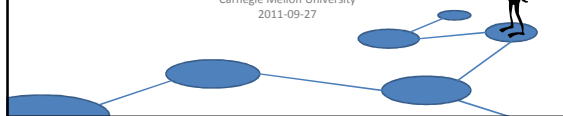


Power Iteration Clustering

Frank Lin
15-826 Multimedia Databases and Data Mining
School of Computer Science
Carnegie Mellon University
2011-09-27



Talk Outline

- ➡ • Clustering
- Spectral Clustering
- Power Iteration Clustering (PIC)
 - PIC with Path Folding
 - PIC Extensions

2

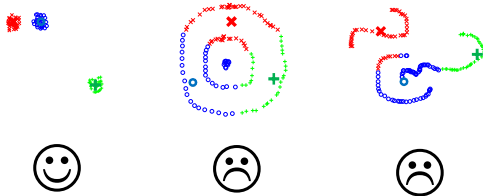
Clustering

- Automatic grouping of data points
- 3 example datasets:



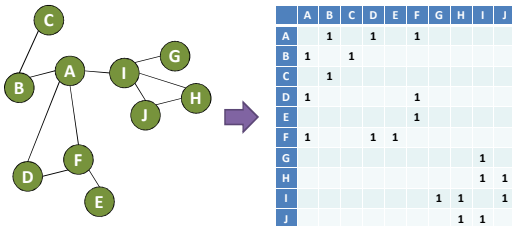
k-means

- A well-known clustering method
 - Given: Points in Euclidean space and an integer k
 - Find: k clusters determined by k centroids
 - Objective: Minimize within-cluster sum of square distances



Graph Clustering

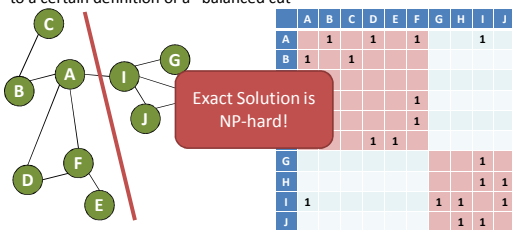
Given: Data = Network = Graph = Matrix



Graph Cluster

Find: Partitions of the graph

Objective: Minimizes (or maximizes) an objective function according to a certain definition of a "balanced cut"



Talk Outline

- Clustering
- ➔ • Spectral Clustering
- Power Iteration Clustering (PIC)
 - PIC with Path Folding
 - PIC Extensions

7

Spectral Clustering

- Does two things:
 1. Provides good polynomial-time approximation to the balanced graph cut problem
 2. Clustering according to similarity, not Euclidean space

Relax solution to take on real values, then compute via eigendecomposition

Recall that similarity can be represented as a graph/matrix

8

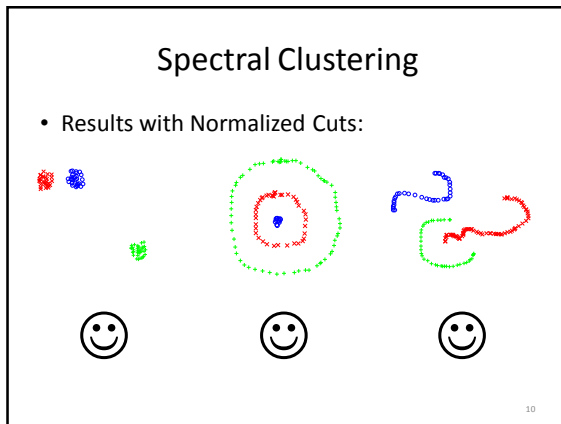
Spectral Clustering

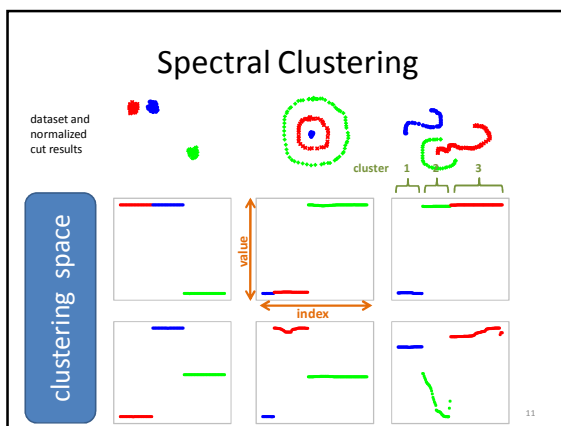
- How: Cluster data points in the space spanned by the “significant” eigenvectors (spectrum) of a [Laplacian] similarity matrix

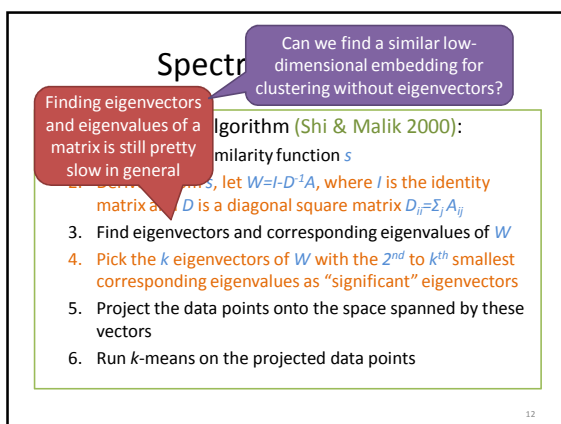
Not familiar with eigenvectors? Details on *spectral analysis*, *eigen-vectors/values*, *SVD* will all be covered extensively 6-7 lectures later! Stay tuned!

A popular spectral clustering method: normalized cuts (NCut)

9







Talk Outline

- Clustering
- Spectral Clustering
- ➔ • Power Iteration Clustering (PIC)
 - PIC with Path Folding
 - PIC Extensions

13

Power Iteration Clustering

- Spectral clustering methods are nice, and a natural choice for graph data
- But they are rather expensive and slow

Power iteration clustering (PIC) can provide a similar solution at a very low cost (fast)!

14

The Power Iteration

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

Typically converges quickly; fairly efficient if W is a sparse matrix

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

\mathbf{v}^t : the vector at iteration t ;

c : a normalizing constant to keep \mathbf{v}^t from getting too large or too small

W : a square matrix

\mathbf{v}^0 typically a random vector

15

The Power Iteration

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

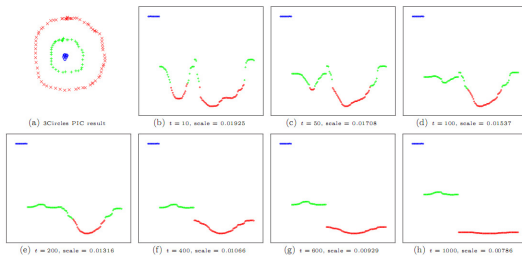
$$\mathbf{v}^{t+1} = cW\mathbf{v}^t$$

Row-normalized similarity matrix

What if we let $W=D^{-1}A$ (like Normalized Cut)?

16

The Power Iteration

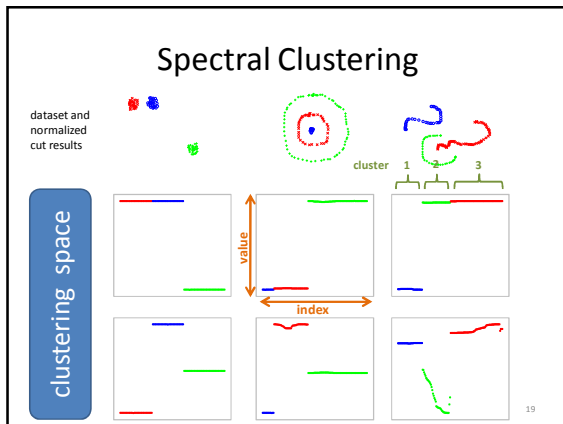


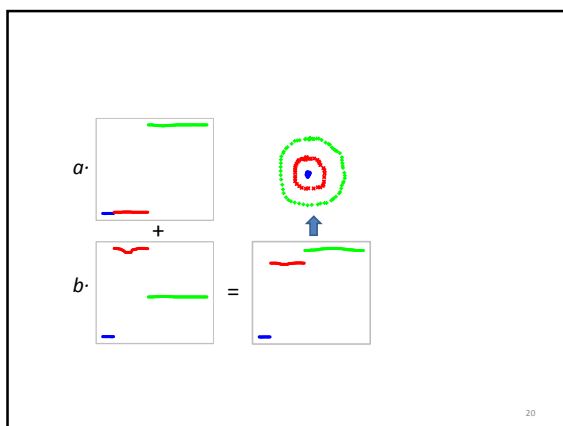
17

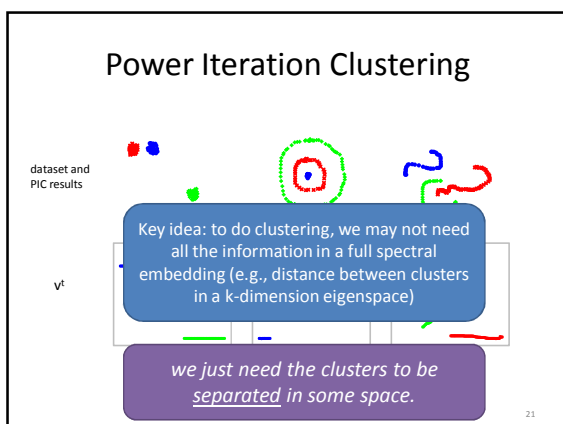
Power Iteration Clustering

- The 2^{nd} to k^{th} 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
- The linear combination of piece-wise constant vectors is also piece-wise constant!

18







Details

When to Stop

Recall:

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

Then:

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

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

When "noise terms" have gone to zero, v changes slowly ("constant speed") because only larger λ terms ($2 \dots k$) 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 \mathbf{e}_1

Power Iteration Clustering

- A basic power iteration clustering (PIC) algorithm:

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

Output: Clusters C_1, C_2, \dots, C_k

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

i.e., when acceleration is nearly zero

PIC Runtime

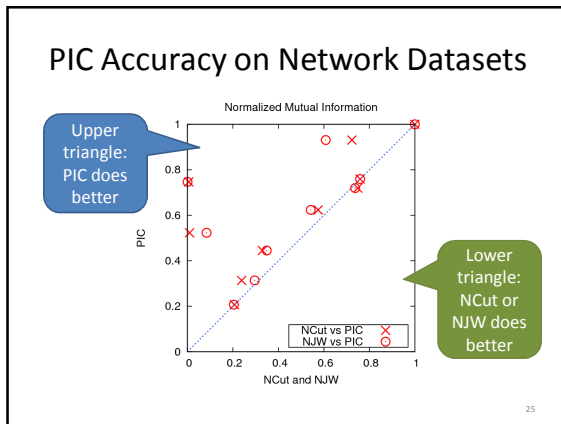
Normalized Cut

Normalized Cut, faster implementation

Table 4. Runtime comparison (in milliseconds) of PIC and spectral clustering algorithms on synthetic datasets.

Nodes	Edges	NCutE	NCutI	PIC
1k	10k	1,885	177	1
5k	250k	154,797	6,939	7
10k	1,000k	1,111,441	42,045	34
50k	25,000k	-	-	849
100k	100,000k	-	-	2,960

Ran out of memory (24GB)



Talk Outline

- Clustering
- Spectral Clustering
- Power Iteration Clustering (PIC)
 - ➔ – PIC with Path Folding
 - PIC Extensions

26

Clustering Text Data

- Spectral clustering methods are nice
- We want to use them for clustering text data

(A lot of)

27

The Problem with Text Data

- Documents are often represented as feature vectors of words:

The importance of a Web page is an inherently subjective matter, which depends on the readers...

In this paper, we present Google, a prototype of a large-scale search engine which makes heavy use...

You're not cool just because you have a lot of followers on twitter, get over yourself...

	cool	web	search	make	over	you
0	4	8	2	5	3	
0	8	7	4	3	2	
1	0	0	0	1	2	

The Problem with Text Data

- Feature vectors are often sparse
- But similarity matrix is not!

Mostly non-zero - any two documents are likely to have a word in common

Mostly zeros - any document contains only a small fraction of the vocabulary

	cool	web	search	make	over	you
0	4	8	2	5	3	
0	8	7	4	3	2	
1	0	0	0	1	2	

The Problem with Text

- A similarity matrix is the input to many methods, including *spectral clustering*
- Spectral clustering requires the computation of the eigenvectors of a similarity matrix

In general $O(n^3)$; approximation methods still not very fast

$O(n^2)$ time to construct

$O(n^2)$ space to store

$> O(n^2)$ time to operate on

Too expensive! Does not scale up to big datasets!

	cool	web	search	make	over	you
0	4	8	2	5	3	
0	8	7	4	3	2	
1	0	0	0	1	2	

The Problem with Text Data

- We want to use the similarity matrix for clustering (like spectral clustering), but:
 - Without calculating eigenvectors
 - Without constructing or storing the similarity matrix

Power Iteration Clustering

+ Path Folding

31

Path Folding

Okay, we have a fast clustering method – but there's the W that requires $O(n^2)$ storage space and construction and operation time!

Input: W (similarity matrix), k (number of clusters)

Output: C_1, C_2, \dots, C_k (clusters)

- Pick an initial v
- Repeat
 - Set $v^{t+1} \leftarrow Wv^t$
 - Set $\delta^{t+1} \leftarrow |v^{t+1} - v^t|$
 - Increment t
 - Stop when $\delta^{t+1} \approx 0$
- Use k -means to cluster v^{t+1} into clusters C_1, C_2, \dots, C_k

Key operation in PIC: $v^{t+1} \leftarrow Wv^t$


Note: matrix-vector multiplication!

32

Path Folding

- What's so good about matrix-vector multiplication?
- If we can decompose the matrix...

$$v^{t+1} = Wv^t = (ABC)v^t$$



- Then we arrive at the same solution doing a series of matrix-vector multiplications!

$$v^{t+1} = (A(B(Cv^t)))$$

33

Path Folding

- As long as we can decompose the matrix into a series of sparse matrices, we can turn a dense matrix-vector multiplication into a series of sparse matrix-vector multiplications.

This means that we can turn an operation that requires $O(n^2)$ storage and runtime into one that requires $\sim O(n)$ storage and runtime!

This is exactly the case for text data

And many other kinds of data as well!

Details

Path Folding

- Example – inner product similarity:

$$W = D^{-1} F F^T$$

Why is it $\sim n$ and not n ?

Diagonal matrix that normalizes W so rows sum to 1

Storage: $\sim n$

The original feature matrix

Construction: given
Storage: $\sim O(n)$

The feature matrix transposed

Construction: given
Storage: just use F

Details

Path Folding

- Example – inner product similarity:

Construction: $\sim O(n)$

Storage: $\sim O(n)$

Operation: $\sim O(n)$

- Iteration update:

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

Okay...how about a similarity function we actually use for text data?

35

Path Folding

- Example – cosine similarity:

Construction: $\sim O(n)$ Storage: $\sim O(n)$ Operation: $\sim O(n)$

Iteration update:

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

Compact storage: we don't need a cosine-normalized version of the feature vectors

Diagonal cosine normalizing matrix

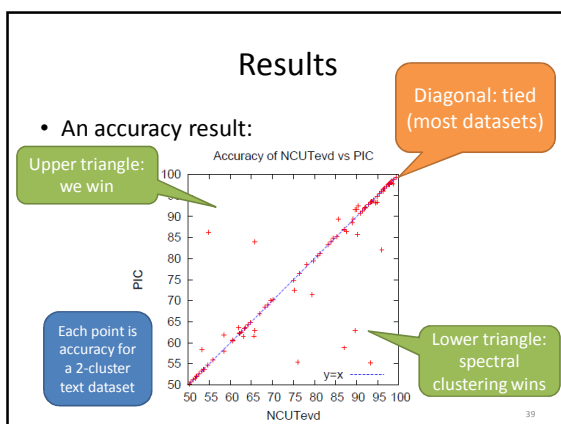
More on cosine similarity 5 or 6 lectures later!

37

Path Folding

- We refer to this technique as path folding due to its connections to “folding” a bipartite graph into a unipartite graph.

38



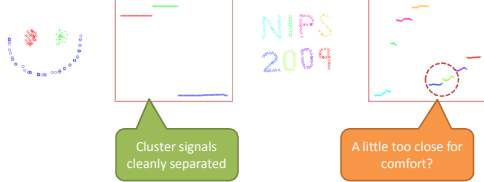
Talk Outline

- Clustering
- Spectral Clustering
- Power Iteration Clustering (PIC)
 - PIC with Path Folding
 - ➡ – PIC Extensions

40

PIC Extension: Avoiding Collisions

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

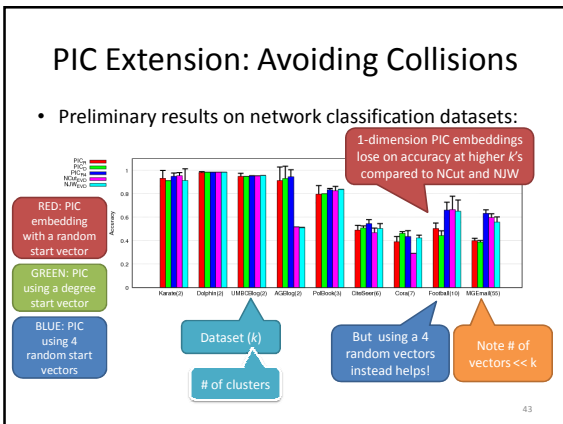


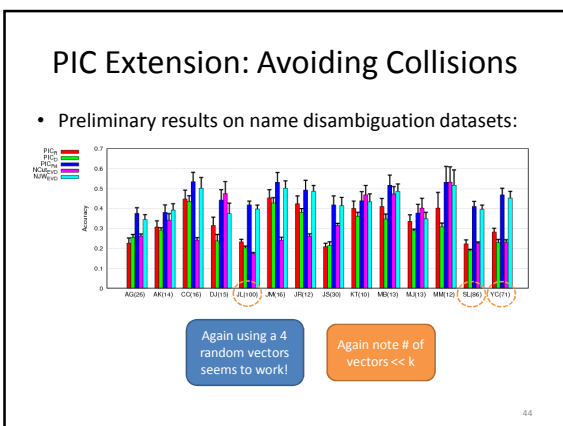
41

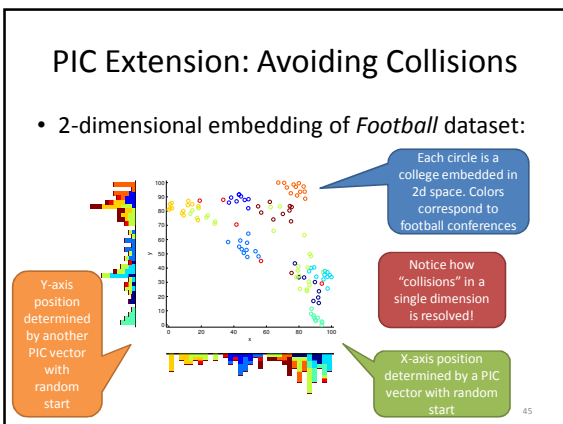
PIC Extension: Avoiding Collisions

- A solution:
 - Run PIC d times with different random starts and construct a d -dimension embedding
 - Unlikely two clusters collide on all d dimensions
 - We can afford it because PIC is fast and space-efficient!

42







PIC Extension: Hierarchical Clustering

- Real, large-scale data may not have a “flat” clustering structure
- A hierarchical view may be more useful

Good News:
The dynamics of a PIC embedding display a hierarchically convergent behavior!

46

Details

PIC Extension: Hierarchical Clustering

- Why?
- Recall PIC embedding at time t :

$\mathbf{v}^t = \frac{c_1}{c_1 \lambda_1} \mathbf{e}_1 + \frac{c_2}{c_1} \left(\frac{\lambda_2}{\lambda_1} \right)^t \mathbf{e}_2 + \frac{c_3}{c_1} \left(\frac{\lambda_3}{\lambda_1} \right)^t \mathbf{e}_3 + \dots + \frac{c_n}{c_1} \left(\frac{\lambda_n}{\lambda_1} \right)^t \mathbf{e}_n$

More salient structure stick around

There may not be a clear eigengap - a gradient of cluster saliency

Less significant eigenvectors / structures go away first, one by one

47

PIC Extension: Hierarchical Clustering

NIPS 2009

Same dataset you've seen

Similar behavior also noted in matrix-matrix power methods (diffusion maps, mean-shift, multi-resolution spectral clustering)

NIPS 2009

PIC already converged to 8 clusters...

But let's keep on iterating...

"N" still a part of the "2009" cluster...

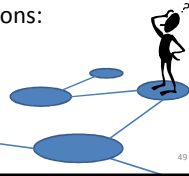
Yes!
(it might take a while)

48

Questions & Discussion

- Further questions & discussions:

- frank@cs.cmu.edu
- GHC 5507



Additional Information

50

PIC: Related Clustering Work

- Spectral Clustering
 - (Roxborough & Sen 1997, Shi & Malik 2000, Meila & Shi 2001, Ng et al. 2002)
- Kernel k -Means (Dhillon et al. 2007)
- Modularity Clustering (Newman 2006)
- Matrix Powering
 - Markovian relaxation & the information bottleneck method (Tishby & Slonim 2000)
 - matrix powering (Zhou & Woodruff 2004)
 - diffusion maps (Lafon & Lee 2006)
 - Gaussian blurring mean-shift (Carreira-Perpinan 2006)
- Mean-Shift Clustering
 - mean-shift (Fukunaga & Hostetler 1975, Cheng 1995, Comaniciu & Meer 2002)
 - Gaussian blurring mean-shift (Carreira-Perpinan 2006)

51

PIC: Some "Powering" Methods at a Glance

Method	W	Iterate	Stopping	Final
Tishby & Stolin 2000	$W=D^{-1}A$	$W^{t+1}=W^t$	rate of information loss	information bottleneck method
Zhou & Woodruff 2004	$W=A$	$W^{t+1}=W^t$	a small t	a threshold ϵ
Carreira-Perpinan 2006	$W=D^{-1}A$	$X^{t+1}=WX$	entropy	a threshold ϵ
PIC	$W=D^{-1}A$	$V^{t+1}=WV^t$	acceleration	k-means

How far can we go with a one- or low-dimensional embedding?

52

PIC: Versus Popular Fast Sparse Eigencomputation Methods

For Symmetric Matrices	For General Matrices	Improvement
	Successive Power Method	Basic; numerically unstable, can be slow
Lanczos Method	Arnoldi Method	More stable, but may require lots of time and memory
Implicitly Restarted Lanczos Method (IRLM)	Implicitly Restarted Arnoldi Method (IRAM)	More time- and memory-efficient

Randomized sampling methods are also popular

n = # nodes
 e = # edges
 k = # eigenvectors
 $m(k)$ = Arnoldi Length

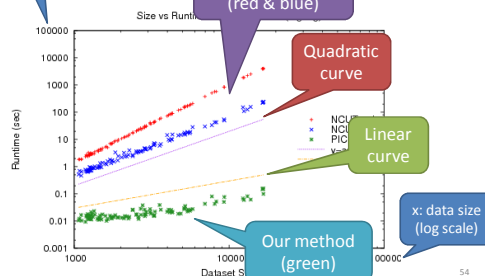
Method	Time	Space
IRAM	$O(m^2) + O(nm) + O(e) \times O(m-k) \times (\# \text{ restart})$	$O(e) + O(nm)$
PIC	$O(e) \times (\# \text{ iterations})$	$O(e)$

53

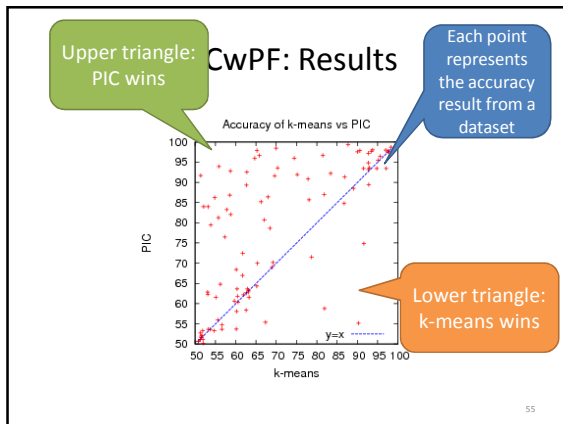
PICwPF: Results

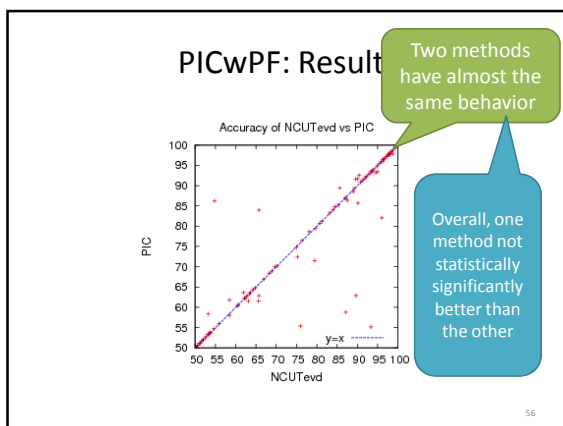
y: algorithm runtime (log scale)

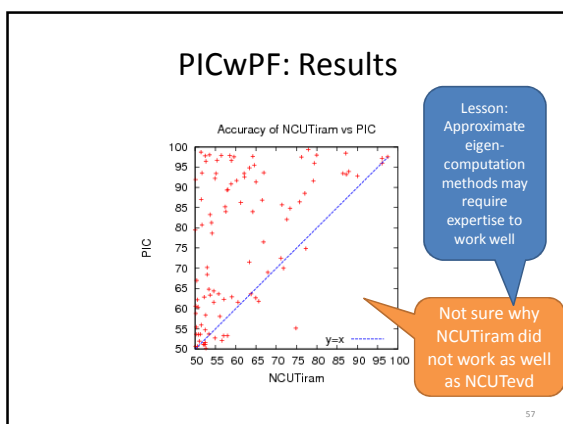
- A scalability result:



54

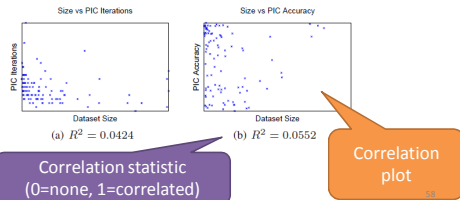






PICwPF: Results

- PIC is $O(n)$ per iteration and the runtime curve looks linear...
- But I don't like eyeballing curves, and perhaps the number of iteration increases with size or difficulty of the dataset?



PICwPF: Results

- Linear run-time implies *constant* number of iterations.
- Number of iterations to “acceleration-convergence” is hard to analyze:
 - Faster than a single complete run of power iteration to convergence.
 - On our datasets
 - 10-20 iterations is typical
 - 30-35 is exceptional

59

PICwPF: Related Work

- Faster spectral clustering
 - Approximate eigendecomposition (Lanczos, IRAM)
 - Sampled eigendecomposition (Nyström)
- Sparser matrix
 - Sparse construction
 - k-nearest-neighbor graph
 - k-matching
 - graph sampling / reduction

Not $O(n)$ time methodsStill require $O(n^2)$ construction in generalNot $O(n)$ space methods

60

PICwPF: Results

	ACC-Avg	NMI-Avg
baseline	57.59	-
k-means	69.43	0.2629
NCUTevd	77.55	0.3962
NCUTiram	61.63	0.0943
PIC	76.67	0.3818

61