Online Feature Selection with Capricious Streaming Features: A General Framework

Di Wu[a,e] *Member, IEEE*, Yi He[d], Xin Luo[a,b], *Senior Member, IEEE*, Mingsheng Shang[a], and Xindong Wu[c], *Fellow, IEEE*

[a] Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences, Chongqing, 400714, China
[b] Department of Big Data Analyses Techniques, Cloudwalk, Chongqing 401331, China
[c] Mininglamp Academy of Sciences, Mininglamp Technology, Beijing, China
[d] University of Louisiana at Lafayette, Lafayette, 70503, USA
[e] University of Chinese Academy of Sciences, Beijing, 100049, China
{wudi, luoxin21, msshang}@cigit.ac.cn, {yi.he1, xwu}@louisiana.edu

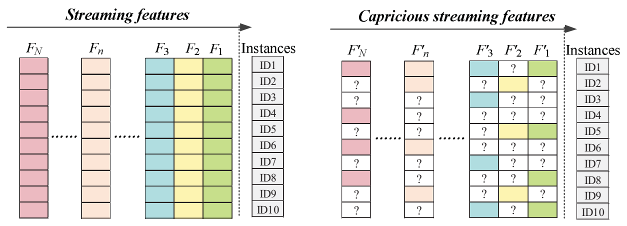
***Abstract*—Online streaming feature selection has received extensive attention in the past few years. Existing approaches have a common assumption that the feature space of the fixed data instances increases dynamically without any missing entry. This assumption, however, does not always hold in many real-world applications. For example, in a credit evaluation system, we cannot collect the complete dynamic features for each person and/or enterprise. Motivated by this observation, this paper aims at conducting online feature selection from *capricious streaming features*, where features flow in one by one with some random missing entries while the number of data instances remains fixed. To do so, we propose a general framework named GF-CSF. The main idea of GF-CSF is to adopt latent factor analysis to preprocess capricious streaming features for completing their missing entries before conducting feature selection. Both theoretical and experimental analyses indicate that GF-CSF can efficiently improve any existing model of online streaming features selection to achieve online capricious streaming features selection.**

***Keywords—big data, online feature selection, latent factor analysis, capricious streaming features*.**

I. INTRODUCTION

High dimensionality is a typical characteristic of big data and ubiquitous in many fields [1]. Data with high dimensionality can cause the problems of high storage and computational cost, performance degradation on unseen data, and difficulty in facilitating data visualization and understanding [2]. Feature selection is one of the most powerful tools for addressing such problems [3]. Its task is to search for an optimal subset from original high-dimensional feature space under some criteria, such as maximizing relevance and minimizing redundancy to class labels in classification[4].

This work is supported in part by the U.S. National Science Foundation under Grant 1613950 and Grant 1763620, in part by the National Natural Science Foundation of China under Grants 61702475, 61772493, 61902370, 61602434, and 91646114, in part by the Natural Science Foundation of Chongqing (China) under grants cstc2019jcyjjqX0013 and cstc2019jcyj-msxmX0578, in part by the Chongqing Research Program of Key Standard Technologies Innovation of Key Industries under grant cstc2017zdcy-zdyfX0076, in part by the Chongqing Municipal Science and Technology Commission under Grant No.cstc2015jcyjA10062, and in part by the Pioneer Hundred Talents Program of Chinese Academy of Sciences (*X. Luo and M. Shang are the Corresponding Authors of this study*).



(a) Streaming features (b) Capricious streaming features

Fig. 1. The illustration for the different dynamic features scenarios. The observed features are marked in multiple colors and the unobserved features (missing entries) are represented by the symbol "?".

Traditional feature selection approaches conduct feature selection under the condition that all candidate features are available [5]. Unfortunately, in real-world applications, the feature space often keeps expanding continuously [6]. Under such circumstances, it is infeasible to gather all the features before conducting feature selection [4]. To address this issue, *online streaming features selection* is achieved by some researchers [4, 7-10]. It can carry out feature selection in real-time rather than waiting for all the features. Representative algorithms on online streaming features selection include Grafting [7], Alpha-investing [8], OSFS [4], SAOLA [9], and OSFASW [10].

Although these algorithms are different from each other on model design, they all have a common assumption that the feature space of fixed data instances increases dynamically without any missing entry (called *streaming features* [4]), as illustrated in Fig.1(a). This assumption, however, does not always hold in many real-world applications where the dynamic features cannot be collected completely. For example, in a smart healthcare platform [11], since a patient's features (describing the symptoms) come from different inspection devices (pulse monitors, thermometers, *etc.*) and service providers (labs, hospitals, *etc.*), it is impracticable to collect each feature for each patient. Motivated by this example, we formulate such dynamic features as *capricious streaming features*, i.e., features flow in one by one with some random missing entries while the number of data instances remains fixed, as illustrated in Fig.1(b).

To address the newly formulated problem, we propose a general framework, which is highly compatible with any existing model of online streaming features selection, for selecting features from capricious streaming features. We name the proposed framework as GF-CSF. Its main idea is to adopt the latent factor analysis (LFA) [12-14] to preprocess the capricious streaming features for completing their missing entries before conducting feature selection. The contributions of this paper are threefold:

1) Our work advances the online streaming features selection for handling capricious streaming features,
2) A general framework (GF-CSF) based on the latent factor analysis (LFA) is proposed to manage capricious streaming features, and
3) Extensive empirical studies on 12 benchmark datasets are carried out to evaluate the proposed GF-CSF.

II. Preliminaries

A. Symbol Annotations

TABLE I. Symbol Annotations.

Symbol	Explanation
M	The number of instances.
F	The streaming features set, $F=\{F_1, F_2, \dots, F_N\}$.
F_n	The nth streaming feature vector of F describing M instances, $n \in \{1, 2, \dots, N\}$.
$f_{m,n}$	The mth element of F_n corresponding to mth instance, $m \in \{1, 2, \dots, M\}$.
t_n	A time point at the nth time, $n \in \{1, 2, \dots, N\}$.
O_n	The observed streaming features set till time t_n, $O_n=\{F_1, F_2, \dots, F_n\}$.
S_n	The selected streaming features set at time t_n, $S_n \subseteq O_n$
F'	The capricious streaming features set, $F'=\{F'_1, F'_2, \dots, F'_N\}$.
F'_n	The nth capricious streaming feature vector of F' describing M instances, $n \in \{1, 2, \dots, N\}$.
K_n	The Known entry set of F'_n.
α_n	The missing rate of F'_n.
α	The total missing rate of F'.
O'_n	The observed capricious streaming features set till time t_n, $O'_n=\{F'_1, F'_2, \dots, F'_n\}$.
S'_n	The selected capricious streaming features set at time t_n, $S'_n \subseteq O'_n$.
C	The label vector of M instances, $C=[c_1, c_2, \dots, c_M]^T$.
Γ, Λ, ζ	Each one represents a set.
B_S	The size of buffer matrix.
B	A buffer matrix to buffer the arrived F'_n during the time from t_n to t_{n+Bs}.
d	The latent factor dimension.
$\widehat{B}$	B's rank-d approximation built on its known entry set by using latent factor analysis.
$\hat{F}'_n$	The complete streaming feature in $\widehat{B}$, which is the corresponding prediction for F'_n.
$\hat{S}'_n$	The selected complete streaming features set till time t_n
$MB(C)_n$	The Markov blanket for C at time t_n.
R	A redundant complete streaming feature to C.
$P(.\|.)$	The conditional probability.
$Y^{W\times Z}$	A sparse matrix with W rows and Z columns.
$y_{w,z}$	Y's element at wth row and zth column, $w \in \{1, 2, \dots, W\}$, $z \in \{1, 2, \dots, Z\}$.
$U^{W\times f}$	The latent factor matrix corresponding to $Y^{W\times Z}$.
$u_{w,\cdot}$	The wth row-vector of U.
$V^{Z\times f}$	The latent factor matrix corresponding to $Y^{W\times Z}$.
$v_{z,\cdot}$	The zth row-vector of V.
$\hat{Y}$	Y's rank- d approximation built on its known entry by using latent factor analysis.
$\hat{y}_{w,z}$	$\widehat{Y}$'s element at wth row and zth column denoting the prediction for $y_{w,z}$.
λ	The regularization coefficient of l_2-norm-based regularization for latent factor analysis.
η	Learning rate for latent factor analysis.
Ω	A $W\times Z$ binary index matrix.
$\|\cdot\|_F$	The Frobenius norm of a matrix.

B. Online Feature Selection from Streaming Features

***Definition* 1 (*Streaming features* [4]).** Given a data instance set with M instances; a feature set F with N features, $F=\{F_1, F_2, \dots, F_N\}$ where $F_n=[f_{1,n}, f_{2,n}, \dots, f_{M,n}]^T$ is a vector corresponding to M instances, $n \in \{1, 2, \dots, N\}$; streaming features means that F is presented in a sequential order, and at each time t_n we only observe the F_n but the exact number of N is unknown in advance or even infinite.

***Definition* 2 (*Online streaming features selection* [4, 9]).** Let $O_{n-1}=\{F_1, F_2, \dots, F_{n-1}\}$ be the observed streaming features set till time t_{n-1}, and S_{n-1} be the selected streaming features set at time t_{n-1}, $S_{n-1} \subseteq O_{n-1}$; online streaming features selection means that at any time t_n, only processing the F_n to select a minimum feature subset S_n from $S_{n-1} \cup F_n$ for maximizing its performance in modeling a predictive model.

C. Latent Factor Analysis (LFA)

***Definition* 3 (*Latent factor analysis*).** Given a sparse matrix $Y^{W\times Z}$ and the latent factor dimension d, latent factor analysis (LFA) aims at extracting two corresponding latent factor matrices $U^{W\times d}$ and $V^{Z\times d}$ to achieve Y's rank-d approximation $\widehat{Y}$ by minimizing the errors between Y and $\widehat{Y}$ on the known entry set of Y, where $\widehat{Y}$ is given by $\widehat{Y}=UV^T$ and its each element $\hat{y}_{w,z}$ is the prediction for Y's each corresponding element $y_{w,z}$.

From definition 3, we know that an appropriate objective function of minimizing the errors between Y and $\widehat{Y}$ is highly important. An Euclidean distance-based objective is a common choice [12-15]:

$$\underset{U,\,V}{\arg\min}\,\varepsilon(U,\ V)=\frac{1}{2}\Omega\left\|Y-\hat{Y}\right\|_F^2=\frac{1}{2}\Omega\left\|Y-UV^T\right\|_F^2, \tag{1}$$

where Ω is *a* $W\times Z$ binary index matrix:

$$\Omega_{w,z}=\begin{cases}1 & \text{if } y_{w,z} \text{ is observed}\\ 0 & \text{otherwise}\end{cases}. \tag{2}$$

To avoid overfitting on the known entry set of Y, L_2-norm regularization is usually incorporated into (1) [12-15]:

$$\underset{U,\,V}{\arg\min}\,\varepsilon(U,\ V)=\frac{1}{2}\Omega\left\|Y-UV^T\right\|_F^2+\frac{\lambda}{2}\left(\left\|U\right\|_F^2+\left\|V\right\|_F^2\right). \tag{3}$$

With an optimization algorithm, such as stochastic gradient descent (SGD), U and V can be successfully extracted.

III. The proposed general framework (GF-CSF)

A. Problem Setting

***Definition* 4 (*Capricious streaming features*).** Given streaming features set $F=\{F_1, F_2, \dots, F_N\}$; capricious streaming features mean that at each time t_n, the observed streaming feature vector $F_n=[f_{1,n}, f_{2,n}, \dots, f_{M,n}]^T$, $n \in \{1, 2, \dots, N\}$, has some random missing entries; $F'=\{F'_1, F'_2, \dots, F'_N\}$ represents the capricious streaming features set, the missing rate α_n of F'_n is calculated by: $\alpha_n=1-|K_n|/M$, where K_n is the known entry set of F'_n, $n \in \{1, 2, \dots, N\}$.

Given capricious streaming features set $F'=\{F'_1, F'_2, \dots, F'_N\}$; let $O'_{n-1}=\{F'_1, F'_2, \dots, F'_{n-1}\}$ be the observed capricious

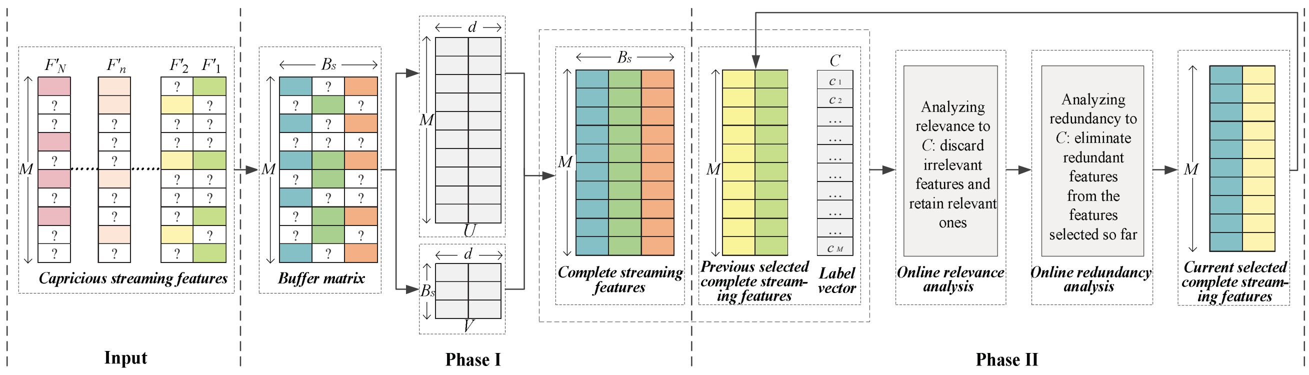

F'_N
F'_n
F'_2
F'_1
M
Capricious streaming features
B_S
Buffer matrix
d
U
V
Complete streaming features
C
c_1
c_2
c_M
Previous selected complete streaming features
Label vector
Analyzing relevance to C: discard irrelevant features and retain relevant ones
Online relevance analysis
Analyzing redundancy to C: eliminate redundant features from the features selected so far
Online redundancy analysis
Current selected complete streaming features
Input
Phase I
Phase II

Fig.2. The flowchart of GF-CSF for any existing model of online streaming features selection.

streaming features set till time t_{n-1}, and S'_{n-1} be the selected capricious streaming features set at time t_{n-1}, $S'_{n-1}\subseteq O'_{n-1}$; the challenge is that at any time t_n, how to select a minimum feature subset S'_n from $S'_{n-1}\cup F'_n$ for maximizing its performance in modeling a predictive model. Supposing we develop a classification model and $C=[c_1, c_2, \ldots, c_M]^{\mathrm{T}}$ is a label vector corresponding to M instances, our problem can be formulated as follow:

$$S'_n=\underset{\Gamma}{\arg\min}\left\{|\Gamma|:\Gamma=\underset{\Lambda\subseteq\{S'_{n-1}\cup F'_n\}}{\arg\max}P\left(C\,|\,\Lambda\right)\right\}. \tag{4}$$

B. GF-CSF

We design GF-CSF as depicted in Fig. 2, where there are two phases. Phase I is to preprocess capricious streaming features for completing their missing entries. Phase II is to select features from these complete features. Next, we explain the two phases with a case that starts from any time t_n.

1) Phase I

First, we use a buffer matrix B to buffer the arrived capricious streaming features, the size of buffer matrix is set as B_S. When the buffer matrix is full (during the time from t_n to t_{n+Bs}), we have a temporary sparse $M\times B_S$ matrix B, which consists of $\{F'_n, F'_{n+1}, \ldots, F'_{n+Bs}\}$.

***Definition* 5 (*Complete streaming features*).** Given buffer matrix B=$\{F'_n, F'_{n+1}, \ldots, F'_{n+Bs}\}$, $\hat{\mathrm{B}}$ is B's rank-d approximation built on its known entry set by using latent factor analysis. Then, the features in $\hat{\mathrm{B}}$, denoted as $\{\hat{F}'_n, \hat{F}'_{n+1}, \ldots, \hat{F}'_{n+B_S}\}$, are the complete streaming features.

Next, we explain how to obtain the complete streaming features $\{\hat{F}'_n, \hat{F}'_{n+1}, \ldots, \hat{F}'_{n+B_S}\}$ by using LFA. As B contains numerous missing entries, (3) should be expanded into data density-oriented form for high efficiency in both storage and computation [12, 14]. Hence, we minimize the following objective function:

$$\forall m\in\{1,2,\ldots,M\},\forall j\in\{n,n+1,\ldots,n+B_S\}:\underset{U,V}{\arg\min}\,\varepsilon(U,V)$$
$$=\frac{1}{2}\sum_{(m,j)\in K_j}\left(f_{m,j}-\sum_{k=1}^{d}u_{m,k}v_{j,k}\right)^2+\frac{\lambda}{2}\sum_{(m,j)\in K_j}\left(\sum_{k=1}^{d}u_{m,k}^2+\sum_{k=1}^{d}v_{j,k}^2\right). \tag{5}$$

As discussed in [13, 14, 16, 17], SGD algorithm has fast convergence and ease of implementation in optimizing such a bilinear objective (5). Considering the instant loss of (5) on a single entry $f_{m,j}$:

$$\varepsilon_{m,j}=\frac{1}{2}\left(f_{m,j}-\sum_{k=1}^{d}u_{m,k}v_{j,k}\right)^2+\frac{\lambda}{2}\left(\sum_{k=1}^{d}u_{m,k}^2+\sum_{k=1}^{d}v_{j,k}^2\right). \tag{6}$$

With SGD, the latent factors involved in (6) are trained by moving them along the opposite of the stochastic gradient of (6) with respect to each single latent factor.

$$\forall k\in\{1,2,\ldots,d\}:\begin{cases}u_{m,k}\leftarrow u_{m,k}-\eta\dfrac{\partial\varepsilon_{m,j}}{\partial u_{m,k}}\\[2ex] v_{j,k}\leftarrow v_{j,k}-\eta\dfrac{\partial\varepsilon_{m,j}}{\partial v_{j,k}}\end{cases}. \tag{7}$$

Then, we extend (7) to obtain the following training rules:

$$On\ f_{m,j},\ for\ k=1\sim d:$$
$$\begin{cases}u_{m,k}\leftarrow u_{m,k}+\eta v_{j,k}\left(f_{m,j}-\sum_{k=1}^{d}u_{m,k}v_{j,k}\right)-\lambda\eta u_{m,k}\\ v_{j,k}\leftarrow v_{j,k}+\eta u_{m,k}\left(f_{m,j}-\sum_{k=1}^{d}u_{m,k}v_{j,k}\right)-\lambda\eta v_{j,k}\end{cases}. \tag{8}$$

After all the known entries $f_{m,j}$ in K_j, $j\in\{n, n+1, \ldots, n+B_S\}$, are used to train with (8), we can extract the latent factor matrices U and V. Finally, based on them, the complete streaming features $\{\hat{F}'_n, \hat{F}'_{n+1}, \ldots, \hat{F}'_{n+B_S}\}$ in $\hat{\mathrm{B}}$ can be obtained by:

$$\hat{\mathrm{B}}=UV^{\mathrm{T}}. \tag{9}$$

2) Phase II

After phase I, we obtain the complete streaming features $\{\hat{F}'_n, \hat{F}'_{n+1}, \ldots, \hat{F}'_{n+B_S}\}$. Let $\hat{S}'_{n-1}$ be the selected complete streaming features set from $\hat{F}'_i$ till time t_{n-1}, $i\in\{1, 2, \ldots, n-1\}$. Then, some related definitions on $\hat{F}'_n$ and $\hat{S}'_{n-1}$ are given as follows to introduce phase II.

***Definition* 6 (*Irrelevant feature* [18]).** $\hat{F}'_n$ is an irrelevant feature to C, if and only if

$$\forall\zeta\subseteq\hat{S}'_{n-1}\ s.t.\ P\left(C\,|\,\zeta,\hat{F}'_n\right)=P\left(C\,|\,\zeta\right), \tag{10}$$

Definition* 7** (Markov blanket*** [4]). A Markov blanket of C at time t_{n-1} is a subset of $\hat{S}'_{n-1}$, denoted as $MB(C)_{n-1}$ and $MB(C)_{n-1} \subseteq \hat{S}'_{n-1}$, that makes the following formula hold:

$$\forall \zeta \subseteq \hat{S}'_{n-1} - MB(C)_{n-1}\ s.t.\ P(C \mid MB(C)_{n-1}, \zeta) = P(C \mid MB(C)_{n-1}), \tag{11}$$

where $\hat{S}'_{n-1} - MB(C)_{n-1}$ indicates the subset of $\hat{S}'_{n-1}$ excluding $MB(C)_{n-1}$.

Definition* 8** (Redundant complete streaming features***). At time t_n, supposing $\hat{F}'_n$ is not an irrelevant feature to C, the redundant complete streaming features to C satisfy the following formula:

$$\begin{aligned} &\forall R \in MB(C)_{n-1} \cup \{\hat{F}'_n\},\ \exists \zeta \subseteq MB(C)_{n-1} \cup \{\hat{F}'_n\} - \{R\} \\ &s.t.\ P(C \mid R, \zeta) = P(C \mid \zeta), \end{aligned} \tag{12}$$

where R denotes a redundant complete streaming feature to C and MB(C)n-1$\cup\{\hat{F}'_n\}$-$\{R\}$ denotes the subset of MB(C)n-1$\cup\{\hat{F}'_n\}$ excluding $\{R\}$.

In phase II, we conduct online feature selection based on the above definitions, which has been concluded in Table II.

TABLE II. CONDUCTING FEATURE SELECTION IN PHASE II.

1	**Input**
(1)	Input $\widehat{B} = \{\hat{F}'_n, \hat{F}'_{n+1}, \dots, \hat{F}'_{n+B_S}\}$, C, and a cache set $CS=\hat{S}'_{n-1}$
2	**Online relevance analysis**
(2)	For i=0 to B_S
(3)	Get $\hat{F}'_{n+i}$ from $\widehat{B}$
(4)	Determine whether $\hat{F}'_{n+i}$ is relevant to C based on definition 6
(5)	If $\hat{F}'_{n+i}$ is irrelevant to C, continue
(6)	Otherwise, $\hat{F}'_{n+i}$ is added into CS
3	**Online redundancy analysis**
(7)	Identify redundant complete streaming features in CS based on definitions 7 and 8
(8)	Remove redundant complete streaming features from CS
(9)	End if
(10)	End for
4	**Output**
(11)	Output CS to $\hat{S}'_{n+B_S}$

C. Computational Complexity Analysis

Compared with the existing models of online streaming features selection, like OSFS [4], SAOLA [9], and OSFASW [10], GF-CSF needs extra computation for completing the missing entries of capricious streaming features (phase I). According to [12-14], the computational complexity of LFA is decided by the latent factor dimension d and the input known entry count. Supposing M instances with N capricious streaming features have arrived so far, the total missing rate of these capricious streaming features is α, then the computational cost of GF-CSF for phase I is $O(M \times N \times (1-\alpha) \times d)$.

IV. EXPERIMENTS AND RESULTS

A. General Settings

12 benchmark classification datasets are randomly selected from *NIPS* 2003 feature selection challenge [19], Rosenwald *et al.*[20], Chapelle *et al.*[21], and UCI repositories [22], to validate the effectiveness of GF-CSF. Table III summarizes the properties of all the datasets.

TABLE III. PROPERTIES OF ALL THE DATASETS.

No.	Name	#Instances	#Features	#Class
D1	colon	62	2000	2
D2	lung	203	3312	4
D3	lungcancer	181	12533	2
D4	lymphoma	62	4026	3
D5	marti1	500	1024	2
D6	prostate	102	6033	2
D7	reged1	500	999	2
D8	SMK_CAN_187	187	19993	2
D9	COIL	1500	241	6
D10	DriveFace	606	6400	3
D11	Human activity recognition	10299	561	6
D12	HAPT	10929	561	12

To test GF-CSF's performance, three representative state-of-the-art algorithms of online streaming features selection, i.e., OSFS [4], SAOLA [9], and OSFASW [10], are chosen to conduct the experiments. We respectively integrate these three algorithms into GF-CSF to make them can conduct online feature selection from capricious streaming features. The three modified algorithms are named as M1+GF-CSF, M2+GF-CSF, and M3+GF-CSF, respectively. Besides, since this work focuses on selecting features to implement classification tasks, KNN, SVM, and Random Forest, which are frequently used in classification tasks [23, 24], are adopted as the base classifiers. Table IV summarizes all the parameters of the above algorithms and classifiers.

TABLE IV. ALL THE PARAMETERS USED IN THE EXPERIMENTS.

Mark	Algorithm	Parameters
M1	OSFS	*Alpha*=0.05, *Z test.*
M2	SAOLA	*Alpha*=0.05, *Z test.*
M3	OSFASW	*Alpha*=0.05, *Z test*, Adjustment coefficient of sliding-window=1.
/	KNN	*Number of neighbors*=3*; Euclidean distance.*
/	SVM	*LIBSVM* [25]*: all the parameters are set as default values.*
/	Random Forest	*The number of decision trees*=6.

In the next experiments, the features of each dataset are simulated to flow in one by one with some random missing entries. Then, we conduct comparisons between the original algorithms and their modified versions with GF-CSF. We use 10-fold cross-validation strategy to conduct experiments.

B. Selected Features Analysis

Supposing the missing rate α is 10% and the size of buffer matrix is set as B_S=10. Table V records the selected features on the 12 datasets. Note that the original algorithms are tested on the streaming features (without missing entries) while their modified versions are tested on the capricious streaming features (with 10% missing entries). From Table V, we have two findings. One is that among the three original algorithms, they select a different number of features on the same dataset, which means that they have different characteristics in feature selection. The other one is that the original algorithms and their respective modified versions tend to select the same features on the same dataset in most cases. Two exceptions are on the D2 and D8, where the modified algorithms select

TABLE V. THE SELECTED FEATURES ON ALL THE DATASETS WHEN α=0.1 AND B_S=10.

Dataset	The number of selected features (M1 VS. M1+GF-CSF)			The number of selected features (M2 VS. M2+GF-CSF)			The number of selected features (M3 VS. M3+GF-CSF)		
	M1 (without missing entries)	M1+GF-CSF (with 10% missing entries)	Number of same features	M2 (without missing entries)	M2+GF-CSF (with 10% missing entries)	Number of same features	M3 (without missing entries)	M3+GF-CSF (with 10% missing entries)	Number of same features
D1	3	3	3	4	4	4	2	2	2
D2	16	6	1	30	9	1	11	5	1
D3	8	9	8	39	40	37	4	4	4
D4	6	7	2	39	40	26	5	4	3
D5	1	1	1	1	1	1	1	1	1
D6	5	5	5	15	15	15	4	3	3
D7	13	13	13	16	16	16	13	13	13
D8	10	4	0	11	6	0	4	3	0
D9	6	6	6	3	3	3	6	6	6
D10	8	8	7	5	5	3	6	4	1
D11	17	16	13	12	9	8	12	9	5
D12	22	23	17	8	8	8	8	15	3

TABLE VI. THE CLASSIFICATION ACCURACY (%) OF THE SELECTED FEATURES RECORDED IN TABLE V.

Dataset	SVM							KNN							Random Forest					
	M1	M2	M3	M1+ GF-CSF	M2+ GF-CSF	M3+ GF-CSF		M1	M2	M3	M1+ GF-CSF	M2+ GF-CSF	M3+ GF-CSF		M1	M2	M3	M1+ GF-CSF	M2+ GF-CSF	M3+ GF-CSF
D1	80.95	86.21	78.26	80.05	**86.51**	**79.36**	0	82.13	83.46	81.56	**85.9**	**86.21**	78.38	0	81.54	86.46	79.05	77.79	78.56	78.87
D2	88.77	90.5	89.8	84.74	84.18	80.55	0	88.23	87.33	88.11	85.13	83.9	77.73	0	87.3	86.42	85.8	83.74	83.35	78.25
D3	98.06	97.68	98.28	**98.07**	**98.23**	97.85	0	97.85	98.56	98.78	**98.07**	**98.79**	98.34	0	97.62	97.4	96.79	97.41	**97.74**	97.68
D4	91.59	87.05	92.18	90.69	85.79	**93.23**	0	99.67	99.33	100	**100**	98.41	97.67	0	95.21	92.38	95.51	91.46	**94.79**	**96.44**
D5	52.16	45.92	59.72	**58.16**	**61.68**	56.64	0	84.28	84.8	84.48	**85.44**	83.8	**85.12**	0	76.28	77	77.92	**77.32**	**77.2**	76.6
D6	86.7	91.39	88.45	**87.63**	**91.59**	**91.95**	0	95.71	93.14	94.5	95.29	**94.15**	94.3	0	92.75	91.99	92.34	91.75	91.72	92.1
D7	87.63	91.59	91.95	**99**	**98.6**	**98.92**	0	95.29	94.15	94.3	**99.2**	**98.92**	**99.34**	0	91.75	91.72	92.1	**97.76**	**98.08**	**98.84**
D8	76.81	73.44	76.41	76.46	**76.36**	76.23	0	75.3	70.39	72.31	**75.51**	**71.58**	70.03	0	71.47	71.01	72.19	**72.81**	70.07	71.45
D9	50.77	26.47	42.43	**51.36**	**26.69**	**43.13**	0	80.64	53.69	80.53	**81.16**	53.16	**80.56**	0	79.27	53.89	77.68	**79.57**	**54.24**	77.25
D10	90.2	92.34	90.99	**90.33**	92.14	**92.41**	0	94.75	94.49	94.49	**95.51**	94.22	**94.62**	0	93.86	93.93	94.02	**93.86**	**94.39**	**94.09**
D11	75.81	75.14	81.54	75.21	**75.26**	75.76	0	80.16	75.46	83.94	78.15	74.85	78.11	0	84.97	79.4	88.33	84.17	**80.85**	81.04
D12	81.48	70.69	50.84	81.06	67.59	**79.92**	0	82.52	69.59	52.13	81.96	67.59	**79.01**	0	88.15	77.88	58.89	**88.51**	75.53	**87.33**
Average	80.08	77.37	78.40	**81.06**	**78.72**	**80.50**		88.04	83.70	85.43	**88.44**	**83.80**	**86.10**		86.68	83.29	84.22	86.35	83.04	**85.83**

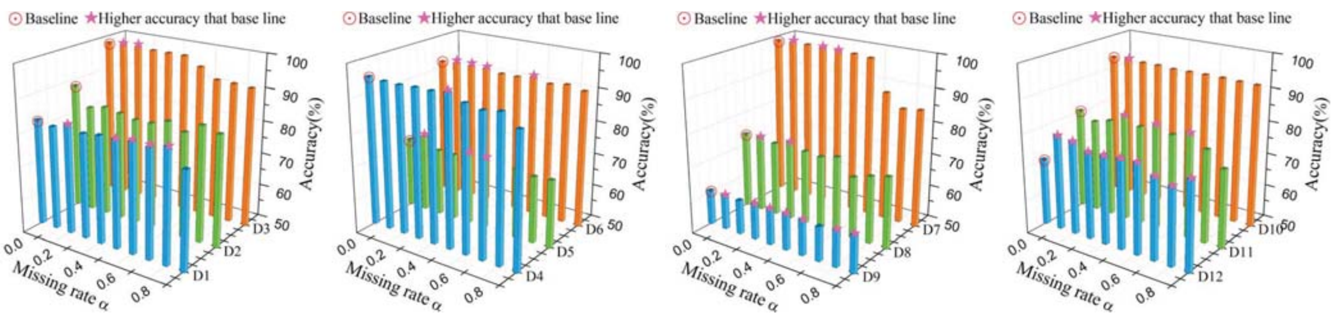


Fig. 3. The classification accuracy as α increases from 0.1 to 0.9. Baseline stands for the average accuracy of three original algorithms on three classifiers tested on streaming features (without missing entries). The other each bar stands for the average accuracy of three modified algorithms on three classifiers tested on capricious streaming features (with missing entries).

fewer features than their original versions. These two findings verify that GF-CSF can improve the original algorithms to achieve online capricious streaming features selection without changing their characteristics in feature selection.

Next, we evaluate the quality of the above selected features. On each dataset, we use the features selected by the original algorithm to train a classifier first and then utilize the features selected by its modified version to train another classifier. Finally, we compare the classification accuracy between the above two classifiers. The experimental results are recorded in table VI, where modified algorithms perform better than their respective original versions are highlighted. Form it, we observe that: 1) On D2, the modified algorithms have lower accuracy than their original versions with all the classifiers. One reason is that, as shown in Table V, the modified algorithms select the less and different features; 2) With SVM and KNN, the modified algorithms achieve higher accuracy than their respective original versions on more datasets; 3) With Random Forest, although the modified algorithms have not improved the accuracy of their original versions on more datasets, their accuracies are very close. Hence, the above observations validate that the quality of features selected by modified algorithms is higher than, or at least comparable to that selected by their original versions.

C. Impacts of Missing Rate α

In this section, we conduct the experiments that the missing rate α of capricious streaming features is increased from 0.1 to 0.9. The size of buffer matrix is also set as B_S=10. The results are shown in Fig. 3. As α increases, we see that the average classification accuracy decreases in general as expected because more entries are missing. However, the decrease of the average classification accuracy is not obvious in the initial phases (α is below 0.6). Specially, we find that there are many cases that their average classification accuracies are higher than the baseline, such as on D1 with α=0.2, 0.5, 0.6, 0.7, and 0.8, on D3 with α=0.1 and 0.2, on D4 with α=0.5, on D5 with α=0.1, 0.4 and 0.5, on D6 with α=0.1, 0.2, 0.3, and 0.6, on D7 with α=0.1, 0.3 and 0.4, on D8 with α=0.1 and 0.3, on D9 with α=0.1, 0.3, 0.4, 0.5, 0.6, 0.8, and 0.9, *etc*.

Therefore, we conclude that GF-CSF can make the original algorithms to process capricious streaming features without performance degradation or even with performance improvement when missing rate α is below 0.6.

V. Conclusions

This paper proposes a general framework, named as GF-CSF, to achieve online feature selection from capricious streaming features. The main idea of GF-CSF is to adopt the latent factor analysis to preprocess the capricious streaming features for completing their missing entries before conducting feature selection. GF-CSF is highly compatible with any existing model of online streaming features selection. In particular, three representative state-of-the-art algorithms of online streaming features selection are modified based on GF-CSF to conduct the comparison experiments. In the experiments, 12 benchmark classification datasets are simulated as the capricious streaming features. The experimental results validate that GF-CSF can efficiently improve these three representative algorithms to achieve online capricious streaming features selection.

References

[1] X. Hu, P. Zhou, P. Li, J. Wang, and X. Wu, "A survey on online feature selection with streaming features," *Frontiers of Computer Science,* vol. 12, pp. 479-493, 2018.

[2] J. Li, K. Cheng, S. Wang, F. Morstatter, R. P. Trevino, J. Tang*, et al.*, "Feature selection: A data perspective," *ACM Computing Surveys (CSUR),* vol. 50, pp. 94, 2018.

[3] S. Alelyani, J. Tang, and H. Liu, "Feature selection for clustering: A review," in *Data Clustering*, Chapman and Hall/CRC, 2018, pp. 29-60.

[4] X. Wu, K. Yu, W. Ding, H. Wang, and X. Zhu, "Online feature selection with streaming features," *IEEE transactions on pattern analysis and machine intelligence,* vol. 35, pp. 1178-1192, 2013.

[5] I. Rodriguez-Lujan, R. Huerta, C. Elkan, and C. S. Cruz, "Quadratic programming feature selection," *Journal of Machine Learning Research,* vol. 11, pp. 1491-1516, 2010.

[6] Y. Shen, C. Wu, C. Liu, Y. Wu, and N. Xiong, "Oriented feature selection SVM applied to cancer prediction in precision medicine," *IEEE Access,* vol. 6, pp. 48510-48521, 2018.

[7] S. Perkins and J. Theiler, "Online feature selection using grafting," *In Proceedings of the 20th International Conference on Machine Learning*, 2003, pp. 592-599.

[8] J. Zhou, D. P. Foster, R. A. Stine, and L. H. Ungar, "Streamwise feature selection," *Journal of Machine Learning Research,* vol. 7, pp. 1861-1885, 2006.

[9] K. Yu, X. Wu, W. Ding, and J. Pei, "Scalable and accurate online feature selection for big data," *ACM Transactions on Knowledge Discovery from Data,* vol. 11, pp. 16, 2016.

[10] D. You, X. Wu, L. Shen, S. Deng, Z. Chen, C. Ma*, et al.*, "Online Feature Selection for Streaming Features Using Self-Adaption Sliding-Window Sampling," *IEEE Access,* vol. 7, pp. 16088-16100, 2019.

[11] Y. He, B. Wu, D. Wu, E. Beyazit, S. Chen and X. Wu, " Online Learning from Capricious Data Streams: A Generative Approach," *In Proceedings of the 28th International Joint Conference on Artificial Intelligence, IJCAI*, 2019, pp. 2491-2497.

[12] D. Wu, X. Luo, M. Shang, Y. He, G. Wang, and X. Wu, "A Data-Aware Latent Factor Model for Web Service QoS Prediction," *In proceeding of the 23rd Pacific-Asia Conference on Knowledge Discovery and Data Mining, PAKDD*, Springer, 2019, pp. 384-399.

[13] Y. Koren, R. Bell, and C. Volinsky, "Matrix Factorization Techniques for Recommender Systems," *Computer,* vol. 42, pp. 30-37, 2009.

[14] X. Luo, M. Zhou, Y. Xia, and Q. Zhu, "An Efficient Non-Negative Matrix-Factorization-Based Approach to Collaborative Filtering for Recommender Systems," *IEEE Transactions on Industrial Informatics,* vol. 10, pp. 1273-1284, 2014.

[15] D. Wu, X. Luo, M. Shang, Y. He, G. Wang, and M. Zhou, "A Deep Latent Factor Model for High-Dimensional and Sparse Matrices in Recommender Systems," *IEEE Transactions on Systems, Man, and Cybernetics: Systems,* pp. 1-12, 2019.

[16] X. Luo, M. Zhou, S. Li, Y. Xia, Z.-H. You, Q. Zhu*, et al.*, "Incorporation of efficient second-order solvers into latent factor models for accurate prediction of missing QoS data," *IEEE transactions on cybernetics,* vol. 48, pp. 1216-1228, 2018.

[17] X. Luo, M. Zhou, Y. Xia, Q. Zhu, A. C. Ammari, and A. Alabdulwahab, "Generating highly accurate predictions for missing QoS data via aggregating nonnegative latent factor models," *IEEE transactions on neural networks and learning systems,* vol. 27, pp. 524-537, 2016.

[18] R. Kohavi and G. H. John, "Wrappers for feature subset selection," *Artificial intelligence,* vol. 97, pp. 273-324, 1997.

[19] I. Guyon, S. Gunn, A. Ben-Hur, and G. Dror, "Result analysis of the NIPS 2003 feature selection challenge," in *Advances in neural information processing systems*, 2005, pp. 545-552.

[20] A. Rosenwald, G. Wright, W. C. Chan, J. M. Connors, E. Campo, R. I. Fisher*, et al.*, "The use of molecular profiling to predict survival after chemotherapy for diffuse large-B-cell lymphoma," *New England Journal of Medicine,* vol. 346, pp. 1937-1947, 2002.

[21] O. Chapelle, B. Scholkopf, and A. Zien, "Semi-Supervised Learning (Chapelle, O. et al., Eds.; 2006)[Book reviews]," *IEEE Transactions on Neural Networks,* vol. 20, pp. 542-542, 2009.

[22] A. Frank and A. Asuncion. UCI Machine Learning Repository 2010. http://archive.ics.uci.edu/ml/index.php

[23] D. Wu, X. Luo, G. Wang, M. Shang, Y. Yuan, and H. Yan, "A Highly Accurate Framework for Self-Labeled Semisupervised Classification in Industrial Applications," *IEEE Transactions on Industrial Informatics,* vol. 14, pp. 909-920, 2018.

[24] D. Wu, M. S. Shang, X. Luo, J. Xu, H. Y. Yan, W. H. Deng*, et al.*, "Self-training semi-supervised classification based on density peaks of data," *Neurocomputing,* vol. 275, pp. 180-191, 2018.

[25] C.-C. Chang and C.-J. Lin, "LIBSVM: a library for support vector machines," *ACM Transactions on Intelligent Systems and Technology,* vol. 2, pp. 27, 2011.