Extremely High-Dimensional Feature Selection via Feature Generating Samplings

Shutao Li, Member, IEEE and Dan Wei

Abstract—To select informative features on extremely high-dimensional problems, in this paper, a sampling scheme is proposed to enhance the efficiency of recently developed feature generating machines (FGMs). Note that in FGMs $O(n \log r)$ time complexity should be taken to order the features by their scores; the entire computational cost of feature ordering will become unbearable when $m$ is very large, for example, $m > 10^3$, where $m$ is the feature dimensionality and $r$ is the size of the selected feature subset. To solve this problem, in this paper, we propose a feature generating sampling method, which can reduce this computational complexity to $O(G \log(G) + G + \log(G))$ while preserving the most informative features in a feature buffer, where $G$ is the maximum number of nonzero features for each instance and $G$ is the buffer size. Moreover, we show that our proposed sampling scheme can be deemed as the birth-death process based on random processes theory, which guarantees to include most of the informative features for feature selections. Empirical studies on real-world datasets show the effectiveness of the proposed sampling method.

Index Terms—Extremely high dimensional problem, feature generating machine, feature selection, informative feature.

I. INTRODUCTION

WITH THE fast development of the Internet, we are facing fast-growing large-scale data which restrict the applicability of many machine learning algorithms [1]. Particularly, in many machine learning applications, we have to deal with extremely high-dimensional problems, where the dimension of each data item can be over $10^9$ [1]. High dimensionality not only increases the learning cost, but also deteriorates the learning performance, a problem known as the “curse of dimensionality” [2]–[4]. More seriously, in some classification problems, due to the nonlinear intrinsic property of dataset, we need to do some expansions to make the dataset linearly separable such that the fast linear training methods can be applicable. For example, several works have been proposed to speed-up the training of kernel methods as well as their prediction process by using explicit feature mappings for linear optimization techniques [5], [6]. However, a critical challenge for explicit feature mappings is that the feature vectors are generally in very high-dimensional space. A commonly used feature mapping is the low-degree polynomial kernel feature mapping [5]. This extension can be of great importance in data mining, such as in text analysis [5]. With two-degree polynomial mappings, we can at least learn the second-order information of variables. However, the dimension of such features would be very high in the mapped feature space. The two-degree low-degree polynomial feature mapping can be defined as:

$$\phi(x) = [1, \sqrt{2}x_1, \ldots, \sqrt{2}x_m, \lambda x_1^2, \ldots, \lambda x_m^2, \sqrt{2}x_1x_2, \ldots]$$

where $m$ is the dimensionality. The dimension of this feature map is $(m + 2)(m + 1)/2$. Even with a medium dimension $m$, the dimension of this feature space is very huge, which restricts their applications. Even worse, in information retrieval applications, the dimension of documents can be as large as $O(10^{11})$ to $O(10^{21})$ by using the $w$-shingles to represent the documents [1], [7].

The extremely high dimensionality of the dataset brings great challenges in the training of learning algorithms. Fortunately, typical high-dimensional datasets are often highly sparse [7], [8]. Again, if the original dataset is sparse, some explicit feature mappings exist to make their mappings very sparse in the mapped feature space [5]. Obviously, if the data is sparse, it is not wise to perform the learning process using all the features. Rather, a small subset of features that can statistically represent the structure of the original dataset is more favorable [7]. Feature sampling is an apparent way to form such a small feature subset. However, the sampling of features is much more challenging than sampling of instances, especially on high-dimensional sparse data. A conventional method is to randomly choose a small fraction of the original features, which is simple but not accurate and performs poorly when most of the entries are zeros [7], [8]. Different from simple random picking, Li et al. [8] introduced a new feature sampling method, conditional random sampling (CRS), to directly sample the data sketches on the high-dimensional sparse dataset; it shows several advantages over conventional sampling methods. CRS is based on the sparse presentation (called postings) of a dataset. In CRS, firstly, the columns of the sparse dataset are randomly permuted. Based on the permuted postings, the elements with their indicies(ID) $ID > D_i$ are excluded, where the effective sample size $D_i$ can be estimated by:

$$D_i = \min(\max(ID(\mathbf{k})), \ldots, \max(ID(\mathbf{k})))$$

where $\mathbf{k}, i = 1, \ldots, n$ denotes the permuted instance (or postings) and $\max(ID(\mathbf{k}))$ denotes the ID of the last element.
for each sample. Note CRS is proposed to preserve the pair-wise distance, for example, $\ell_2$ norm distance and $\ell_1$ norm distance. However, it cannot guarantee good generalization performance for feature selection. In addition, because $D_t$ is highly related to the dataset, it is difficult for CRS to control the sampling size. In addition, if $D_t$ is occasionally very large, CRS cannot sufficiently decrease the computational cost.

Another approach to handle the extremely high-dimensional dataset is by integrating various hashing techniques with the linear learning algorithms such as linear SVM and logistic regression [1]. By using the hashing techniques, such methods can much reduce the data dimensionality while preserve the similarity between data items, and thus much reduce the training cost and storage burdens. However, they are incapable of selecting the most informative features regarding the original dataset.

Besides the sampling and hashing, feature selection that aims to select the most informative features from the original dataset can be another effective way to do dimension reductions [9], [10]. In addition, because feature selection can eliminate the irrelevant features, it can much improve the generalization ability if there are a lot of irrelevant features in the original dataset [4], [9], [10], [12], [13]. Therefore, it plays an important role in data mining, image recognition, and microarray data analysis [4], [14]. Over the past decades, many supervised feature selection methods have been proposed [4], [14]-[16]. For classification problems, a direct and efficient method is to solve the following [14]-[16].

For classification problems, a direct and efficient supervised feature selection methods have been proposed [4], [14], [15]. Over the past decades, many supervised feature selection methods have been proposed [4], [14]-[16]. For classification problems, a direct and efficient method is to solve the following $\ell_1$ regularized problem [11]:

$$\min \alpha \left\{ \sum_{i=1}^{n} \xi_i^2 + \|w\|_1 \right\}.$$  \hspace{1cm} (2)

However, since the explicit presentation of all points is required to be known in advance in the training process of the $\ell_1$ models, the computational cost and memory requirements will be unbearable for extremely high-dimensional dataset. Recently, Tan et al. [17] proposed a new feature selection based on $\ell_0$ regularization, namely, feature generating machine (FGM), which facilitate high-dimensional feature selection [17]. However, it is still challenging for FGM to tackle extremely high-dimensional problems, where the number of features can exceed $O(10^3)$.

To address the extremely high-dimensional feature selections, motivated by the feature generating scheme, we propose a new sampling technique for feature selection by making use of the data sparsity. In a sparse dataset, it is a natural assumption that a feature is a candidate of informative feature only if it has higher occurrences than other features. In other words, those features with smaller occurrences have less impact to the learning performance and therefore an informative feature should have denser format. With this assumption, the proposed sampling method tries to keep the score of the dense features in a buffer and then do the feature selection from the buffered features. Accordingly, the computational cost can be much reduced. The validity of such sampling is verified via a birth-death model. By integrating the proposed technique with FGM, the proposed feature generating sampling (FGS) can efficiently and effectively tackle the very high-dimensional problems up to the scale of $O(10^3)$. Compared with CRS, the proposed FGS method can achieve much better performance for the task of feature selections on several real-world datasets.

The rest of this paper is organized as follows. First, we will illustrate our proposed sampling algorithm in Section II. Empirical studies will be presented in Section III, and the conclusion remarks will be discussed in Section IV.

II. CHAIN SAMPLING FOR LARGE-SCALE AND EXTREMELY HIGH-DIMENSIONAL PROBLEMS

A. Preliminaries

In the following, we denote the transpose of a vector/matrix by the superscript $\top$, and operator $\odot$ represents the elementwise product between two vectors/matrices.

The feature generating sampling is motivated by the feature generating scheme [17]. The key idea of feature generating is to find a subset of features which contributes to the largest margin separation between two opposite classes. Specifically, a vector $d = [d_1, \ldots, d_m]^T$, where $d_i \in [0, 1], i = 1, \ldots, m$, is introduced to each feature in a sample $x \in \mathbb{R}^m$ so as to indicate whether this feature is selected or not for SVM training. In addition, a constraint $\sum_{i=1}^{m} d_i \leq r$ is introduced for the purpose of feature selection, where $r$ is the maximum number of features to select.

Let $D = \{ d \mid \sum_{i=1}^{m} d_i \leq r, d_i \in [0, 1], i = 1, \ldots, m \}$ be the domain of $d$, the resultant problem is formulated as a mixed integer programming problem as follows:

$$\min \alpha \left\{ \sum_{i=1}^{n} \xi_i^2 + \|w\|_1 \right\}.$$  \hspace{1cm} (3)

$$\text{s.t.} \quad y_i w(x_i \odot d) \geq 1 \text{ if } \alpha_i \geq 1, \quad i = 1, \ldots, n$$

$$\xi_i \geq 0 \text{ denotes the hinge loss of the } i\text{th sample } x_i \text{ and } y_i \|w\| \text{ denotes the margin. By adopting a tight convex relaxation, (3) can be transformed as a quadratically constrained quadratic programming (QCQP) problem}$$

$$\max \theta : \theta \geq \frac{1}{2} \sum_{i=1}^{n} \alpha_i \xi_i (x_i \odot d) + \frac{1}{2} \alpha \|w\|.$$  \hspace{1cm} (4)

where $\alpha = [\alpha_0, \ldots, \alpha_n]$ is the dual variables and $A = \{ \alpha \mid \sum_{i=1}^{n} \alpha_i = 1, \alpha_i \geq 0, i = 1, \ldots, m \}$ defines the domain of $\alpha$. There are $O(mn^2)$ quadratic constraints in (4). However, the cutting-plane algorithm can be adopted to solve (4) easily. One advantage of FGM is that the learning process is only performed on those selected feature subsets and hence its training complexity can be greatly reduced to $O(mnR)$ [17], which is independent to the feature number $m$.

However, in the worst-case analysis of the cutting-plane algorithm, it still needs to select a fixed size of $r$ informative feature subset, which scales $O(mn)$ [17], [18]. Specifically, for simplicity, we only consider the squared hinge loss here. However, any smoothing loss, such as logistic loss and least-square loss, can be also applicable.
to identify the most violated constraint, FGM solves the following integer optimization problem:

$$\max \frac{1}{2} \sum_{j=1}^{n} a_j y_i (x_k \odot e^x_j) - \frac{1}{2} \sum_{j=1}^{n} c^+ j$$

(5)

where $c_j = \sum_{j=1}^{n} a_j y_i x_{ij}$. Problem (5) can be easily solved by sorting $c_j$s. The time complexity of this operation is $O(nm)$. Therefore, the linear complexity with respect to the number of features on the worst-case analysis restricts it for extremely high-dimensional problems, where the time complexity and the memory requirement for the sorting of $c_j$s will be unbearable.

In the worst-case analysis of FGM, one needs to compute the score for all the features. With small training examples, the score vector $e$ can be very sparse and we can use a sparse data structure to store it. But if the training size is very large, the sparsity of $e$ will no longer be retained. In the worst case, we should maintain a dense representation of $e$ with dimensions equal to that of whole dataset (or the mapped feature space). For example, if the dimension of dataset becomes extremely high (more than $10^{11}$, for example), it will be very time consuming for exact score calculation and sorting. In such case, we also possibly cannot afford enough memory to store the $e$ [5], and let alone to do feature rankings. In the following, we propose to solve this problem by developing a chain sampling method.

B. Chain-Sampling Strategy

Note that in the worst-case analysis of FGM, one needs to calculate

$$c_j = \sum_{i=1}^{n} a_i (y_i x_{ij} - \sum_{i=1}^{n} a_i x_{ij})$$

(6)

where $n^*$ and $n^+$ denote the number of instances for each class, respectively. After we obtain the feature scores, we need to sort all $|c_j|$s and find those $r$ features with the largest score as the most informative features.

In this paper, we intend to recover the features selected by FGM with sampling. Hence, without any ambiguity, we call those $r$ features in each most violated constraints selection as the ground truth features to FGM. Recall that a feature with smaller $|c_j|$ will have less chance to be selected in (5). Intuitively, these features have a higher chance to be dropped without loss of the generalization performance when doing feature score sorting and selection. It is easy to see that, given $a$, if most of the entries of the $j$th feature are close to zeros, with a high probability, its score $c_j = \sum_{i=1}^{n} a_i y_i x_{ij}$ would be very small and not likely to be the informative feature during the summation of $c_j$. Note that, each feature in $X$ can be considered as a random variable from the view of statistics [3]. Therefore, a dataset with high-dimensional features can be deemed as a multivariate random variable [3]. In addition, each feature $f$ follows an unknown distribution $p(f)$, and the instances regarding each feature can be considered as the observations. In our paper, we assume that, for sparse dataset, the feature with very few nonzero observations can be deemed as noninformative features. To better illustrate this issue, here we give a formal definition of the sparsity of features.

Definition 1: [19] A feature $f$ is $(\epsilon, \delta)$-sparse if $P(f = \mu | \epsilon) \leq \delta$. Here $\mu$ is a bias value\(^2\) and $\delta$ is the confidence level.

Given a predefined $\epsilon$, a feature with smaller $\delta$ is generally more sparse than the feature with larger $\delta$. The confidence level $\delta$ for different features are different. Based on Definition 1, we can categorize the features into dense (or nonsparse) and sparse features with a threshold $\delta_0$ under given $\epsilon$. That is, a feature is $(\epsilon, \delta_0)$-dense if $P(f = \mu | \epsilon) > \delta_0$, otherwise, it is $(\epsilon, \delta_0)$-sparse if $P(f = \mu | \epsilon) \leq \delta_0$. Let $Q$ be the number of features with $\delta > \delta_0$ (equivalently, the number of $(\epsilon, \delta_0)$-dense features) of the dataset. Similarly, we let $Q^+$ and $Q^−$ be the number of $(\epsilon, \delta_0)$-dense features for the positive and negative classes, respectively. Considering that a highly sparse feature is not likely an informative feature, we make an assumption about the features based on the above definition.

Assumption 1: A feature $f$ is informative only if it is $(\epsilon, \delta)$-dense. Otherwise we can simply drop this feature without loss of generality with probability $(1-\delta)$, where $\delta$ is the confidence level.

Based on the above assumption, we propose a new sampling scheme that tries to keep those dense features in a buffer and drop those sparse features, which can be considered as a truncated summation process. The basic idea is, we treat the summation of the positive and negative classes separately. Given a sampling size $G$, we create two buffers $c_+, c_- \in \mathbb{R}^G$ and $e \in \mathbb{R}^G$ to store the sampled features. Here $G$ satisfies $\max(Q^+, Q^−) \leq G \leq m$. Let $[\cdot + 1]$ and $[\cdot - 1]$ denote the truncated numeric addition and subtraction operator that keeps the $G$ largest scores with addition and subtraction operations, respectively. Then we can do a truncated summation process. Taking the case of the positive class as an example, we just keep the $G$ features with the largest scores $|c_j|$ in the buffer $c_+$ when the iteration evolves. For the ith sample with $a_i > 0$, given a new coming feature $j$, it will be exchanged and included in the buffer with some probability. Let $P_e$ be the exchange probability, given an unknown feature, it is

\[P_e(f(j) = a_i | f(j) = a_i) = \frac{|c_j|}{\mu} \]

$\odot$ denotes the truncated numeric addition and subtraction operator that keeps a maximum of $G$ numbers.

Algorithm 1 Chain-Sampling

1. Given $a$, $n^+$, $n^-$, sampling size $G$, and the initial score $c^+ = [0, 0, \ldots, 0] \in \mathbb{R}^n$ and $c^- = [0, 0, \ldots, 0] \in \mathbb{R}^n$.
2. For $t = 1 : n^+$ of positive class
3. $c^+ = [c^+ + e_k]_G$.
4. End
5. For $t = 1 : n^-$ of negative class
6. $c^- = [c^- + e_k]_G$.
7. End
8. $e = c_+ + e_−$.
9. Relinking with those features listed in $e$.
10. Find the $r$ features with the largest feature scores $|c_j|$ as the most-violated features.

\(^2\)If $f$ follows normal distribution, the bias value $\mu$ is the mean value. In most sparse cases, the bias value $\mu = 0$. If $\mu \neq 0$, we can subtract the entries of $f$ by $\mu$ to achieve the sparsity.
reasonable to set $P_e = 0.5$. The truncated summation process is stream-wise and started from an example with its $a_i > 0$. The general scheme of the sampling algorithm, which is called as the chain sampling, is depicted in Algorithm 1. Here we called it as chain sampling since this sampling scheme can be interpreted as a discrete time Markovian chain model [20], which will be detailed in the next subsection.

In Algorithm 1, the notation $c = e, c_1$ denotes the merging operation after we obtain $e$ and $c$. After that, we rerank the features captured by $e$. There remains one important issue to be addressed. That is, how to select the feature from the buffer that is to be exchanged with the coming feature during the summation? In this paper, the following two strategies are considered.

1) Type I: The features listed in the buffer have equal probability to be exchanged with the coming feature.

2) Type II: Only the feature with the least score is chosen to be exchanged. This is an extreme case to Type I.

When determining which feature can be exchanged, Type I equally takes all features in the buffer into consideration. Note in general, the $|c_i|$ value for dense feature increases faster than those sparse features due to the higher occurrences. In other words, the probability that the dense features will be chosen to be exchanged should decrease much faster than the sparse features. Hence Type I strategy is not optimal but the simplest method while Type II remedies the drawback of Type I, which only exchange the feature with the least score. The final issue is regarding to the stopping of FGM with sampling, where we may not find the exact most violated constraints and hence cannot guarantee the finite convergence property. As to this issue, we stop the FGM after predefined iterations in this paper.

C. Implementation Details and Complexity Analysis

In our sampling method, for Type I strategy, we maintain a hash map as the buffer to store the scores of the selected features and an additional vector to store the keys. The complexity of hash operations is $O(1)$. Hence its complexity is $O(nG_s)$, where $G_s$ denotes the maximum number of nonzero among all instances and $r'$ denotes the number of support vectors. For Type II, besides the hash map, we use a minheap, of which the root is the minimum value, to store the scores based on their scores. The complexity of inserting and deleting an element is $O(\log(G))$ while the search for the minimum node needs $O(1)$. The construction of minheap takes $O(G \log(G))$. In addition, the complexity of updating the feature score for each feature in the heap is about $O(G + \log(G))$. Finally, the reranking process will take $O(nG_s + (G \log(G))$. Hence the total complexity is about $O(nG_s \log(G) + nG(G + \log(G)))$. In the training, we make a full copy of the selected features, hence we need $O(nr)$ time for training and additional $O(nr)$ memory for storage.

D. Performance Analysis

The criterion for CRS is how it preserves the pairwise distance of the original dataset. However, from the perspective of FGM on feature selection problem, a good sampling method should cover all the $r$ ground truth features in the worst case analysis of FGM. Hence, the sampling for feature selection should be more rigorous than pair-wise distance preserving.

For chain sampling, on what probability it can cover the desired ground truth features with given sampling size $G$ is the major concern. The analysis of the Type II sampling becomes even more difficult. Hence our analysis is based on the Type I sampling method. To model the buffer used in Algorithm 1, we introduce a buffer model in Fig. 1(a) to help the analysis of the performance of the proposed sampling scheme, where $G$ is the buffer size and $Q$ is the number of $(\epsilon, \delta_m)$-dense features. Here we drop the sign from $Q$ for simplicity. Typically, for a buffer $c$, suppose there are $i$ informative features are not covered by $c$, then there are still $Q - i$ informative features in $c$, as shown in Fig. 1(b). With this buffer model, the sampling in Algorithm 1 can be interpreted by a discrete time Markovian birth–death process [20] from queueing theory, as shown in Fig. 1(b). Due to the complexity of the data statistics, a rigorous analysis of the model is impossible. Therefore, we will present the model assumptions and detailed discussions about the buffer model and the birth–death model.

To make the model analysis tractable, first, we assume that all the sparse features have the same statistics. Without loss of generality, let $\delta_s$ denote the average confidence level of the $(\epsilon, \delta_s)$-sparse features and $\delta_{m}$ be the average confidence level of the $(\epsilon, \delta_m)$-dense features. We also define

$$n = \frac{\epsilon m}{\epsilon m - Q\delta_m}$$

(7)

to measure the ratio between the $(\epsilon, \delta_s)$-sparse and $(\epsilon, \delta_m)$-dense features, which means that the proportion of the nonzeros entries generated from $(\epsilon, \delta_s)$-sparse features to the $(\epsilon, \delta_m)$-dense features. Second, the chain-sampling method can definitely cover all the ground truth features if and only if it can cover all the $(\epsilon, \delta_m)$-dense features in the buffer.

In the chain sampling, we treat the positive class and negative class separately. Without loss of generality, we only focus on the positive class. The analysis of the negative class can be conducted in the similar way.

Note that in chain sampling, even we cannot exactly cover all the $(\epsilon, \delta_m)$-dense features, we still have the chance to cover the $r$ features. Let $P_r(\epsilon)$ be the probability that all the $r$
ground truth features are covered by $c_i$, when there are $i$ informative ($c, h_o$)-dense features not in $c_i$. If we assume that each ($c, h_o$)-dense feature listed in the buffer have equal chance to be the ground truth feature, then we have

$$P_{c}(r|i) = \binom{Q - r}{i}$$

(8)

where $0 \leq i \leq Q - r$. The above result is derived as follows. Suppose there are $i$ informative features not are covered by $c_i$, then there are still $Q - i$ informative features in $c_i$. In addition, the features listed in the buffer have equal chance to be the ground truth features as they share the same statistics. Then $P_{c}(r|i)$ is equal to the probability that we randomly pick $i$ ($c, h_o$)-dense features that do not include those ground truth features from $Q - (c, h_o)$-dense features. Hence, we can obtain

$$P_{c}(r|i) = \frac{\binom{Q - r}{i}}{\binom{Q}{i}}$$

(9)

After the sampling is terminated, there are only $Q^* + 1$ states for $c_i$, denoted by $0, \ldots, i, \ldots, Q^*$. Here $i$ means there are $i$ dense features not in $c_i$. From the law of total probability, we have

$$P_{c} = \sum_{i=0}^{Q^*} \pi_i P_{c}(r|i)$$

(10)

where $\pi_i$ is probability that there are $i$ informative ($c, h_o$)-dense features not in $c_i$. Similarly, we have $P_{c}^r$ as

$$P_{c}^r = \sum_{i=0}^{Q^*} \pi_i P_{c}^r(r|i)$$

(11)

Let $P_r$ be the probability that $c$ can cover the ground truth features. Because $c$ merges $c_i$ and $c_{i+1}$, then

$$P_r = P(c_i \cup c_{i+1}) \geq \max(P_{c}, P_{c}^r)$$

(12)

In other words, the chain sampling can exactly cover the $r$ most violated features with probability no less than $\max(P_{c}, P_{c}^r)$.

Now we turn to the derivation of $\pi$ and $\pi_r$. The modeling using a discrete time birth-death process permits computation of these two quantities. We define $S$ to be the state space of the birth-death process where $S = \{0, 1, \ldots, Q^*\}$ and a state, say state $i$, represents that the system state that there are $i$ informative ($c, h_o$)-dense features not in $c_i$.

Consider state $i$, based on the process evolution, in the next time slot, the probability that a nonzero feature $f$ is from the ($c, h_o$)-dense feature set and not in $c_i$ is

$$P(f \in Q^*, f \notin c_i) = \frac{\binom{i}{n_Q^*}}{\binom{i}{n_Q^*} + \binom{n_Q^* (m - Q^*)}{i}}$$

(13)

which is equal to the probability that we randomly pick one feature $f$ from the $i$ ($c, h_o$)-dense features from all nonzero features. Here, $\binom{i}{n_Q^*} + \binom{n_Q^* (m - Q^*)}{i}$ counts the number of nonzero elements from the rest $n$ samples.

In addition, the probability that a ($c, h_o$)-sparse feature in $c_i$ is chosen to be exchanged is $\frac{i}{\binom{i}{n_Q^*}}$, where $(G - Q^* + i)$ is the number of ($c, h_o$)-sparse features captured in buffer $c_i$, as shown in Fig. 1(a). Hence, $\frac{i}{\binom{i}{n_Q^*}}$ describes the probability that we randomly pick a ($c, h_o$)-sparse feature from $c_i$ when $c_i$ is at state $i$. Based on the above results, we can obtain the transition probability from state $i$ to $i - 1$ as

$$P_{c}(i, i - 1) = \frac{i}{\binom{i}{n_Q^*}} \frac{G - Q^* + i}{G}$$

(14)

where $P_r$ is probability that an exchange will happen. Using the similar analysis, we can determine the transition probability for $i$ to $i + 1$ to be

$$P_{c}(i, i + 1) = \frac{m - G - i}{\delta_Q^* (m - Q^*) + \delta_Q^*(m - Q^*) + 1}$$

(15)

which describes the probability that a ($c, h_o$)-dense feature in $c_i$ is exchanged by a ($c, h_o$)-sparse feature not included in $c_i$. Here $(m - G - i)$ denotes the number of ($c, h_o$)-sparse features not captured by $c_i$ while $(Q^* - i)$ represents the number of ($c, h_o$)-dense features in $c_i$. Accordingly, we have

$$P_{c}(i, i + 1) = 1 - P_{c}(i, i - 1) - P_{c}(i, i + 1)$$

(16)

which means there is no arrival in $c_i$ at this time slot.

Define the ratio

$$\rho_i = \frac{P_{c}(i, i + 1)}{P_{c}(i, i - 1)} = \frac{(m - G - i) \delta_Q^* (Q^* - i)}{\delta_Q^* (m - Q^*) + 1}$$

(17)

Let $\pi_i, i = 0, 1, \ldots, Q^*$, denote the steady state probability for the birth-death process. Then we have $\pi_i = \pi_0 \prod_{i=1}^{Q^*-1} \rho_i$ and $\sum_{i=0}^{Q^*} \pi_i = 1$. By solving these equations, we have

$$\pi_0 = \frac{1}{\sum_{i=0}^{Q^*} \prod_{i=1}^{Q^*-1} \rho_i}$$

(18)

Fig. 2 shows that the numerical results of above models. Note with

$$\eta = \frac{(m - G - i) \delta_Q^*}{\delta_Q^* Q^*}$$

(19)

$$\rho^* = \frac{(m - G - i) \delta_Q^* (Q^* - i)}{\delta_Q^* (m - Q^*) (Q^* - i)}$$

(20)
In the tests, we set $Q^* = 200$ and $m = 10^6$ and measure the performance regarding different sampling size $G$ under given $r$ and $\eta$, where $G$ varies from 250 to 5,000. In the first test, we fixed $r = 5$ and varied $\eta$ in the range $[0.01, 0.05, 0.10, 0.50, 1.00]$. The results are shown in Fig. 2(a). We can see that with given $\eta$, the chain sampling can identify the ground truth features with very high probability with large enough $G$. In general, larger $\eta$ means less sparse dataset and requires larger sampling size. The performance of chain sampling is also affected by $r$. To show this, we fixed $\eta = 0.10$ and varied $r$ within range $[2, 4, 6, 8, 10]$. The results are shown in Fig. 2(b). We can see that, in general, larger $r$ requires larger $G$. But with large enough $G$, the sampling algorithm can obtain good results under different level of $r$. The results of high coverage probability indicate that our proposed chain sampling remains effective in feature selection.

E. Speed-up Using Sampling on Instances

Recall that the complexity of the chain-sampling method is $O(nG, \log(G) + nG \log(G))$. When $n$ is very small, in other words, only a small number of support vectors are obtained, the proposed sampling method can be very efficient; otherwise, it is still very computationally expensive. Fortunately, the central limit theorem allows us to further reduce the computational cost by using part of support vectors.

Note in FGM, we just need to compute

$$c_j^r = \sum_{i=1}^{n} a_i x_{ij}$$

(21)

and

$$c_j^* = \sum_{i=1}^{n} a_i x_{ij}$$

(22)

for the $j$th feature. Here we only consider the $j$th feature for the positive class. Let $x_j = a_j x_{ij}$ and we get

$$c_j^r = \sum_{i=1}^{n} a_i x_j$$

(23)

Suppose that the expectation of $x_j$, denoted by $\mu$ and the standard variation of $x_j$, denoted by $\sigma$, exist and $\sigma$ satisfies $\sigma < \infty$. Then $x_j$ follows the central limit theorem [21]. Let

$$S_n = \sum_{i=1}^{n} x_j$$

(24)

and

$$S_n^* = \frac{S_n - n\mu}{\sqrt{n}\sigma}$$

(25)

where $x_j$ is a sample, then the calculation of $c_j^r$ and $c_j^*$ becomes a mean estimation problem. In addition, there is only an equality constraint imposed on each feature, namely

$$\sum a_i = 1.$$

(26)

Hence, when $n$ is very large, $x_j$ can be considered as i.i.d distributed random variable. Further, from the central limit theorem, $S_n$ converges to the Gaussian distribution $N(\mu, \frac{\sigma^2}{n})$, no matter what kind of distribution $x_j$ follows. The following theorem says that with $O(n^\frac{1}{2})$ sample size, we can obtain very good approximation to the ground truth. Hereafter we will let $S$ denote the sample size on support vectors.

**Theorem 1:** [21] For the mean estimation problem, we can obtain the desired accuracy $\epsilon$ under a predefined confidence level $\delta$ with the sample size $O(n^\frac{1}{2})$.

Note that the sampling on instances level is done on the support vectors rather than on all instances. Specifically, given the approximation error $\epsilon$, from Theorem 1, we can easily do the sampling on instances by randomly choosing a small portion of support vectors, where the sample size is proportional to $\frac{1}{\epsilon^2}$.

III. EXPERIMENTS

A. Experimental Settings

In the experiments, we will verify the effectiveness of the proposed sampling techniques on four-medium dimensional dataset, namely `text`, `aut-avn`, `pcmac`, and `real-sim`, and two large and high-dimensional datasets, namely `rcv1.binary` and `URL0`. Among them, `real-sim` and `rcv1.binary` are from LIBSVM website, while `aut-avn` and `pcmac` are from SVMlin website. The URL dataset is from the Anonymized 120-day subset of the ICML-09 URL data [22] and studied by [17]. For the convenience of experiments, we manually split the datasets into training and testing part. The details of these datasets are listed in Table I, where $m_{p,0}$ denotes the dimension of their two-degree polynomial mappings, $n_{train}$ and $n_{test}$ denotes the number of training examples and testing examples, respectively. All the experiments are performed on Intel(R) XEON(R) of 24 GB memory with 64-Windows server 2003 operating system.

We evaluate the performances on these datasets for the following feature selection schemes:

1) FGM with sampling (SFGM);
2) FGM with CRS sampling (FGM-CRS);
3) two-degree polynomial FGM (PFGM);
4) PFGM with sampling (SPFGM);
5) PFGM with simple random sampling (SPFGM-RAND);
6) L1 regularized SVM (L1-SVM) with two-degree polynomial mapping (L1-PSVM) [5], which is the state-of-the-art L1-SVM implementation.

For CRS sampling, we did not do random permutation and simply recorded the features with $ID \leq D_r$. This is valid because the original feature order is a permutation of itself. The $D_r$ for different datasets are shown in Table II, from where we can observe that $D_r$ varies a lot for different datasets and larger $D_r$ means CRS will cover more features. Note $D_r$ for some datasets can be very small (even less than $r$ on `real-sim`). In such cases, we alternatively set $D_r = r + 100$ for FGM-CRS to make FGM valid. In the simple random sampling, we randomly choose 100,000 features and find $r$ features among them to maximize problem (5). In all large-scale experiments,
we set the trade-off parameter $C = 5$ for FGM-based methods. FGM without any sampling is adopted as the baseline method.

The following experiments are organized as follows: In the first experiment, we will show how well the proposed methods can approximate to the baseline methods from the perspective of the objective values. In the second experiment, we will do the comparison on the four medium-dimensional datasets. Especially, we will compare the performance with $\ell_1$ regularized SVM with polynomial mapping on these datasets. In the third experiments, we will conduct the comparison on the last two high-dimensional datasets, where only our sampling methods can work with polynomial mappings. In the forth experiment, we will show the performance of sampling on support vectors based on rcv1 dataset. In the final experiments, we compare our method with three related methods for nonlinear feature selections.

B. Approximation Quality

For the proposed sampling scheme, two types of sampling methods are discussed in the previous sections. A major concern is how well they can approximate to the baseline method. The objective value of problem (4) is a direct criterion for comparison. Hence in the first experiment, we take URL0, which is with very high dimensionality and good sparsity, for case study. Here only the linear kernel is considered and FGM with several mentioned sampling methods are adopted for comparison. For the experiment settings, we set $r = 20$, the sample size on support vectors $S = 1000$ and vary $G$ in the range of $\{1000, 3000, 5000, 8000, 10000, 30000\}$. We stop the algorithm after 7 iterations.

The comparisons of the two sampling methods regarding the objective values of problem (4) are shown in Fig. 3. From Fig. 3, we can easily obtain that the two proposed methods can both have good approximation to FGM without any sampling, namely the baseline method. And, the proposed methods are much better than FGM with random sampling. Specifically, for Type I sampling, when $G \geq 3 \times 10^3$, the approximation is quite close to the baseline method. While for Type II sampling, the approximation can be very good when $G \geq 1 \times 10^3$. Also, Type II sampling is better than Type I sampling for it can reserve those highly informative features and not likely exchanges them. Hereafter we will use suffix I and II to SFGM and SPFGM to distinguish the two proposed sampling methods. From the comparison, we can find that based on FGM, the CRS sampling can also achieve very good approximation to the baseline method (FGM), which is mainly because the $D_i$ for URL0 dataset is relatively very large that can cover most of the informative features. Among all the methods, the simple random sampling shows the worst performance, which demonstrates that the random sampling is not suitable for informative feature discoveries. In general, a larger $G$ leads to better approximation. However, the computational cost will increase when $G$ becomes large. We will further study this issue in the latter experiments.

C. Performance Comparison

In the second experiment, we will show the effectiveness of our method with polynomial feature mappings on the medium-dimensional datasets, where a full feature mapping (no sampling) can be used for L1-SVM and FGM. For the convenience of comparison, here we only consider the case of polynomial mapping. Note that the four medium datasets are usually adopted for semisupervised learning, where some nonlinear structures (such as manifold structures) are supposed to be existed. For the polynomial feature mapping, we set $\lambda = 1$ and $\tau = 0.25$ to impose more importance on the nonlinear features. For FGM-based methods, we can use different $r$ to control the desired sparsity with fixed $C$. For L1-SVM methods, with the tuning of the trade-off parameter $C$, we can achieve different sparsity levels. To conduct the comparison, in the experiments, we fix $G = 10000$ and $C = 5$, and vary the value of $r$ for FGM-based methods to achieve different number of features. For L1-SVM, we vary $C$ to obtain different number of features. For fair comparison, we record the corresponding testing accuracy and feature selection time with different number of selected features in Fig. 4 and Fig. 5, respectively.

We can observe from Fig. 4 that: First, with given a buffer size, FGM with the two proposed sampling methods can obtain competitive or better testing accuracy compared with FGM and L1-SVM with full polynomial mappings. Specifically, FGM with proposed sampling methods shows better performance than L1-PSVM under relatively small sparsity levels. Second, compared with FGM with full polynomial mappings (PFGM), ideally the proposed sampling methods should be closer to FGM, the better. This property is demonstrated from aut-ant and real-sim for the Type II sampling methods (SPFGM-II). However, as we can see, SPFGM-II is slightly better than FGM on text and pcmac. One possible reason can be that some irrelevant features (which can be selected by FGM) are filtered out when doing the sampling process. Third, as to the two types of sampling methods, SPFGM-II tends to preserve the possible informative features and not to change them, and so it is generally better than SPFGM-I.

The efficiency of the proposed sampling method can be observed from Fig. 5. From Fig. 5, SPFGM-I and SPFGM-II are much more efficient than PFGM and L1-PSVM on all datasets, which validates the efficiency of the proposed method. Especially on aut-ant and real-sim, which are with relatively high dimensions, both of the proposed sampling methods can be about ten times faster than L1-PSVM. That is
D. Performance on Very High-Dimensional Problems

In the third experiment, we compare the performance of the proposed methods under different buffer sizes on two high-dimensional datasets. Here both linear kernel and polynomial kernel mappings are considered. From Table I, the dimension for the polynomial mappings on the high-dimensional datasets can be extremely high, and we cannot obtain the results of PFGM and L1-PSVM from our experiments. For the experimental settings, here we set $\lambda = 1$ and $\tau = 1$, and do not perform any parameter selections. In addition, we fix $r = 20$ for both rcv1 and URL0 dataset. We set the number of support vectors to $S = 1000$ for each class and vary the buffer size $G$ in the range of $\{500, 800, 1000, 3000, 5000, 8000, 30000\}$. Finally, we set the sampling size for simple random sampling to $100000$.

The prediction accuracy and the feature selection time versus different buffer sizes are reported in Fig. 6 and Fig. 7, respectively. From Fig. 6, we can see that for SFGM-I and SFGM-II, with a relatively small buffer size $G$, very close results to FGM can be achieved and the feature selection time is comparable. For SPFGM, it can obtain a good approximation to the ground truth may not (FGM) with an acceptable computational cost to SFGM when $G$ is not very large. It proves that with a not very large sampling size, the proposed sampling can efficiently deal with extremely high-dimensional problems. Specifically, in our experiments, we can deal with problems of scale $O(10^{12})$ within several hundred seconds. In general, as from Fig. 6, a buffer with larger size can produce better predication accuracy for it has a higher chance to capture the informative features. Compared with Type II sampling, Type I sampling usually needs a much larger buffer size to achieve the similar performance. However, the time complexity is highly related to the buffer size $G$. When $G$ becomes very large, the time complexity would quickly increase.

We observe that the feature selection time on URL0 decreases with increasing $G$, which is apparently different from the trend on rcv1. Recall that the training process of FGM includes the most-violated constraints selection and the MKL training which adopts LIBlinear as the SVM solver [17]. When dealing with higher dimensional problem, as suggested in Section II, a small buffer size $G$ is more likely to miss the informative or good features. Specifically, because the dimension of URL0 is much higher than rcv1, the buffer for URL0 might contain relatively more noninformative (or nondiscriminative) features when the buffer size is small. On the other hand, nondiscriminative features would lead to longer SVM feature selection time, resulting in a higher computational cost on MKL training. Accordingly, a larger buffer size can reduce the whole computational cost for it to catch more informative features and hence the complexity of MKL training can be reduced. For FGM-CRS methods, as expected, the performance is not stable. The reason is that the former $D_s$ features in general cannot cover all possible informative features when it is very small. On URL0, FGM-CRS can achieve very close results to FGM for to say, by using the proposed sampling methods, the training efficiency can be greatly improved. It is worth noting that the most challenging problem for explicit feature mapping is the unbearable memory requirement. For example, in our experiment, we can conduct the experiments on real-sim dataset. However, we cannot afford enough memory for L1-PSVM and PFGM on rcv1 dataset even it is in the same scale on dimension as real-sim. On the contrary, the two proposed sampling methods can successfully alleviate the memory requirement.
that most of the informative features are luckily covered by the former $D_s$ features. For simple random sampling, because we used a relatively large sample size, the performance is acceptable. However, for general problems, random sampling cannot guarantee to find the desired features.

Note that our algorithm is mainly based on the sparse assumption of the dataset. To verify whether the proposed sampling scheme for feature selection method can obtain denser representation of the dataset, in Table II, we recorded the sparsity ratio of the original dataset, denoted by $\delta$, and the sparsity ratio of the obtained feature subset for all datasets, denoted by $\delta_f$. The comparisons show that the feature selection can greatly improve the density of dataset.

E. Performance of Sampling on Instances

In this experiment, we test the influences of the sampling on instances. As we show previously, we can use a relatively small sample size on support vectors to obtain good enough approximations. To verify this, we fixed the buffer size $G = 5000$ and varied the sample size on support vectors $S \in \{500, 800, 1000, 3000, 5000, 8000, 10000, 30000\}$. Here we only study rcv1 dataset since it has a large number of instances. The testing accuracy and the training time are shown in Fig. 8(a). From Fig. 8(a), we can see that with very small number (500 for example) of support vectors, SFGM-I(II) and SPFGM-II can obtain very close results to the baseline methods (FGM) in terms of testing accuracy. SPFGM-II shows even better performance than FGM. From Fig. 8(a), SPFGM-I does not perform well under this setting. This is mainly caused by the relatively small buffer size. For the Type I sampling, usually we need a large enough buffer to include the informative features. We can also obtain from Fig. 8(b) that the computational cost will increase with more support vectors. However, as from Fig. 8(a), more support vectors does not significantly improve the accuracy on both types of sampling methods, which demonstrates that we can use a relatively small number of support vectors in FGM while maintaining good performance. Our experiments on rcv1 verify the validity of the sampling on instances.

F. Performance Comparisons with Other Related Methods

In this experiment, we compare the performance of the proposed method with three related feature selection methods for small scale feature selection tasks [9], [10], [13]. For simplicity, we only study SPFGM-II. The similarity preserving feature selection (SPFS) [9] is a wrapper-filter feature selection method.
performance than the three baseline methods on text and pcamc dataset. On leukemia dataset, although the proposed method does not show significant improvements, it obtains competitive performance over other methods. The reason is that, leukemia is a linearly separable dataset. Therefore, the incorporation of the nonlinear information does not necessarily improve the performance.

IV. Conclusion

In this paper, a novel sampling scheme was proposed for feature selection on large-scale and extremely high-dimensional sparse problems. Based on the feature generating framework, the proposed algorithm included the sampling both on features and instances (support vectors), which made FGM applicable for extremely high-dimensional and large-scale problems. Theoretical analysis regarding the performance of the proposed method based on the birth–death process and central limit theory was presented. Finally, the empirical studies on a wide range of real-world datasets verified its efficiency and effectiveness.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their insightful comments and suggestions, which have greatly improved this paper.

REFERENCES


Shutao Li (M’07) received the B.S., M.S., and Ph.D. degrees in electrical engineering from the Hunan University, Changsha, China, in 1995, 1997, and 2001, respectively. He joined the College of Electrical and Information Engineering, Hunan University, in 2001. He was a Research Associate with the Department of Computer Science, Hong Kong University of Science and Technology, from May 2001 to October 2001. From November 2002 to November 2003, he was a Post-Doctoral Fellow at the Royal Holloway College, University of London, and was with Prof. John Shawe-Taylor. During April 2005 to June 2005, he was with the Department of Computer Science, Hong Kong University of Science and Technology, as a Visiting Professor. Currently, he is a Full Professor with the College of Electrical and Information Engineering, Hunan University, Changsha, China. He has authored or co-authored more than 160 refereed papers. His current research interests include computational intelligence, information fusion, pattern recognition, and image processing.

Dr. Li has served as a member of the Neural Networks Technical Committee from 2007 to 2008. He won two Second-Grade National Awards at the Science and Technology Progress of China in 2004 and 2006.

Dan Wei (M’06) received the B.S., M.S., and Ph.D. degrees from Hunan University, Changsha, China, in 2006, 2008, and 2012, respectively. His current research interests include image processing and pattern recognition.