Feature Selection

Suhang Wang, Jiliang Tang and Huan Liu
Arizona State University

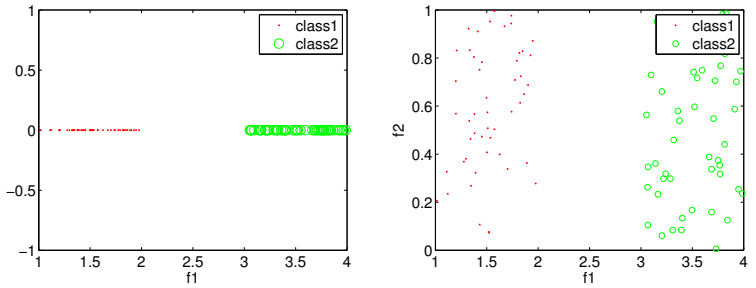
Abstract

Data dimensionality is growing exponentially, which poses challenges to the vast majority of existing mining and learning algorithms, such as the curse of dimensionality, large storage requirement, and high computational cost. Feature selection has been proven to be an effective and efficient way to prepare high dimensional data for data mining and machine learning. The recent emergence of novel techniques and new types of data and features not only advances existing feature selection research but also makes feature selection evolve more rapidly, becoming applicable to a broader range of applications. In this article, we aim to provide a basic introduction to feature selection including basic concepts, classifications of existing systems, recent development, and applications.

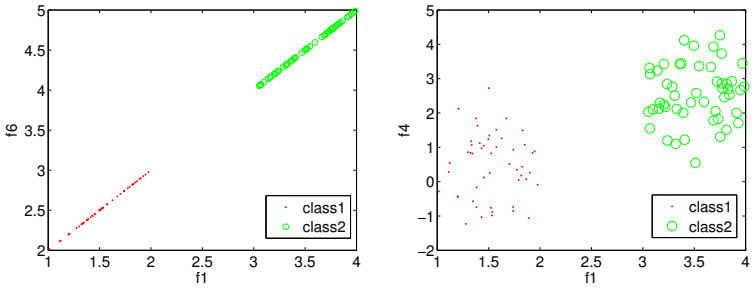
Synonyms: Feature subset selection, Feature weighting, Attributes selection

Definition (or Synopsis): Feature selection, as a dimensionality reduction technique, aims to choosing a small subset of the relevant features from the original ones by removing irrelevant, redundant or noisy features. Feature selection usually leads to better learning performance, i.e., higher learning accuracy, lower computational cost, and better model interpretability.

Generally speaking, irrelevant features are features that cannot discriminate samples from different classes(supervised) or clusters(unsupervised). Removing irrelevant features will not affect learning performance. In fact, removal of irrelevant features may help learn a better model, as irrelevant features may confuse the learning system and cause memory and computation inefficiency. For example, in figure 1(a), f_1 is a relevant feature because f_1 can discriminate class1 and class2. In figure 1(b), f_2 is a redundant feature because f_2 cannot distinguish points from class1 and class2. Removal of f_2 doesn't affect the ability of f_1 to distinguish samples from class1 and class2.



(a) relevant feature (b) irrelevant feature



(c) redundant feature (d) noisy feature

Figure 1: A toy example to illustrate the concept of irrelevant, redundant and noisy features. $f1$ is a relevant feature and can discriminate class1 and class2. $f2$ is an irrelevant feature. Removal of $f2$ will not affect the learning performance. $f4$ is a noisy feature. Presence of noisy features may degenerate the learning performance. $f6$ is a redundant feature when $f1$ is present. If $f1$ is selected, removal of $f6$ will not affect the learning performance.

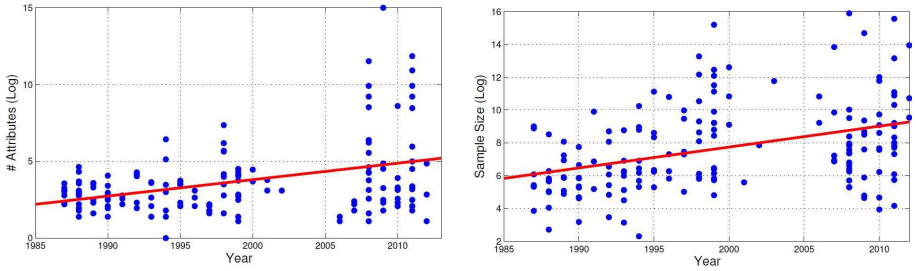
A redundant feature is a feature that implies the co-presence of another feature. Individually, each redundant feature is relevant, but removal of one of them will not affect the learning performance. For example, in figure 1(c), f_1 and f_6 are strongly correlated. f_6 is a relevant feature itself. However, when f_1 is selected first, the later appearance of f_6 doesn't provide additional information. Instead, it adds more memory and computational requirement to learn the classification model.

A noisy feature is a type of relevant feature. However, due to the noise introduced during the data collection process or because of the nature of this feature, a noisy feature may not be so relevant to the learning or mining task. As shown in figure 1(d), f_4 is a noisy feature. It can discriminate a part of

the points from the two classes and may confuse the learning model for the overlapping points [1].

Motivation and Background

In many real world applications, such as data mining, machine learning, computer vision, and bioinformatics, we need to deal with high dimensional data. In the past thirty years, the dimensionality of the data involved in these areas has increased explosively. The growth of the number of attributes in the UCI machine learning repository is shown in figure 2(a). In addition, the number of samples also increases explosively. The growth of the number of samples in the UCI machine learning repository is shown in figure 2(b). The huge number of high dimensional data has presented serious challenges to existing learning methods. First, due to the large number of features and relatively small number of training samples, a learning model tends to overfit and their learning performance degenerates. Data with high dimensionality not only degenerates many algorithms' performance due to the curse of dimensionality and the existence of irrelevant, redundant, and noisy dimensions, it also significantly increases the time and memory requirement of the algorithms. Second, storing and processing such amounts of high dimensional data becomes a challenge.



(a) UCI ML Repository Number of Attributes Growth

(b) UCI ML Repository Number of Samples Growth

Figure 2: Growth of the number of features and the number of samples in the UCI ML repository

Dimensionality reduction is one of the most popular techniques to reduce dimensionality, and can be categorized into feature extraction and feature

[1] Noisy features are very subtle. One feature may be a noisy feature itself. However, in some cases, when two or more noisy features can complement each other to distinguish samples from different classes, they may be selected together to benefit the learning model.

selection. Both feature extraction and feature selection are capable of improving performance, lowering computational complexity, building better generalization models, and decreasing required storage. Feature extraction maps the original feature space to a new feature space with lower dimensionality by combining the original feature space. Therefore, further analysis of new features is problematic since there is no physical meaning for the transformed features obtained from feature extraction. In contrast, feature selection selects a subset of features from the original feature set. Therefore, feature selection keeps the actual meaning of each selected feature, which makes it superior in terms of feature readability and interpretability.

Structure of the Learning System

From the perspective of label availability, feature selection methods can be broadly classified into supervised, unsupervised, and semi-supervised methods. In terms of different selection strategies, feature selection can be categorized as filter, wrapper and embedded models. Figure 3 shows the classification of feature selection methods.

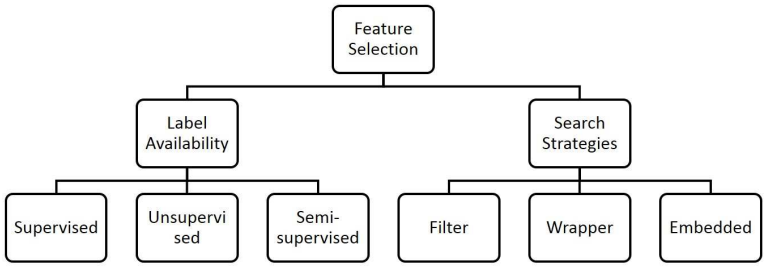


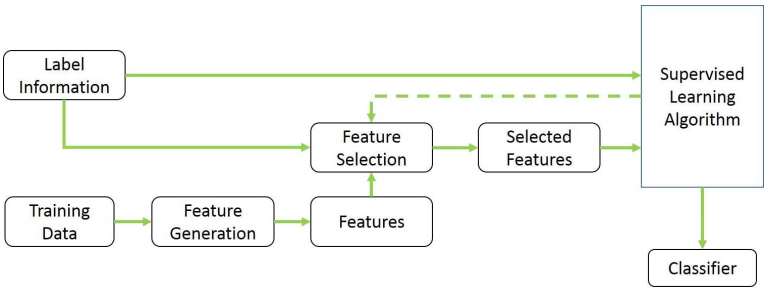
Figure 3: Feature Selection Categories

Supervised Feature Selection is usually used for classification tasks. The availability of the class labels allows supervised feature selection algorithms to effectively select discriminative features to distinguish samples from different classes. A general framework of supervised feature selection is shown in fig 4(a). Features are first generated from training data. Instead of using all the data to train the supervised learning model, supervised feature selection will first select a subset of features and then process the data with the selected features to the learning model. The feature selection phase will use the label information and the characteristics of the data, such as information

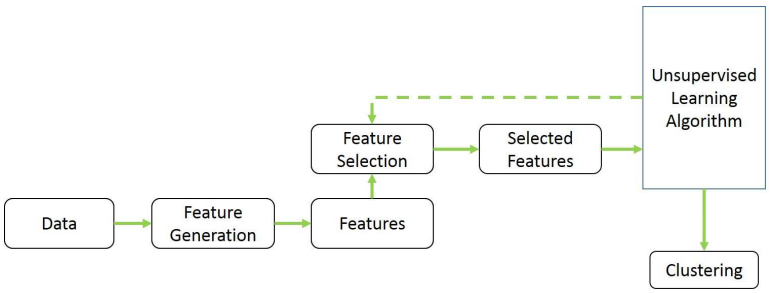
gain or Gini index, to select relevant features. The final selected features, as well as with the label information, are used to train a classifier, which can be used for prediction.

Unsupervised Feature Selection is usually used for clustering tasks. A general framework of unsupervised feature selection is described in figure 4(b), which is very similar to supervised feature selection, except that there's no label information involved in the feature selection phase and the model learning phase. Without label information to define feature relevance, unsupervised feature selection relies on other alternative criterion during the feature selection phase. One commonly used criterion chooses features that can best preserve the manifold structure of the original data. Another frequently used method is to seek cluster indicators through clustering algorithms and then transform the unsupervised feature selection into a supervised framework. There are two different ways to use this method. One way is to seek cluster indicators and simultaneously perform the supervised feature selection within one unified framework. The other way is to first seek cluster indicators, then perform feature selection to remove or select certain features, and finally to repeat these two steps iteratively until certain criteria is met. In addition, certain supervised feature selection criterion can still be used with some modification.

Semi-supervised Feature selection is usually used when a small portion of the data is labeled. When such data is given to perform feature selection, both supervised and unsupervised feature selection might not be the best choice. Supervised feature selection might not be able to select relevant features because the labeled data is insufficient to represent the distribution of the features. Unsupervised feature selection will not use the label information, while label information can give some discriminative information to select relevant features. Semi-supervised feature selection, which takes advantage of both labeled data and unlabeled data, is a better choice to handle partially labeled data. The general framework of semi-supervised feature selection is the same as that of supervised feature selection, except that data is partially labeled. Most of the existing semi-supervised feature selection algorithms rely on the construction of the similarity matrix and select features that best fit the similarity matrix. Both the label information and the similarity measure of the labeled and unlabeled data is used to construct the similarity matrix so that label information can provide discriminative information to select relevant features while unlabeled data provide complementary information.



(a) A General Framework of Supervised Feature Selection



(b) A General Framework of Unsupervised Feature Selection

Figure 4: General frameworks of supervised and unsupervised feature selection

Filter Models For filter models, features are selected based on the characteristics of the data without utilizing learning algorithms. This approach is very efficient. However, it doesn't consider the bias and heuristics of the learning algorithms. Thus, it may miss features that are relevant for the target learning algorithm. A filter algorithm usually consists of two steps. In the first step, features are ranked based on certain criterion. In the second step, features with the highest rankings are chosen. A lot of ranking criteria, which measures different characteristics of the features, are proposed: the ability to effectively separate samples from different classes by considering between class variance and within class variance, the dependence between feature and the class label, the correlation between feature-class and feature-feature, the ability to preserve the manifold structure, mutual information between the features, and so on.

Wrapper Models The major disadvantage of the filter approach is that it totally ignores the effects of the selected feature subset on the perfor-

mance of the clustering or classification algorithm. The optimal feature subset should depend on the specific biases and heuristics of the learning algorithms. Based on this assumption, wrapper models use a specific learning algorithm to evaluate the quality of the selected features. Given a predefined learning algorithm, a general framework of the wrapper model is shown in Figure 5. The feature search component will produce a set of features based on certain search strategies. The feature evaluation component will then use the predefined learning algorithm to evaluate the performance, which will be returned to the feature search component for the next iteration of feature subset selection. The feature set with the best performance will be chosen as the final set. The search space for m features is $O(2^m)$. To avoid exhaustive search, a wide range of search strategies can be used, including hill-climbing, best-first, branch-and-bound, and genetic algorithms.

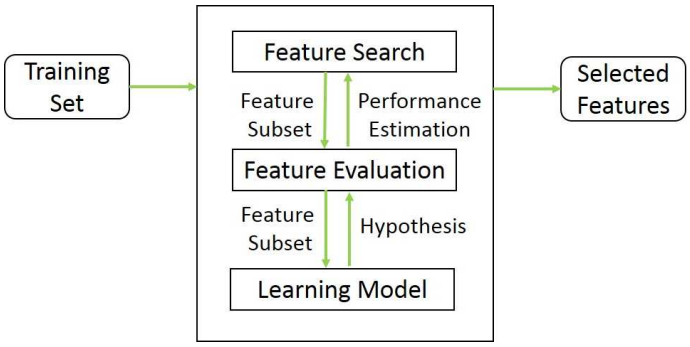


Figure 5: A General Framework of Wrapper Models

Embedded Models Filter models are computationally efficient, but totally ignore the biases of the learning algorithm. Compared with filter models, wrapper models obtain better predictive accuracy estimates, since they take into account the biases of the learning algorithms. However, wrapper models are very computationally expensive. Embedded models are a tradeoff between the two models by embedding the feature selection into the model construction. Thus, embedded models take advantage of both filter models and wrapper models: (1) they are far less computationally intensive than wrapper methods, since they don't need to run the learning models many times to evaluate the features, and (2) they include the interaction with the learning model. The biggest difference between wrapper models and embedded models is that wrapper models first train learning models using the candidate features and then perform feature selection by evaluating features

using the learning model, while embedded models select features during the process of model construction to perform feature selection without further evaluation of the features.

Recent Developments

The recent emergence of new machine learning algorithms such as sparse learning, and new types of data, such as social media data, has accelerated the evolution of feature selection. In this section, we will discuss recent developments of feature selection from both feature and data perspectives.

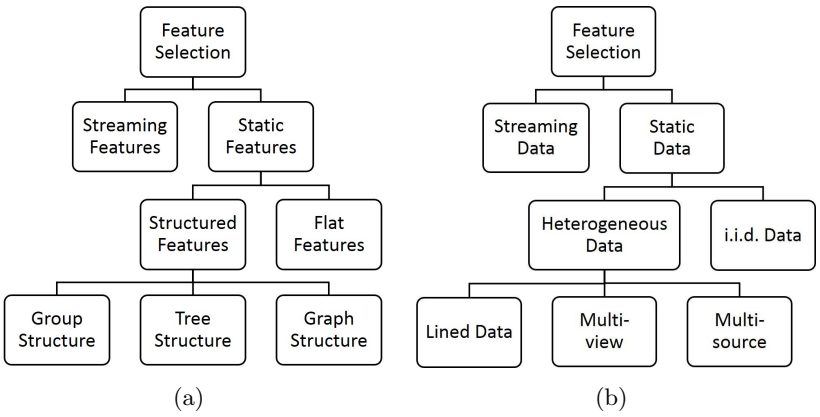


Figure 6: Classification of Recent Development of Feature Selection from Feature Perspective and Data Perspective

From the feature perspective, features can be categorized as static and streaming features, as shown in Figure 6(a). Static features can be further categorized as flat features and structured features. The recent development of feature selection from the feature perspective mainly focuses on streaming and structure features.

Usually we assume that all features are known in advance. These features are designated as static features. In some scenarios, new features are sequentially presented to the learning algorithm. For example, Twitter produces more than 250 millions tweets per day, and many new words (features) are generated, such as abbreviations. In these scenarios, the candidate features are generated dynamically, and the size of features is unknown. These features are usually named as streaming features and feature selection for streaming features is called streaming feature selection. For flat features, we

assume that features are independent. However, in many real-world applications, features may exhibit certain intrinsic structures, such as overlapping groups, trees, and graph structures. For example, in speed and signal processing, different frequency bands can be represented by groups. Figure 6(a) shows the classification of structured features. Incorporating knowledge about feature structures may significantly improve the performance of learning models and help select important features. Feature selection algorithms for the structured features usually use the recently developed sparse learning techniques such as group lasso and tree-guided lasso.

From the data perspective, data can be categorized as streaming data and static data as shown in Figure 6(b). Static data can be further categorized as independent identically distributed (i.i.d) data and heterogeneous data. The recent development of feature selection from the data perspective is mainly concentrated on streaming and heterogeneous data.

Similar to streaming features, streaming data comes sequentially. Online streaming feature selection is proposed to deal with streaming data. When new data instances come, an online feature selection algorithm needs to determine (1) whether adding the newly generated features from the coming data to the currently selected features, and (2) whether removing features from the set of currently selected features. Traditional data is usually assumed to be i.i.d data, such as text and gene data. However, heterogeneous data, such as linked data, apparently contradicts this assumption. For example, linked data is inherently not i.i.d., since instances are linked and correlated. New types of data cultivate new types of feature selection algorithms correspondingly, such as feature selection for linked data, multi-view, and multi-source feature selection.

Applications

High dimensional data is very ubiquitous in the real-world, which makes feature selection a very popular and practical preprocessing technique for various real-world applications, such as text categorization, remote sensing, image retrieval, microarray analysis, mass spectrum analysis, sequence analysis, and so on.

Text Clustering The task of text clustering is to group similar documents together. In text clustering, a text or document is always represented as a bag of words, which causes high dimensional feature space and sparse representation. Obviously, a single document has a sparse vector over the set of

all terms. The performance of clustering algorithms degrades dramatically due to high dimensionality and data sparseness. Therefore, in practice, feature selection is a very important step to reduce the feature space in text clustering.

Genomic Microarray Data Microarray data is usually short and fat data - high dimensionality with a small sample size, which poses a great challenge for computational techniques. Their dimensionality can be up to tens of thousands of genes, while their sample sizes can only be several hundreds. Furthermore, additional experimental complications like noise and variability render the analysis of microarray data an exciting domain. Because of these issues, various feature selection algorithms are adopted to reduce the dimensionality and remove noise in microarray data analysis.

Hyperspectral Image Classification Hyperspectral sensors record the reflectance from the Earth's surface over the full range of solar wavelengths with high spectral resolution, which results in high dimensional data that contains rich information for a wide range of applications. However, this high dimensional data contains many irrelevant, noisy, and redundant features that are not important, useful, or desirable for specific tasks. Feature selection is a critical preprocessing step to reduce computational cost for hyperspectral data classification by selecting relevant features.

Sequence Analysis In bioinformatics, sequence analysis is a very important process to understand a sequence's features, functions, structure, or evolution. In addition to basic features that represent nucleotide or amino acids at each position in a sequence, many other features, such as k-mer patterns, can be derived. By varying the pattern length k, the number of features grows exponentially. However, many of these features are irrelevant or redundant; thus, feature selection techniques are applied to select a relevant feature subset and essential for sequence analysis.

Open Problems

Scalability With the rapid growth of dataset size, the scalability of current feature selection algorithms may be a big issue, especially for online classifiers. Large data cannot be loaded to the memory with a single scan. However, full dimensionality data must be scanned for some feature selection. Usually, they require a sufficient number of samples to obtain statistically

significant result. It is very difficult to observe the feature relevance score without considering the density around each sample. Therefore, scalability is a big issue.

Stability Feature selection algorithms are often evaluated through classification accuracy or clustering accuracy. However, the stability of algorithms is also an important consideration when developing feature selection methods. For example, when feature selection is applied on gene data, the domain experts would like to see the same or at least similar sets of genes selected after each time they obtain new samples with a small amount of perturbation. Otherwise, they will not trust the algorithm. However, well known feature selection methods, especially unsupervised feature selection algorithms, can select features with low stability after perturbation is introduced to the training data. Developing algorithms of feature selection with high accuracy and stability is still an open problem.

Parameter Selection In feature selection, we usually need to specify the number of features to select. However, the optimal number of features for the dataset is unknown. If the number of selected features is too few, the performance will be degenerated, since some relevant features are eliminated. If the number of selected features is too large, the performance may also not be very good since some noisy, irrelevant, or redundant features are selected to confuse the learning model. In practice, we would grid search the number of features in a range and pick the one that has relatively better performance on learning models, which is computationally expensive. In particular, for supervised feature selection, cross validation can be used to search the number of features to select. How to automatically determine the best number of selected features remains an open problem.

For many unsupervised feature selection methods, in addition to choosing the optimal number of features, we also need to specify the number of clusters. Since there is no label information and we have limited knowledge about each domain, the actual number of clusters in the data is usually unknown and not well defined. The number of clusters specified by users will result in selecting different feature subsets by the unsupervised feature selection algorithm. How to choose the number of clusters for unsupervised feature selection is an open problem.

Cross References

* Feature Extraction
* Dimensionality Reduction
* Classification
* Clustering

Recommended Reading

[1] Salem Alelyani, Jiliang Tang, and Huan Liu. Feature selection for clustering: A review., 2013.

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

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

[4] Anil Jain and Douglas Zongker. Feature selection: Evaluation, application, and small sample performance. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(2):153–158, 1997.

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

[6] Daphne Koller and Mehran Sahami. Toward optimal feature selection. 1996.

[7] Huan Liu and Hiroshi Motoda. *Computational methods of feature selection*. CRC Press, 2007.

[8] Huan Liu, Hiroshi Motoda, Rudy Setiono, and Zheng Zhao. Feature selection: An ever evolving frontier in data mining. In *FSDM*, pages 4–13, 2010.

[9] Huan Liu and Lei Yu. Toward integrating feature selection algorithms for classification and clustering. *Knowledge and Data Engineering, IEEE Transactions on*, 17(4):491–502, 2005.

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

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

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

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

[14] Zheng Zhao, Fred Morstatter, Shashvata Sharma, Salem Alelyani, Aneeth Anand, and Huan Liu. Advancing feature selection research. *ASU Feature Selection Repository*, 2010.

[15] Zheng Alan Zhao and Huan Liu. *Spectral feature selection for data mining*. Chapman & Hall/CRC, 2011.