Unsupervised Learning: Power Iteration Clustering

Gonzalo Recio (gonzalo.recio@fib.upc.edu) — June 2020

Abstract: *Clustering is a widely used method for knowledge and pattern discovery in datasets. Here we present and analyze a spectral clustering algorithm called Power Iteration Clustering which is a simple, fast and effective approach for finding data clusters based on spectral properties of affinity matrices.*

1 Introduction

This coursework presents the implementation and analysis of the algorithm presented in the paper of Lin and Cohen (2010) called Power Iteration Clustering [1], which belongs to the category of spectral clustering-based algorithms.

Spectral clustering is an effective and elegant clustering method based on the pairwise similarity between data samples. It is based on spectral graph theory, which defines properties that hold the eigenvalues and eigenvectors of the adjacency matrix or Laplacian matrix of a graph. In datasets and machine learning, the similarity or affinity matrix of data instances can be used to take advantage of spectral graph properties for clustering purposes. One common approach of spectral clustering is to find a low-dimensional subspace from the Laplacian matrix of an affinity matrix, following these steps:

1. Compute the Laplacian matrix from the similarity matrix
2. Compute the first k eigenvalues of the Laplaceian matrix
3. Use the eigenvectors as new datapoints
4. Apply k-means as clustering algorithm on the eigenvectors

In fact, this method consists in embedding the dataset in a space of lower dimensionality.

However, Lin and Cohen present a fast and simple spectral clustering method to compute comparable or better clusters to previous approaches: Power Iteration Clustering. The main contributions and advantadges of this altorithm are the simplicity and scalability. In particular, the used subspace is a one-dimensional embedding generated using the power iteration (PI) method with *early stopping* on a normalized affinity matrix (not needing to run the method until convergence). Hence, this technique is simple, scalable, easily to parallelize and well-suited for large datasets.

2 Power Iteration Clustering (PIC)

Given a dataset $X = \{x_1, \dots x_n\}$ and a similarity function $s(x_i, x_j)$ that follows the properties defined in previous work [2] (symmetry, non-negativity, etc.), we can define the *affinity matrix* $A \in \mathcal{R}^{n \times n}$ as $A_{ij} = s(x_i, x_j)$. The *degree matrix* D associated to A is a diagonal matrix with $d_{ii} = \sum_j A_{ij}$ (containing the sum of the rows in the diagonal). Now we can directly obtain the *normalized affinity matrix* W with $D^{-1}A$.

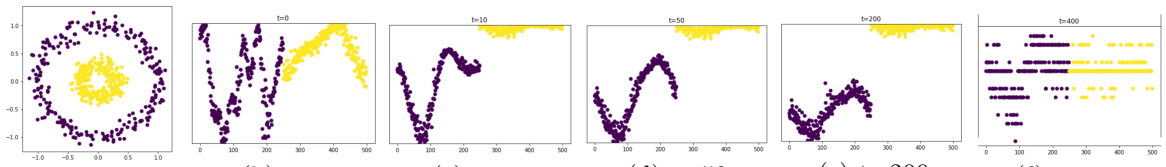
W is closely related to the *normalized Laplacian matrix* L, since $L = I - D^{-1}A = I - W$. L has several useful properties: most important to this work, the second-smallest eigenvector of L is defines a partition of the graph represented by W, that approximately maximizes the Normalized Cut criteria [3]. More generally, the k smallest eigenvectors of L are often well suited for clustering the graph W into k components.

However, the main point of the algorithm presented in this work lies in the idea of working with matrix W instead of the Laplacian matrix L. Note that the k smallest eigenvectors of L are also the k largest eigenvectors of W (since $L = I - W$). One simple method for computing the largest eigenvector of a matrix is the *power iteration* (PI). PI is an iterative method that starting with an arbitrary vector $\mathbf{v}^0 \neq \mathbf{0}$ and repeatedly performs

$$\mathbf{v}^t = cW\mathbf{v}^t$$

where c is a normalization factor to keep $\mathbf{v}^t$ from getting too large, typically $c = 1/\left\|W\mathbf{v}^t\right\|_1$ or $1/\left\|W\mathbf{v}^t\right\|_2$. Unfortunately, while the k smallest eigenvectors of L (hence, the largest in W) are in general interesting, the very smallest eigenvector of L is not (in fact it is a constant vector). However, what the authors noticed is that the intermediate solutions of the PI are in fact really useful, and seemed to represent a good embedding space of the datapoints.

This behaviour can be observed in figures 1(b)-(f), which illustrate the eigenvector embedding at different iterations of the PI. Figure 1(e) shows that PI converges locally to a good embedding by iteration $t = 200$, however, if we let the PI algorithm run for longer, all datapoints collapse to a not interesting embedding, as shown in figure 1(f). For visualization purposes, in figures 1(b)-(f) instances are ordered by class along the x-axis; the main interest is to notice the class separability in the y-axis of the embedding (x-axis only represents the instance number). For this execution, we generated a simple dataset of points in the $\mathcal{R}^2$ space and The affinity matrix has been defined as $A_{ij} = exp(\frac{-||x_i - x_j||^2}{2\sigma^2})$, also knows as Gaussian or radial basis function.



(a) PIC result (b) t=0 (c) t=10 (d) t=50 (e) t=200 (f) t=400

Figure 1: Results of the embedding computed by PIC along iterations. Plots (b)-(e) are re-scaled in the y-axis in order to display the embedded points *zoomed*.

To understand this behavior, we can see PI as an iterative averaging method. This means that the element i is set to a weighted average of its neighbors in the underlying graph weighted according to W_{ij}. If the underlying graph is connected, repeated averaging will eventually make all nodes have the same value; however, sets of nodes that are near each other will quickly attain a similar value, while nodes that are far away in the graph will converge in value slowly.

2.1 Power Iteration Convergence

The central observation of this paper is that running PI until convergence does not lead to interesting results. Instead, they are interested in intermideate vectors obtained by PI which represent a good embeddings for separating data easier, as seen in figure 1.

In the paper, Lin and Cohen perform an accurately profound and numerically-based study of how the convergence of the PI behaves and the evolution of the eigenvectors values the along iterations.

Their observations suggest that an effective clustering algorithm might run PI for some number of iterations t, stopping when $\mathbf{v}^t$ becomes a useful linear combination of the first k eigenvectors, followed by clustering the eigenvalues of $\mathbf{v}^t$. The main concern is to be able to detect and stop PI *after* it has converged within clusters but *before* the entire dataset converges, we will have an approximately piece-wise constant vector, where the elements that are in the same cluster having similar values.

From their analytical study, they define the *velocity* at iteration t to be the vector $\delta^t = \mathbf{v}^t - \mathbf{v}^{t-1}$ and define the *acceleration* at t as the vector $\epsilon = \delta^t - \delta^{t-1}$. Given that, the authors claim that the PI should be stopped when acceleration reaches a small threshold $\hat{\epsilon}$, for instance $\hat{\epsilon} = \frac{10^{-5}}{n}$, where n is the number of data samples.

The complete Power Iteration Clustering algorithm is described in algorithm 1.

Algorithm 1 -Input: A row-normalized affinity matrix W and the number of clusters k

$\mathbf{v}^0(i) \leftarrow \sum_j A_{ij}$
$\delta^0 \leftarrow \|\mathbf{v}^0\|_2$
$\epsilon = 10^{-5}$
while $\|\delta^{t+1} - \delta^t\| > \epsilon$ **do**
  $\mathbf{v}^{t+1} \leftarrow \frac{W\mathbf{v}^t}{\|W\mathbf{v}^t\|_1}$
  $\delta^{t+1} \leftarrow |\mathbf{v}^{t+1} - \mathbf{v}^t|$
  $t \leftarrow t + 1$
end while
Use k-means to cluster points on $\mathbf{v}^t$
return k-means clusters $C_1, C_2, \ldots, C_k$

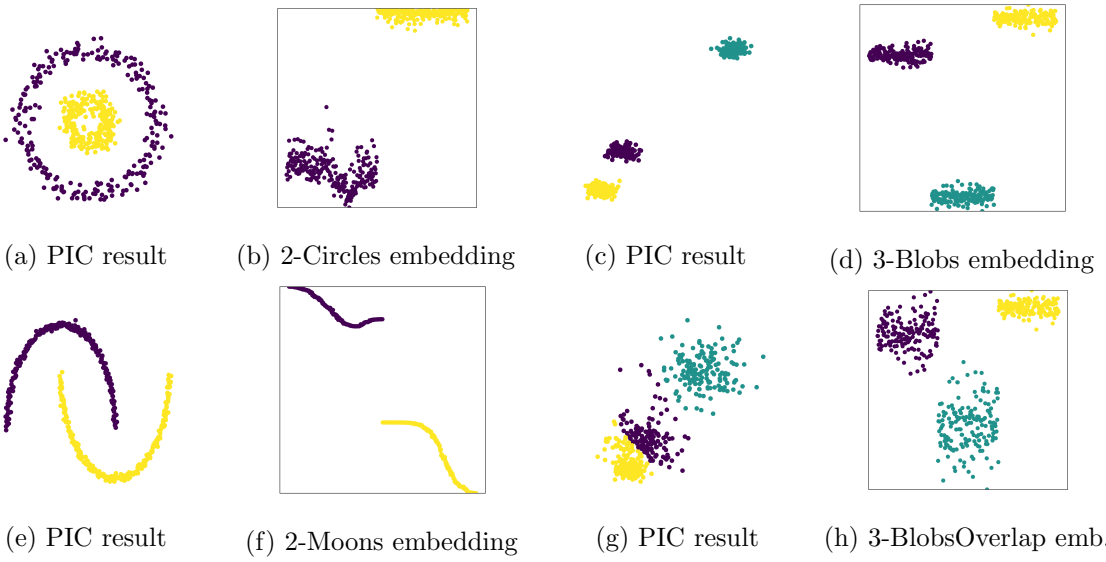
3 Experiments

We are going to compare the algorithm with the well-known clustering method k-means and with another spectral algorithm called Normalized Cuts [4][5]. Both algorithms are implemented in the `sklearn` python library [6]. Actually, the implementation of k-Means in `skleanr` is the improved version called k-Means++ [7]. For the Power Iteration Clustering algorithm, it had to be implemented by our own, which can be found in this github repository.

3.1 Synthetic Datasets

In this section we experiment the with some 2-dimensional synthetic datasets (with points in the $\mathcal{R}^2$ space) in order to see how PIC behaves on simple scenarios and how the embedding is generated. We are going to test with linearly/non-linearly separable classes and more/less overlapping clusters. In figure 2, we can see the clustering performed by PIC and the embedding computed. For all the datasets, affinity matrix A is defined by $A_{ij} = exp(\frac{-||x_i - x_j||^2}{2\sigma^2})$, except for 2-Moons, in which matrix A is defined by as a connectivity matrix of k-nearest neighbors ($k = 10$) that results in much better results.

We can observe that the more variance and noise in the data clusters, the less clear are the embeddings computed by PIC, as we can see in the thinner embeddings from 3-Blobs (2(d)) vs. the sparser embeddings from 3-BlobsOvelap (2(h)). In addition, we noticed that the algorithm performance is very dependent on the similarity metric selected for the affinity matrix, and some metrics generate very poor quality embeddings.



(a) PIC result (b) 2-Circles embedding (c) PIC result (d) 3-Blobs embedding

(e) PIC result (f) 2-Moons embedding (g) PIC result (h) 3-BlobsOverlap emb.

Figure 2: Power Iteration Clustering results and embeddings on synthetic datasets.

3.2 Real Datasets

To demonstrate the effectiveness of PIC, we are going to test it on real classification datasets.

- **Iris** dataset is a classic and very easy multi-class classification dataset where features are flower petal and sepal measurements from 3 species of irises. Contains 150 instances, 50 per class.
- **Wine** dataset contains the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines. Consists of 178 instances.
- **Breast**, or Breast Cancer Wisconsin (Diagnostic) Dataset contains features computed from a digitized image of a fine needle aspirate (FNA) of a breast mass which describe characteristics of the cell nuclei present in the image. Consists of 569 and 32 features each.
- **Digits** dataset has 8x8 gray image of a digit as datapoints. Since we are only taking 5 classes from this dataset (from 0 to 4), we are going to call it **Digits(0-4)**, which has 901 instances with 64 features each (8x8).
- **Splice** is a categorical dataset where instances are points on a DNA sequence at which "superfluous" DNA is removed during the process of protein creation in higher organisms. The problem posed in this dataset is to recognize, given a sequence of DNA, the boundaries between exons (3 types). Consists of 3190 with 61 attributes each.

To evaluate the quality of the algorithms, we are going to use cluster metrics: purity score, normalized mutual information (NMI) and adjusted random index (ARI). Furthermore, we are going to keep track of the execution times of the algorithms (affinity matrix computation is considered to be an input of the clustering algorithms, hence it is not taken into account for the execution times).

- Purity: it is a measure of the extent to which clusters contain a single class. It can be seen as the accuracy of a given assignment.

- Normalized Mutual Information (NMI): NMI is a information-theory measure which measures the mutual information of the true labeling and the cluster and normalizes it by their respective entropies.
- Adjusted Rand Index (ARI): compares the labeling assignment and the ground truth for every possible pair of instances. It is the number of correctly clustered pairs divided by the total number of possible pairs, with chance correction. Intuitively, high ARI values tells that the assignment is far from coming from a random labeling.

In table 1, we can see the cluster results from each algorithm. Note that, regarding the input affinity matrix A for PIC and N-Cut algorithms, we have chosen the cosine similarity for Iris and Breast datasets; nearest neighbors connectivity for Digits(0-4) and Splice datasets; and similarity based on Canberra distance for Wine dataset.

Furthermore, we can observe that we managed to replicate the authors' implementation by achieving same scores on Purity and NMI on Iris dataset and, hence, this somehow validates the correctness of our custom PIC implementation.

			PIC				N-Cuts				K-Means			
Dataset	k	Affinity metric	Purity	NMI	ARI	Exec. time	Purity	NMI	ARI	Exec. time	Purity	NMI	ARI	Exec. time
Iris	3	cosine	**0.9800**	**0.9306**	**0.9410**	**0.0349s**	0.7800	0.5898	0.5388	0.0383s	0.8933	0.7582	0.7302	0.0388s
Wine	3	Canberra	**0.8652**	**0.6647**	**0.6346**	0.0478s	0.7135	0.4199	0.3591	0.0411s	0.7022	0.4288	0.3711	**0.0342s**
Breast	2	cosine	0.8787	0.4902	0.5674	**0.0475s**	**0.8822**	**0.4993**	**0.5784**	0.0679s	0.8541	0.4648	0.4914	0.0566s
Digits(0-4)	5	k-NN	**0.9501**	**0.8924**	**0.8852**	0.45s	0.8280	0.8877	0.7756	0.2667s	0.8912	0.7708	0.7592	**0.1471s**
Splice	3	k-NN	0.7216	0.3744	0.3418	**0.3783s**	0.6382	0.2581	0.2151	1.6327s	**0.8690**	**0.5837**	**0.6382**	1.2289s

Table 1: Clustering performance of PIC, N-Cut and K-Means on real datasets. For clustering metrics, the higher the better; for execution time, the lower the better. Bold results represent the best obtained in each row.

The results from table 1 show that the cluster quality obtained with PIC is in general quantitatively comparable to the ones obtained with N-Cut and k-Means, and in some cases even better. On the other hand, we can observe that the performance in time is also great in most of the cases. In particular, on Digits(0-4), PIC execution took more time than the other algorithms because it did not manage to converge and was stopped by max. iteration threshold. Even though, the cluster quality is still good and indicates that limiting the algorithm iterations does not lead to poor results.

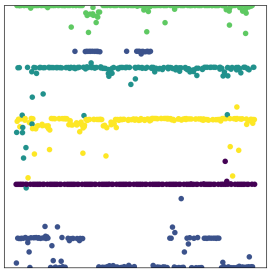


Figure 3: Good embedding generated with PIC from Digits(0-4) dataset.

Regarding the embeddings, figure 3 illustrates the embedding generated from Digits(0-4) dataset, which it is, in particular, a really good one where classes are linearly separable. We notice that in this embedding, datapoints belonging to the same class converged very close into same values, showing a clear separability between the 5 different classes.

3.3 Performance

Now, we are are interested in testing and evaluating the performance of the algorithms when dealing with larger datasets. To do so, we are going to take a synthetic dataset where we know PIC performs well (e.g. 2-Circles), run them on the two spectral clustering algorithms (PIC and N-Cuts), and analyze the impact in time performance when the number of samples is increased.

Table 2 shows the runtime of PIC and N-Cuts for different sizes of the 2-Circles dataset. We can clearly see that PIC generally requires much less time than N-Cuts, which can be understood because PI

method is usually stopped before reaching eigenvector convergence. This difference in execution time is also illustrated in figure 4, which makes more evident the time difference between the algorithms. In addition, the number of PIC iterations until convergence also seem to increase along the size of the datasets (although it is not clear, there is a 0.83 of correlation between the dataset sizes and the num. of iterations).

	PIC		N-Cut
Dataset size	iter.	time (s)	time (s)
1000	100	0.068	0.165
2500	79	0.54	1.21
5000	188	4.72	6.91
7500	211	11.74	18.05
10000	190	13.40	42.82
15000	224	48.11	123.16

Table 2: Execution times and number of iterations until convergence of PIC.

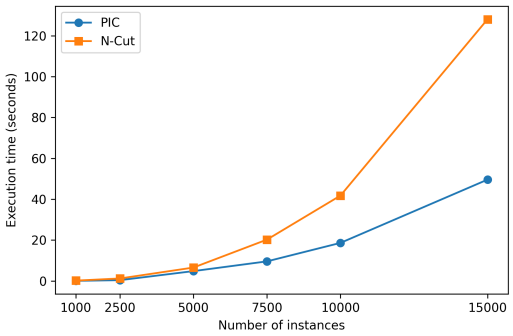


Figure 4: Execution times with respect to dataset size of PIC and N-Cut algorithms.

4 Conclusions

The main concern of this method is the selection of the similarity metric. The quality of the embedding and its datapoints separability strongly depends on the affinity metric and can perform perfectly in some cases while very poorly on others, as shown in figure 5. There is no general rule to determine which affinity metric is more adequate other than exploring the data distribution and trying which gives better performance.

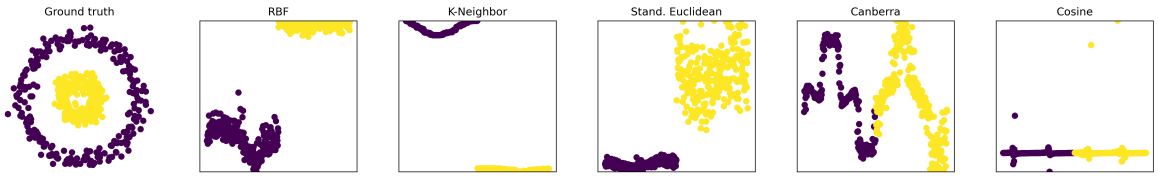


Figure 5: Embeddings obtained with different affinity metrics.

However, although this might seem a drawback for spectral clustering algorithms, being able to choose the similarity metric can be beneficial in contrast to having a fixed affinity metric. Having freedom to choose an appropriate similarity metric can help us to achieve better data embeddings. For instance, in non-linearly separable datasets, k-Means is limited to euclidean distances, while spectral clustering algorithms (e.g. PIC, N-Cuts) can manage to find linearly separable embeddings if a suitable affinity metric is applyied, as we can see in figure 6.

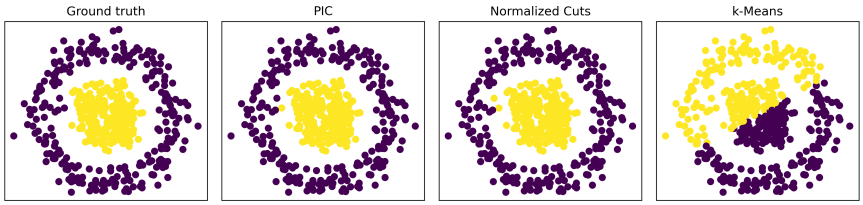


Figure 6: Clustering on non-linearly separable data.

Regarding the cluster quality, we can observe from table 1 that clusters obtained with PIC are comparable (and in some cases even bett) to clusters obtained from implemented clustering algorithms from `sklearn`, specially working well on non-linearly separable datasets outperforming the well-known k-Means algorithm. This good performance has been observed not only in synthetic datasets but also in real one.

Moreover, we have evaluated that the time performance of PIC algorithm is really good considering that we compared our custom implementation with efficient algorithms from `sklearn` package. This is explained by the property of PIC that does not need to execute PI until eigenvector convergence to find good embeddings. Also, PIC is an space-efficient algorithm since it only requires storing the W matrix and the iteratively updated $\mathbf{v}^t$ vector needed for the power iteration method. These properties make PIC algorithm more suitable for large datasets due to its fast performance.

References

[1] F. Lin and W. Cohen, "Power iteration clustering," pp. 655–662, 08 2010.

[2] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Advances in neural information processing systems*, pp. 849–856, 2002.

[3] M. Meila and J. Shi, "A random walks view of spectral segmentation," 2001.

[4] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, p. 888–905, Aug. 2000.

[5] X. Y. Stella and J. Shi, "Multiclass spectral clustering," in *null*, p. 313, IEEE, 2003.

[6] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[7] D. Arthur and S. Vassilvitskii, "K-means++: the advantages of careful seeding," in *In Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms*, 2007.