Speeding Up Joint Mutual Information Feature Selection with an Optimization Heuristic

Heng Liu and Gregory Ditzler
The University of Arizona
Dept. of Electrical & Computer Engineering
Tucson, AZ 85721
{hengl, ditzler}@email.arizona.edu

***Abstract*—Feature selection is an important pre-processing stage for nearly all data analysis pipelines that impact applications, such as genomics, life & biomedical sciences, and cybersecurity. These application-driven fields rely on feature selection to not only build classifiers but also understand the data by trying to discover knowledge. Furthermore, these dimensionality reduction methods address the curse of dimensionality as well if classification is the end objective. Feature selection is formally defined as the process of optimizing the minimum subset of features that allow for the maximum predictive power. In this work, we address the problem of feature selection by using information theory. Many of the information-theoretic approaches perform a greedy forward optimization, however, the overall efficacy is a concern since most of the existing greedy algorithms only work in a centralized fashion (e.g. single sequential thread). Recent work has unified approximately 20 years of Information-theoretic feature selection works into a unified conditional likelihood maximization framework, which showed that the joint mutual information maximization (JMI) objective is generally the best option (i.e., empirically evaluated on many datasets). Unfortunately, JMI is infeasible on extremely high dimensional datasets. In this contribution, we propose a semi-parallel implementation of JMI that takes advantage of the fact that multiple features can be selected at the same time after observing a pattern in the greedy forward search. This work also bridges the gap between greedy feature selection and parallel feature selection, which can deliver significant speedup for large scale feature selection tasks. We evaluated the efficacy of the proposed approach with the traditional implementation of JMI using a greedy forward selection on several UCI datasets. The experimental results demonstrate that the proposed approach is capable of selecting the same features that traditional JMI returns; however, the features are selected in a fraction of the time.**

I. Introduction

The curse of dimensionality has increasingly been a key problem in machine learning and data mining caused by the abundant data that need to be analyzed from application-driven fields, such as bioinformatics, metagenomics, and business analytics [1]–[3]. The main impact of high dimensionality on the computational complexity of a learning algorithm can be alleviated by performing feature selection/dimension reduction. Unfortunately, not all of the variables that are collected from a data source are informative for making a prediction on a class. High dimensional datasets usually contain numerous features that are either: (a) redundant with other features; or (b) simply irrelevant to an outcome that is of interest (i.e., positive or negative). Thus, practitioners can take advantage of feature selection approaches to greatly reduce the dimensionality without losing too much predictive ability.

Feature selection is typically applied in a centralized manner (i.e., SVM-RFE [1], LASSO [4]). This type of implementation is infeasible for datasets that are amounting to TBs of data. Actually, it has been increasingly seen as a difficult challenge for feature selection algorithms to acquire knowledge from large datasets due to various issues coming along, such as high dimensional feature size, dynamic load balancing and parallel file system, etc.

At the same time, recent progress in machine learning has been driven considerably by the explosion of the data and the low-cost computing system (i.e. low price CPU and storage). New computing platforms provide us new opportunities of developing innovative algorithms, examples of such platforms include but is not limited to: Distributed Computing Systems(e.g. Apache Spark, Hadoop MapReduce), High-Performance Computing and GPU-accelerated computing, etc. Many of these platforms have developed distributed machine learning libraries, such as Hadoop Mahout and Apache Spark MLlib. Unfortunately, feature selection remains an open problem in parallel and distributed computing.

Feature selection methodologies can be broadly categorized into three groups: embedded, wrapper and filter approaches [5]–[7]. Wrapper approaches exploit the feature subspaces by using the cross-validation accuracy of particular learning models as a measure of utility for a feature subspace that is of interest [8], [9]. SVM-RFE is one of the most famous wrapper approaches that recursively remove feature candidates while constructing a SVM model [1]. Embedded methods perform feature selection by cooperatively optimizing the learning model and the feature selector. One example is the LASSO algorithm, which is minimizing the sum of squared errors for a linear regression model while including a penalization of the ℓ_1-norm of the feature parameters [4]. The ℓ_1 penalization brings about a sparse solution, therefore conducts a feature selection within a linear model. Both embedded and wrapper methods use a score (i.e., cost) of a particular learning model. Filter approaches, however, incorporate a classifier-independent criteria (e.g., correlation, K-L divergence and information theory, etc. [8], [10], [11]) to evaluate a feature subset. In this work, we concentrate on filter approaches using

978-1-5386-2726-6/17/$31.00 ©2017 IEEE

information theory since they do not need to build a classifier and can deliver significant advantages w.r.t. the computational cost.

Filter approaches are usually favored from computational perspective due to it being classifier independent; however, the search for best solution must be exhaustive. Obviously, an exhaustive search is NP-hard, which becomes quickly computationally intractable even for low dimensional datasets [5]. Thus, many filter methods (e.g., joint mutual information maximization (JMI) [12], maximum relevancy-minimum redundancy (mRMR) [13] etc.) balance the quality of the reduced feature set with the searching complexity by implementing a greedy optimization. For example, forward selection and backward elimination are two of the most popular search strategies with filter-based methods [1]. It is worth commenting that these greedy approaches can be sub-optimal in some cases and run in polynomial time [1]. Among the aforementioned filter methods, Brown et al. demonstrated that JMI maximization could achieve the best trade-off between feature relevancy and redundancy [8], [14]; however, JMI's greedy optimization is scaling quadratically, which prevents successfully applying JMI on high dimensional datasets.

In this contribution, we present a novel semi-parallel information theoretic feature selection algorithm that aims to speed up JMI maximization's greedy searching by using a coded strategy to create redundancy among parallel workers and by deriving a theoretical boundary to minimize the searching complexity of optimal features. The proposed approach was benchmarked on 8 datasets against the original centralized JMI maximization where we can show that our approach can recover the same feature subset as JMI at a fraction of the time.

This paper is organized as follows: Section II reviews some state-of-the-art algorithms, Section III describes our proposed algorithm for decreasing the computational cost of JMI and we report our experimental results in Section IV. Section V draws the conclusion of our contribution and provides some future works. Table I contains commonly used mathematical notations throughout this work.

TABLE I: List of commonly used mathematical notation.

$\mathcal{D}$	Dataset
i, j	Index for feature set
t	Arbitrary time
r	Time increment
N	Number of samples
F_i	i^{th} feature
$\mathcal{F}$	Feature complete set
$\mathcal{S}$	Selected feature subset
$\mathcal{F} \backslash \mathcal{S}$	Candidate feature set
Y	Class label
$X_t^{(n)}$, $T_t^{(n)}$	Ordered sequence of length n at time t
n	length of a sequence
X_t^p, X_t^q / T_t^p, T_t^q	p^{th}, q^{th} element of sequence $X_t^{(n)}$ / $T_t^{(n)}$
p, q	Index for sequence
M	Number of parallel workers
W	Converged Interfering Zone size
$\mathcal{P}$	Warm start
$DistF$	Distributed feature indices
$DistS$	Distributed feature JMI score
$Flag$	Vector of identification results
$Intrs$	Vector of intruders

II. Related Works

Filter-based feature selection approaches use a classifier-independent scoring criteria (e.g., mutual information [15], Laplacian score [16], or K-L divergence [17]) to identify if a candidate feature is relevant. This scoring may also select features that are independently informative with the class; however, this is insufficient simply because independent usefulness does not guarantee that the features selected will not be redundant [5]. Thus, maximizing joint informativeness and minimizing redundancy are both at the core of feature selection [18]. It is also important to note that the design of the scoring function is quite important to the definitions of relevancy and redundancy.

Lewis et al. proposed mutual information maximization (MIM), which evaluates each feature's relevance using straightforward criteria [19]. The criteria for MIM simply identifies the k best features based on the largest mutual information between a feature ($F_i \in \mathcal{F}$) and class variable (Y). Unfortunately, MIM fails to consider any level of redundancy within the selected features, which is suboptimal in the sense that there could be redundant information in the subset. Not considering the redundancy within the selected feature set can also increase the computational complexity of a classifier. It is generally accepted that an informative set of features' redundancy should be minimized. Hence, Battiti et al. proposed an objective to balance the tradeoff between relevance and redundancy by using the mutual information feature selection (MIFS) objective:

$$J_{MIFS}(F_i) = I(F_i;Y) - \gamma \sum_{F_j \in \mathcal{S}} I(F_i;F_j) \qquad (1)$$

where $I(F_i;Y)$ is the mutual information between two random variables. Note that the latter term in (1) removes the redundancy from the cost function for F_i (i.e., the candidate feature) and F_j (i.e., a feature that has already been selected). Feature selection with the scoring function (1) is maximized using a similar procedure as the greedy algorithm in Figure 1.

Cheng et al. proposed CMIFS because the MIFS objective function is biased since the objective ignores the class label and the interaction information in the redundancy term [20]. Peng et al. addressed the redundancy by selecting features using an objective similar to (1) [13]; however, $\gamma = 1/|\mathcal{S}|$, which again does not consider any level of redundancy with the class label.

Finally, Yang and Moody [12], also later Meyer [21], presented a different perspective of the tradeoff between redundancy and relevancy by using the joint mutual information maximization(JMI) objective, which depends on maximizing the complimentary information of candidate features. More-

over formally, this expression is given by:

$$J_{JMI}(F_i) = \sum_{F_j \in \mathcal{S}} I(F_i, F_j; Y) \tag{2}$$

where $F_j \in \mathcal{S}$ is the feature that has already been selected via the JMI maximization. Feature selection with the scoring function (2) is maximized using the greedy algorithm in Figure 1.

The JMI objective function provides a more intuitive perspective of selecting features in a way that balances relevancy and redundancy without free parameters. Other approaches, such as MIFS, require quite a bit of knowledge *a priori* to select features. For example, choosing different γ values for MIFS could cause significantly different results. Thus, the JMI objective simply avoids the heuristic choosing of the trade-off parameter (i.e., γ) by maximizing the complimentary information. Brown et al. recently presented a unified framework on conditional likelihood maximization that concludes various works published since 1992 [8]. Moreover, they found that the JMI objective function is generally the better objective w.r.t. accuracy and stability. Unfortunately, JMI's greedy optimization is scales quadratically as a function of the number of features, which is the primary issue addressed in this manuscript.

Distributed/parallel approaches for information-theoretic feature selection have remained an under-explored area of research. Given the convenient parallel/distributed computing platforms and the rapidly scaling machine learning problems emerged in the era of Big Data, feature selection approaches should take advantage of processing multiple feature subsets simultaneously. There are generally two methods for partitioning/distributing the data into different subsets: (a) vertically by features, (b) horizontally by instances, after all subsets have been processed in parallel, a merging procedure of multiple subsets is performed to find the global optimal solution [22], [23].

In [22], Cañedo et al. describe a distributed information-theoretic feature selection methodology using a vertical partition strategy. Their contribution takes advantage of a heuristic to avoid calculating the redundancy between different subsets on workers by aggregating features with similar relevancy in the same subset then merge multiple subsets by evaluating the accuracy of the learning model. Zhang et al. in [24] proposed a parallel feature selection inspired by the group testing theory. This procedure consists of a collection of randomized tests in parallel. Each test corresponds to a subset of features that a scoring function may be applied to measure the quality. Cañedo et al. in [25] evaluated and compared the effect of distributing with different feature filters, merging with different aggregation methods.

In this work, we propose a vertical semi-parallel filter methodology, which is guaranteed to end up with the same result as the centralized JMI maximization by applying the JMI objective in parallel, while can avoid the problematic quadratic complexity.

Input: Feature set $\mathcal{F}$, integer n
Initialize: Feature set $\mathcal{S} = 0$, integer k
for $t = 1, \ldots, n$

- Choose the feature i such that

$$F_i = \arg\max_{i \in \mathcal{F}} \sum_{F_j \in \mathcal{S}} I(F_i, F_j; Y)$$

- $\mathcal{F} \leftarrow \mathcal{F} \backslash F_i$
- $\mathcal{S} \leftarrow \mathcal{S} \cup F_i$

Fig. 1: Greedy algorithm for maximizing JMI.

III. Parallel Aggregating Joint Mutual Information

This section formalizes the task of feature selection and revisits the JMI maximization methodology, then several preliminary terms for the proposed framework are defined, finally, this section presents a *Parallel Aggregating Joint Mutual Information* (PAJMI) feature selection algorithm.

A. JMI maximization revisit

The JMI objective proposed by Yang and Moody [12] focuses on maximizing the complementary information of a candidate feature w.r.t. the features that have already been selected. Recall features are selected sequentially at each time step by choosing the candidate feature F_i that maximizes (2), then the candidate feature is added into the relevant set $\mathcal{S}$ as showed in Figure 1.

We begin our discussion by defining some terms, assume at time t all the candidates $F_i \in \mathcal{F} \backslash \mathcal{S}$ are evaluated w.r.t. the current selected subset $\mathcal{S}$ and we end up with a JMI score vector of size $|\mathcal{F} \backslash \mathcal{S}|$, by sorting this vector we can obtain the following two sequences:

- $T_t^{(n)}$: The descending sequence sorted from the JMI score vector at time t in terms of the values. Note that a parenthesis in the superscript is a sequence and an index represents an element of the ordered sequence.
- $X_t^{(n)}$: The sequence of candidate features indices sorted according to $T_t^{(n)}$. The same notation applies with parenthesis as mentioned above.

Note that n is the length of each sequence which is denoted with a superscript in parenthesis, and we use X_t^p (T_t^p) for $1 \leq p \leq n$ to index an element within the sequence $X_t^{(n)}$ ($T_t^{(n)}$).

Now consider the following scenario: assume at time t, two features $X_t^p, X_t^q \in X_t^{(n)}$ with corresponding JMI scores T_t^p and T_t^q for $1 \leq p, q \leq n$ and $p < q$. In the next successive r time steps there will be r features selected which are denoted as a set $\mathcal{S}^*$ and the elements are denoted as X^* (Note that $\mathcal{S}^*$ and X^* remain unknown at time t). If $X_t^p, X_t^q \notin \mathcal{S}^*$ then we can end up with the new indices for X_t^p, X_t^q respectively as: $X_{t+r}^{p'}, X_{t+r}^{q'} \in X_{t+r}^{(n-r)}$, where $X_{t+r}^{(n-r)}$ is the sorted candidate

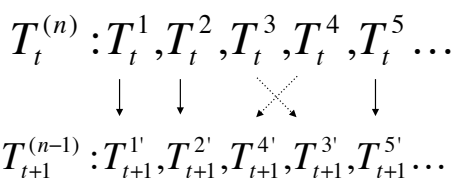


Fig. 2: Two ordered score sequences $T_t^{(n)}$ and $T_{t+1}^{(n-1)}$ at time t and $t+1$, the correspondence of each candidate feature at two time steps are marked with arrows, e.g., $T_{t+1}^{p'}$ is the corresponding feature T_t^p at time $t+1$ in a different position.

feature sequence at time $t+r$. Then we can write the JMI scores for $X_{t+r}^{p'}, X_{t+r}^{q'}$ at time $t+r$ as follows:

$$T_{t+r}^{p'} = T_t^p + \sum_{X^* \in \mathcal{S}^*} I(X^*, X_t^p; Y)$$
$$T_{t+r}^{q'} = T_t^q + \sum_{X^* \in \mathcal{S}^*} I(X^*, X_t^q; Y)$$

The above equations are the scores of the indices X_t^p and X_t^q at time $t+r$. The superscript p', q' are the indices in the ordered sequence $X_{t+r}^{(n-r)}$. Let us consider an example, Figure 2 shows the ordering of two ordered scores at time t and $t+1$. We observe that in this example that the score T_t^1 is the largest score at time $t+1$ followed by the score T_t^2. That is to say from time t to $t+1$ the order of the leading features did not change. However, there is a swap between the 3rd and 4th ordered features from time t to $t+1$. Carefully observe the notation with the p and p' related to the previous equations. Therefore for $T_{t+r}^{p'}$ and $T_{t+r}^{q'}$, whether $p' < q'$ or $p' > q'$ remains unknown at time t due to the involvement of unknown terms: $X^* \in \mathcal{S}^*$. Knowing this definition, we can evaluate the comparison of $T_{t+r}^{p'}, T_{t+r}^{q'} \in T_{t+r}^{(n-r)}$ by:

$$(T_t^p - T_t^q) + \left(\sum_{X^* \in \mathcal{S}^*} I(X^*, X_t^p; Y) - \sum_{X^* \in \mathcal{S}^*} I(X^*, X_t^q; Y) \right) \tag{3}$$

which consists two differences, the first difference satisfies $T_t^p > T_t^q$, and the second remains unknown at time t since $\mathcal{S}^*$ is unknown. However, the second difference can be bounded since we know $|\mathcal{S}^*| = r$ and joint mutual information greater than 0 and less than entropy, i.e.: $0 \leq I(X^*, X_t^p; Y), I(X^*, X_t^q; Y) \leq H(Y)$, which leads to: $0 \leq \sum_{X^* \in \mathcal{S}^*} I(X^*, X_t^p; Y), \sum_{X^* \in \mathcal{S}^*} I(X^*, X_t^q; Y) \leq rH(Y)$. Thus, the second difference falls into the range: $[-rH(Y),\ rH(Y)]$.

In general, the above analysis shows that the informativeness of two candidates at the current time is influencing their likely informativeness at future time by exploiting JMI's sequential greedy maximization. Candidate features' usefulness at time $t+r$ ($r > 0$) depends on how informative they are at current time t to some extent. Thus, informativeness is decided by how much one candidate dominates the other (i.e. the first difference of 3), and how well the new feature can incorporate with the newly selected r features (i.e. the second difference of 3).

Therefore, when r is fixed, the range of the second contributing part (i.e., second difference of (3)) will be fixed. Then as the difference $T_t^p - T_t^q$ increases, $T_{t+r}^{p'} - T_{t+r}^{q'}$ also increases. On the other hand, if the difference $T_t^p - T_t^q$ is given, the range of the second contributing part expands as r increases. In this case, we can't tell how $T_{t+r}^{p'} - T_{t+r}^{q'}$ updates, but the range of $T_{t+r}^{p'} - T_{t+r}^{q'}$ expands as r increases. Fortunately, the first difference is a summation over all selected features and will eventually overwhelm the second contributing part as more features are selected.

Now let us generalize to multiple features. Once JMI greedy maximization has been initialized, there are not many features in $\mathcal{S}$. For some parts of the sequence: $X_t^{(n)}$, we can observe that the order remains the same for some successive future times. For example:

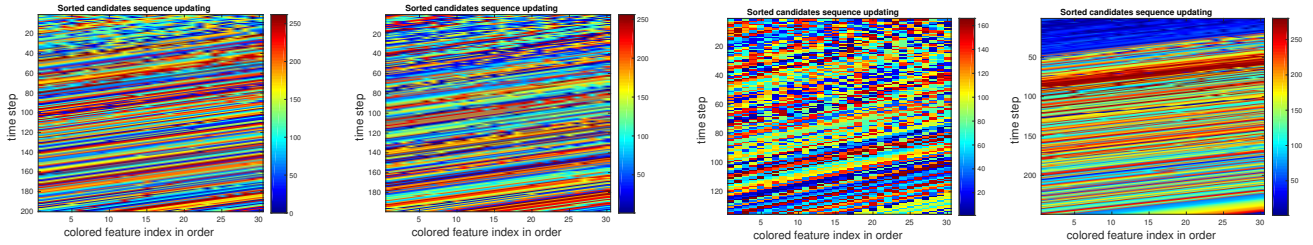
$$X_t^{(n)}: 144, 66, 201, 90, 35, 230$$
$$X_{t+1}^{(n-1)}: 66, 201, 35, 90, 230, 98$$

As we can see the partial sequence $[66,\ 201]$ is identical to next time step with shifting one step to the left. This observation of the partial is shifting is because feature 144 is selected at time t. However, the partial sequence $[66,\ 201,\ 90]$ does not shift down to the next time in the same order because of the feature 35. This shift is caused by a result of scores within $T_t^{(n)}$ are not large enough to compensate the influence brought by the newly selected feature 144. Hence, the shifting behavior did not remain exactly the same from time t to $t+1$.

For a better illustration, Figures 3(a)-3(d) show the rankings of the features at a time t (rows) for benchmark data sets, "arrhythmia", "semeion", "musk" and "isolet". Thus each row is the rearranged candidates sequence $X_t^{(n)}$ at each time. The different colors in the heatmap are used to represents different feature indices in $X_t^{(n)}$ (i.e., has no connection to relevancy).

As we can see, the repeated off-diagonal stripes are observed after a short time (i.e., after time step 40 in "arrhythmia"). This observation indicates that these features are merely shifted repeatedly to the left by one step within the sorted candidate feature sequence, when the scores within $T_t^{(n)}$ are increasing as more and more features added to the selected subset $\mathcal{S}$. Thus, instead of greedily selecting features one at a time, we can simultaneously evaluate multiple candidates that have a high probability to be selected in a future time. This selection process can accelerate the sequential greedy procedure in a semi-parallel fashion. Motivated by the pattern observed in Figures 3(a)-3(d), the goal of our methodology is as follows:

- Instead of selecting only the most informative candidate X_t^1, we propose to evaluate other candidates after X_t^1 in



(a) arrhythmia (b) semeion (c) musk (d) blog

Fig. 3: This figure uses different colors to represent different features for the dataset"arrhythmia", "semeion", "musk" and "blog", the vertical and horizontal axis are successive time steps and sorted feature indices transformed to color, respectively.

the sequence $X_t^{(n)}$ with high probabilities to be selected at future times (i.e., $X_t^2, X_t^3, \cdots, X_t^M$), where M is specifying the amount of candidates to be examined.

- Examine multiple candidates in parallel using a non-exhaustive way, then find the longest sub-sequence $X_t^{(M)}$ of successive candidates starting after X_t^1 that will be selected in exactly the same order as JMI.

B. Definitions for the Parallel Framework

In this section, we define some terms that will be used in the proposed methodology. Assume at time t, after each candidate is evaluated w.r.t. the selected subset $\mathcal{S}$, we can end up with the sorted score sequence $T_t^{(n)}$ and rearranged candidate sequence $X_t^{(n)}$.

Definition 1: For two features $X_t^p, X_t^q \in X_t^{(n)}$ with $p < q$, if the following inequality holds then we call X_t^q is an Intruder for X_t^p.

$$T_t^p + \sum_{w<p} I(X_t^w, X_t^p; Y) < T_t^q + \sum_{w<p} I(X_t^w, X_t^q; Y) \quad (4)$$

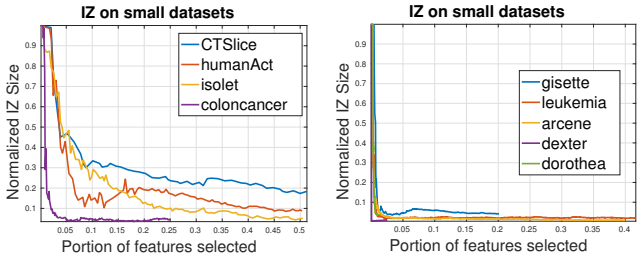
This definition shows that within the sequence $X_t^{(n)}$ if the features $X_t^{w'}s$ for $w < p$ have already been selected, then X_t^q is more likely to be selected than X_t^p when the above inequality holds.

Definition 2: For a feature $X_t^p \in X_t^{(n)}$, among its multiple intruders $X_t^{q'}s$ if existed ($p < q$), we call the feature X_t^q the Maximal Intruder when X_t^q satisfies that:

$$X_t^q = \arg\max_{X_t^q} T_t^q + \sum_{w<p} I(X_t^w, X_t^q; Y) \quad (5)$$

After the Intruders are identified for feature X_t^p within $X_t^{(n)}$ using Definition 1, by finding the Maximal Intruder, we can guarantee X_t^p will be replaced by its Maximal Intruder, if X_t^w for $w < p$ will be selected in advance.

Definition 3: If there are no Intruders existed for a feature X_t^p then we know X_t^p will be selected if X_t^w will be selected in advance for $w < p$. In such situation, we call X_t^p as a Hit, otherwise we call X_t^p is a Miss, in such case we can find a Maximal Intruder for X_t^p.



(a) IF for smaller data sets (b) IF for large data sets

Fig. 4: Interfering Zones (IZ) for different data sets.

Definition 4: In order to find the possible Intruders for a feature X_t^p, we need to compare the difference between two sides of the inequality 4. We do the following transformation:

$$\begin{aligned} &T_t^p + \sum_{w<p} I(X_t^w, X_t^p; Y) - (T_t^q + \sum_{w<p} I(X_t^w, X_t^q; Y)) \\ &= (T_t^p - T_t^q) + \sum_{w<p} I(X_t^w, X_t^p; Y) - \sum_{w<p} I(X_t^w, X_t^q; Y) \\ &\geq (T_t^p - T_t^q) + \sum_{w<p} I(X_t^w, X_t^p; Y) - \sum_{w<p} H(Y) \end{aligned}$$

As we can see, the difference between the two sides of inequality 4 can be bounded and is only depending on T_t^q. Thus, we only need to search an interval of $X_t^{q'}s$ such that only within this interval can X_t^q possibly be an Intruder, we call this interval the Interfering Zone.

To show the rapid convergence of the Interfering Zone, we plotted the trends of Interfering Zone for 8 benchmark datasets in Figure 4(a) and 4(b). As we can see, the size of Interfering Zone converges rapidly to an amount far less than the size of the feature space for all small and large datasets. Thus, knowing the Interfering Zone can reduce the complexity of searching space.

C. Parallel Aggregating Joint Mutual Information (PAJMI) Feature Selection algorithm and Pseudo Code

In this section, we propose a semi-parallel framework that applies the definitions that allow multiple features to be selected simultaneously. The procedure is illustrated as follows: At each time t, the candidates are evaluated w.r.t.

$\mathcal{S}$ to obtain the sequences $X_t^{(n)}, T_t^{(n)}$, which are broadcast to all workers as context variables. Then PAJMI works with candidates within $X_t^2 \rightarrow X_t^{M+1}$ from the sequence $X_t^{(n)}$ at M workers simultaneously. We skip the first element X_t^1 of $X_t^{(n)}$ because the feature is guaranteed to be selected. Then each worker identifies the feature as a Hit or Miss and finds the Maximal Intruders for the Miss's using Definition 3. Finally, our model finds a vector of identification results along with a vector of Maximal Intruders. PAJMI then performs a merging procedure to select multiple features from $X_t^2 \rightarrow X_t^{M+1}$. A detailed algorithm is given in the Pseudo code Algorithm 1 as follows:

Algorithm 1 PAJMI pseudo-code

1: **Input**: Warm Start $\mathcal{P}$; Number to select k; Worker Number M; Feature set $\mathcal{F}$
2: **Initialization**: $\mathcal{S}$ = $\mathcal{P}$, $DistF$, $DistS$, $Flag$, $Intrs$
3: Distribute $\{\mathcal{F} \backslash \mathcal{S}\} \rightarrow DistF$
4: **for** $|\mathcal{S}| < k$ **do**
5: $\quad DistS = ParUpdate(DistF, \mathcal{S})$
6: $\quad (X_t{}^{(n)}, T_t{}^{(n)}) = Sort(DistS, DistF)$
7: $\quad (Flag, Intrs) = ParIdent(X_t{}^{(2)}, \cdots, X_t{}^{(M+1)})$
8: $\quad \mathcal{S} = Update(Flag, Intrs, DistF, \mathcal{S})$
9: $\quad DistF = \{DistF\} \backslash S$
10: **end for**

In the pseudo code, PAJMI begins by taking the following parameters as inputs: (a) k features to select; (b) M parallel workers; (c) the feature space $\mathcal{F}$; and (d) the warm start $\mathcal{P}$ which is the preliminary selected features by JMI maximization, the reason of starting with JMI maximization is to avoid the large searching space before the interfering zone is converged. Then the variables for PAJMI are initialized as follows: the selected subset $\mathcal{S}$ is initialized as the warm start $\mathcal{P}$, and $DistF$ / $DistS$ are two matrices that store the feature candidates and their corresponding JMI scores at each time, respectively. Thus, these matrices are initialized as empty with size M by $\lceil \frac{|\mathcal{F} \backslash \mathcal{S}|}{M} \rceil$. In Definition 3, a candidate is identified as a Hit or $Miss$. Therefore, we initialize the empty vector $Flag$ of size M to store the identification results. Similarly, the empty vector $Intrs$ of size M is used to store the $Maximal\ Intruder$ for each candidate identified as a $Miss$.

After the variables are initialized in step 2, the candidate features are distributed equally to the M workers denoted by the feature matrix $DistF$ in step 3; the for loop is then applied until desired amount of features are selected. Within the for loop, the JMI score matrix $DistS$ is updated for all candidate features in parallel w.r.t. the selected feature subset $\mathcal{S}$ in step 5 (using function $ParUpdate$), which is then merged and sorted to obtain the sequence pair $(X_t^{(n)}, T_t^{(n)})$ in step 6. In step 7, after the corresponding interfering zone of each candidate feature to identify from $(X_t^2, \cdots, X_t^{M+1})$ is calculated, each candidate is identified as a Hit or a $Miss$ inside a parallel worker, by applying definition 3 within its interfering zone (using function $ParIdent$). Still in step 7, for all candidates in $(X_t^2, \cdots, X_t^{M+1})$, PAJMI collects the identification results in the vector: $Flag$ and the Maximal Intruders in the vector: $Intrs$. Since the first candidate X_t^1 is a Hit for sure (i.e. will be selected), it provides a sufficient condition for the first candidate in $(X_t^2, \cdots, X_t^{M+1})$ to be selected when it's identified as a Hit. This procedure proceeds until the first candidate identified as a $Miss$ is found in step 8 (using function $Update$), in such case, its Maximal Intruder will be absorbed to the selected subset $\mathcal{S}$, and it will be discarded along with the rest candidates in $(X_t^2, \cdots, X_t^{M+1})$ even identified as a Hit, finally the selected features are removed from the candidate space in step 9.

IV. Experiments

This section presents the performance of PAJMI against JMI on several datasets that have become standard benchmarks [26], [27]. Our primary objective in this section is to demonstrate that the feature sets returned by both of the approaches to be nearly identical; however, PAJMI should show superior runtime. The experiments were run on on a machine with four Intel Xeon E7-4830 and 256GB of RAM. All algorithms are implemented using MATLAB to allow for a fair comparison.

A. Data Preparation and Preprocessing

Table II shows the datasets used in the benchmarks. The datasets make up a variety of challenging feature selection problems (e.g., moderate N & large p, large N & moderate p, and large N & large p). The number of instances and features range from 48 to 53,550 and 386 to 100,000, respectively. Recall that the greedy forward selection algorithms becomes quite infeasible for very large feature sets. Therefore, we have included three very large datasets (i.e., "dexter", "arcene" and "dorothea") in the experiments.

The third and fourth columns of Table II indicate specific details about the implementation of the JMI and PAJMI. Specifically, "Cached" indicates if the redundancy calculations for JMI and PAJMI are cached for future calculations. "Boot" indicates if the bootstrapping is applied to the dataset, which is performing to average out the statistics. Note that we do not perform bootstrapping on the large datasets simply due to the long runtime of JMI. The remaining columns in Table II describe the parameters of the experiments. "Select" indicates the number of features to select and M specifies the number of parallel workers. As discussed in previous section, M influences the maximum number of features that can selected at each PAJMI batch. Thus, the desire value of M should be the average amount of features that PAJMI can select at each batch, however, which cannot be known in advance. One option for a user would be to run PAJMI for several iterations and observe the average amount of features that can be selected. Finally, the "WStart" indicates the warm start point for PAJMI, which is how many time PAJMI will execute the greedy JMI algorithm before running Algorithm 1. Setting this parameter allows PAJMI to avoid the interfering zones that have not converged for PAJMI.

TABLE II: Datasets

Data Set	$\lvert\mathcal{F}\rvert$	N	Cached	Boot	k	M	P
humanAct	562	7,352	-	✓	281	10	30
isolet	618	6,238	-	✓	309	10	50
coloncancer	2,000	62	-	✓	1,000	10	80
gisette	5,000	1,000	✓	-	1,000	5	70
leukemia	7,129	48	✓	-	3,000	10	30
arcene	10,000	100	✓	-	4,000	10	130
dexter	20,000	300	✓	-	5,000	10	20
dorothea	100,000	800	✓	-	2,000	10	270

TABLE III: Kuncheva consistency between the feature sets returned by JMI and PAJMI.

Data Set	Consistency(JMI, PAJMI)
humanAct	1.0
isolet	1.0
coloncancer	1.0
gisette	0.999
leukemia	0.999
arcene	1.0
dexter	0.999
dorothea	0.999

Finally, we use histogram estimators to discretize any continuous feature in a data set. This needs to be performed since several of the datasets used in the benchmark have continuous features and the discretization must be performed to estimate the information-theoretic quantities. The histograms are produced to have fixed-width bins.

B. Comparison of runtime and consistency

The proposed approach was developed to provide the users with a parallel algorithm that provide the same feature set that JMI returns; however, this algorithm runs in a fraction of the time. Recall that JMI was empirically found to perform better than many other information-theoretic approaches in [8]; however, JMI quickly becomes infeasible to implement in the face of large or "big" features. Therefore, we report the accuracy (i.e., agreement with JMI) of PAJMI and the runtime in this experiment. Kuncheva's stability/consistency index is used to measure the agreement of the feature sets selected by PAJMI with JMI [28]. The Kuncheva index measures the consistency between two sets of length z and is a widely used measurement of a feature selection algorithms stability [23], [29], [30]. Kuncheva's stability is formally given by:

Definition 5 (Kuncheva Consistency [28]): The consistency index for two subsets $\mathcal{A} \subset \mathcal{F}$ and $\mathcal{B} \subset \mathcal{F}$, such that $m = |\mathcal{A} \cap \mathcal{B}|$ and $|\mathcal{A}| = |\mathcal{B}| = n$, where $1 \leq z \leq |\mathcal{F}| = p$, is:

$$\text{Consistency}(\mathcal{A}, \mathcal{B}) = \frac{mp - z^2}{z(p - z)} \in [-1, +1]$$

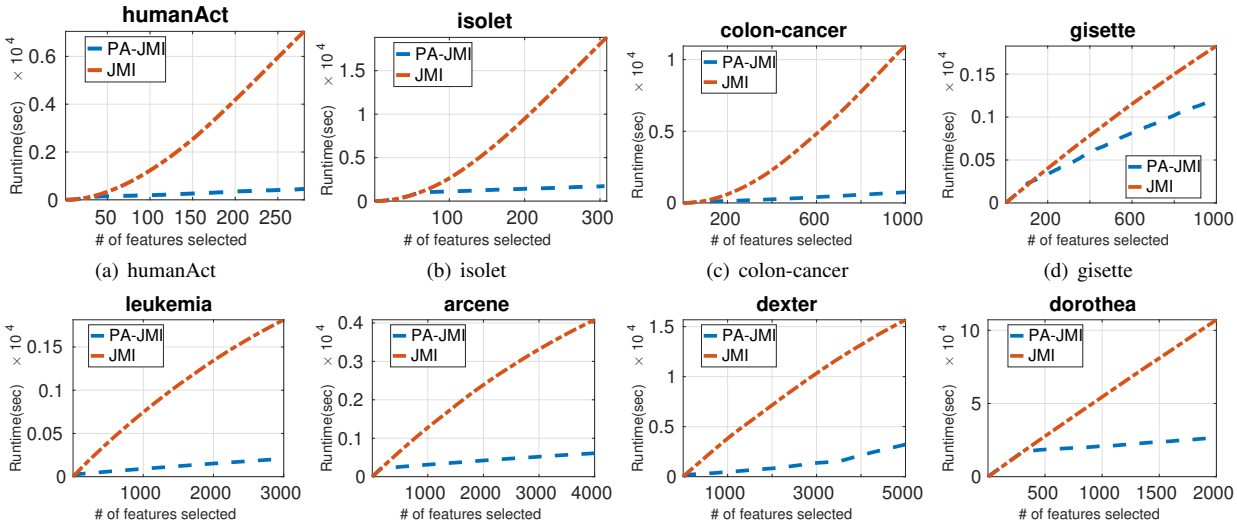
According to Kuncheva's recommendation on consistency, a measurement over 0.5 is considered to be highly consistent.

Table III shows Kuncheva's consistency index for the benchmark datasets. We observe a consistency of 1.0 for three small data sets, for the rest we can always observe a high level of consistency, the slight difference here is caused by the equivalent features existed during the selection. Note that it is possible for PAJMI to have a consistency less than 1.0. An example of a situation would be equivalent features existed in the dataset, hence, causing of a slight disagreement between JMI and PAJMI, which is small enough to be neglected.

Figure 5 shows the runtime of JMI and PAJMI on the datasets in Table II. The first observation is that PAJMI has a significant decrease in the runtime especially as the number of selected features increases. Second, PAJMI gains a significant advantage for datasets that have larger feature sets. For example, selecting 100 features from the "humanAct" dataset (see Figure 5(a)) is approximately the point where PAJMI begins to gain an advantage in the runtime; however, we need to select approximately 18% of the feature set to observe these gains. Now consider a similar scenario when we are selecting 1000 features from the "dexter" dataset (see Figure 5(g)), we observe a significantly larger improvement in the speedup compared to "humanAct"; however, PAJMI is only select 5% of the features. Even more significant improvements are observed on "dorothea", which has $5\times$ as many features as dexter. The "gisette" data set has a less significant improvement in the speedup compared to other seven datasets. The three small data sets are implemented without caching, although their speed up is more significant than in "gisette". The final observation to make is the impact of caching, which can be found in Figures 5(d)-5(h). Caching has a significant improvement to a non-caching implementation of JMI and PAJMI (i.e., the shape of the curve is not quadratic).

V. Conclusions

Many machine learning and data science applications are now associated with very large data sets whose feature size is also a significant concern. Fortunately, not all of the features are relevant to the prediction task in the era of big data. Furthermore, many of the features are redundant, which means even fewer features need to be selected. This problem of dimensionality reduction can be addressed using Information-theoretic feature selection, which was unified by Brown et al. [8]. In [8], the experiments demonstrated that JMI is typically the preferred objective function for Information-theoretic feature selection. Unfortunately, optimizing JMI with a greedy forward search has a quadratic computational complexity, which makes the approach infeasible for large datasets. In this work, we revisited the task of feature selection JMI. We derived a lemma that avoided the blind greedy searching and proposed the "Interfering Zone" to further reduce the searching complexity. To accelerate the greedy forward optimization, we introduced a *Parallel Aggregating Joint Mutual Information* (PAJMI) feature selection framework that can select multiple features at the same time. The experimental results demonstrate that PAJMI selects the same feature set that JMI would return; however, the selection of the features can be achieved in a fraction of the time JMI can select them. This trend was shown on eight benchmark datasets.



(a) humanAct (b) isolet (c) colon-cancer (d) gisette

(e) leukemia (f) arcene (g) dexter (h) dorothea

Fig. 5: Runtime (in seconds) of JMI and PA-JMI on eight data sets shown in Table II.

The future work includes applying the PAJMI to extremely high dimensional benchmark datasets and generalizing a unified framework so that can be used with other greedy forward searches with state-of-art information-theoretic objectives.

REFERENCES

[1] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik, "Gene selection for cancer classification using support vector machines," *Machine Learning*, vol. 46, pp. 389–422, 2002.

[2] Y. Zhia, Y.-S. Ong, and I. W. Tsang, "The Emerging "Big Dimensionality"," *IEEE Comput. Intell. Mag*, vol. 9, no. 3, pp. 14–26, 2014.

[3] G. Ditzler, J. C. Morrison, Y. Lan, and G. Rosen, "Fizzy: Feature selection for metagenomics," *BMC Bioinformatics*, vol. 16, no. 358, 2015.

[4] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of Royal Statistics Society*, vol. 58, no. 1, pp. 267–288, 1996.

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

[6] I. Guyon, S. Gunn, M. Nikravesh, and L. A. Zadeh, *Feature Extraction: Foundations and Applications*. Springer, 2006.

[7] G. Chandrashekar and F. Sahin, "A survey on feature selection methods," *Computers and Electrical Engineering*, 2013.

[8] G. Brown, A. Pocock, M.-J. Zhao, and M. Luján, "Conditional likelihood maximisation: A unifying framework for information theoretic feature selection," *J. Mach. Learn. Res*, vol. 13, pp. 27–66, 2012.

[9] R. Kohavi and G. John, "Wrappers for feature subset selection," *Artificial Intelligence*, pp. 273–324, 1997.

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

[11] G. Qu, S. Hariri, and M. Yousif, "A new dependency and correlation analysis for features," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 9, pp. 1199–1207, 2005.

[12] H. H. Yang and J. Moody, "Feature selection based on joint mutual information," in *Advances in Intelligent Data Analysis*, 1999.

[13] H. Peng, F. Long, and C. Ding, "Feature selection based on mutual information: criteria of max–dependency, max–relevance, and min–redundancy," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 8, pp. 1226–1238, 2005.

[14] H. Yang and J. Moody, "Data visualization and feature selection: New algorithms for non-Gaussian data," in *Advances in Neural Information Processing Systems*, 1999.

[15] C. E. Shannon, "A mathematical theory of communication," *The Bell System Technical Journal*, 1948.

[16] X. He, D. Cai, and P. Niyogi, "Laplacian score for feature selection," in *Advances in neural information processing systems*, 2005.

[17] S. Kullback and R. A. Leibler, "On information and sufficiency," *The Annals of Mathematical Statistics*, 1951.

[18] L. Lefakis and F. Fleuret, "Jointly informative feature selection made tractable by gaussian modeling," *Journal of Machine Learning Research*, vol. 17, no. 182, pp. 1–39, 2016.

[19] D. D. Lewis, "Feature selection and feature extraction for text categorization," in *Proceedings of the Workshop on Speech and Natural Language*, pp. 212–217, 1992.

[20] H. Cheng, Z. Qin, C. Feng, Y. Wang, and F. Li, "Conditional mutual information-based feature conditional mutual information-based feature conditional mutual information-based feature selection analyzing for synergy and redundancy," *ETRI Journal*, 2011.

[21] F. Meyer, D. Paarmann, M. D'Souza, R. Olson, E. M. Glass, M. Kubal, T. Paczian, A. Rodriguez, R. Stevens, A. Wilke, J. Wilkening, and R. A. Edwards, "The metagenomics RAST server – a public resource for the automatic phylogenetic and functional analysis of metagenomes," *BMC Bioinformatics*, vol. 9, no. 386, 2008.

[22] N. S.-M. Verónica Bolón-Canedo and A. Alonso-Betanzos, "Distributed feature selection: An application to microarray data classification," *Applied Soft Computing*, 2015.

[23] G. Ditzler, R. Polikar, and G. Rosen, "A bootstrap based neyman–pearson test for identifying variable importance," *IEEE Trans. Neural Netw. Learn. Syst*, vol. 26, no. 4, pp. 880–886, 2015.

[24] C. Z. H. Q. N. Yingbo Zhou, Utkarsh Porwal and L. Nguyen, "Parallel feature selection inspired by group testing," in *Neural Information Processing Systems*, 2014.

[25] V. Bolon-Canedo, K. Sechidis, N. Sanchez-Marono, and G. Brown, "Exploring the consequences of distributed feature selection in dna microarray data," in *Intl. Joint Conf. on Neural Networks*, 2017.

[26] M. Fernández-Delgado, E. Cernadas, S. Barro, and D. Amorim, "Do we need hundreds of classifiers to solve real world classification problems?," *Journal of Machine Learning Research*, vol. 15, pp. 3133–3181, 2014.

[27] A. Frank and A. Asuncion, "UCI machine learning repository," 2010.

[28] L. I. Kuncheva, "A stability index for feature selection," in *International Conference on Artifical Intelligence and Application*, pp. 390–395, 2007.

[29] S. Nogueira and G. Brown, "Measuring the stability of feature selection with applications to ensemble methods," in *International Workshop on Multiple Classifier Systems*, 2015.

[30] T. Khoshgoftaar, A. Fazelpour, H. Wang, and R. Wald, "A survey of stability analysis of feature subset selection techniques," in *International Conference on Information Reuse and Integration*, pp. 424–431, 2013.