Challenges of Feature Selection for Big Data Analytics

Jundong Li and Huan Liu, *Arizona State University*

Feature selection has shown its effectiveness in many applications, but the unique characteristics of big data present challenges.

Massive amounts of high-dimensional data are pervasive in multiple domains, including social media, e-commerce, bioinformatics, healthcare, transportation, and online education. As an example, Figure 1 shows the growth trend of instance and feature numbers in the University of California, Irvine (UCI), machine learning repository. As can be observed, both the data sample size and feature numbers are continuously growing over time. When applying data mining and machine learning algorithms on high-dimensional data, a critical issue is known as the curse of dimensionality. It refers to the phenomenon that data becomes sparser in high-dimensional space, adversely affecting algorithms designed for low-dimensional space. In addition, the existence of high-dimensional features will significantly demand more on the computational and memory storage requirements.

Feature selection, as a type of dimension reduction technique, has been proven to be effective and efficient in handling high-dimensional data.[1,2] It directly selects a subset of relevant features for the model construction. Because feature selection keeps a subset of original features, one of its major merits is that it maintains the physical meanings of the original feature sets and gives better model readability and interpretability. Accordingly, it's more widely applied in many real-world applications such as gene analysis and text mining, obtaining relevant features by removing irrelevant and redundant ones. The removal of these irrelevant and redundant features reduces the computational and storage costs without significant loss of information or negative degradation of the learning performance. Taking Figure 2 as an example, feature f_1 is a relevant feature that can separate two classes (clusters) in Figure 2a, whereas in Figure 2b, feature f_2 is considered a redundant feature with regard to feature f_1 because feature f_1 already can discriminate two classes (clusters) well. In Figure 2c, feature f_3 is an irrelevant feature as it doesn't contain useful information to separate two classes (clusters).

According to the availability of class labels, we can categorize feature selection algorithms into supervised and unsupervised methods. Supervised feature selection is usually taken as a preprocessing step for the classification/regression task. It chooses features that can discriminate data instances from different classes or regression targets. Because the label information is known a priori, a feature's relevance is normally assessed by its correlation with class labels. Unsupervised feature

1541-1672/17/$33.00 © 2017 IEEE
Published by the IEEE Computer Society

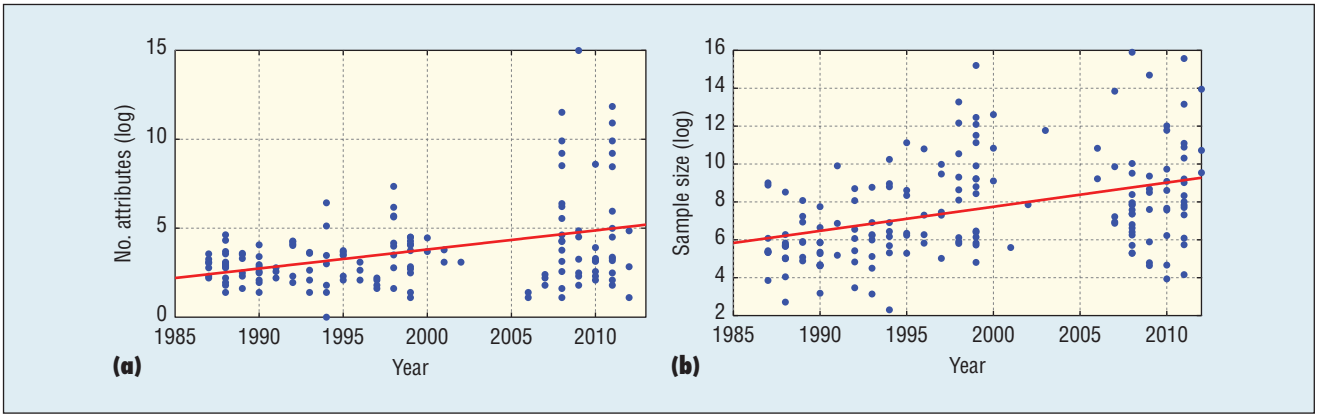

Figure 1. University of California, Irvine's (UCI's), machine learning repository: (a) features and (b) sample growth trend over the past 30 years.

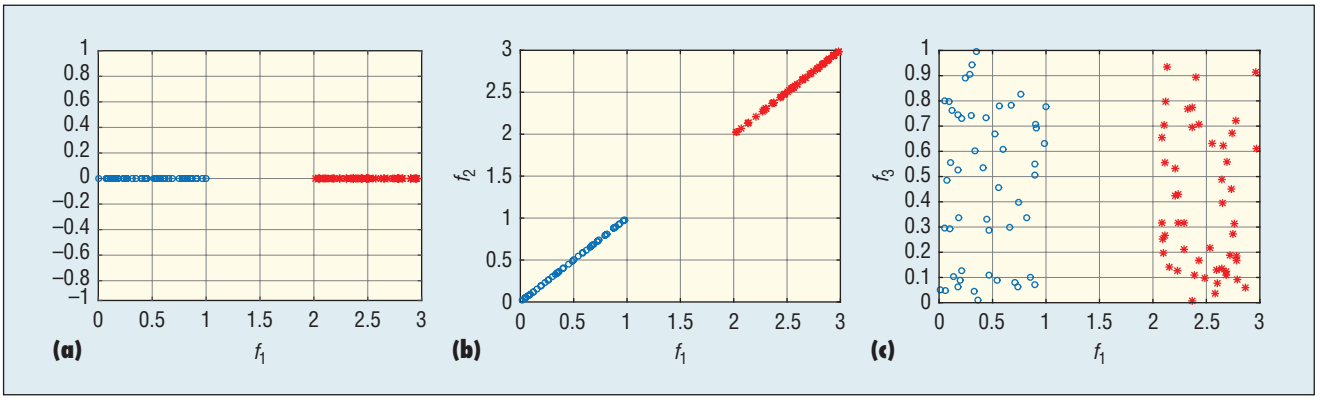

Figure 2. Examples of (a) relevant, (b) irrelevant, and (c) redundant features. Specifically, f_1 is a relevant feature, f_2 is a redundant feature, and f_3 is an irrelevant feature.

selection is generally applied for the clustering task. Without class labels to guide feature selection, it evaluates feature importance by some alternative criteria such as data similarity, local discriminative information, or data reconstruction error.

With regard to search strategies, feature selection algorithms can be divided into wrapper, filter, and embedded methods. Wrapper methods typically use a predefined model's learning performance to evaluate feature relevance. Specifically, they repeatedly choose a subset of features and then evaluate the learning performance with these selected features until the highest learning performance is obtained. Because these methods scan through the whole search space, they're slow and seldom used in practice. Filter methods, on the other hand, don't rely on any learning algorithms and are therefore more efficient. They exploit the data characteristics to measure feature relevance, typically by measuring the scores of features based on some ranking criteria and then returning the top ranked features. Because these methods don't explicitly consider the bias of learning algorithms, the selected features might not be optimal for a particular learning task. Embedded methods provide a tradeoff solution between filter and wrapper methods by embedding the feature selection with the model learning, thereby inheriting the benefits of wrapper and filter methods: they include interactions with the learning algorithm, and they're far more efficient than wrapper methods because they don't need to evaluate feature sets iteratively.

Challenges of Feature Selection

Big data's recent popularity presents some challenges for the traditional feature selection task. That said, its characteristics also bring new opportunities.

Structured Features

Most existing feature selection algorithms are designed for generic data, and they're based on a strong assumption that features don't have explicit correlations. In other words, they completely ignore the intrinsic structures among features—for example,

these feature selection methods might select the same subset of features even though the features are reshuffled.[3] In many real-world applications, features exhibit various kinds of structures, such as spatial or temporal smoothness, disjoint groups, overlap groups, trees, and graphs. When applying feature selection algorithms on datasets with structured features, it's beneficial to explicitly incorporate this prior knowledge, which could improve post-learning tasks such as classification and clustering.

The first structure that features can exhibit is group structure, examples of which include different frequency bands represented as groups in signal processing[4] and genes with similar functionalities acting as groups in bioinformatics.[5] Therefore, when performing feature selection, it's more appealing to explicitly take into consideration the group structure among features. Figure 3 shows an illustrative example of features with group structures (for four groups).

In addition to group structures, features can also form tree structures—for example, in image processing on faces, different pixels (features) can be represented as a tree, with the root node indicating the whole face, its child nodes the different organs, and each specific pixel considered as a leaf node. In other words, these pixels enjoy a spatial locality structure. Figure 4 shows an example of eight features with four layers of tree structure. Another motivating example is that genes/proteins can form certain hierarchical tree structures.[6]

Features can also form graph structures—for example, in natural language processing, if we take each word as a feature, we have synonyms and antonyms relationships between different words.[7] Moreover, many biological studies show that genes tend to work in groups according to their bio-

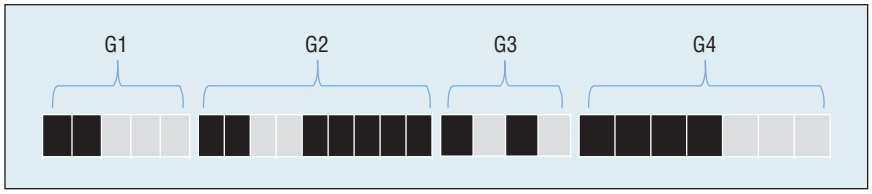

Figure 3. Group structures among features. These features form a group structure with four different groups.

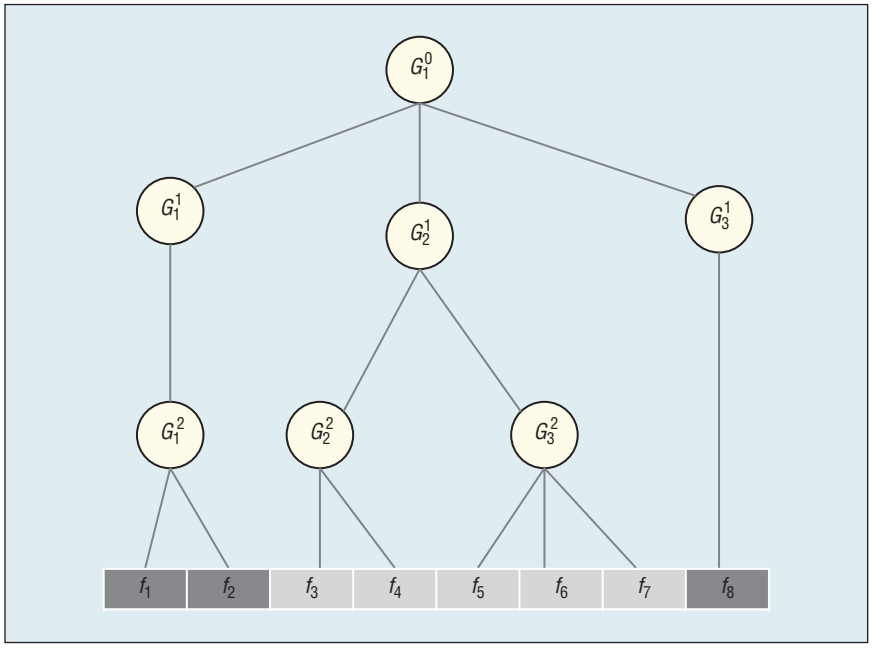

Figure 4. Tree structures among features. These features form a tree structure with four layers.

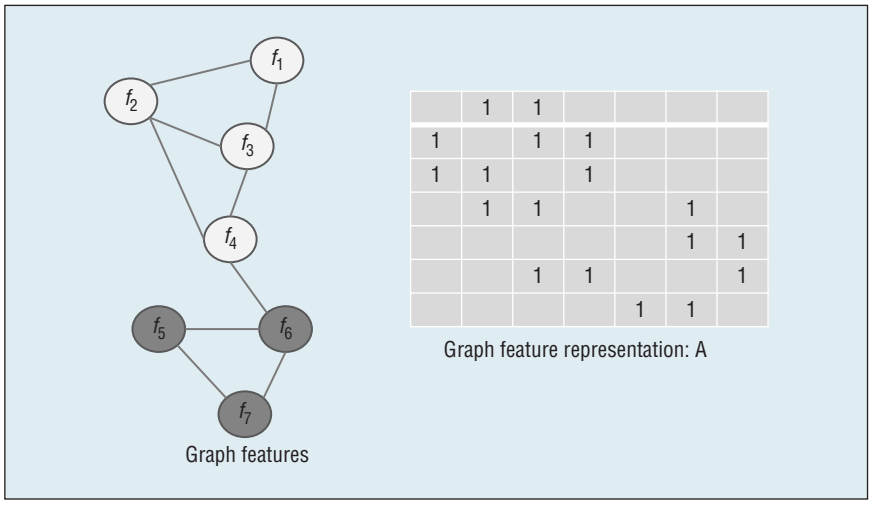

Figure 5. Graph structures among features. These features form a graph structure that can be represented as a weighted adjacency matrix.

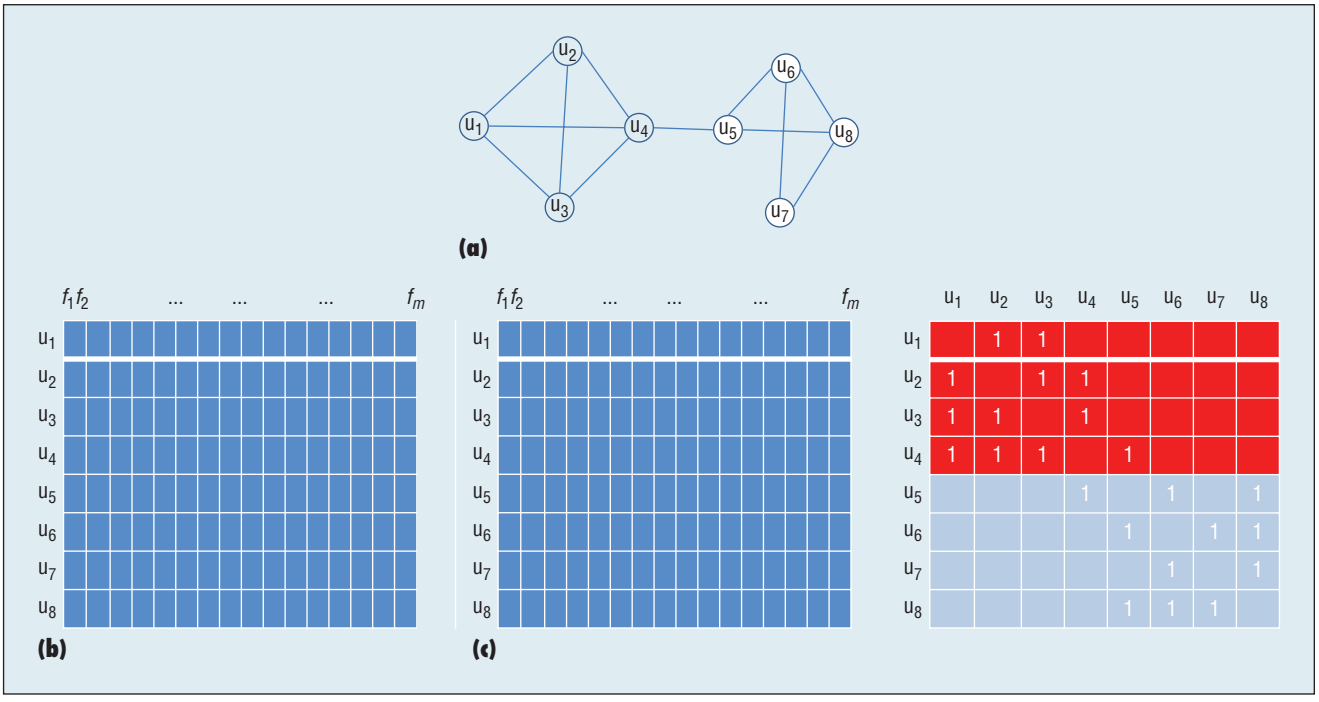

Figure 6. An illustrative example of linked data: (a) instances, (b) attribute-value data, and (c) adjacency matrix. Specifically, (a) and (b) show the adjacency matrix representation and attributed-value representation of these eight linked instances (u_1 to u_8), respectively, while (c) is the complete representation of combing these two.

logical functions, and there are strong dependencies between some genes. Because features show some dependencies, we can model the features by an undirected graph, in which nodes represent features, and edges among nodes show the pairwise dependencies between features. Figure 5 gives an illustrative example of seven features with graph structure.

Linked Data

Linked data is ubiquitous in many networks such as microblogging platforms (tweets linked through hyperlinks on Twitter), social networks (Facebook users connected by friendships), and biological networks (protein interactions). Because linked data is correlated via different types of links, it's distinct from traditional attribute-value data. Figure 6 presents an illustrative example of linked data and its two representations. Figure 6a shows eight linked instances (u_1 to u_8), and Figure 6b gives a conventional representation of attribute-value data such that each row corresponds to one instance and each column corresponds to one feature. As mentioned earlier, linked data provides an extra source of information in the form of links, which can be represented by an adjacency matrix, illustrated in Figure 6c. The challenges of feature selection for linked data[8,9] lie in the following three aspects: how to exploit relations among data instances, how to take advantage of these relations for feature selection, and, since linked data is often unlabeled, how to evaluate the relevance of features without label information to guide the way.

Multisource and Multiview Data

In many data mining and machine learning tasks, we have multiple data sources for the same set of data instances—for example, recent advancement in bioinformatics reveals the existence of some non-coding RNA species in addition to the widely used messenger RNA, and these non-coding RNA species functions cross a variety of biological processes. The availability of multiple data sources makes it possible to address some problems otherwise unsolvable using a single source because the multifaceted representations of data can help depict some intrinsic patterns hidden in a single source of data. For multisource feature selection, we usually have a target source, with other sources complementing the selection of features on the target source.[10]

Multiview sources represent different facets of data instances via different feature spaces that are naturally dependent and also high dimensional, suggesting that feature selection is necessary to prepare these sources for effective data mining tasks such as multiview clustering. A related task of multiview feature selection aims to select features from different feature spaces simultaneously by using their relations.[11] For example, we want to select pixel features, tag them, and text information

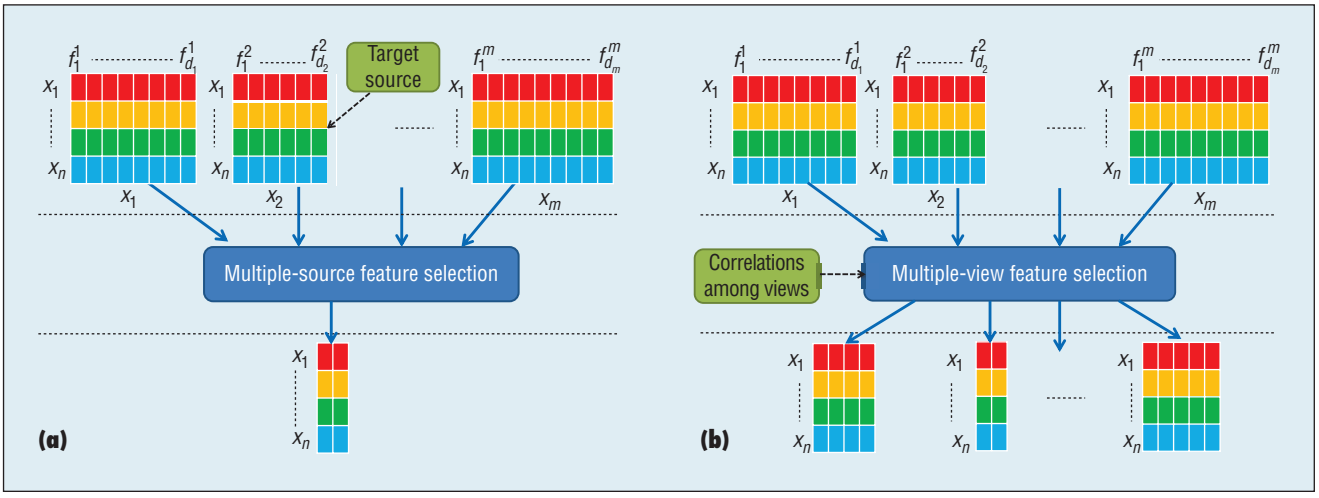

Figure 7. Differences between multisource and multiview feature selection. For multisource feature selection, we usually have a target source among which we perform feature selection. For multiview feature selection, we attempt to find representative features across different views.

about the images to Flickr simultaneously. Because multiview feature selection is designed to select features across multiple views by using their relations, they're naturally different from multisource feature selection (Figure 7).

Streaming Data and Features

In many scenarios, we're faced with a significant amount of data that needs to be processed in real time to gain insights. In the worst cases, data size and features are unknown or even infinite; thus it isn't practical to wait to perform feature selection. For streaming data, one motivating example is online spam email detection: new emails are constantly arriving, so it isn't easy to employ batch-mode feature selection methods in a timely manner. Therefore, some feature selection algorithms maintain and update a feature subset for new data streams (Figure 8). In some settings, when streaming data can't be loaded into memory, one pass of the data is required—we can only make one pass of the data when the second pass is either unavailable or computationally expensive. How to select timely, relevant features in one data pass[12] is still a challenging problem.

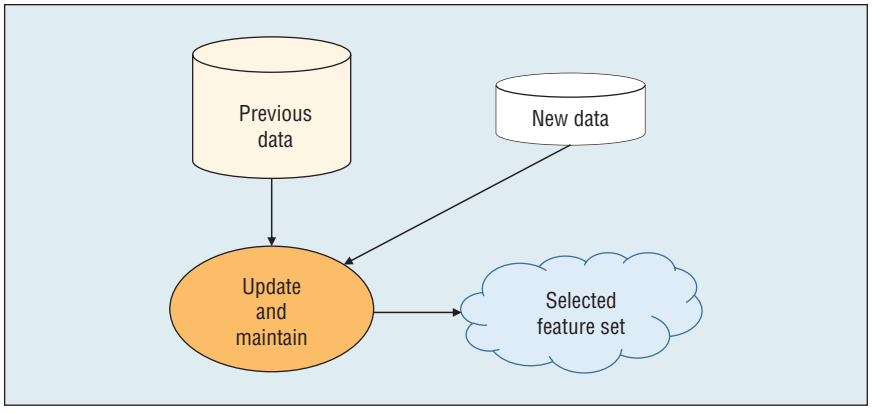

Figure 8. A framework of feature selection in data streams. The target is to update and maintain the selected feature subset upon the arrival of new data samples.

On an orthogonal setting, feature selection for streaming features also has its practical significance. For example, Twitter produces more than 320 million tweets every day, with a large amount of continuously generated slang words (features). These slang words promptly grab users' attention and become popular in a short time. Therefore, it's more preferable to perform streaming feature selection to rapidly adapt to the changes (Figure 9).[13] At each time step, a typical streaming feature selection algorithm first determines whether to accept the most recently arrived feature; if the feature is added to the selected feature set, it then determines whether to discard some existing features from the selected feature set. The process repeats until no new features show up.

Scalability

With the tremendous growth of dataset sizes, the scalability of most current feature selection algorithms could be jeopardized. In many scientific and business applications, data is usually measured in terabytes (1 Tbyte = 10^{12} Bytes). Normally, datasets in the scale of terabytes can't be loaded into

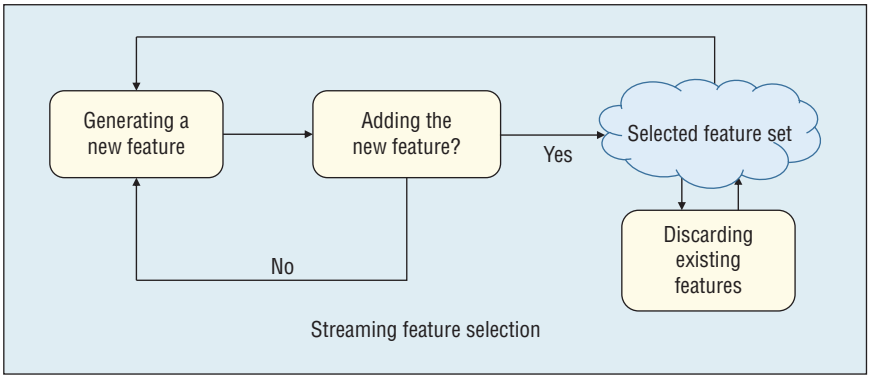

Figure 9. A framework of streaming feature selection. Typically, it consists of two steps: first, it checks whether to accept new features, and then second, if so, it decides whether to remove existing features.

memory directly, thereby limiting the usability of most feature selection algorithms. Currently, there are some attempts to use distributed programming frameworks such as MapReduce and MPI to perform parallel feature selection for very large-scale datasets.[14] Recently, big data of ultrahigh dimensionality has emerged in many real-world applications such as text mining and information retrieval. Most feature selection algorithms don't scale well on ultrahigh-dimensional data; their efficiency deteriorates quickly or is even computationally infeasible. In such cases, well-designed feature selection algorithms in linear or sublinear running time are more preferred.

Stability

Stability is an important consideration when developing new feature selection algorithms.[15] A motivating example from the field of bioinformatics shows that domain experts would like to see the same or a similar set of genes (features) to be selected each time they obtain new samples with a small amount of perturbation. Otherwise, domain experts wouldn't trust these algorithms when they get wildly different sets of features with small data perturbations. Underlying data characteristics can greatly affect feature selection algorithm stability, meaning that the stability issue could also be data dependent—these factors include feature dimensionality, number of data instances, and so on. Studying stability for unsupervised feature selection is much more difficult than it is for supervised methods: in unsupervised feature selection, we don't have enough prior knowledge about data cluster structure, so we're uncertain whether the new data instance belongs to an existing cluster or will introduce new ones.

Feature Selection Repository

To tackle the challenges of feature selection for big data analytics and to promote feature selection research in this community, we present an open source feature selection repository called *scikit-feature* (http://featureselection.asu.edu/). Its purpose is to collect widely used feature selection algorithms and serve as a platform for facilitating their application, comparison, and joint study. The feature selection repository also effectively assists researchers to achieve more reliable evaluation in the process of developing new feature selection algorithms.

Currently, *scikit-feature* consists of popular feature selection algorithms in the following categories:

- similarity-based feature selection,
- information theoretical-based feature selection,
- statistical-based feature selection,
- sparse learning-based feature selection,
- wrapper-based feature selection,
- structural feature selection, and
- streaming feature selection.

Among these different categories of feature selection methods, similarity-, information theoretical-, and statistical-based methods correspond to the filter methods discussed above. Wrapper- and sparse learning-based methods correspond to the wrapper methods and embedded methods, respectively. We also include structural features, linked data, and multiview and multisource data to the category of structural feature selection, and streaming data and features to the streaming feature selection category.

In addition, *scikit-feature* provides many benchmark feature selection datasets and examples of how to evaluate feature selection algorithms via classification or clustering tasks. Experimental results can be obtained from our repository project website (http://featureselection.asu.edu/datasets.php). For each dataset, we list all applicable feature selection algorithms, along with its evaluation on either classification or clustering tasks.

In the future, we'll develop sophisticated feature selection algorithms to tackle the challenges of big data, such as data velocity and data variety. Also, we'll study how to make feature selection algorithms more stable and more scalable. ■

Acknowledgments

This material is, in part, supported by the US National Science Foundation (NSF) under grant numbers IIS-1217466 and 1614576.

THE AUTHORS

Jundong Li is working toward a PhD in computer science and engineering at Arizona State University. His research interests include data mining and machine learning and their applications in social media. Li received an MS in computing science from the University of Alberta, Canada. Contact him at jundong.li@asu.edu.

Huan Liu is a a professor of computer science and engineering at Arizona State University. His research interests are in data mining, machine learning, social computing, and artificial intelligence. Liu received a PhD in computer science from the University of Southern California, Los Angeles. He's a fellow of IEEE and an ACM Distinguished Scientist. Contact him at huan.liu@asu.edu.

References

1. J. Li et al., "Feature Selection: A Data Perspective," 2016; arXiv preprint arXiv:1601.07996.
2. H. Liu and H. Motoda, *Computational Methods of Feature Selection*, CRC Press, 2007.
3. J. Ye and J. Liu, "Sparse Methods for Biomedical Data," *ACM SIGKDD Explorations Newsletter*, vol. 14, no. 1, 2012, pp. 4–15.
4. J. McAuley et al., "Subband Correlation and Robust Speech Recognition," *IEEE Trans. Speech and Audio Processing*, vol. 13, no. 5, 2005, pp. 956–964.
5. S. Ma, X. Song, and J. Huang, "Supervised Group Lasso with Applications to Microarray Data Analysis," *BMC Bioinformatics*, vol. 8, no. 1, 2007, p. 60.
6. R. Jenatton, J.-Y. Audibert, and F. Bach, "Structured Variable Selection with Sparsity-Inducing Norms," *J. Machine Learning Research*, vol. 12, 2011, pp. 2777–2824.
7. C. Fellbaum, *WordNet*, Wiley Online Library, 1998.
8. J. Li et al., "Robust Unsupervised Feature Selection on Networked Data," *Proc. 2016 SIAM Int'l Conf. Data Mining*, 2016, pp. 387–395.
9. J. Tang and H. Liu, "Unsupervised Feature Selection for Linked Social Media Data," *Proc. 18th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining*, 2012, pp. 904–912.
10. Z.A. Zhao and H. Liu, *Spectral Feature Selection for Data Mining*, CRC Press, 2011.
11. H. Wang, F. Nie, and H. Huang, "Multi-view Clustering and Feature Learning via Structured Sparsity," *Proc. 30th Int'l Conf. Machine Learning*, 2013, pp. 352–360.
12. H. Huang, S. Yoo, and S. Kasiviswanathan, "Unsupervised Feature Selection on Data Streams," *Proc. 25th ACM Int'l Conf. Information and Knowledge Management*, 2015, pp. 1031–1040.
13. J. Li et al., "Unsupervised Streaming Feature Selection in Social Media," *Proc. 25th ACM Int'l Conf. Information and Knowledge Management*, 2015, pp. 1041–1050.
14. S. Singh et al., "Parallel Large Scale Feature Selection for Logistic Regression," *Proc. 2009 SIAM Int'l Conf. Data Mining*, 2009, pp. 1172–1183.
15. Z. He and W. Yu, "Stable Feature Selection for Biomarker Discovery," *Computational Biology and Chemistry*, vol. 34, no. 4, 2010, pp. 215–225.

Intelligent IEEE Systems

IEEE Computer Society Publications Office
10662 Los Vaqueros Circle, PO Box 3014
Los Alamitos, CA 90720-1314

Lead Editor
Bonnie Wylie
b.wylie@computer.org

Editorial Management
Jenny Stout

Publications Coordinator
isystems@computer.org

Director, Products & Services
Evan Butterfield

Senior Manager, Editorial Services
Robin Baldwin

Digital Library Marketing Manager
Georgann Carter

Senior Business Development Manager
Sandra Brown

Senior Advertising Coordinator
Marian Anderson
manderson@computer.org

Submissions: For detailed instructions and formatting, see the author guidelines at www.computer.org/intelligent/author.htm or log onto *IEEE Intelligent Systems*' author center at Manuscript Central (www.computer.org/mc/intelligent/author.htm). Visit www.computer.org/intelligent for editorial guidelines.

Editorial: Unless otherwise stated, bylined articles, as well as product and service descriptions, reflect the author's or firm's opinion. Inclusion in *IEEE Intelligent Systems* does not necessarily constitute endorsement by the IEEE or the IEEE Computer Society. All submissions are subject to editing for style, clarity, and length.