Feature Selection: A Data Perspective

Jundong Li JUNDONGL@ASU.EDU
Arizona State University, Tempe, AZ 85281, USA

Kewei Cheng KCHENG18@ASU.EDU
Arizona State University, Tempe, AZ 85281, USA

Suhang Wang SWANG187@ASU.EDU
Arizona State University, Tempe, AZ 85281, USA

Fred Morstatter FMORSTAT@ASU.EDU
Arizona State University, Tempe, AZ 85281, USA

Robert P. Trevino RPTREVIN@ASU.EDU
Arizona State University, Tempe, AZ 85281, USA

Jiliang Tang JLT@YAHOO-INC.COM
Yahoo! Labs, Sunnyvale, CA, 94085, USA

Huan Liu HUANLIU@ASU.EDU
Arizona State University, Tempe, AZ 85281, USA

Editor:

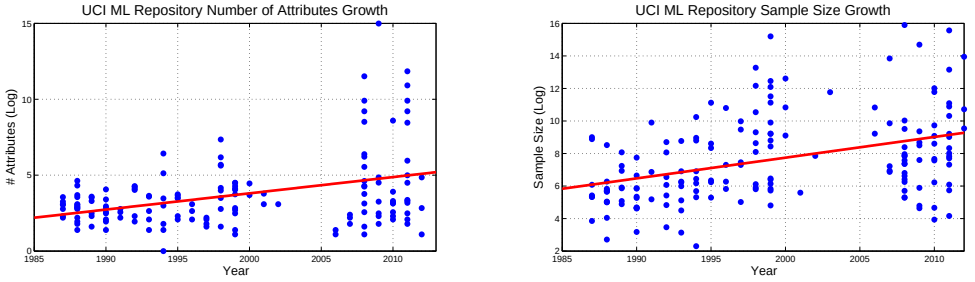
Abstract

Feature selection, as a data preprocessing strategy, has been proven to be effective and efficient in preparing high-dimensional data for data mining and machine learning problems. The objectives of feature selection include: building simpler and more comprehensible models, improving data mining performance, and preparing clean, understandable data. The recent proliferation of big data has presented some substantial challenges and opportunities of feature selection algorithms. In this survey, we provide a comprehensive and structured overview of recent advances in feature selection research. Motivated by current challenges and opportunities in the big data age, we revisit feature selection research from a data perspective, and review representative feature selection algorithms for generic data, structured data, heterogeneous data and streaming data. Methodologically, to emphasize the differences and similarities of most existing feature selection algorithms for generic data, we generally categorize them into four groups: similarity based, information theoretical based, sparse learning based and statistical based methods. Finally, to facilitate and promote the research in this community, we also present a open-source feature selection repository that consists of most of the popular feature selection algorithms (`http://featureselection.asu.edu/`). At the end of this survey, we also have a discussion about some open problems and challenges that need to be paid more attention in future research.

Keywords: Feature Selection

1. Introduction

We are now in the era of big data, where massive amounts of high dimensional data has become ubiquitous in our daily life, such as social media, e-commerce, health care, bioinformatics, transportation, online education, etc. Figure (1) shows an example by plotting the growth trend of UCI machine learning repository (Bache and Lichman, 2013). Rapid growth of data presents challenges for effective and efficient data management. Therefore, it is desirable and of great importance to apply data mining and machine learning techniques to automatically discover knowledge from these data.



(a) attribute number growth (b) sample size growth

Figure 1: Number of samples and number of features growth trend during the past thirty years in UCI machine learning repository.

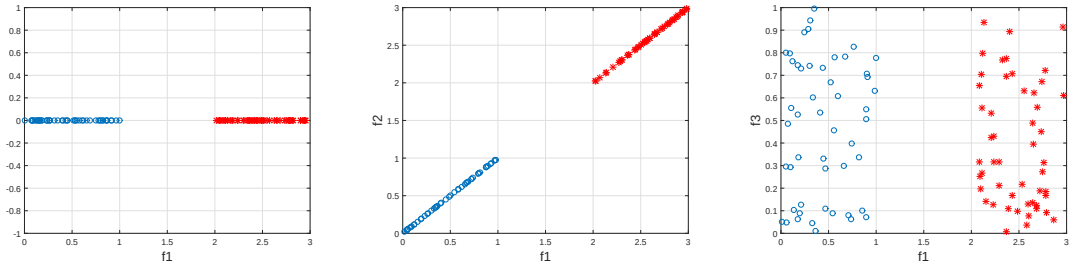
When applying data mining and machine learning algorithms on high dimensional data, a critical issue is known as curse of dimensionality (Hastie et al., 2005). It refers to the phenomenon that data becomes sparser in high dimensional space, adversely affecting algorithms designed for low dimensional space. In addition, with the existence of a large number of features, learning models tend to overfit which may cause performance degradation on unseen data. Moreover, data of high dimension significantly increases the memory storage requirements and computational costs for data analytics.

Dimensionality reduction is one of the most powerful tools to address the previously described issues. It can be categorized mainly into into two main components: feature extraction and feature selection. Feature extraction projects original high dimensional feature space to a new feature space with low dimensionality. The new constructed feature space is usually a linear or nonlinear combination of the original feature space. Examples of feature extraction methods include Principle Component Analysis (PCA) (Jolliffe, 2002), Linear Discriminant Analysis (LDA) (Scholkopft and Mullert, 1999), Canonical Correlation Analysis (CCA) (Hardoon et al., 2004), Singular Value Decomposition (Golub and Van Loan, 2012), ISOMAP (Tenenbaum et al., 2000) and Locally Linear Embedding (LLE) (Roweis and Saul, 2000). Feature selection, on the other hand, directly selects a subset of relevant features for the use model construction. Lasso (Tibshirani, 1996), Information Gain (Cover and Thomas, 2012), Relief (Kira and Rendell, 1992a), MRMR (Peng et al., 2005), Fisher Score (Duda et al., 2012), Laplacian Score (He et al., 2005), and SPEC (Zhao and Liu, 2007) are some of the well known feature selection techniques.

Both feature extraction and feature selection have the advantage of improving learning performance, increasing computational efficiency, decreasing memory storage requirements,

and building better generalization models. However, since feature extraction builds a set of new features, further analysis is problematic as we cannot get the physical meaning of these features in the transformed space. In contrast, by keeping some original features, feature selection maintains physical meanings of original features, and gives models better readability and interpretability. Therefore, feature selection is often preferred in many real-world applications such as text mining and genetic analysis compared to feature extraction.

Real-world data is usually imperfect, containing some irrelevant and redundant features. Removing these features by feature selection reduces storage and computational cost while avoiding significant loss of information or negative degradation of learning performance. For example, in Figure (2(a)), feature f_1 is a relevant feature which is able to discriminate two classes (clusters). However, given feature f_1, feature f_2 in Figure (2(b)) is redundant as f_2 is strongly correlated with f_1. In Figure (2(c)), feature f_3 is an irrelevant feature as it cannot separate two classes (clusters) at all. Therefore, the removal of f_2 and f_3 will not negatively impact the learning performance.



(a) relevant feature f_1 (b) redundant feature f_2 (c) irrelevant feature f_3

Figure 2: A toy example of relevant, irrelevant and redundant features. In Figure (2(a)), feature f_1 is a relevant feature which can discriminate two classes (clusters), i.e., the blue points and the red points. In Figure (2(b)) and Figure (2(c)), feature f_2 and feature f_3 are irrelevant and redundant w.r.t. feature f_1, respectively.

In the following subsections, we first review traditional categorizations of feature selection algorithms from the availability of labels and from the search strategy perspectives in Section 1.1. In Section 1.2, we revisit feature selection from a data perspective motivated by challenges and opportunities from big data. Meanwhile, we discuss the necessity for a comprehensive and structured overview of current advances on feature selection, and our efforts to build a open source machine learning repository to cover state-of-the-art feature selection algorithms. In Section 1.3, we give an outline and organization of the survey.

1.1 Traditional Categorizations of Feature Selection Algorithms

1.1.1 Label Perspective

According to the availability of label information, feature selection algorithms can be broadly classified as supervised, unsupervised and semi-supervised methods.

Supervised Feature Selection
Supervised feature selection is generally designed for classification or regression problems.

It aims to select a subset of features that are able to discriminate samples from different classes. With the existence of class labels, the feature relevance is usually assessed via its correlation with class labels. A general framework of supervised feature selection is illustrated in Figure (3). The training phase of the classification highly depends on feature selection. After splitting the data into training and testing sets, classifiers are trained based on a subset of features selected by supervised feature selection. Note that the feature selection phase can either be independent of the learning algorithm (filter methods), or it may iteratively take advantage of the learning performance of a classifier to assess the quality of selected features so far (wrapper methods). Finally, the trained classifier predicts class labels of samples in the test set on the selected features.

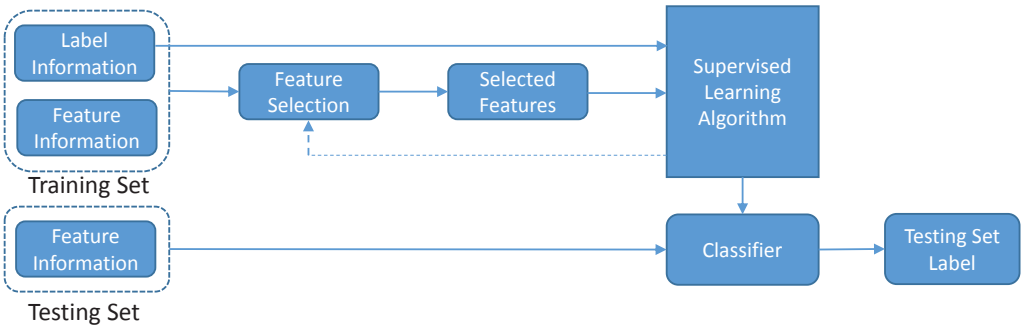


Figure 3: A General Framework of Supervised Feature Selection.

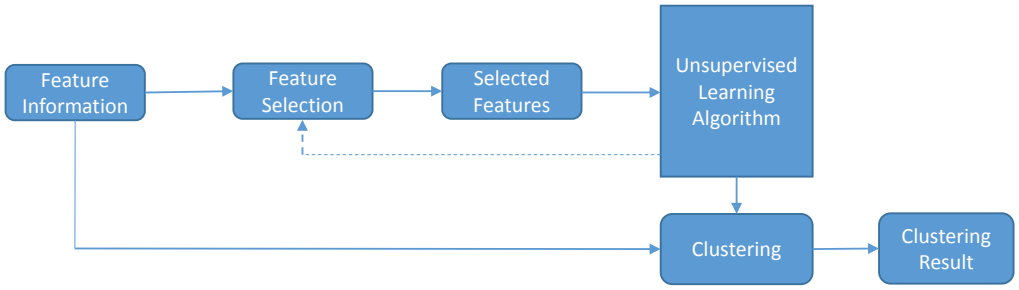


Figure 4: A General Framework of Unsupervised Feature Selection.

Unsupervised Feature Selection

Unsupervised feature selection is generally designed for clustering problems. Since acquiring labeled data is particularly expensive in both time and effort, unsupervised feature selection on unlabeled data has recently gained considerable attention recently. Due to the lack of label information to evaluate the importance of features, unsupervised feature selection methods seek alternative criteria to define the relevance of features such as data similarity and local discriminative information. A general framework of unsupervised feature selection is illustrated in Figure (4). Different from supervised feature selection, unsupervised feature selection usually uses all instances are available in the feature selection phase. The feature selection phase is either be independent of the unsupervised learning algorithms

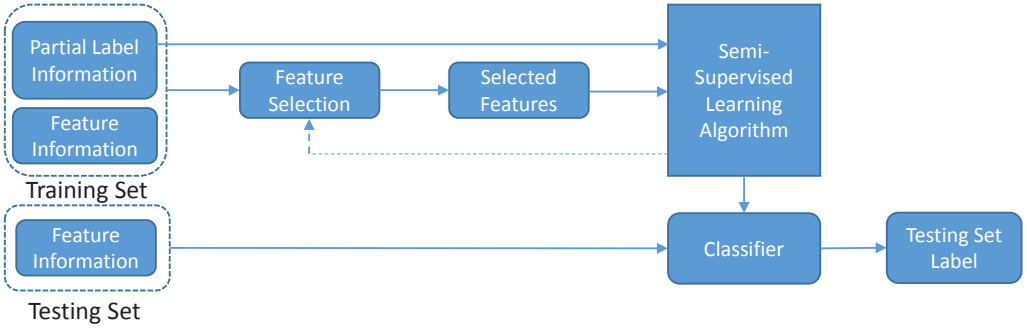


Figure 5: A General Framework of Semi-Supervised Feature Selection.

(filter methods), or it relies on the learning algorithms to iteratively improve the quality of selected features (wrapper methods). After the feature selection phase, it outputs the cluster structure of all data samples on the selected features by using a typical clustering algorithm.

Semi-Supervised Feature Selection
Supervised feature selection works when sufficient label information is available while unsupervised feature selection algorithms do not require any label information. However, in many real-world applications, we usually have a small number of labeled samples and a large number of unlabeled samples. Both supervised and unsupervised feature selection algorithms cannot fully take advantage of all samples in this scenario. For supervised methods, the small number of labeled samples may be insufficient to provide correlation information of features; while unsupervised methods totally ignore class labels which could provide useful information to discriminate different classes. Therefore, it is desirable to develop semi-supervised methods by exploiting both labeled and unlabeled samples. We provide a general framework of semi-supervised feature selection in Figure (5), it is similar to the framework of supervised feature selection except that in semi-supervised methods only partial label information is available.

1.1.2 Search Strategy Perspective

With respect to different selection strategies, feature selection methods can be categorized as wrapper, filter and embedded methods.

Wrapper Methods Wrapper methods rely on the predictive performance of a predefined learning algorithm to evaluate the quality of selected features. Given a specific learning algorithm, a typical wrapper method performs two steps: (1) Searches for a subset of features and (2) evaluate selected features. It repeats (1) and (2) until some stopping criteria are satisfied or the desired learning performance is obtained. The workflow of wrapper methods is illustrated in Figure (6). It can be observed that the feature set search component first generates a subset of features, then the learning algorithm acts as a black box to evaluate the quality of these features based on the learning performance. The whole process works iteratively until the highest learning performance is achieved. The feature subset that gives the highest learning performance is output as the selected features. Unfortunately,

a known issue of wrapper methods is that the search space for d features is 2^d, which makes the exhaustive search impractical when d is large. Therefore, many different search strategies such as sequential search (Guyon and Elisseeff, 2003), hill-climbing search, best-first search (Kohavi and John, 1997), branch-and-bound search (Narendra and Fukunaga, 1977), genetic algorithms (Golberg, 1989) are proposed to yield a local optimum learning performance. However, the search space is still extremely large for high dimensional datasets. As a result, wrapper methods are seldom used in practice.

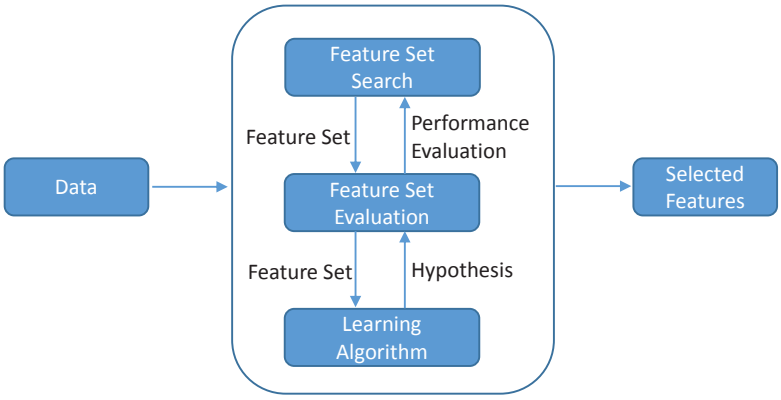


Figure 6: A General Framework of Wrapper Feature Selection Methods.

Filter Methods Filter methods are independent of any learning algorithms. They rely on certain characteristics of data to assess the importance of features. Filter methods are typically more efficient than wrapper methods. However, due to the lack of a specific learning algorithm guiding the feature selection phase, the selected features may not be optimal for the target learning algorithms. A typical filter method consists of two steps. In the first step, feature importance is ranked by a feature score according to some feature evaluation criteria. The feature importance evaluation process can be either univariate or multivariate. In the univariate scheme, each feature is ranked individually regardless of other features, while the multivariate scheme ranks multiple features in a batch way. In the second step of a typical filter method, low ranking features are filtered out and the remaining features are selected. In the past decades, many different evaluation criteria for filter methods have been proposed. Some representative criteria include feature discriminative ability to separate samples (Kira and Rendell, 1992b; Robnik-Šikonja and Kononenko, 2003), feature correlation (Koller and Sahami, 1995; Guyon and Elisseeff, 2003), mutual information (Yu and Liu, 2003; Peng et al., 2005), feature ability to preserve data manifold structure (He et al., 2005; Gu et al., 2011; Zhao and Liu, 2007), and feature ability to reconstruct the original data (Masaeli et al., 2010; Farahat et al., 2011).

Embedded Methods Filter methods select features that are independent of any learning algorithms, therefore they are computational efficient. However, they fail to consider the bias of the learning algorithms, and the selected features may not be optimal for the learning tasks. On the contrast, wrapper methods evaluate the importance of features by the given learning algorithms iteratively and can obtain better predictive accuracy for that specific

learning algorithm. However, due to the exponential search space, it is computational intractable in many applications when the feature dimension is high. Embedded methods provide a trade-off solution between filter and wrapper methods which embed the feature selection with the model learning, thus they inherit the merits of wrapper and filter methods – (1) they include the interactions with the learning algorithm; and (2) they are far more efficient than the wrapper methods since they do not need to evaluate feature sets iteratively. The most widely used embedded methods are the regularization models which targets to fit a learning model by minimizing the fitting errors and forcing the feature coefficients to be small (or exact zero) simultaneously. Afterwards, both the regularization model and selected feature sets are output as results.

1.2 Feature Selection Algorithms from A Data Perspective

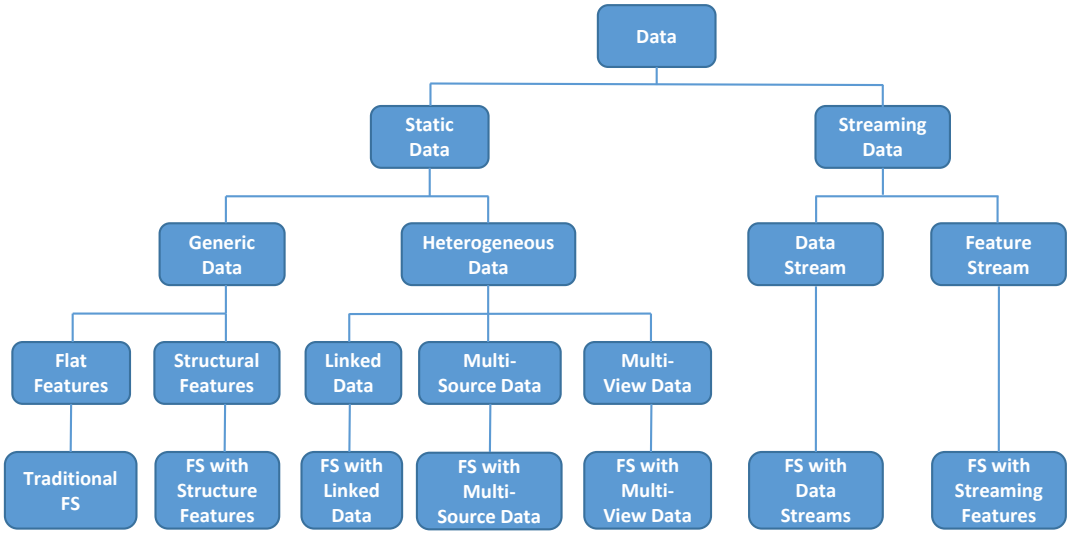


Figure 7: Feature Selection Algorithms from the Data Perspective.

The recent popularity of big data presents some challenges for the traditional feature selection task. Meanwhile, some characteristics of big data like velocity and variety tend to promote the development of novel feature selection algorithms. Here we briefly present and discuss some major concerns when we apply feature selection algorithms.

Streaming Data and Features Streaming data and features have become more prevalent in real world applications. This poses a significant challenge to traditional feature selection algorithms, which assume static datasets with fixed features. For example in Twitter, new data like posts and new features like slang words are continuously being generated. It is impractical to apply traditional batch-mode feature selection algorithms to find relevant features at each round when new data or new feature arrives. In addition, the volume data may sometimes be too large to be loaded into memory directly with a single data scan. This is especially a problem when a second pass is either unavailable or very expensive. Due to aforementioned reasons, it is more appealing to apply feature selection

in a streaming fashion to dynamically maintain a best set of feature from all features and data seen up to that point.

Heterogeneous Data Most existing algorithms of feature selection are designed to handle tasks with single data source and always assume that the data is independent and identically distributed (*i.i.d.*). However, multi-source data is quite prevalent in many domains. For example, in social media, data come from heterogeneous sources such as text, images, tags. In addition, linked data is ubiquitous and presents itself in various forms such as user-post relations and user-user relations. The availability of multiple data sources brings unprecedented opportunities as we can leverage shared intrinsic characteristics and correlations to find more relevant features. However, challenges are also unequivocally presented with additional data sources. For instance, with the existence of link information, the widely adopted *i.i.d.* assumption in most machine learning algorithms does not hold. How to appropriately utilize link information for feature selection is still a challenging problem.

Structures Between Features Sometimes, features can exhibit certain types of structures in many real-world applications. Some well-known structures among features are group structure, tree structure, graph structure, etc. When performing feature selection, if the feature structure is not taken into consideration, the intrinsic dependencies may not be captured and the selected features may not be suitable for the data. Incorporating the prior knowledge of feature structures can possibly help select relevant features to greatly improve the learning performance.

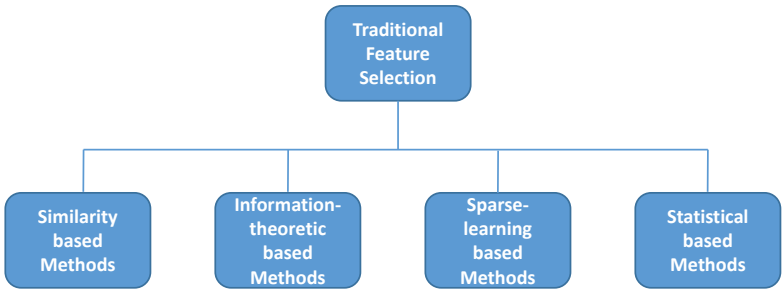


Figure 8: Categorization of Traditional Feature Selection Algorithms According to The Adopted Techniques.

The aforementioned reasons motivate us to investigate feature selection algorithms from a different view to include recent advances and frontiers about feature selection. In this survey, we revisit feature selection algorithms from a data perspective, the categorization is illustrated in Figure (7). It is shown that data consists of static data and streaming data. First, for static data, it can be further grouped into generic data and heterogeneous data. In generic data, features can either be flat or possess some intrinsic structures. Traditional feature selection algorithms are proposed to deal with these flat features in which features are considered to be independent. The past few decades have witnessed hundreds of feature selection algorithms. Based on their technical characteristics, we propose to classify them into four groups, i.e., similarity based methods, information-theoretic based methods, sparse learning based methods and statistical based methods as shown in Figure (8). It should

be noted that this categorization only involves filter methods and embedded methods while the wrapper methods are excluded. The reason for excluding wrapper methods is that they are computationally expensive and, therefore, are seldom used in practice. More details about these four categories will be investigated later. When features express some structures, specific feature selection algorithms for structural features are more desirable. On the other hand, data can be heterogeneous, in many real-world applications we are often encompassed by linked, multi-source or multi-view data, we also show how well-designed feature selection algorithms are designed to deal with these situations. Second, in the streaming settings, data arrive sequentially in a streaming fashion where the size of data instances is unknown, feature selection algorithms that make only one pass over the data is proposed to tackle streaming data. Similarly, in a orthogonal setting, features can also be generated dynamically – new features are sequentially added and the size of features is even unknown in some cases. Streaming feature selection algorithms are designed to determine if accepting the newly added features and if removing existing but outdated features.

Currently, there exist a number of surveys about feature selection algorithms (Guyon and Elisseeff, 2003; Alelyani et al., 2013; Chandrashekar and Sahin, 2014; Tang et al., 2014). These surveys either focus on traditional feature selection algorithms or detailed learning task like classification and clustering. However, none of them provide a comprehensive and structured overview of traditional feature selection algorithms in conjunction with recent advances in feature selection from a data perspective. In this survey, we will introduce representative feature selection algorithms to cover all components mentioned in Figure (7) and Figure (8). We also release a feature selection repository in Python named *scikit-feature* which is built upon the widely used machine learning package *scikit-learn*[1] and two scientific computing packages *Numpy*[2] and *Scipy*[3]. It includes around 40 representative feature selection algorithms. The website of the repository is available at `http://featureselection.asu.edu/`.

1.2.1 Organization of the Survey

We present this survey in five parts and the covered topics are listed as follows:

1. Feature Selection with Generic Data (Section 2)
   (a) Similarity based Feature Selection Methods
   (b) Information Theoretical based Feature Selection Methods
   (c) Sparse Learning based Feature Selection Methods
   (d) Statistical based Feature Selection Methods
2. Feature Selection with Structure Features (Section 3)
   (a) Feature Selection Algorithms with Group Structure Features
   (b) Feature Selection Algorithms with Tree Structure Features
   (c) Feature Selection Algorithms with Graph Structure Features

1. http://scikit-learn.org/stable/
2. http://www.numpy.org/
3. http://www.scipy.org/

3. Feature Selection with Heterogeneous Data (Section 4)
   (a) Feature Selection Algorithms with Linked Data
   (b) Feature Selection Algorithms with Multi-Source Data
   (c) Feature Selection Algorithms with Multi-View Data
4. Feature Selection with Streaming Data (Section 5)
   (a) Feature Selection Algorithms with Data Streams
   (b) Feature Selection Algorithms with Feature Streams
5. Performance Evaluation (Section 6)
6. Open Problems and Challenges (Section 7)
7. Summary of the Survey (Section 8)

2. Feature Selection on Generic Data

Over the past two decades, hundreds of feature selection algorithms have been proposed. In this Section, we broadly group traditional feature selection algorithms for generic data into four categories: similarity based, information theoretical based, sparse learning based and statistical based methods according to the techniques they adopt during the feature selection process. In the following subsections, we will briefly review each category with some representative algorithms.

We summarize some common symbols used throughout this survey in Table 1. We use bold uppercase characters for matrices (e.g. $\mathbf{A}$), bold lowercase characters for vectors (e.g. $\mathbf{a}$), calligraphic fonts for sets (e.g. $\mathcal{F}$). We follow the matrix settings in Matlab to represent i-th row of matrix $\mathbf{A}$ as $\mathbf{A}(i,:)$, j-th column of $\mathbf{A}$ as $\mathbf{A}(:,j)$, (i,j)-th entry of $\mathbf{A}$ as $\mathbf{A}(i,j)$, transpose of $\mathbf{A}$ as $\mathbf{A}'$, and the trace of $\mathbf{A}$ as $tr(\mathbf{A})$. For any matrix $\mathbf{A} \in \mathbb{R}^{n \times d}$, its Frobenius norm is defined as $||\mathbf{A}||_F = \sqrt{\sum_{i=1}^{n} \sum_{j=1}^{d} \mathbf{A}(i,j)^2}$, and its $\ell_{2,1}$-norm is $||\mathbf{A}||_{2,1} = \sum_{i=1}^{n} \sqrt{\sum_{j=1}^{d} \mathbf{A}(i,j)^2}$. For any vector $\mathbf{a} = [a_1, a_2, ..., a_n]'$, its ℓ_2-norm is defined as $||\mathbf{a}||_2 = \sqrt{\sum_{i=1}^{n} a_i^2}$, and its ℓ_1-norm is $||\mathbf{a}||_1 = \sum_{i=1}^{n} |a_i|$. $\mathbf{I}$ is an identity matrix and $\mathbf{1}$ is a vector whose elements are all 1.

2.1 Similarity based Methods

Different feature selection algorithms exploit various types of criteria to define the relevance of features such as distance, separability, information, correlation, dependency, and reconstruction error. Among them, there is a family of methods assessing the importance of features by its ability to preserve data similarity. We call these kinds of methods to be similarity based feature selection methods. For supervised feature selection, data similarity can be derived from label information; while for unsupervised feature selection methods, most methods take advantage of different distance metric measures to obtain data similarity.

Notations	Definitions or Descriptions
n	number of instances in the data
d	number of features in the data
k	number of selected features
c	number of classes (if exist)
$\mathcal{F}$	original feature set which contains d features
$\mathcal{S}$	selected feature set which contains k selected features
$\{i_1, i_2, ..., i_k\}$	index of k selected features in $\mathcal{S}$
$f_1, f_2, ..., f_d$	d features
$f_{i_1}, f_{i_2}, ..., f_{i_k}$	k selected features
$x_1, x_2, ..., x_n$	n data instances
$\mathbf{f}_1, \mathbf{f}_2, ..., \mathbf{f}_d$	d feature vectors corresponding to $f_1, f_2, ..., f_d$
$\mathbf{f}_{i_1}, \mathbf{f}_{i_2}, ..., \mathbf{f}_{i_k}$	k feature vectors corresponding to $f_{i_1}, f_{i_2}, ..., f_{i_k}$
$\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n$	n data vectors corresponding to $x_1, x_2, ..., x_n$
$y_1, y_2, ..., y_n$	class labels of all n instances (if exist)
$\mathbf{X} \in \mathbb{R}^{n \times d}$	data matrix with n instances and d features
$\mathbf{X}_{\mathcal{F}} \in \mathbb{R}^{n \times k}$	data matrix on the selected k features
$\mathbf{y} \in \mathbb{R}^n$	class label vector for all n instances (if exist)

Table 1: Symbols.

Given a dataset $\mathbf{X} \in \mathbb{R}^{n \times d}$ with n instances and d features, the pairwise similarity among instances can be encoded in an affinity matrix $\mathbf{S} \in \mathbb{R}^{n \times n}$. The affinity matrix $\mathbf{S}$ is symmetric and its (i, j)-th entry indicates the similarity between the i-th instance $\mathbf{x}_i$ and the j-th instance $\mathbf{x}_j$, the larger the value of $\mathbf{S}_{i,j}$ is, the more similarity $\mathbf{x}_i$ and $\mathbf{x}_j$ share. Suppose we want to select k most relevant features from $\mathcal{F}$, then the utility of these k features is maximized as follows:

$$\max_{\mathcal{F}} \sum_{f \in \mathcal{F}} SC(f) = \max_{\mathcal{F}} \sum_{\mathbf{f} \in \mathcal{F}} \hat{\mathbf{f}}' \hat{\mathbf{S}} \hat{\mathbf{f}}, \tag{1}$$

where SC is a function that measures the utility of feature $\mathbf{f}$, $\hat{\mathbf{f}}'$ and $\hat{\mathbf{S}}$ are the normalized feature and refined affinity matrix obtained from $\mathbf{f}$ and $\mathbf{S}$, respectively. The maximization problem in Eq. (1) shows that we would select a subset of features from $\mathcal{F}$ such that they can well preserve the data similarity structures defined in $\hat{\mathbf{S}}$. This problem is usually solved by greedily selecting the top k features that maximize their individual utility $\hat{\mathbf{f}}' \mathbf{S} \hat{\mathbf{f}}$. Methods in this category vary in the way the similarity matrix $\mathbf{S}$ is designed. We subsequently discuss about the original formulations of some representative algorithms in this group and then introduce how they can be reformulated under the unified framework.

2.1.1 Laplacian Score (He et al., 2005) (Unsupervised)

Laplacian Score is an unsupervised feature selection algorithm which selects features that can best preserve the data manifold structure. It consists of three phases. First, it construct a nearest neighbor graph $\mathcal{G}$ with n nodes where the i-th node corresponds to x_i. If x_i is among the p nearest neighbors of x_j or x_j is among the p nearest neighbors of x_i, nodes i

and j are connected in $\mathcal{G}$ (p is a predefined number). Second, if nodes i and j are connected, the entry in the affinity matrix $\mathbf{S}_{ij}$ is $\mathbf{S}(i,j) = e^{-\frac{||\mathbf{x}_i - \mathbf{x}_j||^2}{t}}$, where t is a constant, otherwise $\mathbf{S}(i,j) = 0$. The diagonal matrix $\mathbf{D}$ is defined as $\mathbf{D}(i,i) = \sum_{j=1}^{n} \mathbf{S}(i,j)$ and the laplacian matrix $\mathbf{L}$ is $\mathbf{L} = \mathbf{D} - \mathbf{S}$ (Chung, 1997). Lastly, the Laplacian Score of each feature f_i is computed as:

$$laplacian\_score(f_i) = \frac{\tilde{\mathbf{f}}_i' \mathbf{L} \tilde{\mathbf{f}}_i}{\tilde{\mathbf{f}}_i' \mathbf{D} \tilde{\mathbf{f}}_i}, \text{ where } \tilde{\mathbf{f}}_i = \mathbf{f}_i - \frac{\mathbf{f}_i' \mathbf{D} \mathbf{1}}{\mathbf{1}' \mathbf{D} \mathbf{1}} \mathbf{1}. \tag{2}$$

Since Laplacian Score evaluates the importance of each feature individually, the task of selecting the k features can be solved by greedily picking the top k features with the smallest Laplacian Scores.

The Laplacian Score of each feature can be reformulated as follows:

$$\begin{aligned} laplacian\_score(f_i) &= \frac{\tilde{\mathbf{f}}_i' \mathbf{L} \tilde{\mathbf{f}}_i}{\tilde{\mathbf{f}}_i' \mathbf{D} \tilde{\mathbf{f}}_i} = \frac{\tilde{\mathbf{f}}_i' (\mathbf{D} - \mathbf{S}) \tilde{\mathbf{f}}_i}{\tilde{\mathbf{f}}_i' \mathbf{D} \tilde{\mathbf{f}}_i} = 1 - \frac{\tilde{\mathbf{f}}_i' \mathbf{S} \tilde{\mathbf{f}}_i}{\tilde{\mathbf{f}}_i' \mathbf{D} \tilde{\mathbf{f}}_i} \\ &= 1 - \left(\frac{\tilde{\mathbf{f}}_i}{||\mathbf{D}^{\frac{1}{2}} \tilde{\mathbf{f}}_i||} \right)' \mathbf{S} \left(\frac{\tilde{\mathbf{f}}_i}{||\mathbf{D}^{\frac{1}{2}} \tilde{\mathbf{f}}_i||} \right). \end{aligned} \tag{3}$$

Since $\tilde{\mathbf{f}}_i' \mathbf{D} \tilde{\mathbf{f}}_i$ is the weighted data variance of feature f_i (denoted as σ_i^2), $||\mathbf{D}^{\frac{1}{2}} \tilde{\mathbf{f}}_i||$ is the standard data variance (denoted as σ_i), and the term $\tilde{\mathbf{f}}_i / ||\mathbf{D}^{\frac{1}{2}} \tilde{\mathbf{f}}_i||$ is interpreted as a normalized feature vector $\hat{\mathbf{f}}_i = (\mathbf{f}_i - \mu_i \mathbf{1})/\sigma_i$. Therefore, Laplacian Score feature selection can be reformulated by maximizing the following term:

$$\max_{\mathcal{F}} \sum_{\mathbf{f} \in \mathcal{F}} \hat{\mathbf{f}}' \hat{\mathbf{S}} \hat{\mathbf{f}}, \tag{4}$$

and it is a special case of the unified framework of similarity based feature selection.

2.1.2 SPEC (Zhao and Liu, 2007) (Unsupervised and Supervised)

SPEC is an extension of Laplacian Score that work for both supervised and unsupervised scenarios. For example, in the unsupervised scenario, without label information, the data similarity is measured by the RBF kernel function:

$$\mathbf{S}(i,j) = e^{-\frac{||\mathbf{x}_i - \mathbf{x}_j||}{2\sigma^2}}, \tag{5}$$

in the supervised scenario, using label information, data similarity can be defined by:

$$\mathbf{S}(i,j) = \begin{cases} \frac{1}{n_l} & \text{if } y_i = y_j = l \\ 0 & \text{otherwise,} \end{cases} \tag{6}$$

where n_l is the number of data samples in the class l. Afterwards the construction of affinity matrix $\mathbf{S}$, the diagonal matrix $\mathbf{D}$ is defined as $\mathbf{D}(i,i) = \sum_{j=1}^{n} \mathbf{S}(i,j)$ and the normalized laplacian matrix $\mathbf{L}_{norm}$ is $\mathbf{L}_{norm} = \mathbf{D}^{-\frac{1}{2}} (\mathbf{D} - \mathbf{S}) \mathbf{D}^{-\frac{1}{2}}$ (Chung, 1997). The basic idea of SPEC is similar to Laplacian Score, a feature that is consistent with the data manifold

structure assigns similar values to instances that are near each other. The feature relevance are measured by three different criteria:

$$
\begin{aligned}
SPEC\_score1(f_i) &= \hat{\mathbf{f}}_i{}'\gamma(\mathbf{L}_{norm})\hat{\mathbf{f}}_i = \sum_{j=1}^{n}\alpha_j^2\gamma(\lambda_j) \\
SPEC\_score2(f_i) &= \frac{\hat{\mathbf{f}}_i{}'\gamma(\mathbf{L}_{norm})\hat{\mathbf{f}}_i}{1-(\hat{\mathbf{f}}_i{}'\xi_1)^2} = \frac{\sum_{j=2}^{n}\alpha_j^2\gamma(\lambda_j)}{\sum_{j=2}^{n}\alpha_j^2} \\
SPEC\_score3(f_i) &= \sum_{j=1}^{m}(\gamma(2)-\gamma(\lambda_j))\alpha_j^2.
\end{aligned}
\tag{7}
$$

In the above three equations, $\hat{\mathbf{f}}_i = \mathbf{D}^{\frac{1}{2}}\mathbf{f}_i/||\mathbf{D}^{\frac{1}{2}}\mathbf{f}_i||$; (λ_j, ξ_j) is the j-th eigenpair of the normalized laplacian matrix $\mathbf{L}_{norm}$; $\alpha_j = \cos\theta_j$, θ_j is the angle between ξ_j and $\mathbf{f}_i$; $\gamma(.)$ is an increasing function to penalize high frequency components of the eigensystem to reduce noise. If the data is noise free, the function $\gamma(.)$ can be removed and $\gamma(x) = x$. When the second evaluation criteria $SPEC\_score2(f_i)$ is used, SPEC is equivalent to Laplacian Score. For $SPEC\_score3(f_i)$, it uses the top m eigenpairs to evaluate the importance of feature f_i.

Next we show how SPEC can be reduced to the generalized similarity based feature selection framework. Selecting top k features in SPEC with three different criteria in Eq. (7) can be reformulated as:

$$
\begin{aligned}
&\max_{\mathcal{F}}\sum_{\mathbf{f}\in\mathcal{F}}\hat{\mathbf{f}}_i'\hat{\mathbf{S}}\hat{\mathbf{f}}_i \\
&\text{in } SPEC\_score1: \hat{\mathbf{f}}_i = \mathbf{f}_i/||\mathbf{D}^{\frac{1}{2}}\mathbf{f}_i||, \hat{\mathbf{S}} = \mathbf{D}^{\frac{1}{2}}\mathbf{U}(\mathbf{I}-\gamma(\mathbf{\Sigma}))\mathbf{U}'\mathbf{D}^{\frac{1}{2}} \\
&\text{in } SPEC\_score2: \hat{\mathbf{f}}_i = (\mathbf{f}_i-\mu\mathbf{1})/||\mathbf{D}^{\frac{1}{2}}\mathbf{f}_i||, \hat{\mathbf{S}} = \mathbf{D}^{\frac{1}{2}}\mathbf{U}(\mathbf{I}-\gamma(\mathbf{\Sigma}))\mathbf{U}'\mathbf{D}^{\frac{1}{2}} \\
&\text{in } SPEC\_score3: \hat{\mathbf{f}}_i = \mathbf{f}_i/||\mathbf{D}^{\frac{1}{2}}\mathbf{f}_i||, \hat{\mathbf{S}} = \mathbf{D}^{\frac{1}{2}}\mathbf{U}_m(\gamma(2\mathbf{I})-\gamma(\mathbf{\Sigma}_m))\mathbf{U}_m'\mathbf{D}^{\frac{1}{2}},
\end{aligned}
\tag{8}
$$

where $\mathbf{U}$ and $\mathbf{\Sigma}$ contain the singular vectors and singular values of the normalized laplacian matrix $\mathbf{L}_{norm}$, $\mathbf{L}_{norm} = \mathbf{U}\mathbf{\Sigma}\mathbf{U}'$.

2.1.3 Fisher Score (Duda et al., 2012) (Supervised)

Fisher Score is a supervised feature selection algorithm. Suppose the class labels of n samples $\mathbf{y} = \{y_1, y_2, ..., y_n\}$ come from c classes, Fisher Score selects the features such that the feature values of the samples within the same class are small while the feature values of the samples from different classes are large. The Fisher score of each feature f_i is evaluated as follows:

$$
fisher\_score(f_i) = \frac{\sum_{j=1}^{c} n_j(\mu_{i,j}-\mu_i)^2}{\sum_{j=1}^{c} n_j\sigma(i,j)^2},
\tag{9}
$$

where n_j, μ_i, $\mu_{i,j}$ and $\sigma_{i,j}^2$ indicate the number of samples in class j, mean value of feature f_i, mean value of feature f_i for samples in class j, variance value of feature f_i for samples in class j, respectively. Similar to Laplacian Score, the top k features can be obtained by greedily selecting the features with the largest Fisher Scores.

According to (He et al., 2005), Fisher Score can be considered as a special case of Laplacian Score as long as the affinity matrix is as follows:

$$\mathbf{S}(i,j) = \begin{cases} \frac{1}{n_l} & \text{if } y_i = y_j = l \\ 0 & \text{otherwise,} \end{cases} \tag{10}$$

where n_l is the number of data samples in the class l. With this specific affinity matrix, we can get the following relationship between Fisher Score and Laplacian Score as follows:

$$fisher\_score(f_i) = 1 - \frac{1}{laplacian\_score(f_i)}. \tag{11}$$

Therefore, the computation of Fisher Score can be reduced to the unified framework of similarity based feature selection.

Fisher Score measures the relevance of each feature individually as Laplacian Score and SPEC. This leads to a suboptimal subset of features that is incapable of removing redundant features. To tackle this issue, a Generalized Fisher Score method Gu et al. (2011) is proposed to jointly select features. It aims to find a subset of features by maximizing the lower bound of Fisher Score, which results in the following objective function:

$$\begin{aligned} \min_{\mathbf{W},\mathbf{p}} \quad & \frac{1}{2}||\mathbf{X}diag(\mathbf{p})\mathbf{W} - \mathbf{H}||_F^2 + \frac{\alpha}{2}||\mathbf{W}||_F^2 \\ s.t. \quad & \mathbf{p} \in \{0,1\}^d, \mathbf{p}'\mathbf{1} = k. \end{aligned} \tag{12}$$

$\mathbf{p}$ is a feature indicator vector to indicate whether the feature is selected or not, α is a regularization parameter, $\mathbf{H} \in \mathbb{R}^{n \times c}$ is a label matrix, its (i,j)-th entry is given by:

$$\mathbf{H}(i,j) = \begin{cases} \sqrt{\frac{n}{n_j}} - \sqrt{\frac{n_j}{n}} & \text{if } y_i = j \\ -\sqrt{\frac{n_j}{n}} & \text{otherwise,} \end{cases} \tag{13}$$

where n_j is the number of data instances in class j.

2.1.4 Trace Ratio Criterion (Nie et al., 2008) (Supervised)

Recently, the trace ratio criterion has been proposed to directly select the global optimal feature subset based on the corresponding score, which is computed in a trace ratio norm. It builds two affinity matrices $\mathbf{S}_w$ and $\mathbf{S}_b$ to characterize within-class (local affinity) and between-class (global affinity) data similarity. Their corresponding diagonal matrices and laplacian matrices are defined as $\mathbf{D}_w(i,i) = \sum_{j=1}^n \mathbf{S}_w(i,j)$, $\mathbf{D}_b(i,i) = \sum_{j=1}^n \mathbf{S}_b(i,j)$, $\mathbf{L}_w = \mathbf{D}_w - \mathbf{S}_w$, $\mathbf{L}_b = \mathbf{D}_b - \mathbf{S}_b$, respectively. Let $\mathbf{W} = [\mathbf{w}_{i_1}, \mathbf{w}_{i_2}, ..., \mathbf{w}_{i_k}] \in \mathbb{R}^{d \times k}$ be the selection indicator matrix such that only the i_j-th entry in $\mathbf{w}_{i_j}$ is 1 and all the other entries are 0. With these, the trace ratio criterion of all k features in $\mathcal{F}$ is:

$$trace\_ratio(\mathcal{F}) = \frac{tr(\mathbf{W}'\mathbf{X}'\mathbf{L}_b\mathbf{X}\mathbf{W})}{tr(\mathbf{W}'\mathbf{X}'\mathbf{L}_w\mathbf{X}\mathbf{W})}. \tag{14}$$

The basic idea is to maximize the data similarity for the instances from the same class (or close to each other) while minimizing the data similarity for the instances from different

classes (or far away from each other). The larger the score, the more important the feature set is.

The trace ratio criterion score in Eq. (14) provides a general framework for feature selection. Different between-class and within-class similarity matricse $\mathbf{S}_b$ and $\mathbf{S}_w$ lead to different feature selection algorithms such as batch-mode Lalpacian Score and batch-mode Fisher Score; all of them can be considered as special cases of the general similarity based feature selection framework. For example, in batch-mode Fisher Score, the within-class data similarity and the between-class data similarity are defined as follows:

$$\mathbf{S}_w(i,j) = \begin{cases} 1/n_l & \text{if } y_i = y_j = l \\ 0 & \text{otherwise,} \end{cases} \tag{15}$$

$$\mathbf{S}_b(i,j) = \begin{cases} 1/n - 1/n_l & \text{if } y_i = y_j = l \\ 1/n & \text{otherwise,} \end{cases} \tag{16}$$

where n_l is the number of instances in class l. The trace ratio criterion for the feature subset $\mathcal{F}$ can be calculated as:

$$trace\_ratio\_fisher(\mathcal{F}) = \frac{tr(\mathbf{W}'\mathbf{X}'\mathbf{L}_b\mathbf{X}\mathbf{W})}{tr(\mathbf{W}'\mathbf{X}'\mathbf{L}_w\mathbf{X}\mathbf{W})} = \frac{\sum_{s=1}^{k}\mathbf{f}'_{i_s}\mathbf{S}_w\mathbf{f}_{i_s}}{\sum_{s=1}^{k}\mathbf{f}'_{i_s}(\mathbf{I}-\mathbf{S}_w)\mathbf{f}_{i_s}}. \tag{17}$$

Maximizing the score in above equation is also equivalent to maximize the following term:

$$\frac{\sum_{s=1}^{k}\mathbf{f}'_{i_s}\mathbf{S}_w\mathbf{f}_{i_s}}{\sum_{s=1}^{k}\mathbf{f}'_{i_s}\mathbf{f}_{i_s}} = \frac{\mathbf{X}'_{\mathcal{F}}\mathbf{S}_w\mathbf{X}_{\mathcal{F}}}{\mathbf{X}'_{\mathcal{F}}\mathbf{X}_{\mathcal{F}}}. \tag{18}$$

Since $\mathbf{X}'_{\mathcal{F}}\mathbf{X}_{\mathcal{F}}$ is constant and the above maximization problem can be reduced to the unified similarity based feature selection framework:

$$\max_{\mathcal{F}} \sum_{\mathbf{f}\in\mathcal{F}} \hat{\mathbf{f}}'\hat{\mathbf{S}}\hat{\mathbf{f}}, \text{ where } \hat{\mathbf{f}} = \mathbf{f}/||\mathbf{f}|| \text{ and } \hat{\mathbf{S}} = \mathbf{S}_w. \tag{19}$$

In batch-mode Laplacian Score, the within-class data similarity and the between-class data similarity are defined as follows:

$$\mathbf{S}_w(i,j) = \begin{cases} e^{-\frac{||\mathbf{x}_i-\mathbf{x}_j||^2}{t}} & \text{if } \mathbf{x}_i \in \mathcal{N}_p(\mathbf{x}_j) \text{ or } \mathbf{x}_j \in \mathcal{N}_p(\mathbf{x}_i) \\ 0 & \text{otherwise,} \end{cases} \tag{20}$$

$$\mathbf{S}_b = (\mathbf{1}'\mathbf{D}_w\mathbf{1})^{-1}\mathbf{D}_w\mathbf{1}\mathbf{1}'\mathbf{D}. \tag{21}$$

The trace ratio criterion score can be computed by:

$$trace\_ratio\_laplacian(\mathcal{F}) = \frac{tr(\mathbf{W}'\mathbf{X}'\mathbf{L}_b\mathbf{X}\mathbf{W})}{tr(\mathbf{W}'\mathbf{X}'\mathbf{L}_w\mathbf{X}\mathbf{W})} = \frac{\sum_{s=1}^{k}\mathbf{f}'_{i_s}\mathbf{D}_w\mathbf{f}_{i_s}}{\sum_{s=1}^{k}\mathbf{f}'_{i_s}(\mathbf{D}_w-\mathbf{S}_w)\mathbf{f}_{i_s}}. \tag{22}$$

Maximizing the above trace ratio criterion score is also equivalent to solve the following problem:

$$\max_{\mathcal{F}} \sum_{\mathbf{f}\in\mathcal{F}} \hat{\mathbf{f}}'\hat{\mathbf{S}}\hat{\mathbf{f}}, \text{ where } \hat{\mathbf{f}} = \mathbf{f}/||\mathbf{D}^{\frac{1}{2}}\mathbf{f}|| \text{ and } \hat{\mathbf{S}} = \mathbf{S}_w, \tag{23}$$

therefore it is a special case of the unified framework.

2.1.5 ReliefF (Robnik-Šikonja and Kononenko, 2003) (Supervised)

Relief and its multi-class variant ReliefF are supervised filter algorithms that select features to separate instances from different classes. Assume that l data instances are randomly selected among all n instances, then the feature score of f_i in Relief is defined as follows:

$$Relief\_score(f_i) = \frac{1}{2}\sum_{j=1}^{l} d(\mathbf{X}(j,i) - \mathbf{X}(NM(j),i)) - d(\mathbf{X}(j,i) - \mathbf{X}(NH(j),i)), \quad (24)$$

where $NM(j)$ and $NH(j)$ indicates the nearest data instances to x_j with the same class label and different class, respectively. $d(.)$ is a distance metric which is usually set to be Euclidean distance. Relief only works for binary classification task. To tackle the multiclass classification problem, the feature score in Eq. (24) is extended in ReliefF:

$$ReliefF\_score(f_i) = \frac{1}{c}\sum_{j=1}^{l}\left(-\frac{1}{m_j}\sum_{x_r \in NH(j)} d(\mathbf{X}(j,i) - \mathbf{X}(r,i)) \quad (25)$$

$$+ \sum_{y \neq y_j}\frac{1}{h_{jy}}\frac{p(y)}{1-p(y)}\sum_{x_r \in NM(j,y)} d(\mathbf{X}(j,i) - \mathbf{X}(r,i))\right), \quad (26)$$

where $NH(j)$ and $NM(j,y)$ indicate the nearest data instances to x_j in the same class and a different class y, respectively, and their sizes are h_{jy} and m_j. $p(y)$ is the ratio of instances with class label y.

ReliefF is equivalent to selecting features that preserve a special form of data similarity matrix which can be derived from the class labels. Assume that the dataset has the same number of instances in each of the c classes, there are q instances in both $NM(j)$ and $NH(j,y)$, the Euclidean distance is used and each feature vector has been normalized. Then according to (Zhao and Liu, 2007), the criterion of ReliefF is equivalent to the following with above assumptions:

$$ReliefF\_score(f_i) = \sum_{j=1}^{n}\left(\sum_{s=1}^{q}\frac{1}{q}(\mathbf{X}(j,i) - \mathbf{X}(NM(j)_s))^2 \quad (27)$$

$$- \sum_{y \neq y_j}\frac{\sum_{s=1}^{q}\mathbf{X}(j,i) - \mathbf{X}(NH(j,y)_s)^2}{(c-1)q}\right), \quad (28)$$

where $NM(j)_s$ denote the s-th nearest hit of x_j and $NH(j,y)_s$ denote the s-th nearest miss of x_j in class y. It is easy to show that maximizing the ReliefF score in Eq. (28) is equivalent to the following optimization problem:

$$\max_{\mathcal{F}} \sum_{\mathbf{f} \in \mathcal{F}} -1 + \mathbf{f}'\hat{\mathbf{S}}\mathbf{f}, \quad (29)$$

where the affinity matrix is defined as:

$$\hat{\mathbf{S}}(i,j) = \begin{cases} 1 & \text{if } i = j \\ -\frac{1}{q} & x_j \in NH(i) \\ \frac{1}{(c-1)q} & x_j \in NH(i,y). \end{cases} \quad (30)$$

The optimization can be solved in a greedy manner by selecting the top k features with the highest ReliefF scores.

2.2 Information Theoretical based Methods

A large family of existing feature selection algorithms are information theoretical based methods. Algorithms in this family mainly exploit different heuristic filter criteria to measure the importance of features. As indicated in (Duda et al., 2012), many hand-designed information theoretic criteria are proposed to maximize feature relevance and minimize feature redundancy. Since the relevance of a feature is usually measured by its correlation with class labels, most algorithms in this family are performed in a supervised way. In addition, most information theoretic concepts can only be applied on discrete variables. Therefore, feature selection algorithms in this family can only work with discrete data. For numeric feature values, some data discretization techniques (Dougherty et al., 1995; Kotsiantis and Kanellopoulos, 2006) are required. Two decades of research on information theoretic criteria can be unified in a conditional likelihood maximization framework, and most algorithms can be reduced to be a specific case of the unified framework (Brown et al., 2012) . In this subsection, we introduce some representative algorithms in this family. First, we show the general forms of the algorithms, and then we show how they can be reduced to the unified framework. Before presenting the detailed algorithms, we first give a brief introduction about basic information theoretic concepts.

The first concept is called *entropy*, it is a measure of the uncertainty of a discrete random variable. The entropy of a discrete random variable X is defined as follows:

$$H(X) = -\sum_{x_i \in X} P(x_i) log(P(x_i)), \tag{31}$$

where x_i denotes a specific value of random variable X, $P(x_i)$ denotes the probability of x_i over all possible values of X which can be estimated from the data.

The second concept is the *conditional entropy* of X given another discrete random variable Y:

$$H(X|Y) = \sum_{y_j \in Y} P(y_j) \sum_{x_i \in X} P(x_i|y_j) log(P(x_i|y_j)), \tag{32}$$

where $P(y_i)$ is the prior probability of y_i, while $P(x_i|y_j)$ is the conditional probability of x_i given y_j. The measure of conditional entropy shows the uncertainty of X given another discrete random variable Y.

The concept of *information gain* (Shannon, 2001) between X and Y is used to measure their dependency with entropy and conditional entropy. Since it measures the amount of information shared by X and Y together, it is often referred as *mutual information*, which is calculated as:

$$\begin{aligned} I(X;Y) =& H(X) - H(X|Y) \\ =& \sum_{x_i \in X} \sum_{y_j \in Y} P(x_i, y_j) log \frac{P(x_i, y_j)}{P(x_i)P(y_j)}, \end{aligned} \tag{33}$$

where $P(x_i, y_j)$ is the joint probability of x_i and y_j. It can be noticed that information gain is symmetric such that $I(X;Y) = I(Y;X)$, and is zero if the discrete variables X and Y are independent.

Similar to the concept of entropy, the *conditional information gain* (or *conditional mutual information*) of discrete variables X and Y given the third discrete variable Z is given as follows:

$$\begin{aligned} I(X;Y|Z) =& H(X|Z) - H(X|Y,Z) \\ =& \sum_{z_k \in Z} P(z_k) \sum_{x_i \in X} \sum_{y_j \in Y} P(x_i, y_j|z_k) log \frac{P(x_i, y_j|z_k)}{P(x_i|z_k)P(y_j|z_k)}. \end{aligned} \tag{34}$$

It shows the amount of mutual information shared by X and Y when the third discrete variable Z is given.

Searching for the global best set of features is NP-hard, thus, most algorithms in this family exploit heuristic sequential search approaches to add/remove features one by one. In this survey, we explain the feature selection problem by forward sequential search such that features are added into the selected feature set one by one. We denote $\mathcal{S}$ as the current selected feature set that is initially empty. Y represents the class labels. $X_j \in \mathcal{S}$ is a specific feature in the current $\mathcal{S}$. $J(.)$ is a feature selection criterion (score) where, generally, the higher the value of $J(X_k)$, the more important the feature X_k is. In the unified conditional likelihood maximization feature selection framework, the selection criterion (score) for a new unselected feature X_k is given as follows:

$$J_{cmi}(X_k) = I(X_k;Y) + \sum_{X_j \in \mathcal{S}} g[I(X_j;X_k), I(X_j;X_k|Y)], \tag{35}$$

where $g(.)$ is a function w.r.t. two variables $I(X_j;X_k)$ and $I(X_j;X_k|Y)$. If $g(.)$ is a non-linear function w.r.t. these two variables, it is referred as a criterion by linear combinations of Shannon information terms such that:

$$J_{CMI}(X_k) = I(X_k;Y) - \beta \sum_{X_j \in \mathcal{S}} I(X_j;X_k) + \lambda \sum_{X_j \in \mathcal{S}} I(X_j;X_k|Y). \tag{36}$$

where β and λ are two nonnegative parameters between zero and one. On the other other hand, if $g(.)$ is a linear function w.r.t. these two variables, it is referred as a criterion by non-linear combinations of Shannon information terms.

2.2.1 Mutual Information Maximization (or Information Gain) (Lewis, 1992)

Mutual Information Maximization (MIM)(also known as Information Gain) measures the importance of a feature by its correlation with the class label. It assumes that when a feature has a strong correlation with the class label, it can help achieve good classification performance. The Mutual Information score for a new unselected feature X_k is:

$$J_{MIM}(X_k) = I(X_k;Y). \tag{37}$$

It can be observed that in MIM, the scores of features are assessed individually which is independent of other features. Therefore, in MIM, only the feature correlation is considered

while the feature redundancy property is completely ignored. After it obtains the MIM feature score for all unselected features, we choose the feature with the highest feature score and add it to the selected feature set. The process repeats until the desired number of selected features are obtained.

It can also be observed that MIM is a special case of linear combination of Shannon information terms such that:

$$J_{CMI}(X_k) = I(X_k;Y) - \beta \sum_{X_j \in \mathcal{S}} I(X_j;X_k) + \lambda \sum_{X_j \in \mathcal{S}} I(X_j;X_k|Y), \tag{38}$$

where both β and λ are equal to zero.

2.2.2 Mutual Information Feature Selection (Battiti, 1994)

A limitation of MIM feature selection criterion is that it assumes that features are independent of each other. However, in reality, good features should not only be strongly correlated with class labels, but also should not be highly correlated with each other. In other words, the correlation between features should be minimized. Mutual Information Feature Selection (MIFS) criterion considers both the feature relevance and feature redundancy in the feature selection phase, the feature score for a new unselected feature X_k can be formulated as follows:

$$J_{MIFS}(X_k) = I(X_k;Y) - \beta \sum_{X_j \in \mathcal{S}} I(X_k;X_j). \tag{39}$$

In MIFS, the feature relevance of the new feature is evaluated by the first term $I(X_k;Y)$, while the second term tries to penalize the feature that has a high mutual information with the current selected features such that feature redundancy is minimized. In the original paper, the parameter β is empirically set to be one.

MIFS can also be reduced to be a special case of the linear combination of Shannon information terms:

$$J_{CMI}(X_k) = I(X_k;Y) - \beta \sum_{X_j \in \mathcal{S}} I(X_j;X_k) + \lambda \sum_{X_j \in \mathcal{S}} I(X_j;X_k|Y), \tag{40}$$

where β is between zero and one, and λ is set to be zero.

2.2.3 Minimum Redundancy Maximum Relevance (Peng et al., 2005)

Unlike MIFS that empirically sets β to be one, (Peng et al., 2005) proposed a Minimum Redundancy Maximum Relevance (MRMR) criterion to set the value of β the reverse of the number of selected features:

$$J_{MRMR}(X_k) = I(X_k;Y) - \frac{1}{|\mathcal{S}|} \sum_{X_j \in \mathcal{S}} I(X_k;X_j). \tag{41}$$

With more selected features, the effect of feature redundancy is gradually reduced. The intuition is that with more non-redundant features selected, it is becoming more difficult for new features to be redundant to the features that have already been in $\mathcal{S}$. In (Brown et al.,

2012), it gives another interpretation that the pairwise independence between features becomes stronger as more features are added to $\mathcal{S}$, possibly because of noise information in the data.

The MRMR criterion is strongly linked to the Conditional likelihood maximization framework:

$$J_{CMI}(X_k) = I(X_k;Y) - \beta \sum_{X_j \in \mathcal{S}} I(X_j;X_k) + \lambda \sum_{X_j \in \mathcal{S}} I(X_j;X_k|Y), \tag{42}$$

if we iteratively revise the value of β to be $\frac{1}{|\mathcal{S}|}$, and set the other parameter λ to be zero.

2.2.4 Conditional Infomax Feature Extraction (Lin and Tang, 2006)

MIFS and MRMR consider both feature relevance and feature redundancy at the same time. However, some studies Lin and Tang (2006); El Akadi et al. (2008); Guo and Nixon (2009) show that in contrast to minimize the feature redundancy, the conditional redundancy between unselected features and already selected features given class labels should be maximized. In other words, as long as the feature redundancy given class labels is stronger than the intra feature redundancy, the feature selection will be affected negatively. A typical feature selection under this argument is Conditional Infomax Feature Extraction (CIFE) criterion, in which the feature score for a new unselected feature X_k is:

$$J_{CIFE}(X_k) = I(X_k;Y) - \sum_{X_j \in \mathcal{S}} I(X_j;X_k) + \sum_{X_j \in \mathcal{S}} I(X_j;X_k|Y). \tag{43}$$

It can be observed that compared with MIFS, it adds a third term $\sum_{X_j \in \mathcal{S}} I(X_j;X_k|Y)$ to maximize the conditional redundancy.

CIFE is also a special case of the linear combination of Shannon information terms:

$$J_{CMI}(X_k) = I(X_k;Y) - \beta \sum_{X_j \in \mathcal{S}} I(X_j;X_k) + \lambda \sum_{X_j \in \mathcal{S}} I(X_j;X_k|Y), \tag{44}$$

by setting β and γ to be one.

2.2.5 Joint Mutual Information (Yang and Moody, 1999)

MIFS and MRMR reduce feature redundancy in the feature selection process. An alternative criterion, Joint Mutual Information (Yang and Moody, 1999; Meyer et al., 2008) was proposed to increase complementary information that are shared between new unselected feature and selected features given the class labels. The feature selection criterion is listed as follows:

$$J_{JMI}(X_k) = \sum_{X_j \in \mathcal{S}} I(X_k, X_j;Y). \tag{45}$$

The basic idea of JMI is that we should include new features that are complementary to the existing features given the class labels.

Unlike previous mentioned approaches that can be directly represented by the linear combination of Shannon information terms, JMI can not be directly reduced to the condition

likelihood maximization framework. In (Brown et al., 2012), the authors demonstrate that with simple manipulations, the JMI criterion can be re-written as:

$$J_{JMI}(X_k) = I(X_k;Y) - \frac{1}{|\mathcal{S}|}\sum_{X_j\in\mathcal{S}} I(X_j;X_k) + \frac{1}{|\mathcal{S}|}\sum_{X_j\in\mathcal{S}} I(X_j;X_k|Y). \tag{46}$$

Therefore, it is also a special case of the linear combination of Shannon information terms by iteratively setting β and λ to be $\frac{1}{|\mathcal{S}|}$.

2.2.6 Conditional Mutual Information Maximization (Fleuret, 2004)

Previously mentioned information theoretic feature selection criterion can be reduced to a linear combination of Shannon information terms. Next we show some other algorithms that can only reduce to a non-linear combination of Shannon information terms. Among them, Conditional Mutual Information Maximization (CMIM) (Vidal-Naquet and Ullman, 2003; Fleuret, 2004) is a criterion which iteratively selects features which maximize the mutual information with the class labels given the selected features so far. In other words, CMIM does not select the feature that is similar to previous selected ones even though its predictable power for the class labels is strong. Mathematically, during the selection phase, the feature score for each new unselected feature X_k can be formulated as follows:

$$J_{CMIM}(X_k) = \min_{X_j\in\mathcal{S}}[I(X_k;Y|X_j)]. \tag{47}$$

Note that the value of $I(X_k;Y|X_j)$ is small if X_k is not strongly correlated with the class label Y or if X_k is redundant when $\mathcal{S}$ is known. By selecting the feature that maximizes this minimum value, it can guarantee that the selected feature has strong predictive ability as well as reducing redundancy to the selected features.

The CMIM criterion is equivalent to the following form after some derivations:

$$J_{CMIM}(X_k) = I(X_k;Y) - \max_{X_j\in\mathcal{S}}[I(X_j;X_k) - I(X_j;X_k|Y)]. \tag{48}$$

Therefore, CMIM is also a special case of the conditional likelihood maximization framework:

$$J_{cmi}(X_k) = I(X_k;Y) + \sum_{X_j\in\mathcal{S}} g[I(X_j;X_k), I(X_j;X_k|Y)]. \tag{49}$$

2.2.7 Informative Fragments (Vidal-Naquet and Ullman, 2003)

In (Vidal-Naquet and Ullman, 2003), the authors propose a feature selection criterion called Informative Fragments (IG). The feature score of each new unselected features is given as:

$$J_{IF}(X_k) = \min_{X_j\in\mathcal{S}}[I(X_jX_k;Y) - I(X_j;Y)]. \tag{50}$$

The intuition behind Informative Fragments is that the addition of the new feature X_k should maximize the value of conditional mutual information between X_k and existing features in $\mathcal{S}$ over the mutual information between X_j and Y. An interesting phenomenon of Informative Fragments is that with the chain rule that $I(X_kX_j;Y) = I(X_j;Y) + I(X_k;Y|X_j)$, Informative Fragments has the equivalent form as CMIM, therefore, it can also be reduced to the conditional likelihood maximization framework.

2.2.8 Interaction Capping (Jakulin, 2005)

Interaction Capping is a similar feature selection criterion as CMIM in Eq. (48), but instead of restricting the term $I(X_j; X_k) - I(X_j; X_k|Y)$ to be nonnegative:

$$J_{CMIM}(X_k) = I(X_k; Y) - \sum_{X_j \in \mathcal{S}} \max[0, I(X_j; X_k) - I(X_j; X_k|Y)]. \tag{51}$$

Apparently, it is a special case of non-linear combination of Shannon information terms by setting the function $g(.)$ to be $\max[0, I(X_j; X_k) - I(X_j; X_k|Y)]$.

2.2.9 Double Input Symmetrical Relevance (Meyer and Bontempi, 2006)

Another class of information theoretical based methods such as Double Input Symmetrical Relevance (DISR) (Meyer and Bontempi, 2006) exploits normalization techniques to normalize mutual information (Guyon et al., 2008):

$$J_{DISR}(X_k) = \sum_{X_j \in \mathcal{S}} \frac{I(X_j X_k; Y)}{H(X_j X_k Y)}. \tag{52}$$

It is easy to validate that DISR is a non-linear combination of Shannon information terms and can be reduced to the conditional likelihood maximization framework.

2.2.10 Fast Correlation Based Filter (Yu and Liu, 2003)

There are other information theoretical based feature selection methods that can not be simply be reduced to the unified conditional likelihood maximization framework. Here, we introduce one algorithm named Fast Correlation Based Filter (FCBF) (Yu and Liu, 2003). It is a filter method that exploits feature-class correlation and feature-feature correlation simultaneously. The algorithm works as follows: (1) given a predefined threshold δ, it selects a subset of features $\mathcal{S}$ that is highly correlated with the class label with $SU \geq \delta$, where SU is the symmetric uncertainty, the SU between a set of features $X_{\mathcal{S}}$ and the class label Y is given as follows:

$$SU(X_{\mathcal{S}}, Y) = 2\frac{I(X_{\mathcal{S}}; Y)}{H(X_{\mathcal{S}}) + H(Y)}. \tag{53}$$

A specific feature X_k is called predominant iff $SU(X_k, Y) \geq \delta$ and there does not exist a feature $X_j \in \mathcal{S}(j \neq k)$ such that $SU(X_j, X_k) \geq SU(X_k, Y)$. Feature X_j is considered to be redundant to feature X_k if $SU(X_j, X_k) \geq SU(X_k, Y)$; (2) the set of redundant features is denoted as $\mathcal{S}_{P_i}$, which will be further split into $\mathcal{S}^+_{P_i}$ and $\mathcal{S}^-_{P_i}$ where they contain redundant features to feature X_k with $SU(X_j, Y) > SU(X_k, Y)$ and $SU(X_j, Y) < SU(X_k, Y)$, respectively; and (3) different heuristics are applied on $\mathcal{S}_P$, $\mathcal{S}^+_{P_i}$ and $\mathcal{S}^-_{P_i}$ to remove reundant features and keep the features that are most relevant features to the class label. Different from feature weighting methods that assign a score to each feature and select the features with the highest score, FCBF is a subset search algorithm which cannot determine the number of selected features.

2.3 Sparse Learning based Methods

Filter feature selection methods select features that are independent of any learning algorithms. However, they do not take into account the bias of the learning algorithms such that the selected features may not be optimal for a specific learning task. To tackle this issue, embedded methods embed the feature selection phase into the learning algorithm construction where these two phases compliment each other. The selected features are suitable for that learning algorithm which can be used for further analysis. There are three main types of embedded feature selection methods: The first type of embedded methods are pruning methods. At the very beginning, they use the whole set of features to train a learning model and then attempt to remove some features by setting the feature coefficients to zero, while maintaining the model performance, one example method in this category is recursive feature elimination methods using support vector machine (SVM) (Guyon et al., 2002). The second type of embedded methods contain a built-in feature selection mechanism such as ID3 (Quinlan, 1986) and C4.5 (Quinlan, 1993). The third type of methods are sparse learning based methods which aim to minimize the fitting errors along with some sparse regularization terms. The sparse regularizer forces some feature coefficients to be small or exactly zero, and then the corresponding features can be simply eliminated. Sparse learning based methods have received great attention in recent years due to their good performance and interpretability. In the following subsections, we review the sparse learning based feature selection methods in both supervised and unsupervised perspective. We first cover some representative supervised sparse learning based methods and then introduce some unsupervised sparse learning based methods.

2.3.1 Feature Selection with ℓ_1-norm Regularizer (Supervised) (Tibshirani, 1996; Hastie et al., 2015)

First, we consider the binary classification (y_i is either 0 or 1) or regression problem with only one regression target. Without loss of generality, we only consider linear classification or linear regression model, but it can be easily extended to non-linear problems. The classification label or regression target $\mathbf{y}$ can be considered as a linear combination of data instances $\mathbf{X}$ like SVM (Cortes and Vapnik, 1995) and logistic regression (Hosmer Jr and Lemeshow, 2004). To achieve feature selection, the ℓ_1-norm penalty term is added on the classification or regression model. One main advantage of ℓ_1-norm regularization (Lasso) (Tibshirani, 1996; Hastie et al., 2015) is that it forces some feature coefficients to to become smaller and, in some cases, exactly zero. This property makes it suitable for feature selection, as we can select features with corresponding non-zero coefficients. Specifically, let $\mathbf{w}$ denote the model parameter (feature coefficient) that can be obtained by solving following optimization problem:

$$\mathbf{w} = \underset{\mathbf{w}}{\operatorname{argmin}}\ loss(\mathbf{w}; \mathbf{X}, \mathbf{y}) + \alpha\, penalty(\mathbf{w}), \tag{54}$$

where $loss(.)$ is a loss function which denotes a classification or regression model, $penalty(\mathbf{w})$ is a sparse regularization term for feature selection, and α is a regularization parameter to balance the contribution of the loss function and the regularization term.

Some widely used loss functions $loss(.)$ include least squares, hinge loss and logistic loss. They are defined as follows:

- Least Square Loss:

$$loss(\mathbf{w}; \mathbf{X}, \mathbf{y}) = \sum_{i=1}^{n} (y_i - \mathbf{w}'\mathbf{x}_i)^2, \tag{55}$$

- Hinge Loss:

$$loss(\mathbf{w}; \mathbf{X}, \mathbf{y}) = \sum_{i=1}^{n} \max(0, 1 - y_i \mathbf{w}'\mathbf{x}_i), \tag{56}$$

- Logistic Loss:

$$loss(\mathbf{w}; \mathbf{X}, \mathbf{y}) = \sum_{i=1}^{n} log(1 + exp(-y_i \mathbf{w}'\mathbf{x}_i)), \tag{57}$$

As mentioned above, the most widely used sparse regularization is ℓ_1-norm regularization, but there are also different types of sparse regularization such as adaptive lasso and elastic net. Next, we briefly introduce these sparse regularization terms.

- Lasso Regularization (Tibshirani, 1996): Lasso is short for *least absolute shrinkage and selection operator*, it is based on the ℓ_1-norm regularization term on the feature coefficient $\mathbf{w}$:

$$penalty(\mathbf{w}) = ||\mathbf{w}||_1 = \sum_{i=1}^{d} |\mathbf{w}_i|. \tag{58}$$

  The ℓ_1-norm regularization term forces some feature coefficient to be zero and the corresponding features can be simply eliminated since the elimination of these features will not greatly affect the learning performance. After obtaining the feature weight $\mathbf{w}$ by some optimization algorithms, we can sort the feature importance according to the feature weight – the higher the feature weight, the more important the feature is.

- Adaptive Lasso Regularization (Zou, 2006): The Lasso variable selection phase is consistent if it satisfies non-trivial solutions. However, this condition is difficult to satisfy in some scenarios (Zhao and Yu, 2006). Another critical issue of Lasso is that the lasso shrinkage produces biased estimates for the large coefficients, and thus it could be suboptimal in terms of estimation risk (Fan and Li, 2001). To tackle these problems, the adaptive Lasso regularization is proposed:

$$penalty(\mathbf{w}) = \sum_{i=1}^{d} \frac{|\mathbf{w}_i|}{\mathbf{b}_i}, \tag{59}$$

  where $\mathbf{b}$ is a given weight vector to control the contribution of each feature coefficient in the ℓ_1-norm penalty term. It is easy to see that adaptive Lasso is a weighted version of Lasso. In (Zou, 2006), the authors show that the adaptive lasso enjoys the oracle properties and can be solved by the same efficient algorithm for solving the Lasso.

- Elastic Net Regularization (Zou and Hastie, 2005): In Lasso, the number of selected features is usually bounded by the number of data instances, which is unrealistic in many applications. In addition, in many applications such as bioinformatics, image processing and natural language processing (Mitra et al., 2002; Segal et al., 2003;

Liu et al., 2007), it is common that features may have some strong correlations with each other. However, Lasso tends to randomly select features from a group and discards the others. To handle features with high correlations, Elastic Net regularization (Zou and Hastie, 2005) is proposed as:

$$penalty(\mathbf{w}) = \sum_{i=1}^{d} |\mathbf{w}_i|^{\gamma} + (\sum_{i=1}^{d} \mathbf{w}_i^2)^{\lambda}, \tag{60}$$

with $0 < \gamma \leq 1$ and $\lambda \geq 0$. The parameters γ and λ are usually set to be 1 such that Elastic Net regularization is simply a combination of ℓ_1-norm and ℓ_2-norm regularization.

In addition to the different variations of Lasso regularization, another way to obtain a sparse representation of feature coefficients $\mathbf{w}$ is Dantzig selector; it is based on the normal score equations and controls the correlation of residuals with $\mathbf{X}$ as:

$$\begin{aligned} &\min_{\mathbf{w}} ||\mathbf{w}||_1 \\ &s.t. ||\mathbf{X}'(\mathbf{y} - \mathbf{X}\mathbf{w})||_{\infty} \leq \lambda, \end{aligned} \tag{61}$$

where $||.||_{\infty}$ denotes the infinity norm of a vector. Dantzig selector was designed for linear regression models. In (Candes and Tao, 2007; James et al., 2009), the authors show that the errors of Dantzig selector is up to a logarithmic factor $log(d)$ (d is the feature dimensionality). Strong theoretical results in (James et al., 2009) show that LASSO and Dantzig selector are closely related.

2.3.2 Feature Selection with $\ell_{2,1}$-norm Regularizer (Supervised) (Liu et al., 2009b)

As mentioned above, for binary classification and regression with one target, we can achieve feature selection via ℓ_1-norm regularization since it will force some feature coefficient to be exact zero. Here, we discuss how to perform feature selection for the general classification problems. The problem is more difficult because of multiple classification or regression targets since we would like the feature selection phase to be consistent over multiple targets. In other words, we want multiple predictive models for different targets to share the similar parameter sparsity patterns – each feature either has small scores for all data points or has large scores over all data points. This problem can be solved by the $\ell_{2,1}$-norm regularization which is widely applied in many applications (Obozinski et al., 2007; Evgeniou and Pontil, 2007; Bi et al., 2008; Zhang et al., 2008). Similar to ℓ_1-norm regularization, $\ell_{2,1}$-norm regularization is also convex and a global optimal solution can be achieved. The $\ell_{2,1}$-norm regularization has strong connections with group lasso (Yuan and Lin, 2006) which will be explained later.

Assume that $\mathbf{X}$ denotes the data matrix, and $\mathbf{y}$ denotes the label vector such that it contains s different class labels $\{c_1, c_2, ..., c_s\}$. First, we can transform the feature vector $\mathbf{y}$ to one-hot label matrix $\mathbf{Y}$ such that if $\mathbf{y}_i = c_j$ then the only the j-th element in the corresponding row vector $\mathbf{Y}(i,:)$ is 1, other elements are 0. We further assume that the linear classification problem is parameterized by a weight matrix $\mathbf{W}$ such that the j-th

column of $\mathbf{W}$ contains the feature coefficient for the j-th class label. If the least square loss function is specified, then the model is formulated as follows:

$$\min_{\mathbf{W}} ||\mathbf{XW} - \mathbf{Y}||_F^2 + \alpha||\mathbf{W}||_{2,1}, \tag{62}$$

where the parameter α is used to control the contribution from the loss function and the regularization term. By solving this optimization problem, we can obtain a sparse matrix $\mathbf{W}$ where many rows are exact zero or small numbers. The features corresponding to these rows can then be eliminated. By selecting a few number of rows, it achieves joint feature selection among different classification labels.

Similar to Lasso, we can also exploit different loss functions such as hinge loss and logistic loss. Meanwhile, the $\ell_{2,1}$-norm regularization can also be modified as adaptive Lasso and elastic net. Generally, the $\ell_{2,1}$-norm regularization problem can be solved efficiently by the state-of-the-art proximal gradient descent methods (Liu et al., 2009a). After we obtain the feature coefficient matrix $\mathbf{W} \in \mathbb{R}^{d \times s}$, we can compute the ℓ_2-norm of each row vector $||\mathbf{W}(i,:)||_2$ which corresponds to the i-th feature – the larger the value of the ℓ_2-norm, the more important the feature is.

2.3.3 Efficient and Robust Feature Selection (Supervised) (Nie et al., 2010)

In (Nie et al., 2010), the authors propose an efficient and robust feature selection (REFS) method by employing a joint $\ell_{2,1}$-norm minimization on both the loss function and the regularization. Their argument is that the ℓ_2-norm based loss function is sensitive to noisy data while the $\ell_{2,1}$-norm based loss function is more robust to noise. The reason is that $\ell_{2,1}$-norm loss function has a rotational invariant property (Ding et al., 2006). Consistent with $\ell_{2,1}$-norm regularized feature selection model, a $\ell_{2,1}$-norm regularizer is added to the $\ell_{2,1}$-norm loss function to achieve group sparsity. The objective function of REFS is formulated as follows:

$$\min_{\mathbf{W}} ||\mathbf{XW} - \mathbf{Y}||_{2,1} + \alpha||\mathbf{W}||_{2,1}, \tag{63}$$

The objective function of REFS is difficult to solve since both terms are convex but non-smooth. In (Nie et al., 2010), an efficient algorithm is proposed to solve this optimization problem with strict convergence analysis.

2.3.4 Multi-Cluster Feature Selection (Unsupervised) (Cai et al., 2010)

Most of existing sparse learning based approaches build a learning model with the supervision of class labels. The feature selection phase is derived afterwards on the sparse feature coefficients. However, since labeled data is costly and time consuming to obtain, unsupervised sparse learning based feature selection has received increasing attention in recent years (Cai et al., 2010; Yang et al., 2011; Hou et al., 2011; Li et al., 2012; Qian and Zhai, 2013; Liu et al., 2014; Du and Shen, 2015). Multi-Cluster Feature Selection (MCFS) (Cai et al., 2010) is one of the first unsupervised feature selection algorithm using sparse learning techniques. Without class labels to guide the feature selection process, MCFS proposes to select features that can cover the multi-cluster structure of the data where spectral analysis is used to measure the correlation between different features.

MCFS consists of three steps: (1) the spectral clustering step, (2) sparse coefficient learning step and (3) feature selection step. In the first step, spectral clustering (Chan et al., 1994; Ng et al., 2002) is applied on the dataset to detect the cluster structure of the data. It first constructs a k-nearest neighbor graph to capture the local geometric structure of the data and gets the graph affinity matrix $\mathbf{S}$, where k is a predefined parameter. There are many different ways to define the affinity matrix. Typical ways include 0-1 weighting, heart kernel weighting and dot-product weighting. For example, if the affinity matrix is built by the heart kernel weighting scheme, $\mathbf{S}$ is computed as follows:

$$\mathbf{S}(i,j) = e^{\frac{-||\mathbf{x}_i - \mathbf{x}_j||^2}{\sigma}}, \tag{64}$$

where $\mathbf{x}_i$ and $\mathbf{x}_j$ are connected instances in the k-nearest neighbor graph and the bandwidth σ is a predefined parameter. Then, the graph Laplacian matrix is defined as $\mathbf{L} = \mathbf{D} - \mathbf{W}$, where $\mathbf{D}$ is a diagonal matrix with its diagonal element $D(i,i) = \sum_j \mathbf{W}(i,j)$. With the graph Laplacian matrix, the flat embedding that unfolds the data manifold can be obtained by solving the following generalized eigen-problem:

$$\mathbf{L}\mathbf{e} = \lambda \mathbf{D}\mathbf{e}. \tag{65}$$

Let $\mathbf{E} = \{\mathbf{e}_1, \mathbf{e}_2, ..., \mathbf{e}_K\}$ denote the eigenvectors of the above eigen-problem w.r.t. the smallest K eigenvalues. Each column of $\mathbf{E}$, i.e., $\mathbf{e}_i$ denotes the i-th embedding of the data $\mathbf{X}$, and K denotes the intrinsic dimensionality of the data that is usually set to the number of clusters if the number of clusters is known in advance.

In the second step, since the embedding of the data $\mathbf{X}$ is known, MCFS takes advantage of them to measure the importance of a feature by a regression model with a ℓ_1-norm regularization. Specifically, given the i-th embedding $\mathbf{e}_i$, MCFS regards it as a regression target to minimize the following objective function:

$$\min_{w_i} ||\mathbf{X}\mathbf{w}_i - \mathbf{e}_i||_2^2 + \alpha||\mathbf{w}_i||_1, \tag{66}$$

where $\mathbf{w}_i$ denotes the feature coefficient vector for the i-th embedding.

By solving all K sparse regression problems, MCFS obtains K sparse feature coefficient vectors $\mathbf{W} = [\mathbf{w}_1, ..., \mathbf{w}_K]$ and each vector corresponds to one embedding of $\mathbf{X}$. In the third step, for each feature f_j, the MCFS score for that feature can be computed as:

$$MCFS(j) = \max_i |\mathbf{W}(j,i)|. \tag{67}$$

The higher the MCFS score, the more important the feature is.

2.3.5 $\ell_{2,1}$-norm Regularized Discriminative Feature Selection (Unsupervised) (Yang et al., 2011)

To perform unsupervised feature selection, one widely accepted criterion is to select features that best preserve the manifold structure of the data (He et al., 2005; Zhao and Liu, 2007; Cai et al., 2010). An alternative way is to exploit the discriminative information encoded in the data that has been proven to be effective in many learning tasks (Fukunaga, 2013). In (Yang et al., 2011), the authors propose a new unsupervised feature selection algorithm

(UDFS) to select the most discriminative features by exploiting both the discriminative information and feature correlations.

First, we briefly introduce discriminative information. Suppose n instances come from s classes and there are n_i instances in the i-th class. $\mathbf{Y} \in \{0,1\}^{n \times s}$ denotes the class label matrix for n instances such that $\mathbf{Y}(i,j) = 1$ if $\mathbf{x}_i$ belongs to the j-th class, otherwise $\mathbf{Y}(i,j) = 0$. Let $\mathbf{H}_n = \mathbf{I}_n - \frac{1}{n}\mathbf{1}_n\mathbf{1}_n'$, then the total scatter matrix $\mathbf{S}_t$ and between class scatter matrix $\mathbf{S}_b$ are defined as follows:

$$\begin{aligned} \mathbf{S}_t &= \sum_{i=1}^{n} (\mathbf{x_i} - \mu)(\mathbf{x_i} - \mu)' = \tilde{\mathbf{X}}'\tilde{\mathbf{X}} \\ \mathbf{S}_b &= \sum_{i=1}^{c} n_i (\mu_\mathbf{i} - \mu)(\mu_\mathbf{i} - \mu)' = \tilde{\mathbf{X}}'\mathbf{G}\mathbf{G}'\tilde{\mathbf{X}}, \end{aligned} \tag{68}$$

where $\mu = \frac{\mathbf{x}_1 + ... + \mathbf{x}_n}{n}$ is the mean of all data instances, μ_i is the mean of all instances in the i-th class, $\tilde{\mathbf{X}}$ is the centered data matrix such $\tilde{\mathbf{X}} = \mathbf{H}_n\mathbf{X}$ and $\mathbf{G} = [\mathbf{G}_1, \mathbf{G}_1, ..., \mathbf{G}_n]' = \mathbf{Y}(\mathbf{Y}'\mathbf{Y})^{-\frac{1}{2}}$ is the weighted label indicator matrix. Linear discriminant analysis aims to obtain a linear transformation matrix $\mathbf{W} \in \mathbb{R}^{d \times s}$ that projects $\mathbf{X}$ from a d-dimensional space to a low dimensional space such that $\mathbf{S}_t$ is minimized and $\mathbf{S}_b$ is maximized.

Instead of using global discriminative information, authors in (Yang et al., 2011) propose to utilize the local discriminative information (Sugiyama, 2006; Yang et al., 2010) to select discriminative features. The advantage of using local discriminative information are two folds. First, it has been demonstrated to be more important than global discriminative information in many classification and clustering tasks. Second, when it considers the local discriminative information, the data manifold structure is also taken into account. For each data instance x_i, it constructs a k-nearest neighbor set for that instance $\mathcal{N}_k(x_i) = \{x_{i_1}, x_{i_2}, ..., x_{i_k}\}$. Let $\mathbf{X}_{\mathcal{N}_k(i)} = [\mathbf{x}_i, \mathbf{x}_{i_1}, ..., \mathbf{x}_{i_k}]$ denote the local data matrix around x_i, then the local total scatter matrix $\mathbf{S}_t^{(i)}$ and local between class scatter matrix $\mathbf{S}_b^{(i)}$ are defined as:

$$\begin{aligned} \mathbf{S}_t^{(i)} &= \tilde{\mathbf{X}_i}'\tilde{\mathbf{X}_i} \\ \mathbf{S}_b^{(i)} &= \tilde{\mathbf{X}_i}'\mathbf{G}_i\mathbf{G}_i'\tilde{\mathbf{X}_i}, \end{aligned} \tag{69}$$

where $\tilde{\mathbf{X}_i} = \mathbf{H}_{k+1}\mathbf{X}_i$ and $\mathbf{G}_{(i)} = [\mathbf{G}_i, \mathbf{G}_{i_1}, ..., \mathbf{G}_{i_k}]'$. Note that $\mathbf{G}_{(i)}$ is a subset from $\mathbf{G}$ and $\mathbf{G}_{(i)}$ can be obtained by a selection matrix $\mathbf{P}_i \in \{0,1\}^{n \times (k+1)}$ such that $\mathbf{G}_{(i)} = \mathbf{P}_{(i)}'\mathbf{G}$. Without label information in unsupervised feature selection, UDFS assumes that there is a linear classifier $\mathbf{W} \in \mathbb{R}^{d \times s}$ to map each data instance $\mathbf{x}_i \in \mathbb{R}^d$ to a low dimensional space $\mathbf{G}_i \in \mathbb{R}^s$. Following the definition of global discriminative information (Yang et al., 2010; Fukunaga, 2013), the local discriminative score for each instance x_i is computed as:

$$\begin{aligned} DS_i &= tr[(\mathbf{S}_t^{(i)} + \lambda\mathbf{I}_d)^{-1}\mathbf{S}_b^{(i)}] \\ &= tr[\mathbf{W}'\mathbf{X}'\mathbf{P}_{(i)}\tilde{\mathbf{X}_i}'(\tilde{\mathbf{X}_i}\tilde{\mathbf{X}_i}' + \lambda\mathbf{I}_d)^{-1}\tilde{\mathbf{X}_i}\mathbf{P}_{(i)}'\mathbf{X}\mathbf{W}], \end{aligned} \tag{70}$$

where λ is a small number to make $\tilde{\mathbf{X}_i}\tilde{\mathbf{X}_i}'$ invertible. If the instance has a high local discriminative score, it indicates that the linear classifier $\mathbf{W}$ can discriminate these instances

well. Therefore, UDFS tends to train $\mathbf{W}$ which obtains the highest local discriminative score for all instances in $\mathbf{X}$; it incorporates a $\ell_{2,1}$-norm regularizer to achieve feature selection, the objective function is formulated as follows:

$$\min_{\mathbf{W}'\mathbf{W}=\mathbf{I}_d} \sum_{i=1}^{n} \{tr[\mathbf{G}'_{(i)}\mathbf{H}_{k+1}\mathbf{G}_{(i)} - DS_i]\} + \alpha||\mathbf{W}||_{2,1}, \tag{71}$$

where $\mathbf{G}'_{(i)}\mathbf{H}_{k+1}\mathbf{G}_{(i)}$ is added to avoid overfit and $||\mathbf{W}||_{2,1}$ will make $\mathbf{W}$ to be sparse in rows, which is suitable for feature selection, α is a regularization parameter to control the sparsity of the model. With some mathematical derivation, the objective function in Eq. (71) can be reformulated as follows:

$$\min_{\mathbf{W}'\mathbf{W}=\mathbf{I}_d} tr(\mathbf{W}'\mathbf{M}\mathbf{W}) + \alpha||\mathbf{W}||_{2,1}, \tag{72}$$

where $\mathbf{M} = \mathbf{X}'[\sum_{i=1}^{n}(\mathbf{P}_i\mathbf{H}_{k+1}(\tilde{\mathbf{X}}_i\tilde{\mathbf{X}}_i + \lambda\mathbf{I}_{k+1})^{-1}\mathbf{H}_{k+1}\mathbf{S}'_i)]\mathbf{X}$. After we obtain the sparse coefficient matrix, we can rank the features according to its ℓ_2-norm value and return the top ranked ones.

2.3.6 Feature Selection Using Nonnegative Spectral Analysis (Unsupervised) (Li et al., 2012)

In addition to UDFS, there are some other ways to exploit discriminative information for unsupervised feature selection. Nonnegative Discriminative Feature Selection (NDFS) (Hou et al., 2011) is an algorithm which performs spectral clustering and feature selection simultaneously in a joint framework to select a subset of discriminative features. NDFS assumes that pseudo class label indicators can be obtained by spectral clustering techniques. Different from most existing spectral clustering techniques, NDFS imposes nonnegative and orthogonal constraints during the spectral clustering phase. The argument is that with these constraints, the learnt pseudo class labels are more close to real cluster results. These nonnegative pseudo class labels then act as regression constraints to guide the feature selection phase. Instead of performing these two tasks separately, NDFS incorporates these two phases into a joint framework.

Suppose that these n data instances come from s classes $\{c_1, ..., c_s\}$ and $\mathbf{Y} \in \{0,1\}^{n\times s}$ denotes the class label matrix such that $\mathbf{Y}(i,j) = 1$ if $\mathbf{x}_i$ belongs to j-th class, otherwise $\mathbf{Y}(i,j) = 0$. Similar to the definition in UDFS, we use $\mathbf{G} = [\mathbf{G}_1, \mathbf{G}_1, ..., \mathbf{G}_n]' = \mathbf{Y}(\mathbf{Y}'\mathbf{Y})^{-\frac{1}{2}}$ to denote the weight cluster indicator matrix. It is easy to show that for the weight cluster indicator matrix $\mathbf{G}$, we have $\mathbf{G}\mathbf{G}' = \mathbf{I}_n$. NDFS adopts a strategy to learn the weight cluster matrix such that the local geometric structure of the data can be well preserved (Shi and Malik, 2000; Yu and Shi, 2003). Since the local geometric structure of the dataset can be modeled by a k-nearest neighbor graph, the affinity matrix $\mathbf{S}$ is defined as:

$$\mathbf{S}(i,j) = e^{\frac{-||\mathbf{x}_i - \mathbf{x}_j||^2}{\sigma}}. \tag{73}$$

$\mathbf{x}_i$ and $\mathbf{x}_j$ are connected instances in the k-nearest neighbor graph, where the bandwidth σ is a predefined parameter. Then the local geometric structure can be fully used by minimizing

the following normalized graph Laplacian:

$$\min_{\mathbf{G}} \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\mathbf{S}(i,j)||\frac{\mathbf{G}_i}{\sqrt{\mathbf{D}(i,i)}}-\frac{\mathbf{G}_j}{\sqrt{\mathbf{D}(j,j)}}||_2^2 = tr(\mathbf{G}'\mathbf{L}\mathbf{G}), \tag{74}$$

where $\mathbf{D}$ is a diagonal matrix with its diagonal element $\mathbf{D}(i,i) = \sum_{j=1}^{n}\mathbf{S}(i,j)$, $\mathbf{L} = \mathbf{D}^{-\frac{1}{2}}(\mathbf{D}-\mathbf{S})\mathbf{D}^{-\frac{1}{2}}$ is the normalized Laplacian matrix. Given the pseudo labels $\mathbf{G}$, NDFS assumes that there exists a linear transformation matrix $\mathbf{W} \in \mathbb{R}^{d\times s}$ between the data instances $\mathbf{X}$ and pseudo labels $\mathbf{G}$. These pseudo class labels are utilized as constraints to guide the feature selection process by minimizing the following objective function:

$$\begin{aligned}\min_{\mathbf{W}} &||\mathbf{X}\mathbf{W}-\mathbf{G}||_F^2+\alpha||\mathbf{W}||_{2,1}\\ \text{s.t.}\quad &\mathbf{G}\mathbf{G}'=\mathbf{I_n}, \mathbf{G}\geq 0,\end{aligned} \tag{75}$$

where α is a parameter to control the sparsity of the model. $||\mathbf{W}||_{2,1}$ ensures that NDFS achieves group sparsity among all s psuedo class labels. Note that NDFS imposes a non-negative constraint on $\mathbf{G}$ because with both the orthogonal and nonnegative constraint, there is only one positive element in each row of $\mathbf{G}$ and all the other elements are zero. In this way, the learnt pseudo class labels $\mathbf{G}$ are more accurate and can better help to select discriminative features.

By combining the objective functions in Eq. (74) and Eq. (75), the objective function of NDFS is formulated as follows:

$$\begin{aligned}\min_{\mathbf{G},\mathbf{W}} &\ tr(\mathbf{G}'\mathbf{L}\mathbf{G})+\beta(||\mathbf{X}\mathbf{W}-\mathbf{G}||_F^2+\alpha||\mathbf{W}||_{2,1})\\ \text{s.t.}\quad &\mathbf{G}\mathbf{G}'=\mathbf{I_n}, \mathbf{G}\geq 0.\end{aligned} \tag{76}$$

The objective function of NDFS in Eq. (76) can be solved in an alternating way. After obtaining the feature coefficient matrix $\mathbf{W}$, NDFS takes the same strategy as UDFS to rank the features according to its ℓ_2-norm value in $\mathbf{W}$ and return the top ranked ones.

2.3.7 Feature selection via joint embedding learning and sparse regression (Unsupervised) (Hou et al., 2011)

Feature selection via joint embedding learning and sparse regression (JELSR) (Hou et al., 2011) is an unsupervised feature selection that is similar to NDFS. They both embed the pseudo class label learning process into the sparse regression for feature selection. The difference is that NDFS uses graph Laplacian to learn the pseudo class labels, while JELSR utilizes local linear embedding. In addition, NDFS imposes both orthogonal and nonnegative constraints on the pseudo class labels while JELSR only considers the orthogonal constraint.

In the first step, JELSR builds a k-nearest neighbor graph. In the second step, instead of using some explicit affinity matrix $\mathbf{S}$ like MCFS and NDFS, JELSR takes advantage of the local linear embedding (Roweis and Saul, 2000) to learn the local approximation matrix, i.e., affinity matrix. Specifically, if the i-th instance x_i and the j-th instance x_j

are connected in the k-nearest neighbor graph; its entry in the affinity matrix is positive $\mathbf{S}(i,j) > 0$, otherwise $\mathbf{S}(i,j) = 0$. The nonzero entry can be learnt by solving the following optimization problems:

$$\begin{aligned} \min_{\mathbf{S}} \sum_{i=1}^{n} ||\mathbf{x}_i - \sum_{j=1}^{n} \mathbf{S}(i,j)\mathbf{x}_j||_2^2 \\ \text{s.t.} \quad \sum_{j=1}^{n} \mathbf{S}(i,j) = 1. \end{aligned} \tag{77}$$

The basic idea is that each instance in $\mathbf{X}$ can be approximated by a linear combination of some of its k-nearest neighbors. JELSR assumes that the low dimensional representation, i.e., pesudo class labels, shares the same local structure as the instances $\mathbf{X}$ in the high dimensional space, therefore it aims to minimize the following:

$$\begin{aligned} \min_{\mathbf{G}} \sum_{i=1}^{n} ||\mathbf{G}_i - \sum_{j=1}^{n} \mathbf{S}(i,j)\mathbf{G}_j||_2^2 = tr(\mathbf{G}'\mathbf{L}\mathbf{G}) \\ \text{s.t.} \quad \mathbf{G}\mathbf{G}' = \mathbf{I_n}. \end{aligned} \tag{78}$$

By integrating the embedding learning phase into the feature selection phase, the objective function of JELSR is formulated as follows:

$$\begin{aligned} \min_{\mathbf{G},\mathbf{W}} tr(\mathbf{G}'\mathbf{L}\mathbf{G}) + \beta(||\mathbf{X}\mathbf{W} - \mathbf{G}||_F^2 + \alpha||\mathbf{W}||_{2,1}) \\ \text{s.t.} \quad \mathbf{G}\mathbf{G}' = \mathbf{I_n}. \end{aligned} \tag{79}$$

where α and β are two balance parameters. The $\ell_{2,1}$-norm regularization term ensures that $\mathbf{W}$ is sparse in rows. Similar to UDFS and NDFS, after deriving $\mathbf{W}$ by some optimization algorithms, it uses ℓ_2-norm, i.e., $\mathbf{W}(i,:)$ to rank features. The higher ranked features are relatively more important.

2.4 Statistical based Methods

Another category of feature selection algorithms is based on different statistical measures; we group them as statistical based methods in this survey. Since they rely on some statistical measures instead of learning algorithms to evaluate the relevance of features, most of them are filter based methods. In addition, most statistical based algorithms analyze features individually. Hence, feature redundancies is inevitably ignored during the selection phase. We introduce some representative feature selection algorithms in this category. Note that the vast majority algorithms in this category works with discrete data, numerical datasets have to perform discretization first.

2.4.1 Low Variance (Pedregosa et al., 2011) (Unsupervised)

Low Variance is a simple feature selection algorithm which eliminates the feature whose variance is below some threshold. For example, for the features that have the same values on all instances, the variance is 0 and should be removed since it cannot help to discriminate instances from different classes. Suppose that the dataset consists of only boolean features,

i.e., the feature values are either 0 and 1, since the boolean features are Bernoulli random variables, its variance value can be computed as:

$$variance\_score(f_i) = p(1-p), \tag{80}$$

where p denotes the percentage of instances that take the feature value of 1. After the variance of features are obtained, the feature with a variance score below a predefined threshold can be directly pruned.

2.4.2 T-score (Davis and Sampson, 1986) (Supervised)

t-score is used for binary classification problems. For each feature f_i, suppose that μ_1 and μ_2 are the mean feature values for the instances from the first class and the second class, σ_1 and σ_2 are the corresponding standard deviation values, n_1 and n_2 denote the number of instances from these two classes. Then the t-score for the feature f_i can be computed as:

$$t\_score(f_i) = \frac{|\mu_1 - \mu_2|}{\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}}. \tag{81}$$

The basic idea of t-score is to assess whether the feature can make the means of two classes to be different statistically by computing the ratio between the mean difference and the variance of two classes. Usually, the higher the t-score, the more important the feature is.

2.4.3 F-score (Wright, 1965) (Supervised)

t-score can only be applied for binary classification task, therefore it has some limitations. f-score can handle the multi-class situation by testing if a feature is able to well separate samples from different classes. Considering both the within class variance and between class variance, the f-score can be computed as follows:

$$f\_score(f_i) = \frac{\sum_j \frac{n_j}{c-1}(\mu_j - \mu)^2}{\frac{1}{n-c}\sum_j (n_j - 1)\sigma_j^2}. \tag{82}$$

Given feature f_i, n_j, μ, μ_j, σ_j denote the number of instances from class j, the mean feature value, the mean feature value on class j, the standard deviation of feature value on class j, respectively. Similar to t-score, the higher the t-score, the more important the feature is.

2.4.4 Chi-Square Score (Liu and Setiono, 1995) (Supervised)

Chi-square score utilizes the test of independence to assess whether the feature is independent of the class label. Given a particular feature with r different feature values, the Chi-square score of that feature can be computed as:

$$Chi\_square\_score(f_i) = \sum_{j=1}^{r}\sum_{s=1}^{c}\frac{(n_{js} - \mu_{js})^2}{\mu_{js}}, \tag{83}$$

where n_{js} is the number of instances with the j-th feature value given feature f_i. In addition, $\mu_{js} = \frac{n_{*s}n_{j*}}{n}$, where n_{j*} indicates the number of data instances with the j-th feature value given feature f_i, n_{*s} denotes the number of data instances in class r. A higher Chi-square score indicates that the feature is relatively more important.

2.4.5 Gini Index (Gini, 1912) (Supervised)

Gini index is a statistical measure to quantify if the feature is able to separate instances from different classes. Given a feature f_i with r different feature values, for the j-th feature value, let $\mathcal{W}$ denote the set of instances with the feature value smaller than or equal to the j-th feature value, let $\overline{\mathcal{W}}$ denote the set of instances with the feature value larger than the j-th feature value. In other words, the j-th feature value can separate the dataset into $\mathcal{W}$ and $\overline{\mathcal{W}}$, then the Gini index score for the feature f_i is given as follows:

$$gini\_index\_score(f_i) = \min_{\mathcal{W}} \left(p(\mathcal{W})(1 - \sum_{s=1}^{c} p(C_s|\mathcal{W})^2) + p(\overline{\mathcal{W}})(1 - \sum_{s=1}^{c} p(C_s|\overline{\mathcal{W}})^2) \right), \quad (84)$$

where C_s indicates that the class label is s. $p(.)$ denotes the probability, for instance, $p(C_s|\mathcal{W})$ indicates the conditional probability of class s given the set of $\mathcal{W}$. In Eq. (84), the gini index score is obtained by going through all the possible split $\mathcal{W}$. Usually for binary classification problem, it can take a maximum value of 0.5, but it can also be used in multi-classification problems. Unlike previous statistical measures, the lower the gini index value, the more relevant the feature is.

2.4.6 CFS (Hall and Smith, 1999) (Supervised)

The basic idea of CFS is to use a correlation based heuristic to evaluate the worth of feature subset $\mathcal{F}$:

$$CFS\_score(\mathcal{F}) = \frac{k\overline{r_{cf}}}{\sqrt{k + k(k-1)\overline{r_{ff}}}}, \quad (85)$$

where the CFS score shows the heuristic "merit" of the feature subset $\mathcal{F}$ with k features. $\overline{r_{cf}}$ is the mean feature class correlation and $\overline{r_{ff}}$ is the average feature-feature intercorrelation. In Eq. (85), the numerator indicates the predictive power of the feature set while the denominator shows how much feature redundancy the feature set has. The basic idea of CFS is that a good feature subset should have strong correlation with class labels and are weakly intercorrelated. In order to get the feature-class correlation and feature-feature correlation, CFS uses symmetrical uncertainty (Vetterling et al., 1992) to estimate the degree of associations between two attributes. Since finding the global optimal feature subset is computational prohibitive, CFS adopts a best-search strategy to find a local optimal feature subset. At the very beginning, it computes the utility of each feature by considering both feature-class and feature-feature correlation with the symmetrical uncertainty measure. It then starts with an empty set and expand the set by the feature with the highest utility until it satisfies a stopping criteria.

3. Feature Selection with Structure Features

Existing feature selection methods for generic data are based on a strong assumption that features are independent of each other while completely ignoring the intrinsic structures among features. For example, these feature selection methods may select the same subset of features even though the features are reshuffled (Ye and Liu, 2012). However, in many real applications features also exhibit various kinds of structures, e.g., spatial or temporal smoothness, disjoint groups, overlap groups, trees and graphs (Tibshirani et al., 2005;

Jenatton et al., 2011; Yuan et al., 2011; Huang et al., 2011; Zhou et al., 2012). With the existence of these intrinsic feature structures, feature selection algorithms which incorporate knowledge about the structures of features may help select more relevant features and therefore improve some post-learning tasks such as classification and clustering. One motivating example is from bioinformatics. In the study of arrayCGH such features (the DNA copy numbers along the genome) have some natural spatial order, incorporating such spatial structure can help select more meaningful features and achieve more accurate classification accuracy. Therefore, we discuss some representative feature selection algorithms which explicitly consider feature structures. Specifically, we will focus on group structure, tree structure and graph structure.

Since most existing algorithms in this family are supervised algorithm for binary classification and regression task. Without loss of generality, we focus on linear classification or regression problems such that the class label or regression target $\mathbf{y}$ can be considered as a linear combination of data instances $\mathbf{X}$, the linear combination (feature coefficient) is encoded in a vector $\mathbf{w}$. A popular and successful approach to achieve feature selection with structural features is to minimize a empirical error penalized by a structural regularization term, which can be formulated as:

$$\mathbf{w} = \underset{\mathbf{w}}{\operatorname{argmin}}\ loss(\mathbf{w}; \mathbf{X}, \mathbf{y}) + \alpha\, penalty(\mathbf{w}, \mathcal{G}), \tag{86}$$

where $\mathcal{G}$ denotes the structures among features and α is a trade-off parameter between the loss function and the sparse regularization term. To achieve feature selection, $penalty(\mathbf{w}, \mathcal{G})$ is usually set to be a sparse regularization term. Note that the above formulation is similar to that in Eq. (54), the only difference is that for feature selection with structural features, we explicitly consider the structural information $\mathcal{G}$ among features in the sparse regularization term.

3.1 Feature Selection with Group Feature Structures

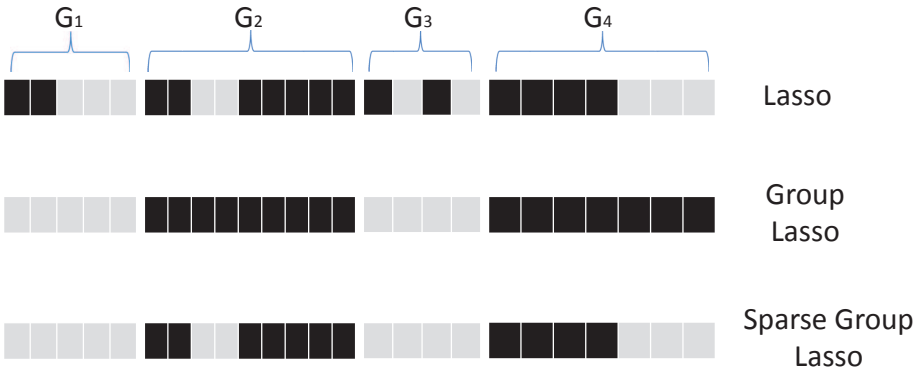


Figure 9: Illustration of Lasso, Group Lasso and Sparse Group Lasso. The whole feature set can be divided into four groups G_1, G_2, G_3 and G_4. Each column denotes a feature, the column with dark color denotes the selected features while the column with light color denotes the unselected features.

In many real-world applications, features exhibit group structures. One of the most common example is that in multifactor analysis-of-variance (ANOVA), each factor is associated with several groups and can be expressed by a set of dummy features (Yuan and Lin, 2006). Some other examples include different frequency bands represented as groups in signal processing (McAuley et al., 2005) and genes with similar functionalities acting as groups in bioinformatics (Ma et al., 2007). Therefore, when performing feature selection, it is more appealing to explicitly take into consideration the group structure among features.

3.1.1 Group Lasso (Supervised) (Yuan and Lin, 2006)

Group Lasso (Yuan and Lin, 2006; Bach, 2008; Jacob et al., 2009; Meier et al., 2008), which derives feature coefficients from some groups to be exact zero, is a solution to this problem. In other words, it selects or does not select a group of features as a whole. The difference between Lasso and Group Lasso is shown by the toy example in Figure (9). Assume that a total of 25 features come from 4 different groups $\mathcal{G} = \{G_1, G_2, G_3, G_4\}$ (each column denotes a feature) and there is no overlap between these groups. Considering the explicit feature structure, we can reorganize the feature coefficients for these 25 features into 4 parts $\mathbf{w} = \{\mathbf{w}_1; \mathbf{w}_2; \mathbf{w}_3; \mathbf{w}_4\}$, where $\mathbf{w}_i$ denotes the feature coefficients for the features from the i-th group G_i. Lasso completely ignores the group structures among features and the selected features (dark column) are across four different groups. In contrast, Group Lasso tends to select or not select the features from different groups as a whole. In the toy example, Group Lasso only selects the second and the fourth group G_2 and G_4, features in other two groups G_1 and G_3 are not selected. Mathematically, Group Lasso first uses a ℓ_2-norm regularization term for feature coefficients $\mathbf{w}_i$ in each group G_i, then it performs a ℓ_1-norm regularization for previous ℓ_2-norm terms. The objective function of group lasso is formulated as follows:

$$\min_{\mathbf{w}} loss(\mathbf{w}; \mathbf{X}, \mathbf{y}) + \alpha \sum_{i=1}^{g} h_i ||\mathbf{w}_{G_i}||_2, \tag{87}$$

where g is the total number of groups, h_i is a weight for the i-th group $\mathbf{w}_{G_i}$ which can be considered as a prior to measure the contribution of the i-th group in the feature selection process. Through optimizing Eq. (89), we obtain the feature coefficients for all the features, these feature coefficients are ranked in a descending order, the higher the value, the more the important the feature is.

3.1.2 Sparse Group Lasso (Supervised) (Friedman et al., 2010; Peng et al., 2010)

Once Group Lasso selects a group, all the features in the group will be selected. However, for some applications that require the diversity of selected features, Group Lasso is not appropriate anymore. In other words, it is desirable to consider the intrinsic feature structures and select features from different selected groups simultaneously. Sparse Group Lasso (Friedman et al., 2010) takes advantage of both Lasso and Group Lasso, and it produces a solution with simultaneous intra-group and inter-group sparsity. The sparse regularization term of Sparse Group Lasso is a combination of the penalty term of Lasso

and Group Lasso. The formulation of Sparse Group Lasso is formulated as follows:

$$\min_{\mathbf{w}} loss(\mathbf{w}; \mathbf{X}, \mathbf{y}) + \alpha||\mathbf{w}||_1 + (1-\alpha)\sum_{i=1}^{g} h_i||\mathbf{w}_{G_i}||_2, \tag{88}$$

where α is parameter between 0 and 1 to balance the contribution of inter-group sparsity and intra-group sparsity for feature selection.

The differences between Lasso, Group Lasso and Sparse Group Lasso are illustrated in Figure (9):

- Lasso does not consider the group structures among features and selects a subset of relevant features among all groups;
- Group Lasso performs group selection and selects or not selects a whole group of features;
- Sparse Group Lasso performs group selection and selects a subset of relevant features in each group.

3.1.3 Overlapping Sparse Group Lasso (Supervised) (Jacob et al., 2009)

Above methods consider the disjoint group structures among features. However, groups may overlap with each other in some applications (Jacob et al., 2009; Jenatton et al., 2011; Zhao et al., 2009). One motivating example is the usage of biologically meaningful gene/protein groups mentioned in (Ye and Liu, 2012). Different groups of genes may overlap, i.e., one protein/gene may belong to multiple groups. Under this scenario, Group Lasso and Sparse Group Lasso are not applicable. A general overlapping Sparse Group Lasso regularization is similar to the regularization term of Sparse Group Lasso:

$$\min_{\mathbf{w}} loss(\mathbf{w}; \mathbf{X}, \mathbf{y}) + \alpha||\mathbf{w}||_1 + (1-\alpha)\sum_{i=1}^{g} h_i||\mathbf{w}_{G_i}||_2. \tag{89}$$

The only difference for overlapping Sparse Group Lasso is that different feature groups G_i can have a overlap, i.e., there exists at least two groups G_i and G_j such that $G_i \bigcap G_j \neq \emptyset$.

3.2 Feature Selection with Tree Feature Structures

In addition to group structures, features can also exhibit other kinds of structures such as tree structures. For example, in image processing such as face images, different pixels (features) can be represented a tree, where the root node indicates the whole face, its child nodes can be the different organs, and each specific pixel is considered as a leaf node. In other words, these pixels enjoy a spatial locality structure. Another motivating example is that genes/proteins may form certain hierarchical tree structures (Liu and Ye, 2010). Recently, Tree-guided Group Lasso is proposed to handle the feature selection for features that can be represented in an index tree (Kim and Xing, 2010; Liu and Ye, 2010; Jenatton et al., 2010).

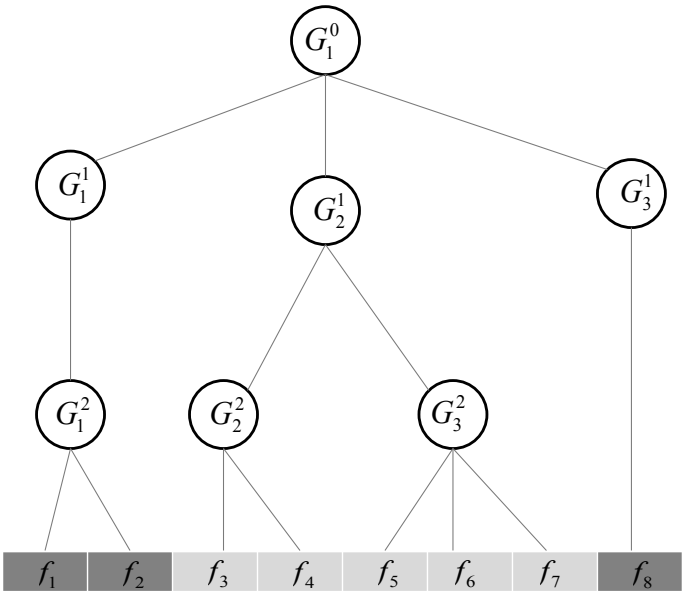


Figure 10: Illustration of tree structure among features. The 8 features form a simple index tree with a depth of 3.

3.2.1 Tree-guided Group Lasso (Supervised) (Liu and Ye, 2010)

In Tree-guided Group Lasso, the structure over the features can be represented as a tree with leaf nodes as features. Each internal node denotes a group of features such that the internal node is considered as a root of a subtree and the group of features are considered as leaf nodes. Each internal node in the tree is associated with a weight that represents the height of its subtree, or how tightly the features in this subtree are correlated.

We follow the definition from (Liu and Ye, 2010) to define Tree-guided Group Lasso, for an index tree $\mathcal{G}$ with a depth of d, $\mathcal{G}_i = \{G_1^i, G_2^i, ..., G_{n_i}^i\}$ denotes the whole set of nodes (features) in the i-th level (the root node is defined as the level 0), and n_i denotes the number of nodes in the i-th level. With these, the nodes in Tree-guided Group Lasso have to satisfy the following two conditions: (1) internal nodes from the same depth level have non-overlapping indices, i.e., $G_j^i \bigcap G_k^i = \emptyset$, $\forall i = 1, 2, ..., d$, $j \neq k$, $i \leq j, k \leq n_i$; and (2) if G_m^{i-1} is the parent node of G_j^i, $G_j^i \subseteq G_m^{i-1}$.

We explain these conditions via a toy example in Figure (10). In the figure, we can observe that 8 features are organized in an indexed tree of depth 3. For the internal nodes in each level, we have:

$$\begin{aligned} &G_1^0 = \{f_1, f_2, f_3, f_4, f_5, f_6, f_7, f_8\} \\ &G_1^1 = \{f_1, f_2\}, G_1^2 = \{f_3, f_4, f_5, f_6, f_7\}, G_1^3 = \{f_8\} \\ &G_1^2 = \{f_1, f_2\}, G_2^2 = \{f_3, f_4\}, G_2^2 = \{f_5, f_6, f_7\}. \end{aligned} \tag{90}$$

It can be observed from the figure that G_1^0 is the root node of the index tree which includes 8 features. In addition, internal nodes from the same level do not overlap while

the parent node and the child node have some overlap such that the features of the child node is a subset of these of the parent node.

With the definition of the index tree, the objective function of Tree-guided Group Lasso is formulated as follows:

$$\min_{\mathbf{w}} loss(\mathbf{w}; \mathbf{X}, \mathbf{y}) + \alpha \sum_{i=0}^{d} \sum_{j=1}^{n_i} h_j^i ||\mathbf{w}_{G_j^i}||_2, \tag{91}$$

where $\alpha \geq 0$ is a regularization parameter and $h_j^i \geq 0$ is pre-defined parameter to measure the contribution of the internal node G_j^i. Since parent node is a superset of its child nodes, thus, if a parent node is not selected (i.e., the corresponding model coefficient is zero), all of its child nodes will not be selected. For example, as illustrated in Figure (10), if the internal node G_2^1 is not selected, both of its child nodes G_2^2 and G_3^2 will not be selected.

3.3 Feature Selection with Graph Feature Structures

In many real-world applications, features may have strong dependencies. For example, in natural language processing, if we take each word as a feature, we have synonyms and antonyms relationships between different words (Fellbaum, 1998). Moreover, many biological studies show that genes tend to work in groups according to their biological functions, and there are strong dependencies between some genes. Since features show some dependencies in these cases, we can model the features by an undirected graph, where nodes represent features and edges among nodes show the pairwise dependencies between features. Recent studies have shown that the learning performance can be improved if we explicitly take into account the dependency information among features (Sandler et al., 2009; Kim and Xing, 2009; Yang et al., 2012).

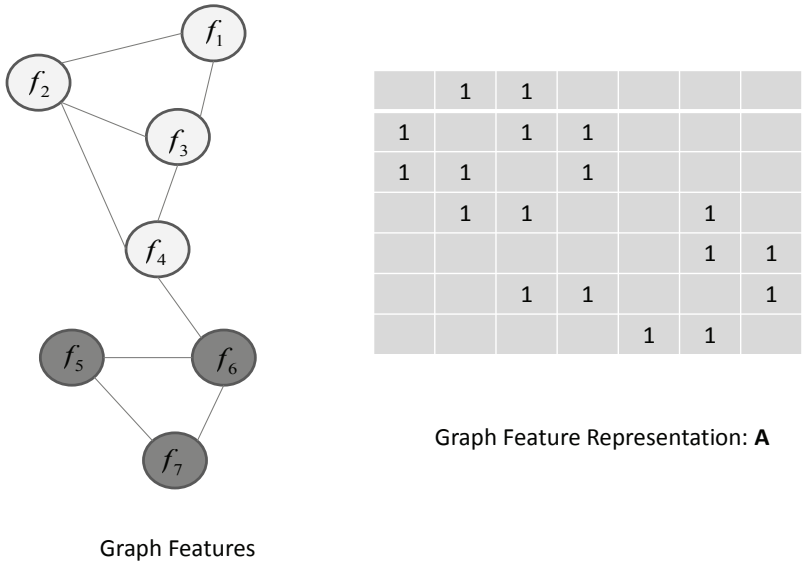


Figure 11: A toy example of graph of 7 features $\{f_1, f_2, ..., f_7\}$ and its dependencies information encoded in an adjacency matrix $\mathbf{A}$.

When there exists some dependencies among features, we can use a undirected graph $\mathcal{G}(N, E)$ to encode these dependencies. Assume that there are n nodes $N = \{N_1, N_2, ..., N_n\}$ and a set of E edges $\{E_1, E_2, ..., E_e\}$ in $\mathcal{G}(N, E)$. Then node N_i corresponds to the i-th feature and the pairwise feature dependencies can be represented by an adjacency matrix $\mathbf{A} \in \mathbb{R}^{N_n \times N_n}$ of $\mathcal{G}(N, E)$. Figure (11) shows an example of the graph with 7 features $\{f_1, f_2, ..., f_7\}$ and its pairwise dependence information encoded in an adjacency matrix $\mathbf{A}$. Note that entries in $\mathbf{A}$ does not necessarily have to be 0 or 1, it can be any numerical numbers that can reflect the correlations between features.

3.3.1 Laplacian Lasso (Supervised) (Ye and Liu, 2012)

Since features exhibit graph structures, when two nodes (features) N_i and N_j are connected by an edge in $\mathcal{G}(N, E)$, the features f_i and f_j are more likely to be selected together, and they should have similar feature coefficients. One way to achieve this target is via Graph Lasso – adding a graph regularizer on the feature graphs on the basis of Lasso. The formulation is as follows:

$$\min_{\mathbf{w}} loss(\mathbf{w}; \mathbf{X}, \mathbf{y}) + \alpha||\mathbf{w}||_1 + (1-\alpha) \sum_{i,j} \mathbf{A}(i,j)(\mathbf{w}_i - \mathbf{w}_j)^2, \tag{92}$$

where the first regularization term $\alpha||\mathbf{w}||_1$ is the same as the regularization term as Lasso while the second term enforces that if a pair of features shows strong dependencies, i.e., large $\mathbf{A}(i,j)$, their feature coefficients should also be close to each other. We can reformulate above loss function into a matrix format:

$$\min_{\mathbf{w}} loss(\mathbf{w}; \mathbf{X}, \mathbf{y}) + \alpha||\mathbf{w}||_1 + (1-\alpha)\mathbf{w}'\mathbf{L}\mathbf{w}, \tag{93}$$

where $\mathbf{L} = \mathbf{D} - \mathbf{A}$ is the Laplacian matrix and $\mathbf{D}$ is a diagonal matrix with $\mathbf{D}(i,i) = \sum_j^{N_i} \mathbf{A}(i,j)$. The Laplacian matrix in Eq. (93) is positive semi-definite and captures the local geometric structure of features. It can be observed that when the Laplacian matrix is the identity matrix $\mathbf{I}$, $\mathbf{w}'\mathbf{L}\mathbf{w} = ||\mathbf{w}||_2^2$, the penalty term in Eq. (93) reduces to the elastic net penalty (Zou and Hastie, 2005). Since the graph regularization term $\mathbf{w}'\mathbf{L}\mathbf{w}$ is convex and differentiable, existing efficient algorithms like LARS (Efron et al., 2004) and proximal gradient descent methods (Liu and Ye, 2009) can be directly applied.

3.3.2 GFLasso (Supervised) (Kim and Xing, 2009)

In Eq. (93), graph feature structures are represented by an unsigned graph, and it encourages features connected together with similar feature coefficients. However, in many cases, features can also be negatively correlated. In this case, the feature graph $\mathcal{G}(N, E)$ is represented by a signed graph, with both positive and negative edges. Recently, GFLasso is proposed to explicitly consider both the positive and negative feature correlations, the objective function of GFLasso is formulated as follows:

$$\min_{\mathbf{w}} loss(\mathbf{w}; \mathbf{X}, \mathbf{y}) + \alpha||\mathbf{w}||_1 + (1-\alpha) \sum_{i,j} \mathbf{A}(i,j)(\mathbf{w}_i - \text{sign}(r_{i,j})\mathbf{w}_j)^2, \tag{94}$$

where $r_{i,j}$ indicates the correlation between two features f_i and f_j. When two features are positively correlation, we have $\mathbf{A}(i,j) = 1$ and $r_{i,j} > 0$, and the penalty term forces

the feature coefficients $\mathbf{w}_i$ and $\mathbf{w}_j$ to be similar; on the other hand, if two features are negatively correlated, we have $\mathbf{A}(i,j) = 1$ and $r_{i,j} < 0$, and the penalty term makes the feature coefficients $\mathbf{w}_i$ and $\mathbf{w}_j$ to be dissimilar. An major limitation of GFLasso is that it uses pairwise sample correlations to measure feature dependencies, which may lead to additional estimation bias. The feature dependencies cannot be correctly estimated when the sample size is small.

3.3.3 GOSCAR (Supervised) (Yang et al., 2012)

To address the limitations of GFLasso, (Yang et al., 2012) proposed a GOSCAR algorithm by putting a ℓ_∞-norm regularization to enforce the pairwise feature coefficients to be equal if they are connected over the feature graph $\mathcal{G}(N, E)$. The formulation of GOSCAR is defined as:

$$\min_{\mathbf{w}} loss(\mathbf{w}; \mathbf{X}, \mathbf{y}) + \alpha||\mathbf{w}||_1 + (1-\alpha) \sum_{i,j} \mathbf{A}(i,j)\max(|\mathbf{w}_i|, |\mathbf{w}_j|). \tag{95}$$

In the above formulation, the ℓ_1-norm regularization is used for feature selection while the pairwise ℓ_∞-norm term penalizes large coefficients. The pairwise ℓ_∞-norm term can be decomposed as:

$$\begin{aligned} \max(|\mathbf{w}_i|, |\mathbf{w}_j|) &= \frac{1}{2}(|\mathbf{w}_i + \mathbf{w}_j| + |\mathbf{w}_i - \mathbf{w}_j|) \\ &= |\mathbf{u}'\mathbf{w}| + |\mathbf{v}'\mathbf{w}|, \end{aligned} \tag{96}$$

where $\mathbf{u}$ and $\mathbf{v}$ are sparse vectors with only two non-zero entries such that $\mathbf{u}_i = \mathbf{u}_j = \frac{1}{2}$, $\mathbf{v}_i = -\mathbf{v}_j = \frac{1}{2}$. The formulation of the GOSCAR is more general than the OSCAR algorithm which is proposed in (Bondell and Reich, 2008). OSCAR algorithm assumes that all features form a complete graph whose entries in the adjacency matrix $\mathbf{A}$ are all 1. In contrast, GOSCAR can deal with an arbitrary undirected graph as long as the adjacency matrix $\mathbf{A}$ is symmetric.

Different signs of feature coefficients can introduce additional penalty in the objective function in Eq. (93). Even though GOSCAR is designed to mitigate this side effect, it may still over penalize the feature coefficient $\mathbf{w}_i$ or $\mathbf{w}_j$ due to the property of the max operator. Therefore, a new formulation with non-convex grouping penalty is also proposed in (Yang et al., 2012):

$$\min_{\mathbf{w}} loss(\mathbf{w}; \mathbf{X}, \mathbf{y}) + \alpha||\mathbf{w}||_1 + (1-\alpha) \sum_{i,j} \mathbf{A}(i,j)||\mathbf{w}_i| - |\mathbf{w}_j||_1, \tag{97}$$

where the grouping penalty term controls only magnitudes of differences of coefficients ignoring their signs over the feature graph $\mathcal{G}(N, E)$.

For feature selection problem with graph structured features, a subset of highly connected features in the graph is more likely to be selected or not selected as a whole. For example, in the toy example in Figure (11), features $\{f_5, f_6, f_7\}$ are selected while features $\{f_1, f_2, f_3, f_4\}$ are not selected.

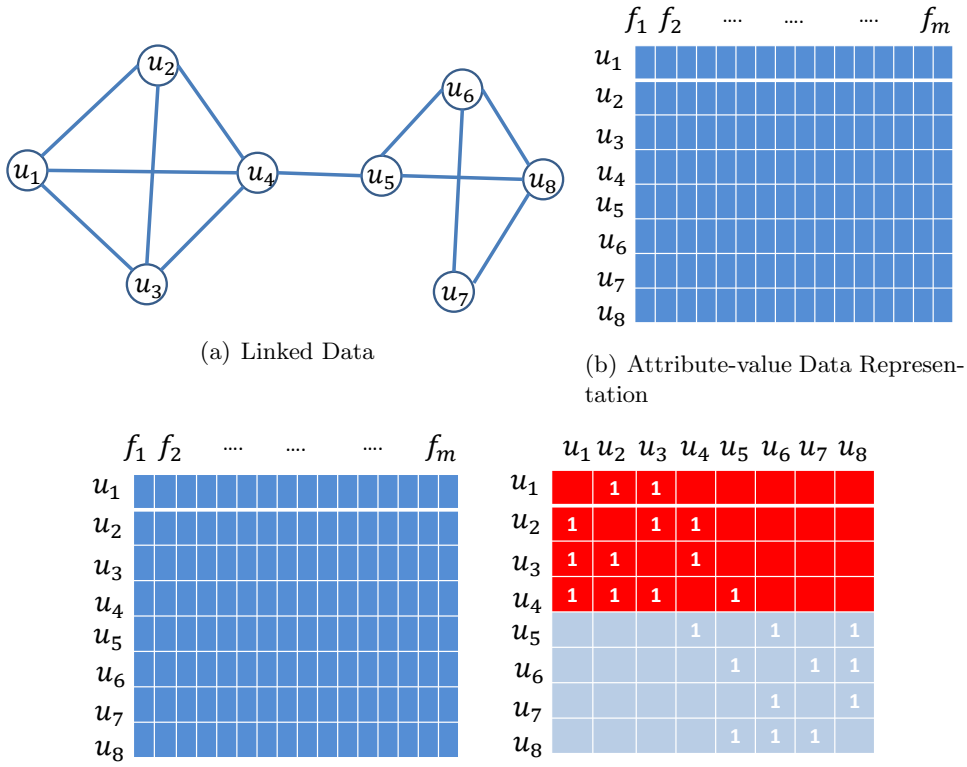
4. Feature Selection with Heterogeneous Data

Traditional feature selection algorithms can only work with generic data from a single data source which is based on the data independent and identically distributed (i.i.d.) assumption. However, heterogeneous data are becoming more prevalent in the era of big data. For example, in the medical domain, high dimensional gene features are often considered associated with different types clinical features. Since data of each source can be noisy, partial, or redundant, how to select relevant sources and how to use them together for effective feature selection is a challenging problem. Another example is in social media platforms, instances with high-dimensional features are often linked together, how to integrate link information to guide feature selection is another difficult problem. In this section, we review current feature selection algorithms for heterogeneous data from three aspects: (1) feature selection for linked data; (2) feature selection for multi-source data; and (3) feature selection for multi-view data. Note that multi-source feature selection is similar to multi-view feature selection, but they are different in two ways. First, multi-source feature selection aims select features from the original feature space by integrating multiple sources while multi-view feature selection select features from different feature spaces for all views simultaneously. Second, multi-source feature selection normally ignores the correlations among sources while multi-view feature selection exploits the relations among features from different views.

4.1 Feature Selection Algorithms with Linked Data

Linked data has become ubiquitous in real-world applications such as Twitter[4] (tweets linked through hyperlinks), social networks in Facebook[5] (people connected by friendships) and biological networks (protein interaction networks). Since linked data are related to each other by different types of links, they are distinct from traditional attribute value data (or "flat" data). Figure (12) illustrates a toy example of linked data and its two representations. Figure (12(a)) shows 8 linked instances (u_1 to u_8) while Figure (12(b)) is a conventional representation of attribute-value data such that each row corresponds to one instance and each column corresponds to one feature[6]. As mentioned above, in addition to feature information, linked data provides an extra source of information in the form of links, which can be represented by an adjacency matrix, illustrated in Figure (12(c)). Many linked data related learning tasks are proposed such as collective classification (Macskassy and Provost, 2007; Sen et al., 2008), relational learning (Long et al., 2006, 2007), and active learning (Bilgic et al., 2010; Hu et al., 2013), but the task of feature selection is not well studied due to some of its unique challenges: (1) how to exploit relations among data instances; (2) how to take advantage of these relations for feature selection; and (3) linked data are often unlabeled, how to evaluate the relevance of features without the guide of label information. Until recently, feature selection for linked data receives some attention. Next, we introduce some representative algorithms which leverage link information for feature selection.

4. https://twitter.com/
5. https://www.facebook.com/
6. The data can either be labeled or unlabeled. In the example in Figure (12), the data is unlabeled.



(a) Linked Data

(b) Attribute-value Data Representation

(c) Linked Data Representation

Figure 12: A toy example of Linked Data.

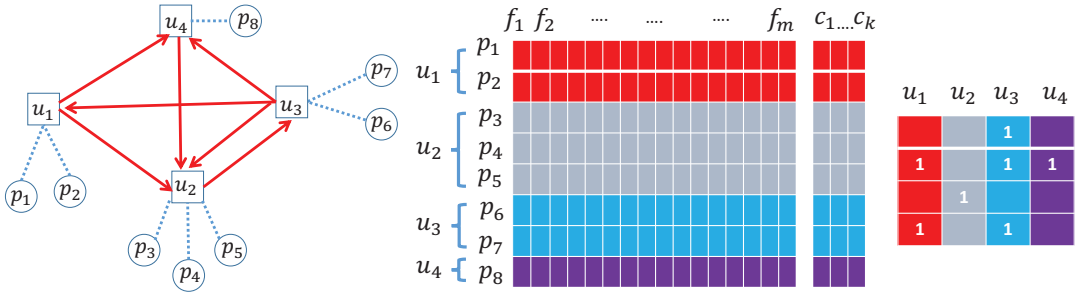
4.1.1 FEATURE SELECTION ON NETWORKS (SUPERVISED) (GU AND HAN, 2011)

In (Gu and Han, 2011), authors propose a supervised feature selection algorithm (FSNet) based on Laplacian Regularized Least Squares (LapRLS). In detail, they propose to use linear classifier to capture the relationship between content information and class labels, and incorporate link information by graph regularization. Suppose that $\mathbf{X} \in \mathbb{R}^{n \times d}$ denotes the content matrix with n instances and each instance is associated with a d-dimensional feature vector; $\mathbf{Y} \in \mathbb{R}^{n \times c}$ denotes the class labels matrix such that $\mathbf{Y}(i,k) = 1$ if the class label for the i-th instance is k, $\mathbf{Y}(i,k) = 0$ otherwise; $\mathbf{A}$ denotes the adjacency matrix for all n linked instances. The network can either be directed and undirected. With these notations, LapRLS first attempts to learn a linear classifier $\mathbf{W} \in \mathbb{R}^{d \times c}$ to map $\mathbf{X}$ to $\mathbf{Y}$:

$$\min_{\mathbf{W}} \|\mathbf{X}\mathbf{W} - \mathbf{Y}\|_F^2 + \alpha\|\mathbf{W}\|_{2,1} + \beta\|\mathbf{W}\|_F^2. \tag{98}$$

The regularization term $\|\mathbf{W}\|_{2,1}$ is included to achieve feature selection. It makes $\mathbf{W}$ be sparse in rows, which makes the feature sparse across all c class labels. The other regularization term $\|\mathbf{W}\|_F^2$ presents the overfitting of the model and makes the model more robust in the same time.

Till now, FSNet has modeled the content information for feature selection. However, link information is not considered yet. To capture the correlation between link information and



(a) Social Media Data Example

(b) Attribute-value Data Representation

Figure 13: A toy example of Linked Data in Social Media.

content information to select more relevant features, FSNet uses the graph regularization and the basic assumption is that if two instances are linked, their class labels are likely to be similar. Taking into consideration the graph regularization, the objective function of FSNet can be mathematically formulated as:

$$\min_{\mathbf{W}} \|\mathbf{XW} - \mathbf{Y}\|_F^2 + \alpha\|\mathbf{W}\|_{2,1} + \beta\|\mathbf{W}\|_F^2 + \gamma tr(\mathbf{W}'\mathbf{X}'\mathbf{LXW}), \tag{99}$$

where $tr(\mathbf{W}'\mathbf{X}'\mathbf{LXW})$ is the graph regularization, γ controls the contribution of content information and link information. For undirected network, $\mathbf{L} = \mathbf{D} - \mathbf{A}$ is the laplacian matrix where $\mathbf{D}$ is a diagonal matrix with its diagonal entry as $\mathbf{D}(i,i) = \sum_{j=1}^{n} \mathbf{A}(i,j)$. For directed network, the laplacian matrix $\mathbf{L}$ can be obtained either by transforming the network to be undirected $\mathbf{A} = \max(\mathbf{A}, \mathbf{A}')$ or by applying the random walk method in (Zhou et al., 2005a). The objective function in Eq. (99) can be solved by proximal gradient descent methods (Nesterov, 2004). Afterwards, NetFS calculates the ℓ_2-norm value for each feature coefficient vector in $\mathbf{W}$ and return the top ranked ones.

4.1.2 Feature Selection for Social Media Data (Supervised) (Tang and Liu, 2012a, 2014a)

In (Tang and Liu, 2012a, 2014a), the authors investigate feature selection problem on social media data by evaluating the effects of user-user and user-post relationships as illustrated in Figure (13). The target is to perform feature selection for high dimensional posts, and take into account the user-user and user-post relationships manifested in the linked data.

In the proposed supervised feature selection framework (LinkedFS), different social relationships are extracted to enhance the feature selection performance. As illustrated in Figure (14), LinkedFS extracts four basic types of relations as hypotheses: (a) CoPost - a user u_2 can have multiple posts (p_3, p_4 and p_5) and these posts are more similar than those randomly selected; (b) CoFollowing - two users u_1 and u_3 follow a user u_4, its counterpart in citation analysis is bibliographic coupling (Morris, 2005), and their posts are more likely to be of similar topics; (c) CoFollowed - two users u_2 and u_4 are followed by a third user u_1, similar to co-citation relation (Morris, 2005) in citation analysis, and their posts are more likely to be similar to each other; and (d) Following - a user u_1 follows another user u_2, and their posts (e.g., $\{p_1, p_2\}$ and $\{p_3, p_4, p_5\}$ are more likely similar in

terms of topics. These four hypotheses are supported by social correlation theories such as homophily (McPherson et al., 2001) and social influence (Marsden and Friedkin, 1993) in explaining the existence of similarity as what these relations suggest. For example, homophily indicates that people with similar interests are more likely to be linked.

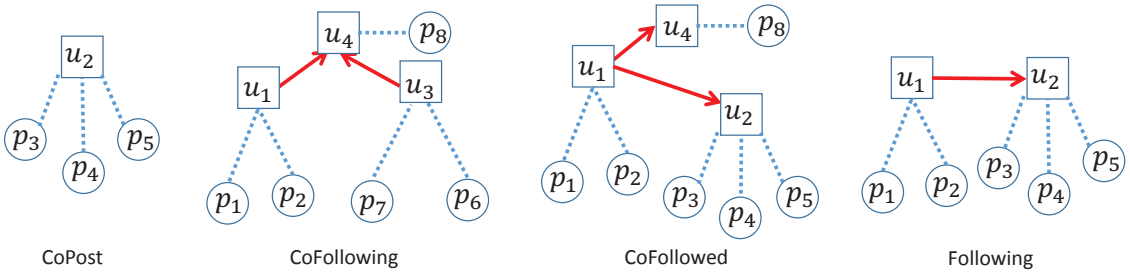


Figure 14: Different Types of Relations Extracted from Social Correlations among Posts

Next, we use the CoPost relation as an example to illustrate how these relations can be integrated into feature selection. A comprehensive report of other three relations can be referred to the original paper (Tang and Liu, 2012a, 2014a). Let $\mathbf{p} = \{p_1, p_2, ..., p_N\}$ be the post set and $\mathbf{X} \in \mathbb{R}^{N \times d}$ be the matrix representation of these posts where N is the number of posts and each post is associated with a d dimensional feature vector; $\mathbf{Y} \in \mathbb{R}^{n \times c}$ denotes the class labels matrix where $\mathbf{Y}(i,k) = 1$ if the i-th post belongs to class c_j, otherwise zero; $\mathbf{u} = \{u_1, u_2, ..., u_n\}$ denotes the set of n users and their link information is encoded in an adjacency matrix $\mathbf{A}$; $\mathbf{P} \in \mathbb{R}^{n \times N}$ denotes the user-post relationships such that $\mathbf{P}(i,j) = 1$ if u_i posts p_j, otherwise 0. To integrate the CoPost relations among users into the feature selection framework, the authors propose to add a regularization term enforces the hypothesis that the class labels (i.e., topics) of posts by the same user are similar. Hence, feature selection with the CoPost hypothesis can be formulated as the following optimization problem:

$$\min_{\mathbf{W}} \|\mathbf{XW} - \mathbf{Y}\|_F^2 + \alpha \|\mathbf{W}\|_{2,1} + \beta \sum_{u \in \mathbf{u}} \sum_{\{p_i, p_j\} \in \mathbf{p}_u} \|\mathbf{X}(i,:)\mathbf{W} - \mathbf{X}(j,:)\mathbf{W}\|_2^2, \tag{100}$$

where $\mathbf{p}_u$ denotes the whole set of posts by user u. The parameter α controls the sparseness of $\mathbf{W}$ in rows across all class labels and β controls the contribution of the CoPost relations. The CoPost regularization term indicates that if a user posts multiple posts, the class labels of these posts should be similar. After optimizing the objective function in Eq. (100), we can obtain the sparse matrix $\mathbf{W}$ and get the ranking of all d features thereby.

4.1.3 Unsupervised Feature Selection for Linked Data (Unsupervised) (Tang and Liu, 2012b, 2014b)

Linked Unsupervised Feature Selection (LUFS) is an unsupervised feature selection framework for linked data and in essence, LUFS investigates how to take advantage of link information for unsupervised feature selection. Generally speaking, feature selection aims to select a subset of relevant features with some constraints, while in supervised feature selection, the class labels play the role of constraints such that distances of instances with the

same class labels should be closer than these of instances from different classes. However, in unsupervised feature selection, without class labels to assess feature relevance, some alternative criteria has to be exploited. The problem is formulated as follows: given n linked instances $\{x_1, x_2, ..., x_n\}$, their feature information and link information can be represented a feature matrix $\mathbf{X} \in \mathbb{R}^{n\times d}$ and an adjacency matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$, respectively, where d denotes the feature dimension. The task is to select a subset of relevant features from all d features by utilizing both feature information $\mathbf{X}$ and link information $\mathbf{A}$. LUFS introduces the concept of pseudo-class labels to guide the unsupervised feature selection. Particularly, LUFS assumes the pseudo class labels come from c classes, and uses $\mathbf{Y} \in \mathbb{R}^{n\times c}$ to denote the label matrix such each row of $\mathbf{Y}$ has only one nonzero entry. Like most feature selection algorithms, LUFS assumes a linear mapping matrix $\mathbf{W} \in \mathbb{R}^{d\times c}$ exists between feature matrix $\mathbf{X}$ and pseudo-class label matrix $\mathbf{Y}$. With these notations, LUFS seeks the pseudo-class labels by extracting constraints from link information and attribute-value information through social dimension approach and spectral analysis, respectively.

First, to consider the constraints from link information, LUFS employs social dimension approach to exploit the hidden factors that incur the interdependency among instances. Particularly, it uses modularity maximization (Newman and Girvan, 2004) to extract hidden factor matrix $\mathbf{H}$. Since the extracted hidden factor matrix $\mathbf{H}$ indicates some affiliations among linked instances, according to the Linear Discriminative Analysis, within, between and total hidden factor scatter matrix $\mathbf{S}_w$, $\mathbf{S}_b$ and $\mathbf{S}_t$ are defined as $\mathbf{S}_w = \mathbf{Y'Y} - \mathbf{Y'FF'Y}$, $\mathbf{S}_b = \mathbf{Y'FF'Y}$, $\mathbf{S}_t = \mathbf{Y'Y}$, where $\mathbf{F} = \mathbf{H}(\mathbf{H'H})^{-\frac{1}{2}}$ is the weighted hidden factor matrix. Considering the fact that instances with similar hidden factors are similar and instances with different hidden factors are dissimilar, the constraint from link information can be incorporated by solving the following maximization problem:

$$\max_{\mathbf{W}} tr((\mathbf{S}_t)^{-1}\mathbf{S}_b). \tag{101}$$

Second, to take advantage of information from attribute-value part, LUFS obtains the constraints by the spectral analysis (Von Luxburg, 2007):

$$\min tr(\mathbf{Y'LY}), \tag{102}$$

where $\mathbf{L} = \mathbf{D} - \mathbf{S}$ is the laplacian matrix and $\mathbf{D}$ is the diagonal matrix with its diagonal entry as $\mathbf{D}(i,i) = \sum_{j=1}^{n} \mathbf{S}(i,j)$. $\mathbf{S}$ denotes the affinity matrix from $\mathbf{X}$ and LUFS adopts the RBF kernel to get the affinity matrix. Incorporating the constraints from Eq. (101) and Eq. (102), the objective function of LUFS is formulated as follows:

$$\min_{W} tr(\mathbf{Y'LY}) - \alpha tr((\mathbf{S}_t)^{-1}\mathbf{S}_b), \tag{103}$$

where α is a regularization parameter to balance the contribution from these two constraints. To achieve feature selection, LUFS further adds a $\ell_{2,1}$-norm regularization term on $\mathbf{W}$, and with spectral relaxation of the pseudo-class label matrix, the objective function in Eq. (103) can be eventually represented as:

$$\begin{aligned} \min_{\mathbf{W}} & tr(\mathbf{W'}(\mathbf{X'LX} + \alpha\mathbf{X'}(\mathbf{I}_n - \mathbf{FF'}))\mathbf{W}) + \beta\|\mathbf{W}\|_{2,1} \\ & \text{s.t.} \quad \mathbf{W'}(\mathbf{X'X} + \lambda\mathbf{I}_d)\mathbf{W} = \mathbf{I}_c, \end{aligned} \tag{104}$$

where β is a parameter to control the sparseness of $\mathbf{W}$ in rows and $\lambda\mathbf{I}_d$ is included to make $\mathbf{X}'\mathbf{X} + \lambda\mathbf{I}_d$ invertible. Similar to previous mentioned feature selection algorithms, the ranking of features can be obtained from the sparse matrix $\mathbf{W}$.

4.2 Feature Selection Algorithms with Multi-Source Data

Over the past few decades, many feature selection algorithms are proposed and they have proven to be effective in handling high dimensional data. However, most of them are designed for single source of data. In many data mining and machine learning tasks, we may have multiple data sources for the same set of data instances. For example, recent advancement in bioinformatics reveal the existence of some non-coding RNA species in addition widely used messenger RNA, these non-coding RNA species functions across a variety of biological process. The availability of multiple data sources makes it possible to solve some problems unsolvable using a single source since the multi-faceted representations of data can help depict some intrinsic patterns hidden in a single source of data. The task of multi-source feature selection is formulated as follows: given m sources of data depicting the same set of n instances, and their matrix representations $\mathbf{X}_1 \in \mathbb{R}^{n\times d_1}, \mathbf{X}_2 \in \mathbb{R}^{n\times d_2}, ..., \mathbf{X}_m \in \mathbb{R}^{n\times d_m}$ (where $d_1, ..., d_m$ denote the feature dimensions), select a subset of relevant features from a target source (e.g., $\mathbf{X}_i$) by taking advantage of all information in all m sources.

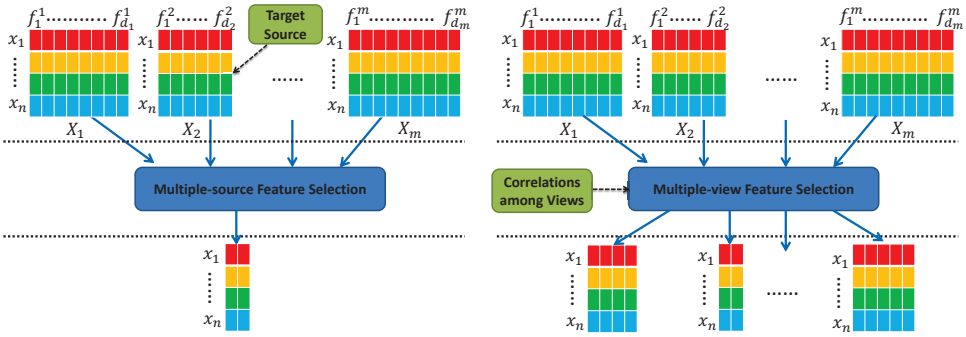
4.2.1 Multi-Source Feature Selection via Geometry-Dependent Covariance Analysis (Unsupervised) (Zhao and Liu, 2008)

To integrate information from multiple sources, authors in (Zhao and Liu, 2008) propose an intuitive way to learn a global geometric pattern from all sources that reflects the intrinsic relationships among instances (Lanckriet et al., 2004). They introduce a concept of geometry-dependent covariance that enables the usage of the global geometric pattern in covariance analysis for feature selection. Given multiple local geometric patterns in an affinity matrix $\mathbf{S}_i$ where i denotes the i-th data source, a global pattern can be obtained by linearly combining all affinity matrices as $\mathbf{S} = \sum_{i=1}^{m} \alpha_i \mathbf{S}_i$, where α_i controls the contribution of the i-th source. With the global geometric pattern obtained from multiple data sources, one can build a geometry-dependent sample covariance matrix for the target source $\mathbf{X}_i$ as follows:

$$\mathbf{C} = \frac{1}{n-1}\mathbf{\Pi}\mathbf{X}_i'(\mathbf{S} - \frac{\mathbf{S}\mathbf{1}\mathbf{1}'\mathbf{S}}{\mathbf{1}'\mathbf{S}\mathbf{1}})\mathbf{X}_i\mathbf{\Pi}, \tag{105}$$

where $\mathbf{\Pi}$ is a diagonal matrix with $\mathbf{\Pi}(j,j) = \|\mathbf{D}^{\frac{1}{2}}\mathbf{X}_i(:,j)\|^{-1}$, and $\mathbf{D}$ is also a diagonal matrix from $\mathbf{S}$ with $\mathbf{D}(k,k) = \sum_{j=1}^{n} \mathbf{S}(k,j)$.

After getting a geometry-dependent sample covariance matrix, a subsequent question is how to use it effectively for feature selection. Basically, two methods are proposed. The first method, GPCOVvar sorts the diagonal of the covariance matrix and selects the features that have the biggest variances. Selecting features based on this method is equivalent to choose features that are consistent with the global geometry pattern. The basic idea of the first method is similar to Laplacian score (He et al., 2005) and SPEC (Zhao and Liu, 2007) mentioned above, hence one limitation is that feature redundancy is not considered since it measures features individually. While on the other hand, the second method, GPCOVspca, applies sparse principle component analysis (SPCA) (d'Aspremont et al., 2007)



(a) Multi-source Feature Selection (b) Multi-view Feature Selection

Figure 15: Differences between multi-source and multi-view feature selection.

to select features that is able to retain the total variance maximally, and hence considers the interactions among features and is able to select features with less redundancy.

4.3 Feature Selection Algorithms with Multi-View Data

Multi-view sources represent different facets of data instances via different feature spaces. These feature spaces are naturally dependent and also high dimensional, which suggests that feature selection is necessary to prepare these sources for effective data mining tasks such as multi-view clustering. A task of multi-view feature selection thus arises, which aims to select features from different feature spaces simultaneously by using their relations. For example, selecting pixels, tags, and terms about images in Flickr[7] simultaneously. Since multi-view feature selection is designed to select features across multiple views by using their relations, they are naturally different from multi-source feature selection. The difference between multi-source feature selection and multi-view feature selection is illustrated in Figure (15). For supervised multi-view feature selection, the most common approach is Sparse Group Lasso (Friedman et al., 2010; Peng et al., 2010). In this subsection, we review some representative algorithms for unsupervised multi-view feature selection.

4.3.1 Adaptive Multi-view Feature Selection (Unsupervised) (Feng et al., 2013)

Adaptive unsupervised multi-view feature selection (AUMFS) takes advantages of data cluster structure, data similarity and correlations among views simultaneously for feature selection. More specifically, let $\mathbf{X}_1 \in \mathbb{R}^{n\times d_1}, \mathbf{X}_2 \in \mathbb{R}^{n\times d_2}, ..., \mathbf{X}_1 \in \mathbb{R}^{n\times d_m}$ denote the description of n instances from m different views, $\mathbf{X} = [\mathbf{X}_1, \mathbf{X}_2, ..., \mathbf{X}_m] \in \mathbb{R}^d$ denotes the concatenated data, where $d = d_1 + d_2 + ... + d_m$. AUMFS first builds a feature selection model by using a $\ell_{2,1}$-norm regularized least square loss function:

$$\min_{\mathbf{W},\mathbf{F}} \|\mathbf{XW} - \mathbf{F}\|_{2,1} + \alpha\|\mathbf{W}\|_{2,1}, \tag{106}$$

7. https://www.flickr.com/

where $\mathbf{F} \in \mathbb{R}^{n \times c}$ is the pseudo class label matrix. The $\ell_{2,1}$-norm loss function is imposed since it is robust to outliers in the data instances and $\ell_{2,1}$-norm regularization selects features across all c pseudo class labels with joint sparsity. Then AUMFS uses spectral clustering on an affinity matrix from different views to learning the shared pseudo class labels. For the data matrix $\mathbf{X}_i$ in each view, they first build an affinity matrix $\mathbf{S}_i$ based on the data similarity on that view and get the the corresponding laplacian matrix $\mathbf{L}_i$. Then it aims to learn the pseudo class label matrix by considering the spectral clustering from all views:

$$\begin{aligned} &\min_{\mathbf{F},\lambda} \sum_{i=1}^{m} \lambda_i tr(\mathbf{F}'\mathbf{L}_i\mathbf{F}) = \min tr(\mathbf{F}' \sum_{i=1}^{m} \lambda_i \mathbf{L}_i \mathbf{F}) \\ &\text{s.t.} \quad \mathbf{F}'\mathbf{F} = \mathbf{I}_c, \mathbf{F} \geq 0, \sum_{i=1}^{m} \lambda_i = 1, \lambda_i \geq 0, \end{aligned} \tag{107}$$

where the contribution of each feature for the joint spectral clustering is balanced by a nonnegative weight λ_i and the summation of all λ_i equals 1. By combing the objective function in Eq. (106) and Eq. (107) together, the final objective function of AUMFS which jointly performs pseudo class label learning and feature selection is as follows:

$$\begin{aligned} &\min tr(\mathbf{F}' \sum_{i=1}^{m} \lambda_i \mathbf{L}_i \mathbf{F}) + \beta(\|\mathbf{X}\mathbf{W} - \mathbf{F}\|_{2,1} + \alpha\|\mathbf{W}\|_{2,1}) \\ &\text{s.t.} \quad \mathbf{F}'\mathbf{F} = \mathbf{I}_c, \mathbf{F} \geq 0, \sum_{i=1}^{m} \lambda_i = 1, \lambda_i \geq 0. \end{aligned} \tag{108}$$

Through optimizing the objective function in Eq. (109) and obtaining the feature coefficient matrix $\mathbf{W}$. AUMFS takes a commonly accepted way to rank all features according to the value of $\|\mathbf{W}(i,:)\|_2^2$ in a descending order and return the top ranked ones.

4.3.2 Unsupervised Feature Selection for Multi-View Data (Unsupervised) (Tang et al., 2013)

AUMFS (Feng et al., 2013) learns one feature weight matrix for all features from different views to approximate the pseudo class labels. In (Tang et al., 2013), the authors propose a novel unsupervised feature selection method called Multi-view Feature Selection (MVFS). Similar to AUMFS, MVFS also uses spectral clustering with the affinity matrix from different views to learn the pseudo class labels. But it differs from AUMFS that it learns one feature weight matrix for each view to fit the pesudo class labels by the joint least squared loss and $\ell_{2,1}$-norm regularization. The optimization problem of MVFS can be formulated as follows:

$$\begin{aligned} &\min tr(\mathbf{F}' \sum_{i=1}^{m} \lambda_i \mathbf{L}_i \mathbf{F}) + \sum_{i=1}^{m} \beta(\|\mathbf{X}_i\mathbf{W}_i - \mathbf{F}\|_{2,1} + \alpha\|\mathbf{W}_i\|_{2,1}) \\ &\text{s.t.} \quad \mathbf{F}'\mathbf{F} = \mathbf{I}_c, \mathbf{F} \geq 0, \sum_{i=1}^{m} \lambda_i = 1, \lambda_i \geq 0. \end{aligned} \tag{109}$$

Similar to AUMFS, the orthogonal and nonnegative constraints on the pseudo class label matrix $\mathbf{F}$ is imposed to guarantee there is one and only one positive entry in each row and

other entries are all zero. The parameter λ_i is employed to control the contribution of each view and $\sum_{i=1}^{m} \lambda_i = 1$.

4.3.3 Multi-view Clustering and Feature Learning via Structured Sparsity (Wang et al., 2013a)

In some cases, it is possible that features from a certain view contains more discriminative information than features from other views for different clusters. According to (Wang et al., 2013a), one example is that in image processing, the color features are more useful than other types of features in identifying stop signs. To address this issue in multi-view feature selection, a novel feature selection algorithm is proposed in (Wang et al., 2013a) with a joint group ℓ_1-norm and $\ell_{2,1}$-norm regularization.

For the feature weight matrix $\mathbf{W}_1, ..., \mathbf{W}_m$ from m different views, the group ℓ_1-norm is defined as $\|\mathbf{W}\|_{G_1} = \sum_{j=1}^{c} \sum_{i=1}^{m} \|\mathbf{W}_i(:, j)\|$. Crucially, the group ℓ_1-norm regularization term is able to capture the global relations among different views and is able to achieve view-wise sparsity such that only a few views are selected. In addition to group ℓ_1-norm, a $\ell_{2,1}$-norm regularizer on $\mathbf{W}$ is also included to achieve feature sparsity among selected views. It can be observed that the basic idea is very similar to sparse group lasso (Friedman et al., 2010; Peng et al., 2010) which also requires intra-group sparsity and inter-group sparsity. Hence, the objective function of the proposed method is formulated as follows:

$$\begin{aligned} &\min_{\mathbf{W},\mathbf{F}} \|\mathbf{X}\mathbf{W} - \mathbf{F}\|_F^2 + \alpha\|\mathbf{W}\|_{2,1} + \beta\|\mathbf{W}\|_{G_1} \\ &\text{s.t.} \quad \mathbf{F}'\mathbf{F} = \mathbf{I}_c, \mathbf{F} \geq 0, \end{aligned} \tag{110}$$

where α and β are two parameters to control the inter-view sparsity and intra-view sparsity. Through the proposed two regularizer, many features in the discriminative views and a small number of features in the non-discriminative views will output as the final set of features to discriminate cluster structures.

5. Feature Selection with Streaming Data

Methods introduced in the previous sections assume that all data instances and features are known in advance. However, it is not the case in many real-world applications that we are more likely facing with dynamic data streams and feature streams. In the worst cases, the size of data or the features are unknown or even infinite, thus it is not practical to wait until all data instances or features are available to perform feature selection. For streaming data, one motivating example is that in online spam email detection problem, new emails are constantly arriving, it is not easy to employ batch-mode feature selection methods to select relevant feature in a time manner. On a orthogonal setting, feature selection for streaming features also have its practical significances. For example, Twitter produces more than 320 millions of tweets everyday and a large amount of slang words (features) are continuously being generated. These slang words promptly grab users' attention and become popular in a short time. Therefore, it is more preferable to perform streaming feature selection to rapidly adapt to the changes. Recently, there exists some work trying to combine these two dual problems together, the problem is referred as feature selection on Trapezoidal data streams (Zhang et al., 2015). In the following two sections, we will review

some representative algorithms for these two orthogonal problems, i.e., feature selection for streaming data and feature selection for streaming features.

5.1 Feature Selection Algorithms with Feature Streams

For the feature selection problem with streaming features, the number of instances is considered to be constant while candidate features arrive one at a time, the task is to timely select a subset of relevant features from all features seen so far. Instead of searching for the whole feature space which is costly, streaming feature selection (SFS) processes a new feature upon its arrival. A general framework of streaming feature selection is presented in Figure (16). At each time step, a typical SFS algorithm first determines whether to accept the most recently arrived feature; if the feature is added to the selected feature set, it then determines whether to discard some existing features from the selected feature set. The process repeats until no new features show up anymore. Different algorithms have different implementations in the first step that checks newly arrived features. The second step which checks existing features is an optional step for some algorithms.

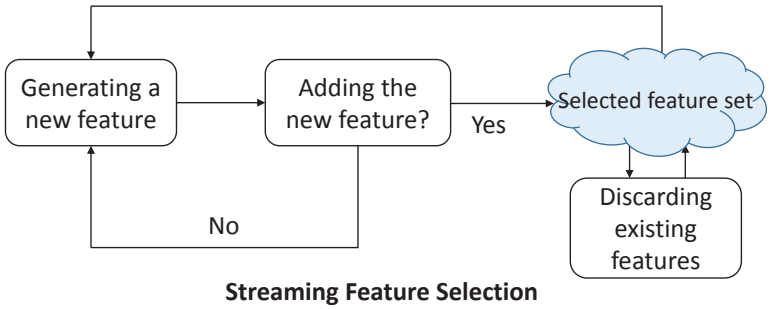


Figure 16: A framework of streaming feature selection. It consists of two phases: testing newly arrived feature and testing existing features.

5.1.1 Grafting Algorithm (Supervised) (Perkins and Theiler, 2003)

The first attempt to perform streaming feature selection is credited to (Perkins and Theiler, 2003). They proposed a streaming feature selection framework based on stagewise gradient descent regularized risk framework (Perkins et al., 2003). Grafting is a general technique that can deal with a variety of models that are parameterized by a feature weight vector $\mathbf{w}$ subject to ℓ_1-norm regularization, such as Lasso:

$$\min_{\mathbf{w}} loss(\mathbf{w}; \mathbf{X}, \mathbf{y}) + \alpha||\mathbf{w}||_1. \tag{111}$$

In general, Grafting can work with the following objective function in which the features are assumed to arrive one at a time, the key challenge is how to efficiently update the parameter $\mathbf{w}$ when new features are continuously coming. The basic idea of Grafting streaming feature selection algorithm is based on the observation – incorporating a new feature into the model in Eq. (111) involves in adding a new penalty term into the model. For example, at the time step j, when a new feature f_j arrives, it incurs a regularization penalty

of $\alpha|\mathbf{w}_j|$. Therefore, the addition of the new feature f_j reduces the objective function value in Eq. (111) only when the reduction in the loss function part $loss(\mathbf{w}; \mathbf{X}, \mathbf{y})$ outweighs the increase in the ℓ_1-norm regularization. With this observation, the condition of accepting the new feature f_j is as follows:

$$\left| \frac{\partial loss(\mathbf{w}; \mathbf{X}, \mathbf{y})}{\partial \mathbf{w}_j} \right| > \alpha. \tag{112}$$

Otherwise, the Grafting algorithm will set the feature coefficient $\mathbf{w}_j$ of the new feature f_j to be zero. In the second step, when new features like f_j are accepted and included in the model, Grafting adopts a conjugate gradient (CG) procedure to optimize the model with respect to all current parameters.

5.1.2 Alpha-investing Algorithm (Supervised) (Zhou et al., 2005b)

Alpha-investing (Zhou et al., 2005b) is an adaptive complexity penalty method which dynamically changes the threshold of error reduction that is required to accept a new feature. It is motivated from a desire to control the false discovery rate (FDR) of newly arrived features such that a small portion of spurious features does not greatly affect the model accuracy. The detailed algorithm works as follows:

- Initialize $w_0 = 0$ (probability of false positives), $i = 0$ (index of features), selected features in the model $SF = \emptyset$
- *Step 1*: Get a new feature f_i
- *Step 2*: Set $\alpha_i = w_i/(2i)$
- *Step 3*:

$$\begin{aligned} &w_{i+1} = w_i - \alpha_i, SF = SF && \text{if } p\_value(f_i, SF) \geq \alpha_i \\ &w_{i+1} = w_i + \alpha_\Delta - \alpha_i,\ SF = SF \cup f_i && \text{if } p\_value(f_i, SF) < \alpha_i \end{aligned}$$

- *Step 4*: $i = i + 1$
- *Step 5*: Repeat *Step 1* to *Step 4*.

The threshold α_i corresponds to the probability of selecting a spurious feature at the time step i. The threshold α_i is adjusted by the wealth w_i, which denotes acceptable number of false positivly detected features at the current moment. The wealth w_i is increased when a feature is added to the model, otherwise it is decreased when a feature is not included to save for future features. More precisely, at each time step, the method calculates the p-value by using the fact that ΔLogliklohood is equivalent to t-statistics, the p-value denotes the probability that a feature coefficient could be set to nonzero when it is not (false positively detected). The basic idea of alpha-investing is to adaptively adjust the threshold such that when new features are selected and included into the model, it allows a higher chance of including incorrect features in the future. On the opposite side, each time when a new feature is not included (found to be not statistically significant), the wealth

is wasted and lowers the chance of finding more spurious features. However, one major limitation of this method is that it only tests newly arrived features while failing to take into consideration of the feature redundancy for old existing features. In (Dhillon et al., 2010), authors extended the alpha-investing algorithm and proposed a proposed a multiple streamwise feature selection algorithm to the case where there are multiple feature streams.

5.1.3 Online Streaming Feature Selection Algorithm (Supervised) (Wu et al., 2010, 2013)

Different from grafting and alpha-investing, authors studied the streaming feature selection problem from an information theoretic perspective by using the concept of Markov blanket (Wu et al., 2010, 2013). According to their definition, the whole feature set consists of four types of features: irrelevant features, redundant feature, weakly relevant but non-redundant features, and strongly relevant features. An optimal feature selection should select non-redundant and strongly relevant features. But features are dynamically arrived in a streaming fashion, it is difficult to find all strongly relevant and non-redundant features. The proposed method, OSFS is able to capture these non-redundant and strongly relevant features via two steps: (1) online relevance analysis, and (2) online redundancy analysis. A general framework of OSFS is listed as follows:

- Initialize selected features in the model $SF = \emptyset$
- *Step 1*: Get a new feature f_i
- *Step 2*: Online relevance analysis

$$\begin{array}{rl} \text{Discard feature } f_i & \text{if } f_i \text{ is relevant to the class label} \\ SF = SF \cup f_i & \text{otherwise} \end{array}$$

- *Step 3*: Online Redundancy Analysis
- *Step 4*: Repeat *Step 1* to *Step 3* until some stopping criteria are satisfied.

In step 2, the online relevance analysis step, OSFS discovers weakly relevant and strongly relevant features, and these features are added into the best candidate features (BCF). Otherwise, if the newly arrived feature is not relevant to the class label, it is discarded and not considered in the future step. In step 3, the online redundancy analysis step, OSFS dynamically eliminates redundant features in the selected subset using Markov Blanket. For each feature f_j in the best candidate set BCF, if there exists a subset of BCF making f_j and the class label conditionally independent, then f_j is removed from BCF. The most time consuming part of OSFS is the redundancy analysis phase, therefore, a fast-OSFS is proposed to improve efficiency. Fast-OSFS further divides this phase into inner-redundancy analysis part and outer-redundancy analysis part. In the inner-redundancy analysis part, fast-OSFS only re-examines the feature newly added into BCF, while the outer-redundancy analysis part re-examines each feature of BCF only when the process of generating a feature is stopped.

5.1.4 Streaming Feature Selection with Group Structures (Supervised) (Wang et al., 2013b; Li et al., 2013)

Previous mentioned streaming feature selection algorithms evaluate new features individually. However, streaming features may also exhibit group structures and current group feature selection algorithms such as Group Lasso cannot handle online processing.

Therefore, in (Wang et al., 2013b, 2015), authors propose an streaming group feature selection algorithm (OGFS) which consists of two parts: online intra-group selection and online inter-group selection. In the online intra-group selection phase, for the streaming features in a specific group, OGFS uses spectral feature selection techniques to assess if the newly arrived feature will increase the ratio between between-class distances and within-class distances or it is a significant feature with discriminative power. If the inclusion of this new feature will increase this ratio or is statistically significant, the new feature is included, otherwise it is discarded. After all feature groups are processed, in the online inter-group step, for the features from different feature groups, OGFS uses Lasso to select a subset of features to obtain an ultimate subset.

In addition to OGFS, a similar algorithm is proposed in (Li et al., 2013). The proposed algorithm, GFSSF also contains two steps: the feature level selection and group level selection. The difference is that it performs feature level selection and group level selection from an information theoretic perspective. In the feature level selection, it only processes features from the same group, and seeks for the best feature subset from the arrived features so far via relevance and redundancy analysis. Then in the group selection phase, it seeks for a set of feature groups that can cover as much uncertainty of the class labels as possible with a minimum cost. Afterwards, it obtains a subset of relevant features that is sparse at both the group level and the individual feature level.

5.1.5 Unsupervised Streaming Feature Selection in Social Media (Unsupervised) (Li et al., 2015)

Vast majority of streaming feature selection methods are supervised which utilize label information to guide feature selection process. However, in social media, it is easy to amass vast quantities of unlabeled data, while it is time and labor consuming to obtain labels. To deal with large-scale unlabeled data in social media, authors in (Li et al., 2015) proposed an USFS algorithm to study unsupervised streaming feature selection. The key idea of USFS is to utilize source information such as link information to enable unsupervised streaming feature selection. The work flow of the proposed framework USFS is shown in Figure 2. USFS first uncovers hidden social factors from link information by mixed membership stochastic blockmodel (Airoldi et al., 2009). Suppose that the number of instances is n and each instance is associated with a k dimensional latent factors. After obtaining the social latent factors $\mathbf{\Pi} \in \mathbb{R}^{n\times k}$ for each linked instance, USFS takes advantage of them as a constrain to perform selection. At a specific time step t, let $\mathbf{X}^{(t)}$, $\mathbf{W}^{(t)}$ denote the corresponding feature matrix, feature coefficient matrix. To model feature information, USFS constructs a graph $\mathcal{G}$ to represent feature similarity and $\mathbf{A}^{(t)}$ denotes the adjacency matrix of the graph, $\mathbf{L}^{(t)}$ is the corresponding laplacian matrix. Then the objective function

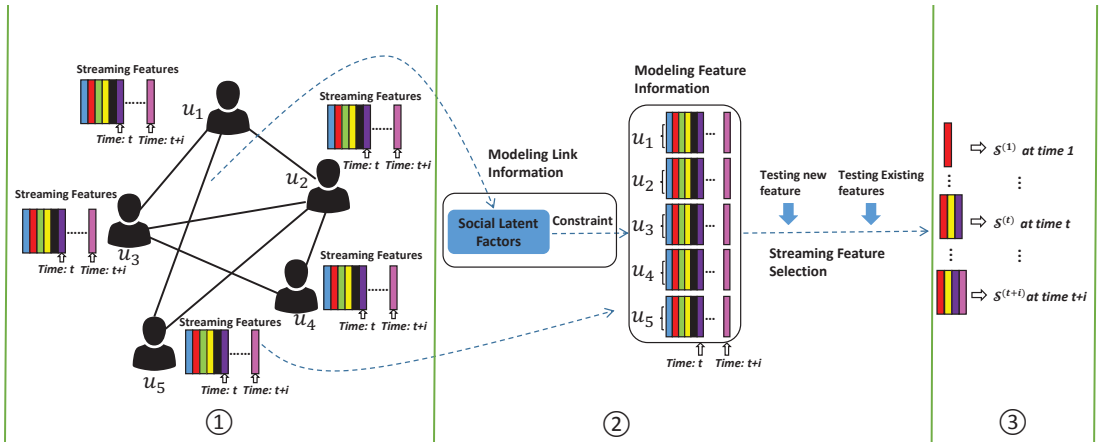


Figure 17: Workflow of USFS.

to achieve feature selection at the time step t is given as follows:

$$\min_{\mathbf{W}^{(t)}} \frac{1}{2}||\mathbf{X}^{(t)}\mathbf{W}^{(t)} - \mathbf{\Pi}||_F^2 + \alpha \sum_{i=1}^{k} \|(\mathbf{w}^{(t)})^i\|_1 + \frac{\beta}{2}||\mathbf{W}^{(t)}||_F^2 + \frac{\gamma}{2}||(\mathbf{X}^{(t)}\mathbf{W}^{(t)})^T(\mathbf{L}^{(t)})^{\frac{1}{2}}||_F^2, \quad (113)$$

where α is a sparse regularization parameter, β controls the robustness of the model and γ balances link information and feature information.

Assume at the next time step $t+1$ a new feature arrives, to test new features, USFS takes a similar strategy as Grafting to perform gradient test. Specifically, if the inclusion of the new feature is going to reduce the objective function in Eq. (113) at the new time step, the feature is accepted, otherwise the new feature can be removed. When new features are continuously being generated, they may take place of some existing features and some existing features may become outdated, therefore, USFS also investigates if it is necessary to remove any existing features by re-optimizing the model through A Broyden-Fletcher-Goldfarb-Shanno (BFGS) quasinewton method (Boyd and Vandenberghe, 2004).

5.2 Feature Selection Algorithms with Data Streams

In this subsection, we review the problem of feature selection with data streams which is considered as a dual problem of streaming feature selection. Most existing feature selection algorithms assume that all the data instances are available before performing feature selection. However, such assumptions are not always true in real-world applications that data instances are dynamically generated and arrive in a sequential manner. Therefore, it is necessary and urgent to come up with some solutions to deal with sequential data of high dimensionality.

5.2.1 Online Feature Selection (Supervised) (Wang et al., 2014)

In (Wang et al., 2014), an online feature selection algorithm (OFS) for binary classification is proposed. Let $\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_t ...\}$ and $\{y_1, y_2, ..., y_t ...\}$ denote a sequence of input data instances and input class labels, where each data instance $\mathbf{x}_i \in \mathbb{R}^d$ is in a d-dimensional

space and class label $y_i \in \{-1, +1\}$. The task of OFS is to learn a linear classifier $\mathbf{w}^{(t)} \in \mathbb{R}^d$ that can be used to classify each instance $\mathbf{x}_i$ by a linear function $\text{sign}(\mathbf{w}^{(t)}\mathbf{x}_i)$. To achieve the feature selection purpose, it requires that the linear classifier $\mathbf{w}^{(t)}$ has at most B-nonzero elements such that $\|\mathbf{w}^{(t)}\|_0 \leq B$. It indicates that at most B features will be used for classification. With a regularization parameter λ and a step size η, the algorithm of OFS works as follows:

- *Step 1*: Get a new data instance $\mathbf{x}_t$ and its class label y_t
- *Step 2*: Make a class label prediction $\text{sign}(\mathbf{w}^{(t)}\mathbf{x}_t)$ for the new instance
- *Step 3(a)*: If $\mathbf{x}_t$ is misclassified such that $y_i\mathbf{w}^{(t)}\mathbf{x}_t < 0$

$$\begin{aligned}\tilde{\mathbf{w}}_{t+1} &= (1 - \lambda\eta)\mathbf{w}_t + \eta y_t \mathbf{x}_t \\ \hat{\mathbf{w}}_{t+1} &= \min\{1, 1/\sqrt{\lambda}\|\tilde{\mathbf{w}}_{t+1}\|_2\}\tilde{\mathbf{w}}_{t+1} \\ \mathbf{w}_{t+1} &= Truncate(\hat{\mathbf{w}}_{t+1}, B)\end{aligned}$$

- *Step 3(b)*: $\mathbf{w}_{t+1} = (1 - \lambda\eta)\mathbf{w}_t$
- *Step 4*: Repeat *Step 1* to *Step 3(a)* or *Step 3(b)* until no new data instances arrive.

In Step 3(a), each time when a training instance $\mathbf{x}_t$ is misclassified, $\mathbf{w}_t$ is first updated by online gradient descent and then it is projected to a ℓ_2-norm ball to ensure that the classifier is bounded. After that, the new classifier $\hat{\mathbf{w}}_{t+1}$ is truncated by taking the most important B features. A subset of B features is output at each time step. The process repeats until there are no new data instances arrive anymore.

5.2.2 Unsupervised Feature Selection on Data Streams (Unsupervised) (Huang et al., 2015)

OFS assumes that the class labels of continuously generated data streams are available. However, it is not the case in many real-world applications that label information is costly to obtain. To timely select a subset of relevant features when unlabeled data are continuously being generated, authors in (Huang et al., 2015) propose a novel unsupervised feature selection method (FSDS) that is able to perform feature selection timely with only one pass of the data and utilize limited storage. The basic idea of FSDS is to use matrix sketching to efficiently maintain a low-rank approximation of the current observed data and then apply regularized regression to obtain the feature coefficients, which can further be used to obtain the importance of features. The authors empirically show that when some orthogonality conditions are satisfied, the ridge regression can replace the Lasso for feature selection, which is more computational efficient. Assume at a specific time step t, $\mathbf{X}^{(t)} \in \mathbb{R}^{n_t \times d}$ denotes the data matrix at that time step, its rank-k approximation is $\mathbf{X}_k^{(t)} = \sum_{i=1}^{k} \delta_i \mathbf{u}_i \mathbf{v}_i'$, where δ_i $(i \leq k)$ are the top-k singular values, $\mathbf{u}_i$ and $\mathbf{v}_i'$ are the corresponding left and right singular vectors, respectively. With these, the feature coefficients can be obtained by minimizing the following ridge regression problem:

$$\min_{\mathbf{W}^{(t)}} \|\mathbf{X}^{(t)}\mathbf{W}^{(t)} - \mathbf{U}_k^{(t)}\|_F^2 + \alpha\|\mathbf{W}^{(t)}\|_F^2, \tag{114}$$

where $\mathbf{U}_k^{(t)} \in \mathbb{R}^{n_t \times k}$ are the top-k left singular vectors of $\mathbf{X}^{(t)}$, α is a regularization parameter to avoid overfitting.

The bottleneck of Eq. (114) is that the singular value decomposition of $\mathbf{X}^{(t)}$ is computational expensive, especially when n_t is very large. Therefore, FSDS utilizes the matrix sketching strategy from (Liberty, 2013) to maintain a low-rank approximation of $\mathbf{X}^{(t)}$. Let $\mathbf{B}^{(t)} \in \mathbb{R}^{\ell \times d}$ denote the matrix sketch of $\mathbf{X}^{(t)}$ (ℓ). After some mathematical derivations, the objective of ridge regression is reformulated as follows:

$$\min_{\mathbf{W}^{(t)}} \|\mathbf{B}^{(t)}\mathbf{W}^{(t)} - \{\mathbf{e}_1, ..., \mathbf{e}_k\}\|_F^2 + \alpha\|\mathbf{W}^{(t)}\|_F^2, \tag{115}$$

where $\mathbf{e}_i \in \mathbb{R}^\ell$ is a vector with its i-th location as 1 and other locations as 0. By solving the optimization problem in Eq. (115), the importance of each feature f_i can be computed as:

$$score(j) = \max_i |\mathbf{W}^{(t)}(j, i)|. \tag{116}$$

The higher the feature score, the more important the feature is.

6. Performance Evaluation

In this section, we discuss the evaluation of feature selection algorithms, focusing on feature selection algorithms for generic data. We first introduce the developed feature selection repository, then we introduce the algorithms to be evaluated and some publicly available benchmark datasets we collect. At last, we introduce some widely adopted evaluation metrics and present the empirical experimental results.

6.1 Feature Selection Repository

First, we introduce our efforts in developing a feature selection repository – *scikit-feature*. The purpose of this feature selection repository is to collect some widely used feature selection algorithms that have been developed in the feature selection research to serve as a platform for facilitating their application, comparison and joint study. The feature selection repository also effectively assists researchers to achieve more reliable evaluation in the process of developing new feature selection algorithms.

We develop the open source feature selection repository *scikit-feature* by one of the most popular programming language – python. It contains around 40 popular feature selection algorithms, including most traditional feature selection algorithms mentioned in this survey and some structural and streaming feature selection algorithms. It is built upon one widely used machine learning package *scikit-learn* and two scientific computing packages *Numpy* and *Scipy*. At the same time, we also maintain a website (`http://featureselection.asu.edu/`) for this project which offers several sources such as public available benchmark datasets, performance evaluation of algorithms, test cases to run each algorithm. The source code of this repository is available at Github (`https://github.com/jundongl/scikit-feature`).

6.2 Algorithms

We empirically evaluate the performance of feature selection algorithms for generic data provided in the repository. Next, we will provide detailed information how these algorithms

are evaluated, including the datasets, evaluation criteria and experimental setup. The selected feature selection algorithms that will be evaluated are listed as follows. We list the following information of each algorithm: (1) supervised or unsupervised; (2) similarity based, information theoretical based, sparse learning based, or statistical based; (3) output: feature weighting or subset selection; (4) feature type: numerical or categorical. The first two items have been mentioned previously. The third item categorizes these algorithms based on the output. Feature weighing algorithms basically give each feature a score for ranking and feature subset algorithms only show which features are selected. The last item shows the feature types the feature selection method can handle with, continuous or discrete. For supervised feature selection methods, we also list if the method can tackle binary-class or multi-class classification problem.

1. Fisher Score: supervised, similarity, feature weight, continuous and discrete(multi-class)
2. ReliefF: supervised, similarity, feature weight, continuous and discrete(multi-class)
3. Trace Ratio: supervised, similarity, feature weight, continuous and discrete(multi-class)
4. Laplacian Score: unsupervised, similarity, feature weight, continuous and discrete
5. SPEC: unsupervised, similarity, feature weight, continuous and discrete
6. MIM: supervised, information theoretic, feature weight, discrete(multi-class)
7. MIFS: supervised, information theoretic, feature weight, discrete(multi-class)
8. MRMR: supervised, information theoretic, feature weight, discrete(multi-class)
9. CIFE: supervised, information theoretic, feature weight, discrete(multi-class)
10. JMI: supervised, information theoretic, feature weight, discrete(multi-class)
11. CMIM: supervised, information theoretic, feature weight, discrete(multi-class)
12. ICAP: supervised, information theoretic, feature weight, discrete(multi-class)
13. DISR: supervised, information theoretic, feature weight, discrete(multi-class)
14. FCBF: supervised, information theoretic, feature subset, discrete(multi-class)
15. RFS: supervised, sparse learning, feature weight, continuous and discrete(multi-class)
16. Least square loss ($\ell_{2,1}$): supervised, sparse learning, feature weight, continuous and discrete(multi-class)
17. Logistic loss ($\ell_{2,1}$): supervised, sparse learning, feature weight, continuous and discrete(multi-class)
18. MCFS: unsupervised, sparse learning, feature weight, continuous and discrete
19. UDFS: unsupervised, sparse learning, feature weight, continuous and discrete
20. NDFS: unsupervised, sparse learning, feature weight, continuous and discrete
21. Low variance: unsupervised, statistical, feature subset, discrete(binary-class)
22. T-score: supervised, statistical, feature weight, continuous and discrete(binary-class)
23. F-score: supervised, statistical, feature weight, continuous and discrete(multi-class)
24. Chi-square: supervised, statistical, feature weight, discrete(multi-class)
25. Gini Index: supervised, statistical, feature weight, discrete(multi-class)

Dataset	Type	Feature value	# of Features	# of Instances	# of Classes
BASEHOCK	Text	Continuous	4862	1993	2
PCMAC	Text	Continuous	3289	1943	2
RELATHE	Text	Continuous	4322	1427	2
COIL20	Image	Continuous	1024	1440	20
ORL	Image	Continuous	1024	400	40
orlraws10P	Image	Continuous	10304	100	10
pixraw10P	Image	Continuous	10000	100	10
warpAR10P	Image	Continuous	2400	130	10
warpPIE10P	Image	Continuous	2420	210	10
Yale	Image	Continuous	1024	165	15
USPS	Image	Continuous	256	9298	10
ALLAML	Bio	Continuous	7129	72	2
Carcinom	Bio	Continuous	9182	174	11
CLL_SUB_111	Bio	Continuous	11340	111	3
colon	Bio	Discrete	2000	62	2
GLA_BRA_180	Bio	Continuous	49151	180	4
GLI_85	Bio	Continuous	22283	85	2
GLIOMA	Bio	Continuous	4434	50	4
leukemia	Bio	Discrete	7070	72	2
lung	Bio	Continuous	3312	203	5
lung_small	Bio	Discrete	325	73	7
lymphoma	Bio	Discrete	4026	96	9
nci9	Bio	Discrete	9712	60	9
gisette	Image	Continuous	5000	7000	2
Prostate_GE	Bio	Continuous	5966	102	2
SMK_CAN_187	Bio	Continuous	19993	187	2
TOX_171	Bio	Continuous	5748	171	4
arcene	Mass Spectrometry	Continuous	10000	200	2
Isolet	Spoken letter	Continuous	617	1560	26
madelon	Artificial	Continuous	500	2600	2

Table 2: Detailed information of benchmark datasets.

6.3 Datasets

To test different algorithms, we collect 25 public available benchmark datasets to evaluate the performance of feature selection algorithms. We list the detailed information of each dataset in Table 2. We carefully select datasets from different categories, e.g., text data, image data, biological data, and some others. The features in these datasets are either in numerical or categorical values. We also present the number of features, number of instances and the number of classes for each dataset. The heterogeneity of the data is important for exposing the strength and weakness of algorithms in different applications.

6.4 Evaluation Metrics

Next, we introduce the widely adopted way to evaluate the performance of feature selection algorithms. We have different evaluation metrics for supervised and unsupervised methods. For algorithms of different output types, different evaluation strategies are used:

1. If it is a feature weighting method that outputs the feature score for each feature, then the quality of the first $\{5, 10, 15, ..., 295, 300\}$ features are evaluated respectively.

2. If it is a feature subset selection method that only outputs which features are selected, then we use all the selected features to perform the evaluation.

Supervised Methods: To test performance of the supervised feature selection algorithms, the evaluation framework introduced in Figure (3) is used. The whole dataset is usually divided into two parts - the training set $\mathcal{T}$ and test set $\mathcal{U}$. Feature selection algorithms will be first applied on the training set $\mathcal{T}$ to obtain a subset of relevant features $\mathcal{F}$. Then the test set on the selected features are acting as input to a classification model for the testing purpose. In the experiments, we use classification accuracy to evaluate the classification performance, and three classification models, Linear SVM, Decision Tree, Naïve Bayes are used. To get more reliable results, we adopt 10-fold cross validation, and the final classification performance are reported as an average over 10 folds. The higher the average classification accuracy, the better the feature selection algorithm is.

Unsupervised Methods: Following the standard way to assess unsupervised feature selection, we evaluate the unsupervised feature selection algorithms in terms of clustering performance. Two commonly used clustering performance metrics, i.e., *normalized mutual information* (NMI) and *accuracy* (ACC) are used.

Specifically, Let C and C' denote the clustering results from ground truth class labels and the predicted cluster labels, respectively. The mutual information between two clusters C and C' is:

$$MI(C, C') = \sum_{c_i \in C, c'_j \in C'} p(c_i, c'_j) log \frac{p(c_i, c'_j)}{p(c_i)p(c'_j)} \tag{117}$$

where $p(c_i)$ and $p(c'_j)$ are the probabilities of instances in cluster c_i and c'_j, respectively. $p(c_i, c'_j)$ indicates the probability of instances in cluster c_i and in c'_j at the same time. Then, NMI is defined as:

$$NMI(C, C') = \frac{MI(C, C')}{max(H(C), H(C')} \tag{118}$$

where $H(C)$ and $H(C')$ represent the entropies of clusterings C and C', respectively.

Let p_i and q_i be the clustering result and the ground truth label for instance u_i, respectively. Then, accuracy (ACC) is defined as:

$$ACC = \frac{1}{n} \sum_{i=1}^{n} \delta(q_i, map(p_i)) \tag{119}$$

where n is the total number of data instances, $\delta(.)$ is an indicator function such that $\delta(x, y) = 1$ if $x = y$, otherwise $\delta(x, y) = 0$. $map(x)$ permutes the predicted cluster labels to match the ground truth as much as possible.

Each feature selection algorithm is first applied to select features, then K-means clustering is performed based on the selected features. We repeat the K-means algorithm 20 times and report the average clustering results since K-means may converge to local optimal.

6.5 Experimental Results

The experimental results can be obtained from our repository project website (`http://featureselection.asu`
For each dataset, we list all applicable feature selection algorithms along with its evaluation on either classification or clustering task.

7. Open Problems and Challenges

Over the past two decades, there are tremendous amount of work in developing feature selection algorithms for both theoretical analysis and real-world applications. However, we still believe there are more work can be done in this community. Here are several challenges and concerns that we need to mention and discuss.

7.1 Scalability

With the tremendous growth of dataset sizes, the scalability of most current feature selection algorithms may be jeopardized. In many scientific and business applications, data are usually measured in terabyte (1TB = 10^{12} bytes). Normally, datasets in the scale of terabytes cannot be loaded into the memory directly and therefore limits the usability of most feature selection algorithms. Currently, there are some attempts to use distributed programming frameworks such as MapReduce and MPI to perform parallel feature selection for very large scale datasets (Singh et al., 2009; Zhao et al., 2013; Yamada et al., 2014). In addition, most feature selection algorithms proposed so far require time complexity proportional to $O(d^2)$ or even $O(d^3)$, where d is the feature dimension. Recently, big data of ultrahigh dimensionality has emerged in many real-world applications such as text mining and information retrieval. Most feature selection algorithms does not scale well on the ultrahigh dimensional data, its efficiency deteriorates quickly or is even computational infeasible. In this case, well-designed feature selection algorithms in linear or sub-linear running time is more preferred (Fan et al., 2009; Tan et al., 2014). Moreover, in some online classification or online clustering tasks, the scalability of feature selection algorithms is also a big issue. For example, the data streams or feature streams may be infinite and cannot be loaded into the memory, therefore we can only make one pass of the data where the second pass is either unavailable or computational expensive. Even though feature selection algorithms can reduce the issue of scalability for online classification or clustering, these methods either require to keep full dimensionality in the memory or require iterative processes to visit data instances more than once, which limit their practical usages. In conclusion, even though there are preliminary work to increase the scalability of feature selection algorithms, we believe that the scalability problem should be given more attention to keep pace with the rapid growth of very large-scale and fast streaming data.

7.2 Stability

For supervised feature selection algorithms, their performance are usually evaluated by the classification accuracy. In addition to accuracy, the stability of these algorithms is also an important consideration when developing new feature selection algorithms. A motivating example from the field of bioinformatics shows that domain experts would like to see the same set or similar set of genes (features) to be selected each time when they obtain new samples in the small amount of perturbation. Otherwise domain experts would not trust these algorithms when they get quite different sets of features with small data perturbation. Considering its importance in practical usage, stability of feature selection algorithms has received increasing attention in the community (Kalousis et al., 2007; He and Yu, 2010). It is observed that many well-known feature selection algorithms suffer from the low stability

problem after the small data perturbation is introduced in the training set. It is also found in (Alelyani et al., 2011) that the underlying characteristics of data may greatly affect the stability of feature selection algorithms and the stability issue may also be data dependent. These factors include the dimensionality of feature, number of data instances, etc.

In against with supervised feature selection, stability of unsupervised feature selection algorithms has not be well studied yet. Studying stability for unsupervised feature selection is much more difficult than that of the supervised methods. The reason is that in unsupervised feature selection, we do not have enough prior knowledge about the cluster structure of the data. Thus we are uncertain that if the new data instance, i.e., the perturbation belongs to any existing clusters or will introduce new clusters. While in supervised feature selection, we have the prior knowledge about the label of each data instance, and a new sample that does not belong to any existing classes will be considered as an outlier and we do not need to modify the selected feature set to adapt to the outliers. In other words, unsupervised feature selection is more sensitive to noise and the noise will affects the stability of the algorithm.

7.3 Model Selection

For most feature selection algorithms especially for feature weighting methods, we have to specify the number of selected features. However, it is often unknown what is the optimal number of selected features. With too large number of selected features, it may increase the risk in including some noisy, redundant and irrelevant features which may jeopardize the learning performance. On the other hand, it is also not good to include too small number of selected features, since some relevant features may be eliminated. In practice, we usually adopt a heuristic way to grid search the number of selected features and pick the number that has the best classification or clustering performance, but the whole process is computational expensive. It is still an open and challenging problem to determine the optimal number of selected features.

In addition to the optimal number of selected features, we also need to specify the number of clusters or pseudo classes for unsupervised feature selection algorithms. In real world problems, we usually have limited knowledge about the clustering structure of the data. Choosing different number of clusters may lead to merging totally different small clusters into one big cluster or splitting one big cluster into smaller ones. As a consequence, it may result in finding totally different subsets of features. Hence, determining the optimal number of clusters is almost impossible. Some work has been done to estimate these tricky parameters. For instance, in (Tibshirani et al., 2001), a principled way to estimate the number of suitable clusters in a dataset is proposed. However, it is still not clear how to find the best number of clusters for unsupervised feature selection. All in all, we believe that the model selection problem should be paid more attention.

8. Conclusion

Feature selection is effective in preprocessing data and reducing data dimensionality that is an essential to successful data mining and machine learning applications. Meanwhile, it has been a hot research topic with practical significance in many areas such as statistics, pattern recognition, machine learning, and data mining (including web, text, image, and

microarrays). The objectives of feature selection include: building simpler and more comprehensible models, improving data mining performance, and helping prepare, clean, and understand data. The past few years have witnessed the development of hundreds of new feature selection methods. This survey article aims to provide a comprehensive overview about recent advances in feature selection. We first introduce basic concepts of feature selection and emphasis the importance of applying feature selection algorithms to solve practical problems. Then, we classify traditional feature selection algorithms from label perspective and search strategy perspective. Since current categorization cannot meet the rapid development in feature selection, we propose to review recent advances in feature selection algorithms from a data perspective. Following the taxonomy in Figure (7), we surveyed the family of feature selection algorithms in four parts: (1) feature Selection with generic Data; (2) feature selection with structure features; (3) feature selection with heterogeneous data; and (4) feature selection with streaming data. Specifically, we further classify feature selection algorithms with generic data into similarity based, information theoretical based, sparse learning based and statistical based methods from their properties. For feature selection with structure features, we consider three types of structural features, namely group, tree and graph features. The third part feature selection with heterogeneous data consists of feature selection algorithms for linked data, multi-source and multi-view data. The fourth part consists of feature selection for streaming data and streaming features. To facilitate the research on feature selection, in this survey, we also present a feature selection repository - *scikit-feature*, which includes some of the most popular feature selection algorithms that have been developed in the past few decades. We also provide some suggestions on how to evaluate these feature selection algorithms, either supervised or unsupervised methods. At the end of the survey, we present some open problems and challenges that need to be paid more attention in the future feature selection research.

It also should be mentioned that the aim of the survey is not to claim the superiority of some feature selection algorithms over others. On the other hand, our goal is to provide a comprehensive structured list of recent advances of feature selection algorithms and a feature selection repository to promote the research in this community. As a matter of fact, it is at the discretion of users to decide which algorithms or which tools to use in practice.

References

Edoardo M Airoldi, David M Blei, Stephen E Fienberg, and Eric P Xing. Mixed membership stochastic blockmodels. In *Advances in Neural Information Processing Systems*, pages 33–40, 2009.

Salem Alelyani, Huan Liu, and Lei Wang. The effect of the characteristics of the dataset on the selection stability. In *Tools with Artificial Intelligence (ICTAI), 2011 23rd IEEE International Conference on*, pages 970–977. IEEE, 2011.

Salem Alelyani, Jiliang Tang, and Huan Liu. Feature selection for clustering: A review. *Data Clustering: Algorithms and Applications*, 29, 2013.

Francis R Bach. Consistency of the group lasso and multiple kernel learning. *The Journal of Machine Learning Research*, 9:1179–1225, 2008.

Kevin Bache and Moshe Lichman. Uci machine learning repository, 2013.

Roberto Battiti. Using mutual information for selecting features in supervised neural net learning. *Neural Networks, IEEE Transactions on*, 5(4):537–550, 1994.

Jinbo Bi, Tao Xiong, Shipeng Yu, Murat Dundar, and R Bharat Rao. An improved multi-task learning approach with applications in medical diagnosis. In *Machine Learning and Knowledge Discovery in Databases*, pages 117–132. Springer, 2008.

Mustafa Bilgic, Lilyana Mihalkova, and Lise Getoor. Active learning for networked data. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 79–86, 2010.

Howard D Bondell and Brian J Reich. Simultaneous regression shrinkage, variable selection, and supervised clustering of predictors with oscar. *Biometrics*, 64(1):115–123, 2008.

Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

Gavin Brown, Adam Pocock, Ming-Jie Zhao, and Mikel Luján. Conditional likelihood maximisation: a unifying framework for information theoretic feature selection. *The Journal of Machine Learning Research*, 13(1):27–66, 2012.

Deng Cai, Chiyuan Zhang, and Xiaofei He. Unsupervised feature selection for multi-cluster data. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 333–342. ACM, 2010.

Emmanuel Candes and Terence Tao. The dantzig selector: statistical estimation when p is much larger than n. *The Annals of Statistics*, pages 2313–2351, 2007.

Pak K Chan, Martine DF Schlag, and Jason Y Zien. Spectral k-way ratio-cut partitioning and clustering. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 13(9):1088–1096, 1994.

Girish Chandrashekar and Ferat Sahin. A survey on feature selection methods. *Computers & Electrical Engineering*, 40(1):16–28, 2014.

Fan RK Chung. *Spectral graph theory*, volume 92. American Mathematical Soc., 1997.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3): 273–297, 1995.

Thomas M Cover and Joy A Thomas. *Elements of information theory*. John Wiley & Sons, 2012.

Alexandre d'Aspremont, Laurent El Ghaoui, Michael I Jordan, and Gert RG Lanckriet. A direct formulation for sparse pca using semidefinite programming. *SIAM review*, 49(3): 434–448, 2007.

John C Davis and Robert J Sampson. *Statistics and data analysis in geology*, volume 646. Wiley New York et al., 1986.

Paramveer S Dhillon, Dean P Foster, and Lyle H Ungar. Feature selection using multiple streams. In *International Conference on Artificial Intelligence and Statistics*, pages 153–160, 2010.

Chris Ding, Ding Zhou, Xiaofeng He, and Hongyuan Zha. R 1-pca: rotational invariant l 1-norm principal component analysis for robust subspace factorization. In *Proceedings of the 23rd international conference on Machine learning*, pages 281–288. ACM, 2006.

James Dougherty, Ron Kohavi, Mehran Sahami, et al. Supervised and unsupervised discretization of continuous features. In *Machine learning: proceedings of the twelfth international conference*, volume 12, pages 194–202, 1995.

Liang Du and Yi-Dong Shen. Unsupervised feature selection with adaptive structure learning. *arXiv preprint arXiv:1504.00736*, 2015.

Richard O Duda, Peter E Hart, and David G Stork. *Pattern classification*. John Wiley & Sons, 2012.

Bradley Efron, Trevor Hastie, Iain Johnstone, Robert Tibshirani, et al. Least angle regression. *The Annals of statistics*, 32(2):407–499, 2004.

Ali El Akadi, Abdeljalil El Ouardighi, and Driss Aboutajdine. A powerful feature selection approach based on mutual information. *International Journal of Computer Science and Network Security*, 8(4):116, 2008.

A Evgeniou and Massimiliano Pontil. Multi-task feature learning. *Advances in neural information processing systems*, 19:41, 2007.

Jianqing Fan and Runze Li. Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American statistical Association*, 96(456):1348–1360, 2001.

Jianqing Fan, Richard Samworth, and Yichao Wu. Ultrahigh dimensional feature selection: beyond the linear model. *The Journal of Machine Learning Research*, 10:2013–2038, 2009.

Ahmed K Farahat, Ali Ghodsi, and Mohamed S Kamel. An efficient greedy method for unsupervised feature selection. In *Data Mining (ICDM), 2011 IEEE 11th International Conference on*, pages 161–170. IEEE, 2011.

Christiane Fellbaum. *WordNet*. Wiley Online Library, 1998.

Yinfu Feng, Jun Xiao, Yueting Zhuang, and Xiaoming Liu. Adaptive unsupervised multi-view feature selection for visual concept recognition. In *Computer Vision–ACCV 2012*, pages 343–357. Springer, 2013.

François Fleuret. Fast binary feature selection with conditional mutual information. *The Journal of Machine Learning Research*, 5:1531–1555, 2004.

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. A note on the group lasso and a sparse group lasso. *arXiv preprint arXiv:1001.0736*, 2010.

Keinosuke Fukunaga. *Introduction to statistical pattern recognition*. Academic press, 2013.

CW Gini. Variability and mutability, contribution to the study of statistical distribution and relaitons. *Studi Economico-Giuricici della R*, 1912.

David E Golberg. Genetic algorithms in search, optimization, and machine learning. *Addion wesley*, 1989, 1989.

Gene H Golub and Charles F Van Loan. *Matrix computations*, volume 3. JHU Press, 2012.

Quanquan Gu and Jiawei Han. Towards feature selection in network. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 1175–1184. ACM, 2011.

Quanquan Gu, Zhenhui Li, and Jiawei Han. Generalized fisher score for feature selection. In *Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence*, pages 266–273, 2011.

Baofeng Guo and Mark S Nixon. Gait feature subset selection by mutual information. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 39(1):36–46, 2009.

Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *The Journal of Machine Learning Research*, 3:1157–1182, 2003.

Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. Gene selection for cancer classification using support vector machines. *Machine learning*, 46(1-3):389–422, 2002.

Isabelle Guyon, Steve Gunn, Masoud Nikravesh, and Lofti A Zadeh. *Feature extraction: foundations and applications*, volume 207. Springer, 2008.

Mark A Hall and Lloyd A Smith. Feature selection for machine learning: Comparing a correlation-based filter approach to the wrapper. In *FLAIRS conference*, volume 1999, pages 235–239, 1999.

David R Hardoon, Sandor Szedmak, and John Shawe-Taylor. Canonical correlation analysis: An overview with application to learning methods. *Neural computation*, 16(12):2639–2664, 2004.

Trevor Hastie, Robert Tibshirani, Jerome Friedman, and James Franklin. The elements of statistical learning: data mining, inference and prediction. *The Mathematical Intelligencer*, 27(2):83–85, 2005.

Trevor Hastie, Robert Tibshirani, and Martin Wainwright. *Statistical Learning with Sparsity: The Lasso and Generalizations*. CRC Press, 2015.

Xiaofei He, Deng Cai, and Partha Niyogi. Laplacian score for feature selection. In *Advances in neural information processing systems*, pages 507–514, 2005.

Zengyou He and Weichuan Yu. Stable feature selection for biomarker discovery. *Computational biology and chemistry*, 34(4):215–225, 2010.

David W Hosmer Jr and Stanley Lemeshow. *Applied logistic regression*. John Wiley & Sons, 2004.

Chenping Hou, Feiping Nie, Dongyun Yi, and Yi Wu. Feature selection via joint embedding learning and sparse regression. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, pages 1324–1329. Citeseer, 2011.

Xia Hu, Jiliang Tang, Huiji Gao, and Huan Liu. Actnet: Active learning for networked texts in microblogging. In *SDM*, pages 306–314. Citeseer, 2013.

Hao Huang, Shinjae Yoo, and S Kasiviswanathan. Unsupervised feature selection on data streams. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 1031–1040. ACM, 2015.

Junzhou Huang, Tong Zhang, and Dimitris Metaxas. Learning with structured sparsity. *The Journal of Machine Learning Research*, 12:3371–3412, 2011.

Laurent Jacob, Guillaume Obozinski, and Jean-Philippe Vert. Group lasso with overlap and graph lasso. In *Proceedings of the 26th annual international conference on machine learning*, pages 433–440. ACM, 2009.

Aleks Jakulin. *Machine learning based on attribute interactions*. PhD thesis, Univerza v Ljubljani, 2005.

Gareth M James, Peter Radchenko, and Jinchi Lv. Dasso: connections between the dantzig selector and lasso. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 71(1):127–142, 2009.

Rodolphe Jenatton, Julien Mairal, Francis R Bach, and Guillaume R Obozinski. Proximal methods for sparse hierarchical dictionary learning. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 487–494, 2010.

Rodolphe Jenatton, Jean-Yves Audibert, and Francis Bach. Structured variable selection with sparsity-inducing norms. *The Journal of Machine Learning Research*, 12:2777–2824, 2011.

Ian Jolliffe. *Principal component analysis*. Wiley Online Library, 2002.

Alexandros Kalousis, Julien Prados, and Melanie Hilario. Stability of feature selection algorithms: a study on high-dimensional spaces. *Knowledge and information systems*, 12 (1):95–116, 2007.

Seyoung Kim and Eric P Xing. Statistical estimation of correlated genome associations to a quantitative trait network. *PLoS Genet*, 5(8):e1000587, 2009.

Seyoung Kim and Eric P Xing. Tree-guided group lasso for multi-task regression with structured sparsity. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 543–550, 2010.

Kenji Kira and Larry A Rendell. The feature selection problem: Traditional methods and a new algorithm. In *AAAI*, volume 2, pages 129–134, 1992a.

Kenji Kira and Larry A Rendell. A practical approach to feature selection. In *Proceedings of the ninth international workshop on Machine learning*, pages 249–256, 1992b.

Ron Kohavi and George H John. Wrappers for feature subset selection. *Artificial intelligence*, 97(1):273–324, 1997.

Daphne Koller and Mehran Sahami. Toward optimal feature selection. In *In 13th International Conference on Machine Learning*, 1995.

Sotiris Kotsiantis and Dimitris Kanellopoulos. Discretization techniques: A recent survey. *GESTS International Transactions on Computer Science and Engineering*, 32(1):47–58, 2006.

Gert RG Lanckriet, Nello Cristianini, Peter Bartlett, Laurent El Ghaoui, and Michael I Jordan. Learning the kernel matrix with semidefinite programming. *The Journal of Machine Learning Research*, 5:27–72, 2004.

David D Lewis. Feature selection and feature extraction for text categorization. In *Proceedings of the workshop on Speech and Natural Language*, pages 212–217. Association for Computational Linguistics, 1992.

Haiguang Li, Xindong Wu, Zhao Li, and Wei Ding. Group feature selection with streaming features. In *Data Mining (ICDM), 2013 IEEE 13th International Conference on*, pages 1109–1114. IEEE, 2013.

Jundong Li, Xia Hu, Jiliang Tang, and Huan Liu. Unsupervised streaming feature selection in social media. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 1041–1050. ACM, 2015.

Zechao Li, Yi Yang, Jing Liu, Xiaofang Zhou, and Hanqing Lu. Unsupervised feature selection using nonnegative spectral analysis. In *AAAI*, 2012.

Edo Liberty. Simple and deterministic matrix sketching. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 581–588. ACM, 2013.

Dahua Lin and Xiaoou Tang. Conditional infomax learning: an integrated framework for feature extraction and fusion. In *Computer Vision–ECCV 2006*, pages 68–82. Springer, 2006.

Huan Liu and Rudy Setiono. Chi2: Feature selection and discretization of numeric attributes. In *tai*, page 388. IEEE, 1995.

J. Liu, S. Ji, and J. Ye. *SLEP: Sparse Learning with Efficient Projections*. Arizona State University, 2009a. URL `http://www.public.asu.edu/~jye02/Software/SLEP`.

Jun Liu and Jieping Ye. Efficient euclidean projections in linear time. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 657–664. ACM, 2009.

Jun Liu and Jieping Ye. Moreau-yosida regularization for grouped tree structure learning. In *Advances in Neural Information Processing Systems*, pages 1459–1467, 2010.

Jun Liu, Shuiwang Ji, and Jieping Ye. Multi-task feature learning via efficient l 2, 1-norm minimization. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*, pages 339–348. AUAI Press, 2009b.

Xinwang Liu, Lei Wang, Jian Zhang, Jianping Yin, and Huan Liu. Global and local structure preservation for feature selection. *Neural Networks and Learning Systems, IEEE Transactions on*, 25(6):1083–1095, 2014.

Zhenqiu Liu, Feng Jiang, Guoliang Tian, Suna Wang, Fumiaki Sato, Stephen J Meltzer, and Ming Tan. Sparse logistic regression with lp penalty for biomarker identification. *Statistical Applications in Genetics and Molecular Biology*, 6(1), 2007.

Bo Long, Zhongfei Mark Zhang, Xiaoyun Wu, and Philip S Yu. Spectral clustering for multi-type relational data. In *Proceedings of the 23rd international conference on Machine learning*, pages 585–592. ACM, 2006.

Bo Long, Zhongfei Mark Zhang, and Philip S Yu. A probabilistic framework for relational clustering. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 470–479. ACM, 2007.

Shuangge Ma, Xiao Song, and Jian Huang. Supervised group lasso with applications to microarray data analysis. *BMC bioinformatics*, 8(1):60, 2007.

Sofus A Macskassy and Foster Provost. Classification in networked data: A toolkit and a univariate case study. *The Journal of Machine Learning Research*, 8:935–983, 2007.

Peter V Marsden and Noah E Friedkin. Network studies of social influence. *Sociological Methods & Research*, 22(1):127–151, 1993.

Mahdokht Masaeli, Yan Yan, Ying Cui, Glenn Fung, and Jennifer G Dy. Convex principal feature selection. In *SDM*, pages 619–628. SIAM, 2010.

James McAuley, Ji Ming, Darryl Stewart, and Philip Hanna. Subband correlation and robust speech recognition. *Speech and Audio Processing, IEEE Transactions on*, 13(5): 956–964, 2005.

Miller McPherson, Lynn Smith-Lovin, and James M Cook. Birds of a feather: Homophily in social networks. *Annual review of sociology*, pages 415–444, 2001.

Lukas Meier, Sara Van De Geer, and Peter Bühlmann. The group lasso for logistic regression. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 70(1):53–71, 2008.

Patrick E Meyer and Gianluca Bontempi. On the use of variable complementarity for feature selection in cancer classification. In *Applications of Evolutionary Computing*, pages 91–102. Springer, 2006.

Patrick Emmanuel Meyer, Colas Schretter, and Gianluca Bontempi. Information-theoretic feature selection in microarray data using variable complementarity. *Selected Topics in Signal Processing, IEEE Journal of*, 2(3):261–274, 2008.

Pabitra Mitra, CA Murthy, and Sankar K. Pal. Unsupervised feature selection using feature similarity. *IEEE transactions on pattern analysis and machine intelligence*, 24(3):301–312, 2002.

Steven A Morris. Manifestation of emerging specialties in journal literature: A growth model of papers, references, exemplars, bibliographic coupling, cocitation, and clustering coefficient distribution. *Journal of the American Society for Information Science and Technology*, 56(12):1250–1273, 2005.

Patrenahalli M Narendra and Keinosuke Fukunaga. A branch and bound algorithm for feature subset selection. *Computers, IEEE Transactions on*, 100(9):917–922, 1977.

Yurii Nesterov. *Introductory lectures on convex optimization*, volume 87. Springer Science & Business Media, 2004.

Mark EJ Newman and Michelle Girvan. Finding and evaluating community structure in networks. *Physical review E*, 69(2):026113, 2004.

Andrew Y Ng, Michael I Jordan, Yair Weiss, et al. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 2:849–856, 2002.

Feiping Nie, Shiming Xiang, Yangqing Jia, Changshui Zhang, and Shuicheng Yan. Trace ratio criterion for feature selection. In *AAAI*, volume 2, pages 671–676, 2008.

Feiping Nie, Heng Huang, Xiao Cai, and Chris H Ding. Efficient and robust feature selection via joint 2, 1-norms minimization. In *Advances in neural information processing systems*, pages 1813–1821, 2010.

Guillaume Obozinski, Ben Taskar, and Michael Jordan. Joint covariate selection for grouped classification. Technical report, Technical report, Statistics Department, UC Berkeley, 2007.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Hanchuan Peng, Fuhui Long, and Chris Ding. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(8):1226–1238, 2005.

Jie Peng, Ji Zhu, Anna Bergamaschi, Wonshik Han, Dong-Young Noh, Jonathan R Pollack, and Pei Wang. Regularized multivariate regression for identifying master predictors with application to integrative genomics study of breast cancer. *The annals of applied statistics*, 4(1):53, 2010.

Simon Perkins and James Theiler. Online feature selection using grafting. In *ICML*, pages 592–599, 2003.

Simon Perkins, Kevin Lacker, and James Theiler. Grafting: Fast, incremental feature selection by gradient descent in function space. *The Journal of Machine Learning Research*, 3:1333–1356, 2003.

Mingjie Qian and Chengxiang Zhai. Robust unsupervised feature selection. In *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*, pages 1621–1627. AAAI Press, 2013.

J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

J Ross Quinlan. *C4. 5: programs for machine learning*. Morgan Kaufmann, 1993.

Marko Robnik-Šikonja and Igor Kononenko. Theoretical and empirical analysis of relieff and rrelieff. *Machine learning*, 53(1-2):23–69, 2003.

Sam T Roweis and Lawrence K Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

Ted Sandler, John Blitzer, Partha P Talukdar, and Lyle H Ungar. Regularized learning with networks of features. In *Advances in neural information processing systems*, pages 1401–1408, 2009.

Bernhard Scholkopft and Klaus-Robert Mullert. Fisher discriminant analysis with kernels. *Neural networks for signal processing IX*, 1:1, 1999.

Mark R Segal, Kam D Dahlquist, and Bruce R Conklin. Regression approaches for microarray data analysis. *Journal of Computational Biology*, 10(6):961–980, 2003.

Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93, 2008.

Claude Elwood Shannon. A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(1):3–55, 2001.

Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(8):888–905, 2000.

Sameer Singh, Jeremy Kubica, Scott Larsen, and Daria Sorokina. Parallel large scale feature selection for logistic regression. In *SDM*, pages 1172–1183. SIAM, 2009.

Masashi Sugiyama. Local fisher discriminant analysis for supervised dimensionality reduction. In *Proceedings of the 23rd international conference on Machine learning*, pages 905–912. ACM, 2006.

Mingkui Tan, Ivor W Tsang, and Li Wang. Towards ultrahigh dimensional feature selection for big data. *The Journal of Machine Learning Research*, 15(1):1371–1429, 2014.

Jiliang Tang and Huan Liu. Feature selection with linked data in social media. In *SDM*, pages 118–128. SIAM, 2012a.

Jiliang Tang and Huan Liu. Unsupervised feature selection for linked social media data. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 904–912. ACM, 2012b.

Jiliang Tang and Huan Liu. Feature selection for social media data. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 8(4):19, 2014a.

Jiliang Tang and Huan Liu. An unsupervised feature selection framework for social media data. *Knowledge and Data Engineering, IEEE Transactions on*, 26(12):2914–2927, 2014b.

Jiliang Tang, Xia Hu, Huiji Gao, and Huan Liu. Unsupervised feature selection for multi-view data in social media. In *SDM*, pages 270–278, 2013.

Jiliang Tang, Salem Alelyani, and Huan Liu. Feature selection for classification: A review. *Data Classification: Algorithms and Applications*, page 37, 2014.

Joshua B Tenenbaum, Vin De Silva, and John C Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

Robert Tibshirani, Guenther Walther, and Trevor Hastie. Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(2):411–423, 2001.

Robert Tibshirani, Michael Saunders, Saharon Rosset, Ji Zhu, and Keith Knight. Sparsity and smoothness via the fused lasso. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(1):91–108, 2005.

William T Vetterling, Saul A Teukolsky, and William H Press. *Numerical recipes: example book (C)*. Press Syndicate of the University of Cambridge, 1992.

Michel Vidal-Naquet and Shimon Ullman. Object recognition with informative features and linear classification. In *ICCV*, volume 3, page 281, 2003.

Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4): 395–416, 2007.

Hua Wang, Feiping Nie, and Heng Huang. Multi-view clustering and feature learning via structured sparsity. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 352–360, 2013a.

Jialei Wang, Peilin Zhao, Steven CH Hoi, and Rong Jin. Online feature selection and its applications. *Knowledge and Data Engineering, IEEE Transactions on*, 26(3):698–710, 2014.

Jing Wang, Zhong-Qiu Zhao, Xuegang Hu, Yiu-Ming Cheung, Meng Wang, and Xindong Wu. Online group feature selection. In *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*, pages 1757–1763. AAAI Press, 2013b.

Jing Wang, Meng Wang, Peipei Li, Luoqi Liu, Zhongqiu Zhao, Xuegang Hu, and Xindong Wu. Online feature selection with group structure analysis. *IEEE Transactions on Knowledge and Data Engineering*, 27(11):3029–3041, 2015.

Sewall Wright. The interpretation of population structure by f-statistics with special regard to systems of mating. *Evolution*, pages 395–420, 1965.

Xindong Wu, Kui Yu, Hao Wang, and Wei Ding. Online streaming feature selection. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 1159–1166, 2010.

Xindong Wu, Kui Yu, Wei Ding, Hao Wang, and Xingquan Zhu. Online feature selection with streaming features. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(5):1178–1192, 2013.

Makoto Yamada, Avishek Saha, Hua Ouyang, Dawei Yin, and Yi Chang. N3lars: Minimum redundancy maximum relevance feature selection for large and high-dimensional data. *arXiv preprint arXiv:1411.2331*, 2014.

Howard Hua Yang and John E Moody. Data visualization and feature selection: New algorithms for nongaussian data. In *NIPS*, volume 99, pages 687–693. Citeseer, 1999.

Sen Yang, Lei Yuan, Ying-Cheng Lai, Xiaotong Shen, Peter Wonka, and Jieping Ye. Feature grouping and selection over an undirected graph. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 922–930. ACM, 2012.

Yi Yang, Dong Xu, Feiping Nie, Shuicheng Yan, and Yueting Zhuang. Image clustering using local discriminant models and global integration. *Image Processing, IEEE Transactions on*, 19(10):2761–2773, 2010.

Yi Yang, Heng Tao Shen, Zhigang Ma, Zi Huang, and Xiaofang Zhou. l2, 1-norm regularized discriminative feature selection for unsupervised learning. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, pages 1589–1594. Citeseer, 2011.

Jieping Ye and Jun Liu. Sparse methods for biomedical data. *ACM SIGKDD Explorations Newsletter*, 14(1):4–15, 2012.

Lei Yu and Huan Liu. Feature selection for high-dimensional data: A fast correlation-based filter solution. In *ICML*, volume 3, pages 856–863, 2003.

Stella X Yu and Jianbo Shi. Multiclass spectral clustering. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 313–319. IEEE, 2003.

Lei Yuan, Jun Liu, and Jieping Ye. Efficient methods for overlapping group lasso. In *Advances in Neural Information Processing Systems*, pages 352–360, 2011.

Ming Yuan and Yi Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67, 2006.

Jian Zhang, Zoubin Ghahramani, and Yiming Yang. Flexible latent variable models for multi-task learning. *Machine Learning*, 73(3):221–242, 2008.

Qin Zhang, Peng Zhang, Guodong Long, Wei Ding, Chengqi Zhang, and Xindong Wu. Towards mining trapezoidal data streams. In *Data Mining (ICDM), 2015 IEEE 15th International Conference on*. IEEE, 2015.

Peng Zhao and Bin Yu. On model selection consistency of lasso. *The Journal of Machine Learning Research*, 7:2541–2563, 2006.

Peng Zhao, Guilherme Rocha, and Bin Yu. The composite absolute penalties family for grouped and hierarchical variable selection. *The Annals of Statistics*, pages 3468–3497, 2009.

Zheng Zhao and Huan Liu. Spectral feature selection for supervised and unsupervised learning. In *Proceedings of the 24th international conference on Machine learning*, pages 1151–1157. ACM, 2007.

Zheng Zhao and Huan Liu. Multi-source feature selection via geometry-dependent covariance analysis. In *FSDM*, pages 36–47, 2008.

Zheng Zhao, Ruiwen Zhang, James Cox, David Duling, and Warren Sarle. Massively parallel feature selection: an approach based on variance preservation. *Machine learning*, 92(1): 195–220, 2013.

Dengyong Zhou, Jiayuan Huang, and Bernhard Schölkopf. Learning from labeled and unlabeled data on a directed graph. In *Proceedings of the 22nd international conference on Machine learning*, pages 1036–1043. ACM, 2005a.

Jiayu Zhou, Jun Liu, Vaibhav A Narayan, and Jieping Ye. Modeling disease progression via fused sparse group lasso. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1095–1103. ACM, 2012.

Jing Zhou, Dean Foster, Robert Stine, and Lyle Ungar. Streaming feature selection using alpha-investing. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 384–393. ACM, 2005b.

Hui Zou. The adaptive lasso and its oracle properties. *Journal of the American statistical association*, 101(476):1418–1429, 2006.

Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005.