# Unsupervised Feature Selection for Efficient Exploration of High Dimensional Data

Arnab Chakrabarti[1], Abhijeet Das[1], Michael Cochez[2], Christoph Quix[3,4]

[1] RWTH Aachen University, Germany
[2] Vrije Universiteit Amsterdam, Netherlands
[3] Hochschule Niederrhein, University of Applied Sciences, Germany
[4] Fraunhofer Institute for Applied Information Technology FIT, Germany
chakrabarti@dbis.rwth-aachen.de,abhijeet.das@rwth-aachen.de,m.cochez@vu.nl,
christoph.quix@hs-niederrhein.de

**Abstract.** The exponential growth in the ability to generate, capture, and store high dimensional data has driven sophisticated machine learning applications. However, high dimensionality often poses a challenge for analysts to effectively identify and extract relevant features from datasets. Though many feature selection methods have shown good results in supervised learning, the major challenge lies in the area of unsupervised feature selection. For example, in the domain of data visualization, high-dimensional data is difficult to visualize and interpret due to the limitations of the screen, resulting in visual clutter. Visualizations are more interpretable when visualized in a low dimensional feature space. To mitigate these challenges, we present an approach to perform unsupervised feature clustering and selection using our novel graph clustering algorithm based on Clique-Cover Theory. We implemented our approach in an interactive data exploration tool which facilitates the exploration of relationships between features and generates interpretable visualizations.

## 1 Introduction

The ability to collect and generate a wide variety of complex, high-dimensional datasets continues to grow in the era of Big Data. Increasing dimensionality and the growing volume of data pose a challenge to the current data exploration systems to unfold hidden information in high dimensional data. For example, in the field of data visualization human cognition limits the number of data dimensions that can be visually interpreted. The potential amount of overlapping data points projected on to a two-dimensional display hinders the interpretation of meaningful patterns in the data. Though dimensionality reduction has proved to be a promising solution to this problem, there exists the risk of discarding interesting properties of the data. There are two prominent approaches for dimensionality reduction: *Feature Extraction* and *Feature Selection*. Feature extraction strategies such as Principle Component Analysis (PCA), Linear Discriminant Analysis (LDA), or Multidimensional Scaling (MDS), try to mitigate the effect by projecting a high-dimensional feature space into a lower dimensional space which in turn results in information loss due to the transformation

of the locally relevant dimensions. Hence, these methods are often not suitable for data exploration tasks, especially when the user is interested to explore the local structure of the data. On the other hand, feature selection focuses on finding meaningful dimensions, thereby removing irrelevant and redundant features [13], while maintaining the underlying structure. Feature selection methods can be classified as supervised and unsupervised; the latter has gained popularity in recent years.

In this paper, we propose a novel graph clustering approach for unsupervised feature selection using the concept of *"Clique-Cover"* as an underlying foundation. A clique, for an undirected graph, is a subgraph where any two vertices of the subgraph are adjacent to each other. To enumerate such a graph in order to find the largest clique (a clique with most vertices) is termed as the *maximal-clique problem* in graph theory. The maximal-clique model has been studied extensively to detect clusters in large graphs and has found its application in varied domains such as information retrieval [4] or pattern recognition [15].

The contributions of this paper are (i) the integration of Clique-Cover theory in an advanced feature selection pipeline, and (ii) a detailed evaluation of the approach with various experiments using real-world datasets. In our approach, we transform the problem space into a complete graph where the features are nodes and the edge weights denote the degree of correlation between the features. Then, we apply our proposed *maximal-clique* based algorithm for non-overlapping cluster detection. Finally, we select highly relevant features from the detected clusters using *graph-centrality measures*. The algorithm is embedded in a novel *Feature Selection Pipeline* to select features from datasets lacking class labels. This main contribution of the paper is presented in section 3, after we introduced the basics of Clique-Cover theory and discussed related work in section 2.

To verify the efficiency of our approach, we performed experiments to compare our results with that of the existing approaches using real-world datasets. Our experiments presented in section 4 demonstrate that the proposed method performs better than baseline approaches in terms of clustering and classification accuracy. Furthermore, by evaluating our model in terms of computational efficiency and robustness we report the scalability of our model towards increasing dimensionality.

## 2 Background and Related Work

**Clique-Cover** is a graph clustering approach based on the underlying notion of maximal cliques [8]. A *Clique* for an undirected graph $G = (V, E)$ is defined as the set of vertices $C$ such that each of the distinct pair of vertices in $C$ is adjacent (i.e, there exists an edge connecting the pairs). A Clique, which is not a subset of a larger clique, is known as a maximal clique. Thus, given a graph $G$, a subgraph $H$ of a graph $G$ is a maximal clique if $H$ is isomorphic to a complete graph, and there is no vertex $v \in V(G)$ such that $v$ is adjacent to each vertex of $H$. In other words, a subgraph $H$ of a graph $G$ is a maximal clique if $H$ is a

clique, and there is no vertex in $G$ that sends an edge to every vertex of $H$. In this work, we have extended the concept of maximal cliques to the edge-weighted cliques. A maximal clique having the maximum sum of edge-weights highlights the notion of a cluster. The recursive process of determining such cliques leads to the generation of clusters of different sizes. In terms of graph theory, a cluster can be termed as a cover on the respective nodes of the graph such that the subset of nodes is strongly connected within the cover. Thus a *Clique-Cover* is formally defined as follows: *Let $G$ be a graph. A clique of a graph $G$ is a nonempty subset $S$ of $V(G)$ where $S$ is a complete graph. A set $\vartheta$ of cliques in $G$ is a* clique cover *of $G$ if for every $u \in V(G)$ there exists $S \in \vartheta$ such that $u \in S$.*

**Feature Clustering** algorithms localize the search for relevant features and attempt to find clusters that exist in multiple overlapping subspaces. There are two major branches of feature clustering: (i) The bottom-up algorithms find dense regions in low dimensional spaces and combine them to form clusters. (ii) The top-down algorithms start with the full set of dimensions as the initial cluster and evaluate the subspaces of each cluster, iteratively improving the results. *Clustering in Quest(CLIQUE)* [2] is a bottom-up approach that uses a static grid size to determine the clusters within the subspace of the dataset. It combines density and grid-based clustering and uses an a-priori-style technique to identify clusters in subspaces. Tuning the parameters for grid size and the density threshold is difficult in CLIQUE. *PROjected CLUstering(PROCLUS)* [1] is a top-down algorithm, which samples the data, selects a set of $k$ medoids and iteratively improves the clustering. Although we can achieve an enhanced cluster quality, it depends on parameters like the number of clusters and the size of the subspaces, which are difficult to determine in advance.

**Unsupervised Feature Selection** can be broadly classified into three main approaches: *Filter*, *Wrapper*, and *Hybrid*. *Filter methods* select the most relevant features from the data itself without using any clustering algorithm. However, they are unable to model feature dependencies and yield better results mostly with supervised data. Relief [16], Laplacian Score [10], Spectral feature selection [23] are some of the filter methods. *Wrapper methods*, on the other hand, use the results of a specific clustering algorithm to evaluate feature subsets. Although they can model feature dependencies, the main disadvantage of wrapper approaches is that they have a high computational cost and are prone to overfitting. *Hybrid methods* combine filter and wrapper models and aim at achieving a compromise between efficiency and effectiveness (significance of the feature subsets) by using feature ranking metrics and learning algorithms. As highlighted in [18], the limitation of these models is that they require the specification of the hyper-parameters in advance. Moreover, most of the traditional methods are designed to handle only numerical data, whereas the data generated in real-world applications is a combination of numerical and non-numerical features. In the next section we present our proposed graph clustering algorithm based on the Clique-Cover theory. A weighted feature graph is constructed where the nodes represent the feature set and the edges represent the feature correlation measures.

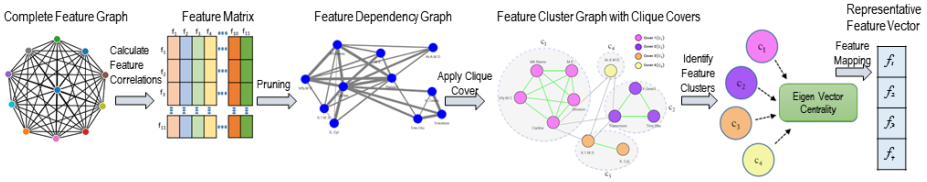


**Fig. 1.** Feature Selection Pipeline

## 3 Unsupervised Feature Selection with Clique Covers

Our approach is unsupervised as it does not require class label information. The main idea is to model the dependency between features by clustering them. Interpretability of features is retained as features are not transformed but only selected. To prove the effectiveness of our approach in the domain of data visualization, we provide an interface to visualize the *Representative Features*. The interface allows the user to explore the intermediate results, visualize feature graphs, and visually inspect the data of the resulting feature sets.

We depict the complete workflow of our feature selection pipeline in Figure 1. As a first step, we create a **Complete Feature Graph** from the dataset. Next, we assign weights to the feature graph using feature correlation measures. These weights are stored in the **Feature Matrix** which acts as the internal data structure for our **Feature Correlation Graph**. In the next step, we apply our graph pruning algorithm to detect and remove weakly connected edges from the complete Feature Correlation Graph and generate the **Feature Dependency Graph**. As a next step, we iteratively apply our *Clique-Cover algorithm* to the Feature Dependency Graph to identify the *clique-covers*. From these clusters, we now apply our algorithm for *Representative Feature Selection*, using *eigen-vector centrality measures*, to construct the **Representative Feature Vector**, which gives us the dimensionality-reduced feature space. In the following subsections, we describe each of the steps in detail.

**Data Model & Preprocessing.** We assume the data to be in tabular format, where the columns represent the *features* and the rows represent the *data points*. After the data is cleaned and pre-processed, we split the features in categorical and numerical features. For data pre-processing, we perform the following steps: 1. Data Cleaning for the removal of empty and duplicate columns, 2. Data Normalization for standardizing both numerical and the categorical data, 3. Data Imputation for dealing with missing values by using the principle of predictive mean matching [21], and 4. Data Segregation for identifying numerical and categorical features. The reason to segregate is that we apply the most suitable correlation measures in the respective groups in order to capture the maximum trends of association. As discussed in the following section, a single correlation measure cannot work well with both groups. A selection of appropriate measures is required for the proper construction of the feature graph.

**Construction of the Feature Correlation Graph.** First, we need to determine the pairwise feature correlations. In the feature graph, the nodes are the features, and the edge-weights are the correlation or association coefficients between the features. As described above, we construct two feature correlation graphs, one for numerical and one for categorical features. To calculate the weights of the edges for these graphs we use the following correlation measures: (i) The Chi-square test of association followed by Cramer's V for categorical feature groups. (ii) Maximal Information Coefficient (MIC) for numerical feature groups. *The Chi-square test* is used to determine the correlation between categorical variables. While it is advantageous because it is symmetric in nature and invariant with respect to the order of the categories, it suffers from certain weaknesses. For example, it fails to specify the strength of the association between the variables and it is sensitive to the sample size. To address this challenge we use a further test known as the *Cramer's V*. This is an essential test that we conduct in order to determine feature correlation as it is immune to the sample size and provides a normalized value, where 0 implies no association and 1 implies a strong association between the attributes. We use *MIC* to capture non-linear trends between variables of numerical feature groups by using the concept of Information Entropy. However, in high-dimensional space this method becomes computationally expensive [19]. To overcome this, we calculate MIC using the normalized Mutual Information. A square matrix is created using the MIC pairs that represents the correlations between the numerical feature groups.

At the end of this step, we get two weighted square Feature Matrices for each feature correlation graphs. Both feature groups were handled independently and they undergo identical processes in the feature selection pipeline. In the rest of the paper, for the purpose of explainability, we describe a common feature selection process (as in Figure 1). This represents the overall workflow for feature selection using our proposed approach.

**Feature Pruning.** The complete feature graph obtained from the previous step may contain weakly connected edges between the nodes. We identify a *weak-edge* as those edges whose weights are below the *Threshold Coefficient*. The Threshold Coefficients are determined using the concepts of K-Nearest Neighbors [3]. The KNN method is a common approach in graph algorithms to determine the proximity of nodes [14]. The reason for using the KNN algorithm is that it does not require any assumptions or training steps to build the model. Moreover, it is easy to implement and robust to noisy data as well. The value of $K$ is set as $K = \sqrt{N}$, where $N$ is the number of features in each of the feature sets.

After determining the threshold coefficients, the K strongest connections for each feature are retained and the others are pruned, resulting in the *Feature Dependency Graph*. We store the Feature Dependency Graph in the form of an *affinity matrix* which is symmetric in nature. The steps of this process are shown in Algorithm 1, which takes the correlation matrix ($corrMat$) and $K$ (described above) as inputs and gives the *affinity matrix* ($affMat$) as the output. The correlation matrices (*MIC and Cramer*) are square weighted adjacency matrices obtained by applying correlation measures on the Feature Correlation Graphs.

**Algorithm 1:** Identifying Threshold Coefficient

```
Procedure : makeAffinity
Input: corrMat (MIC and/or Cramer) and K;
Output: affinitymatrix(affMat);
totalNodes ← length(corrMat);
if K > totalNodes then
    affMat ← corrMat;
end
else
    /* Determine strong connections for every feature node        */
    foreach i-th feature in totalNodes do
        strongConnections ← sort(corrMat[i,], decreasing = TRUE)[1 : K];
        /* Make the affinity matrix symmetric in nature           */
        foreach s-th feature in strongConnections do
            j ← position(corrMat[i,] == s); affMat[i, j] ← corrMat[i, j];
            affMat[j, i] ← corrMat[i, j];
        end
    end
end
return affMat;
```

**Feature Clustering.** To identify relevant clusters in our Feature Dependency Graph, we have used the "Clique Cover Theory" [8]. For our approach of identifying the maximal cliques with respect to the maximum sum of the edge-weights from the undirected edge-weighted Feature Dependency Graphs, we evaluate the sub-graphs satisfying the following properties: (i) Internal homogeneity: Elements belonging to a group have high associations with each other. (ii) Maximality: A maximal clique cannot be further extended by introducing external elements. These properties emphasize the notion of a cluster. Such a cluster is termed as *Clique-Cover* in graph theory which partitions an undirected graph into cliques of various sizes. To explain the use of this approach for constructing our proposed algorithm of finding feature clusters, let us consider the example of a Feature Dependency Graph as shown in Figure 2.

The example has seven nodes, representing the features of the dataset, and the edge-weight corresponds to the correlation coefficient between the feature pairs. The algorithm initially determines the cliques from the graph and further determines the maximal cliques. It then proceeds to incorporate the edge-weights of the maximal cliques. The maximal clique with respect to the maximum sum of edge-weight is identified as a cluster. In the graph from Figure 2 we can see that there exists many cliques such as {3,6}, {1,7},{5,6},{3,5,6}. However only five maximal cliques can be identified namely, {3,4,6}, {1,4}, {2,3,7}, {3,5,6} and {1,7}. We assign the weight of the maximal clique to be equal to the sum of the weights of the edges in that clique. So the corresponding weight of the 5 maximal cliques is 1.82, 0.65, 1.90, 1.42, and 0.60, respectively. In this case, {2,3,7} is the maximal clique with respect to the maximum weight of 1.90. This clique sat-

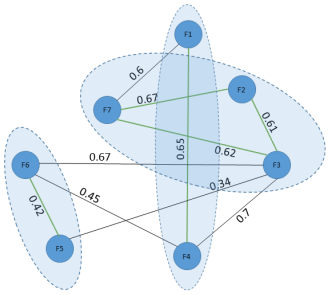


**Fig. 2.** Identification of Cliques Covers on a Feature Dependency Graph

**Algorithm 2:** Unsupervised Feature Clustering w. Clique Cover Theory

**Procedure** : *FeatureClustering*;
**Input :** Weighted Adjacency Matrix obtained from Feature Correlation
**Output :** ClusterNodeIds and ClusterNodeLabels
**Initialize**: ClusterNodeIds ← (); ClusterNodeLabels ← (); remainDim ← totalDim;
**Step 1:** Identify threshold coefficient
**Step 2:** Generate a feature dependency graph
**Step 3:** Determine Cliques Q, Maximal Cliques $Q_i$ and # of Maximal Cliques
**Step 4:** Determine the weight of all the Maximal Cliques
**Step 5:** Determine the Maximal Clique with maximum weight and set it as the first cluster or the Clique Cover
**Step 6:** Update the output with the cluster node Ids and labels
**Step 7:** Remove the clustered nodes and edges from the feature graph
**Step 8:** Update the feature graph with remaining dimensions
**Step 9:** Recursive call to *FeatureClustering* procedure
**Step 10:** If there is one feature node present, then update the output with the last node

isfies the properties of "internal homogeneity" and "maximality", because it has strong interconnections and is maximal. This can be termed as the first cluster or the *Clique Cover*. We now remove the clustered nodes and edges from the existing graph by dynamically truncating the affinity matrix and updating the dimensions. Then, the new feature dependency graph contains the remaining four nodes with features F1, F4, F5, F6 respectively. The cluster identification is applied recursively on the remaining subgraph, and it outputs two more clusters {1,4} and {5,6}. Therefore, the Clique Cover graph clustering algorithm generates three clusters {2,3,7}, {1,4} and {5,6} of sizes 3, 2 and 2 respectively. It can be seen that, the *Clique Cover* always creates non-overlapping/exclusive clusters, which is evident from the fact that none of the features can be present in more than one cluster. Moreover, the approach does not require a prior estimation of the number of clusters. The number of clusters and the size of each cluster is determined dynamically from the intrinsic properties of the graph.

Algorithm 2 presents our feature clustering approach. As an input to the algorithm we give the weighted adjacency matrix obtained from the Feature Correlation step. It initializes two output lists; one for storing the cluster node IDs and the other for storing the cluster node labels. Initially, the remaining dimensions are set to the total dimensions of the feature graph. The algorithm proceeds by identifying the threshold coefficient for each node and generates a feature dependency graph. Next, the algorithm determines the cliques, maximal cliques, and the total number of maximal cliques in the Feature Dependency Graph. It then iterates through all of the maximal cliques and identifies the maximal clique having the maximal weight by summing up the edge-weights in the maximal clique. This maximal clique is the first cluster or Clique Cover. It updates the output list with the corresponding node Ids and labels of the first cluster. The algorithm then removes the clustered nodes and edges and updates the feature graph with the remaining dimensions. It recursively calls the *FeatureClustering* procedure to generate more clusters. This way, the algorithm recursively reduces the size of the remaining dimensions and assigns the feature nodes as part of some clusters. The terminating condition for the recursive process is reached when there is a single node. It terminates by updating the output list with node Ids and labels of the last node.

**Feature Mapping.** In the last step, we map the features from the high dimensional feature space to the *representative features* in the low dimensional space. These features are selected from each of the generated feature clusters. The selection is made using the concepts of graph centrality. Centrality in social networks is an important measure of the influence of a node in the network [7]. In our approach, we have used Eigenvector Centrality to determine the importance of a node in a cluster. It is a globally based centrality measure based on the principle that a node is important if it is linked by other important nodes. Bonacich, in his work [5] has shown that Eigenvector Centrality gives better results when clusters are formed by the determination of maximal cliques. In our approach, the process of determining the Eigenvector Centrality score is carried out for all nodes within each cluster. The maximum score corresponds to the node, which is the most central node in the cluster. The central node is termed as the representative feature.

## 4 Evaluation

We have evaluated our approach over ten datasets and compared with five of the most prominent unsupervised feature selection methods. We demonstrate that our approach discovers more meaningful feature clusters from complex datasets and gives good results in terms of clustering and classification accuracy. From visualization perspective, we show that by using our approach the visualizations render much less clutter and in turn making high dimensional data much easier and intuitive to explore.

**Table 1.** Experimental results for selected Datasets. Dimensions are the total number of features in the dataset and #Features are the final set of selected features after the application of our feature selection algorithm.

| Dataset | Dimensions | Time(in seconds) | #Features |
|---|---|---|---|
| Automobile | 25 | 0.10 | 9 |
| QSAR Biodegradation | 41 | 0.16 | 12 |
| Emotions | 78 | 0.97 | 17 |
| Robot Failure | 91 | 1.08 | 27 |
| Yeast | 116 | 2.42 | 20 |
| Musk | 168 | 11.61 | 32 |
| Arrhythmia | 280 | 22.52 | 44 |
| AirlineTicketPrice | 417 | 43.59 | 37 |
| GAMETES Genome | 1000 | 111.50 | 70 |
| Colon | 2000 | 576.27 | 115 |

**Experimental setting.** We have conducted our experiments in a server running Ubuntu 14.04, with two Intel Xeon X5647@2.93GHz CPUs (8 logical cores/CPU) and 16G RAM.

**Datasets.**[5] For the evaluation, we have considered high-dimensional datasets from various categories. The datasets also have different aspects like binary class, multi-class, missing values, and skewed classes. This enables us to perform a stress test in order to compare with existing approaches. For the quantitative evaluation, supervised datasets are selected because the class labels are needed to evaluate the classification and clustering accuracy, and also for the cost-sensitive analysis. Since our approach is unsupervised, we conducted the following steps: 1. we have removed the class labels from the selected datasets, 2. we run our algorithms for feature selection on the unsupervised datasets, 3. the class labels are then appended to the results obtained from each of the feature selection approaches, and 4. the supervised reduced feature sets obtained are then used for quantitative evaluation. Table 1 shows the list of the selected datasets used for evaluation along with the time taken to construct the reduced feature set using our proposed feature selection pipeline.

### 4.1 Baseline Algorithms

We compare the performance of our proposed Clique-Cover based Unsupervised Feature Selection against the following five baseline algorithms. (1) Laplacian Score for Feature Selection [10], (2) Spectral Feature Selection for Supervised and Unsupervised Learning [23], (3) l2,1-Norm Regularized Discriminative Feature Selection for Unsupervised Learning(UDFS) [22], (4) Unsupervised Feature Selection Using Nonnegative Spectral Analysis(NDFS) [11]., (5) Unsupervised feature selection for multi-cluster data(MCFS) [6]. **Evaluation Metrics**. The reduced feature sets obtained from each approach are quantitatively evaluated using these metrics:

[5] Data Repository: https://figshare.com/s/1807247ef2165735465c

- **Evaluation using Classification Accuracy** - The accuracy of the reduced feature sets are evaluated using classifiers: Naive Bayes, Support Vector Machine (SVM), Random Forests and Logistic Regression. K-fold cross validation is used to evaluate the classifiers.
- **Evaluation using Clustering Accuracy** - The accuracy of the reduced feature sets are evaluated using two clustering algorithms: K-means and Expectation Maximization clustering approaches. The Clustering Accuracy metric is used for assessing the clustering quality. The number of clusters is set to the number of classes present in the respective datasets.
- **Evaluation in terms of the redundancy of the Selected Features** - We have used "Representation Entropy" [13] as a metric to evaluate the redundancy of the selected features. Let the eigenvalues of the $d \times d$ covariance matrix of a feature set of size $d$ be $\lambda_j$, where $j = 1...d$ and $\tilde{\lambda_j} = \frac{\lambda_j}{\sum_{j=1}^{d} \lambda_j}$ where $0 \leq \tilde{\lambda_j} \leq 1$, then we define *Representation Entropy* as: $H_R = \sum_{j=1}^{d} \tilde{\lambda_j} log \tilde{\lambda_j}$ The Representation Entropy($H_R$) measures of the amount of information compression achieved by dimensionality reduction. This is equivalent to the amount of redundancy present in the reduced feature set. The goal of our approach is to have a low value of $H_R$ for the individual clusters but a high $H_R$ for the final reduced feature set, which in turn would indicate that the representative feature set has low information redundancy.
- **Evaluation using ROC Curves for cost-sensitive analysis:** ROC curve is used to check the performance of a classification model. We have used this metric for cost-sensitive analysis. The higher AUC signifies the better performance of the classifier corresponding to relevant features in the dataset. We have considered all the classifiers mentioned above and plotted the ROC curve for each of the reduced feature sets obtained from different approaches.

Because of limited space, we describe the evaluation result only from one experiment. However, the extensive evaluation report using the remaining nine datasets can be found here (https://figshare.com/s/01d10e873bd0896fa30a). Below, we show the performance of our model and the comparisons with the baseline approaches using the 'Colon Tumor' dataset[6] as it has the highest number of features (2000 features).

The classification and the clustering accuracy are depicted in Figure 3. From the results, we can conclude that the classification accuracy of the reduced feature space from our proposed approach has shown relatively better results in all the four selected classifiers in comparison with the baseline methods. Regarding clustering accuracy, although the overall clustering accuracy is low as compared to the classification accuracy, the relative performance of our approach is good. The low accuracy is because the number of clusters in the data are different from what we have assigned. As seen in Figure 3, we have determined the clustering and the classification accuracy using the *Full Feature Set* in order to estimate the relative accuracy of our proposed method.

[6] Colon Tumor Data: http://csse.szu.edu.cn/staff/zhuzx/Datasets.html

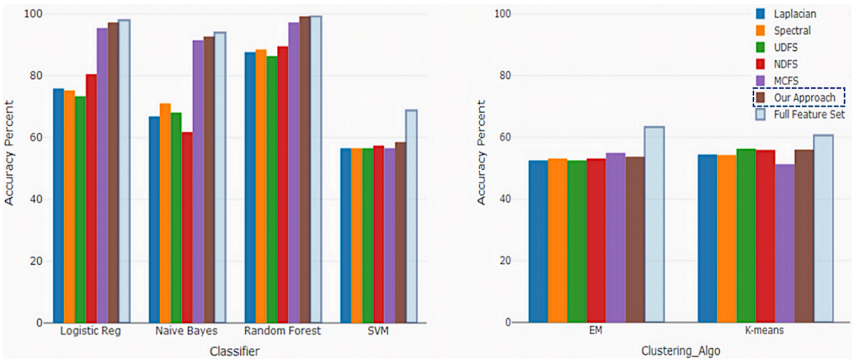


**Fig. 3.** Classification and Clustering Accuracy with the Colon Tumor Dataset

In Figure 4, we plot the ROC curves of the reduced feature sets measured using different classifiers which shows that our approach outperforms the selected methods. Whereas, Figure 5 gives the *Representation Entropy* ($H_R$) obtained from the reduced feature sets of our approach along with the corresponding values obtained from the baseline methods. The resulting Representation Entropy of our proposed approach is higher, which indicates that the features selected by our method have a relatively low information redundancy rate.

### 4.2 Performance Measure

The computational complexity is regarded starting after the ingestion and the feature correlation phase. We have determined the computational complexity from the construction of the Complete Feature Graph until we obtain the Representative Feature Vector. The recursive process of determining the feature clusters mainly depends on three steps: identifying the threshold coefficient using K-NN method, maximal clique determination, and finding the weight of each maximal clique. In our case, the complexity of identifying the threshold coefficient depends on the number of nodes in the complete feature graph($n$) and the value of $k$ ($k$ is the number of nearest neighbors). The complexity is given as $O(kn)$. It has already been proved that the complexity of maximal clique determination is equal to $O(2^{n/3})$, where $n$ is the number of nodes [20]. The complexity of finding the weights of each of the maximal clique depends on the number of edges($e$), and is equivalent to $O(e)$. After the feature clusters are determined, the algorithm identifies representative features from each cluster based on the Eigenvector Centrality measure. The complexity of determining Eigenvector Centrality is $O(qE)$, where $q$ is the number of iterations needed before convergence, and $E$ is the number of edges in each cluster. The combined computational complexity for feature clustering and selection can be written as: $[O(kn) + O(2^{n/3}) + (p \times O(e))] + O(qE)$ where $n$ is the total number of features in the feature graph, $k$ is the number of nearest neighbors, $p$ is the number of intermediate maximal cliques obtained, $e$ is the number of edges in each maximal

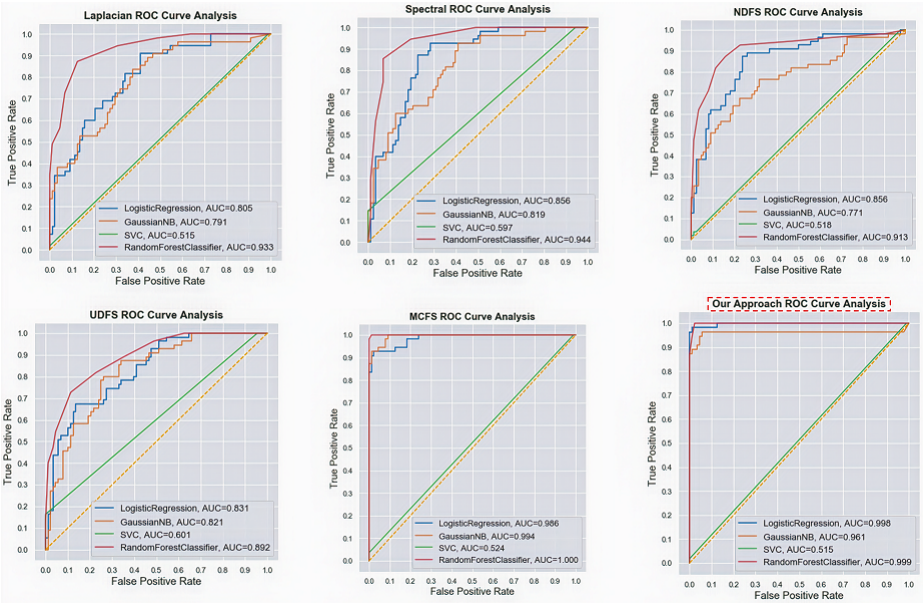


**Fig. 4.** ROC Plots using the Colon Tumor Dataset

| | Laplacian | Spectral | UDFS | NDFS | MCFS | Our Approach |
|---|---|---|---|---|---|---|
| **Rep Entropy** | 0.91315 | 0.90626 | 0.84794 | 0.7526 | 0.94620 | 1.0111 |

**Fig. 5.** Representation Entropy (Colon Tumor Dataset)

clique, $q$ is the number of iterations required to determine Eigenvector Centrality and $E$ is the number of edges in the cluster. The core complexity of this step is represented by the exponential function to determine maximal cliques in the graph. This indicates that the time complexity to determine maximal cliques increases exponentially with the number of features. To investigate the practical consequences, we have calculated the time taken to process the feature graph to determine the Representative Features. As seen in the Table 1, the time taken increases exponentially with respect to the number of features in the dataset. On analyzing the time taken for processing the features, the run time of the algorithm is found to be $t = 1.38^n$.

We would like to mention that the enumeration of maximal cliques has been proven as an NP-hard combinatorial optimization problem. Over the past decade, several algorithms have been designed to address this issue. However, most of these heuristic algorithms fail for massive graphs. Lately, prunning based exact and parameterized algorithms have been proposed which are able to achieve linear runtime scaling for massive graphs [17], [9], [12]. For the purpose of determining the maximal cliques in our proposed feature selection pipeline,

we have used the *Parallel Maximum Clique Solver*[7] which have implemented the algorithms presented in the work of Rossi et al. [17].

## 5 Conclusion

In this work, we have presented a novel graph-based clustering algorithm based on the *Clique Cover Theory*. The number of clusters along with their size is determined dynamically using the intrinsic properties of the data without any prior estimation from the user. The approach was also evaluated on several datasets having a varying number of features and properties. The results indicate that our proposed approach can be used in an effective way for selecting important features in an unsupervised manner, thus proving to be an efficient strategy for dimensionality reduction.

To identify meaningful features from high dimensional data sets and to visualize them efficiently we have tested the implementation of our approach with an interactive data exploration tool[8]. Our visualization tool provides two main functionalities: 1. Explore Data Features and 2. Visualize data in the reduced feature space. With the help of this tool, the features are visualized using feature graphs like cluster feature graphs and representative feature graphs. The correlation between features is explored using correlation heatmaps. The data points in the reduced feature space are visualized using standard methods. The reduced dimensional space allows many visualization techniques to demonstrate various characteristics of the data.

With this tool, we have presented an interface for the efficient exploration of large multidimensional data. One limitation of our approach is that we have segregated the datasets into numerical and categorical feature groups and the feature clusters are determined separately for these individual groups. In the future, we plan to extend our approach so that the resulting clusters are a mix of both the feature groups. We are currently investigating techniques to determine clusters by incorporating correlation measures that determine the relationship between numerical and categorical features. Another interesting direction would be to extend the feature selection to deal with skewed clusters. For example, suppose a dataset has 24 features, and in the clustering phase, 20 features become a part of the first cluster, and the remaining four features are part of the second cluster. Since there are only two clusters, there will be two representative features from each cluster. Thus, the resulting feature set can have very low accuracy. Instead, more than one representative feature for the skewed clusters could be considered.

**Acknowledgment:** This work was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – EXC-2023 Internet of Production – 390621612.

[7] https://github.com/ryanrossi/pmc

[8] VizExploreTool:http://dbis.rwth-aachen.de/cms/staff/chakrabarti/unsupervised-feature-selection/eval/view

## References


1. Aggarwal, C.C., Wolf, J.L., Yu, P.S., Procopiuc, C., Park, J.S.: Fast algorithms for projected clustering. ACM SIGMOD Record **28**(2), 61–72 (1999)
2. Agrawal, R., Gehrke, J., Gunopulos, D., Raghavan, P.: Automatic subspace clustering of high dimensional data for data mining applications. In: Proc. ACM SIGMOD Conf. pp. 94–105 (1998)
3. Altman, N.S.: An introduction to kernel and nearest-neighbor nonparametric regression. The American Statistician **46**(3), 175–185 (1992)
4. Augustson, J.G., Minker, J.: An analysis of some graph theoretical cluster techniques. Journal of the ACM (JACM) **17**(4), 571–588 (1970)
5. Bonacich, P.: Some unique properties of eigenvector centrality. Social networks pp. 555–564 (2007)
6. Cai, D., Zhang, C., He, X.: Unsupervised feature selection for multi-cluster data. In: Proc. ACM SIGKDD. pp. 333–342 (2010)
7. Elgazzar, H., Elmaghraby, A.: Evolutionary centrality and maximal cliques in mobile social networks. Intl. J. of Computer Science & Information Tech. **10** (2018)
8. Erdös, P., Goodman, A.W., Pósa, L.: The representation of a graph by set intersections. Canadian Journal of Mathematics **18**, 106–112 (1966)
9. Gramm, J., Guo, J., Hüffner, F., Niedermeier, R.: Data reduction and exact algorithms for clique cover. J. of Experimental Algorithmics (JEA) **13**, 2 (2009)
10. He, X., Cai, D., Niyogi, P.: Laplacian score for feature selection. In: Advances in neural information processing systems. pp. 507–514 (2006)
11. Li, Z., Yang, Y., Liu, J., Zhou, X., Lu, H.: Unsupervised feature selection using nonnegative spectral analysis. In: Proc. 26th AAAI Conf. (2012)
12. Lu, C., Yu, J.X., Wei, H., Zhang, Y.: Finding the maximum clique in massive graphs. PVLDB **10**(11), 1538–1549 (2017)
13. Mitra, P., Murthy, C., Pal, S.K.: Unsupervised feature selection using feature similarity. IEEE Trans. Pattern Analysis & Machine Intelligence **24**(3), 301–312 (2002)
14. Paredes, R., Chávez, E.: Using the k-nearest neighbor graph for proximity searching in metric spaces. In: Proc. SPIRE. pp. 127–138. Springer (2005)
15. Pavan, M., Pelillo, M.: A new graph-theoretic approach to clustering and segmentation. In: Proc. IEEE Conf. Computer Vision & Pattern Recognition (2003)
16. Robnik-Šikonja, M., Kononenko, I.: Theoretical and empirical analysis of ReliefF and RReliefF. Machine learning **53**(1-2), 23–69 (2003)
17. Rossi, R.A., Gleich, D.F., Gebremedhin, A.H., Patwary, M.M.A.: Fast maximum clique algorithms for large graphs. In: Proc. WWW. pp. 365–366 (2014)
18. Solorio-Fernández, S., Carrasco-Ochoa, J.A., Martínez-Trinidad, J.F.: A review of unsupervised feature selection methods. Artificial Intelligence Rev. pp. 1–42 (2019)
19. Speed, T.: A correlation for the 21st century. Science **334**(6062), 1502–1503 (2011)
20. Tarjan, R.E., Trojanowski, A.E.: Finding a maximum independent set. SIAM Journal on Computing **6**(3), 537–546 (1977)
21. Wright, M.N., Ziegler, A.: ranger: A fast implementation of random forests for high dimensional data in c++ and r. arXiv preprint arXiv:1508.04409 (2015)
22. Yang, Y., Shen, H.T., Ma, Z., Huang, Z., Zhou, X.: L2, 1-norm regularized discriminative feature selection for unsupervised. In: Proc. IJCAI (2011)
23. Zhao, Z., Liu, H.: Spectral feature selection for supervised and unsupervised learning. In: Proc. Intl. Conf. on Machine Learning. pp. 1151–1157. ACM (2007)