电子科技大学
University of Electronic Science and Technology of China

Power-iteration-based Clustering

Zhen Wang

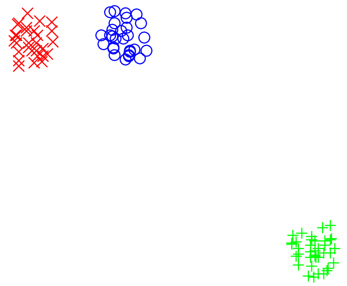
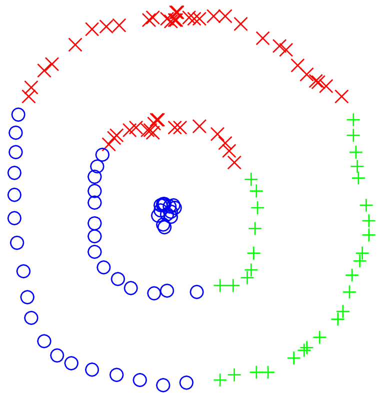
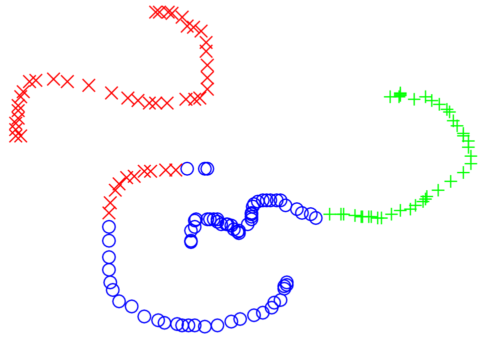
Data Mining Lab,
Big Data Research Center, UESTC
Email: zhen.wang@std.uestc.edu.cn

- ➢Preview
- ➢Motivation
- ➢Power Iteration Clustering (PIC)
  - Power Iteration
  - Stopping
- ➢Related Work
- Power method in text clustering
- Fuse, DPIC
- ➢Conclusion

K-means

数据挖掘实验室
Data Mining Lab

- A well-known clustering method
- 3-cluster examples:

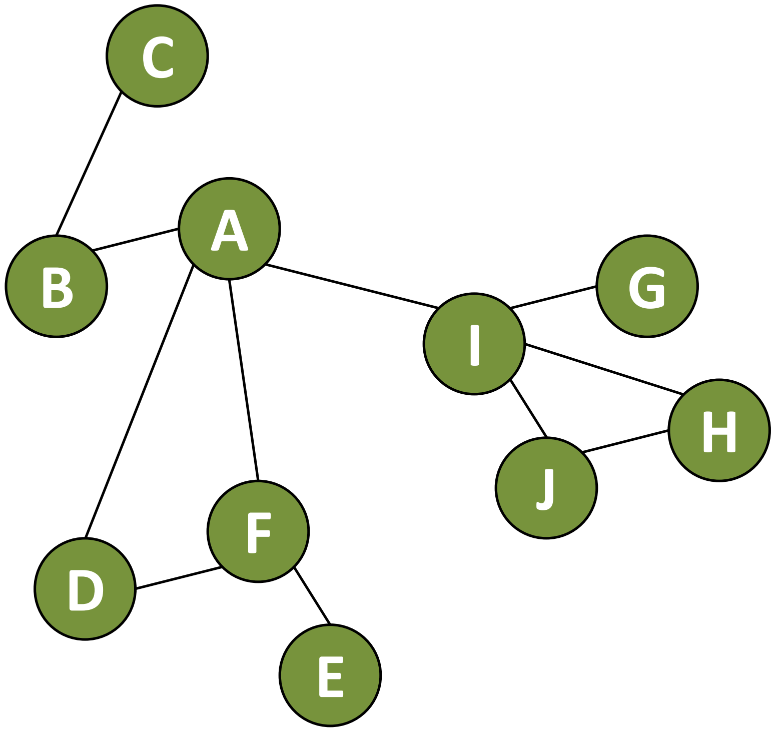




☺ ☹ ☹

Spectral Clustering

- Network = Graph = Matrix

	A	B	C	D	E	F	G	H	I	J
A		1		1		1				
B	1		1							
C		1								
D	1					1				
E						1				
F	1			1	1					
G									1	
H									1	1
I							1	1		1
J								1	1	



Spectral Clustering

- Normalized Cut algorithm (Shi &

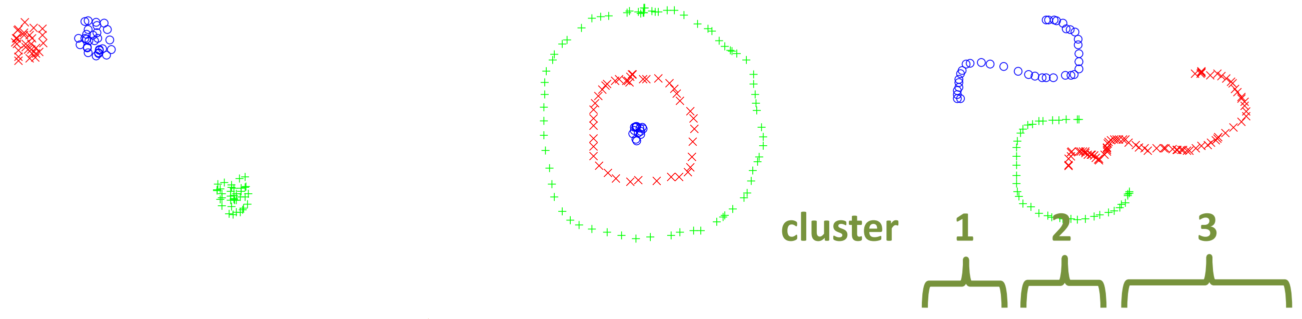
1. Choose k and similarity function s
2. Derive A from s, let W=I-$D^{-1}A$ matrix and D is a diagona quare matrix $D_{ii}=\Sigma_j A_{ij}$
3. Find eigenvectors and corresponding eigenvalues of W
4. Pick the k eigenvectors of W with the 2nd to kth smallest corresponding eigenvalues as “significant” eigenvectors
5. Project the data points onto the space spanned by these vectors
6. Run k-means on the projected data points

Finding eigenvectors and eigenvalues of a matrix is very slow in general: $O(n^3)$

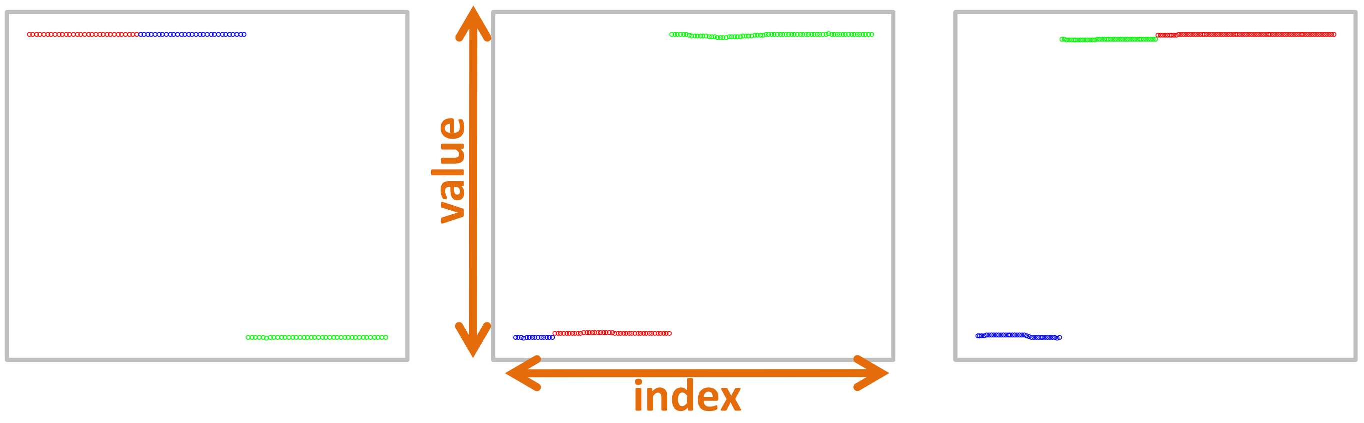
Spectral Clustering

数据挖掘实验室
Data Mining Lab

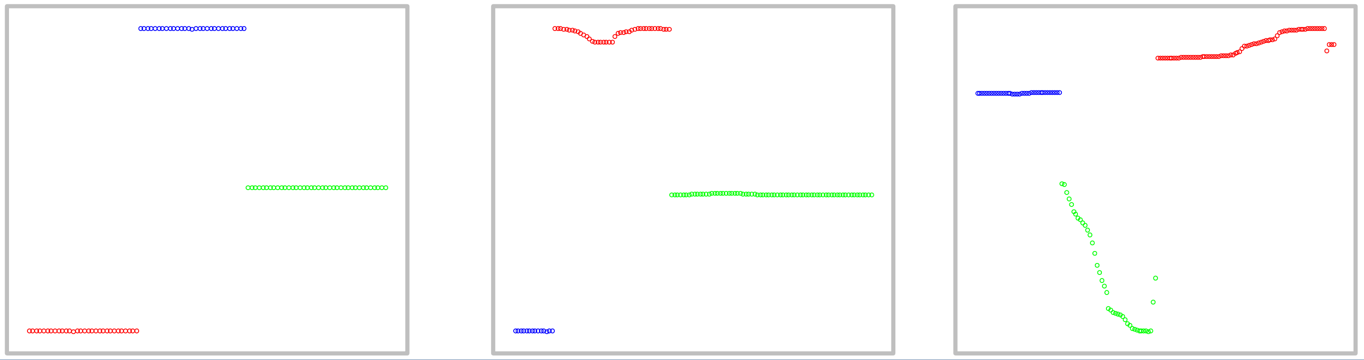
dataset and normalized cut results



2nd smallest eigenvector



3rd smallest eigenvector



Spectral Clustering

➢ Things to consider:

- Choosing a similarity function
- Choosing the number of clusters k?
- Which eigenvectors should be considered "significant"?
- The top or bottom k is not always the best for k clusters, especially on noisy data (Li et al. 2007, Xiang & Gong 2008)
- Finding eigenvectors and eigenvalues of a matrix is very slow in general: $O(n^3)$
- Construction and storage of, and operations on a dense similarity matrix could be expensive: $O(n^2)$

Hmm…

- Can we find a low-dimensional embedding for clustering, as spectral clustering, but without calculating these eigenvectors?

Power Iteration

- The power method, is a simple iterative method for finding the dominant eigenvector of a matrix:

$$\mathbf{v}^{t+1} = cW\mathbf{v}^{t}$$

- W – a square matrix $\boldsymbol{D}^{-1}\boldsymbol{A}$
- v^t – the vector at iteration t; v^0 is typically a random vector
- c – a normalizing constant to avoid v^t from getting too large or too small
- Typically converges quickly, and is fairly efficient if W is a sparse matrix

Power Iteration Clustering

- Recall the power iteration update:

$$\begin{aligned}\mathbf{v}^{t} &= W\mathbf{v}^{t-1}\\ &= W^{2}\mathbf{v}^{t-2}\\ &\dots\\ &= W^{t}\mathbf{v}^{0}\\ &= c_1 W^{t}\mathbf{e}_1 + c_2 W^{t}\mathbf{e}_2 + \dots + c_n W^{t}\mathbf{e}_n\end{aligned}$$

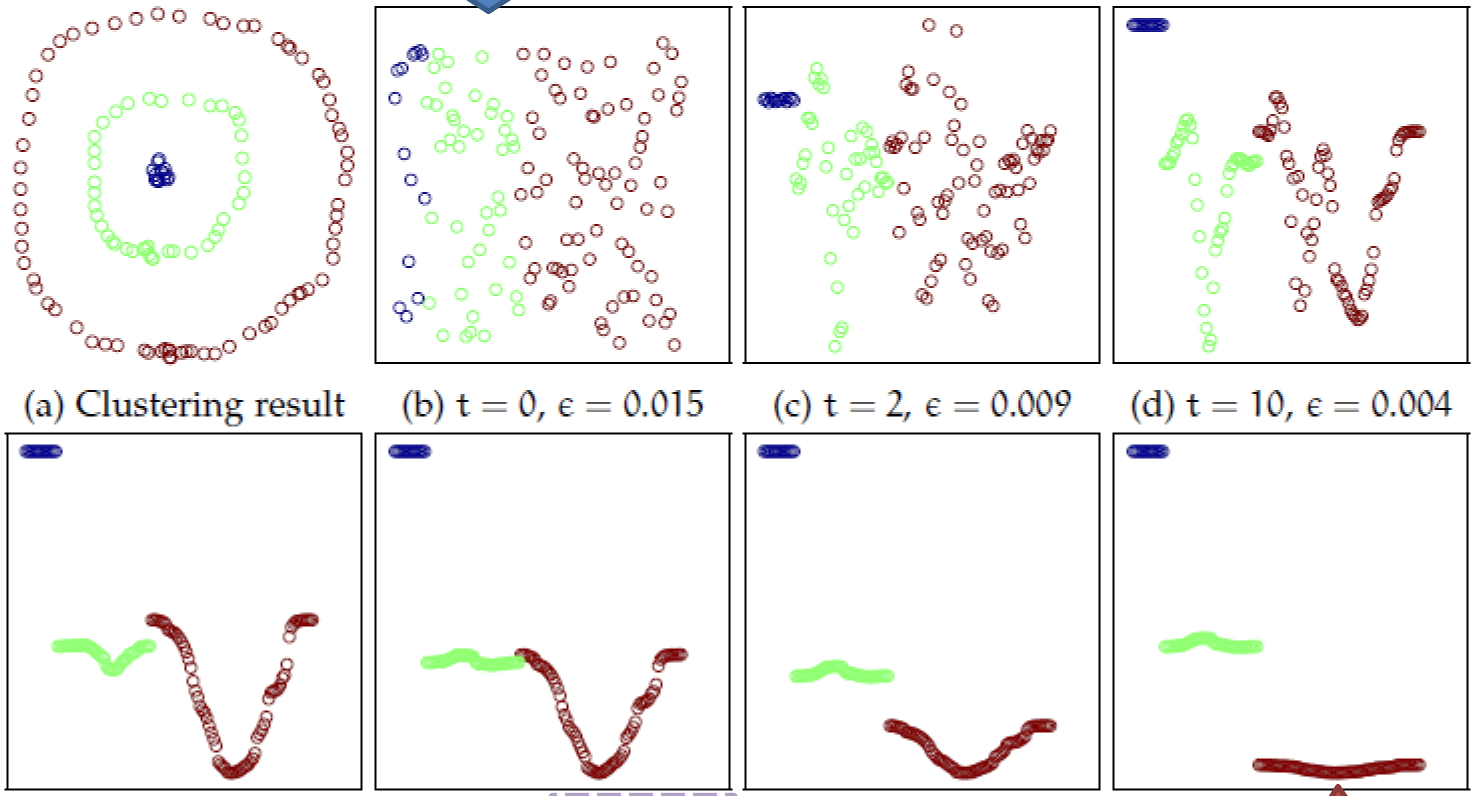
$${}_2 + \dots + c_n \lambda_n^{t}\mathbf{e}_n$$

c_i - the ith coefficient of v when projected onto the space spanned by the eigenvectors of W

$\mathbf{e}_i$ – the eigenvector corresponding to λ_i

λ_i - the ith largest eigenvalue of W

Begins with a random vector



(a) Clustering result (b) $t = 0$, $\epsilon = 0.015$ (c) $t = 2$, $\epsilon = 0.009$ (d) $t = 10$, $\epsilon = 0.004$

(e) $t = 100$, $\epsilon = 0.003$ (f) $t = 200$, $\epsilon = 0.002$ (g) $t = 500$, $\epsilon = 0.002$ (h) $t = 1000$ $= 0.001$

Overall absolute distance between points decreases, here we show relative distance

Ends with a piece-wise constant vector!

Power Iteration Clustering

- Group the $c_i\lambda_i e_i$ terms, and define $pic^t(a,b)$ to be the absolute difference between elements in the v^t, where a and b corresponds to indices a and b on v^t:

$$pic^t(a,b) =$$

$$\left| \left[\mathrm{e}_1(a) - \mathrm{e}_1(b) \right] c_1 \lambda_1^t + \sum_{i=2}^{k} \left[\mathrm{e}_i(a) - \mathrm{e}_i(b) \right] c_i \lambda_i^t + \sum_{j=k+1}^{n} \left[\mathrm{e}_j(a) - \mathrm{e}_j(b) \right] c_j \lambda_j^t \right|$$

Power Iteration Clustering

- Group the $c_i\lambda_i e_i$ terms, and define $pic^t(a,b)$ to be the absolute difference between elements in the v^t, where a and b corresponds to indices a and b on v^t:

$$pic^t(a,b) =$$

$$\left| \cancel{[\mathbf{e}_1(a) - \mathbf{e}_1(b)]c_1\lambda_1^t} + \sum_{i=2}^{k}[\mathbf{e}_i(a)-\mathbf{e}_i(b)]c_i\lambda_i^t + \cancel{\sum_{j=k+1}^{n}[\mathbf{e}_j(a)-\mathbf{e}_j(b)]c_j\lambda_j^t} \right|$$

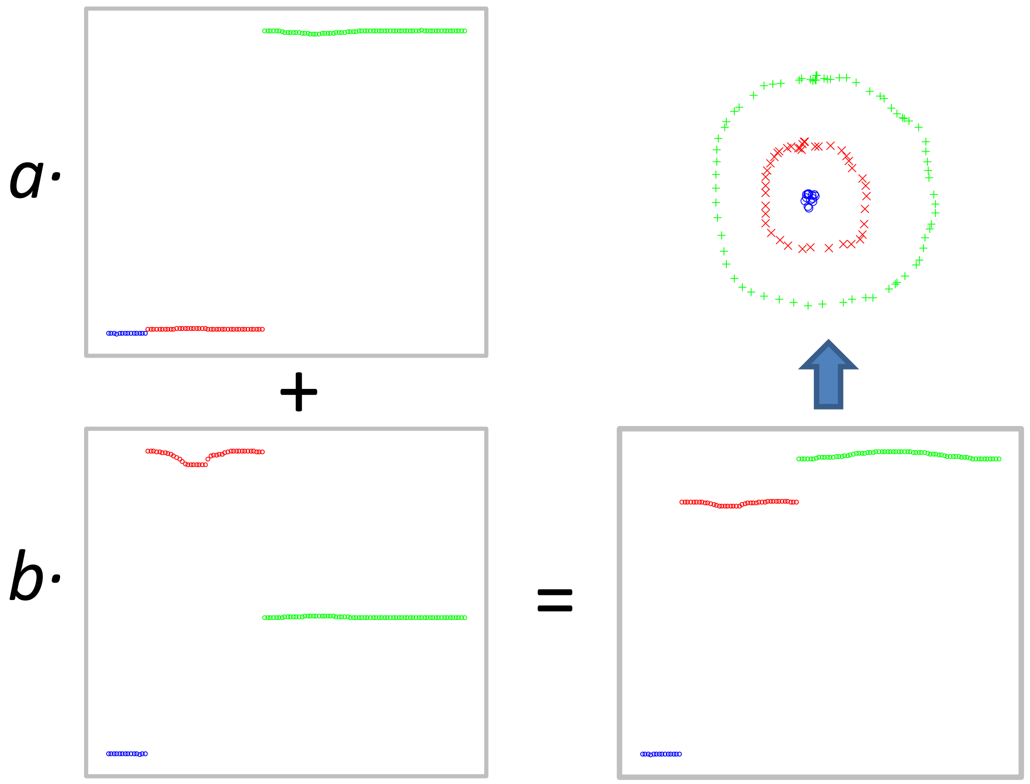
The first term is 0 because the first (dominant) eigenvector is a constant vector

We are left with the term that “signals” the cluster corresponding to eigenvectors!

As *t* gets bigger, the last term goes to 0 quickly

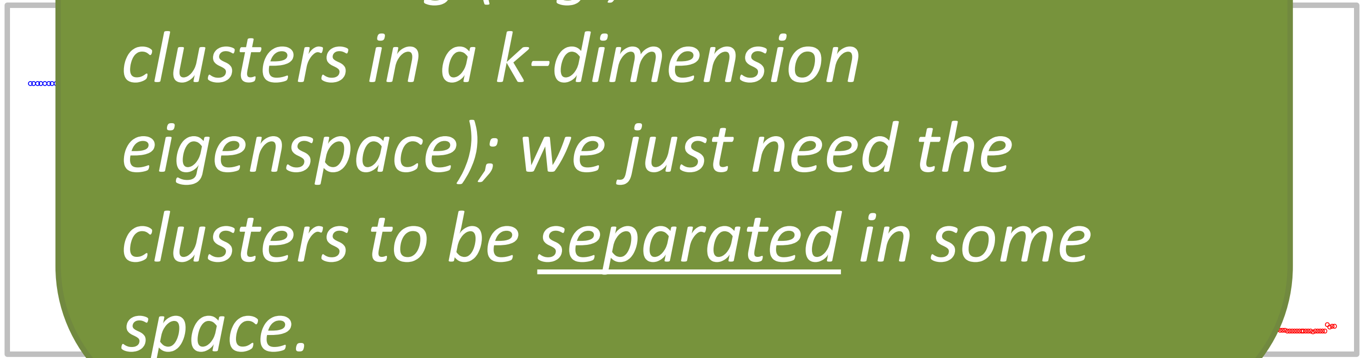
Power Iteration Clustering

- *The 2nd to kth eigenvectors of $W=D^{-1}A$ are roughly piece-wise constant with respect to the underlying clusters, each separating a cluster from the rest of the data* (Meila & Shi 2001)
- The linear combination of piece-wise constant vectors is also piece-wise constant!


a·
+
b·
=

dataset and PIC results

v^t



The Take-Away

To do clustering, we may not need all the information in a spectral embedding (e.g., distance between clusters in a k-dimension eigenspace); we just need the clusters to be separated in some space.

When to Stop

Recall:

$$\mathbf{v}^t = c_1\lambda_1^t\mathbf{e}_1 + \quad \mathbf{e}_{k+1} + \ldots + c_n\lambda_n^t\mathbf{e}_n$$

At the beginning, *v* changes fast ("accelerating") to converge locally due to "noise terms" (*k+1…n*) with small *λ*

Then:

$$\frac{\mathbf{v}^t}{c_1\lambda_1^t} = \mathbf{e}_1 + \ldots + \frac{c_k}{c_1}\left(\frac{\lambda_k}{\lambda_1}\right)^t\mathbf{e}_k + \frac{c_{k+1}}{c_1}\left(\frac{\lambda_{k+1}}{\lambda_1}\right)^t\mathbf{e}_{k+1} + \ldots + \frac{c_n}{c_1}\left(\frac{\lambda_n}{\lambda_1}\right)^t\mathbf{e}_n$$

When "noise terms" have gone to zero, v changes slowly ("constant speed") because only larger *λ* terms (*2…k*) are left, where the eigenvalue ratios are close to 1

Because they are raised to the power *t*, the eigenvalue ratios determines how fast *v* converges to e_1

When to Stop

Power iteration convergence depends on this term (could be very slow)

$\lambda_k^t \mathbf{e}_k + c_k$

PIC convergence depends on this term (always fast)

$\lambda_n^t \mathbf{e}_n$

Then:

$$\frac{\mathbf{v}^t}{c_1\lambda_1^t} = \mathbf{e}_1 + \frac{c_2}{c_1}\left(\boxed{\frac{\lambda_2}{\lambda_1}}\right)^t \mathbf{e}_k + \ldots + \frac{c_{k+1}}{c_1}\left(\boxed{\frac{\lambda_{k+1}}{\lambda_1}}\right)^t \mathbf{e}_{k+1} + \ldots + \frac{c_n}{c_1}\left(\frac{\lambda_n}{\lambda_1}\right)^t \mathbf{e}_n$$

➢So we can stop when the “acceleration” is nearly zero.

PIC as a General Method

数据挖掘实验室
Data Mining Lab

➢ A tio ori

Different ways to pick v^0 (random, node degree, exponential)

W can be swapped for other graph cut criteria or similarity function

Can be determined automatically at the end (e.g., G-means) since embedding does not require *k*

```
Input: A row-norm    ed affinity matrix $W$ and the number of clusters $k$
Output: Clusters $C_1$,      …, $C_k$

1.  Pick an initial vector $\mathbf{v}^0$
2.  Repeat
      •  Set $\mathbf{v}^{t+1} \leftarrow W\mathbf{v}^t$
      •  Set $\boldsymbol{\delta}^{t+1} \leftarrow |\mathbf{v}^{t+1} - \mathbf{v}^t|$
      •  Increment $t$
      •  Stop when $|\boldsymbol{\delta}^t - \boldsymbol{\delta}^{t-1}| \approx \mathbf{0}$
3.  Use k-means to cluster points on $\mathbf{v}^t$ and return clusters $C_1, C_2, …, C_k$
```

Better stopping condition? Suggested: entropy, mutual information, modularity, …

Use multiple v^t's from different v^0's for multi-dimensional embedding

Use other methods for final clustering (e.g., Gaussian mixture model)

Methods become fast and/or exact on a one-dimension embedding (e.g., k-means)!

Large Scale Considerations

- But…what if the dataset is large and the similarity matrix is dense? For example, a large document collection where each data point is a term vector?
- Constructing, storing, and operating on an NxN dense matrix is very inefficient in time and space.

Large Scale Considerations

数据挖掘实验室
Data Mining Lab

- Recall PIC's update is
  - $v^t = W * v^{t-1 =} = D^{-1}A * v^{t-1}$
  - where D is the [diagonal] degree matrix: $D=A*\mathbf{1}$
  - Let F(i,k)=TFIDF weight of word w_k in document v_i
  - Compute $N(i,i)=||v_i||$ … and $N(i,j)=0$ for $i!=j$
  - **Don't** compute $A = N^{-1}FF^TN^{-1}$
  - Let $D(i,i)= N^{-1}FF^TN^{-1}*\mathbf{1}$ where **1** is an all-1's vector
    - Computed as $D=N^{-1(}F^{\,(}F^T (N^{-1}*\mathbf{1})))$ for efficiency
  - New update:
    - $v^t = D^{-1}A * v^{t-1} = D^{-1} N^{-1}FF^TN^{-1} *v^{t-1}$

Path Folding

➤Example – cosine similarity:

Construction: O(n)

Storage: O(n)

Operation: O(n)

- Iteration update:

$$\mathbf{v}^{t+1} = D^{-1}(N(F(F^T(N\mathbf{v}^t))))$$

Random Walk with Restart

- The walk distribution **r** satisfies a simple equation:

Start node(s)

Transition matrix of the network

Equivalent to the well-known PageRank ranking if all nodes are start nodes! (**u** is uniform)

$$\mathbf{r} = (1-d)\mathbf{u} + dW\mathbf{r}$$

Restart probability

“Keep-going” probability (damping factor)

$$\boldsymbol{v} = (1-d)r + dP\boldsymbol{v}$$
$$v - dP\boldsymbol{v} = (1-d)r$$
$$(I - dP)\boldsymbol{v} = (1-d)r$$

But the matrix inversion, however, is computationally infeasible if n is large

However, we can approximate v iteratively with the power method:

$$v^{t+1} = (1-d)r + dPv^{t}$$

Here, we show v^t converges to v.

$$v^{t} = (1-d)r + dPv^{t-1}$$
$$v^{t} = (1-d)r + dP\big((1-d)r + dPv^{t-2}\big)$$
$$v^{t} = (1-d)r + dP((1-d)r + dP\big((1-d)r + dPv^{t-3}\big))$$
…
$$v^{t} = (1-d)\sum_{i=0}^{t-1}(dP)^{i}r + (dP)^{t-1}v^{0}$$

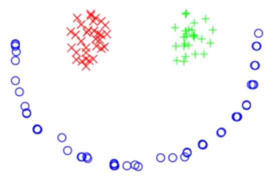
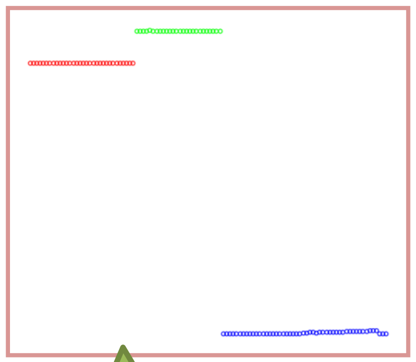
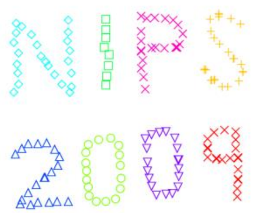
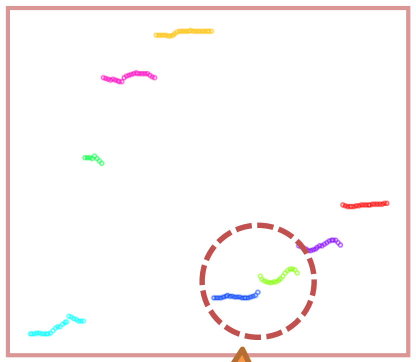
Given $\lim_{t\to\infty}(dP)^{t-1}v^{0} = 0$, $\lim_{t\to\infty}\sum_{i=0}^{t-1}(dP)^{i} = (I-dP)^{-1}$

$$\lim_{t\to\infty} v^{t} = (1-d)(I-dP)^{-1}r$$
$$= v$$

PIC-k

数据挖掘实验室
Data Mining Lab

- One robustness question for vanilla PIC as data size and complexity grow:
- How many (noisy) clusters can you fit in one dimension without them “colliding”?

Cluster signals cleanly separated

A little too close for comfort?

Multi-Dimensional PIC

➤ Solution:

- Run PIC *d* times with different random starts and construct a *d*-dimension embedding
- Unlikely any pair of clusters collide on all *d* dimensions

Fuse

Though the low-dimensional embedding we find is a linear combination of the dominant vectors of data,
But it also contains noise, which is of no use to the clustering……

And this is exactly the motivation of FUSE (FUll Spectral ClustEring)

Problem: Statistically Independent Pseudo-eigenvectors.

Given a pseudo-eigenvector matrix V $\in R^{p\times n}$ generated by running PI p times, find a demixing matrix M $\in R^{p\times p}$ such that E = MV and the sum of mutual information between pairwise components of E is minimized, where E $\in R^{p\times n}$ is a resulting independent pseudo-eigenvector matrix.

$$J_1(M) := \min \sum_{1\le i,j\le p, i\ne j} I(e_i\,; e_j)$$

$$\min I(e_i; e_j)$$
$$\text{subject to } E = MV, 1 \leq i, j \leq p, i \neq j$$

Now it comes to how to select k independent components. Since ICA is interested in searching for non-Gaussian directions, in which negentropy is minimized.

$$Kurtosis = \frac{\mu_4}{\sigma^4} = \frac{E((X-\mu)^4)}{(E((X-\mu)^2))^2}$$

DPIC (Deflation-PIC)

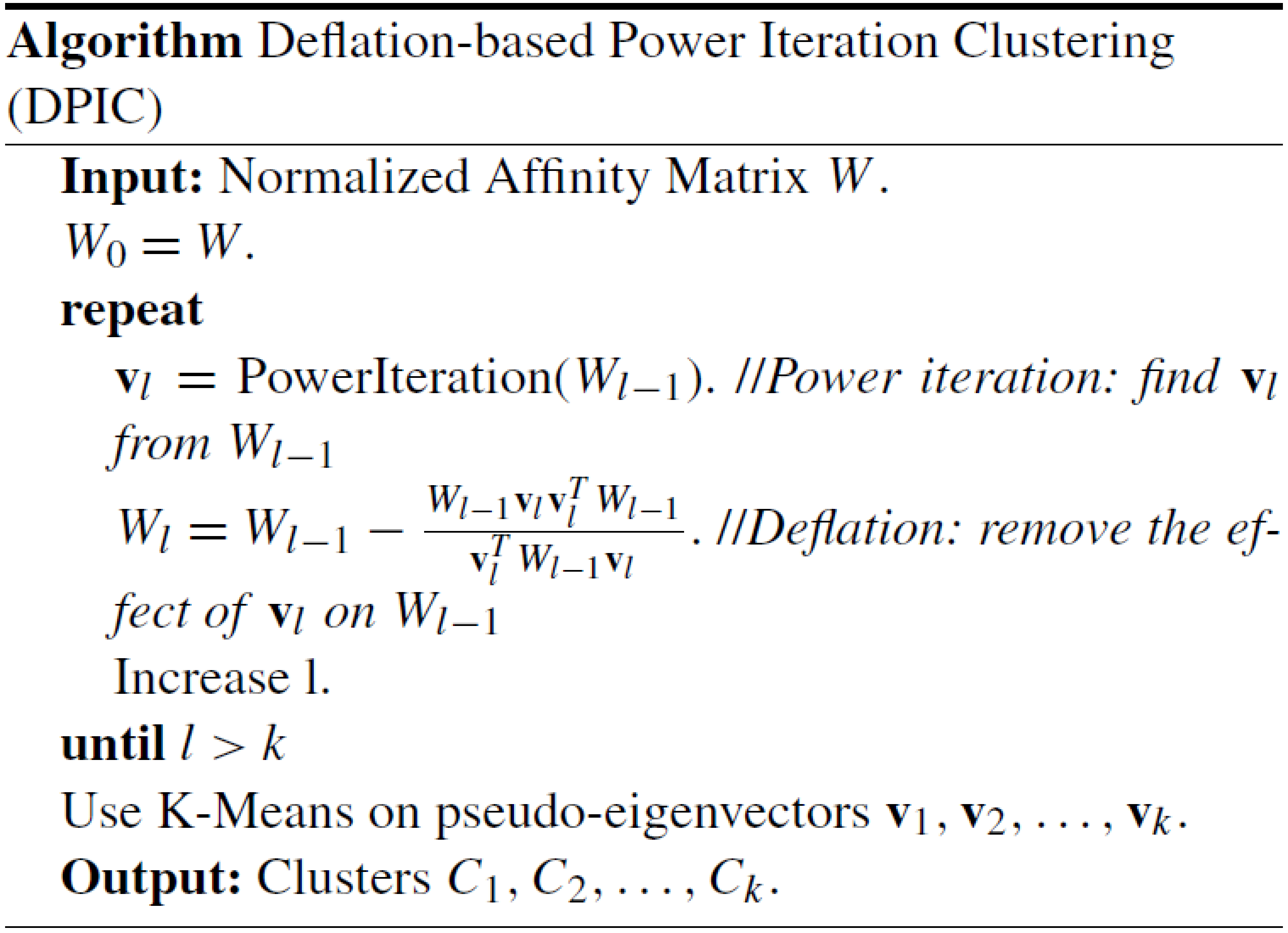
we propose a novel method based on the deflation technique to compute multiple orthogonal pseudo-eigenvectors (orthogonality is used to avoid redundancy)

DPIC (Deflation-PIC)

Assuming that we have a matrix A_{t-1} and its eigenvector v_t, the Schur complement deflation creates a new matrix A_t , which is computed by the following formula:

$$A_t = A_{t-1} - \frac{A_{t-1} v_t v_t^T A_{t-1}}{v_t^T A_{t-1} v_t}$$

DPIC (Deflation-PIC)



Algorithm Deflation-based Power Iteration Clustering (DPIC)

```
Input: Normalized Affinity Matrix $W$.
$W_0 = W$.
repeat
    $\mathbf{v}_l$ = PowerIteration($W_{l-1}$). //Power iteration: find $\mathbf{v}_l$ from $W_{l-1}$
    $W_l = W_{l-1} - \frac{W_{l-1}\mathbf{v}_l\mathbf{v}_l^T W_{l-1}}{\mathbf{v}_l^T W_{l-1}\mathbf{v}_l}$. //Deflation: remove the effect of $\mathbf{v}_l$ on $W_{l-1}$
    Increase l.
until $l > k$
Use K-Means on pseudo-eigenvectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k$.
Output: Clusters $C_1, C_2, \ldots, C_k$.
```

DPIC (Deflation-PIC)

数据挖掘实验室

Data Mining Lab

The pseudo-eigenvectors produced by our algorithm are mutually orthogonal.

From the Schur equation on the lth loop of algorithm, we obtain

$$W_l = W_{l-1} - \frac{W_{l-1} v_l v_l^T W_{l-1}}{v_l^T W_{l-1} v_l}$$

We multiply both sides with v_l,

$$W_l v_l = W_{l-1} v_l - \frac{W_{l-1} v_l v_l^T W_{l-1}}{v_l^T W_{l-1} v_l} = 0$$

DPIC (Deflation-PIC)

From the Schur equation on the $(l+1)th$ loop

$$W_{l+1} = W_l - \frac{W_l v_{l+1} v_{l+1}^T W_l}{v_{l+1}^T W_l v_{l+1}}$$

We multiply both sides with v_l,

$$W_{l+1} v_l = W_l v_l - \frac{W_l v_{l+1} v_{l+1}^T (W_l v_l)}{v_{l+1}^T W_l v_{l+1}}$$

$$= \ 0$$

In the same manner, we can prove that

$$\forall s \geq l+1, W_{s-1} v_l = 0$$

DPIC (Deflation-PIC)

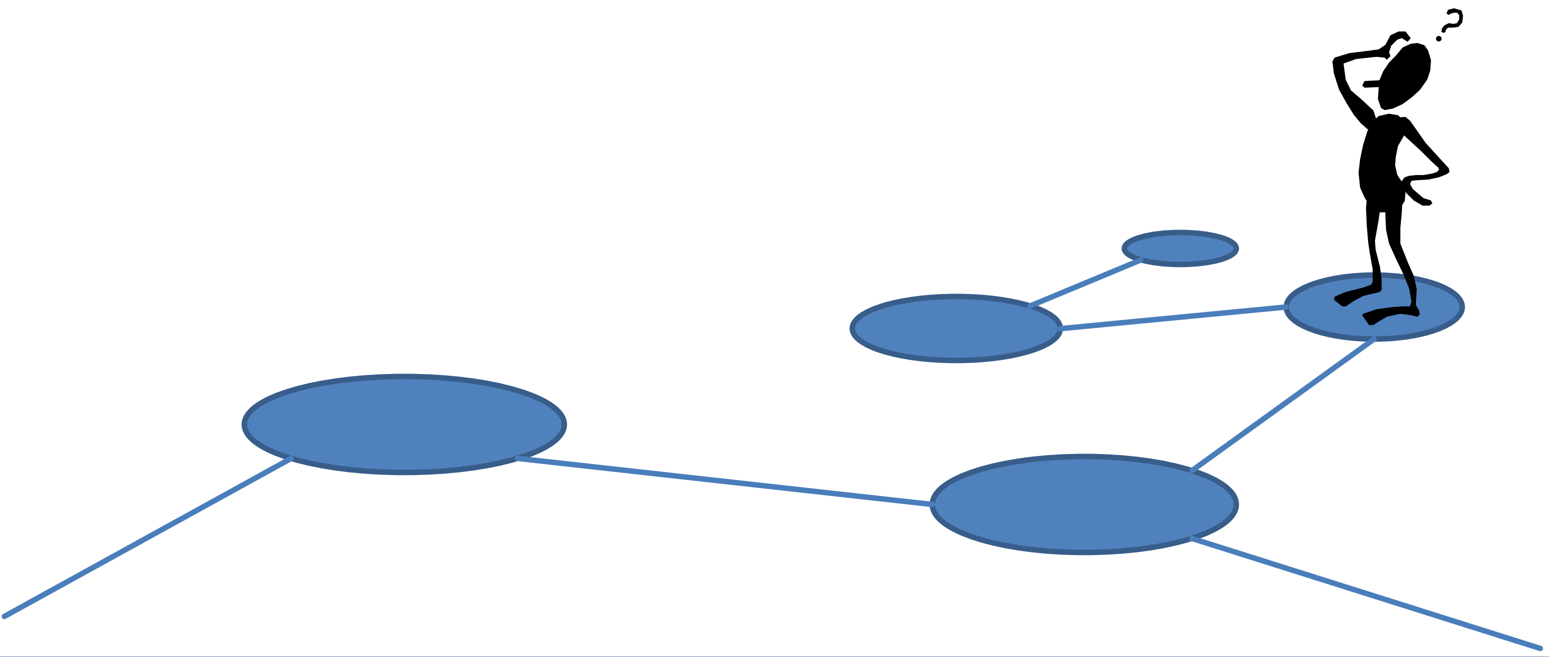
So we can obtain

$$v_l^T v_s = (v_l^T W_{s-1}^T)(W_{s-1})^{t-1} v_0 = 0$$

In sum, the DPIC's pseudo-eigenvectors are mutually orthogonal:

$$v_l^T v_s = 0$$

Questions?



Thanks

Reference:

1. Ramnath Balasubramanyan, Frank Lin, and William W Cohen. Node clustering in graphs: An empirical study. 2010.

2. Frank Lin and William W Cohen. Power iteration clustering. In Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel, pages 655–662, 2010.

3. Nguyen Duc Thang, Young-Koo Lee, Sungyoung Lee, et al. Deflation-based power iteration clustering. Applied intelligence, 39(2):367–385, 2013.

4. Wei Ye, Sebastian Goebl, Claudia Plant, and Christian Boehm. Fuse: Full spectral clustering. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 1985–1994. ACM, 2016.

5. Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. IEEE Transactions on pattern analysis and machine intelligence, 22(8):888–905, 2000.

6. Lin, Frank, and William W. Cohen. "A Very Fast Method for Clustering Big Text Datasets." *ECAI*. 2010.

7. Lin, Frank, and William W. Cohen. "Semi-supervised classification of network data using very few labels." *Advances in Social Networks Analysis and Mining (ASONAM), 2010 International Conference on*. IEEE, 2010.

I'd like to extend my warm thanks to Frank Lin for part of his slides!

Zhen Wang 2017.4.19 Chengdu zhen.wang@std.uestc.edu.cn