangtze River Scholar in the School of Computer Science and Information Engineering at the Hefei University of Technology (China), a Professor of Computer Science at the University of Vermont (USA), and a Fellow of the IEEE. He received his Bachelor's and Master's degrees in Computer Science from the Hefei University of Technology, China, and his Ph.D. degree in Artificial Intelligence from the University of Edinburgh, Britain. His research interests include data mining, knowledge-based systems, and Web information exploration.

Dr. Wu is the Steering Committee Chair of the IEEE International Conference on Data Mining (ICDM), the Editor-in-Chief of Knowledge and Information Systems (KAIS, by Springer), and a Series Editor of the Springer Book Series on Advanced Information and Knowledge Processing (AI&KP). He was the Editor-in-Chief of the IEEE Transactions on Knowledge and Data Engineering (TKDE, by the IEEE Computer Society) between 2005 and 2008. He served as Program Committee Chair/Co-Chair for ICDM '03 (the 2003 IEEE International Conference on Data Mining), KDD-07 (the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining), and CIKM 2010 (the 19th ACM Conference on Information and Knowledge Management).

Kui Yu received the MSc degree in Computer Science from the Hefei University of Technology, China, in 2007. He is currently a Ph.D. student in the School of Computer Science and Information Engineering at the Hefei University of Technology (China), and also a visiting Ph.D. student in the Department of Computer Science at the University of Massachusetts Boston (USA). His research interests include feature selection, probabilistic graphical models and machine learning.

Wei Ding received her Ph.D. degree in Computer Science from the University of Houston in 2008. She has been an Assistant Professor of Computer Science in the University of Massachusetts Boston since 2008. Her research interests include data mining, machine learning, artificial intelligence, computational semantics, and with applications to astronomy, geosciences, and environmental sciences. She has published more than 60 referred research papers, 1 book, and has 1 patent. She is an Associate Editor of Knowledge and Information Systems (KAIS) and an editorial board member of the Journal of System Education (JISE). She is the recipient of a Best Paper Award at the 2011 IEEE International Conference on Tools with Artificial Intelligence (ICTAI), a Best Paper Award at the 2010 IEEE International Conference on Cognitive Informatics (ICCI), a Best Poster Presentation award at the 2008 ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (SIGSPAITAL GIS), and a Best PhD Work Award between 2007 and 2010 from the University of Houston. Her research projects are currently sponsored by NASA and DOE.

Hao Wang received his Ph.D. degree in Computer Science from the Hefei University of Technology, China, in 1997. He is a Professor of the School of Computer Science and Information Engineering, Hefei University of Technology, Hefei, China. His research interests include robotics, artificial intelligence, data mining, probabilistic graphical models and machine learning.

Xingquan Zhu received his Ph.D. degree in Computer Science from Fudan University, Shanghai, China, in 2001. He is a recipient of the Australian Research Council (ARC) Future Fellowship and a Professor of the Centre for Quantum Computation & Intelligent Systems, Faculty of Engineering and Information Technology, University of Technology, Sydney (UTS), Australia. Dr. Zhu's research focuses on data mining, machine learning, and multimedia systems. Since 2000, he has published more than 140 referred journal and conference proceedings papers in these areas. Dr. Zhu is an Associate Editor of the IEEE Transactions on Knowledge and Data Engineering (2009-), a General Co-Chair for the 11th International Conference on Machine Learning and Applications (ICMLA 2012), the Program Committee Co-Chair for the 23rd IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2011), and the 9th International Conference on Machine Learning and Applications (ICMLA 2010).