LOFS User's Manual

Public domain version 1.0 for MATLAB
Release date: 21/12/2015

Kui Yu[1], Wei Ding[2], and Xindong Wu[3]

[1]University of South Australia, Australia

[2]University of Massachusetts Boston, USA

[3]University of Vermont, USA

Kui.Yu@unisa.edu.au, ding@cs.umb.edu, xwu@uvm.edu.cn

LOFS is a software toolbox for online streaming feature selection. It provides the first open-source library for use in MATLAB that implements the state-of-the-art algorithms of online streaming feature selection.

Contents

1. Overview of LOFS

1.1 Introduction

Traditional online feature selection deals with the observations sequentially added while the total dimensionality is fixed. In contrast, as a novel research direction, online streaming feature selection deals with sequentially added dimensions in feature space while the number of data instances is fixed. Many big data applications call for online streaming feature selection to consume sequentially added dimensions over time. Online streaming feature selection provides a new, complementary algorithmic methodology to enrich online feature selection, especially dealing with high dimensionality in big data analytics. Currently, there are two active research topics in this existing research direction. One is to online learning features added individually, and the other is to mine grouped features added sequentially over time.

The library provides the first open-source library for use in MATLAB that implements the state-of-the-art algorithms of online streaming feature selection. It is designed to facilitate the development of new algorithms in this exciting research direction and make the comparisons between new methods and existing ones available.

1.2 Architecture of LOFS

The LOFS architecture is based on a separation of three modules, that is, CM (Correlation Measure), Learning, and SC (Statistical Comparison), as shown in Figure 1. The learning module consists of two submodules, LFI (Learning Features added Individually) and LGF (Learning Grouped Features added sequentially).

The three modules in the LOFS architecture are designed independently, and all codes follow the MATALB standards. This makes that the LOFS library is simple, easy to implement, and extendable flexibly. One can easily add a new algorithm to the LOFS library and share it through the LOFS framework without modifying the other modules.

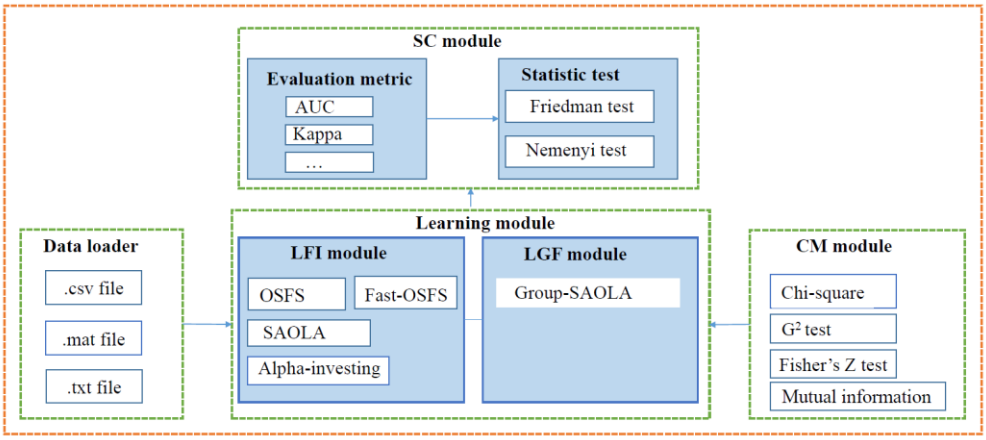

Figure 1 The architecture of LOFS

In the CM module, the library provides four measures, Chi-square test, G^2 test, the Fisher's Z test (Murphy, 2001), and mutual information, to compute correlations between features, where Chi-square test, G^2 test, and mutual information to deal with discrete data while the Fisher's Z test to handle continuous data.

In the learning module, the library provides the following state-of-the-art algorithms of online streaming feature selection for facilitating the development of new algorithms in this exciting research problem and making comparisons between the new and existing methods available.

- Alpha-investing (Zhou et al., 2006). It was proposed by Zhou et al. (2006). It sequentially considers new features as additions to a predictive model by modelling the candidate feature set as a dynamically generated feature stream. Alpha-investing only calculates whether a new coming feature is added to the current feature set, and never considers removing features from the currently selected feature set.

- OSFS (Wu et al., 2010). OSFS maintains a best feature subset from the features available so far by processing each feature upon its arrival with a two-phase subset discovery scheme: online relevance analysis and online redundancy analysis. OSFS not only determine whether to add a new arriving features to the current feature set, but also removes features from the selected feature set currently to keep it as small as possible.

- Fast-OSFS (Wu et al., 2013). It is a fast version of OSFS (Wu et al., 2010). Fast-OSFS divides the process of handling redundant features in OSFS into two steps: (1) determining whether to keep an incoming new feature or not, and (2) identifying which of the selected features observed so far may become redundant once the inclusion of the new feature occurs.

- SAOLA (Yu et al., 2014). To tackle dimensionality in the scale of millions or more, the SAOLA algorithm employs online pairwise comparisons to maintain a parsimonious model over time in an online manner to make online streaming feature selection scalable in big data analytics.

- Group-SAOLA (Yu et al., 2015). In some applications, group information is embedded in feature space. For instance, in image analysis, features are generated in groups which represent colour, texture and other visual information. The group-SAOLA algorithm online yields a set of feature groups that is sparse between groups as well as within each group for maximizing its predictive performance for classification, even the dimensionality in the scale of millions or more.

1.3 Core Function Overview

The core functions provided in the LOFS library are listed in Table 1.

Table 1 Core functions of LOFS

MATLAB Function	Corresponding Algorithm
my_cond_indep_chisquare.m	Chi-square test and G^2 test
my_cond_indep_fisher_z.m	Fisher's Z test
mi.m	mutual information
cmi.m	conditional mutual information
SU.m	symmetrical uncertainty
Alpha_Investing.m	Alpha-investing algorithm
fast-osfs_d.m	Fast-OSFS algorithm with G^2 test
fast-osfs_z.m	Fast-OSFS algorithm with Fisher's Z test
saola_mi.m	SAOLA algorithm with mutual information
saola_z_test.m	SAOLA algorithm with Fisher's Z test
group_f.m	feature grouping algorithm
group-saola_mi.m	group-SAOLA algorithm with mutual information
group-saola_ z_test.m	group-SAOLA algorithm with Fisher's Z test
perfcurve.m	calculating AUC
cal_kappa.m	kappa statistic
Friedmantest.m	Friedman test
Nemenyi test.m	Nemenyi test

2. Setup in MATALB

2.1 Getting and Installing LOFS

The LOFS website is at https://github.com/kuiy/LOFS. LOFS calculates mutual information between variables by calling functions in the library of MIToolbox (Brown et al., 2012)[1]. To run LOFS, it is required that (1) Windows 7 operating system or above version; (2) MATLAB12a, or above to be installed; (3) The folders and subfolders in the LOFS package should be added to the search path in MATLAB, as shown in Figure 2.

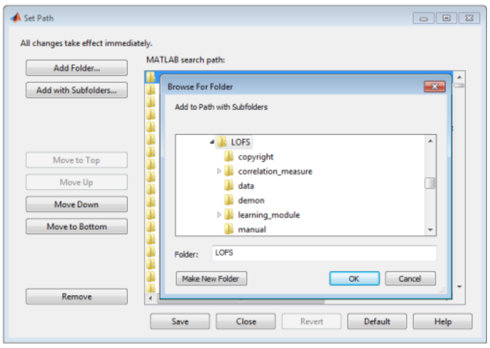

Figure 2 Adding the LOFS package to the MATLAB search path

[1] http://www.cs.man.ac.uk/~gbrown/fstoolbox/

2.2 Data Format

Firstly, it is required that a data set should be correctly imported to MATLAB. Currently, LOFS supports the mat file, the txt file, and the csv file. To read and write the ARFF (Weka Attribute-Relation File Format) files and LIBSVM data files will be supported in our future plan.

Secondly, in a data set, columns represent features[2], rows denote data observations. If a data set includes N features in total, in general, LOFS assumes that the first $N-1$ columns denote the predictive feature matrix, and the last column represents the class attribute vector. An example in MATLAB is given in Figure 3 as follows.

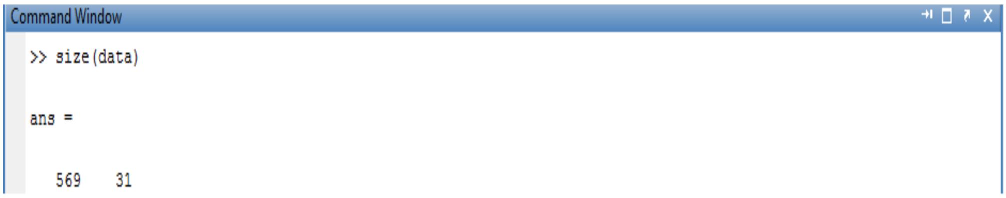

Figure 3 Example of Data format

This denotes that the data set contains 569 data instances, and 31 features. The first 30 columns are predictive features, and the last column is the class attribute.

3. Description of Core functions

3.1 Correlation Module

3.1.1 Chi-square and G^2 test

Function	my_cond_indep_chisquare.m	
	Description: calculate whether two variables are independent or not conditioned on a subset using Chi-square or G^2 test.	
Inputs	data	The data used for training (matrix). For Chi-square test and G^2 test in LOFS, discrete data have to be in a special format: feature X has to take consecutive integer values starting from 1, that is, 1…max_value(X). For example, if max_value(X)=3, this means that feature X takes values {1,2,3}.
	X	index of feature X in data matrix
	Y	index of feature Y in data matrix
	S	Conditioning set. It includes the indices of features in data matrix.
	alpha	Threshold on statistic (the significance level). The parameter is always set to 0.01 or 0.05.
	test	Statistical test desired to use. The parameter *test*= 'chi2' for Pearson's Chi-square test and *test*='g2' for G^2 likelihood ratio test.
	ns	Vector with the sizes of the corresponding maximum values for all variables in data matrix (default: max(data), as shown in Figure 3). For example, *ns* = [2 2 3]. This specifies that the values of the first and the second feature which can take is {1, 2} respectively, and the values of the third one that takes is {1, 2, 3}. The parameter *ns* should be empty (i.e. []) if Fisher's Z test is used (since variables are continuous.).
Output	CI	test result (1=conditional independency, 0=dependency)

[2] In the manual, we use the terms “attribute”, “variable”, and “feature” interchangeably.

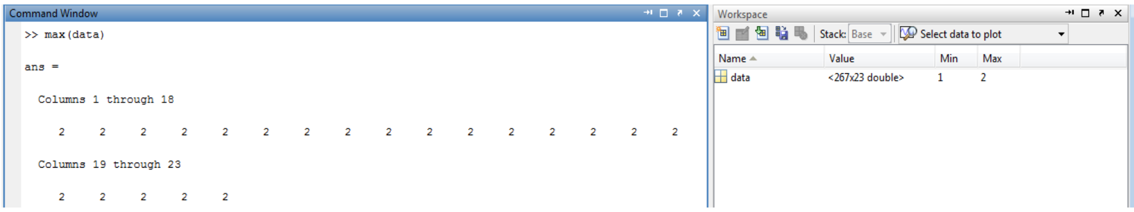

Figure 4 Example of calculating the parameter *ns*

3.1.2 Fisher's Z test

Function	my_cond_indep_fisher_z.m	
	Description: calculate whether two variables are independent or not conditioned on a subset using Fisher's Z test.	
Inputs	data	The data used for training (matrix)
	X	index of feature X in data matrix
	Y	The index of variable Y in data matrix. The variable Y in data matrix should take consecutive integer values starting from 0 for classification if Y is the class attribute.
	S	Conditioning set. It includes the indices of features in data matrix.
	N	Total number of data observations
	alpha	Threshold on statistic (the significance level). The parameter is always set to 0.01 or 0.05.
Outputs	CI	Test result (1=conditional independency, 0=dependency)
	dep	Dependency value of X with respect to Y conditioned on S
	p-value	p-value at the significance level of alpha

3.1.3 Mutual Information

Function	mi.m	
	Description: compute mutual information between variables.	
Inputs	X	A feature vector (or a column) in data matrix
	Y	A feature vector in data matrix. It is distinct from X.
Output	score	Mutual information

3.1.4 Conditional Mutual Information

Function	cmi.m	
	Description: calculate conditional mutual information.	
Inputs	X	A feature vector (or a column) in data matrix
	Y	A feature vector in data matrix. It is distinct from X.
	Z	Conditioning set
Output	score	Conditional mutual information

3.1.5 Symmetrical Uncertainty

Function	SU.m	
	Description: calculate symmetrical uncertainty between variables.	
Inputs	firstVector	A feature vector (or a column) in data matrix
	secondVector	A feature vector in data matrix. It is distinct from the first parameter.
Output	score	symmetrical uncertainty

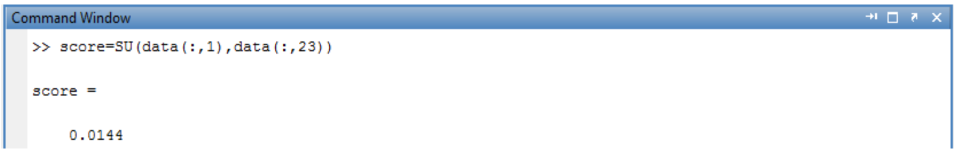

Figure 5 Example of calculate symmetrical uncertainty

3.2 LFI Module

3.2.1 Alpha-Investing

Function	Alpha_Investing.m	
	Description: implement the Alpha_Investing algorithm.	
Inputs	X	Matrix of the predictive features in data matrix
	Y	Vector of the class attribute in data matrix
Outputs	selectFeatures	Vector with the indices of the selected features in data matrix
	time	Computational cost (in seconds)

3.2.2 OSFS for Discrete Data

Function	osfs_d.m	
	Description: implement the OSFS algorithm for discrete data.	
Inputs	data	The discrete data used for training (matrix). For Chi-square test and G^2 test in LOFS, discrete data have to be in a special format: feature X has to take consecutive integer values starting from 1, that is, 1…max_value(X). For example, if max_value(X)=3, this means that feature X takes values {1,2,3}.
	class_index	Index of the class attribute in data matrix. In general, LOFS assumes that the last column represents the class attribute vector in the input data set.
	alpha	Threshold on statistic (the significance level). It is always set to 0.01 or 0.05.
	test	Statistical test desired to use. The parameter test= 'chi2' for Chi-square test and test='g2' for G2 test.
Outputs	selectFeatures	Vector with the indices of the selected features in data matrix
	time	Computational cost (in seconds)

3.2.3 OSFS for Continuous Data

Function	osfs_z.m Description: implement the OSFS algorithm for continuous data.	
Inputs	data	The continuous data used for training matrix
	class_index	Index of the class attribute in data matrix. In general, LOFS assumes that the last column represents the class attribute vector in the input data set.
	alpha	Threshold on statistic (the significance level). It is always set to 0.01 or 0.05.
Outputs	selectFeatures	Vector with the indices of the selected features in data matrix
	time	Computational cost (in seconds)

3.2.4 Fast-OSFS for Discrete Data

Function	fast_osfs_d.m Description: implement the Fast-OSFS algorithm for discrete data.	
Inputs	data	The discrete data used for training (matrix). For Chi-square test and G2 test in LOFS, discrete data have to be in a special format: feature X has to take consecutive integer values starting from 1, that is, 1…max_value(X). For example, if max_value(X)=3, this means that feature X takes values {1,2,3}.
	class_index	Index of the class attribute in data matrix. In general, LOFS assumes that the last column represents the class attribute vector in the input data set.
	alpha	Threshold on statistic (the significance level). It is always set to 0.01 or 0.05.
	test	Statistical test desired to use. The parameter test= 'chi2' for Chi-square test and test='g2' for G2 test.
Outputs	selectFeatures	Vector with the indices of the selected features in data matrix
	time	Computational cost (in seconds)

3.2.5 Fast-OSFS for Continuous Data

Function	fast_osfs_z.m Description: implement the Fast-OSFS algorithm for continuous data.	
Inputs	data	The continuous data used for training matrix
	class_index	Index of the class attribute in data matrix. In general, LOFS assumes that the last column represents the class attribute vector in the input data set.
	alpha	Threshold on statistic (the significance level). It is always set to 0.01 or 0.05.
Outputs	selectFeatures	Vector with the indices of the selected features in data matrix
	time	Computational cost (in seconds)

3.2.6 SAOLA for Discrete Data

Function	saola_mi.m Description: implement the SAOLA algorithm for discrete data.	
Inputs	data	The discrete data used for training (matrix). Each feature in data matrix has to take consecutive integer values starting from 0 or 1.
	class_index	Index of the class attribute in data matrix. In general, LOFS assumes that the last column represents the class attribute vector in the input data set.
	threshold	Threshold on statistic for mutual information measure
Outputs	selectFeatures	Vector with the indices of the selected features in data matrix
	time	Computational cost (in seconds)

3.2.7 SAOLA for Continuous Data

Function	saola_z_test.m Description: implement the SAOLA algorithm for continuous data.	
Inputs	data	The continuous data used for training (matrix), and the class attribute in data matrix has to take consecutive integer values starting from 0 or 1 for classification.
	class_index	Index of the class attribute in data matrix. In general, LOFS assumes that the last column represents the class attribute vector in the input data set. The class attribute has to take consecutive integer values starting from 0 for classification.
	alpha	Threshold on statistic (the significance level). It is always set to 0.01 or 0.05.
Outputs	selectFeatures	Vector with the indices of the selected features in data matrix
	time	Computational cost (in seconds)

3.3 LGF Module

3.3.1 Group Splitting

Function	group_f.m Description: divide features in different groups randomly.	
Inputs	features_index	Vector with indices of attributes except for the class attribute
	numberGroups	Number of groups desired to divide
Output	group_feature	Feature groups (a cell array of group indices). If the dimensionality of a data set is divided into k distinct groups, each group includes the indices of features assigned to this group.

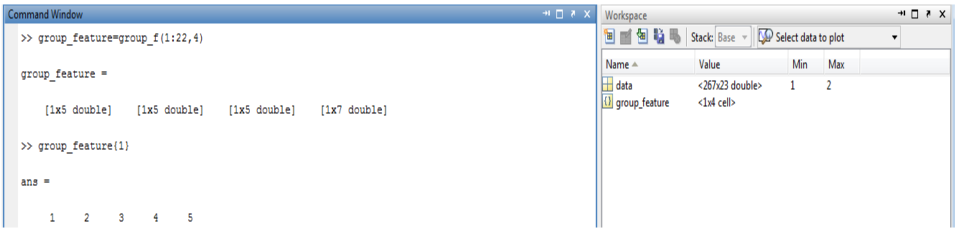

Figure 6 Example of group splitting

3.3.2 group-SAOLA for Discrete Data

Function	saloa_group_mi.m	
	Description: perform the group-SAOLA for discrete data.	
Inputs	group_feature	Cell array of group indices. You can implement the group_f.m function in the library to get the input parameter.
	data	Discrete data used for training (matrix)
	class_index	Index of the class attribute in data matrix
	threshold	Threshold on statistic for mutual information measure
Outputs	selectFeatures	Vector with the indexes of the selected features
	selectGroups	Number of selected groups
	time	Computational cost (in seconds)

3.3.3 group-SAOLA for Continuous Data

Function	saloa_group_z_test.m	
	Description: perform the group-SAOLA for continuous data.	
Inputs	group_feature	Cell array of group indices. You can implement the group_f.m function in the library to get the input parameter.
	data	Continuous data used for training (matrix), and the class attribute in data matrix has to take consecutive integer values starting from 0.
	class_index	Index of the class attribute in data matrix
	alpha	Threshold on statistic (the significance level). It is always set to 0.01 or 0.05.
Outputs	selectFeatures	Vector with the indexes of the selected features
	selectGroups	Number of selected groups
	time	Computational cost (in seconds)

4. Evaluation in MATLAB

Except for using the efficiency (running time), prediction accuracy, and compactness (the ratio of the number of selected features and the total number of features seen so far), in the SC module, AUC (the metric embedded in MATLAB), kappa statistic[3], Friedman test, and Nemenyi test (Demšar, 2006; Li, 2013) are employed to conduct comprehensively empirical comparisons.

[3] http://au.mathworks.com/matlabcentral/fileexchange/15365-cohen-s-kappa

4.1 AUC and Prediction Accuracy

Function	perfcurve.m[4]	
	Description: compute AUC	
Inputs	Labels	True test class labels
	Scores	Predicted class labels
	Posclass	the positive class label
Output	AUC	the area under curve

4.2 Kappa Statistic

Function	cal_kappa.m	
	Description: compute Cohen's kappa coefficient	
Inputs	Labels	True (actual) class labels in test data
	Prelabels	Predicted class labels for test data
	par	The parameter is set to 'class' for classification or 'regress' for regression.
Output	kappa	Estimated Cohen's Kappa coefficient

```
%use KNN clasifier (k=3)
test_class =
knnclassify(testdata(:,selectedFeatures),traindata(:,selectedFeatures),traindata(:,class_index),3);
%calculate AUC, prediction accuracy, and kappa
[X,Y,T,AUC] = perfcurve(testdata(:,class_index),test_class,1);
accuracy=length(find(testdata(:,class_index) == test_class))/length(test_class);
kappa = cal_kappa(class_label(test_indices),test_class,'class');
```

Figure 7 Example of computing AUC, prediction accuracy, and kappa

4.3 Statistical Test

4.3.1 Friedman test

Function	Friedmantest.m	
	Description: Friedman test for comparing multiple methods over multiple data sets	
Inputs	error	error rate matrix of methods
	alpha	Threshold on statistic (the significance level).
Outputs	acceptF	If the parameter equals to 0, then the null hypothesis is rejected.
	rankCl	Vectors of ranks for compared methods

[4] More information about the function, please use "help perfcurve" in MATLAB.

4.3.2 Nemenyi test

Function	Nemenyitest.m	
	Description: Nemenyi test for comparing multiple methods over multiple data sets	
Inputs	error	error rate matrix of methods
	alpha	Threshold on statistic (the significance level)
Output	CD	Critical values

5. Examples of Algorithm implementation

To work with examples below, we downloaded the *wdbc* and *spect* data sets from the UCI repository as sample data sets. The *wdbc* data set contains 569 data instances and 31 features. The first 30 features are continuous predictive attributes and the last one is the class attribute. The *spect* data set includes 267 data instances and 23 binary features (the last column is the class attribute). Both data sets are loaded into the MATLAB workplace, respectively, as shown in Figure 6.

Figure 6 loading the wdbc and spect data sets

5.1 Example for Discrete Data

Algorithm	Example
Fast-OSFS	[selected_features, time]=fast_osfs_d(spect, 23, 0.01,'g2')
Grouping feature	[group_feature] = group_f(1:22,4)
group-SAOLA	[group_feature] = group_f(1:22,4)
	[selected_features, select_group, time] = saloa_group_mi(group_feature, spect, 23, 0)

5.2 Example for Continuous Data

Algorithm	Example
Fast-OSFS	[selectFeatures, time]=fast_osfs_z(wdbc, 31, 0.01)
Alpha_Investing	[selectFeatures, time]= Alpha_Investing (wdbc(:,1:30), wdbc(:,31))
SAOLA	[selectFeatures, time] = saola_z_test(wdbc, 0.01)
group-SAOLA	[group_feature] = group_f(1:30,6)
	[selectFeatures,selectGroup,time] = saloa_group_z_test(group_feature, wdbc, 31, 0.01)

Reference

Gavin Brown, Adam Pocock, Ming-Jie Zhao, and Mikel Luján. (2012) Conditional Likelihood Maximisation: A Unifying Framework for Information Theoretic Feature Selection. The Journal of Machine Learning Research (JMLR), 13: 27-66.

Janez Demšar (2006). Statistical comparisons of classifiers over multiple data sets. The Journal of Machine Learning Research, 7, 1-30.

Kevin Murphy. (2001) The bayes net toolbox for matlab. Computing science and statistics, 33(2), 1024-1034.

Peter Spirtes, Clark N. Glymour, and Richard Scheines. (2000) Causation, prediction, and search, second ed. MIT Press.

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

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

Kui Yu, Xindong Wu, Wei Ding, and Jian Pei. (2014) Towards scalable and accurate online feature selection for big data. In Proceedings of the 14th IEEE International Conference on Data Mining (ICDM'14), 660-669.

Kui Yu, Xindong Wu, Wei Ding, and Jian Pei. (2015) Scalable and accurate online feature selection for big data. arXiv:1511.09263v1 [cs.LG], 2015.

Yifeng Li. (2013) Sparse representation for high-dimensional data analysis, in "Sparse Machine Learning Models in Bioinformatics", PhD Thesis, School of Computer Science, University of Windsor, Canada, 2013. Thesis Available at http://scholar.uwindsor.ca/etd/5023.

Jing Zhou, Dean P. Foster, Robert A. Stine, and Lyle H. Ungar. "Streamwise feature selection." The Journal of Machine Learning Research 7 (2006): 1861-1885.