Feature Selection: A Data Perspective

Jundong Li, Arizona State University
Kewei Cheng, Arizona State University
Suhang Wang, Arizona State University
Fred Morstatter, Arizona State University
Robert P. Trevino, Arizona State University
Jiliang Tang, Michigan State University
Huan Liu, Arizona State University

Feature selection, as a data preprocessing strategy, has been proven to be effective and efficient in preparing data (especially high-dimensional data) for various 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 to feature selection. 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 era of big data, 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 categorize them into four main groups: similarity based, information theoretical based, sparse learning based and statistical based methods. To facilitate and promote the research in this community, we also present an open-source feature selection repository that consists of most of the popular feature selection algorithms (http://featureselection.asu.edu/). Also, we use it as an example to show how to evaluate feature selection algorithms. At the end of the survey, we present a discussion about some open problems and challenges that require more attention in future research.

1. INTRODUCTION

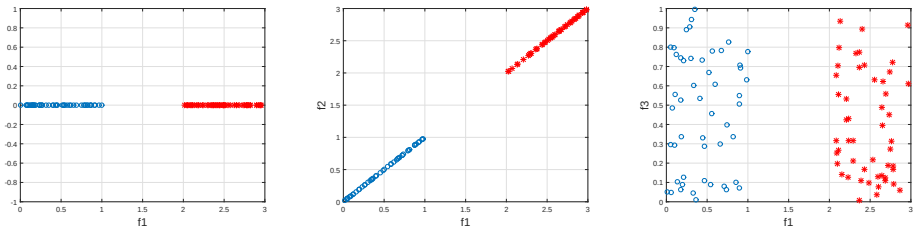
We are now in the era of big data, where huge amounts of high-dimensional data become ubiquitous in a variety of domains, such as social media, healthcare, bioinformatics and online education. The rapid growth of data presents challenges for effective and efficient data management. It is desirable to apply data mining and machine learning techniques to automatically discover knowledge from data of various sorts.

When data mining and machine learning algorithms are applied on high-dimensional data, a critical issue is known as the curse of dimensionality. It refers to the phenomenon that data becomes sparser in high-dimensional space, adversely affecting algorithms designed for low-dimensional space [Hastie et al. 2005]. Also, with a large number of features, learning models tend to overfit which may cause performance degradation on unseen data. Data of high dimensionality can significantly increase 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 mainly categorized into two main components: feature extraction and feature selection. Feature extraction projects the original high-dimensional features to a new feature space with low dimensionality. The newly constructed feature space is usually a linear or nonlinear combination of the original features. Feature selection, on the other hand, directly selects a subset of relevant features for model construction [Guyon and Elisseeff 2003; Liu and Motoda 2007].

Both feature extraction and feature selection have the advantages of improving learning performance, increasing computational efficiency, decreasing memory storage, and building better generalization models. Hence, they are both regarded as effective dimensionality reduction techniques. On one hand, for many applications where the raw input data does not contain any features understandable to a given learning algorithm, feature extraction is preferred. On the other hand, as feature extraction creates a set of new features, further analysis is problematic as we cannot retain

the physical meanings of these features. In contrast, by keeping some of the original features, feature selection maintains physical meanings of the original features and gives models better readability and interpretability. Therefore, feature selection is often preferred in many applications such as text mining and genetic analysis. It should be noted that in some cases even though feature dimensionality is often not that high, feature extraction/selection still plays an essential role such as improving learning performance, preventing overfitting, and reducing computational costs.



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

Fig. 1: An illustrative example of relevant, redundant and irrelevant features.

Real-world data contains a lot of irrelevant, redundant and noisy features. Removing these features by feature selection reduces storage and computational cost while avoiding significant loss of information or degradation of learning performance. For example, in Fig. 1(a), feature f_1 is a relevant feature that is able to discriminate two classes (clusters). However, given feature f_1, feature f_2 in Fig. 1(b) is redundant as f_2 is strongly correlated with f_1. In Fig. 1(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.

1.1. Traditional Categorization of Feature Selection Algorithms

1.1.1. Supervision Perspective. According to the availability of supervision (such as class labels in classification problems), feature selection can be broadly classified as supervised, unsupervised and semi-supervised methods.

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 (classification) or to approximate the regression targets (regression). With supervision information, feature relevance is usually assessed via its correlation with the class labels or the regression target. The training phase highly depends on the selected features: after splitting the data into training and testing sets, classifiers or regression models are trained based on a subset of features selected by supervised feature selection. Note that the feature selection phase can be independent of the learning algorithms (filter methods); or it may iteratively take advantage of the learning performance of a classifier or a regression model to assess the quality of selected features so far (wrapper methods); or make use of the intrinsic structure of a learning algorithm to embed feature selection into the underlying model (embedded methods). Finally, the trained classifier or regression model predicts class labels or regression targets of unseen samples in the test set with the selected features. In the following context, for supervised methods, we mainly focus on classification problems, and use label information, supervision information interchangeably.

Unsupervised feature selection is generally designed for clustering problems. As acquiring labeled data is particularly expensive in both time and efforts, unsupervised feature selection has gained considerable attention recently. Without label information to evaluate the importance of features, unsupervised feature selection methods seek

alternative criteria to define feature relevance. Different from supervised feature selection, unsupervised feature selection usually uses all instances that are available in the feature selection phase. The feature selection phase can be independent of the unsupervised learning algorithms (filter methods); or it relies on the learning algorithms to iteratively improve the quality of selected features (wrapper methods); or embed the feature selection phase into unsupervised learning algorithms (embedded methods). After the feature selection phase, it outputs the cluster structure of all data samples on the selected features by using a standard clustering algorithm [Guyon and Elisseeff 2003; Liu and Motoda 2007; Tang et al. 2014a].

Supervised feature selection works when sufficient label information is available while unsupervised feature selection algorithms do not require any class labels. However, in many real-world applications, we usually have a limited number of labeled data. Therefore, it is desirable to develop semi-supervised methods by exploiting both labeled and unlabeled data samples.

1.1.2. Selection Strategy Perspective. Concerning different selection strategies, feature selection methods can be broadly categorized as wrapper, filter and embedded 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) search for a subset of features; and (2) evaluate the selected features. It repeats (1) and (2) until some stopping criteria are satisfied. 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. For example, the whole process works iteratively until such as the highest learning performance is achieved or the desired number of selected features is obtained. Then the feature subset that gives the highest learning performance is returned as the selected features. Unfortunately, a known issue of wrapper methods is that the search space for d features is 2^d, which is impractical when d is very large. Therefore, many different search strategies such as sequential search [Guyon and Elisseeff 2003], hill-climbing search, best-first search [Kohavi and John 1997; Arai et al. 2016], branch-and-bound search [Narendra and Fukunaga 1977] and genetic algorithms [Golberg 1989] are proposed to yield a local optimum learning performance. However, the search space is still extremely huge for high-dimensional datasets. As a result, wrapper methods are seldom used in practice.

Filter methods are independent of any learning algorithms. They rely on characteristics of data to assess feature importance. Filter methods are typically more computationally 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 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, lowly ranked features are filtered out. 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 1992; Robnik-Šikonja and Kononenko 2003; Yang et al. 2011; Du et al. 2013; Tang et al. 2014b], feature correlation [Koller and Sahami 1995; Guyon and Elisseeff 2003], mutual information [Yu and Liu 2003; Peng et al. 2005; Nguyen et al. 2014; Shishkin et al. 2016; Gao et al. 2016], feature ability to preserve data manifold structure [He et al. 2005; Zhao and Liu 2007; Gu et al. 2011b; Jiang and Ren 2011], and feature ability to reconstruct the original data [Masaeli et al. 2010; Farahat et al. 2011; Li et al. 2017a].

Embedded methods is a trade-off between filter and wrapper methods which embed the feature selection into 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 target to fit a learning model by minimizing the fitting errors and forcing feature coefficients to be small (or exact zero) simultaneously. Afterwards, both the regularization model and selected feature sets are returned as the final results.

It should be noted that some literature classifies feature selection methods into four categories (from the selection strategy perspective) by including the hybrid feature selection methods [Saeys et al. 2007; Shen et al. 2012; Ang et al. 2016]. Hybrid methods can be regarded as a combination of multiple feature selection algorithms (e.g., wrapper, filter, and embedded). The main target is to tackle the instability and perturbation issues of many existing feature selection algorithms. For example, for small-sized high-dimensional data, a small perturbation on the training data may result in totally different feature selection results. By aggregating multiple selected feature subsets from different methods together, the results are more robust and hence the credibility of the selected features is enhanced.

1.2. Feature Selection Algorithms from a Data Perspective

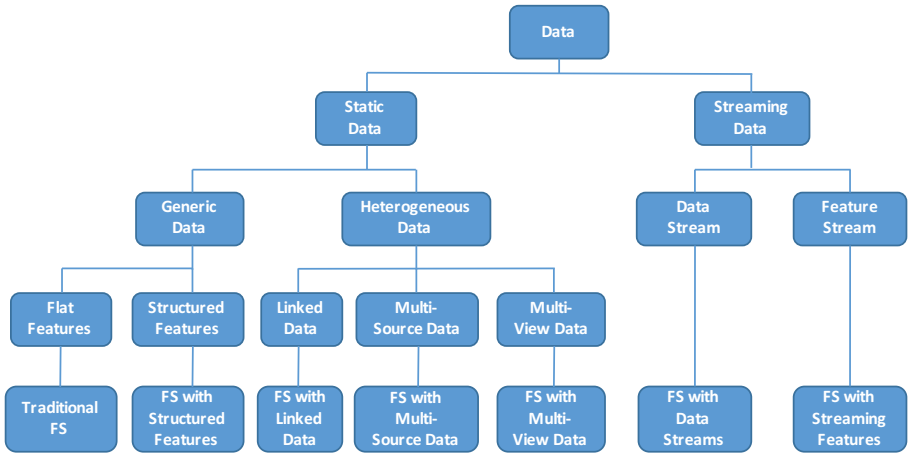


Fig. 2: Feature selection algorithms from the data perspective.

The recent popularity of big data presents unique challenges for traditional feature selection [Li and Liu 2017], and some characteristics of big data such as velocity and variety necessitate the development of novel feature selection algorithms. Here we briefly discuss some major concerns when applying feature selection algorithms.

Streaming data and features have become more and more prevalent in real-world applications. It poses challenges to traditional feature selection algorithms, which are designed for static datasets with fixed data samples and features. For example in Twitter, new data like posts and new features like slang words are continuously being user-generated. It is impractical to apply traditional batch-mode feature selection algorithms to find relevant features from scratch when new data or new feature arrives. Moreover, the volume of data may be too large to be loaded into memory. In many

cases, a single scan of data is desired as further scans is either expensive or impractical. Given the reasons mentioned above, it is appealing to apply feature selection in a streaming fashion to dynamically maintain a set of relevant features.

Most existing algorithms of feature selection are designed to handle tasks with a single data source and always assume that data is independent and identically distributed (*i.i.d.*). However, data could come from multiple sources in many applications. For example, in social media, data comes from heterogeneous sources such as text, images, tags, videos. In addition, linked data is ubiquitous and presents 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. For instance, with link information, the widely adopted *i.i.d.* assumption in most learning algorithms does not hold. How to appropriately utilize link information for feature selection is still a challenging problem.

Features can also exhibit certain types of structures. Some well-known structures among features are group, tree, and graph structures. When performing feature selection, if the feature structure is not taken into consideration, the intrinsic dependencies may not be captured, thus the selected features may not be suitable for the target application. Incorporating prior knowledge of feature structures can help select relevant features to improve the learning performance greatly.

The aforementioned reasons motivate the investigation of feature selection algorithms from a different view. In this survey, we revisit feature selection algorithms from a data perspective; the categorization is illustrated in Fig. 2. It is shown that data consists of static data and streaming data. For the static data, it can be grouped into generic data and heterogeneous data. In generic data, features can either be flat or possess some inherent 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 main groups, i.e., similarity based, information theoretical based, sparse learning based and statistical based methods. 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 are usually used in specific applications. More details about these four categories will be presented later. We present other methods that cannot be fitted into these four categories, such as hybrid methods, deep learning based methods and reconstruction based methods. When features express some structures, specific feature selection algorithms are more desired. Data can be heterogeneous such that data could come from multiple sources and could be linked. Hence, we also show how new feature selection algorithms cope with these situations. Second, in the streaming settings, data arrives 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 accordingly. Similarly, in an orthogonal setting, features can also be generated dynamically. Streaming feature selection algorithms are designed to determine if one should accept the newly added features and remove existing but outdated features.

1.3. Differences with Existing Surveys

Currently, there exist some surveys about feature selection algorithms [Guyon and Elisseeff 2003; Alelyani et al. 2013; Chandrashekar and Sahin 2014; Tang et al. 2014a]. These studies either focus on traditional feature selection algorithms or specific learning tasks 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 Fig. 2. We also release a feature selection repository in Python named *scikit-feature* which is built upon the widely used machine learning package *scikit-learn* (http://scikit-learn.org/stable/) and two scientific computing packages *Numpy* (http://www.numpy.org/) and *Scipy* (http://www.scipy.org/). It includes near 40 representative feature selection algorithms. The web page of the repository is available at http://featureselection.asu.edu/.

1.4. Organization of the Survey

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

(1) Traditional Feature Selection for 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
  (e) Other Methods
(2) Feature Selection with Structured 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
(3) Feature Selection with Heterogeneous Data (Section 4)
  (a) Feature Selection Algorithms with Linked Data
  (b) Multi-Source Feature Selection
  (c) Multi-View Feature Selection
(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)

1.5. Notations

We summarize some symbols used throughout this survey in Table I. 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 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's.

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 as similarity based, information theoretical based, sparse learning based and statistical based methods, and other methods according to the used techniques.

2.1. Similarity based Methods

Different feature selection algorithms exploit various types of criteria to define the relevance of features. Among them, there is a family of methods assessing feature

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 original 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
$\mathbf{X}_{\mathcal{S}} \in \mathbb{R}^{n \times k}$	data matrix on the selected k features
$\mathbf{y} \in \mathbb{R}^n$	class label vector for all n instances (if exist)

Table I: Symbols.

importance by their ability to preserve data similarity. We refer them as similarity based 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.

Given a dataset $\mathbf{X} \in \mathbb{R}^{n \times d}$ with n instances and d features, pairwise similarity among instances can be encoded in an affinity matrix $\mathbf{S} \in \mathbb{R}^{n \times n}$. Suppose that we want to select k most relevant features $\mathcal{S}$, one way is to maximize their utility: $\max_{\mathcal{S}} SC(\mathcal{S})$, where $SC(\mathcal{S})$ denotes the utility of the feature subset $\mathcal{S}$. As algorithms in this family often evaluate features individually, the utility maximization over feature subset $\mathcal{S}$ can be further decomposed into the following form:

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

where $SC(f)$ is a utility function for feature f. $\hat{\mathbf{f}}$ denotes the transformation (e.g., scaling, normalization, etc) result of the original feature vector $\mathbf{f}$. $\hat{\mathbf{S}}$ is a new affinity matrix obtained from affinity matrix $\mathbf{S}$. The maximization problem in Eq. (1) shows that we would select a subset of features from $\mathcal{S}$ such that they can well preserve the data manifold structure encoded in $\hat{\mathbf{S}}$. This problem is usually solved by greedily selecting the top k features that maximize their individual utility. Methods in this category vary in the way the affinity matrix $\hat{\mathbf{S}}$ is designed. We subsequently discuss some representative algorithms in this group that can be reformulated under the unified utility maximization framework.

2.1.1. Laplacian Score. Laplacian Score [He et al. 2005] is an unsupervised feature selection algorithm which selects features that can best preserve the data manifold structure. It consists of three phases. First, it constructs the affinity matrix such that $\mathbf{S}(i,j) = e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{t}}$ if x_i is among the p-nearest neighbor of x_j; otherwise $\mathbf{S}(i,j) = 0$. Then, 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 $\mathbf{L} = \mathbf{D} - \mathbf{S}$. Lastly, the Laplacian Score of each feature f_i is computed as:

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

As Laplacian Score evaluates 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:

$$laplacian\_score(f_i) = 1 - \left(\frac{\tilde{\mathbf{f}}_i}{\|\mathbf{D}^{\frac{1}{2}}\tilde{\mathbf{f}}_i\|_2}\right)' \mathbf{S} \left(\frac{\tilde{\mathbf{f}}_i}{\|\mathbf{D}^{\frac{1}{2}}\tilde{\mathbf{f}}_i\|_2}\right), \tag{3}$$

where $\|\mathbf{D}^{\frac{1}{2}}\tilde{\mathbf{f}}_i\|_2$ is the standard data variance of feature f_i, and the term $\tilde{\mathbf{f}}_i/\|\mathbf{D}^{\frac{1}{2}}\tilde{\mathbf{f}}_i\|_2$ is interpreted as a normalized feature vector of $\mathbf{f}_i$. Therefore, it is obvious that Laplacian Score is a special case of utility maximization in Eq. (1).

2.1.2. SPEC. SPEC [Zhao and Liu 2007] is an extension of Laplacian Score that works for both supervised and unsupervised scenarios. For example, in the unsupervised scenario, the data similarity is measured by RBF kernel; while in the supervised scenario, data similarity can be defined by: $\mathbf{S}(i,j) = \begin{cases} \frac{1}{n_l} & \text{if } y_i = y_j = l \\ 0 & \text{otherwise} \end{cases}$, where n_l is the number of data samples in the l-th class. After obtaining the affinity matrix $\mathbf{S}$ and the diagonal matrix $\mathbf{D}$, the normalized Laplacian matrix $\mathbf{L}_{norm} = \mathbf{D}^{-\frac{1}{2}}(\mathbf{D} - \mathbf{S})\mathbf{D}^{-\frac{1}{2}}$. The basic idea of SPEC is similar to Laplacian Score: a feature that is consistent with the data manifold structure should assign similar values to instances that are near each other. In SPEC, the feature relevance is measured by three different criteria:

$$\begin{aligned} SPEC\_score1(f_i) &= \hat{\mathbf{f}}_i{}'\gamma(\mathbf{L}_{norm})\hat{\mathbf{f}}_i = \sum_{j=1}^{n} \alpha_j^2 \gamma(\lambda_j) \\ SPEC\_score2(f_i) &= \frac{\hat{\mathbf{f}}_i{}'\gamma(\mathbf{L}_{norm})\hat{\mathbf{f}}_i}{1 - (\hat{\mathbf{f}}_i{}'\xi_1)^2} = \frac{\sum_{j=2}^{n} \alpha_j^2 \gamma(\lambda_j)}{\sum_{j=2}^{n} \alpha_j^2} \\ SPEC\_score3(f_i) &= \sum_{j=1}^{m} (\gamma(2) - \gamma(\lambda_j))\alpha_j^2. \end{aligned} \tag{4}$$

In the above equations, $\hat{\mathbf{f}}_i = \mathbf{D}^{\frac{1}{2}}\mathbf{f}_i/\|\mathbf{D}^{\frac{1}{2}}\mathbf{f}_i\|_2$; (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 criterion $SPEC\_score2(f_i)$ is used, SPEC is equivalent to the Laplacian Score. For $SPEC\_score3(f_i)$, it uses the top m eigenpairs to evaluate the importance of feature f_i.

All these three criteria can be reduced to the the unified similarity based feature selection framework in Eq. (1) by setting $\hat{\mathbf{f}}_i$ as $\mathbf{f}_i\|\mathbf{D}^{\frac{1}{2}}\mathbf{f}_i\|_2$, $(\mathbf{f}_i - \mu\mathbf{1})/\|\mathbf{D}^{\frac{1}{2}}\mathbf{f}_i\|_2$, $\mathbf{f}_i\|\mathbf{D}^{\frac{1}{2}}\mathbf{f}_i\|_2$; and $\hat{\mathbf{S}}$ as $\mathbf{D}^{\frac{1}{2}}\mathbf{U}(\mathbf{I} - \gamma(\mathbf{\Sigma}))\mathbf{U}'\mathbf{D}^{\frac{1}{2}}$, $\mathbf{D}^{\frac{1}{2}}\mathbf{U}(\mathbf{I} - \gamma(\mathbf{\Sigma}))\mathbf{U}'\mathbf{D}^{\frac{1}{2}}$, $\mathbf{D}^{\frac{1}{2}}\mathbf{U}_m(\gamma(2\mathbf{I}) - \gamma(\mathbf{\Sigma}_m))\mathbf{U}'_m\mathbf{D}^{\frac{1}{2}}$ in SPEC_score1, SPEC_score2, SPEC_score3, respectively. $\mathbf{U}$ and $\mathbf{\Sigma}$ are the singular vectors and singular values of the normalized Laplacian matrix $\mathbf{L}_{norm}$.

2.1.3. Fisher Score. Fisher Score [Duda et al. 2012] is a supervised feature selection algorithm. It selects features such that the feature values of samples within the same class are similar while the feature values of samples from different classes are dissimilar. The Fisher Score of each feature f_i is evaluated as follows:

$$fisher\_score(f_i) = \frac{\sum_{j=1}^{c} n_j(\mu_{ij} - \mu_i)^2}{\sum_{j=1}^{c} n_j \sigma_{ij}^2}, \tag{5}$$

where n_j, μ_i, μ_{ij} and σ_{ij}^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 $\mathbf{S}(i,j) = \begin{cases} \frac{1}{n_l} & \text{if } y_i = y_j = l \\ 0 & \text{otherwise,} \end{cases}$. In this way, the relationship between Fisher Score and Laplacian Score is $fisher\_score(f_i) = 1 - \frac{1}{laplacian\_score(f_i)}$. Hence, the computation of Fisher Score can also be reduced to the unified utility maximization framework.

2.1.4. Trace Ratio Criterion. The trace ratio criterion [Nie et al. 2008] directly selects the global optimal feature subset based on the corresponding score, which is computed by a trace ratio norm. It builds two affinity matrices $\mathbf{S}_w$ and $\mathbf{S}_b$ to characterize within-class and between-class data similarity. 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 score of the selected k features in $\mathcal{S}$ is:

$$trace\_ratio(\mathcal{S}) = \frac{tr(\mathbf{W}'\mathbf{X}'\mathbf{L}_b\mathbf{X}\mathbf{W})}{tr(\mathbf{W}'\mathbf{X}'\mathbf{L}_w\mathbf{X}\mathbf{W})}, \tag{6}$$

where $\mathbf{L}_b$ and $\mathbf{L}_w$ are Laplacian matrices of $\mathbf{S}_a$ and $\mathbf{S}_b$ respectively. The basic idea is to maximize the data similarity for instances from the same class while minimize the data similarity for instances from different classes. However, the trace ratio problem is difficult to solve as it does not have a closed-form solution. Hence, the trace ratio problem is often converted into a more tractable format called the ratio trace problem by maximizing $tr[(\mathbf{W}'\mathbf{X}'\mathbf{L}_w\mathbf{X}\mathbf{W})^{-1}(\mathbf{W}'\mathbf{X}'\mathbf{L}_b\mathbf{X}\mathbf{W})]$. As an alternative, [Wang et al. 2007] propose an iterative algorithm called ITR to solve the trace ratio problem directly and was later applied in trace ratio feature selection [Nie et al. 2008].

Different $\mathbf{S}_b$ and $\mathbf{S}_w$ lead to different feature selection algorithms such as batch-mode Lalpacian Score and batch-mode Fisher Score. For example, in batch-mode Fisher Score, the within-class data similarity and the between-class data similarity are $\mathbf{S}_w(i,j) = \begin{cases} 1/n_l & \text{if } y_i = y_j = l \\ 0 & \text{otherwise} \end{cases}$ and $\mathbf{S}_b(i,j) = \begin{cases} 1/n - 1/n_l & \text{if } y_i = y_j = l \\ 1/n & \text{otherwise} \end{cases}$ respectively. Therefore, maximizing the trace ratio criterion is equivalent to maximizing $\frac{\sum_{s=1}^{k} \mathbf{f}'_{i_s}\mathbf{S}_w\mathbf{f}_{i_s}}{\sum_{s=1}^{k} \mathbf{f}'_{i_s}\mathbf{f}_{i_s}} = \frac{\mathbf{X}'_{\mathcal{S}}\mathbf{S}_w\mathbf{X}_{\mathcal{S}}}{\mathbf{X}'_{\mathcal{S}}\mathbf{X}_{\mathcal{S}}}$. Since $\mathbf{X}'_{\mathcal{S}}\mathbf{X}_{\mathcal{S}}$ is constant, it can be further reduced to the unified similarity based feature selection framework by setting $\hat{\mathbf{f}} = \mathbf{f}/\|\mathbf{f}\|_2$ and $\hat{\mathbf{S}} = \mathbf{S}_w$. On the other hand in batch-mode Laplacian Score, the within-class data similarity and the between-class data similarity are $\mathbf{S}_w(i,j) = \begin{cases} e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{t}} & \text{if } \mathbf{x}_i \in \mathcal{N}_p(\mathbf{x}_j) \text{ or } \mathbf{x}_j \in \mathcal{N}_p(\mathbf{x}_i) \\ 0 & \text{otherwise} \end{cases}$ and $\mathbf{S}_b = (\mathbf{1}'\mathbf{D}_w\mathbf{1})^{-1}\mathbf{D}_w\mathbf{1}\mathbf{1}'\mathbf{D}_w$ respectively. In this case, the trace ratio criterion score is $\frac{tr(\mathbf{W}'\mathbf{X}'\mathbf{L}_b\mathbf{X}\mathbf{W})}{tr(\mathbf{W}'\mathbf{X}'\mathbf{L}_w\mathbf{X}\mathbf{W})} = \frac{\sum_{s=1}^{k} \mathbf{f}'_{i_s}\mathbf{D}_w\mathbf{f}_{i_s}}{\sum_{s=1}^{k} \mathbf{f}'_{i_s}(\mathbf{D}_w - \mathbf{S}_w)\mathbf{f}_{i_s}}$. Therefore, maximizing the trace ratio criterion is also equivalent to solving the unified maximization problem in Eq. (1) where $\hat{\mathbf{f}} = \mathbf{f}/\|\mathbf{D}_w^{\frac{1}{2}}\mathbf{f}\|_2$ and $\hat{\mathbf{S}} = \mathbf{S}_w$.

2.1.5. ReliefF. ReliefF [Robnik-Šikonja and Kononenko 2003] selects 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 ReliefF is defined as follows:

$$\begin{aligned} ReliefF\_score(f_i) = & \frac{1}{c}\sum_{j=1}^{l}(-\frac{1}{m_j}\sum_{x_r \in NH(j)} d(\mathbf{X}(j,i) - \mathbf{X}(r,i)) \\ & + \sum_{y \neq y_j}\frac{1}{h_{jy}}\frac{p(y)}{1-p(y)}\sum_{x_r \in NM(j,y)} d(\mathbf{X}(j,i) - \mathbf{X}(r,i))), \end{aligned} \tag{7}$$

where $NH(j)$ and $NM(j,y)$ are the nearest instances of x_j in the same class and in class y, respectively. Their sizes are m_j and h_{jy}, respectively. $p(y)$ is the ratio of instances in class y.

ReliefF is equivalent to selecting features that preserve a special form of data similarity matrix. Assume that the dataset has the same number of instances in each of the c classes and there are q instances in both $NM(j)$ and $NH(j,y)$. Then according to [Zhao and Liu 2007], the ReliefF feature selection can be reduced to the utility maximization framework in Eq. (1).

Discussion*: Similarity based feature selection algorithms have demonstrated with excellent performance in both supervised and unsupervised learning problems. This category of methods is straightforward and simple as the computation focuses on building an affinity matrix, and afterwards, the scores of features can be obtained. Also, these methods are independent of any learning algorithms and the selected features are suitable for many subsequent learning tasks. However, one drawback of these methods is that they cannot handle feature redundancy. In other words, they may repeatedly find highly correlated features during the selection phase.*

2.2. Information Theoretical based Methods

A large family of existing feature selection algorithms is information theoretical based methods. Algorithms in this family 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 to discrete variables. Therefore, feature selection algorithms in this family can only work with discrete data. For continuous feature values, some data discretization techniques are required beforehand. Two decades of research on information theoretic criteria can be unified in a conditional likelihood maximization framework [Brown et al. 2012]. In this subsection, we introduce some representative algorithms in this family. We first give a brief introduction about basic information theoretic concepts.

The concept of *entropy* measures the uncertainty of a discrete random variable. The entropy of a discrete random variable X is defined as follows:

$$H(X) = -\sum_{x_i \in X} P(x_i) log(P(x_i)), \tag{8}$$

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.

Second, the *conditional entropy* of X given another discrete random variable Y is:

$$H(X|Y) = \sum_{y_j \in Y} P(y_j) \sum_{x_i \in X} P(x_i|y_j) log(P(x_i|y_j)), \tag{9}$$

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. It shows the uncertainty of X given Y.

Then, *information gain* or *mutual information* between X and Y is used to measure the amount of information shared by X and Y together:

$$I(X;Y) = H(X) - H(X|Y) = \sum_{x_i \in X} \sum_{y_j \in Y} P(x_i, y_j) log \frac{P(x_i, y_j)}{P(x_i)P(y_j)}, \tag{10}$$

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

At last, *conditional information gain* (or *conditional mutual information*) of discrete variables X and Y given a third discrete variable Z is given as follows:

$$I(X;Y|Z) = H(X|Z) - H(X|Y,Z) = \sum_{z_k \in Z} P(z_k) \sum_{x_i \in X} \sum_{y_j \in Y} P(x_i, y_j|z_k) log \frac{P(x_i, y_j|z_k)}{P(x_i|z_k)P(y_j|z_k)}. \quad (11)$$

It shows the amount of mutual information shared by X and Y given Z.

Searching for the global best set of features is NP-hard, thus most algorithms 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:

$$J_{\text{CMI}}(X_k) = I(X_k;Y) + \sum_{X_j \in \mathcal{S}} g[I(X_j;X_k), I(X_j;X_k|Y)], \quad (12)$$

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 linear function w.r.t. these two variables, it is referred as a criterion by linear combinations of Shannon information terms such that:

$$J_{\text{CMI}}(X_k) = I(X_k;Y) - \beta \sum_{X_j \in \mathcal{S}} I(X_j;X_k) + \lambda \sum_{X_j \in \mathcal{S}} I(X_j;X_k|Y). \quad (13)$$

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

2.2.1. Mutual Information Maximization (Information Gain). Mutual Information Maximization (MIM) (a.k.a. Information Gain) [Lewis 1992] measures the importance of a feature by its correlation with class labels. 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 feature X_k is:

$$J_{\text{MIM}}(X_k) = I(X_k;Y). \quad (14)$$

It can be observed that in MIM, the scores of features are assessed individually. Therefore, only the feature correlation is considered while the feature redundancy is completely ignored. After it obtains the MIM feature scores for all features, we choose the features with the highest feature scores and add them to the selected feature set. The process repeats until the desired number of selected features is obtained.

It can also be observed that MIM is a special case of linear combination of Shannon information terms in Eq. (13) where both β and λ are equal to zero.

2.2.2. Mutual Information Feature Selection. A limitation of MIM criterion is that it assumes that features are independent of each other. 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) [Battiti 1994] 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:

$$J_{\text{MIFS}}(X_k) = I(X_k;Y) - \beta \sum_{X_j \in \mathcal{S}} I(X_k;X_j). \quad (15)$$

In MIFS, the feature relevance is evaluated by $I(X_k;Y)$, while the second term penalizes features that have a high mutual information with the currently selected features such that feature redundancy is minimized.

MIFS can also be reduced to be a special case of the linear combination of Shannon information terms in Eq. (13) where β is between zero and one, and λ is zero.

2.2.3. Minimum Redundancy Maximum Relevance. [Peng et al. 2005] proposes a Minimum Redundancy Maximum Relevance (MRMR) criterion to set the value of β to be the reverse of the number of selected features:

$$J_{\text{MRMR}}(X_k) = I(X_k; Y) - \frac{1}{|\mathcal{S}|} \sum_{X_j \in \mathcal{S}} I(X_k; X_j). \tag{16}$$

Hence, with more selected features, the effect of feature redundancy is gradually reduced. The intuition is that with more non-redundant features selected, it becomes 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.

MRMR is also strongly linked to the Conditional likelihood maximization framework 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. 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 also 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) [Lin and Tang 2006], in which the feature score for a new unselected feature X_k is:

$$J_{\text{CIFE}}(X_k) = I(X_k; Y) - \sum_{X_j \in \mathcal{S}} I(X_j; X_k) + \sum_{X_j \in \mathcal{S}} I(X_j; X_k | Y). \tag{17}$$

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. Also, CIFE is a special case of the linear combination of Shannon information terms by setting both β and γ to be 1.

2.2.5. Joint Mutual Information. 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] is proposed to increase the complementary information that is shared between unselected features and selected features given the class labels. The feature selection criterion is listed as follows:

$$J_{\text{JMI}}(X_k) = \sum_{X_j \in \mathcal{S}} I(X_k, X_j; Y). \tag{18}$$

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

JMI cannot 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:

$$J_{\text{JMI}}(X_k) = I(X_k; Y) - \frac{1}{|\mathcal{S}|} \sum_{X_j \in \mathcal{S}} I(X_j; X_k) + \frac{1}{|\mathcal{S}|} \sum_{X_j \in \mathcal{S}} I(X_j; X_k | Y). \tag{19}$$

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. Previously mentioned criteria could be reduced to a linear combination of Shannon information terms. Next, we show some other algorithms that can only be reduced to a non-linear combination of Shannon information terms. Among them, Conditional Mutual Information Maximization (CMIM) [Vidal-Naquet and Ullman 2003; Fleuret 2004] iteratively selects features which maximize the mutual information with the class labels given the selected features so far. Mathematically, during the selection phase, the feature score for each new unselected feature X_k can be formulated as follows:

$$J_{\text{CMIM}}(X_k) = \min_{X_j \in \mathcal{S}} [I(X_k; Y|X_j)]. \tag{20}$$

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 a strong predictive ability, and it can reduce the redundancy w.r.t. the selected features.

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

$$J_{\text{CMIM}}(X_k) = I(X_k; Y) - \max_{X_j \in \mathcal{S}} [I(X_j; X_k) - I(X_j; X_k|Y)]. \tag{21}$$

Therefore, CMIM is also a special case of the conditional likelihood maximization framework in Eq. (12).

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

$$J_{\text{IF}}(X_k) = \min_{X_j \in \mathcal{S}} [I(X_j X_k; Y) - I(X_j; Y)]. \tag{22}$$

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 IF is that with the chain rule that $I(X_k X_j; Y) = I(X_j; Y) + I(X_k; Y|X_j)$, IF has the equivalent form as CMIM. Hence, it can also be reduced to the general framework in Eq. (12).

2.2.8. Interaction Capping. Interaction Capping [Jakulin 2005] is a similar feature selection criterion as CMIM in Eq. (21), it restricts the term $I(X_j; X_k) - I(X_j; X_k|Y)$ to be nonnegative:

$$J_{\text{CMIM}}(X_k) = I(X_k; Y) - \sum_{X_j \in \mathcal{S}} \max[0, I(X_j; X_k) - I(X_j; X_k|Y)]. \tag{23}$$

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. 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]:

$$J_{\text{DISR}}(X_k) = \sum_{X_j \in \mathcal{S}} \frac{I(X_j X_k; Y)}{H(X_j X_k Y)}. \tag{24}$$

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. There are other information theoretical based feature selection methods that cannot be simply reduced to the unified conditional likelihood maximization framework. Fast Correlation Based Filter (FCBF) [Yu and Liu 2003] is an example 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 are highly correlated with the class labels 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:

$$SU(X_{\mathcal{S}}, Y) = 2\frac{I(X_{\mathcal{S}}; Y)}{H(X_{\mathcal{S}}) + H(Y)}. \tag{25}$$

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 redundant features and keep the features that are most relevant to the class labels.

Discussion*: Unlike similarity based feature selection algorithms that fail to tackle feature redundancy, most aforementioned information theoretical based feature selection algorithms can be unified in a probabilistic framework that considers both "feature relevance" and "feature redundancy". Meanwhile, similar as similarity based methods, this category of methods is independent of any learning algorithms and hence are generalizable. However, most of the existing information theoretical based feature selection methods can only work in a supervised scenario. Without the guide of class labels, it is still not clear how to assess the importance of features. In addition, these methods can only handle discrete data and continuous numerical variables require discretization preprocessing beforehand.*

2.3. Sparse Learning based Methods

The third type of methods is sparse learning based methods which aim to minimize the fitting errors along with some sparse regularization terms. The sparse regularizer forces many feature coefficients to be small, or exactly zero, and then the corresponding features can be simply eliminated. Sparse learning based methods have received considerable attention in recent years due to their good performance and interpretability. In the following parts, we review some representative sparse learning based feature selection methods from both supervised and unsupervised perspectives.

2.3.1. Feature Selection with ℓ_p-norm Regularizer. First, we consider the binary classification or univariate regression problem. To achieve feature selection, the ℓ_p-norm sparsity-induced penalty term is added on the classification or regression model, where $0 \leq p \leq 1$. Let $\mathbf{w}$ denotes the feature coefficient, then the objective function for feature selection is:

$$\min_{\mathbf{w}} loss(\mathbf{w}; \mathbf{X}, \mathbf{y}) + \alpha \|\mathbf{w}\|_p, \tag{26}$$

where $loss(.)$ is a loss function, and some widely used loss functions $loss(.)$ include least squares loss, hinge loss and logistic loss. $\|\mathbf{w}\|_p = (\sum_{i=1}^{d} \|w_i\|^p)^{\frac{1}{p}}$ is a sparse regularization term, and α is a regularization parameter to balance the contribution of the loss function and the sparse regularization term for feature selection.

Typically when $p = 0$, the ℓ_0-norm regularization term directly seeks for the optimal set of nonzero entries (features) for the model learning. However, the optimization problem is naturally an integer programming problem and is difficult to solve. There-

fore, it is often relaxed to a ℓ_1-norm regularization problem, which is regarded as the tightest convex relaxation of the ℓ_0-norm. One main advantage of ℓ_1-norm regularization (LASSO) [Tibshirani 1996] is that it forces many feature coefficients to become smaller and, in some cases, exactly zero. This property makes it suitable for feature selection, as we can select features whose corresponding feature weights are large, which motivates a surge of ℓ_1-norm regularized feature selection methods [Zhu et al. 2004; Xu et al. 2014; Wei et al. 2016a; Wei and Yu 2016; Hara and Maehara 2017]. Also, the sparse vector w enables the ranking of features. Normally, the higher the value, the more important the corresponding feature is.

2.3.2. Feature Selection with $\ell_{p,q}$-norm Regularizer. Here, we discuss how to perform feature selection for the general multi-class classification or multivariate regression problems. The problem is more difficult because of the multiple classes and multivariate regression targets, and 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 same parameter sparsity patterns – each feature either has small scores or large scores for all targets. This problem can be generally solved by the $\ell_{p,q}$-norm sparsity-induced regularization term, where $p > 1$ (most existing work focus on $p = 2$ or ∞) and $0 \leq q \leq 1$ (most existing work focus on $q = 1$ or 0). Assume that $\mathbf{X}$ denotes the data matrix, and $\mathbf{Y}$ denotes the one-hot label indicator matrix. Then the model is formulated as follows:

$$\min_{\mathbf{W}} loss(\mathbf{W}; \mathbf{X}, \mathbf{y}) + \alpha \|\mathbf{W}\|_{p,q}, \tag{27}$$

where $\|\mathbf{W}\|_{p,q} = (\sum_{j=1}^{c}(\sum_{i=1}^{d} |\mathbf{W}(i,j)|^p)^{\frac{q}{p}})^{\frac{1}{q}}$; and the parameter α is used to control the contribution of the loss function and the sparsity-induced regularization term. Then the features can be ranked according to the value of $\|\mathbf{W}(i,:)\|_2^2 (i = 1, ..., d)$, the higher the value, the more important the feature is.

Case 1: $p = 2$, $q = 0$. To find relevant features across multiple targets, an intuitive way is to use discrete optimization through the $\ell_{2,0}$-norm regularization. The optimization problem with the $\ell_{2,0}$-norm regularization term can be reformulated as follows:

$$\min_{\mathbf{W}} loss(\mathbf{W}; \mathbf{X}, \mathbf{y}) \ s.t. \|\mathbf{W}\|_{2,0} \leq k. \tag{28}$$

However, solving the above optimization problem has been proven to be NP-hard, and also, due to its discrete nature, the objective function is also not convex. To solve it, a variation of Alternating Direction Method could be leveraged to seek for a local optimal solution [Cai et al. 2013; Gu et al. 2012]. In [Zhang et al. 2014], the authors provide two algorithms, proximal gradient algorithm and rank-one update algorithm to solve this discrete selection problem.

Case 2: $p = 2$, $0 < q < 1$. The above sparsity-reduced regularization term is inherently discrete and hard to solve. In [Peng and Fan 2016; Peng and Fan 2017], the authors propose a more general framework to directly optimize the sparsity-reduced regularization when $0 < q < 1$ and provided efficient iterative algorithm with guaranteed convergence rate.

Case 3: $p = 2$, $q = 1$. Although the $\ell_{2,0}$-norm is more desired for feature sparsity, however, it is inherently non-convex and non-smooth. Hence, the $\ell_{2,1}$-norm regularization is preferred and widely used in many different scenarios. Many $\ell_{2,1}$-norm regularization based feature selection methods have been proposed over the past decade [Zhao et al. 2010; Gu et al. 2011c; Yang et al. 2011; Hou et al. 2011; Li et al. 2012; Qian and Zhai 2013; Shi et al. 2014; Liu et al. 2014; Du and Shen 2015; Wang et al. 2015; Jian et al. 2016; Liu et al. 2016b; Nie et al. 2016; Zhu et al. 2016; Li et al. 2017b]. Similar to ℓ_1-norm regularization, $\ell_{2,1}$-norm regularization is also convex and a global optimal solution can be achieved [Liu et al. 2009a], thus the following discussions about the sparse learning based feature selection will center around the

$\ell_{2,1}$-norm regularization term. The $\ell_{2,1}$-norm regularization also has strong connections with group lasso [Yuan and Lin 2006] which will be explained later. By solving the related optimization problem, we can obtain a sparse matrix $\mathbf{W}$ where many rows are exact zero or of small values, and then the features corresponding to these rows can be eliminated.

Case 4: $p = \infty$, $q = 1$. In addition to the $\ell_{2,1}$-norm regularization term, the $\ell_{\infty,1}$-norm regularization is also widely used to achieve joint feature sparsity across multiple targets [Quattoni et al. 2009]. In particular, it penalizes the sum of maximum absolute values of each row, such that many rows of the matrix will all be zero.

2.3.3. Efficient and Robust Feature Selection. Authors in [Nie et al. 2010] 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 feature sparsity. The objective function of REFS is:

$$\min_{\mathbf{W}} \|\mathbf{X}\mathbf{W} - \mathbf{Y}\|_{2,1} + \alpha\|\mathbf{W}\|_{2,1}, \tag{29}$$

To solve the convex but non-smooth optimization problem, an efficient algorithm is proposed with strict convergence analysis.

It should be noted that the aforementioned REFS is designed for multi-class classification problems where each instance only has one class label. However, data could be associated with multiple labels in many domains such as information retrieval and multimedia annotation. Recently, there is a surge of research work study multi-label feature selection problems by considering label correlations. Most of them, however, are also based on the $\ell_{2,1}$-norm sparse regularization framework [Gu et al. 2011a; Chang et al. 2014; Jian et al. 2016].

2.3.4. Multi-Cluster Feature Selection. 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. Multi-Cluster Feature Selection (MCFS) [Cai et al. 2010] is one of the first attempts. Without class labels to guide the feature selection process, MCFS proposes to select features that can cover multi-cluster structure of the data where spectral analysis is used to measure the correlation between different features.

MCFS consists of three steps. In the first step, it constructs a p-nearest neighbor graph to capture the local geometric structure of data and gets the graph affinity matrix $\mathbf{S}$ and the Laplacian matrix $\mathbf{L}$. Then a flat embedding that unfolds the data manifold can be obtained by spectral clustering techniques. In the second step, since the embedding of data is known, MCFS takes advantage of them to measure the importance of features 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:

$$\min_{w_i} \|\mathbf{X}\mathbf{w}_i - \mathbf{e}_i\|_2^2 + \alpha\|\mathbf{w}_i\|_1, \tag{30}$$

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 $\boldsymbol{MCFS}(j) = \max_i |\mathbf{W}(j, i)|$. The higher the MCFS score, the more important the feature is.

2.3.5. $\ell_{2,1}$-norm Regularized Discriminative Feature Selection. 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, assume $\tilde{\mathbf{X}}$ 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, where $\mathbf{H}_n = \mathbf{I}_n - \frac{1}{n}\mathbf{1}_n\mathbf{1}_n'$. Instead of using global discriminative information, they propose to utilize the local discriminative information 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 well preserved. For each data instance x_i, it constructs a p-nearest neighbor set for that instance $\mathcal{N}_p(x_i) = \{x_{i_1}, x_{i_2}, ..., x_{i_p}\}$. Let $\mathbf{X}_{\mathcal{N}_p(i)} = [\mathbf{x}_i, \mathbf{x}_{i_1}, ..., \mathbf{x}_{i_p}]$ denotes 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 $\tilde{\mathbf{X}_i}'\tilde{\mathbf{X}_i}$ and $\tilde{\mathbf{X}_i}'\mathbf{G}_i\mathbf{G}_i'\tilde{\mathbf{X}_i}$ respectively, where $\tilde{\mathbf{X}_i}$ is the centered data matrix 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 :

$$DS_i = tr[(\mathbf{S}_t^{(i)} + \lambda\mathbf{I}_d)^{-1}\mathbf{S}_b^{(i)}] = tr[\mathbf{W}'\mathbf{X}'\mathbf{P}_{(i)}\tilde{\mathbf{X}_i}'(\tilde{\mathbf{X}_i}\tilde{\mathbf{X}_i}' + \lambda\mathbf{I}_d)^{-1}\tilde{\mathbf{X}_i}\mathbf{P}_{(i)}'\mathbf{X}\mathbf{W}], \tag{31}$$

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

$$\min_{\mathbf{W}'\mathbf{W}=\mathbf{I}_d} \sum_{i=1}^{n}\{tr[\mathbf{G}_{(i)}'\mathbf{H}_{k+1}\mathbf{G}_{(i)} - DS_i]\} + \alpha\|\mathbf{W}\|_{2,1}, \tag{32}$$

where α is a regularization parameter to control the sparsity of the learned model.

2.3.6. Feature Selection Using Nonnegative Spectral Analysis. Nonnegative Discriminative Feature Selection (NDFS) [Li et al. 2012] performs spectral clustering and feature selection simultaneously in a joint framework to select a subset of discriminative features. It 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 learned pseudo class labels are closer 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.

Similar to the 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 weighted cluster indicator matrix. It is easy to show that 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]. The local geometric structure can be preserved by minimizing the normalized graph Laplacian $tr(\mathbf{G}'\mathbf{L}\mathbf{G})$, where $\mathbf{L}$ is the Laplacian matrix that can be derived from RBF kernel. In addition to that, 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 the pseudo labels $\mathbf{G}$. These pseudo class labels are utilized as constraints to guide the feature selection

process. The combination of these two components results in the following problem:

$$\begin{aligned} \min_{\mathbf{G},\mathbf{W}} \ & tr(\mathbf{G}'\mathbf{L}\mathbf{G}) + \beta(\|\mathbf{X}\mathbf{W} - \mathbf{G}\|_F^2 + \alpha\|\mathbf{W}\|_{2,1}) \\ & \text{s.t.} \quad \mathbf{G}\mathbf{G}' = \mathbf{I_n}, \mathbf{G} \geq 0, \end{aligned} \tag{33}$$

where α is a parameter to control the sparsity of the model, and β is introduced to balance the contribution of spectral clustering and discriminative feature selection.

Discussion*: Sparse learning based feature selection methods have gained increasing popularity in recent years. A merit of such type of methods is that it embeds feature selection into a typical learning algorithm (such as linear regression, SVM, etc.). Thus it can often lead very good performance for the underlying learning algorithm. Also, with sparsity of feature weights, the model poses good interpretability as it enables us to explain why we make such prediction. Nonetheless, there are still some drawbacks of these methods: First, as it directly optimizes a particular learning algorithm by feature selection, the selected features do not necessary achieve good performance in other learning tasks. Second, this kind of methods often involves solving a nonsmooth optimization problem, and with complex matrix operations (e.g., multiplication, inverse, etc) in most cases. Hence, the expensive computational cost is another bottleneck.*

2.4. Statistical based Methods

Another category of feature selection algorithms is based on different statistical measures. As they rely on various statistical measures instead of learning algorithms to assess feature relevance, most of them are filter based methods. In addition, most statistical based algorithms analyze features individually. Hence, feature redundancy is inevitably ignored during the selection phase. We introduce some representative feature selection algorithms in this category.

2.4.1. Low Variance. Low Variance eliminates features whose variance are below a predefined threshold. For example, for the features that have the same values for all instances, the variance is 0 and should be removed since it cannot help discriminate instances from different classes. Suppose that the dataset consists of only boolean features, i.e., the feature values are either 0 and 1. As the boolean feature is a Bernoulli random variable, its variance value can be computed as:

$$variance\_score(f_i) = p(1-p), \tag{34}$$

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

2.4.2. T-score. T-score [Davis and Sampson 1986] 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 two different classes, σ_1 and σ_2 are the corresponding standard deviations, n_1 and n_2 denote the number of instances from these two classes. Then the t-score for the feature f_i is:

$$t\_score(f_i) = |\mu_1 - \mu_2| / \sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}. \tag{35}$$

The basic idea of t-score is to assess whether the feature makes the means of two classes statistically different, which can be computed as the ratio between the mean difference and the variance of two classes. The higher the t-score, the more important the feature is.

2.4.3. F-score. T-score is restricted to binary classification task. F-score [Wright 1965] 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, f-score of the feature f_i can be computed as:

$$f\_score(f_i) = \frac{\sum_j \frac{n_j}{c-1}(\mu_j - \mu)^2}{\frac{1}{n-c}\sum_j (n_j - 1)\sigma_j^2}. \tag{36}$$

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

2.4.4. Chi-Square Score. Chi-square score [Liu and Setiono 1995] utilizes the test of independence to assess whether the feature is independent of the class label. Given a particular feature f_i with r different feature values, the Chi-square score of that feature can be computed as:

$$Chi\_square\_score(f_i) = \sum_{j=1}^{r}\sum_{s=1}^{c}\frac{(n_{js} - \mu_{js})^2}{\mu_{js}}, \tag{37}$$

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 index [Gini 1912] is also a widely used 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, suppose $\mathcal{W}$ and $\overline{\mathcal{W}}$ denote the set of instances with the feature value smaller or equal to the j-th feature value, and larger than the j-th feature value, respectively. 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:

$$gini\_index\_score(f_i) = \min_{\mathcal{W}}\left(p(\mathcal{W})(1 - \sum_{s=1}^{c}p(C_s|\mathcal{W})^2) + p(\overline{\mathcal{W}})(1 - \sum_{s=1}^{c}p(C_s|\overline{\mathcal{W}})^2)\right), \tag{38}$$

where $p(.)$ denotes the probability. For instance, $p(C_s|\mathcal{W})$ is the conditional probability of class s given $\mathcal{W}$. For binary classification, Gini Index can take a maximum value of 0.5, it can also be used in multi-class classification problems. Unlike previous statistical measures, the lower the Gini index value, the more relevant the feature is.

2.4.6. CFS. The basic idea of CFS [Hall and Smith 1999] is to use a correlation based heuristic to evaluate the worth of a feature subset $\mathcal{S}$:

$$CFS\_score(\mathcal{S}) = \frac{k\overline{r_{cf}}}{\sqrt{k + k(k-1)\overline{r_{ff}}}}, \tag{39}$$

where the CFS score shows the heuristic "merit" of the feature subset $\mathcal{S}$ with k features. $\overline{r_{cf}}$ is the mean feature class correlation and $\overline{r_{ff}}$ is the average feature-feature correlation. In Eq. (39), the numerator indicates the predictive power of the feature set while the denominator shows how much redundancy the feature set has. The basic idea is that a good feature subset should have a strong correlation with class labels and are weakly intercorrelated. To get the feature-class correlation and feature-feature correlation, CFS uses symmetrical uncertainty [Vetterling et al. 1992]. As finding the globally optimal subset is computational prohibitive, it 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. It then starts with an empty set and expands the set by the feature with the highest utility until it satisfies some stopping criteria.

Discussion*: Most of the statistical based feature selection methods rely on predefined statistical measures to filter out unwanted features, and are simple, straightforward in nature. And the computational costs of these methods are often very low. To this end, they are often used as a preprocessing step before applying other sophisticated feature selection algorithms. Also, as similarity based feature selection methods, these methods often evaluate the importance of features individually and hence cannot handle feature redundancy. Meanwhile, most algorithms in this family can only work on discrete data and conventional data discretization techniques are required to preprocess numerical and continuous variables.*

2.5. Other Methods

In this subsection, we present other feature selection methods that do not belong to the above four types of feature selection algorithms. In particular, we review hybrid feature selection methods, deep learning based and reconstruction based methods.

Hybrid feature selection methods is a kind of ensemble-based methods that aim to construct a group of feature subsets from different feature selection algorithms, and then produce an aggregated result out of the group. In this way, the instability and perturbation issues of most single feature selection algorithms can be alleviated, and also, the subsequent learning tasks can be enhanced. Similar to conventional ensemble learning methods [Zhou 2012], hybrid feature selection methods consist of two steps: (1) construct a set of different feature selection results; and (2) aggregate different outputs into a consensus result. Different methods differ in the way how these two steps are performed. For the first step, existing methods either ensemble the selected feature subsets of a single method on different sample subset or ensemble the selected feature subsets from multiple feature selection algorithms. In particular, a sampling method to obtain different sample subsets is necessary for the first case; and typical sampling methods include random sampling and bootstrap sampling. For example, [Saeys et al. 2008] studied the ensemble feature selection which aggregates a conventional feature selection algorithm such as RELIEF with multiple bootstrapped samples of the training data. In [Abeel et al. 2010], the authors improved the stability of SVM-RFE feature selection algorithm by applying multiple random sampling on the original data. The second step involves in aggregating rankings of multiple selected feature subset. Most of the existing methods employ a simple yet effective linear aggregation function [Saeys et al. 2008; Abeel et al. 2010; Yang and Mao 2011]. Nonetheless, other ranking aggregation functions such as Markov chain-based method [Dutkowski and Gambin 2007], distance synthesis method [Yang et al. 2005], and stacking method [Netzer et al. 2009] are also widely used. In addition to using the aggregation function, another way is to identify the consensus features directly from multiple sample subsets [Loscalzo et al. 2009].

Nowadays, deep learning techniques are popular and successful in various real-world applications, especially in computer vision and natural language processing. Deep learning is distinct from feature selection as deep learning leverages deep neutral networks structures to learn new feature representations while feature selection directly finds relevant features from the original features. From this perspective, the results of feature selection are more human readable and interpretable. Even though deep learning is mainly used for feature learning, there are still some attempts that use deep learning techniques for feature selection. We briefly review these deep learning based feature selection methods. For example, in [Li et al. 2015a], a deep feature selection model (DFS) is proposed. DFS selects features at the input level of a deep neural network. Typically, it adds a sparse one-to-one linear layer between the input layer and the first hidden layer of a multilayer perceptrons (MLP). To achieve feature selection, DFS imposes sparse regularization term, then only the features corresponding to nonzero weights are selected. Similarly, in [Roy et al. 2015], the authors

also propose to select features at the input level of a deep neural network. The difference is that they propose a new concept - net positive contribution, to assess if features are more likely to make the neurons contribute in the classification phase. Since heterogeneous (multi-view) features are prevalent in machine learning and pattern recognition applications, [Zhao et al. 2015] proposes to combine deep neural networks with sparse representation for grouped heterogeneous feature selection. It first extracts a new unified representation from each feature group using a multi-modal neural network. Then the importance of features is learned by a kind of sparse group lasso method. In [Wang et al. 2014a], the authors propose an attentional neural network, which guides feature selection with cognitive bias. It consists of two modules, a segmentation module, and a classification module. First, given a cognitive bias vector, segmentation module segments out an object belonging to one of classes in the input image. Then, in the classification module, a reconstruction function is applied to the segment to gate the raw image with a threshold for classification. When features are sensitive to a cognitive bias, the cognitive bias will activate the corresponding relevant features.

Recently, data reconstruction error emerged as a new criterion for feature selection, especially for unsupervised feature selection. It defines feature relevance as the capability of features to approximate the original data via a reconstruction function. Among them, Convex Principal Feature Selection (CPFS) [Masaeli et al. 2010] reformulates the feature selection problem as a convex continuous optimization problem that minimizes a mean-squared-reconstruction error with linear and sparsity constraint. GreedyFS [Farahat et al. 2011] uses a projection matrix to project the original data onto the span of some representative feature vectors and derives an efficient greedy algorithm to obtain these representative features. Zhao et al. [Zhao et al. 2016] formulates the problem of unsupervised feature selection as the graph regularized data reconstruction. The basic idea is to make the selected features well preserve the data manifold structure of the original data, and reconstruct each data sample via linear reconstruction. A pass-efficient unsupervised feature selection is proposed in [Maung and Schweitzer 2013]. It can be regarded as a modification of the classical pivoted QR algorithm, the basic idea is still to select representative features that can minimize the reconstruction error via linear function. The aforementioned methods mostly use linear reconstruction functions, [Li et al. 2017a] argues that the reconstruction function is not necessarily linear and proposes to learn the reconstruction function automatically function from data. In particular, they define a scheme to embed the reconstruction function learning into feature selection.

3. FEATURE SELECTION WITH STRUCTURED FEATURES

Existing feature selection methods for generic data are based on a strong assumption that features are independent of each other (flat) while ignoring the inherent feature structures. However, in many real applications features could 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; Wang and Ye 2015]. If this is the case, feature selection algorithms incorporating knowledge about the structure information may help find more relevant features and therefore can improve subsequent learning tasks. One motivating example is from bioinformatics, in the study of array CGH, features have some natural spatial order, incorporating such spatial structure can help select more important features and achieve more accurate classification accuracy. Therefore, in this section, we discuss some representative feature selection algorithms which explicitly consider feature structures. Specifically, we will focus on group structure, tree structure and graph structure.

A popular and successful approach to achieve feature selection with structured features is to minimize an empirical error penalized by a structural regularization term:

$$\mathbf{w} = \operatorname*{argmin}_{\mathbf{w}} loss(\mathbf{w}; \mathbf{X}, \mathbf{y}) + \alpha\, penalty(\mathbf{w}, \mathcal{G}), \tag{40}$$

where $\mathcal{G}$ denotes the structures among features and α is a trade-off parameter between the loss function and the structural 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. (26), the only difference is that for feature selection with structured features, we explicitly consider the structural information $\mathcal{G}$ among features in the sparse regularization term.

3.1. Feature Selection with Group Feature Structures

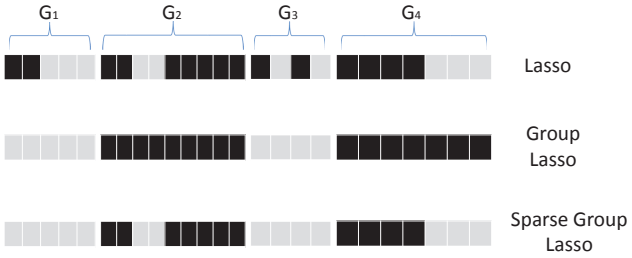


Fig. 3: Illustration of Lasso, Group Lasso, and Sparse Group Lasso. The feature set can be divided into four groups G_1, G_2, G_3 and G_4. The column with dark color denotes selected features while the column with light color denotes unselected features.

First, features could exhibit group structures. One of the most common examples 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 model the group structure explicitly.

3.1.1. Group Lasso. Group Lasso [Yuan and Lin 2006; Bach 2008; Jacob et al. 2009; Meier et al. 2008], which derives feature coefficients from certain groups to be small or exact zero, is a solution to this problem. In other words, it selects or ignores a group of features as a whole. The difference between Lasso and Group Lasso is shown by the illustrative example in Fig. 3. Suppose that these features come from 4 different groups and there is no overlap between these groups. Lasso completely ignores the group structures among features, and the selected features are from four different groups. On the contrary, Group Lasso tends to select or not select features from different groups as a whole. As shown in the figure, Group Lasso only selects the second and the fourth group G_2 and G_4, features in the 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 all previous ℓ_2-norm terms. The objective function of Group Lasso is formulated as follows:

$$\min_{\mathbf{w}} loss(\mathbf{w}; \mathbf{X}, \mathbf{y}) + \alpha \sum_{i=1}^{g} h_i \|\mathbf{w}_{G_i}\|_2, \tag{41}$$

where h_i is a weight for the i-th group $\mathbf{w}_{G_i}$ which can be considered as a prior to measuring the contribution of the i-th group in the feature selection process.

3.1.2. Sparse Group Lasso. Once Group Lasso selects a group, all the features in the selected group will be kept. However, in many cases, not all features in the selected group could be useful, and it is desirable to consider the intrinsic feature structures and select features from different selected groups simultaneously (as illustrated in Fig. 3). Sparse Group Lasso [Friedman et al. 2010; Peng 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:

$$\min_{\mathbf{w}} loss(\mathbf{w}; \mathbf{X}, \mathbf{y}) + \alpha\|\mathbf{w}\|_1 + (1-\alpha)\sum_{i=1}^{g} h_i\|\mathbf{w}_{G_i}\|_2, \tag{42}$$

where α is parameter between 0 and 1 to balance the contribution of inter-group sparsity and intra-group sparsity for feature selection. The difference between Lasso, Group Lasso and Sparse Group Lasso is shown in Fig. 3.

3.1.3. Overlapping Sparse Group Lasso. Above methods consider the disjoint group structures among features. However, groups may also overlap with each other [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. A general Overlapping Sparse Group Lasso regularization is similar to the regularization term of Sparse Group Lasso. The difference is that different feature groups G_i can have an overlap, i.e., there exist 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 the group structures, features can also exhibit tree structures. For example, in face recognition, different pixels can be represented as a tree, where the root node indicates the whole face, its child nodes can be different organs, and each specific pixel is considered as a leaf node. Another motivating example is that genes/proteins may form certain hierarchical tree structures [Liu and Ye 2010; Wang et al. 2017]. 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].

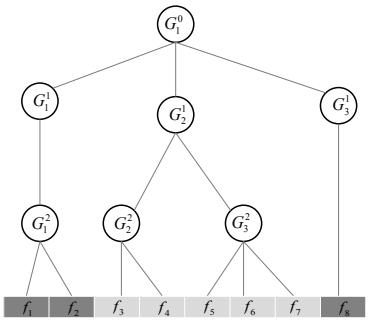


Fig. 4: Illustration of the tree structure among features. These eight features form a simple index tree with a depth of 3.

3.2.1. Tree-guided Group Lasso. In Tree-guided Group Lasso [Liu and Ye 2010], 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 is 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.

In 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 in level 0), and n_i denotes the number of nodes in the level i. 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, then $G_j^i \subseteq G_m^{i-1}$.

We explain these conditions via an illustrative example in Fig. 4. 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 $G_1^0 = \{f_1, f_2, f_3, f_4, f_5, f_6, f_7, f_8\}$, $G_1^1 = \{f_1, f_2\}, G_2^1 = \{f_3, f_4, f_5, f_6, f_7\}, G_3^1 = \{f_8\}$, $G_1^2 = \{f_1, f_2\}, G_2^2 = \{f_3, f_4\}, G_3^2 = \{f_5, f_6, f_7\}$. G_1^0 is the root node of the index tree. 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 those of the parent node. In this way, the objective function of Tree-guided Group Lasso is:

$$\min_{\mathbf{w}} loss(\mathbf{w}; \mathbf{X}, \mathbf{y}) + \alpha \sum_{i=0}^{d} \sum_{j=1}^{n_i} h_j^i \|\mathbf{w}_{G_j^i}\|_2, \tag{43}$$

where $\alpha \geq 0$ is a regularization parameter and $h_j^i \geq 0$ is a predefined 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, all of its child nodes will not be selected. For example, as illustrated in Fig. 4, 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 cases, features may have strong pairwise interactions. 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 there exist strong pairwise dependencies between genes. Since features show certain kinds of dependencies in these cases, we can model them by an undirected graph, where nodes represent features and edges among nodes show the pairwise dependencies between features [Sandler et al. 2009; Kim and Xing 2009; Yang et al. 2012]. We can use an 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 $\mathbf{A} \in \mathbb{R}^{N_n \times N_n}$.

3.3.1. Graph Lasso. 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 for the feature graph on the basis of Lasso [Ye and Liu 2012]. The formulation is:

$$\min_{\mathbf{w}} loss(\mathbf{w}; \mathbf{X}, \mathbf{y}) + \alpha \|\mathbf{w}\|_1 + (1 - \alpha) \sum_{i,j} \mathbf{A}(i, j)(\mathbf{w}_i - \mathbf{w}_j)^2, \tag{44}$$

where the first regularization term $\alpha\|\mathbf{w}\|_1$ is from Lasso while the second term ensures that if a pair of features show strong dependency, i.e., large $\mathbf{A}(i, j)$, their feature coefficients should also be similar to each other.

3.3.2. GFLasso. In Eq. (44), Graph Lasso encourages features connected together have similar feature coefficients. However, 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. GFLasso [Kim and Xing 2009] is proposed to model both positive and negative feature correlations, the objective function is:

$$\min_{\mathbf{w}} loss(\mathbf{w}; \mathbf{X}, \mathbf{y}) + \alpha\|\mathbf{w}\|_1 + (1-\alpha)\sum_{i,j}\mathbf{A}(i,j)|\mathbf{w}_i - \text{sign}(r_{i,j})\mathbf{w}_j|, \tag{45}$$

where $r_{i,j}$ indicates the correlation between two features f_i and f_j. When two features are positively correlated, 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. A 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. To address the limitations of GFLasso, [Yang et al. 2012] propose GOSCAR by putting a ℓ_∞-norm regularization to enforce pairwise feature coefficients to be equivalent if two features are connected in the feature graph. The formulation is:

$$\min_{\mathbf{w}} loss(\mathbf{w}; \mathbf{X}, \mathbf{y}) + \alpha\|\mathbf{w}\|_1 + (1-\alpha)\sum_{i,j}\mathbf{A}(i,j)\text{max}(|\mathbf{w}_i|, |\mathbf{w}_j|). \tag{46}$$

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 $\text{max}(|\mathbf{w}_i|, |\mathbf{w}_j|) = \frac{1}{2}(|\mathbf{w}_i + \mathbf{w}_j| + |\mathbf{w}_i - \mathbf{w}_j|) = |\mathbf{u}'\mathbf{w}| + |\mathbf{v}'\mathbf{w}|$, where $\mathbf{u}$ and $\mathbf{v}$ are sparse vectors with only two nonzero entries such that $\mathbf{u}_i = \mathbf{u}_j = \frac{1}{2}$, $\mathbf{v}_i = -\mathbf{v}_j = \frac{1}{2}$.

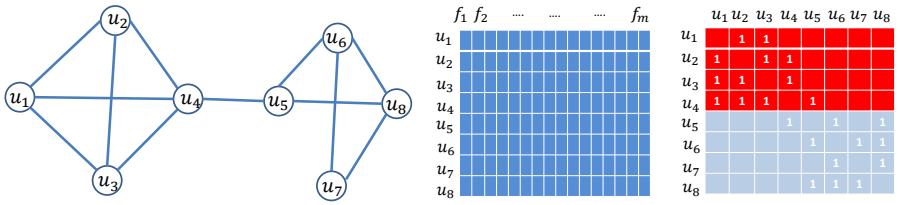
Discussion*: This family of algorithms explicitly take the structures among features as prior knowledge and feed into feature selection. Therefore, the selected features could enhance subsequent learning tasks. However, most of these methods are based on the sparse learning framework, and often involves in solving complex optimization algorithms. Thus, computational costs could be relatively high. Moreover, the feature structure are often given a priori, it is still a challenging problem to automatically infer the structures from data for feature selection.*

4. FEATURE SELECTION WITH HETEROGENEOUS DATA

Traditional feature selection algorithms are heavily based on the data i.i.d. assumption. However, heterogeneous data from different sources is becoming more and more prevalent in the era of big data. For example, in the medical domain, genes are often associated with different types of clinical features. Since data of each source can be noisy, partial, or redundant, how to find relevant sources and how to fuse them together for effective feature selection is a challenging problem. Another example is in social media platforms, instances of high dimensionality 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) linked data; (2) multi-source data; and (3) multi-view data. Note that multi-source and multi-view feature selection are different in two ways: First, multi-source feature selection aims to select features from the original feature space by integrating multiple sources while multi-view feature selection selects 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 relations among features from different sources.

4.1. Feature Selection Algorithms with Linked Data

Linked data is ubiquitous in real-world applications such as Twitter (tweets linked by hyperlinks), Facebook (users connected by friendships) and biological systems (pro-



(a) Linked Data (b) Linked Data Representation

Fig. 5: An illustrative example of linked data.

tein interactions). Due to different types of links, they are distinct from traditional attribute-value data (or so-called "flat" data).

Fig. 5 illustrates an example of linked data and its representation. Fig. 5(a) shows 8 linked instances, the feature information is illustrated in the left part of Fig. 5(b). Linked data provides an extra source of information, which can be represented by an adjacency matrix, illustrated in the right part of Fig. 5(b). 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; Long et al. 2007; Li et al. 2017b], 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 is often unlabeled, how to evaluate the relevance of features without labels. Recent years have witnessed a surge of research interests in performing feature selection on linked data [Gu and Han 2011; Tang and Liu 2012a; Tang and Liu 2012b; Tang and Liu 2013; Wei et al. 2015; Wei et al. 2016b; Li et al. 2015b; Li et al. 2016b; Li et al. 2016a; Cheng et al. 2017]. Next, we introduce some representative algorithms in this family.

4.1.1. Feature Selection on Networks. In [Gu and Han 2011], the authors propose a supervised feature selection algorithm (FSNet) based on Laplacian Regularized Least Squares (LapRLS). In detail, they propose to use a 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 and $\mathbf{Y} \in \mathbb{R}^{n \times c}$ denotes the one-hot label matrix, $\mathbf{A}$ denotes the adjacency matrix for all n linked instances. FSNet first attempts to learn a linear classifier $\mathbf{W} \in \mathbb{R}^{d \times c}$ to map $\mathbf{X}$ to $\mathbf{Y}$:

$$\min_{\mathbf{W}} \|\mathbf{XW} - \mathbf{Y}\|_F^2 + \alpha\|\mathbf{W}\|_{2,1} + \beta\|\mathbf{W}\|_F^2. \tag{47}$$

The term $\|\mathbf{W}\|_{2,1}$ is included to achieve joint feature sparsity across different classes. $\|\mathbf{W}\|_F^2$ prevents the overfitting of the model. To capture the correlation between link information and 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, which results in the following objective function:

$$\min_{\mathbf{W}} \|\mathbf{XW} - \mathbf{Y}\|_F^2 + \alpha\|\mathbf{W}\|_{2,1} + \beta\|\mathbf{W}\|_F^2 + \gamma tr(\mathbf{W}'\mathbf{X}'\mathbf{LXW}), \tag{48}$$

where $tr(\mathbf{W}'\mathbf{X}'\mathbf{LXW})$ is the graph regularization, and γ balances the contribution of content information and link information for feature selection.

4.1.2. Feature Selection for Social Media Data (LinkedFS). [Tang and Liu 2012a] investigate the feature selection problem on social media data by evaluating various social relations such as CoPost, CoFollowing, CoFollowed, and Following. These four types of relations are supported by social correlation theories such as homophily [McPherson et al. 2001] and social influence [Marsden and Friedkin 1993].

We use the CoPost relation as an example to illustrate how these relations can be integrated into feature selection. 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; $\mathbf{Y} \in \mathbb{R}^{n \times c}$ denotes the label matrix; $\mathbf{u} = \{u_1, u_2, ..., u_n\}$ 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 to enforce the hypothesis that the class labels (i.e., topics) of posts by the same user are similar, resulting in the following objective function:

$$\min_{\mathbf{W}} \|\mathbf{XW} - \mathbf{Y}\|_F^2 + \alpha\|\mathbf{W}\|_{2,1} + \beta \sum_{u \in \mathbf{u}} \sum_{\{p_i, p_j\} \in \mathbf{p}_u} \|\mathbf{X}(i,:)\mathbf{W} - \mathbf{X}(j,:)\mathbf{W}\|_2^2, \tag{49}$$

where $\mathbf{p}_u$ denotes the set of posts by user u. The parameter α controls the sparsity of $\mathbf{W}$ in rows across all class labels and β controls the contribution of the CoPost relations.

4.1.3. Unsupervised Feature Selection for Linked Data. Linked Unsupervised Feature Selection (LUFS) [Tang and Liu 2012b] is an unsupervised feature selection framework for linked data. Without label information to assess feature relevance, LUFS assumes the existence of pseudo labels, and uses $\mathbf{Y} \in \mathbb{R}^{n \times c}$ to denote the pseudo label matrix such that each row of $\mathbf{Y}$ has only one nonzero entry. Also, LUFS assumes a linear mapping matrix $\mathbf{W} \in \mathbb{R}^{d \times c}$ between feature $\mathbf{X}$ and $\mathbf{Y}$. First, to consider the constraints from link information, LUFS employs social dimension approach [Tang and Liu 2009] to obtain the hidden factors $\mathbf{H}$ that incur the interdependency among instances. Then, 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 $\mathbf{S}_w = \mathbf{Y}'\mathbf{Y} - \mathbf{Y}'\mathbf{F}\mathbf{F}'\mathbf{Y}$, $\mathbf{S}_b = \mathbf{Y}'\mathbf{F}\mathbf{F}'\mathbf{Y}$, $\mathbf{S}_t = \mathbf{Y}'\mathbf{Y}$ respectively, where $\mathbf{F} = \mathbf{H}(\mathbf{H}'\mathbf{H})^{-\frac{1}{2}}$ 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 maximizing $tr((\mathbf{S}_t)^{-1}\mathbf{S}_b)$. Second, to take advantage of feature information, LUFS obtains the constraints by spectral analysis to minimize $tr(\mathbf{Y}'\mathbf{L}\mathbf{Y})$, where $\mathbf{L}$ is the Laplacian matrix derived from feature affinity matrix $\mathbf{S}$. With these, the objective function of LUFS is formulated as follows:

$$\min_{W} tr(\mathbf{Y}'\mathbf{L}\mathbf{Y}) - \alpha tr((\mathbf{S}_t)^{-1}\mathbf{S}_b), \tag{50}$$

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. (50) can be eventually represented as:

$$\begin{aligned} \min_{\mathbf{W}} & tr(\mathbf{W}'(\mathbf{X}'\mathbf{L}\mathbf{X} + \alpha\mathbf{X}'(\mathbf{I}_n - \mathbf{F}\mathbf{F}'))\mathbf{W}) + \beta\|\mathbf{W}\|_{2,1} \\ & \text{s.t.} \quad \mathbf{W}'(\mathbf{X}'\mathbf{X} + \lambda\mathbf{I}_d)\mathbf{W} = \mathbf{I}_c, \end{aligned} \tag{51}$$

where β controls the sparsity of $\mathbf{W}$ in rows and $\lambda\mathbf{I}_d$ makes $\mathbf{X}'\mathbf{X} + \lambda\mathbf{I}_d$ invertible.

4.1.4. Robust Unsupervised Feature Selection for Networked Data. LUFS performs network structure modeling and feature selection separately, and the feature selection heavily depends on the quality of extracted latent representations. In other words, the performance of LUFS will be jeopardized when there are a lot of noisy links in the network. [Li et al. 2016b] propose a robust unsupervised feature selection framework (NetFS) to embed latent representation learning into feature selection. Specifically, let $\mathbf{X} \in \mathbb{R}^{n \times d}$ and $\mathbf{A} \in \mathbb{R}^{n \times n}$ denote the feature matrix and adjacency matrix respectively. NetFS first uncovers a low-rank latent representation $\mathbf{U}$ by a symmetric NMF model. The latent representation describes a set of diverse affiliation factors hidden in a network, and instances with similar latent representations are more likely to be connected to

each other than the instances with dissimilar latent representations. As latent factors encode some hidden attributes of instances, they should be related to some features. Thus, NetFS takes $\mathbf{U}$ as a constraint to perform feature selection via:

$$\min_{\mathbf{U}\geq 0,\mathbf{W}} \|\mathbf{XW}-\mathbf{U}\|_F^2+\alpha\|\mathbf{W}\|_{2,1}+\frac{\beta}{2}\|\mathbf{A}-\mathbf{UU}'\|_F^2, \tag{52}$$

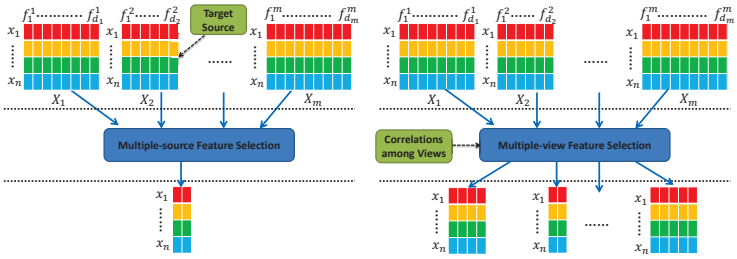
where α and β are two balance parameters. By embedding latent representation learning into feature selection, these two phases could help and boost each other. Feature information can help learn better latent representations which are robust to noisy links, and better latent representations can fill the gap of limited label information and rich link information to guide feature selection. The authors further extended the NetFS model to the dynamic case to obtain a subset of relevant features continuously when both the feature information and network structure evolve over time [Li et al. 2016a]. In addition to positive links, many real-world networks also contain negative links, such as distrust relations in Epinions and foes in Slashdot. Based on NetFS, the authors in [Cheng et al. 2017] further study if negative links have added value over positive links in finding more relevant features.

4.2. Multi-Source Feature Selection

For many learning tasks, we often have multiple data sources for the same set of data instances. For example, recent advancements in bioinformatics reveal that non-coding RNA species function across a variety of biological process. The task of multi-source feature selection in this case 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 from m sources.

4.2.1. Multi-Source Feature Selection via Geometry-Dependent Covariance Analysis (GDCOV). To integrate information from multiple sources, [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 multiple affinity matrices $\mathbf{S}_i$, 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 $\mathbf{C}=\frac{1}{n-1}\mathbf{\Pi X}_i'(\mathbf{S}-\frac{\mathbf{S11}'\mathbf{S}}{\mathbf{1}'\mathbf{S1}})\mathbf{X}_i\mathbf{\Pi}$, 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 highest variances. Selecting features based on this approach is equivalent to choosing features that are consistent with the global geometry pattern. The second method, GPCOVspca, applies Sparse Principal Component Analysis (SPCA) [d'Aspremont et al. 2007] to select features that can retain the total variance maximally. Hence, it considers interactions among features and can select features with less redundancy.



(a) Multi-Source Feature Selection (b) Multi-View Feature Selection

Fig. 6: Differences between multi-source and multi-view feature selection.

4.3. Feature Selection Algorithms with Multi-View Data

Multi-View data represent different facets of data instances in different feature spaces. These feature spaces are naturally dependent and high-dimensional. Hence, the task of multi-view feature selection arises [Feng et al. 2013; Tang et al. 2013; Wang et al. 2013; Liu et al. 2016a], which aims to select features from different feature spaces simultaneously by using their relations. One motivating example is to select relevant features in pixels, tags, and terms associated with images 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 Fig. 6. 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. Adaptive unsupervised multi-view feature selection (AUMFS) [Feng et al. 2013] takes advantages of the data cluster structure, the data similarity and the correlations among views simultaneously. 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 respectively, $\mathbf{X} = [\mathbf{X}_1, \mathbf{X}_2, ..., \mathbf{X}_m] \in \mathbb{R}^d$ denotes the concatenated data, where $d = d_1 + d_2 + ... + d_m$. AUMFS first builds a feature selection model by using $\ell_{2,1}$-norm regularized least squares loss function:

$$\min_{\mathbf{W},\mathbf{F}} \|\mathbf{XW} - \mathbf{F}\|_{2,1} + \alpha\|\mathbf{W}\|_{2,1}, \tag{53}$$

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 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 learn the shared pseudo class labels. For the data matrix $\mathbf{X}_i$ in each view, it first builds an affinity matrix $\mathbf{S}_i$ based on the data similarity on that view and gets 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. Integrating it with Eq. (53), we have the following objective function:

$$\begin{aligned} &\min tr(\mathbf{F}' \sum_{i=1}^{m} \lambda_i \mathbf{L}_i \mathbf{F}) + \beta(\|\mathbf{XW} - \mathbf{F}\|_{2,1} + \alpha\|\mathbf{W}\|_{2,1}) \\ &\text{s.t.} \quad \mathbf{F}'\mathbf{F} = \mathbf{I}_c, \mathbf{F} \geq 0, \sum_{i=1}^{m} \lambda_i = 1, \lambda_i \geq 0. \end{aligned} \tag{54}$$

where the contribution of each view for the joint spectral clustering is balanced by a nonnegative weight λ_i and the summation of all λ_i equals 1. β is a parameter to balance the contribution of spectral clustering and feature selection.

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

$$\begin{aligned} &\min tr(\mathbf{F}' \sum_{i=1}^{m} \lambda_i \mathbf{L}_i \mathbf{F}) + \sum_{i=1}^{m} \beta(\|\mathbf{X}_i \mathbf{W}_i - \mathbf{F}\|_{2,1} + \alpha \|\mathbf{W}_i\|_{2,1}) \\ &\text{s.t.} \quad \mathbf{F}'\mathbf{F} = \mathbf{I}_c, \mathbf{F} \geq 0, \sum_{i=1}^{m} \lambda_i = 1, \lambda_i \geq 0. \end{aligned} \tag{55}$$

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. In some cases, features from a certain view contain more discriminative information than features from other views. 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. 2013] 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. Hence, the objective function of the proposed method is formulated as follows:

$$\begin{aligned} &\min_{\mathbf{W},\mathbf{F}} \|\mathbf{X}\mathbf{W} - \mathbf{F}\|_F^2 + \alpha \|\mathbf{W}\|_{2,1} + \beta \|\mathbf{W}\|_{G_1} \\ &\text{s.t.} \quad \mathbf{F}'\mathbf{F} = \mathbf{I}_c, \mathbf{F} \geq 0, \end{aligned} \tag{56}$$

where α and β are used to control inter-view sparsity and intra-view sparsity.

Discussion*: Feature selection algorithms for heterogeneous data can handle various types of data simultaneously. By fusing multiple data sources together, the selected features are able to capture the inherent characteristics of data and could better serve other learning problems on such data. However, most of the proposed algorithms in this family use matrices to represent the data and often convert the feature selection problem into an optimization algorithm. The resulted optimization problem often requires complex matrix operations which is computationally expensive, and also, limits the scalability of these algorithms for large-scale data. How to design efficient and distributed algorithms to speed up the computation is still a fertile area and needs deeper investigation.*

5. FEATURE SELECTION WITH STREAMING DATA

Previous methods 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 faced with data streams and feature streams. In the worst cases, the size of data or 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 online spam email detection problem, where new emails are continuously arriving; it is not easy to employ batch-mode feature selection methods

to select relevant features in a timely manner. On an orthogonal setting, feature selection for streaming features also has its practical significances. For example, Twitter produces more than 500 million tweets every day, 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 preferred to perform streaming feature selection to adapt to the changes on the fly. There are also some attempts to study these two dual problems together, which is referred as feature selection on Trapezoidal data streams [Zhang et al. 2015]. We will review some representative algorithms for these two orthogonal problems.

5.1. Feature Selection Algorithms with Feature Streams

For the feature selection problems 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 [Perkins and Theiler 2003; Zhou et al. 2005; Wu et al. 2010; Yu et al. 2014; Li et al. 2015b]. At each time step, a typical streaming feature selection 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. The process repeats until no new features show up anymore. Different algorithms have different implementations in the first step. The second step which checks existing features is an optional step.

5.1.1. Grafting Algorithm. The first attempt to perform streaming feature selection is credited to [Perkins and Theiler 2003]. Their method is based on a 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 w subject to ℓ_1-norm regularization, such as Lasso.

The basic idea of Grafting is based on the observation – incorporating a new feature into the Lasso model involves 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 Lasso 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 $\left|\frac{\partial loss(\mathbf{w};\mathbf{X},\mathbf{y})}{\partial \mathbf{w}_j}\right| > \alpha$. 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 are accepted and included in the model, Grafting adopts a conjugate gradient (CG) procedure to optimize the model with respect to all current parameters to exclude some outdated features.

5.1.2. Alpha-investing Algorithm. Alpha-investing [Zhou et al. 2005] 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 by a desire to control the false discovery rate (FDR) of newly arrived features, such that a small portion of spurious features do not affect the model's accuracy significantly. The detailed algorithm works as follows: (1) it initializes $w_0 = 0$ (probability of false positives), $i = 0$ (index of features), and selected features in the model to be empty; (2) it sets $\alpha_i = w_i/2i$ when a new feature arrives; (3) it sets $w_{i+1} = w_i - \alpha_i$ if $p\_value(f_i, SF) \geq \alpha_i$; or set $w_{i+1} = w_i + \alpha_\Delta - \alpha_i$, $SF = SF \cup f_i$ if $p\_value(f_i, SF) < \alpha_i$. The threshold α_i corresponds to the probability of selecting a spurious feature at the time step i. It is adjusted by the wealth w_i, which denotes the acceptable number of false positively detected features at the current moment. The wealth w_i increases when a feature is added to the model. Otherwise, it decreases 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 other hand, each time when a new feature is not included, the wealth is wasted and lowers the chance of finding more spurious features.

5.1.3. Online Streaming Feature Selection Algorithm. Some other researchers study the streaming feature selection problem from an information theoretic perspective [Wu et al. 2010]. According to the definition, the whole feature set consists of four types of features: irrelevant, redundant, weakly relevant but non-redundant, and strongly relevant features. An optimal feature selection should select non-redundant and strongly relevant features. But as features continuously arrive 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. In 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 future steps. In 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.

5.1.4. Unsupervised Streaming Feature Selection in Social Media. Vast majority of streaming feature selection methods are supervised which utilize label information to guide feature selection. 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. 2015b] propose the USFS algorithm to tackle unsupervised streaming feature selection. The key idea of USFS is to utilize source information such as link information. USFS first uncovers hidden social factors from link information by mixed membership stochastic blockmodel [Airoldi et al. 2009]. 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 constraint to perform selection. At a specific time step t, let $\mathbf{X}^{(t)}$, $\mathbf{W}^{(t)}$ denote the corresponding feature matrix and feature coefficient respectively. 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 from $\mathbf{X}^{(t)}$. Then the objective function to achieve feature selection at the time step t is given as follows:

$$\min_{\mathbf{W}^{(t)}} \frac{1}{2}\|\mathbf{X}^{(t)}\mathbf{W}^{(t)} - \mathbf{\Pi}\|_F^2 + \alpha \sum_{i=1}^{k} \|(\mathbf{w}^{(t)})^i\|_1 + \frac{\beta}{2}\|\mathbf{W}^{(t)}\|_F^2 + \frac{\gamma}{2}\|(\mathbf{X}^{(t)}\mathbf{W}^{(t)})'(\mathbf{L}^{(t)})^{\frac{1}{2}}\|_F^2, \tag{57}$$

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 the new feature, 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. (57), the feature is accepted; otherwise the new feature can be removed. When new features are continuously being generated, 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 BFGS 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.

5.2.1. Online Feature Selection. In [Wang et al. 2014b], 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 respectively, 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 sign($\mathbf{w}^{(t)'}\mathbf{x}_i$). To achieve feature selection, 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: (1) get a new data instance $\mathbf{x}_t$ and its class label y_t; (2) make a class label prediction sign($\mathbf{w}^{(t)'}\mathbf{x}_t$) for the new instance; (3) if $\mathbf{x}_t$ is misclassified such that $y_i \mathbf{w}^{(t)'}\mathbf{x}_t < 0$, then $\tilde{\mathbf{w}}_{t+1} = (1 - \lambda\eta)\mathbf{w}_t + \eta y_t \mathbf{x}_t$, $\hat{\mathbf{w}}_{t+1} = \min\{1, 1/\sqrt{\lambda}\|\tilde{\mathbf{w}}_{t+1}\|_2\}\tilde{\mathbf{w}}_{t+1}$, and $\mathbf{w}_{t+1} = Truncate(\hat{\mathbf{w}}_{t+1}, B)$; (4) $\mathbf{w}_{t+1} = (1 - \lambda\eta)\mathbf{w}_t$. In particular, 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 returned 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. To timely select a subset of relevant features when unlabeled data is continuously being generated, [Huang et al. 2015] propose a novel unsupervised feature selection method (FSDS) with only one pass of the data and with 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 computationally 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, the feature coefficients can be obtained by minimizing the following:

$$\min_{\mathbf{W}^{(t)}} \|\mathbf{B}^{(t)}\mathbf{W}^{(t)} - \{\mathbf{e}_1, ..., \mathbf{e}_k\}\|_F^2 + \alpha\|\mathbf{W}^{(t)}\|_F^2, \tag{58}$$

where $\mathbf{B}^{(t)} \in \mathbb{R}^{\ell \times d}$ denote the sketching matrix of $\mathbf{X}^{(t)}$ ($\ell \ll n_t$), $\mathbf{e}_i \in \mathbb{R}^\ell$ is a vector with its i-th location as 1 and other locations as 0. By solving the optimization problem in Eq. (58), the importance of each feature f_i is $score(j) = \max_i |\mathbf{W}^{(t)}(j, i)|$. The higher the feature score, the more important the feature is.

Discussion*: As data is often not static and is generated in a streaming fashion, feature selection algorithms for both feature and data streams are often more desired in practical usage. Most of the existing algorithms in this family employ various strategies to speed up the selection process such that it can deal with new data samples or new features upon the arrival. However, it should be mentioned that most of these algorithms require multiple pass of the data and some even need to store all the historically generated data, which jeopardizes the usage of these algorithms when we only have limited memory or disk storage. It requires further efforts to design streaming algorithms that are effective and efficient with limited storage costs.*

6. PERFORMANCE EVALUATION

We first introduce our efforts in developing an open-source feature selection repository. Then we use algorithms included in the repository as an example to show how to evaluate different feature selection algorithms.

6.1. Feature Selection Repository

First, we introduce our attempt 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 to facilitate 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. 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 publically available benchmark datasets, performance evaluation of algorithms and test cases to run each algorithm. The source code of this repository is available at Github (https://github.com/jundongl/scikit-feature). An interactive tool of the repository is also available [Cheng et al. 2016]. We welcome researchers in this community to contribute algorithms and datasets to our repository.

6.2. Evaluation Methods and Metrics

As an example, we empirically show how to evaluate the performance of feature selection algorithms in the repository. The experimental results can be obtained from our repository project website (http://featureselection.asu.edu/datasets.php). In our project website, for each dataset, we list all applicable feature selection algorithms along with its evaluation on either classification or clustering. Next, we will provide detailed information how these algorithms are evaluated, including evaluation criteria and experimental setup. Different feature selection algorithms can be categorized by the following two criteria: (1) labels: supervised or unsupervised; (2) output: feature weighting or subset selection. The first criterion determines whether we need to use the label information to perform feature selection or not. The second criterion categorizes these algorithms based on the output. Feature weighing algorithms give each feature a score for ranking and feature subset algorithms only show which features are selected.

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 different output types, different evaluation strategies are used: (1) if it is a feature weighting method that outputs the feature scores, 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 the performance of supervised feature selection algorithms, we divide the whole dataset into two parts - the training set $\mathcal{T}$ and test set $\mathcal{U}$. Feature selection algorithms will be first applied to the training set $\mathcal{T}$ to obtain a subset of relevant features $\mathcal{S}$. Then the test set on the selected features acts 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, 10-folds cross validation is used. Normally, the higher the classification performance, the better the selected features are.

Unsupervised Methods. Following the standard way to assess unsupervised feature selection, we evaluate unsupervised algorithms in terms of clustering performance. Two commonly used clustering performance metrics [Cai et al. 2010], i.e., *normalized mutual information* (NMI) and *accuracy* (ACC) are used. 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 a local optimal. The higher the clustering performance, the better the selected features are.

We also list the following information of main algorithms reviewed in this paper in Table II: (1) the type of data: generic data or other types of data; (2) usage of labels: supervised or unsupervised[1]; (3) output: feature weighting or subset selection; (4) feature type: numerical variables or discrete variables (numerical variables can also be divided into continuous variables and discrete variables). For supervised feature selection methods, we also list if the methods are designed to tackle binary-class or multi-class classification problems. Based on the above information, the practitioners can have a more intuitive sense about the applicable scenarios of different methods.

7. OPEN PROBLEMS AND CHALLENGES

Over the past two decades, there are a tremendous amount of attempts in developing feature selection algorithms for both theoretical analysis and real-world applications. However, we still believe there is more work that 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 in the size of data, the scalability of most current feature selection algorithms may be jeopardized. In many scientific and business applications, data is 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 to perform parallel feature selection for large-scale datasets [Singh et al. 2009; Zhao et al. 2013; Yamada et al. 2014; Zadeh et al. 2017]. In addition, most of the existing feature selection methods have a 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 do not scale well on the ultrahigh-dimensional data whose efficiency deteriorates quickly or is even computationally infeasible. In this case, well-designed feature selection algorithms in linear or sublinear running time are 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, hence we can only make one pass of the data where the second pass is either unavailable or computationally expensive. Even though feature selection algorithms can reduce the issue of scalability for online classification or clustering, these methods either require to keep all features in the memory or require iterative processes to visit data instances more than once, which limit their practical usage. In conclusion, even though there is some preliminary work to increase the scalability of feature selection algorithms, we believe that the scalability problem should be paid more attention to keeping pace with the rapid growth of very large-scale and streaming data.

[1]feature selection for regression can also be regarded as a supervised method, here we focus on feature selection for classification problems

Table II: Detailed information of main feature selection algorithms reviewed in the paper.

Data	Methods	Supervision			Output of Features		Feature Type		
		Binary	Multi-class	Unsupervised	Ranking	Subset	Numerical		Categorical
							Continuous	Discrete	
Generic–Flat Features	Fisher Score [Duda et al. 2012]	✓	✓		✓		✓	✓	
	ReliefF [Robnik-Sikonja and Kononenko 2003]	✓	✓		✓		✓	✓	
	Trace Ratio [Nie et al. 2008]	✓	✓		✓		✓	✓	
	Laplacian Score [He et al. 2005]			✓	✓		✓	✓	
	SPEC [Zhao and Liu 2007]	✓	✓	✓	✓		✓	✓	
	MIM [Lewis 1992]	✓	✓		✓			✓	✓
	MIFS [Battiti 1994]	✓	✓		✓			✓	✓
	MRMR [Peng et al. 2005]	✓	✓		✓			✓	✓
	CIFE [Lin and Tang 2006]	✓	✓		✓			✓	✓
	JMI [Meyer et al. 2008]	✓	✓		✓			✓	✓
	CMIM [Fleuret 2004]	✓	✓		✓			✓	✓
	IF [Vidal-Naquet and Ullman 2003]	✓	✓		✓			✓	✓
	ICAP [Jakulin 2005]	✓	✓		✓			✓	✓
	DISR [Meyer and Bontempi 2006]	✓	✓		✓			✓	✓
	FCBF [Yu and Liu 2003]	✓	✓			✓		✓	✓
	ℓ_p-regularized [Liu et al. 2009b]	✓			✓		✓	✓	
	$\ell_{p,q}$-regularized [Liu et al. 2009b]		✓		✓		✓	✓	
	REFS [Nie et al. 2010]	✓	✓		✓		✓	✓	
	MCFS [Cai et al. 2010]			✓	✓		✓	✓	
	UDFS [Yang et al. 2011]			✓	✓		✓	✓	
	NDFS [Li et al. 2012]			✓	✓		✓	✓	
	Low Variance [Pedregosa et al. 2011]			✓		✓		✓	✓
	T-score [Davis and Sampson 1986]	✓			✓		✓	✓	
	F-score [Wright 1965]		✓		✓		✓	✓	
	Chi-square [Liu and Setiono 1995]	✓	✓		✓			✓	✓
	Gini [Gini 1912]	✓	✓		✓			✓	✓
	CFS [Hall and Smith 1999]	✓	✓			✓		✓	✓
Generic – Structured Feature	Group Lasso []	✓			✓		✓	✓	
	Sparse Group Lasso [Friedman et al. 2010]	✓			✓		✓	✓	
	Tree Lasso [Liu and Ye 2010]	✓			✓		✓	✓	
	Graph Lasso [Ye and Liu 2012]	✓			✓		✓	✓	
	GFLasso [Kim and Xing 2009]	✓			✓		✓	✓	
	GOSCAR [Yang et al. 2012]	✓			✓		✓	✓	
Linked Data	FSNet [Gu and Han 2011]	✓	✓		✓		✓	✓	
	LinkedFS [Tang and Liu 2012a]	✓	✓		✓		✓	✓	
	LUFS [Tang and Liu 2012b]			✓	✓		✓	✓	
	NetFS [Li et al. 2016b]			✓	✓		✓	✓	
Multi-Source	GDCOV [Zhao and Liu 2008]			✓	✓		✓	✓	
Multi-View	AUMFS [Feng et al. 2013]			✓	✓		✓	✓	
	MVFS [Tang et al. 2013]			✓	✓		✓	✓	
Streaming Feature	Grafting [Perkins and Theiler 2003]	✓				✓	✓	✓	
	Alpha-Investing [Zhou et al. 2005]	✓				✓	✓	✓	
	OSFS [Wu et al. 2010]	✓				✓		✓	✓
	USFS [Li et al. 2015b]			✓		✓	✓	✓	
Streaming Data	OFS [Wang et al. 2014b]	✓			✓		✓	✓	
	FSDS [Huang et al. 2015]			✓	✓		✓	✓	

7.2. Stability

For supervised feature selection algorithms, their performance is 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. It is defined as the sensitivity of a feature selection algorithm to perturbation in the training data [Kalousis et al. 2007; He and Yu 2010; Saeys et al. 2008; Loscalzo et al. 2009; Yang and Mao 2011]. The perturbation of data could be in various format such as addition/deletion of data samples and the inclusion of noisy/outlier samples. More rigorous definition on the stability of feature selection algorithms can be referred to [Kalousis et al. 2007]. The stability of feature selection algorithms has significant implications in practice as it can help domain experts gain more confidence on the selected features. A motivating example in bioinformatics indicates that domain experts would like to see the same set or similar set of genes (features) selected each time when they obtain new data samples. Otherwise, domain experts would not trust these algorithms and may never use them again. 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 the feature, the number of data instances, etc. In against with supervised feature selection, the stability of unsupervised feature selection algorithms has not been 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 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 affect the stability of these algorithms.

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. A large number of selected features will increase the risk in including 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 computationally 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 numbers of clusters may merge totally different small clusters into one big cluster or split one big cluster into smaller ones. As a consequence, it may result in finding totally different subsets of features. 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 directly for unsupervised feature se-

lection. All in all, we believe that the model selection is an important issue and needs deep investigation.

8. CONCLUSION

Feature selection is effective in preprocessing data and reducing data dimensionality. Meanwhile, it is essential to successful data mining and machine learning applications. It has been a challenging 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 comprehensive models, improving data mining performance, and helping prepare clean and understandable data. The past few years have witnessed the development of many new feature selection methods. This survey article aims to provide a comprehensive review about recent advances in feature selection. We first introduce basic concepts of feature selection and emphasize the importance of applying feature selection algorithms to solve practical problems. Then we classify conventional feature selection methods from the label perspective and the selection strategy perspective. As current categorization cannot meet the rapid development of feature selection research especially in the era of big data, we propose to review recent advances in feature selection algorithms from a data perspective. In particular, we survey feature selection algorithms in four parts: (1) feature selection with generic data with flat features; (2) feature selection with structured features; (3) feature selection with heterogeneous data; and (4) feature selection with streaming data. Specifically, we further classify conventional feature selection algorithms for generic data (flat features) into similarity based, information theoretical based, sparse learning based and statistical based methods, and other types of methods according to the used techniques. For feature selection with structured features, we consider three types of structured features, namely group, tree and graph features. The third part studies feature selection with heterogeneous data, including feature selection with linked data, multi-source and multi-view feature selection. The last part consists of feature selection algorithms for streaming data and streaming features. We analyze the advantages and shortcomings of these different types of feature selection algorithms. To facilitate the research on feature selection, this survey is accompanied by 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. Some suggestions are given on how to evaluate these feature selection algorithms, either supervised or unsupervised methods. At the end of the survey, we present some open problems that require future 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, but to provide a comprehensive structured list of recent advances in feature selection algorithms from a data perspective and a feature selection repository to promote the research in this community.

REFERENCES

Thomas Abeel, Thibault Helleputte, Yves Van de Peer, Pierre Dupont, and Yvan Saeys. 2010. Robust biomarker identification for cancer diagnosis with ensemble feature selection methods. *Bioinformatics* 26, 3 (2010), 392–398.

Edoardo M Airoldi, David M Blei, Stephen E Fienberg, and Eric P Xing. 2009. Mixed membership stochastic blockmodels. In *NIPS*. 33–40.

Salem Alelyani, Huan Liu, and Lei Wang. 2011. The effect of the characteristics of the dataset on the selection stability. In *ICTAI*. 970–977.

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

Jun Chin Ang, Andri Mirzal, Habibollah Haron, and Haza Nuzly Abdull Hamed. 2016. Supervised, unsupervised, and semi-supervised feature selection: a review on gene selection. *IEEE/ACM TCBB* 13, 5 (2016), 971–989.

Hiromasa Arai, Crystal Maung, Ke Xu, and Haim Schweitzer. 2016. Unsupervised feature selection by heuristic search with provable bounds on suboptimality. In *AAAI*. 666–672.

Francis R Bach. 2008. Consistency of the group lasso and multiple kernel learning. *JMLR* 9 (2008), 1179–1225.

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

Mustafa Bilgic, Lilyana Mihalkova, and Lise Getoor. 2010. Active learning for networked data. In *ICML*. 79–86.

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

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

Deng Cai, Chiyuan Zhang, and Xiaofei He. 2010. Unsupervised feature selection for multi-cluster data. In *KDD*. 333–342.

Xiao Cai, Feiping Nie, and Heng Huang. 2013. Exact top-k feature selection via $\ell_{2,0}$-norm constraint. In *IJCAI*. 1240–1246.

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

Xiaojun Chang, Feiping Nie, Yi Yang, and Heng Huang. 2014. A convex formulation for semi-supervised multi-label feature selection. In *AAAI*. 1171–1177.

Kewei Cheng, Jundong Li, and Huan Liu. 2016. FeatureMiner: a tool for interactive feature selection. In *CIKM*. 2445–2448.

Kewei Cheng, Jundong Li, and Huan Liu. 2017. Unsupervised feature selection in signed social networks. In *KDD*.

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

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

Chris Ding, Ding Zhou, Xiaofeng He, and Hongyuan Zha. 2006. R 1-PCA: rotational invariant ℓ_1-norm principal component analysis for robust subspace factorization. In *ICML*. 281–288.

Liang Du and Yi-Dong Shen. 2015. Unsupervised feature selection with adaptive structure learning. In *KDD*. 209–218.

Liang Du, Zhiyong Shen, Xuan Li, Peng Zhou, and Yi-Dong Shen. 2013. Local and global discriminative learning for unsupervised feature selection. In *ICDM*. 131–140.

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

Janusz Dutkowski and Anna Gambin. 2007. On consensus biomarker selection. *BMC bioinformatics* 8, 5 (2007), S5.

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

Jianqing Fan, Richard Samworth, and Yichao Wu. 2009. Ultrahigh dimensional feature selection: beyond the linear model. *JMLR* 10 (2009), 2013–2038.

Ahmed K Farahat, Ali Ghodsi, and Mohamed S Kamel. 2011. An efficient greedy method for unsupervised feature selection. In *ICDM*. 161–170.

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

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

François Fleuret. 2004. Fast binary feature selection with conditional mutual information. *JMLR* 5 (2004), 1531–1555.

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

Keinosuke Fukunaga. 2013. *Introduction to statistical pattern recognition*. Academic Press.

Shuyang Gao, Greg Ver Steeg, and Aram Galstyan. 2016. Variational information maximization for feature selection. In *NIPS*. 487–495.

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

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

Quanquan Gu, Marina Danilevsky, Zhenhui Li, and Jiawei Han. 2012. Locality preserving feature learning. In *AISTATS*. 477–485.

Quanquan Gu and Jiawei Han. 2011. Towards feature selection in network. In *CIKM*. 1175–1184.

Quanquan Gu, Zhenhui Li, and Jiawei Han. 2011a. Correlated multi-label feature selection. In *CIKM*. ACM, 1087–1096.

Quanquan Gu, Zhenhui Li, and Jiawei Han. 2011b. Generalized fisher score for feature selection. In *UAI*. 266–273.

Quanquan Gu, Zhenhui Li, and Jiawei Han. 2011c. Joint feature selection and subspace learning. In *IJCAI*. 1294–1299.

Baofeng Guo and Mark S Nixon. 2009. Gait feature subset selection by mutual information. *IEEE TMSC(A)* 39, 1 (2009), 36–46.

Isabelle Guyon and André Elisseeff. 2003. An introduction to variable and feature selection. *JMLR* 3 (2003), 1157–1182.

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

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

Satoshi Hara and Takanori Maehara. 2017. Enumerate lasso solutions for feature selection. In *AAAI*. 1985–1991.

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

Xiaofei He, Deng Cai, and Partha Niyogi. 2005. Laplacian score for feature selection. In *NIPS*. 507–514.

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

Chenping Hou, Feiping Nie, Dongyun Yi, and Yi Wu. 2011. Feature selection via joint embedding learning and sparse regression. In *IJCAI*. 1324–1329.

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

Hao Huang, Shinjae Yoo, and S Kasiviswanathan. 2015. Unsupervised feature selection on data streams. In *CIKM*. 1031–1040.

Junzhou Huang, Tong Zhang, and Dimitris Metaxas. 2011. Learning with structured sparsity. *JMLR* 12 (2011), 3371–3412.

Laurent Jacob, Guillaume Obozinski, and Jean-Philippe Vert. 2009. Group lasso with overlap and graph lasso. In *ICML*. 433–440.

Aleks Jakulin. 2005. *Machine learning based on attribute interactions*. Ph.D. Dissertation. Univerza v Ljubljani.

Rodolphe Jenatton, Jean-Yves Audibert, and Francis Bach. 2011. Structured variable selection with sparsity-inducing norms. *JMLR* 12 (2011), 2777–2824.

Rodolphe Jenatton, Julien Mairal, Francis R Bach, and Guillaume R Obozinski. 2010. Proximal methods for sparse hierarchical dictionary learning. In *ICML*. 487–494.

Ling Jian, Jundong Li, Kai Shu, and Huan Liu. 2016. Multi-label informed feature selection. In *IJCAI*. 1627–1633.

Yi Jiang and Jiangtao Ren. 2011. Eigenvalue sensitive feature selection. In *ICML*. 89–96.

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

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

Seyoung Kim and Eric P Xing. 2010. Tree-Guided Group Lasso for Multi-Task Regression with Structured Sparsity. In *ICML*. 543–550.

Kenji Kira and Larry A Rendell. 1992. A practical approach to feature selection. In *ICML Workshop*. 249–256.

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

Daphne Koller and Mehran Sahami. 1995. Toward optimal feature selection. In *ICML*.

Gert RG Lanckriet, Nello Cristianini, Peter Bartlett, Laurent El Ghaoui, and Michael I Jordan. 2004. Learning the kernel matrix with semidefinite programming. *JMLR* 5 (2004), 27–72.

David D Lewis. 1992. Feature selection and feature extraction for text categorization. In *Proceedings of the Workshop on Speech and Natural Language*. 212–217.

Jundong Li, Xia Hu, Ling Jian, and Huan Liu. 2016a. Toward time-evolving feature selection on dynamic networks. In *ICDM*. 1003–1008.

Jundong Li, Xia Hu, Jiliang Tang, and Huan Liu. 2015b. Unsupervised streaming feature selection in social media. In *CIKM*. 1041–1050.

Jundong Li, Xia Hu, Liang Wu, and Huan Liu. 2016b. Robust unsupervised feature selection on networked data. In *SDM*. 387–395.

Jundong Li and Huan Liu. 2017. Challenges of feature selection for big data analytics. *IEEE Intelligent Systems* 32, 2 (2017), 9–15.

Jundong Li, Jiliang Tang, and Huan Liu. 2017a. Reconstruction-based unsupervised feature selection: an embedded approach. In *IJCAI*.

Jundong Li, Liang Wu, Osmar R Zaïane, and Huan Liu. 2017b. Toward personalized relational learning. In *SDM*. 444–452.

Yifeng Li, Chih-Yu Chen, and Wyeth W Wasserman. 2015a. Deep feature selection: theory and application to identify enhancers and promoters. In *RECOMB*. 205–217.

Zechao Li, Yi Yang, Jing Liu, Xiaofang Zhou, and Hanqing Lu. 2012. Unsupervised feature selection using nonnegative spectral analysis. In *AAAI*. 1026–1032.

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

Hongfu Liu, Haiyi Mao, and Yun Fu. 2016a. Robust multi-view feature selection. In *ICDM*. 281–290.

Huan Liu and Hiroshi Motoda. 2007. *Computational methods of feature selection*. CRC Press.

Huan Liu and Rudy Setiono. 1995. Chi2: Feature selection and discretization of numeric attributes. In *ICTAI*. 388–391.

Hongfu Liu, Ming Shao, and Yun Fu. 2016b. Consensus guided unsupervised feature selection. In *AAAI*. 1874–1880.

Jun Liu, Shuiwang Ji, and Jieping Ye. 2009a. Multi-task feature learning via efficient $\ell_{2,1}$-norm minimization. In *UAI*. 339–348.

J. Liu, S. Ji, and J. Ye. 2009b. *SLEP: sparse learning with efficient projections*. Arizona State University. http://www.public.asu.edu/~jye02/Software/SLEP

Jun Liu and Jieping Ye. 2010. Moreau-Yosida regularization for grouped tree structure learning. In *NIPS*. 1459–1467.

Xinwang Liu, Lei Wang, Jian Zhang, Jianping Yin, and Huan Liu. 2014. Global and local structure preservation for feature selection. *TNNLS* 25, 6 (2014), 1083–1095.

Bo Long, Zhongfei Mark Zhang, Xiaoyun Wu, and Philip S Yu. 2006. Spectral clustering for multi-type relational data. In *ICML*. 585–592.

Bo Long, Zhongfei Mark Zhang, and Philip S Yu. 2007. A probabilistic framework for relational clustering. In *KDD*. 470–479.

Steven Loscalzo, Lei Yu, and Chris Ding. 2009. Consensus group stable feature selection. In *KDD*. 567–576.

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

Sofus A Macskassy and Foster Provost. 2007. Classification in networked data: a toolkit and a univariate case study. *JMLR* 8 (2007), 935–983.

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

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

Crystal Maung and Haim Schweitzer. 2013. Pass-efficient unsupervised feature selection. In *NIPS*. 1628–1636.

James McAuley, Ji Ming, Darryl Stewart, and Philip Hanna. 2005. Subband correlation and robust speech recognition. *IEEE TSAP* 13, 5 (2005), 956–964.

Miller McPherson, Lynn Smith-Lovin, and James M Cook. 2001. Birds of a feather: Homophily in social networks. *Annual Review of Sociology* (2001), 415–444.

Lukas Meier, Sara Van De Geer, and Peter Bühlmann. 2008. The group lasso for logistic regression. *JRSS(B)* 70, 1 (2008), 53–71.

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

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

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

Michael Netzer, Gunda Millonig, Melanie Osl, Bernhard Pfeifer, Siegfried Praun, Johannes Villinger, Wolfgang Vogel, and Christian Baumgartner. 2009. A new ensemble-based algorithm for identifying breath gas marker candidates in liver disease using ion molecule reaction mass spectrometry. *Bioinformatics* 25, 7 (2009), 941–947.

Xuan Vinh Nguyen, Jeffrey Chan, Simone Romano, and James Bailey. 2014. Effective global approaches for mutual information based feature selection. In *KDD*. 512–521.

Feiping Nie, Heng Huang, Xiao Cai, and Chris H Ding. 2010. Efficient and robust feature selection via joint $\ell_{2,1}$-norms minimization. In *NIPS*. 1813–1821.

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

Feiping Nie, Wei Zhu, Xuelong Li, et al. 2016. Unsupervised feature selection with structured graph optimization. In *AAAI*. 1302–1308.

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. 2011. Scikit-learn: machine learning in Python. *JMLR* 12 (2011), 2825–2830.

Hanyang Peng and Yong Fan. 2016. Direct sparsity optimization based feature selection for multi-class classification. In *IJCAI*. 1918–1924.

Hanyang Peng and Yong Fan. 2017. A general framework for sparsity regularized feature selection via iteratively reweighted least square minimization. In *AAAI*. 2471–2477.

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

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

Simon Perkins, Kevin Lacker, and James Theiler. 2003. Grafting: Fast, incremental feature selection by gradient descent in function space. *JMLR* 3 (2003), 1333–1356.

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

Mingjie Qian and Chengxiang Zhai. 2013. Robust unsupervised feature selection. In *IJCAI*.

Ariadna Quattoni, Xavier Carreras, Michael Collins, and Trevor Darrell. 2009. An efficient projection for $\ell_{1,\infty}$ regularization. In *ICML*. 857–864.

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

Debaditya Roy, K Sri Rama Murty, and C Krishna Mohan. 2015. Feature selection using deep neural networks. In *IJCNN*. 1–6.

Yvan Saeys, Thomas Abeel, and Yves Van de Peer. 2008. Robust feature selection using ensemble feature selection techniques. *ECMLPKDD*, 313–325.

Yvan Saeys, Iñaki Inza, and Pedro Larrañaga. 2007. A review of feature selection techniques in bioinformatics. *Bioinformatics* 23, 19 (2007), 2507–2517.

Ted Sandler, John Blitzer, Partha P Talukdar, and Lyle H Ungar. 2009. Regularized learning with networks of features. In *NIPS*. 1401–1408.

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

Qiang Shen, Ren Diao, and Pan Su. 2012. Feature selection ensemble. *Turing-100* 10 (2012), 289–306.

Jianbo Shi and Jitendra Malik. 2000. Normalized cuts and image segmentation. *IEEE TPAMI* 22, 8 (2000), 888–905.

Lei Shi, Liang Du, and Yi-Dong Shen. 2014. Robust spectral learning for unsupervised feature selection. In *ICDM*. 977–982.

Alexander Shishkin, Anastasia Bezzubtseva, Alexey Drutsa, Ilia Shishkov, Ekaterina Gladkikh, Gleb Gusev, and Pavel Serdyukov. 2016. Efficient high-order interaction-aware feature selection based on conditional mutual information. In *NIPS*. 4637–4645.

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

Mingkui Tan, Ivor W Tsang, and Li Wang. 2014. Towards ultrahigh dimensional feature selection for big data. *JMLR* 15, 1 (2014), 1371–1429.

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

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

Jiliang Tang, Xia Hu, Huiji Gao, and Huan Liu. 2014b. Discriminant analysis for unsupervised feature selection. In *SDM*. 938–946.

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

Jiliang Tang and Huan Liu. 2012b. Unsupervised feature selection for linked social media data. In *KDD*. 904–912.

Jiliang Tang and Huan Liu. 2013. Coselect: Feature selection with instance selection for social media data. In *SDM*. 695–703.

Lei Tang and Huan Liu. 2009. Relational learning via latent social dimensions. In *KDD*. 817–826.

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

Robert Tibshirani, Michael Saunders, Saharon Rosset, Ji Zhu, and Keith Knight. 2005. Sparsity and smoothness via the fused lasso. *JRSS(B)* 67, 1 (2005), 91–108.

Robert Tibshirani, Guenther Walther, and Trevor Hastie. 2001. Estimating the number of clusters in a data set via the gap statistic. *JRSS(B))* 63, 2 (2001), 411–423.

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

Michel Vidal-Naquet and Shimon Ullman. 2003. Object recognition with informative features and linear classification. In *ICCV*. 281–288.

Hua Wang, Feiping Nie, and Heng Huang. 2013. Multi-view clustering and feature learning via structured sparsity. In *ICML*. 352–360.

Huan Wang, Shuicheng Yan, Dong Xu, Xiaoou Tang, and Thomas Huang. 2007. Trace ratio vs. ratio trace for dimensionality reduction. In *CVPR*. 1–8.

Jie Wang and Jieping Ye. 2015. Multi-layer feature reduction for tree structured group lasso via hierarchical projection. In *NIPS*. 1279–1287.

Jialei Wang, Peilin Zhao, Steven CH Hoi, and Rong Jin. 2014b. Online feature selection and its applications. *IEEE TKDE* 26, 3 (2014), 698–710.

Qian Wang, Jiaxing Zhang, Sen Song, and Zheng Zhang. 2014a. Attentional neural network: Feature selection using cognitive feedback. In *NIPS*. 2033–2041.

Suhang Wang, Jiliang Tang, and Huan Liu. 2015. Embedded unsupervised feature selection. In *AAAI*. 470–476.

Suhang Wang, Yilin Wang, Jiliang Tang, Charu Aggarwal, Suhas Ranganath, and Huan Liu. 2017. Exploiting hierarchical structures for unsupervised feature selection. In *SDM*. 507–515.

Xiaokai Wei, Bokai Cao, and Philip S Yu. 2016a. Nonlinear joint unsupervised feature selection. In *SDM*. 414–422.

Xiaokai Wei, Bokai Cao, and Philip S Yu. 2016b. Unsupervised feature selection on networks: a generative view. In *AAAI*. 2215–2221.

Xiaokai Wei, Sihong Xie, and Philip S Yu. 2015. Efficient partial order preserving unsupervised feature selection on networks. In *SDM*. 82–90.

Xiaokai Wei and Philip S Yu. 2016. Unsupervised feature selection by preserving stochastic neighbors. In *AISTATS*. 995–1003.

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

Xindong Wu, Kui Yu, Hao Wang, and Wei Ding. 2010. Online streaming feature selection. In *ICML*. 1159–1166.

Zhixiang Xu, Gao Huang, Kilian Q Weinberger, and Alice X Zheng. 2014. Gradient boosted feature selection. In *KDD*. 522–531.

Makoto Yamada, Avishek Saha, Hua Ouyang, Dawei Yin, and Yi Chang. 2014. N3LARS: minimum redundancy maximum relevance feature selection for large and high-dimensional data. *arXiv preprint arXiv:1411.2331* (2014).

Feng Yang and KZ Mao. 2011. Robust feature selection for microarray data based on multicriterion fusion. *IEEE/ACM TCBB* 8, 4 (2011), 1080–1092.

Howard Hua Yang and John E Moody. 1999. Data visualization and feature selection: new algorithms for nongaussian data. In *NIPS*. 687–693.

Sen Yang, Lei Yuan, Ying-Cheng Lai, Xiaotong Shen, Peter Wonka, and Jieping Ye. 2012. Feature grouping and selection over an undirected graph. In *KDD*. 922–930.

Yi Yang, Heng Tao Shen, Zhigang Ma, Zi Huang, and Xiaofang Zhou. 2011. $\ell_{2,1}$-norm regularized discriminative feature selection for unsupervised learning. In *IJCAI*. 1589–1594.

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

Yee Hwa Yang, Yuanyuan Xiao, and Mark R Segal. 2005. Identifying differentially expressed genes from microarray experiments via statistic synthesis. *Bioinformatics* 21, 7 (2005), 1084–1093.

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

Kui Yu, Xindong Wu, Wei Ding, and Jian Pei. 2014. Towards scalable and accurate online feature selection for big data. In *ICDM*. 660–669.

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

Stella X Yu and Jianbo Shi. 2003. Multiclass spectral clustering. In *ICCV*. 313–319.

Lei Yuan, Jun Liu, and Jieping Ye. 2011. Efficient methods for overlapping group lasso. In *NIPS*. 352–360.

Ming Yuan and Yi Lin. 2006. Model selection and estimation in regression with grouped variables. *JRSS(B)* 68, 1 (2006), 49–67.

Sepehr Abbasi Zadeh, Mehrdad Ghadiri, Vahab S Mirrokni, and Morteza Zadimoghaddam. 2017. Scalable feature selection via distributed diversity maximization. In *AAAI*. 2876–2883.

Miao Zhang, Chris HQ Ding, Ya Zhang, and Feiping Nie. 2014. Feature selection at the discrete limit. In *AAAI*. 1355–1361.

Qin Zhang, Peng Zhang, Guodong Long, Wei Ding, Chengqi Zhang, and Xindong Wu. 2015. Towards mining trapezoidal data streams. In *ICDM*. 1111–1116.

Lei Zhao, Qinghua Hu, and Wenwu Wang. 2015. Heterogeneous feature selection with multi-modal deep neural networks and sparse group lasso. *IEEE TMM* 17, 11 (2015), 1936–1948.

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

Zhou Zhao, Xiaofei He, Deng Cai, Lijun Zhang, Wilfred Ng, and Yueting Zhuang. 2016. Graph regularized feature selection with data reconstruction. *IEEE TKDE* 28, 3 (2016), 689–700.

Zheng Zhao and Huan Liu. 2007. Spectral feature selection for supervised and unsupervised learning. In *ICML*. 1151–1157.

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

Zheng Zhao, Lei Wang, Huan Liu, et al. 2010. Efficient spectral feature selection with minimum redundancy. In *AAAI*. 673–678.

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

Jing Zhou, Dean Foster, Robert Stine, and Lyle Ungar. 2005. Streaming feature selection using alpha-investing. In *KDD*. 384–393.

Jiayu Zhou, Jun Liu, Vaibhav A Narayan, and Jieping Ye. 2012. Modeling disease progression via fused sparse group lasso. In *KDD*. 1095–1103.

Zhi-Hua Zhou. 2012. *Ensemble methods: foundations and algorithms*. CRC press.

Ji Zhu, Saharon Rosset, Robert Tibshirani, and Trevor J Hastie. 2004. 1-norm support vector machines. In *NIPS*. 49–56.

Pengfei Zhu, Qinghua Hu, Changqing Zhang, and Wangmeng Zuo. 2016. Coupled dictionary learning for unsupervised feature selection. In *AAAI*. 2422–2428.