Unsupervised Learning and Dimensionality Reduction

Zheyuan Xu

March 2019

Abstract

The following text provides implementation and specific analysis on 6 unsupervised learning algorithms, of two approaches—clustering and dimensionality reduction. Clustering involves K-means clustering and expectation maximization. For the latter, the algorithms are PCA(principle component analysis), ICA(independent component analysis), RP(randomized projections) as well as IG(information gain) respectively. The hyperparameters will be tuned for each algorithm, and the results generated by the new unsupervised learners will be compared against the results generated by neural network learner.

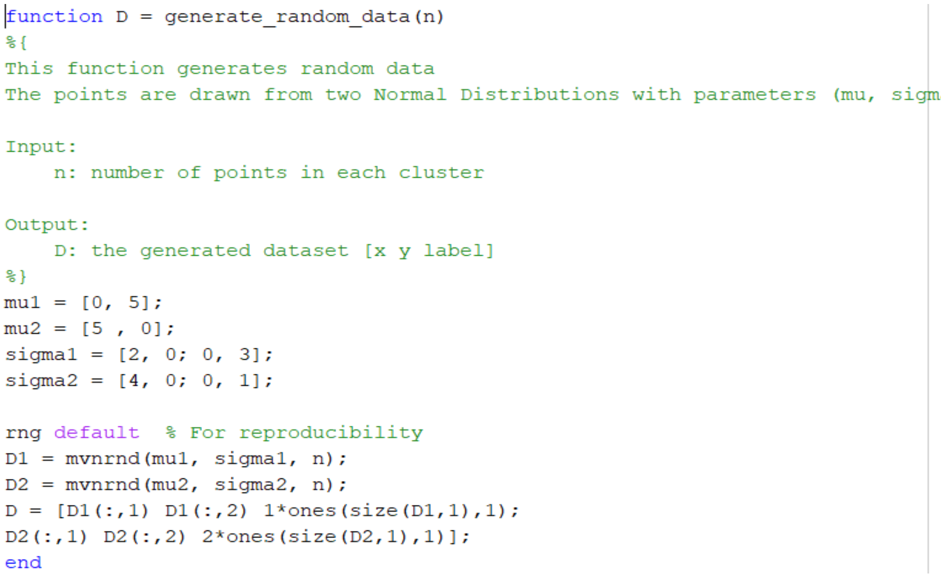
Introduction and Overview of the Algorithms

Datasets

For the first part of implementation (K-means and expectation-maximization) we will use randomly generated 2-D data that follows normal distribution. The reason that the randomized datasets are chosen is caused by several reasons:

- Availability of the datasets
- Ease of implementation (for Caltech101patches as well as ionosphere data, the K value would be 13 and 101 respectively, which makes labelling and plotting extremely hard)
- Similar datasets provide room for comparison

The datasets are generated by MATLAB code specified as follows:



```
function D = generate_random_data(n)
%{
This function generates random data
The points are drawn from two Normal Distributions with parameters (mu, sigma)

Input:
    n: number of points in each cluster

Output:
    D: the generated dataset [x y label]
%}
mu1 = [0, 5];
mu2 = [5 , 0];
sigma1 = [2, 0; 0, 3];
sigma2 = [4, 0; 0, 1];

rng default  % For reproducibility
D1 = mvnrnd(mu1, sigma1, n);
D2 = mvnrnd(mu2, sigma2, n);
D = [D1(:,1) D1(:,2) 1*ones(size(D1,1),1);
D2(:,1) D2(:,2) 2*ones(size(D2,1),1)];
end
```

For the second part of implementation which involves ICA, PCA, RP, as well as IG, we will use two datasets:

- **Ionosphere dataset from UCI**: which contains 34 continuous data fields, with the last column containing the binary label of whether “good” or “bad”. The more detailed description of the dataset can be found on the webpage: https://archive.ics.uci.edu/ml/datasets/ionosphere
- **Caltech101patches:** datasets containing the pictures of objects belonging to 101 categories (hence the name “101”), with about 40-800 images per category, more details can be found at: http://www.vision.caltech.edu/Image_Datasets/Caltech101/

Those two datasets suffice to showcase the pros, cons as well as the performance of each dimensionality reduction algorithms, and could all be implemented by neural network learner (not a deep learning one in this case).

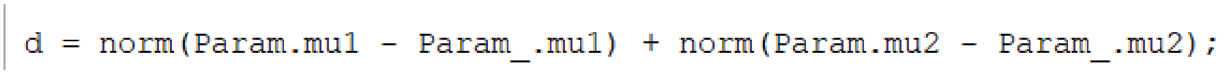
Further, there are also other invariants used in the analysis of 6 algorithms:

- All algorithms are implemented in MATLAB, and plotted in MATLAB. The choice of the platform includes multiple factors including ease of implementation as well as plotting.
- The neural network learner is based on MATLAB deep learning toolbox. Resources and documentations can be found on their websites

Clustering

Introduction to clustering algorithms

In unsupervised learning algorithms, clustering refers to the process of taking in a set of data with points, or continuous values, and, without even knowing the specific attributes of data, grouping the data into clusters that possess similar attributes. The similarity of data points are given by metrics such Euclidean or Manhattan distance. Here it is calculated by a MATLAB script called calc_distance and has the following code inside the main function:



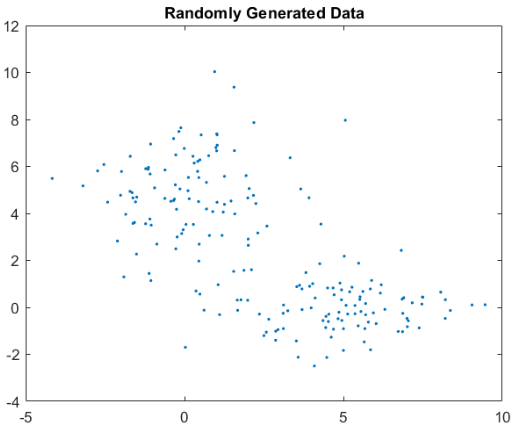
```
d = norm(Param.mu1 - Param_.mu1) + norm(Param.mu2 - Param_.mu2);
```

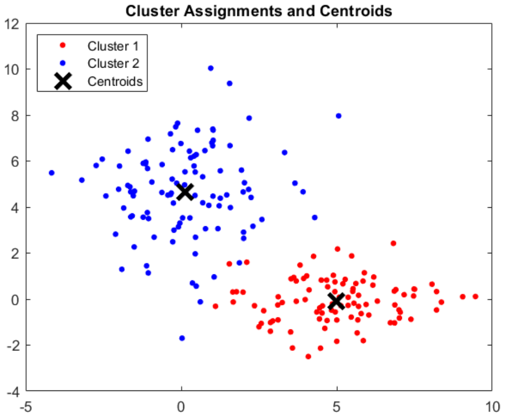
K-means clustering algorithm

As mentioned above the K-means algorithm is implemented on randomly generated 2D normal data. The parameters used in generating the data are kept same so parallel comparisons between K-means as well as EM are made possible. The random nature of the data also provides robustness of the comparison since the conclusions will be more immune to occasion cases.

As our first algorithm to be implemented, we should take a closer look at what K-means does specifically. This algorithm starts by randomly picking K centers, which might or might be the optimal centers, and then new data point is classified to belong to one of those centers which is closest to the data point itself. The new center is updated by taking the average of all data points already belong to that center. The algorithm terminates by convergence or when the movement of centers are less than a threshold value (in theory it could never stop moving when new data points are introduced).

The parameters in which we can only tune one hyper-parameter which is K, here it is chosen to be 2:

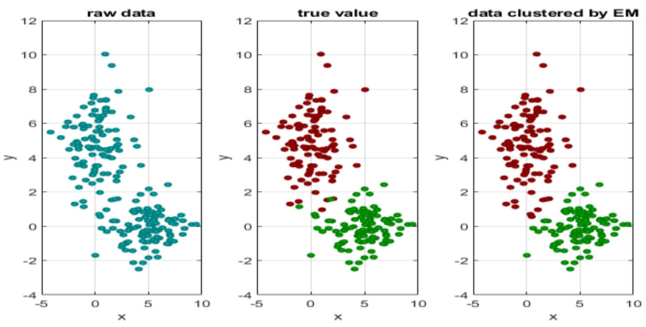




Above are randomly generated dataset used for K-means implementation, the graph on the right indicates the "centers" generated after the algorithm terminates. Notice that in this algorithm, the accuracy depends largely on when it terminates. We could set up a rule specifying that, if the algorithm converges, it automatically stops taking in more data, in order to prevent overfitting. However, without more knowledge of the original datasets, K-means could be easily overfitting as well as underfitting. When the input data is not enough, or contains high noise, it could be insufficient to produce a well-fitting K-mean model. In the case that it contains way too many data points that produce overfitting due to the new centers taking account of the noisy data, it could lead to overfitting. There are no ways to avoid such error if the hyper-parameters are not well tuned, that said, the K value as well as the "terminating threshold value". K-means also suffer from the curse of dimensionality—the higher K is, the more data needed to fill in the entire learning space due to rapidly expanding nature of high-dimensional space. K-means often suffers from lack of input data especially when the desired features are way more than predicted.

Expectation-maximization clustering algorithm (EM)

Unlike K-means which only allow a point to belong to one center, EM algorithm is a soft clustering algorithm, which allows a point to potentially stay in all the centers. The algorithm involves two steps: expectation, maximization. In the first step, each point is assigned a probability to stay in each of the K clusters, which corresponds to the Gaussian distribution of each cluster. We can calculate our expectation for each cluster to contain such a datapoint, using log likelihood, that is, how likely is such a datapoint to be in such a cluster, given all other clusters exist. We then try to find the most likely cluster that the data point could belong to, a.k.a. the maximization step. The results, if more and more datapoints are inputted, will be more fitting to a gaussian model, since gaussian model has the largest log likelihood. Again, the dataset used is also randomly generated and the results are plotted as follows:



The results clustered by EM has relatively high accuracy. However, notice that in the boundary area between two clusters, there are mis-classified points. That does not mean that EM algorithm is inferior to K-means, and on the contrary, K-means also has some datapoints mis-classified on the boundary. The optimal gaussian model approach in EM is unable to entirely eliminate all noisy data points and a perfectly clustered model if the data points are insufficient.

Conclusion for Clustering

Further, due to the nature of EM algorithms, it is more reliable than K-means since it focuses on maximizing the information gain of all data points in classifying. The gaussian model guarantees that as more and more data points are inputted, the total information gain will approach maximum, which somewhat guarantees that the model will be much better. It also makes the clusters less susceptible to noises.

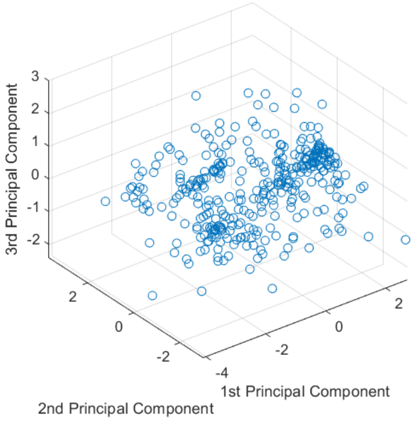
Dimensionality Reduction

Introduction of Dimensionality Reduction Approach

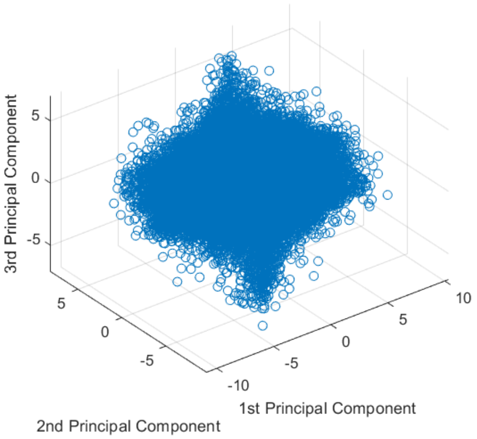
Unlike clustering which attempts to discover some hidden features of the datasets, dimensionality attempts to only keep the most critical features, while still retaining satisfactory results of classification. By eliminating unnecessary or redundant features, dimensionality reduction also tends to reduce the curve of dimensionality, thus reducing the time as well as data required for a learning algorithm. Generally, there are two approaches: feature selection and feature transformation. The former involves finding combinations of the subset of existing features which effectively retain the original information of the data, or even improve the performance of the learning algorithm. The latter instead, is the reconstruction of derived features that are linear or non-linear transformations of the existing features into smaller feature space. The projections still retain similar accuracy, therefore, making the problem linearly separable and less complex to solve. Speaking of that, feature transformation is more advanced method than feature selection, being more capable in dealing with insufficient indicators, false positives as well as false negatives. However, feature selection is faster, and works quite well in some simple cases in which the existing features are already orthogonal. In our case, the four remaining algorithms yet to be implemented are all feature transformation algorithms.

Principal Component Analysis (PCA)

We start with PCA, which is a feature transformation algorithm that interprets the problem as eigenproblems, and attempts to find a linear transformation of the features, so that the projections of data onto the transformed axes have maximum variance. The transformed axes are known as the principal component vector. Orthogonal axes are searched afterwards and each axis has a decreasing eigenvalue which makes it less “important” in classifying the data points.



Results obtained by Running Ionosphere Data



Results obtained by Running Caltech101patches

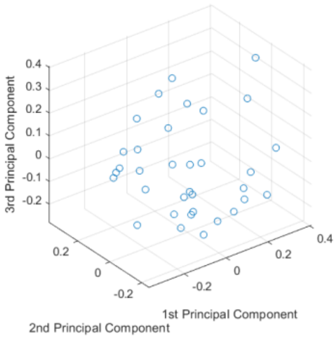
As can be seen in the graphs above, PCA works quite well with datasets which have few critical attributes taking up most of the weights. The first principal component has significantly larger weight than the rest of the components, as the data points are heavily directed in that axis. When situation becomes more complex, say, the datasets contain data influenced by many features with similar weights, PCA does not show the datasets are influenced by some particular features. Instead, the distribution would be a rounder sphere if more components are plotted in 3D space. The algorithm does poor separation of the original data in this case.

Independent Component Analysis (ICA)

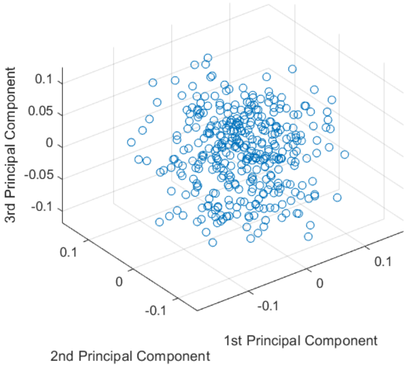
ICA works similarly as PCA, except that it attempts to generate independent components from independent hidden features in a dataset, that provide no mutual information to each other, but maximizing the interaction between inputs and desired outputs. That said, it attempts two goals:

- I(Xi, Xj) = 0
- I(Xi, Yi) = 1

In which “I” is the measure of independence between two variables, X is the input data and Y is the output data.



Results obtained by Running Ionosphere Data with q = 13

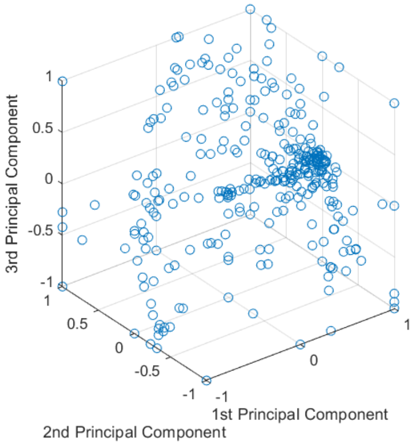


Results obtained by Running Caltech101patches with q = 101

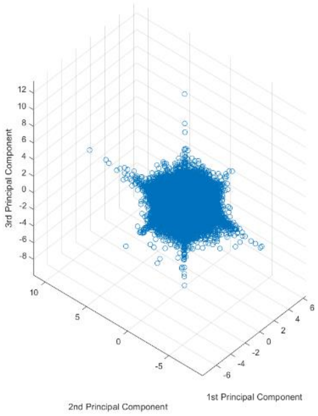
As seen in the pictures above, the distribution for ionosphere dataset is rather sparse and this phenomenon could be caused by the fact that unlike PCA which unconditionally chooses the axis with the largest weight, ICA focuses more on the hidden features which make those axes codependent, separates those features, and generate independent components based on them. The result which could potentially make PCA biased can be “rectified” by ICA. For Caltech101patches data, ICA still performs quite well since the datasets contain a lot of features which are equally important and mostly independent (images for 101 different objects) despite some subtle similarities (such as the similarity between an airplane and a car). The relatively sparse distribution of ICA compared to other algorithms might be caused by the curse of dimensionality, in which the number of data points required by a learning algorithm greatly increases following the increase in the number of features. The extremely sparse data points in the first graph illustrates this issue, which limits ICA’s capability in successfully reconstructing the data points.

Randomized Projections (RP)

RP, unlike PCA and ICA, is much faster and equally powerful. The randomized projections, like its name, works by accounting for the fact that sometimes the mostly weighted components are codependent with features unknown to us, and a seemingly higher projection might capture those hidden features which are misleading for the classifier to generate the optimal model.



Results obtained by Running Ionosphere Data

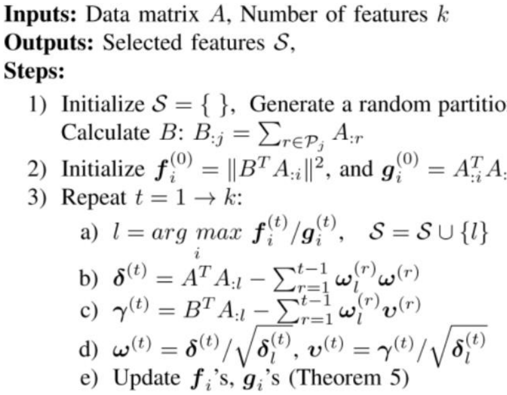


Results obtained by Running caltech101patches

As can be seen, randomized projection works better than PCA when some components with heavy weights are codependent on some hidden features, since the data points are better scattered along all three axes. It, however, performs significantly better in cases where there are multiple features with equally important weights, as can be seen from the "synapses" in the second graph. The randomized feature makes it less vulnerable to problems of codependence, while not being too strict in the independence requirement of the components. As more and more axes are plotted in 3D, the graph will end up a small center with hundreds of synapses radiating from the center.

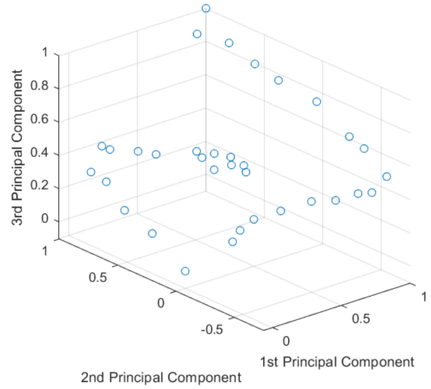
Greedy Feature Selection (GFS)

Unlike PCA, ICA, as well as RP, which are all feature transformation algorithms, GFS is feature selection algorithm. Feature selection algorithms, while being relatively susceptible to insufficient indicators and false claims, works quite well with data points with simpler internal structures and have the advantage of being computationally inexpensive. The pseudo-code is as follows:

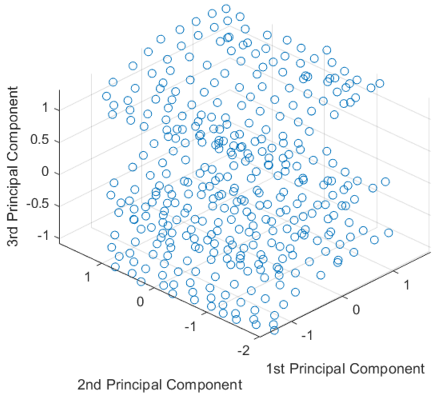


```
Inputs: Data matrix $A$, Number of features $k$
Outputs: Selected features $\mathcal{S}$,
Steps:
  1) Initialize $\mathcal{S} = \{\ \}$, Generate a random partitioning $P$,
     Calculate $B$: $B_{:j} = \sum_{r \in \mathcal{P}_j} A_{:r}$
  2) Initialize $\boldsymbol{f}_i^{(0)} = \|B^T A_{:i}\|^2$, and $\boldsymbol{g}_i^{(0)} = A_{:i}^T A_{:i}$
  3) Repeat $t = 1 \rightarrow k$:
       a) $l = \underset{i}{arg\ max}\ \boldsymbol{f}_i^{(t)} / \boldsymbol{g}_i^{(t)}$,   $\mathcal{S} = \mathcal{S} \cup \{l\}$
       b) $\boldsymbol{\delta}^{(t)} = A^T A_{:l} - \sum_{r=1}^{t-1} \boldsymbol{\omega}_l^{(r)} \boldsymbol{\omega}^{(r)}$
       c) $\boldsymbol{\gamma}^{(t)} = B^T A_{:l} - \sum_{r=1}^{t-1} \boldsymbol{\omega}_l^{(r)} \boldsymbol{\upsilon}^{(r)}$
       d) $\boldsymbol{\omega}^{(t)} = \boldsymbol{\delta}^{(t)} / \sqrt{\boldsymbol{\delta}_l^{(t)}}$, $\boldsymbol{\upsilon}^{(t)} = \boldsymbol{\gamma}^{(t)} / \sqrt{\boldsymbol{\delta}_l^{(t)}}$
       e) Update $\boldsymbol{f}_i$'s, $\boldsymbol{g}_i$'s (Theorem 5)
```

GFS works by calculating inner products of randomly partitioned clusters, and generating scores for feature selection. The algorithm then updates each clusters' scores by excluding previously selected feature, until it reaches the number of clusters (as specified by 'k' in the algorithm); the algorithm can be further speeded by decreasing the number of partitions, at the cost of lower accuracy:



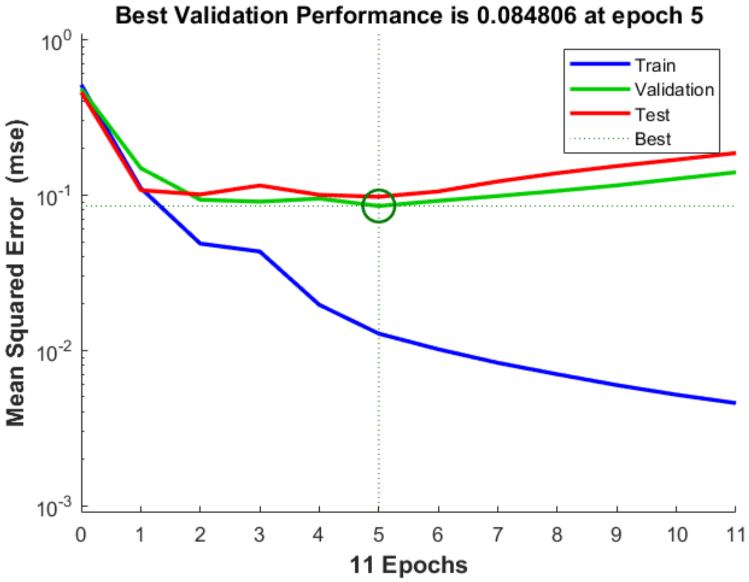
Results obtained by Running Ionosphere Data with K = 32 (same as n with no potential loss of accuracy)



Results obtained by Running caltech101patches Data with K = 363 (same as n with no potential loss of accuracy)

As can be seen from the graphs, GFS does not work quite well since it does not explore the internal structure of the data points. The first graph is rather superficial for the datasets and hardly illustrates any relationship between them. The result by running on Caltech101patches fails miserably since it is showing incorrect correlation between different components which should have minimized interaction. The problem is still caused by superficial choice of existing features especially with greedy approach, which ignores the validity of such features. The algorithm, however, can be run fairly fast, and only takes seconds to terminate.

Results generated Running Neural-Network Learner:



Results obtained by Running Ionosphere Data

Since the caltech101patches data contains 101 categories as its results, it is not trainable by conventional neural network learner. However, the training can still be done by a CNN (convolutional neural network) or similar deep learning algorithms. In this case it is not of our consideration since such algorithms might outperform all the other ones mentioned above by a great margin, as it directly simulates visual neurons and is highly effective in recognizing images.

The result by running neural network is rather poor, as it hardly reaches 90% accuracy. This might be caused by the lack of internal hidden layers. The neural network learner struggles to find out the internal correlation between multiple features and only works for binary outputs.

Conclusion

There is a list of reasons why dimensionality reduction is generally preferred over feature selection. First of all, the datasets generated from most real-life examples are noisy and complex enough, that feature selection (basically recombination of subsets of features) might appear insufficient especially in most cases the data relies on so many different features. On the other hand, while feature selection is computationally inexpensive, the advantage is being continuously weakened by introduction of newer and more powerful GPU as well as parallel computation tools (such as Google Tensorflow) into the market, which makes the latter more suitable for nowadays' needs. Even in the subcategory of clustering methods, EM is preferable to K-means since it takes into account the information gain and Gaussian distribution of the data. For dimensionality reduction methods, it depends more on the internal structures of the datasets, as well as hyper-parameters. ICA needs a lot of data especially when it has to face datasets with many features, since it is more vulnerable to the curse of dimensionality. Randomized projection works quite well by making a compromise between the independence of each component in ICA and the number of data points required in PCA, and ends up being the most optimal. Feature selection algorithms, on the other hand, often fails to discover the real hidden features of the data and unless computational resources are rather limited, should be put aside. It is also true for neural network which does poorly on multi-dimensional data with many features, which are sometimes inseparable by linear separators.

Bibliography

[1] Farahat, A.K., Elgohary, A., Ghodsi, A., and Kamel, M.S. (2014)
Greedy Column Subset Selection for Large-scale Data Sets. Knowledge and Information Systems. 33 pages. Under Review. Accepted http://arxiv.org/abs/1312.6838

[2] Ahmed K. Farahat, Ali Ghodsi, Mohamed S. Kamel, "An Efficient Greedy Method for Unsupervised Feature Selection", In Proceedings of the Eleventh IEEE International Conference on Data Mining, pp.161-170, 2011 http://dx.doi.org/10.1109/ICDM.2011.22

[3] Sawasthi. Principal Component and Factor Analysis. Department of Mechanical Statistics at the University of Texas, Arlington.