Jiliang Tang, Salem Alelyani and Huan Liu

Feature Selection for Classification: A Review

Contents

0.1 Introduction

Nowadays, the growth of the high-throughput technologies has resulted in exponential growth in the harvested data with respect to both dimensionality and sample size. The trend of this growth of the UCI machine learning repository is shown in Figure 1. Efficient and effective management of these data becomes increasing challenging. Traditionally manual management of these datasets to be impractical. Therefore, data mining and machine learning techniques were developed to automatically discover knowledge and recognize patterns from these data.

However, these collected data is usually associated with a high level of noise. There are many reasons causing noise in these data, among which imperfection in the technologies that collected the data and the source of the data itself are two major reasons. For example, in the medical images domain, any deficiency in the imaging device will be reflected as noise for the later process. This kind of noise is caused by the device itself. The development of social media changes the role of online users from traditional content consumers to both content creators and consumers. The quality of social media data varies from excellent data to spam or abuse content by nature. Meanwhile, social media data is usually informally written and suffer from grammatical mistakes, misspelling, and improper punctuation. Undoubtedly, extracting useful knowledge and patterns from such huge and noisy data is a challenging task.

Dimensionality reduction is one of the most popular techniques to remove noisy (i.e. irrelevant) and redundant features. Dimensionality reduction techniques can be categorized mainly into feature extraction and feature selection. Feature extraction approaches project features into a new feature space with lower dimensionality and the new constructed features are usually combinations of original features. Examples of feature extraction techniques include Principle Component Analysis (PCA), Linear Discriminant Analysis (LDA) and Canonical Correlation Analysis (CCA). On the other hand, the feature selection approaches aim to select a small subset of features that minimize redundancy and maximize relevance to the target such as the class labels in classification. Representative feature selection techniques include Information Gain, Relief,Fisher Score and Lasso.

Both Feature extraction and feature selection are capable of improving learning performance, lowering computational complexity, building better generalizable models, and decreasing required storage. Feature extraction maps the original feature space to a new feature space with lower dimensions by combining the original feature space. It is difficult to link the features from original feature space to new features. Therefore further analysis of new features is problematic since there is no physical meaning for the transformed features obtained from feature extraction techniques. While feature selection selects a subset of features from the original feature set without any transformation, and maintains the physical meanings of the original features. In this sense, feature selection is superior in terms of better readability and interpretability. This property has its significance in many practical applications such as finding relevant genes to a specific disease and building a sentiment lexicon for sentiment analysis. Typically feature selection and feature extraction are presented separately. Via sparse learning such as ℓ_1 regularization, feature extraction (transformation) methods can be converted into feature selection methods [48].

For the classification problem, feature selection aims to select subset of highly discriminant features. In other words, it selects features that are capable of discriminating samples that belong to different classes. For the problem of feature selection for classification, due to the availability of label information, the relevance of features is assessed as the capability

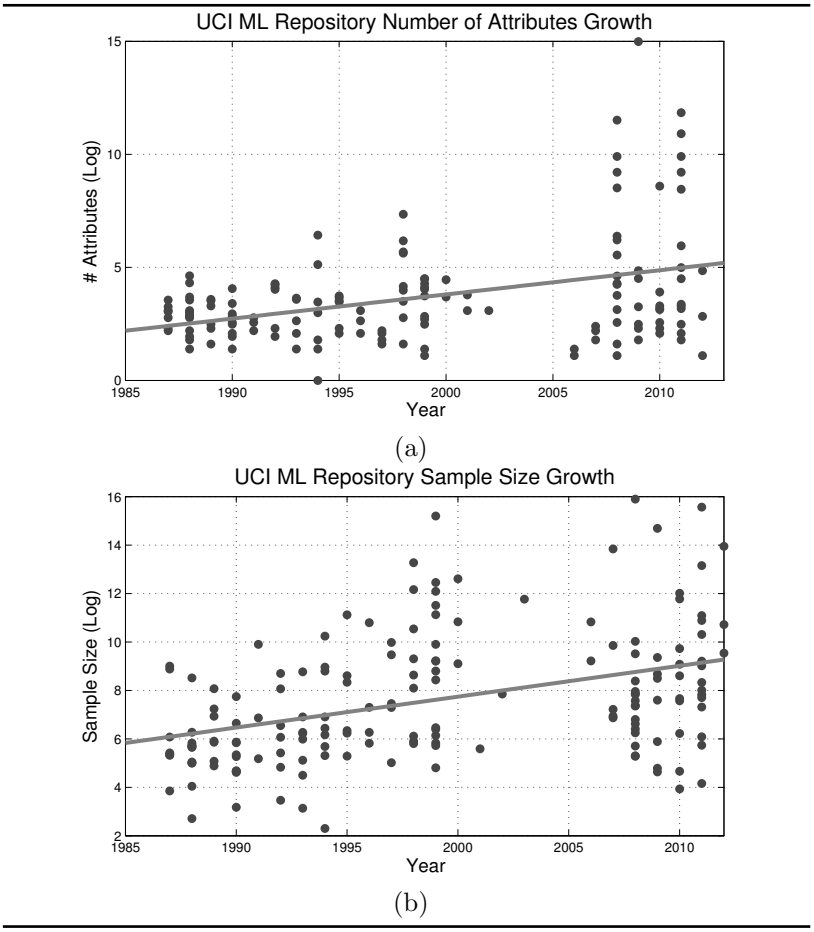


FIGURE 1: Plot (a) shows the dimensionality growth trend in UCI Machine Learning Repository from mid 80s to 2012 while (b) shows the growth in the sample size for the same period.

of distinguishing different classes. For example, a feature f_i is said to be relevant to a class c_j if f_i and c_j are highly correlated.

In the following subsections, we will review the literature of data classification in Section (0.1.1), followed by general discussions about feature selection models in Section (0.1.2) and feature selection for classification in Section (0.1.3).

0.1.1 Data Classification

Classification is the problem of identifying to which of a set of categories (sub-populations) a new observation belongs, on the basis of a training set of data containing observations (or instances) whose category membership is known. Many real-world problems can be modeled as classification problems such as assigning a given email into "spam" or "non-spam" classes, automatically assigning the categories (e.g., "Sports" and "Entertainment") of coming news, and assigning a diagnosis to a given patient as described by observed characteristics of the patient (gender, blood pressure, presence or absence of cer-

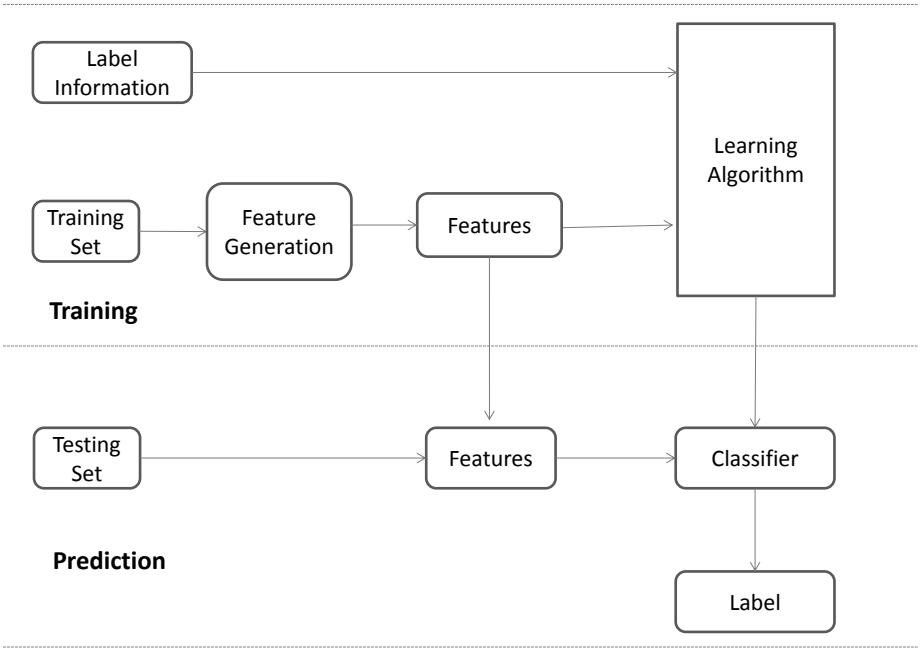


FIGURE 2: A General Process of Data Classification.

tain symptoms, etc.). A general process of data classification is demonstrated in Figure 2, which usually consists of two phases - the training phase and the prediction phase.

In the training phase, data is analyzed into a set of features based on the feature generation models such as the vector space model for text data. These features may either be categorical (e.g. "A", "B", "AB" or "O", for blood type), ordinal (e.g. "large", "medium" or "small"), integer-valued (e.g. the number of occurrences of a part word in an email) or real-valued (e.g. a measurement of blood pressure). Some algorithms work only in terms of discrete data such as ID3 and require that real-valued or integer-valued data be discretized into groups (e.g. less than 5, between 5 and 10, or greater than 10). After representing data through these extracted features, the learning algorithm will utilize the label information as well as the data itself to learn a map function f (or a classifier) from features to labels as,

$$f(features) \rightarrow labels. \tag{0.1}$$

In the prediction phase, data is represented by the feature set extracted in the training process, and then the map function (or the classifier) learned from the training phase will perform on the feature represented data to predict the labels. Note that the feature set used in the training phase should be the same as that in the prediction phase.

There are many classification methods in the literature. These methods can be categorized broadly into Linear classifiers, support vector machines, decision trees and Neural networks. A linear classifier makes a classification decision based on the value of a linear combination of the features. Examples of linear classifiers include Fisher's linear discriminant, logistic regression, the naive bayes classifier and so on. Intuitively, a good separation is achieved by the hyperplane that has the largest distance to the nearest training data point of any class (so-called functional margin), since in general the larger the margin the lower the generalization error of the classifier. Therefore support vector machine constructs

a hyperplane or set of hyperplanes by maximizing the margin. In decision trees, a tree can be learned by splitting the source set into subsets based on an feature value test. This process is repeated on each derived subset in a recursive manner called recursive partitioning. The recursion is completed when the subset at a node has all the same values of the target feature, or when splitting no longer adds value to the predictions.

0.1.2 Feature Selection

In the past thirty years, the dimensionality of the data involved in machine learning and data mining tasks has increased explosively. Data with extremely high dimensionality has presented serious challenges to existing learning methods [39], i.e., the curse of dimensionality [21]. With the presence of a large number of features, a learning model tends to overfit, resulting in their performance degenerates. To address the problem of the curse of dimensionality, dimensionality reduction techniques have been studied, which is an important branch in the machine learning and data mining research area. Feature selection is a widely employed technique for reducing dimensionality among practitioners. It aims to choose a small subset of the relevant features from the original ones according to certain relevance evaluation criterion, which usually leads to better learning performance (e.g., higher learning accuracy for classification), lower computational cost, and better model interpretability.

According to whether the training set is labelled or not, feature selection algorithms can be categorized into supervised [68, 61], unsupervised [13, 51] and semi-supervised feature selection [77, 71]. Supervised feature selection methods can further be broadly categorized into filter models, wrapper models and embedded models. The filter model separates feature selection from classifier learning so that the bias of a learning algorithm does not interact with the bias of a feature selection algorithm. It relies on measures of the general characteristics of the training data such as distance, consistency, dependency, information, and correlation. Relief [60], Fisher score [11] and Information Gain based methods [52] are among the most representative algorithms of the filter model. The wrapper model uses the predictive accuracy of a predetermined learning algorithm to determine the quality of selected features. These methods are prohibitively expensive to run for data with a large number of features. Due to these shortcomings in each model, the embedded model, was proposed to bridge the gap between the filter and wrapper models. First, it incorporates the statistical criteria, as filter model does, to select several candidate features subsets with a given cardinality. Second, it chooses the subset with the highest classification accuracy [40]. Thus, the embedded model usually achieves both comparable accuracy to the wrapper and comparable efficiency to the filter model. The embedded model performs feature selection in the learning time. In other words, it achieves model fitting and feature selection simultaneously [54, 15, 15]. Many researchers also paid attention to developing unsupervised feature selection. Unsupervised feature selection is a less constrained search problem without class labels, depending on clustering quality measures [12], and can eventuate many equally valid feature subsets. With high-dimensional data, it is unlikely to recover the relevant features without considering additional constraints. Another key difficulty is how to objectively measure the results of feature selection [12]. A comprehensive review about unsupervised feature selection can be found in [1]. Supervised feature selection assesses the relevance of features guided by the label information but a good selector needs enough labeled data, which is time consuming. While unsupervised feature selection works with unlabeled data but it is difficult to evaluate the relevance of features. It is common to have a data set with huge dimensionality but small labeled-sample size. High-dimensional data with small labeled samples permits too large a hypothesis space yet with too few constraints (labeled instances). The combination of the two data characteristics manifests a new research challenge. Under the assumption that labeled and unlabeled data are sampled from the same population generated by target

concept, semi-supervised feature selection makes use of both labeled and unlabeled data to estimate feature relevance [77].

Feature weighting is thought of as a generalization of feature selection [69]. In feature selection, a feature is assigned a binary weight, where 1 means the feature is selected and 0 otherwise. However, feature weighting assigns a value, usually in the interval [0,1] or [-1,1], to each feature. The greater this value is, the more salient the feature will be. Most of feature weight algorithms assign a unified (global) weight to each feature over all instances. However, the relative importance, relevance and noise in the different dimensions may vary significantly with data locality. There are local feature selection algorithms where the local selection of features is done specific to a test instance, which is is common in lazy leaning algorithms such as kNN [22, 9]. The idea is that feature selection or weighting is done at classification time (rather than at training time), because knowledge of the test instance sharpens the ability to select features.

Typically, a feature selection method consists of four basic steps [40], namely, subset generation, subset evaluation, stopping criterion, and result validation. In the first step, a candidate feature subset will be chosen based on a given search strategy, which is sent, in the second step, to be evaluated according to certain evaluation criterion. The subset that best fits the evaluation criterion will be chosen from all the candidates that have been evaluated after the stopping criterion are met. In the final step, the chosen subset will be validated using domain knowledge or a validation set.

0.1.3 Feature Selection for Classification

The majority of real-world classification problems require supervised learning where the underlying class probabilities and class-conditional probabilities are unknown, and each instance is associated with a class label [8]. In real-world situations, we often have little knowledge about relevant features. Therefore, to better represent the domain, many candidate features are introduced, resulting in the existence of irrelevant/redundant features to the target concept. A relevant feature is neither irrelevant nor redundant to the target concept; an irrelevant feature is not directly associate with the target concept but affect the learning process, and a redundant feature does not add anything new to the target concept [8]. In many classification problems, it is difficult to learn good classifiers before removing these unwanted features due to the huge size of the data. Reducing the number of irrelevant/redundant features can drastically reduce the running time of the learning algorithms and yields a more general classifier. This helps in getting a better insight into the underlying concept of a real-world classification problem.

A general feature selection for classification framework is demonstrated in Figure 3. Feature selection mainly affects the training phase of classification. After generating features, instead of processing data with the whole features to the learning algorithm directly, feature selection for classification will first perform feature selection to select a subset of features and then process the data with the selected features to the learning algorithm. The feature selection phase might be independent of the learning algorithm, like filter models, or it may iteratively utilize the performance of the learning algorithms to evaluate the quality of the selected features, like wrapper models. With the finally selected features, a classifier is induced for the prediction phase.

Usually feature selection for classification attempts to select the minimally sized subset of features according to the following criteria,

- the classification accuracy does not significantly decrease; and
- the resulting class distribution, given only the values for the selected features, is as close as possible to the original class distribution, given all features.

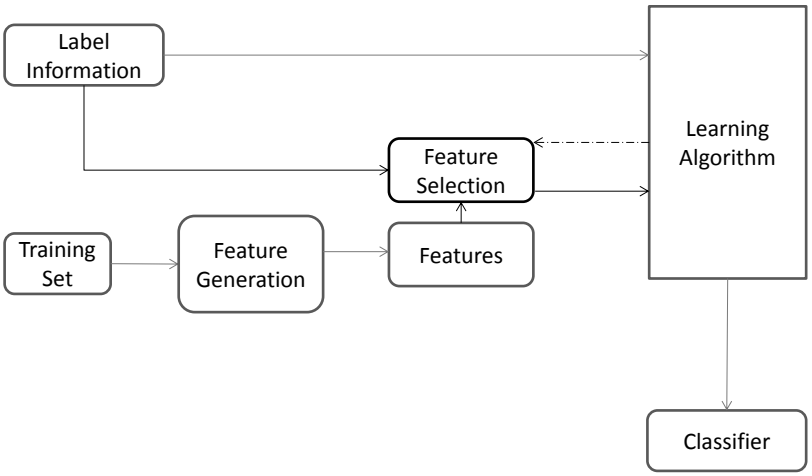


FIGURE 3: A General Framework of Feature Selection for Classification

Ideally, feature selection methods search through the subsets of features and try to find the best one among the competing 2^m candidate subsets according to some evaluation functions [8]. However this procedure is exhaustive as it tries to find only the best one. It may be too costly and practically prohibitive, even for a medium-sized feature set size (m). Other methods based on heuristic or random search methods attempt to reduce computational complexity by compromising performance. These methods need a stopping criterion to prevent an exhaustive search of subsets.

In this chapter, we divide feature selection for classification into three families according to the feature structure - methods for flat features, methods for structured features and methods for streaming features as demonstrated in Figure 4. In the following sections, we will review these three groups with representative algorithms in detail.

Before going to the next sections, we introduce notations we adopt in this book chapter. Assume that $\mathcal{F} = \{f_1, f_2, \ldots, f_m\}$ and $\mathcal{C} = \{c_1, c_2, \ldots, c_K\}$ denote the feature set and the class label set where m and K are the numbers of features and labels, respectively. $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_3\} \in \mathbb{R}^{m \times n}$ is the data where n is the number of instances and the label information of the i-th instance $\mathbf{x}_i$ is denoted as y_i.

0.2 Algorithms for Flat Features

In this section, we will review algorithms for flat features, where features are assumed to be independent. Algorithms in this category are usually further divided into three groups - filter models, wrapper models, and embedded models.

0.2.1 Filter Models

Relying on the characteristics of data, filter models evaluate features without utilizing any classification algorithms [39]. A typical filter algorithm consists of two steps. In the first step, it ranks features based on certain criteria. Feature evaluation could be either

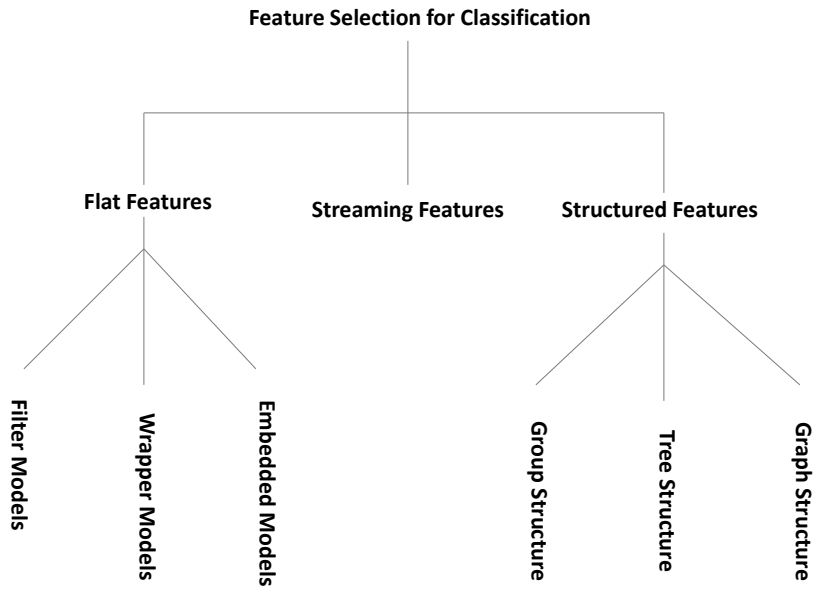


FIGURE 4: An Classification of Algorithms of Feature Selection for Classification.

univariate or *multivariate*. In the univariate scheme, each feature is ranked independently of the feature space, while the multivariate scheme evaluates features in an batch way. Therefore, the multivariate scheme is naturally capable of handling redundant features. In the second step, the features with highest rankings are chosen to induce classification models. In the past decade, a number of performance criteria have been proposed for filter-based feature selection such as Fisher score [10], methods based on mutual information [37, 74, 52] and ReliefF and its variants [34, 56].

Fisher Score [10]: Features with high quality should assign similar values to instances in the same class and different values to instances from different classes. With this intuition, the score for the i-th feature S_i will be calculated by Fisher Score as,

$$S_i = \frac{\sum_{k=1}^{K} n_j(\mu_{ij} - \mu_i)^2}{\sum_{k=1}^{K} n_j \rho_{ij}^2}, \tag{0.2}$$

where μ_{ij} and ρ_{ij} are the mean and the variance of the i-th feature in the j-th class respectively, n_j is the number of instances in the j-th class, and μ_i is the mean of the i-th feature.

Fisher Score evaluates features individually; therefore, it cannot handle feature redundancy. Recently, Gu et al. [17] proposed a generalized Fisher score to jointly select features, which aims to find an subset of features that maximize the lower bound of traditional Fisher score and solve the following problem:

$$\begin{aligned} &\|\mathbf{W}^\top diag(\mathbf{p})\mathbf{X} - \mathbf{G}\|_F^2 + \gamma\|\mathbf{W}\|_F^2, \\ &s.t., \quad \mathbf{p} \in \{0,1\}^m, \quad \mathbf{p}^\top \mathbf{1} = d, \end{aligned} \tag{0.3}$$

where $\mathbf{p}$ is the feature selection vector, d is the number of features to select, and $\mathbf{G}$ is a

special label indicator matrix, as follows:

$$\mathbf{G}(i,j) = \begin{cases} \sqrt{\frac{n}{n_j}} - \sqrt{\frac{n_j}{n}} & \text{if } \mathbf{x}_i \in c_j, \\ -\sqrt{\frac{n_j}{n}} & \text{otherwise} \end{cases} \tag{0.4}$$

Mutual Information based on Methods [37, 74, 52]: Due to its computational efficiency and simple interpretation, information gain is one of the most popular feature selection methods. It is used to measure the dependence between features and labels and calculates the information gain between the i-th feature f_i and the class labels $\mathcal{C}$as

$$IG(f_i, \mathcal{C}) = H(f_i) - H(f_i|\mathcal{C}), \tag{0.5}$$

where $H(f_i)$ is the entropy of f_i and $H(f_i|\mathcal{C})$ is the entropy of f_i after observing $\mathcal{C}$:

$$\begin{aligned} H(f_i) &= -\sum_j p(x_j) log_2(p(x_j)), \\ H(f_i|\mathcal{C}) &= -\sum_k p(c_k) \sum_j p(x_j|c_k) log_2(p(x_j|c_k)) \end{aligned} \tag{0.6}$$

In information gain, a feature is relevant if it has a high information gain. Features are selected in a univariate way, therefore, information gain cannot handle redundant features. In [74], a fast filter method FCBF based on mutual information was proposed to identify relevant features as well as redundancy among relevant features and measure feature-class and feature-feature correlation. Given a threshold ρ, FCBC first selects a set of feature S which is highly correlated to the class with $SU \geq \rho$, where SU is symmetrical uncertainty defined as

$$SU(f_i, \mathcal{C}) = 2\frac{IG(f_i, \mathcal{C})}{H(f_i) + H(\mathcal{C})}. \tag{0.7}$$

A feature f_i is called predominant *iff* $SU(f_i, c_k) \geq \rho$ and there is no $f_j (f_j \in S, j \neq i)$ such as $SU(j, i) \geq SU(i, c_k)$. f_j is a redundant feature to f_i if $SU(j, i) \geq SU(i, c_k)$. Then the set of redundant features is denoted as $S_{(P_i)}$, which is further split into $S^+_{p_i}$ and $S^-_{p_i}$. $S^+_{p_i}$ and $S^-_{p_i}$ contain redundant features to f_i with $SU(j, c_k) > SU(i, c_k)$ and $SU(j, c_k) \leq SU(i, c_k)$, respectively. Finally FCBC applied three heuristics on $S_(P_i)$, $S^+_{p_i}$ and $S^-_{p_i}$ to remove the redundant features and keep the feature that most relevant to the class. FCBC provides an effective way to handle feature redundancy in feature selection.

Minimum-Redundancy-Maximum-Relevance (mRmR) is also a mutual information based method and it selects features according to the maximal statistical dependency criterion [52]. Due to the difficulty in directly implementing the maximal dependency condition, mRmR is an approximation to maximizing the dependency between the joint distribution of the selected features and the classification variable. *Minimize Redundancy* for discrete features and continuous features are defined as,

$$\begin{aligned} &\text{For Discrete Features: } \min W_I, \quad W_I = \frac{1}{|S|^2} \sum_{i,j \in S} I(i,j), \\ &\text{For Continuous Features: } \min W_c, \quad W_c = \frac{1}{|S|^2} \sum_{i,j \in S} |C(i,j)| \end{aligned} \tag{0.8}$$

where $I(i, j)$ and $C(i, j)$ are mutual information and the correlation between f_i and f_j,

respectively. While *Maximize Relevance* for discrete features and continuous features are defined as,

$$\begin{aligned} &\text{For Discrete Features: } \max V_I, \quad V_I = \frac{1}{|S|^2}\sum_{i\in S} I(h,i), \\ &\text{For Continuous Features: } \max V_c, \quad V_c = \frac{1}{|S|^2}\sum_{i} F(i,h) \end{aligned} \tag{0.9}$$

where h is the target class and $F(i,h)$ is the F-statistic.

ReliefF [34, 56]: Relief and its multi-class extension ReliefF select features to separate instance from different classes. Assume that ℓ instances are randomly sampled from the data and then the score of the i-th feature S_i is defined by Relief as,

$$S_i = \frac{1}{2}\sum_{k=1}^{\ell} d(\mathbf{X}_{ik} - \mathbf{X}_{iM_k}) - d(\mathbf{X}_{ik} - \mathbf{X}_{iH_k}), \tag{0.10}$$

where M_k denotes the values on the i-th feature of the nearest instances to $\mathbf{x}_k$ with the same class label, while H_k denotes the values on the i-th feature of the nearest instances to $\mathbf{x}_k$ with different class labels. $d(\cdot)$ is a distance measure. To handle multi-class problem, Eq. (0.10) is extended as,

$$S_i = \frac{1}{K}\sum_{k=1}^{\ell}\big(-\frac{1}{m_k}\sum_{\mathbf{x}_j\in M_k} d(\mathbf{X}_{ik} - \mathbf{X}_{ij}) + \sum_{y\neq y_k}\frac{1}{h_{ky}}\frac{p(y)}{1-p(y)}\sum_{\mathbf{x}_j\in H_k} d(\mathbf{X}_{ik} - \mathbf{X}_{ij})\big) \tag{0.11}$$

where M_k and H_{ky} denotes the sets of nearest points to $\mathbf{x}_k$ with the same class and the class y with sizes of m_k and h_{ky} respectively, and $p(y)$ is the probability of an instance from the class y. In [56], the authors related the relevance evaluation criterion of ReliefF to the hypothesis of margin maximization, which explains why the algorithm provide superior performance in many applications.

0.2.2 Wrapper Models

Filter models select features independent of any specific classifiers. However the major disadvantage of the filter approach is that it totally ignores the effects of the selected feature subset on the performance of the induction algorithm [36, 20]. The optimal feature subset should depend on the specific biases and heuristics of the induction algorithm. Based on this assumption, wrapper models utilize a specific classifier to evaluate the quality of selected features, and offer a simple and powerful way to address the problem of feature selection, regardless of the chosen learning machine [36, 26]. Given a predefined classifier, a typical wrapper model will perform the following steps:

- *Step 1:* searching a subset of features,
- *Step 2:* evaluating the selected subset of features by the performance of the classifier,
- *Step 3:* repeating *Step 1* and *Step 2* until the desired quality is reached.

A general framework for wrapper methods of feature selection for classification [36] is shown in Figure 5, and it contains three major components:

- Feature selection search - how to search the subset of features from all possible feature subsets,

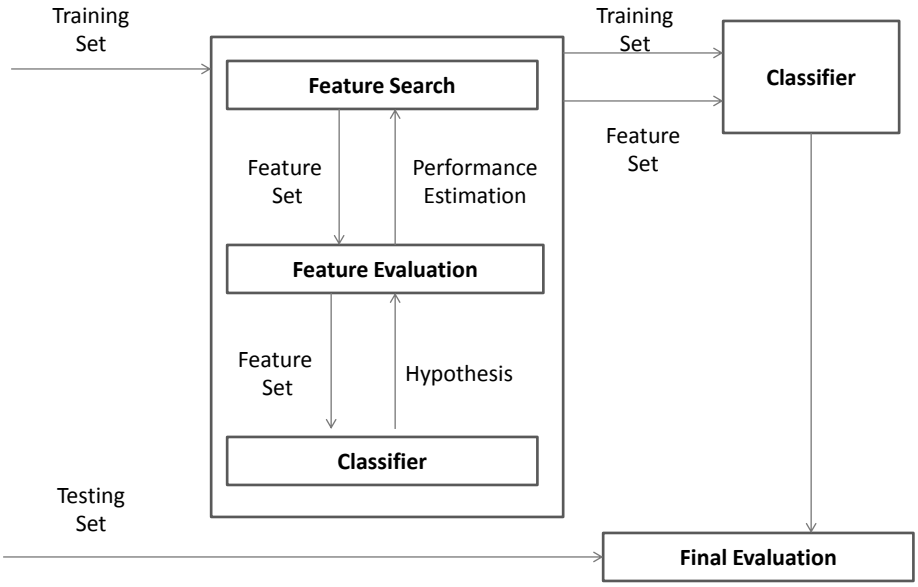


FIGURE 5: A General Framework for Wrapper Methods of Feature Selection for Classification

- Feature evaluation - how to evaluate the performance of the chosen classifier, and
- Induction Algorithm.

In wrapper models, the predefined classifier works as a black box. The feature search component will produce a set of features and the feature evaluation component will use the classifier to estimate the performance, which will be returned back to the feature search component for the next iteration of feature subset selection. The feature set with the highest estimated value will be chosen as the final set to learn the classifier. The resulting classifier is then evaluated on an independent testing set that is not used in during the training process [36].

The size of search space for m features is $O(2^m)$, thus an exhaustive search is impractical unless m is small. Actually the problem is known to be NP-hard [18]. A wide range of search strategies can be used, including hill-climbing, best-first, branch-and-bound, and genetic algorithms [18]. The Hill-climbing strategy expands the current set and moves to the subset with the highest accuracy, terminating when no subset improves over the current set. The best-first strategy is to select the most promising set that has not already been expanded and is a more robust method than hill-climbing [36]. Greedy search strategies seem to be particularly computationally advantageous and robust against overfitting. They come in two flavors - forward selection and backward elimination. Forward selection refers to a search that begins at the empty set of features and features are progressively incorporated into larger and larger subsets, whereas backward elimination begins with the full set of features and progressively eliminates the least promising ones. The search component aims to find a subset of features with the highest evaluation, using a heuristic function to guide it. Since we do not know the actual accuracy of the classifier, we use accuracy estimation as both the heuristic function and the valuation function in the feature evaluation phase. Performance assessments are usually done using a validation set or by cross-validation.

Wrapper models obtain better predictive accuracy estimates than filter models [36, 38]. However, wrapper models are very computationally expensive compared to filter models. It produces better performance for the predefined classifier since we aim to select features that maximize the quality therefore the selected subset of features is inevitably biased to the predefined classifier.

0.2.3 Embedded Models

Filter models select features that are independent of the classifier and avoid the cross-validation step in a typical wrapper model, therefore they are computationally efficient. However, they do not take into account the biases of the classifiers. For example, the relevance measure of Relief would not be appropriate as feature subset selectors for Naive-Bayes because in many cases the performance of Naive-Bayes improves with the removal of relevant features [18]. Wrapper models utilize a predefined classifier to evaluate the quality of features and representational biases of the classifier are avoided by the feature selection process. However, they have to run the classifier many times to assess the quality of selected subsets of features, which is very computationally expensive. Embedded Models embedding feature selection with classifier construction, have the advantages of (1) wrapper models - they include the interaction with the classification model and (2) filter models - they are far less computationally intensive than wrapper methods [40, 57, 46].

There are three types of embedded methods. The first are pruning methods that first utilizing all features to train a model and then attempt to eliminate some features by setting the corresponding coefficients to 0, while maintaining model performance such as recursive feature elimination using support vector machine (SVM) [19]. The second are models with a build-in mechanism for feature selection such as ID3 [55] and C4.5 [54]. The third are regularization models with objective functions that minimize fitting errors and in the mean time force the coefficients to be small or to be exact zero. Features with coefficients that are close to 0 are then eliminated [46]. Due to good performance, regularization models attract increasing attention. We will review some representative methods below based on a survey paper of embedded models based on regularization [46].

Without loss of generality, in this section, we only consider linear classifiers $\mathbf{w}$ in which classification of $\mathbf{Y}$ can be based on a linear combination of $\mathbf{X}$ such as SVM and logistic regression. In regularization methods, classifier induction and feature selection are achieved simultaneously by estimating $\mathbf{w}$ with properly tuned penalties. The learned classifier $\mathbf{w}$ can have coefficients exactly equal to zero. Since each coefficient of $\mathbf{w}$ corresponds to one feature such as $\mathbf{w}_i$ for f_i, feature selection is achieved and only features with nonzero coefficients in $\mathbf{w}$ will be used in the classifier. Specifically, we define $\hat{\mathbf{w}}$ as,

$$\hat{\mathbf{w}} = \min_{\mathbf{w}} c(\mathbf{w}, \mathbf{X}) + \alpha \; penalty(\mathbf{w}) \tag{0.12}$$

where $c(\cdot)$ is the classification objective function, $penalty(\mathbf{w})$ is a regularization term, and α is the regularization parameter controlling the trade-off between the $c(\cdot)$ and the penalty. Popular choices of $c(\cdot)$ include quadratic loss such as least squares, hinge loss such as ℓ_1SVM [5] and logistic loss as BlogReg [15] as

- *Quadratic loss:*

$$c(\mathbf{w}, \mathbf{X}) = \sum_{i=1}^{n} (y_i - \mathbf{w}^\top \mathbf{x}_i)^2, \tag{0.13}$$

- *Hinge loss:*

$$c(\mathbf{w}, \mathbf{X}) = \sum_{i=1}^{n} \max(0, 1 - y_i \mathbf{w}^\top \mathbf{x}_i), \tag{0.14}$$

- *Logistic loss:*

$$c(\mathbf{w}, \mathbf{X}) = \sum_{i=1}^{n} log(1 + exp(-y_i(\mathbf{w}^\top \mathbf{x}_i + b))). \tag{0.15}$$

Lasso Regularization [64]: Lasso regularization is based on ℓ_1-norm of the coefficient of $\mathbf{w}$ and defined as

$$penalty(\mathbf{w}) = \sum_{i=1}^{m} |\mathbf{w}_i|. \tag{0.16}$$

An important property of the ℓ_1 regularization is that it can generate an estimation of $\mathbf{w}$ [64] with exact zero coefficients. In other words, there are zero entities in $\mathbf{w}$, which denotes that the corresponding features are eliminated during the classifier learning process. Therefore, it can be used for feature selection.

Adaptive Lasso [80]: The Lasso feature selection is consistent if the underlying model satisfies a non-trivial condition, which may not be satisfied in practice [76]. Meanwhile the Lasso shrinkage produces biased estimates for the large coefficients, thus, it could be suboptimal in terms of estimation risk [14].

The adaptive Lasso is proposed to improve the performance of as [80]

$$penalty(\mathbf{w}) = \sum_{i=1}^{m} \frac{1}{\mathbf{b}_i} |\mathbf{w}_i|, \tag{0.17}$$

where the only difference between Lasso and adaptive Lasso is that the latter employs a weighted adjustment $\mathbf{b}_i$ for each coefficient $\mathbf{w}_i$. The article shows that the adaptive lasso enjoys the oracle properties and can be solved by the same efficient algorithm for solving the Lasso.

The article also proves that for linear models $\mathbf{w}$ with $n \gg m$, the adaptive Lasso estimate is selection consistent under very general conditions if $\mathbf{b}_i$ is a $\sqrt{n}$ consistent estimate of $\mathbf{w}_i$. Complimentary to this proof, [24] shows that when $m \gg n$ for linear models, the adaptive Lasso estimate is also selection consistent under a partial orthogonality condition in which the covariates with zero coefficients are weakly correlated with the covariates with nonzero coefficients.

Bridge regularization [35, 23]: Bridge regularization is formally defined as

$$penalty(\mathbf{w}) = \sum_{i=1}^{n} |\mathbf{w}_i|^\gamma, \quad 0 \leq \gamma \leq 1 \tag{0.18}$$

Lasso regularization is a special case of bridge regularization when $\gamma = 1$.

For liner models, the bridge regularization is feature selection consistent, even when the Lasso is not [23] when $n \gg m$ and $\gamma < 1$; the regularization is still feature selection consistent if the features associated with the phenotype and those not associated with the phenotype are only weakly correlated when $n \gg m$ and $\gamma < 1$.

Elastic net regularization [81]: In practice, it is common that a few features are highly correlated. In this situation, the Lasso tends to select only one of the correlated

features [81]. To handle features with high correlations, elastic net regularization is proposed as

$$penalty(\mathbf{w}) = \sum_{i=1}^{n} |\mathbf{w}_i|^{\gamma} + (\sum_{i=1}^{n} \mathbf{w}_i^2)^{\lambda}, \tag{0.19}$$

with $0 < \gamma \leq 1$ and $\lambda \geq 1$. The elastic net is a mixture of bridge regularization with different values of γ. [81] proposes $\gamma = 1$ and $\lambda = 1$, which is extended to $\gamma < 1$ and $\lambda = 1$ by [43].

Through the loss function $c(\mathbf{w}, \mathbf{X})$, above mentioned methods control the size of residuals. An alternative way to obtain a sparse estimation of $\mathbf{w}$ is Dantzig selector, which is based on the normal score equations and controls the correlation of residuals with $\mathbf{X}$ as [6],

$$\min \quad \|\mathbf{w}\|_1, \quad s.t. \ \ \|\mathbf{X}^{\top}(\mathbf{y} - \mathbf{w}^{\top}\mathbf{X})\|_{\infty} \leq \lambda, \tag{0.20}$$

$\|\cdot\|_{\infty}$ is the ℓ_{∞}-norm of a vector and Dantzig selector was designed for linear regression models. Candes and Tao have provided strong theoretical justification for this performance by establishing sharp non-asymptotic bounds on the ℓ_2-error in the estimated coefficients, and showed that the error is within a factor of $\log(p)$ of the error that would be achieved if the locations of the non-zero coefficients were known [6, 28]. Strong theoretical results show that LASSO and Dantzig selector are closely related [28].

0.3 Algorithms for Structured Features

The models introduced in the last section assume that features are independent and totally overlook the feature structures [73]. However, for many real-world applications, the features exhibit certain intrinsic structures, e.g., spatial or temporal smoothness [65, 79], disjoint/overlapping groups [29], trees [33], and graphs [25]. Incorporating knowledge about the structures of features may significantly improve the classification performance and help identify the important features. For example, in the study of arrayCGH [65, 66], the features (the DNA copy numbers along the genome) have the natural spatial order, and incorporating the structure information using an extension of the ℓ_1-norm outperforms the Lasso in both classification and feature selection. In this section, we review feature selection algorithms for structured features and these structures include group, tree and graph.

Since most existing algorithms in this category are based on linear classifiers, we focus on linear classifiers such as SVM and logistic classifier in this section. A very popular and successful approach to learn linear classifiers with structured features is to minimize a empirical error penalized by a regularization term as

$$\min_{\mathbf{w}} \quad c(\mathbf{w}^{\top}\mathbf{X}, \mathbf{Y}) + \alpha \ penalty(\mathbf{w}, \mathcal{G}), \tag{0.21}$$

where $\mathcal{G}$ denotes the structure of features, and α controls the trade-off between data fitting and regularization. Eq. (0.21) will lead to sparse classifiers, which lend themselves particularly well to interpretation, which is often of primary importance in many applications such as biology or social sciences [75].

0.3.1 Features with Group Structure

In many real-world applications, features form group structures. For example, in the multifactor analysis-of-variance (ANOVA) problem, each factor may have several levels and

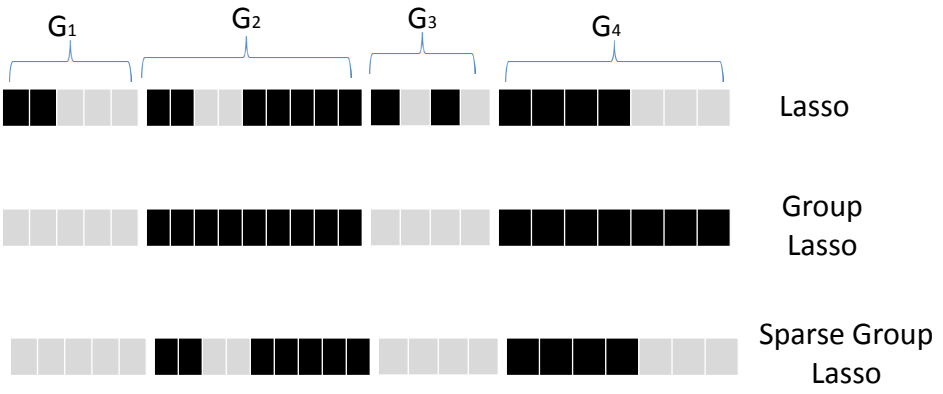


FIGURE 6: Illustration of Lasso, Group Lasso and Sparse Group Lasso. Features can be grouped into 4 disjoint groups $\{G_1, G_2, G_3, G_4\}$. Each cell denotes a feature and light color represents the corresponding cell with coefficient zero.

can be denoted as a group of dummy features [75]; in speed and signal processing, different frequency bands can be represented by groups [49]. When performing feature selection, we tend to select or not select features in the same group simultaneously. Group Lasso, driving all coefficients in one group to zero together and thus resulting in group selection attracts more and more attention [75, 3, 27, 41, 50].

Assume that features form k disjoint groups $\mathcal{G} = \{G_1, G_2, \ldots, G_k\}$ and there is no overlap between any two groups. With the group structure, we can rewrite $\mathbf{w}$ into the block form as $\mathbf{w} = \{\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_k\}$ where $\mathbf{w}_i$ corresponds to the vector of all coefficients of features in the i-th group G_i. Then, the group Lasso performs the $\ell_{q,1}$-norm regularization on the model parameters as

$$penalty(\mathbf{w}, \mathcal{G}) = \sum_{i=1}^{k} h_i \|\mathbf{w}_{G_i}\|_q, \tag{0.22}$$

where $\|\cdot\|_q$ is the ℓ_q-norm with $q > 1$, and h_i is the weight for the i-th group. Lasso does not take group structure information into account and does not support group selection, while group lasso can select or not select a group of features as a whole.

Once a group is selected by the group Lasso, all features in the group will be selected. For certain applications, it is also desirable to select features from the selected groups, i.e., performing simultaneous group selection and feature selection. The sparse group Lasso takes advantages of both Lasso and group Lasso, and it produces a solution with simultaneous between- and within- group sparsity. The sparse group Lasso regularization is based on a composition of the $\ell_{q,1}$-norm and the ℓ_1-norm,

$$penalty(\mathbf{w}, \mathcal{G}) = \alpha\|\mathbf{w}\|_1 + (1-\alpha)\sum_{i=1}^{k} h_i \|\mathbf{w}_{G_i}\|_q, \tag{0.23}$$

where $\alpha \in [0, 1]$, the first term controls the sparsity in the feature level, and the second term controls the group selection.

Figure 6 demonstrates the different solutions among Lasso, group Lasso and sparse group Lasso. In the figure, features form 4 groups $\{G_1, G_2, G_3, G_4\}$. Light color denotes the corresponding feature of the cell with zero coefficients and Dark color indicates non-zero coefficients. From the figure, we observe that

- Lasso does not consider the group structure and selects a subset of features among all groups;

- Group Lasso can perform group selection and select a subset of groups. Once the group is selected, all features in this group are selected; and

- Sparse group Lasso can select groups and features in the selected groups at the same time.

In some applications, the groups overlap. One motivation example is the use of biologically meaningful gene/protein sets (groups) given by [73]. If the proteins/genes appear either appear in the same pathway, or are semantically related in terms of Gene Ontology (GO) hierarchy, or are related from gene set enrichment analysis(GSEA), they are related and assigned to the same groups. For example, the canonical pathway in MSigDB has provided 639 groups of genes. It has been shown that the group (of proteins/genes) markers are more reproducible than individual protein/gene markers and modeling such group information as prior knowledge can improve classification performance [7]. Groups may overlap - one protein/gene may belong to multiple groups. In these situations, group Lasso does not correctly handle overlapping groups and a given coefficient only belongs to one group. Algorithms investigating overlapping groups are proposed as [27, 30, 33, 42]. A general overlapping group Lasso regularization is similar to that for group Lasso regularization in Eq. (0.23)

$$penalty(\mathbf{w}, \mathcal{G}) = \alpha\|\mathbf{w}\|_1 + (1-\alpha)\sum_{i=1}^{k} h_i \|\mathbf{w}_{G_i}\|_q, \tag{0.24}$$

however, groups for overlapping group Lasso regularization may overlap, while groups in group Lasso are disjoint.

0.3.2 Features with Tree Structure

In many applications, features can naturally be represented using certain tree structures. For example, the image pixels of the face image can be represented as a tree, where each parent node contains a series of child nodes that enjoy spatial locality; genes/proteins may form certain hierarchical tree structures [42]. Tree-guided group Lasso regularization is proposed for features represented as an index tree [33, 42, 30].

In the index tree, each leaf node represents a feature and each internal node denotes the group of the features that correspond to the leaf nodes of the subtree rooted at the given internal node. Each internal node in the tree is associated with a weight that represents the height of the subtree, or how tightly the features in the group for that internal node are correlated, which can be formally defined as follows [42].

For an index tree $\mathcal{G}$ of depth d, let $\mathcal{G}_i = \{G_1^i, G_2^i, \ldots, G_{n_i}^i\}$ contain all the nodes corresponding to depth i where n_i is the number of nodes of the depth i. The nodes satisfy the following conditions

- the nodes from the same depth level have non-overlapping indices, i.e., $G_j^i \cap G_k^i = \emptyset,\ \forall i \in \{1, 2, \ldots, d\},\ j \neq k,\ 1 \leq j, k \leq n_i$;

- let $G_{j_0}^{i-1}$ be the parent node of a non-root node G_j^i, then $G_j^i \subseteq G_{j_0}^{i-1}$

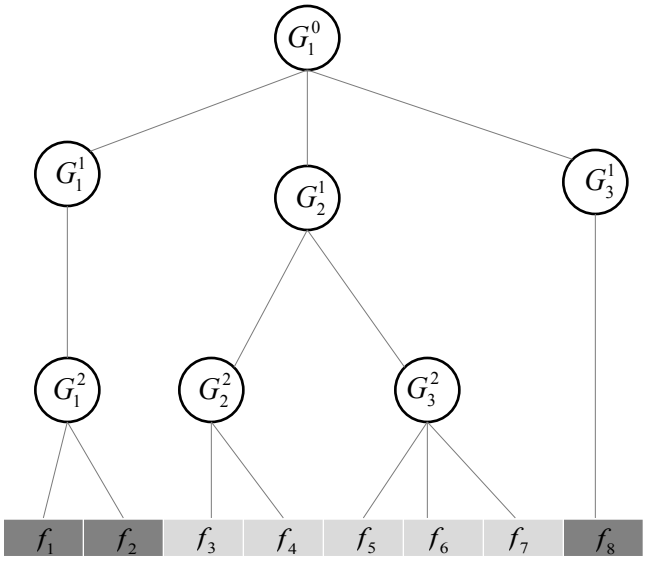


FIGURE 7: An illustration of a simple index tree of depth 3 with 8 features

Figure 7 shows a sample index tree of depth 3 with 8 features, where G_j^i are defined as

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

We can observe that

- G_1^0 contains all features;
- the index sets from different nodes may overlap, e.g., any parent node overlaps with its child nodes;
- the nodes from the same depth level do not overlap; and
- the index set of a child node is a subset of that of its parent node.

With the definition of the index tree, the tree-guided group Lasso regularization is,

$$
penalty(\mathbf{w}, \mathcal{G}) = \sum_{i=0}^{d} \sum_{j=1}^{n_i} h_j^i \|\mathbf{w}_{G_j^i}\|_q, \tag{0.25}
$$

Since any parent node overlaps with its child nodes. Thus, if a specific node is not selected (i.e., its corresponding model coefficient is zero), then all its child node will not be selected. For example, in Figure 7, if G_2^1 is not selected, both G_2^2 and G_3^2 will not be selected, indicating that features $\{f_3, f_4, f_5, f_6, f_7\}$ will be not selected. Note that the tree structured group Lasso is a special case of the overlapping group Lasso with a specific tree structure.

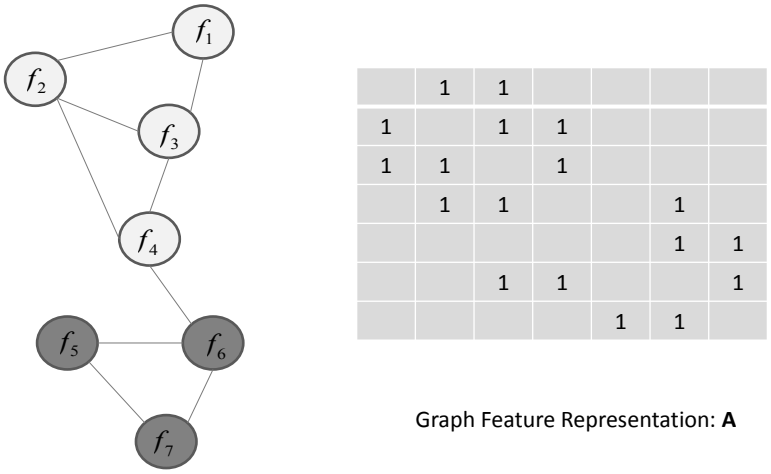


FIGURE 8: An illustration of the graph of 7 features $\{f_1, f_2, \ldots, f_7\}$ and its representation $\mathbf{A}$

0.3.3 Features with Graph Structure

We often have knowledge about pair-wise dependencies between features in many real-world applications [58]. For example, in natural language processing, digital lexicons such as WordNet can indicate which words are synonyms or antonyms; many biological studies have suggested that genes tend to work in groups according to their biological functions, and there are some regulatory relationships between genes. In these cases, features form an undirected graph, where the nodes represent the features, and the edges imply the relationships between features. Several recent studies have shown that the estimation accuracy can be improved using dependency information encoded as a graph.

Let $\mathcal{G}(N, E)$ be a given graph where $N = \{1, 2, \ldots, m\}$ is a set of nodes, and E is a set of edges. Node i corresponds to the i-th feature and we use $\mathbf{A} \in \mathbb{R}^{m \times m}$ to denote the adjacency matrix of $\mathcal{G}$. Figure 8 shows an example of the graph of 7 features $\{f_1, f_2, \ldots, f_7\}$ and its representation $\mathbf{A}$.

If nodes i and j are connected by an edge in E, then the i-th feature and the j-th feature are more likely to be selected together, and they should have similar weights. One intuitive way to formulate graph lasso is to force weights of two connected features close by a squre loss as

$$penalty(\mathbf{w}, \mathcal{G}) = \lambda \|\mathbf{w}\|_1 + (1 - \lambda) \sum_{i,j} \mathbf{A}_{ij} (\mathbf{w}_i - \mathbf{w}_j)^2, \tag{0.26}$$

which is equivalent to

$$penalty(\mathbf{w}, \mathcal{G}) = \lambda \|\mathbf{w}\|_1 + (1 - \lambda) \mathbf{w}^\top \mathcal{L} \mathbf{w}, \tag{0.27}$$

where $\mathcal{L} = \mathbf{D} - \mathbf{A}$ is the Laplacian matrix and $\mathbf{D}$ is a diagonal matrix with $\mathbf{D}_{ii} = \sum_{j=1}^{m} \mathbf{A}_{ij}$. The Laplacian matrix is positive semi-definite and captures the underlying local geometric structure of the data. When $\mathcal{L}$ is an identity matrix, $\mathbf{w}^\top \mathcal{L} \mathbf{w} = \|\mathbf{w}\|_2^2$ and then Eq. (0.27) reduces to the elastic net penalty [81]. Because $\mathbf{w}^\top \mathcal{L} \mathbf{w}$ is both convex and differentiable, existing efficient algorithms for solving the Lasso can be applied to solve Eq. (0.27).

Eq. (0.27) assumes that the feature graph is unsigned, and encourages positive correlation between the values of coefficients for the features connected by an edge in the unsigned graph. However, two features might be negatively correlated. In this situation, the feature graph is signed, with both positive and negative edges. To perform feature selection with a signed feature graph, GFlasso employs a different ℓ_1 regularization over a graph [32],

$$penalty(\mathbf{w}, \mathcal{G}) = \lambda\|\mathbf{w}\|_1 + (1-\lambda)\sum_{i,j}\mathbf{A}_{ij}\|\mathbf{w}_i - sign(r_{ij})\mathbf{x}_j\|, \tag{0.28}$$

where r_{ij} is the correlation between two features. When f_i and f_j are positively connected $r_{ij} > 0$, i.e., with a positive edge, $penalty(\mathbf{w}, \mathcal{G})$ forces the coefficients $\mathbf{w}_i$ and $\mathbf{w}_j$ to be similar, while f_i and f_j are negatively connected $r_{ij} < 0$, i.e., with a negative edge, the penalty forces $\mathbf{w}_i$ and $\mathbf{w}_j$ to be dissimilar. Due to possible graph misspecification, GFlasso may introduce additional estimation bias in the learning process. For example, additional bias may occur when the sign of the edge between f_i and f_j is inaccurately estimated.

In [72], the authors introduced several alternative formulations for graph Lasso. One of the formulations is defined as

$$penalty(\mathbf{w}, \mathcal{G}) = \lambda\|\mathbf{w}\|_1 + (1-\lambda)\sum_{i,j}\mathbf{A}_{ij}\max(|\mathbf{w}_i|, |\mathbf{w}_j|), \tag{0.29}$$

where a pairwise ℓ_∞ regularization is used to force the coefficients to be equal and the grouping constraints are only put on connected nodes with $\mathbf{A}_{ij} = 1$. The ℓ_1-norm of $\mathbf{w}$ encourages sparseness, and $\max(|\mathbf{w}_i|, |\mathbf{w}_j|)$ will penalize the larger coefficients, which can be decomposed as

$$\max(|\mathbf{w}_i|, |\mathbf{w}_j|) = \frac{1}{2}(|\mathbf{w}_i + \mathbf{w}_j| + |\mathbf{w}_i - \mathbf{w}_j|), \tag{0.30}$$

which can be further represented as

$$\max(|\mathbf{w}_i|, |\mathbf{w}_j|) = |\mathbf{u}^\top\mathbf{w}| + |\mathbf{v}^\top\mathbf{w}|, \tag{0.31}$$

where $\mathbf{u}$ and $\mathbf{v}$ are two vectors with only two non-zero entities, i.e., $\mathbf{u}_i = \mathbf{u}_j = \frac{1}{2}$ and $\mathbf{v}_i = -\mathbf{v}_j = \frac{1}{2}$.

The GOSCAR formulation is closely related to OSCAR in [4]. However, OSCAR assumes that all features form a complete graph, which means that the feature graph is complete. OSCAR works for $\mathbf{A}$ whose entities are all 1, while GOSCAR can work with an arbitrary undirected graph where $\mathbf{A}$ is any symmetric matrix. In this sense, GOSCAR is more general. Meanwhile, the formulation for GOSCAR is much more challenging to solve than that of OSCAR.

The limitation of the Laplacian Lasso that the different signs of coefficients can introduce additional penalty can be overcame by the grouping penalty of GOSCAR. However, GOSCAR can easily over penalize the coefficient $\mathbf{w}_i$ or $\mathbf{w}_j$ due to the property of the max operator. The additional penalty would result in biased estimation, especially for large coefficients. As mentioned above, GFlasso will introduce estimation bias when the sign between $\mathbf{w}_i$ and $\mathbf{w}_j$ is wrongly estimated. This motivates the following non-convex formulation for graph features,

$$penalty(\mathbf{w}, \mathcal{G}) = \lambda\|\mathbf{w}\|_1 + (1-\lambda)\sum_{i,j}\mathbf{A}_{ij}\||\mathbf{w}_i| - |\mathbf{w}_j|\|, \tag{0.32}$$

where the grouping penalty $\sum_{i,j}\mathbf{A}_{ij}\||\mathbf{w}_i| - |\mathbf{w}_j|\|$ controls only magnitudes of differences of

coefficients while ignoring their signs over the graph. Via the ℓ_1 regularization and grouping penalty, feature grouping and selection are performed simultaneously where only large coefficients as well as pair wise difference are shrunk [72].

For features with graph structure, a subset of highly connected features in the graph is likely to be selected or not selected as a whole. For example, in Figure 8, $\{f_5, f_6, f_7\}$ are selected, while $\{f_1, f_2, f_3, f_4\}$ are not selected.

0.4 Algorithms for Streaming Features

Methods introduced above assume that all features are known in advance, and another interesting scenario is taken into account where candidate features are sequentially presented to the classifier for potential inclusion in the model [53, 78, 70, 67]. In this scenario, the candidate features are generated dynamically and the size of features is unknown. We call this kind of features as streaming features and feature selection for streaming features is called streaming feature selection. Streaming feature selection has practical significance in many applications. For example, the famous microblogging website Twitter produces more than 250 millions tweets per day and many new words (features) are generated such as abbreviations. When performing feature selection for tweets, it is not practical to wait until all features have been generated, thus it could be more preferable to streaming feature selection.

A general framework for streaming feature selection is presented in Figure 9. A typical streaming feature selection will perform the following steps,

- *Step 1:* Generating a new feature;
- *Step 2:* Determining whether adding the newly generated feature to the set of currently selected features;
- *Step 3:* Determining whether removing features from the set of currently selected features;
- *Step 4:* Repeat *Step 1* to *Step 3*.

Different algorithms may have different implementations for *Step 2* and *Step 3*, and next we will review some representative methods in this category. Note that *Step 3* is optional and some streaming feature selection algorithms only implement *Step 2*.

0.4.1 The Grafting Algorithm

Perkins and Theiler proposed a streaming feature selection framework based on grafting, which is a general technique that can be applied to a variety of models that are parameterized by a weight vector $\mathbf{w}$, subject to ℓ_1 regularization, such as the lasso regularized feature selection framework in the Section 0.2.3 as,

$$\hat{\mathbf{w}} = \min_{\mathbf{w}} c(\mathbf{w}, \mathbf{X}) + \alpha \sum_{j=1}^{m} |\mathbf{w}_j| \tag{0.33}$$

when all features are available, $penalty(\mathbf{w})$ penalizes all weights in $\mathbf{w}$ uniformly to achieve feature selection, which can be applied to streaming features with the grafting technique.

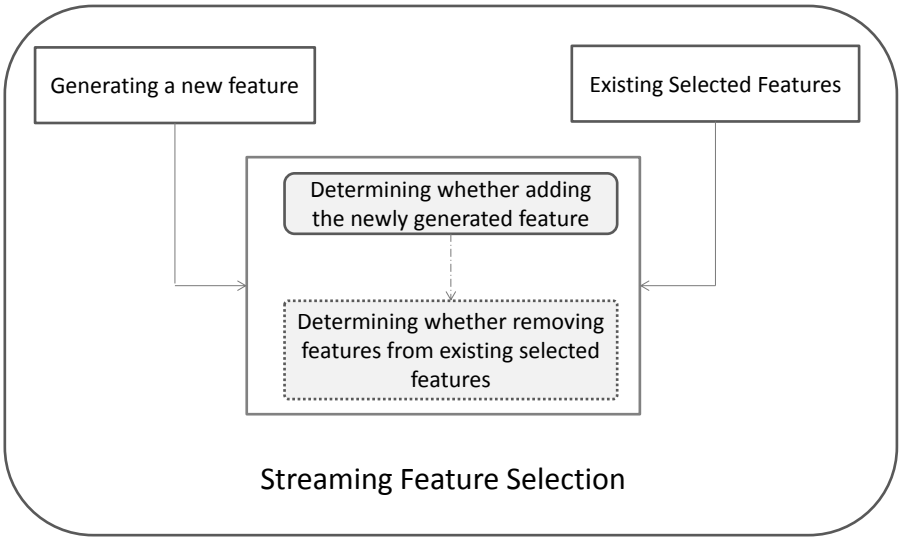


FIGURE 9: A general framework for streaming feature selection.

In Eq. (0.33), every one-zero weight $\mathbf{w}_j$ added to the model incurs a regularize penalty of $\alpha|\mathbf{w}_j|$. Therefore the feature adding to the model only happens when the loss of $c(\cdot)$ is larger than the regularizer penalty. The grafting technique will only take $\mathbf{w}_j$ away from zero if:

$$\frac{\partial c}{\partial \mathbf{w}_j} > \alpha, \tag{0.34}$$

otherwise the grafting technique will set the weight to zero (or excluding the feature).

0.4.2 The Alpha-investing Algorithm

α-investing controls the false discovery rate by dynamically adjusting a threshold on the p-statistic for a new feature to enter the model [78], which is described as

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

$$\begin{aligned} &w_{i+1} = w_i - \alpha_i \qquad \text{if } pvalue(x_i, SF) \geq \alpha_i. \\ &w_{i+1} = w_i + \alpha_\triangle - \alpha_i, \quad SF = SF \cup f_i \quad \text{otherwise} \end{aligned}$$

- *Step 4:* $i = i + 1$
- *Step 5:* Repeat *Step 1* to *Step 3*.

where the threshold α_i is the probability of selecting a spurious feature in the i-th step and it is adjusted using the wealth w_i, which denotes the current acceptable number of future false positives. Wealth is increased when a feature is selected to the model, while wealth

is decreased when a feature is not selected to the model. The p-value is the probability that a feature coefficient could be judged to be non-zero when it is actually zero, which is calculated by using the fact that $\triangle$-Logliklohood is equivalent to t-statistics. The idea of α-investing is to adaptively control the threshold for selecting features so that when new features are selected to the model, one "invests" α increasing the wealth raising the threshold, and allowing a slightly higher future chance of incorrect inclusion of features. Each time a feature is tested and found not to be significant, wealth is "spent", reducing the threshold so as to keep the guarantee of not selecting more than a target fraction of spurious features.

0.4.3 The Online Streaming Feature Selection Algorithm

To determine the value of α, the grafting algorithm requires all candidate features in advance, while the α-investing algorithm needs some prior knowledge about the structure of the feature space to heuristically control the choice of candidate feature selection and it is difficult to obtain sufficient prior information about the structure of the candidate features with a feature stream. Therefore the online streaming feature selection algorithm (OSFS) is proposed to solving these challenging issue in streaming feature selection [70].

An entire feature set can be divided into four basic disjoint subsets - (1) irrelevant features, (2) redundant feature, (3) weakly relevant but non-redundant features, and (4) strongly relevant features. An optimal feature selection algorithm should non-redundant and strongly relevant features. For streaming features, it is difficult to find all strongly relevant and non-redundant features. OSFS finds an optimal subset using a two-phase scheme - online relevance analysis and redundancy analysis. An general framework of OSFS is presented as follows

- Initialize $BCF = \emptyset$;
- *Step 1:* Generate a new feature f_k;
- *Step 2:* Online relevance analysis

  Disregard f_k, if f_k is irrelevant to the class labels.
  $BCF = BCF \cup f_k$ otherwise;

- *Step 3:* Online Redundancy Analysis;
- *Step 4:* Alternate *Step 1* to *Step 3* until the stopping criteria are satisfied.

In the relevance analysis phase, OSFS discovers strongly and weakly relevant features, and adds them into best candidate features (BCF). If a new coming feature is irrelevant to the class label, it is discarded, otherwise it is added to BCF.

In the redundancy analysis, OSFS dynamically eliminates redundant features in the selected subset. For each feature f_k in BCF, if there exists a subset within BCF making f_k and the class label conditionally independent, f_k is removed from BCF. An alternative way to improve the efficiency is to further divide this phase into two analysis - inner-redundancy analysis and outer-redundancy analysis. In the inner-redundancy analysis, OSFS only re-examines the feature newly added into BCF, while the outer-redundancy analysis re-examines each feature of BCF only when the process of generating a feature is stopped.

0.5 Discussions and Challenges

Here are several challenges and concerns that we need to mention and discuss briefly in this chapter about feature selection for classification.

0.5.1 Scalability

With the tremendous growth of dataset sizes, the scalability of current algorithms may be in jeopardy, especially with these domains that require online classifier. For example, data that cannot be loaded into the memory require a single data scan where the second pass is either unavailable or very expensive. Using feature selection methods for classification may reduce the issue of scalability for clustering. However, some of the current methods that involve feature selection in the classification process require to keep full dimensionality in the memory. Furthermore, other methods require an iterative process where each sample is visited more than once until convergence.

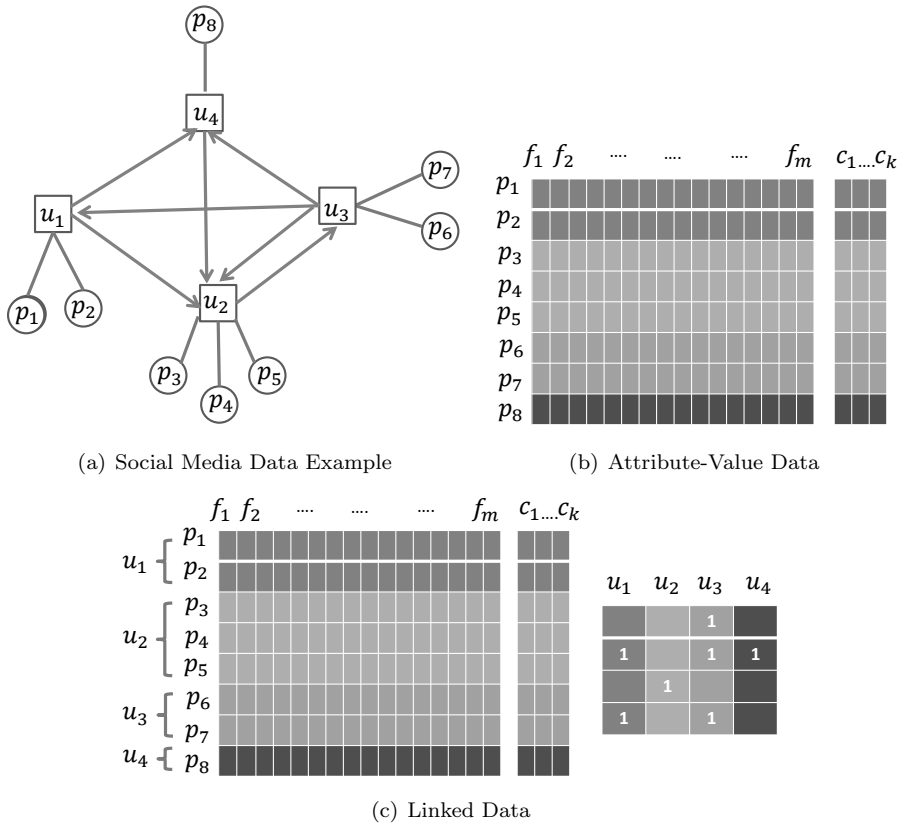
On the other hand, the scalability of feature selection algorithms is a big problem. Usually, they require a sufficient number of samples to obtain, statically, adequate results. It is very hard to observe feature relevance score without considering the density around each sample. Some methods try to overcome this issue by memorizing only samples that are important or a summary. In conclusion, we believe that the scalability of classification and feature selection methods should be given more attention to keep pace with the growth and fast streaming of the data.

0.5.2 Stability

Algorithms of feature selection for classification are often evaluated through classification accuracy. However, the stability of algorithms is also an important consideration when developing feature selection methods. A motivated example is from bioinformatics, the domain experts would like to see the same or at least similar set of genes, i.e. features to be selected, each time they obtain new samples in the presence of a small amount of perturbation. Otherwise they will not trust the algorithm when they get different sets of features while the datasets are drawn for the same problem. Due to its importance, stability of feature selection has drawn attention of the feature selection community. It is defined as the sensitivity of the selection process to data perturbation in the training set. It is found that well-known feature selection methods can select features with very low stability after perturbation is introduced to the training samples. In [2] the authors found that even the underlying characteristics of data can greatly affect the stability of an algorithm. These characteristics include dimensionality m, sample size n, and different data distribution across different folds, and the stability issue tends to be data dependent. Developing algorithms of feature selection for classification with high classification accuracy and stability is still challenging.

0.5.3 Linked Data

Most existing algorithms of feature selection for classification work with generic datasets and always assume that data is independent and identically distributed. With the development of social media, linked data is available where instances contradict the i.i.d. assumption.



(a) Social Media Data Example

(b) Attribute-Value Data

(c) Linked Data

FIGURE 10: Typical Linked Social Media Data and Its Two Representations

Linked data has become ubiquitous in real-world applications such as tweets in Twitter [1] (tweets linked through hyperlinks), social networks in Facebook [2](people connected by friendships) and biological networks (protein interaction networks). Linked data is patently not independent and identically distributed (i.i.d.), which is among the most enduring and deeply buried assumptions of traditional machine learning methods [31, 63]. Figure 10 shows a typical example of linked data in social media. Social media data is intrinsically linked via various types of relations such as user-post relations and user-user relations.

Many linked data related learning tasks are proposed such as collective classification [47, 59], and relational clustering [44, 45], but the task of feature selection for linked data is rarely touched. Feature selection methods for linked data need to solve the following immediate challenges,

- How to exploit relations among data instances;and
- How to take advantage of these relations for feature selection.

An attempt to handle linked data w.r.t. feature selection for classification is LinkedFS [62] and FSNet [16]. FSNet works with networked data and is supervised, while LinkedFS works with social media data with social context and is semi-supervised. In

[1] http://www.twitter.com/
[2] https://www.facebook.com/

LinkedFS, various relations (coPost, coFollowing, coFollowed and Following) are extracted following social correlation theories. LinkedFS significantly improves the performance of feature selection by incorporating these relations into feature selection. There are many issues needing further investigation for linked data such as handling noise, incomplete and unlabeled linked social media data.

Acknowledgments

We thanks Daria Bazzi for useful comments on the representation, and helpful discussions from Yun Li and Charu Aggarwal about the content of this book chapter. This work is, in part, supported by National Science Foundation under Grant No. IIS-1217466.

Bibliography

[1] S. Alelyani, J. Tang, and H. Liu. Feature selection for clustering: A review. Data Clustering: Algorithms and Applications, Editor: Charu Aggarwal and Chandan Reddy, CRC Press, 2013.

[2] S. Alelyani, L. Wang, and H. Liu. The effect of the characteristics of the dataset on the selection stability. In *Proceedings of the 23rd IEEE International Conference on Tools with Artificial Intelligence*, 2011.

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

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

[5] P. S. Bradley and O. L. Mangasarian. Feature selection via concave minimization and support vector machines. pages 82–90. Morgan Kaufmann, 1998.

[6] E. Candes and T. Tao. The dantzig selector: statistical estimation when p is much larger than n. *The Annals of Statistics*, pages 2313–2351, 2007.

[7] H. Y. Chuang, E. Lee, Y.T. Liu, D. Lee, and T. Ideker. Network-based classification of breast cancer metastasis. *Molecular systems biology*, 3(1), 2007.

[8] M. Dash and H. Liu. Feature selection for classification. *Intelligent data analysis*, 1(1-4):131–156, 1997.

[9] C. Domeniconi and D. Gunopulos. Local feature selection for classification. *Computational Methods*, page 211, 2008.

[10] R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern classification*. Wiley-interscience, 2012.

[11] R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern Classification*. John Wiley & Sons, New York, 2 edition, 2001.

[12] J.G. Dy and C.E. Brodley. Feature subset selection and order identification for unsupervised learning. In *In Proc. 17th International Conference on Machine Learning*, pages 247–254. Morgan Kaufmann, 2000.

[13] J.G. Dy and C.E. Brodley. Feature selection for unsupervised learning. *The Journal of Machine Learning Research*, 5:845–889, 2004.

[14] J. Fan and R. Li. Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American Statistical Association*, 96(456):1348–1360, 2001.

[15] N.L. C. Talbot G. C. Cawley and M. Girolami. Sparse multinomial logistic regression via bayesian l1 regularisation. In *Neural Information Processing Systems*, 2006.

[16] Q. Gu, Z. Li, and J. Han. Towards feature selection in network. In *International Conference on Information and Knowledge Management*, 2011.

[17] Q. Gu, Z. Li, and J. Han. Generalized fisher score for feature selection. *arXiv preprint arXiv:1202.3725*, 2012.

[18] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *The Journal of Machine Learning Research*, 3:1157–1182, 2003.

[19] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Machine learning*, 46(1-3):389–422, 2002.

[20] M.A. Hall and L.A. Smith. Feature selection for machine learning: comparing a correlation-based filter approach to the wrapper. In *Proceedings of the Twelfth International Florida Artificial Intelligence Research Society Conference*, volume 235, page 239, 1999.

[21] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, 2001.

[22] T. Hastie and R. Tibshirani. Discriminant adaptive nearest neighbor classification. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 18(6):607–616, 1996.

[23] J. Huang, J. L. Horowitz, and S. Ma. Asymptotic properties of bridge estimators in sparse high-dimensional regression models. *The Annals of Statistics*, 36(2):587–613, 2008.

[24] J. Huang, S. Ma, and C. Zhang. Adaptive lasso for sparse high-dimensional regression models. *Statistica Sinica*, 18(4):1603, 2008.

[25] J. Huang, T. Zhang, and D. Metaxas. Learning with structured sparsity. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 417–424. ACM, 2009.

[26] I. Inza, P. Larrañaga, R. Blanco, and A. J. Cerrolaza. Filter versus wrapper gene selection approaches in dna microarray domains. *Artificial intelligence in medicine*, 31(2):91–103, 2004.

[27] L. Jacob, G. Obozinski, and J. Vert. Group lasso with overlap and graph lasso. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 433–440. ACM, 2009.

[28] G. James, P. Radchenko, and J. Lv. Dasso: connections between the dantzig selector and lasso. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 71(1):127–142, 2009.

[29] R. Jenatton, J. Audibert, and F. Bach. Structured variable selection with sparsity-inducing norms. *The Journal of Machine Learning Research*, 12:2777–2824, 2011.

[30] R. Jenatton, J. Mairal, G. Obozinski, and F. Bach. Proximal methods for sparse hierarchical dictionary learning. International Conference on Machine Learning, 2010.

[31] D. Jensen and J. Neville. Linkage and autocorrelation cause feature selection bias in relational learning. In *International Conference on Machine Learning*, pages 259–266, 2002.

[32] S. Kim and E. Xing. Statistical estimation of correlated genome associations to a quantitative trait network. *PLoS genetics*, 5(8):e1000587, 2009.

[33] S. Kim and E. Xing. Tree-guided group lasso for multi-task regression with structured sparsity. 2010.

[34] K. Kira and L. Rendell. A practical approach to feature selection. In *Proceedings of the ninth international workshop on Machine learning*, pages 249–256. Morgan Kaufmann Publishers Inc., 1992.

[35] K. Knight and W. Fu. Asymptotics for lasso-type estimators. *Annals of Statistics*, pages 1356–1378, 2000.

[36] R. Kohavi and G.H. John. Wrappers for feature subset selection. *Artificial intelligence*, 97(1-2):273–324, 1997.

[37] D. Koller and M. Sahami. Toward optimal feature selection. International Conference on Machine Learning,1996.

[38] H. Liu and H. Motoda. *Feature selection for knowledge discovery and data mining*, volume 454. Springer, 1998.

[39] H. Liu and H. Motoda. *Computational Methods of Feature Selection*. Chapman and Hall/CRC Press, 2007.

[40] H. Liu and L. Yu. Toward integrating feature selection algorithms for classification and clustering. *IEEE Transactions on Knowledge and Data Engineering*, 17(4):491, 2005.

[41] J. Liu, S. Ji, and J. Ye. Multi-task feature learning via efficient l 2, 1-norm minimization. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pages 339–348. AUAI Press, 2009.

[42] J. Liu and J. Ye. Moreau-yosida regularization for grouped tree structure learning. *Advances in Neural Information Processing Systems*, 187:195–207, 2010.

[43] Z. Liu, F. Jiang, G. Tian, S. Wang, F. Sato, S. Meltzer, and M. Tan. Sparse logistic regression with lp penalty for biomarker identification. *Statistical Applications in Genetics and Molecular Biology*, 6(1), 2007.

[44] B. Long, Z.M. Zhang, X. Wu, and P.S. Yu. Spectral clustering for multi-type relational data. In *Proceedings of the 23rd international conference on Machine learning*, pages 585–592. ACM, 2006.

[45] B. Long, Z.M. Zhang, and P.S. Yu. A probabilistic framework for relational clustering. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 470–479. ACM, 2007.

[46] S. Ma and J. Huang. Penalized feature selection and classification in bioinformatics. *Briefings in bioinformatics*, 9(5):392–403, 2008.

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

[48] M. Masaeli, J G Dy, and G. Fung. From transformation-based dimensionality reduction to feature selection. In *Proceedings of the 27th International Conference on Machine Learning*, pages 751–758, 2010.

[49] J. McAuley, J Ming, D. Stewart, and P. Hanna. Subband correlation and robust speech recognition. *Speech and Audio Processing, IEEE Transactions on*, 13(5):956–964, 2005.

[50] L. Meier, S. De Geer, and P. Bühlmann. The group lasso for logistic regression. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 70(1):53–71, 2008.

[51] P. Mitra, C. A. Murthy, and S. Pal. Unsupervised feature selection using feature similarity. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24:301–312, 2002.

[52] H. Peng, F. Long, and C. Ding. Feature selection based on mutual information: criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1226–1238, 2005.

[53] S. Perkins and J. Theiler. Online feature selection using grafting. In *International Conference on Machine Learning*, pages 592–599, 2003.

[54] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[55] J. R.Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

[56] M. Robnik-Šikonja and I. Kononenko. Theoretical and empirical analysis of relieff and rrelieff. *Machine learning*, 53(1-2):23–69, 2003.

[57] Y. Saeys, I. Inza, and P. Larrañaga. A review of feature selection techniques in bioinformatics. *Bioinformatics*, 23(19):2507–2517, 2007.

[58] T. Sandler, P. Talukdar, L. Ungar, and J. Blitzer. Regularized learning with networks of features. *Neural Information Processing Systems*, 2008.

[59] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93, 2008.

[60] M. R. Sikonja and I. Kononenko. Theoretical and empirical analysis of Relief and ReliefF. *Machine Learning*, 53:23–69, 2003.

[61] L. Song, A. Smola, A. Gretton, K. Borgwardt, and J. Bedo. Supervised feature selection via dependence estimation. In *International Conference on Machine Learning*, 2007.

[62] J. Tang and H. Liu. Feature selection with linked data in social media. In *SDM*, 2012.

[63] B. Taskar, P. Abbeel, M.F. Wong, and D. Koller. Label and link prediction in relational data. In *Proceedings of the IJCAI Workshop on Learning Statistical Models from Relational Data*. Citeseer, 2003.

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

[65] R. Tibshirani, M. Saunders, S. Rosset, J. Zhu, and K. Knight. Sparsity and smoothness via the fused lasso. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(1):91–108, 2005.

[66] R. Tibshirani and P. Wang. Spatial smoothing and hot spot detection for cgh data using the fused lasso. *Biostatistics*, 9(1):18–29, 2008.

[67] J. Wang, P. Zhao, S. Hoi, and R. Jin. Online feature selection and its applications. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–14, 2013.

[68] J. Weston, A. Elisseff, B. Schoelkopf, and M. Tipping. Use of the zero norm with linear models and kernel methods. *Journal of Machine Learning Research*, 3:1439–1461, 2003.

[69] D. Wettschereck, D. Aha, and T. Mohri. A review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms. *Artificial Intelligence Review*, 11:273–314, 1997.

[70] X. Wu, K. Yu, H. Wang, and W. Ding. Online streaming feature selection. In *Proceedings of the 27th international conference on machine learning*, pages 1159–1166, 2010.

[71] Z. Xu, R. Jin, J. Ye, M. Lyu, and I. King. Discriminative semi-supervised feature selection via manifold regularization. In *IJCAI' 09: Proceedings of the 21th International Joint Conference on Artificial Intelligence*, 2009.

[72] S. Yang, L. Yuan, Y. Lai, X. Shen, P. Wonka, and J. Ye. Feature grouping and selection over an undirected graph. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 922–930. ACM, 2012.

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

[74] L. Yu and H. Liu. Feature selection for high-dimensional data: A fast correlation-based filter solution. In *International Conference on Machine Learning*, volume 20, page 856, 2003.

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

[76] P. Zhao and B. Yu. On model selection consistency of lasso. *The Journal of Machine Learning Research*, 7:2541–2563, 2006.

[77] Z. Zhao and H. Liu. Semi-supervised feature selection via spectral analysis. In *Proceedings of SIAM International Conference on Data Mining*, 2007.

[78] D. Zhou, J. Huang, and B. Schölkopf. Learning from labeled and unlabeled data on a directed graph. In *International Conference on Machine Learning*, pages 1036–1043. ACM, 2005.

[79] J. Zhou, J. Liu, V. Narayan, and J. Ye. Modeling disease progression via fused sparse group lasso. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1095–1103. ACM, 2012.

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

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