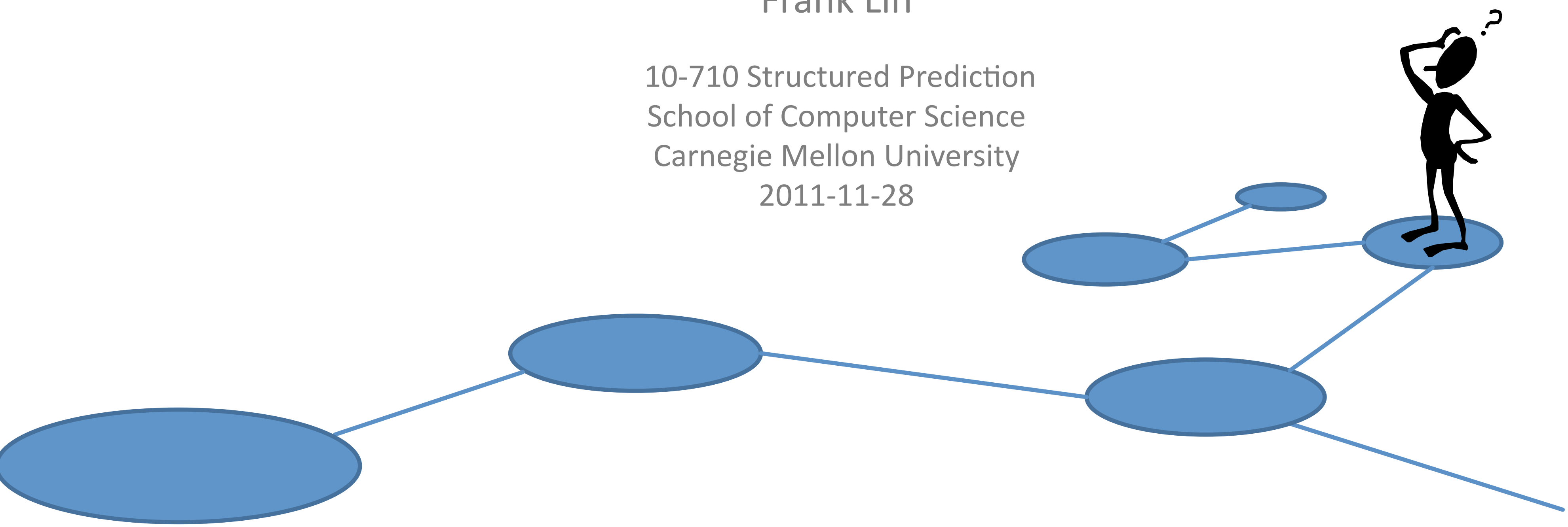
Power Iteration Clustering

Frank Lin

10-710 Structured Prediction
School of Computer Science
Carnegie Mellon University
2011-11-28

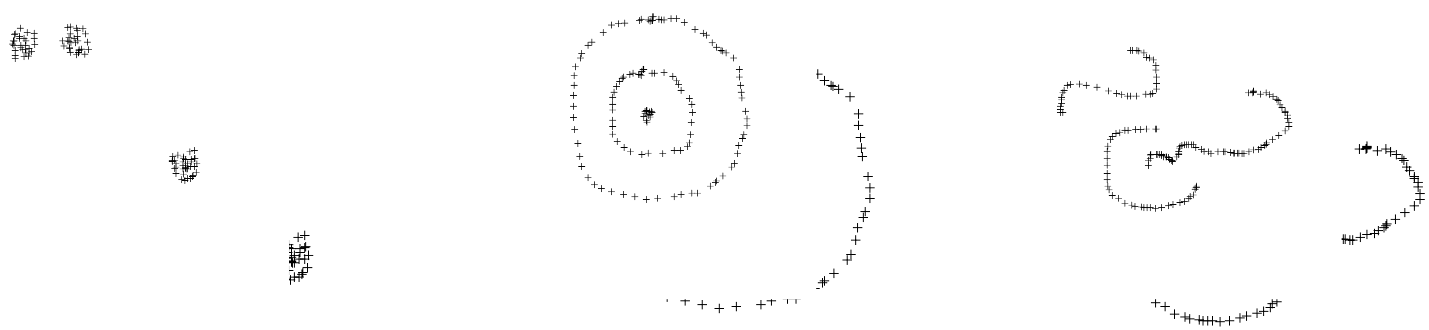


Talk Outline

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

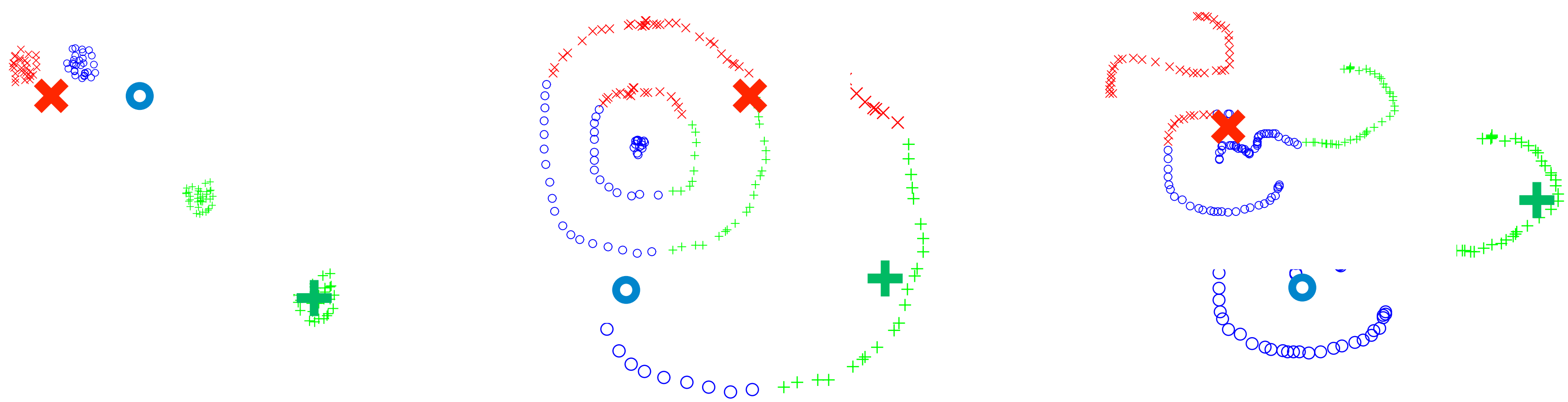
Clustering

- Automatic grouping of data points
- 3 example datasets:



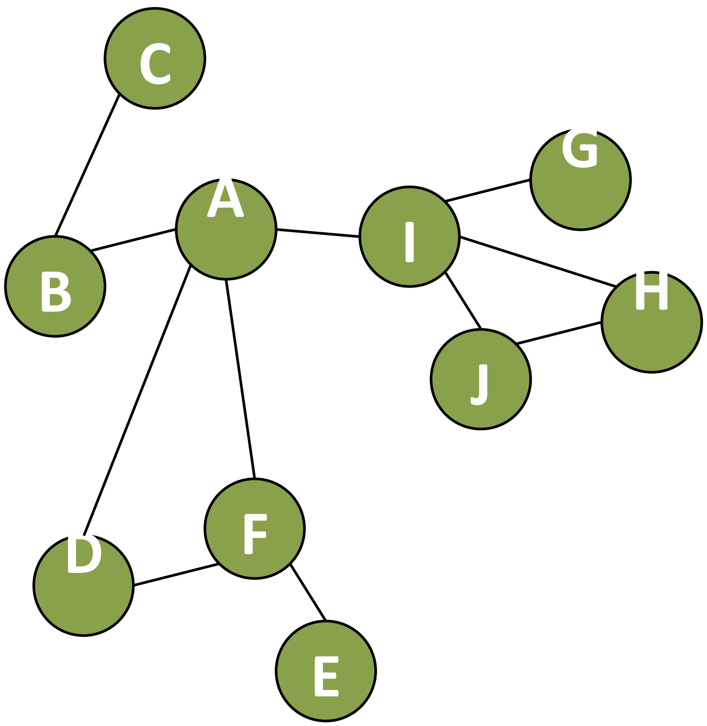
k-means

- 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



	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	

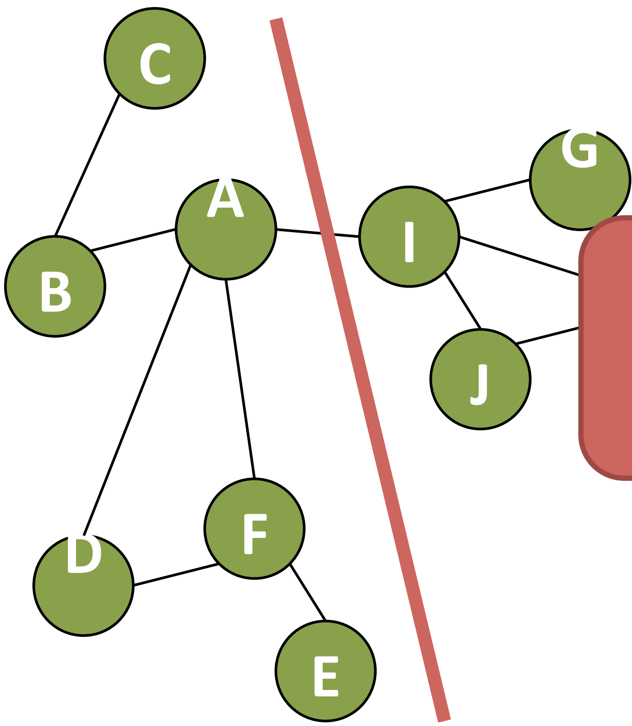
Graph Cluster

Example - Normalized Cut:

$$ncut(A,B) = \frac{w(A,B)}{w(A,V)} + \frac{w(A,B)}{w(B,V)}$$

Find: Partitions of the graph

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



Exact Solution is NP-hard!

	A	B	C	D	E	F	G	H	I	J
A		1		1		1			1	
B	1		1							
[illegible]	[illegible]	[illegible]								
[illegible]	[illegible]	[illegible]				1				
[illegible]	[illegible]	[illegible]				1				
[illegible]	[illegible]	[illegible]		1	1					
G									1	
H									1	1
I	1						1	1		1
J								1	1	

Talk Outline

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

Spectral Clusterin

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

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

Recall that
similarity can be
represented as a
graph/matrix

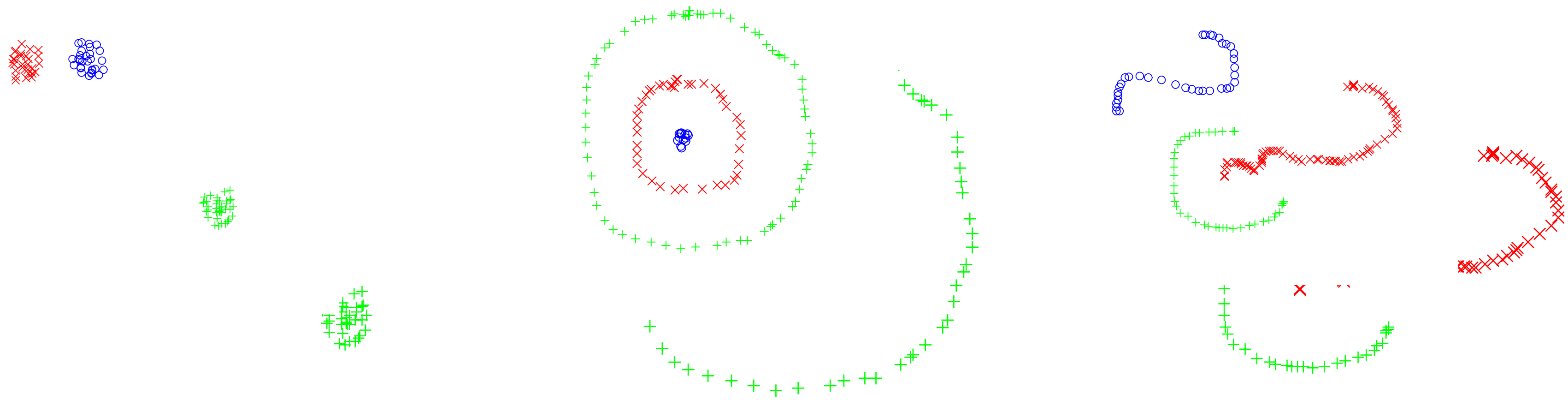
Spectral Clustering

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

A popular spectral clustering method: normalized cuts (NCut)

Spectral Clustering

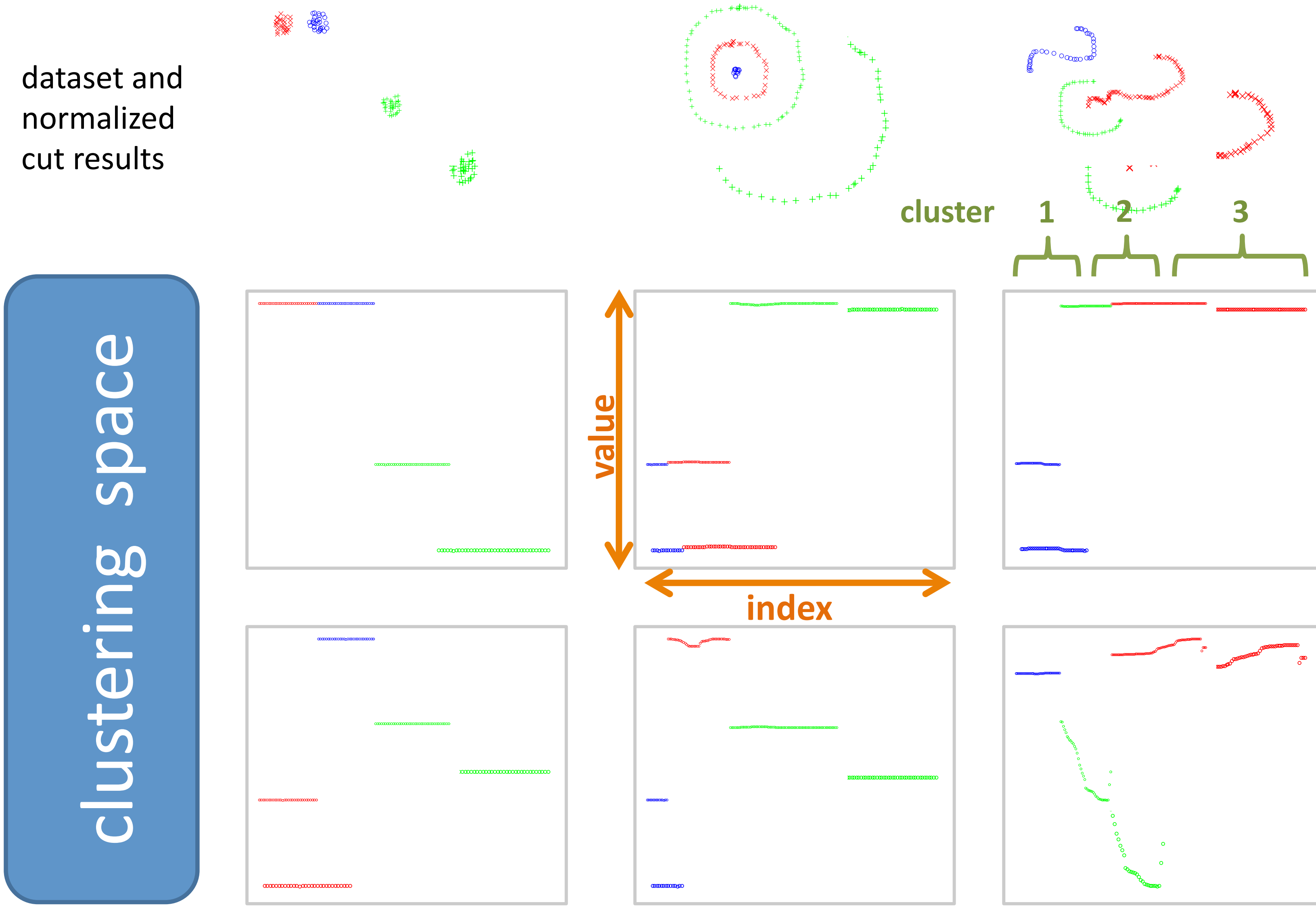
- Results with Normalized Cuts:



☺ ☺ ☺

Spectral Clustering

dataset and normalized cut results



Spectr

Can we find a similar low-dimensional embedding for clustering without eigenvectors?

Finding eigenvectors and eigenvalues of a matrix is still pretty slow in general

lgorithm (Shi & Malik 2000):

milarity function *s*

2. Deriv… *s*, let $W=I-D^{-1}A$, where I is the identity matrix a… D is a diagonal square matrix $D_{ii}=\Sigma_j A_{ij}$
3. Find eigenvectors and corresponding eigenvalues of W
4. Pick the k eigenvectors of W with the 2^{nd} to k^{th} 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

Talk Outline

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

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)!

The Power Iteration

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

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

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

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

W : a square matrix

v^t : the vector at iteration *t*;

v^0 typically a random vector

The Power Iteration

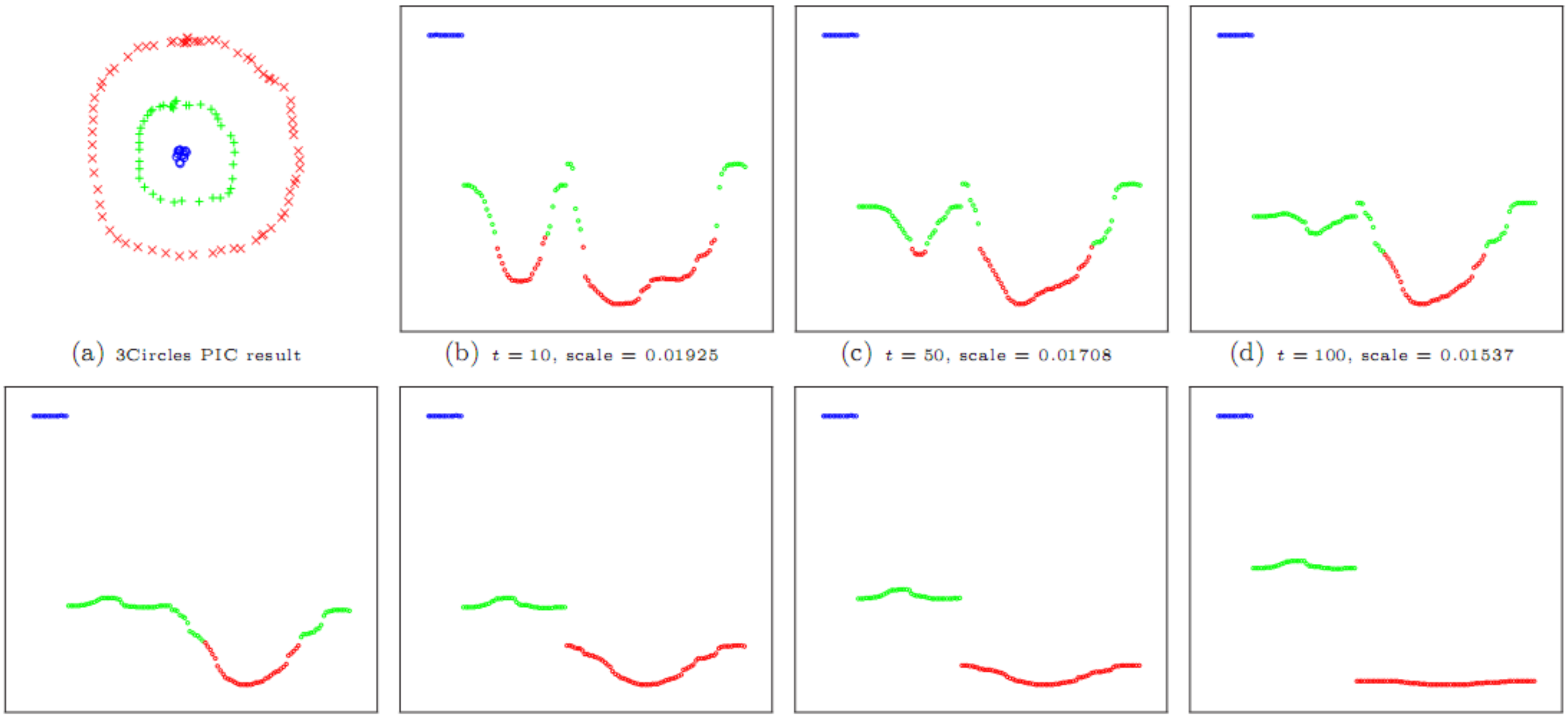
- Or the power method, is a simple iterative method for finding the dominant eigenvector of a matrix:

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

Row-normalized similarity matrix

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

The Power Iteration



(a) 3Circles PIC result (b) $t = 10$, scale = 0.01925 (c) $t = 50$, scale = 0.01708 (d) $t = 100$, scale = 0.01537

(e) $t = 200$, scale = 0.01316 (f) $t = 400$, scale = 0.01066 (g) $t = 600$, scale = 0.00929 (h) $t = 1000$, scale = 0.00786

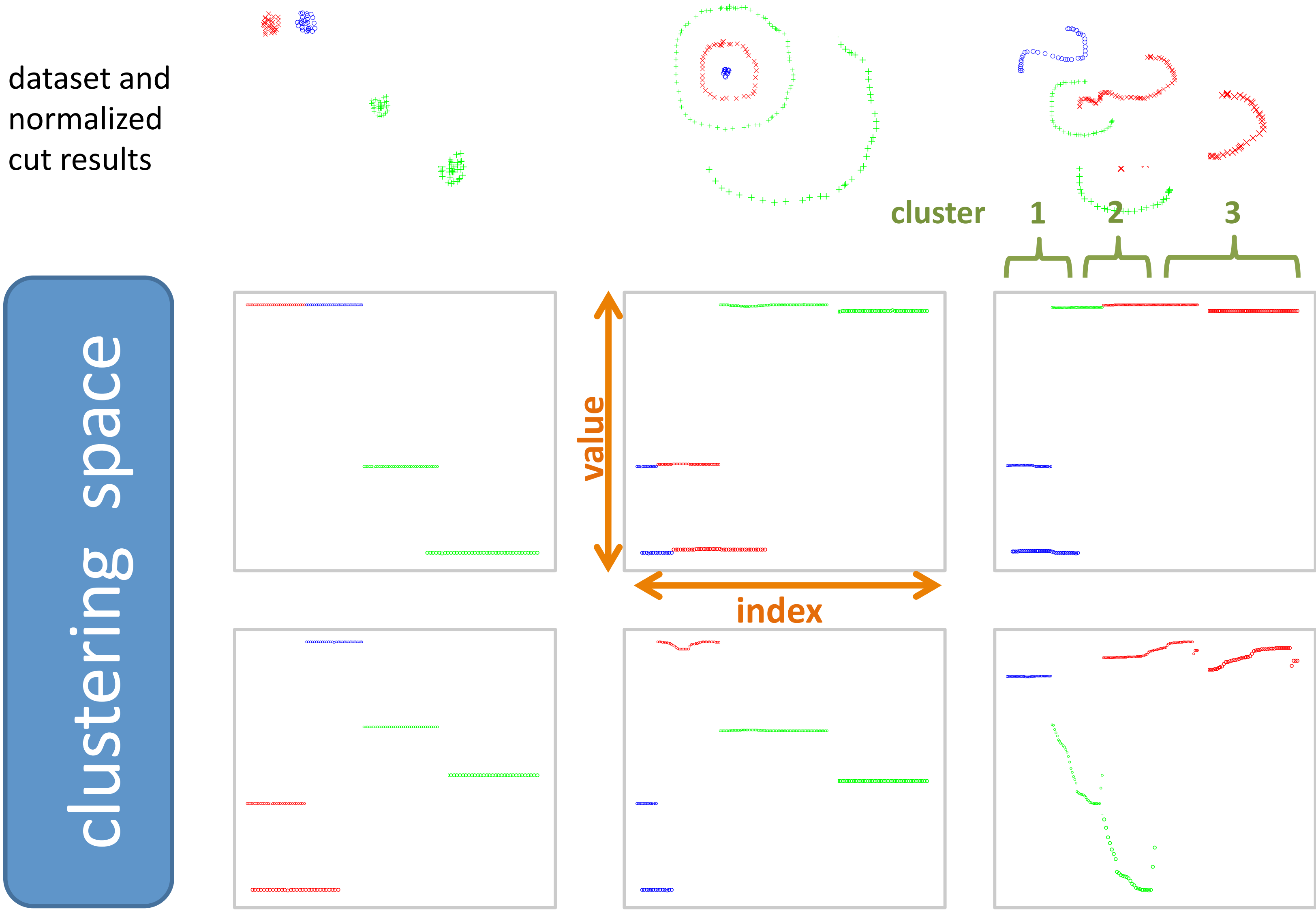
Power Iteration Clustering

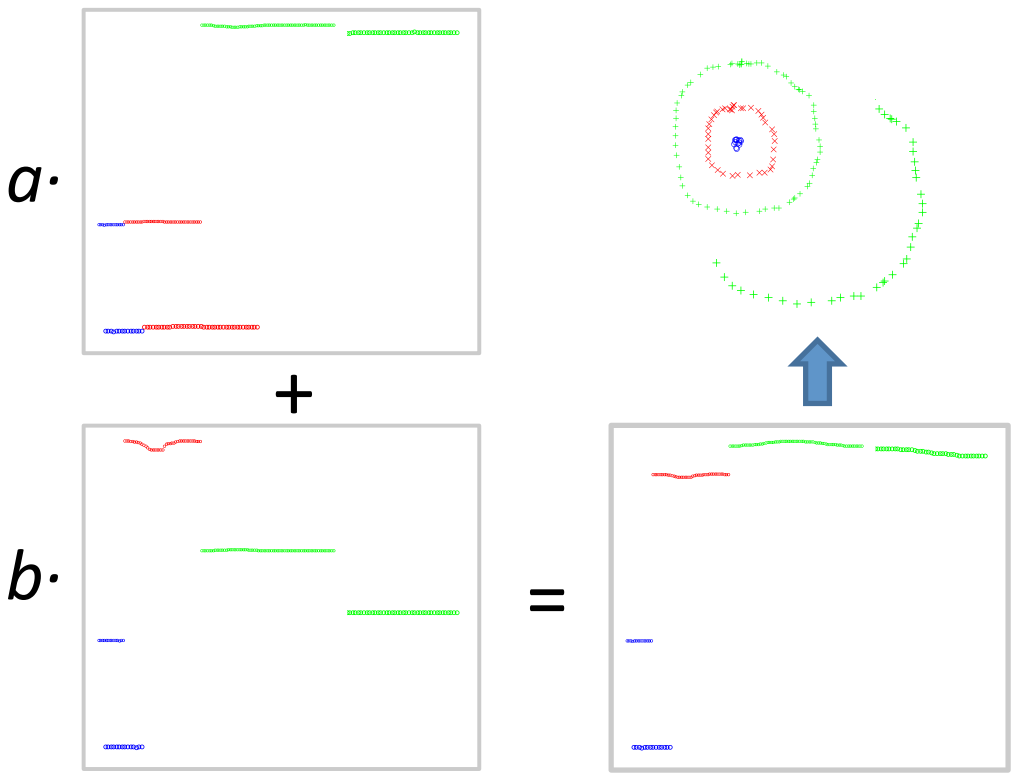
- *The 2nd to kth eigenvectors of $W=D^{-1}A$ are roughly piece-wise constant with respect to the underlying clusters, each separating a cluster from the rest of the data*

- The linear combination of piece-wise constant vectors is also piece-wise constant!

Spectral Clustering

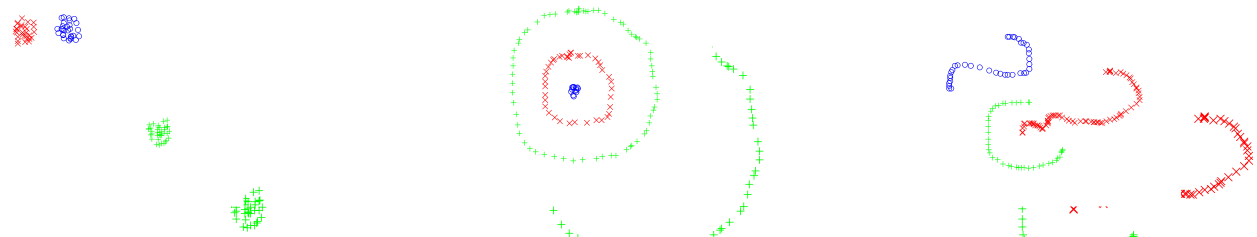
dataset and normalized cut results




a·
+
b·
=

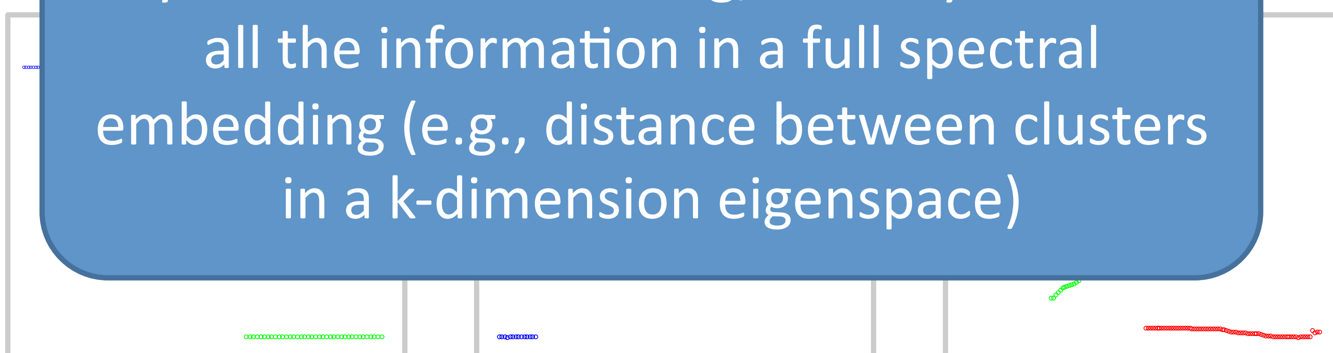
Power Iteration Clustering

dataset and
PIC results



Key idea: to do clustering, we may not need all the information in a full spectral embedding (e.g., distance between clusters in a k-dimension eigenspace)

v^t



we just need the clusters to be <u>separated</u> in some space.

Details

When to Stop

Recall:

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

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

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 $C_1, C_2, \ldots, C_k$

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 $C_1, C_2, \ldots, C_k$

i.e., when acceleration is nearly zero

PIC Runtime

Normalized Cut

Normalized Cut, faster implementation

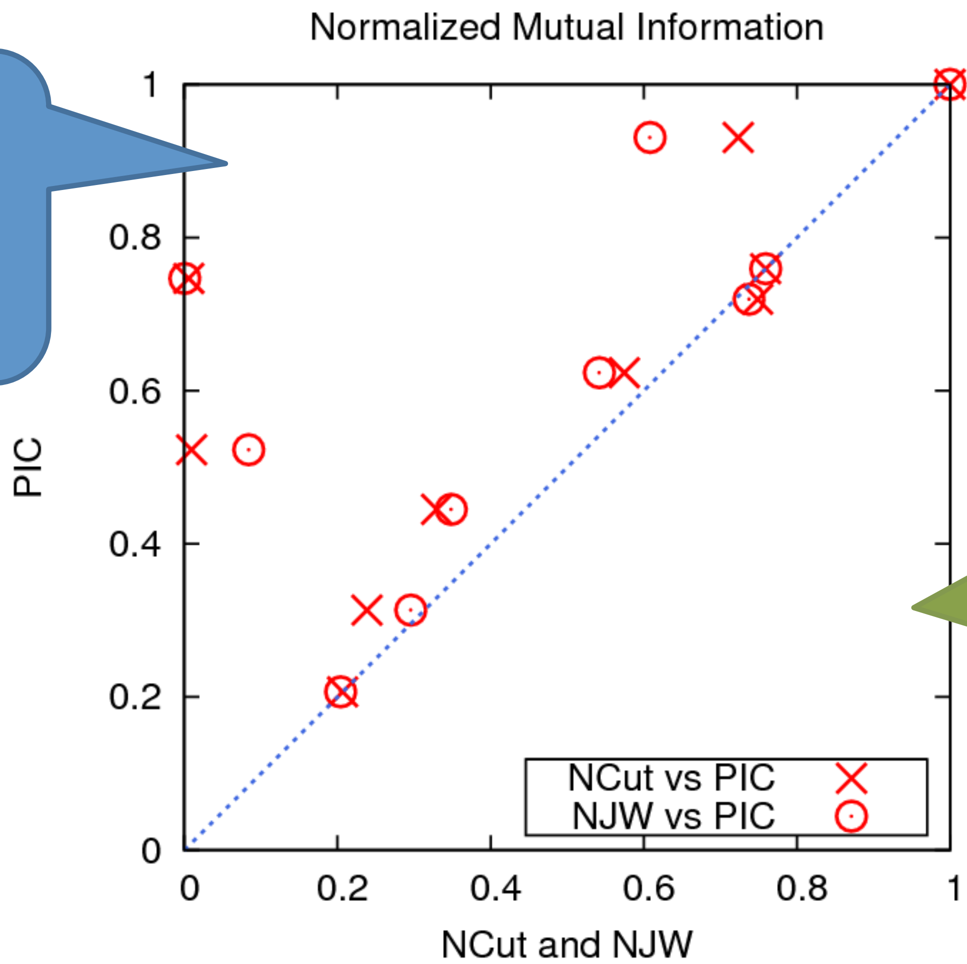
Table 4. Runtime comparison (in milliseconds) of PIC and spectral clustering algorithms on synthetic datasets.

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)

PIC Accuracy on Network Datasets

Upper triangle: PIC does better



Lower triangle: NCut or NJW does better

Talk Outline

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

Clustering Text Data

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

(A lot of)

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

	27	125	-
	23	-	125
	-	23	27

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

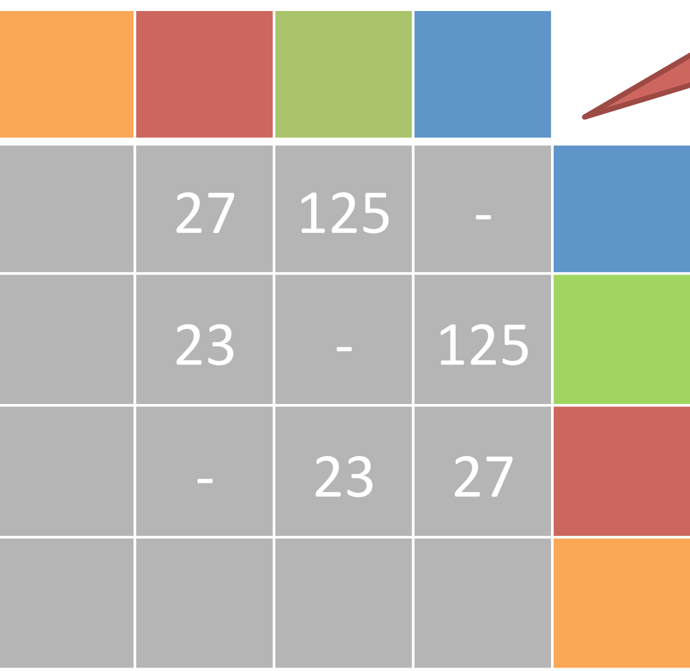
cool	web	search	make	over	you	
0	4	8	2	5	3	
0	8	7	4	3	2	
1	0	0	0	1	2	

The Problem with Text

- A similarity matrix is the input to many clustering 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

Too expensive! Does not scale up to big datasets!



$O(n^2)$ time to construct

$O(n^2)$ space to store

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

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

Path Folding

- A ba ... orithm:

Input: ... usters k
Output ...

1. Pick an initial v ...
2. Repeat
   - Set $\mathbf{v}^{t+1} \leftarrow W\mathbf{v}^t$
   - Set $\boldsymbol{\delta}^{t+1} \leftarrow |$ ... $^{t+1} - \mathbf{v}^t|$
   - Increment ...
   - Stop when ... $^{t+1}| \approx \mathbf{0}$
3. Use *k*-me ... lusters $C_1, C_2, \ldots, C_k$

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

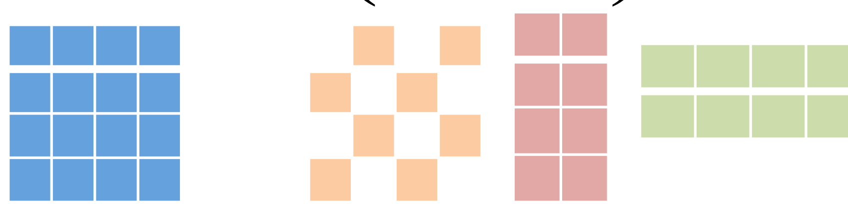
Key operation in PIC

Note: matrix-vector multiplication!

Path Folding

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

$$\mathbf{v}^{t+1} = W\mathbf{v}^{t} = (ABC)\mathbf{v}^{t}$$



How could this be better?

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

$$\mathbf{v}^{t+1} = (A(B(C\mathbf{v}^{t})))$$

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.*

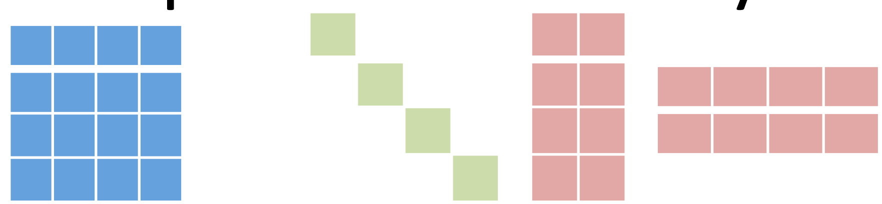
This means that we can turn an operation that requires $O(n^2)$ storage and runtime into one that requires ~O(n) storage and runtime!

This is exactly the case for text data

And many other kinds of data as well!

Path Folding

- Example – inner product similarity:



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

Why is it ~n and not n?

Diagonal matrix that normalizes *W* so rows sum to 1

The original feature matrix

The feature matrix transposed

Storage: ~n

Construction: given
Storage: ~O(n)

Construction: given
Storage: just use F

Details

Path Folding

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

- Example – inner product similarity:

Construction: ~O(n)

Storage: ~O(n)

Operation: ~O(n)

- Iteration update:

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

Details

Path Folding

Diagonal cosine normalizing matrix

- Example – cosine similarity:

Construction: ~O(n)

Storage: ~O(n)

Operation: ~O(n)

- Iteration update:

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

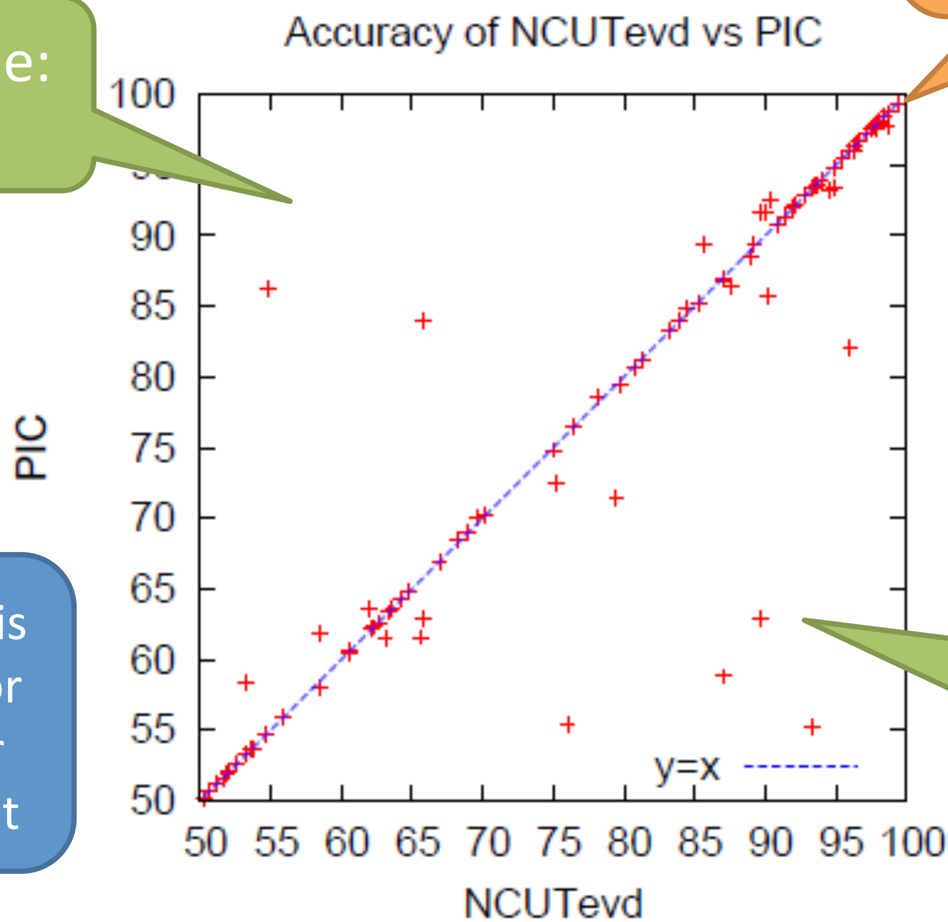
Compact storage: we don't need a cosine-normalized version of the feature vectors

Path Folding

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

Results

- An accuracy result:



Upper triangle: we win

Diagonal: tied (most datasets)

Each point is accuracy for a 2-cluster text dataset

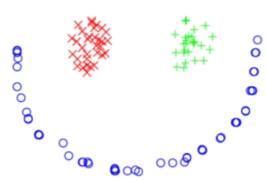
Lower triangle: spectral clustering wins

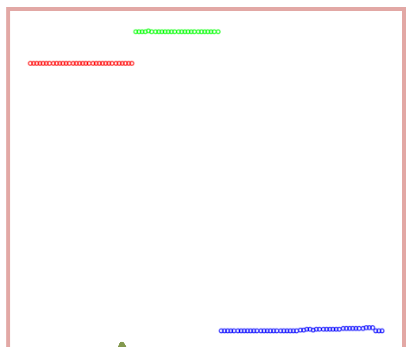
Talk Outline

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

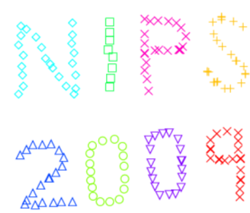
PIC Extension: Avoiding Collisions

- 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"?





Cluster signals cleanly separated



A little too close for comfort?

PIC Extension: Avoiding Collisions

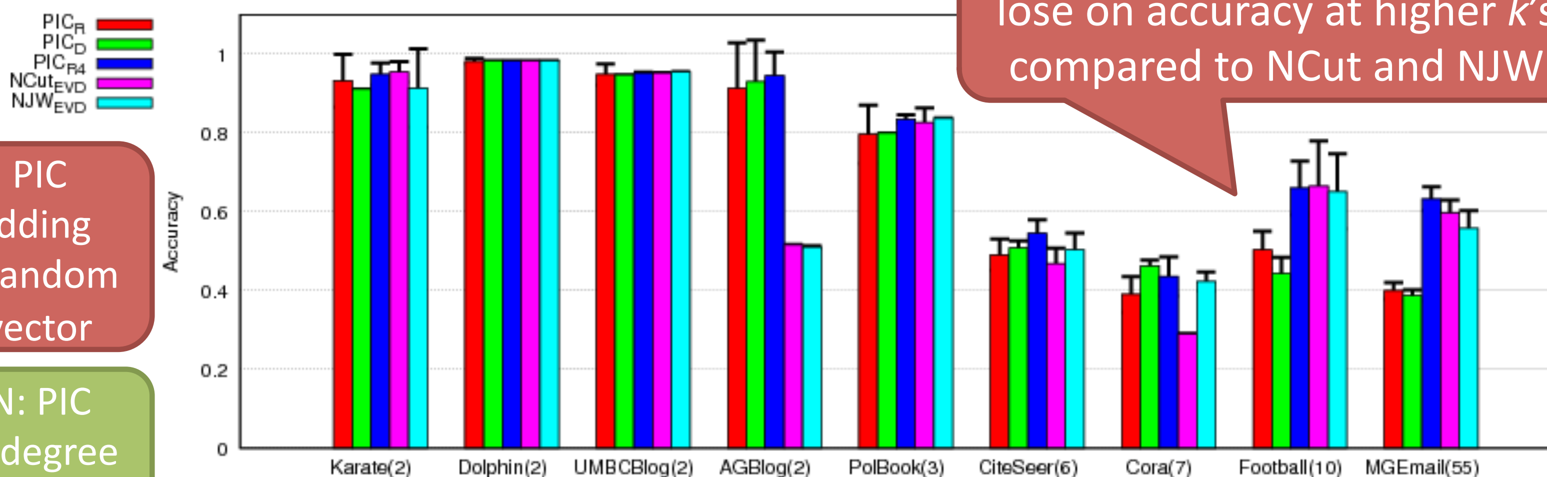
- A solution:

  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!

PIC Extension: Avoiding Collisions

- Preliminary results on network classification datasets:



1-dimension PIC embeddings lose on accuracy at higher *k*'s compared to NCut and NJW

RED: PIC embedding with a random start vector

GREEN: PIC using a degree start vector

BLUE: PIC using 4 random start vectors

Dataset (*k*)

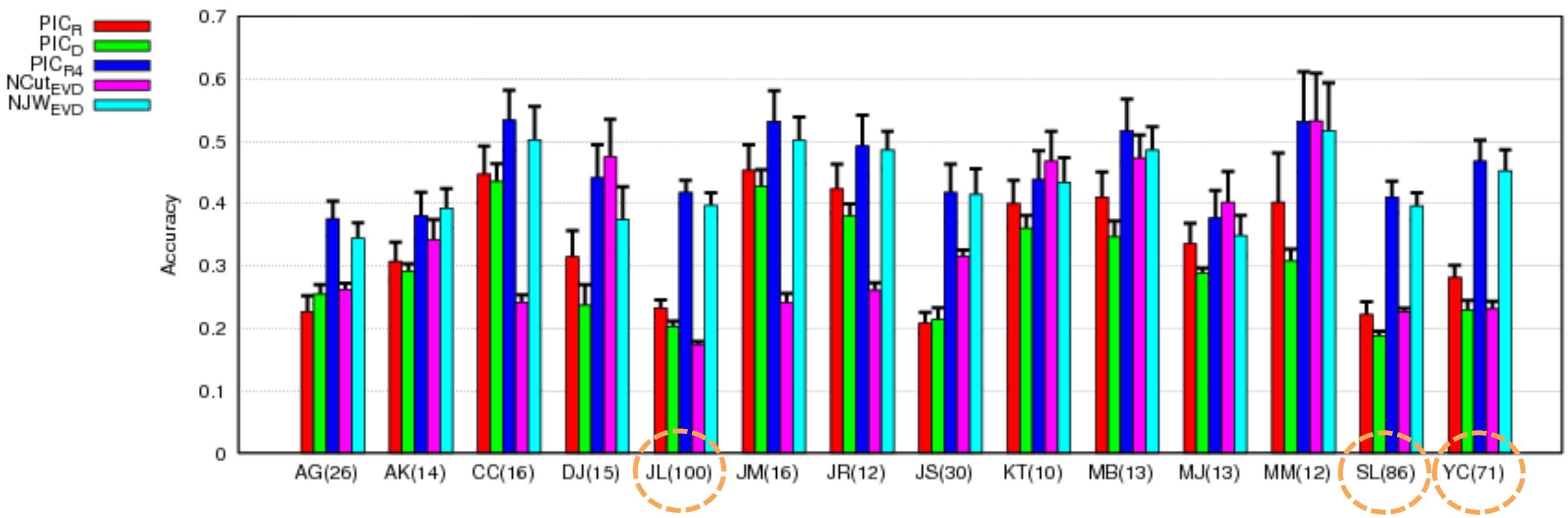
# of clusters

But using a 4 random vectors instead helps!

Note # of vectors << k

PIC Extension: Avoiding Collisions

- Preliminary results on name disambiguation datasets:

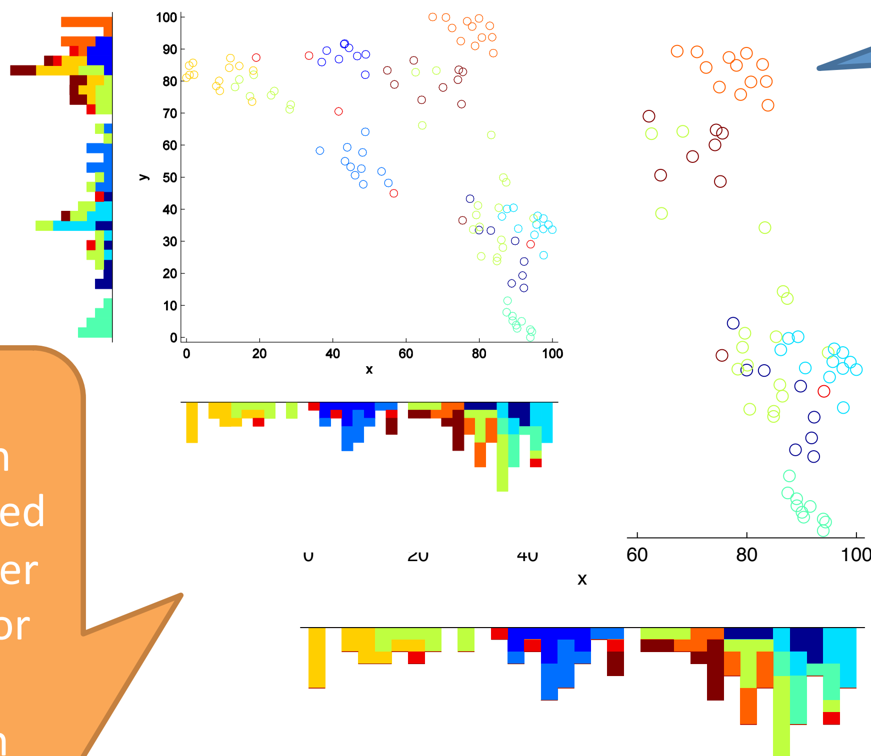


Again using a 4 random vectors seems to work!

Again note # of vectors << k

PIC Extension: Avoiding Collisions

- 2-dimensional embedding of *Football* dataset:



Each circle is a college embedded in 2d space. Colors correspond to football conferences

Y-axis position determined by another PIC vector with random start

Notice how “collisions” in a single dimension is resolved!

X-axis position determined by a PIC vector with random start

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!

Details

PIC Extension: Hierarchical Clustering

- Why?
- Recall PIC embedding at time *t*:

e's – eigenvectors (structure)

Big

Small

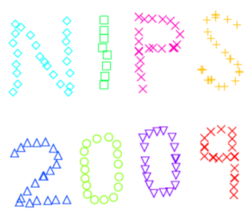
$$\frac{\mathbf{v}^t}{c_1\lambda_1^t} = \mathbf{e}_1 + \frac{c_2}{c_1}\left(\frac{\lambda_2}{\lambda_1}\right)^t \mathbf{e}_2 + \frac{c_3}{c_1}\left(\frac{\lambda_3}{\lambda_1}\right)^t \mathbf{e}_3 + \ldots + \frac{c_n}{c_1}\left(\frac{\lambda_n}{\lambda_1}\right)^t \mathbf{e}_n$$

More salient structure stick around

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

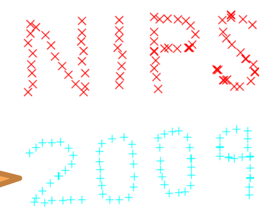
Less significant eigenvectors / structures go away first, one by one

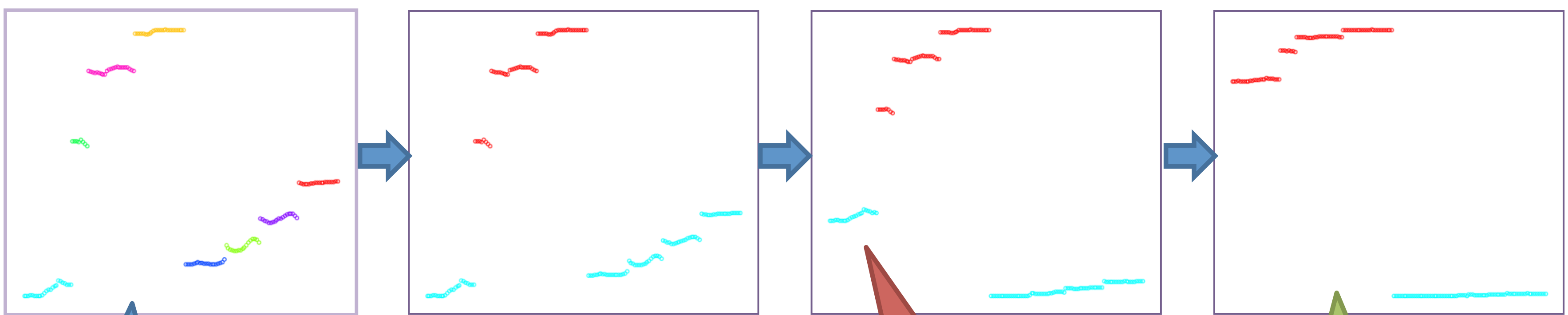
PIC Extension: Hierarchical Clustering



Same dataset you've seen

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





PIC already converged to 8 clusters…

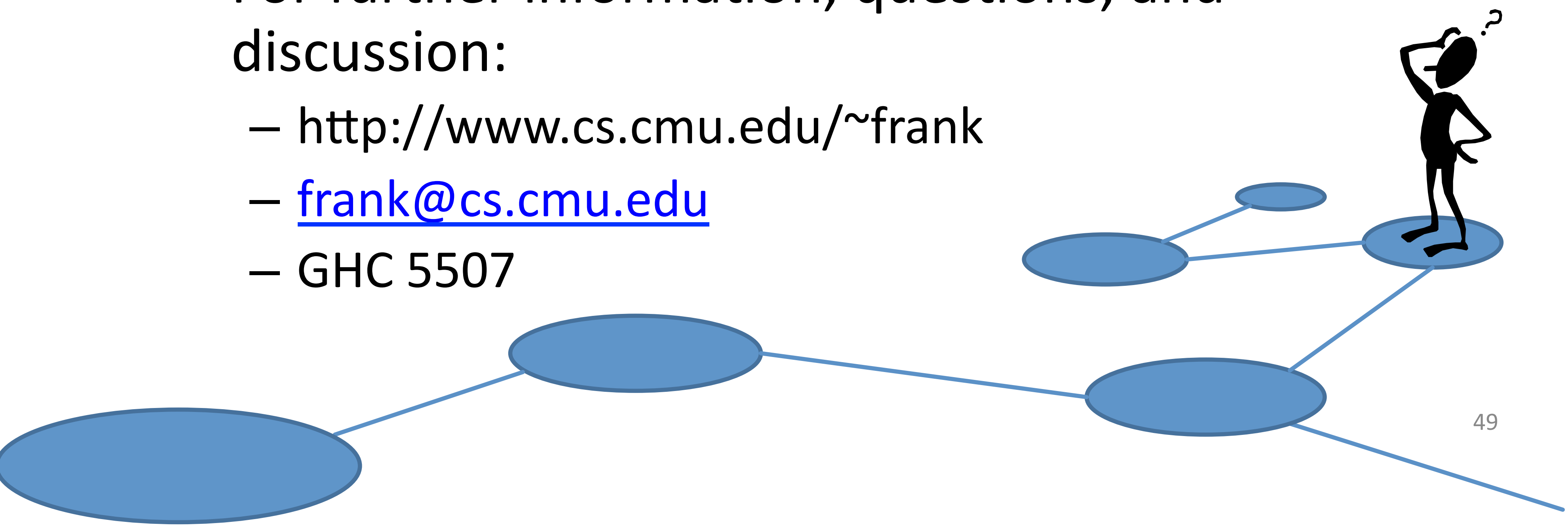
But let's keep on iterating…

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

Yes!
(it might take a while)

Questions & Discussion

- For further information, questions, and discussion:
  - http://www.cs.cmu.edu/~frank
  - frank@cs.cmu.edu
  - GHC 5507



Additional Information

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)

PIC: Some “Powering” Methods at a Glance

Method	W	Iterate	Stopping	Final
Tishby & Slonim 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?

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

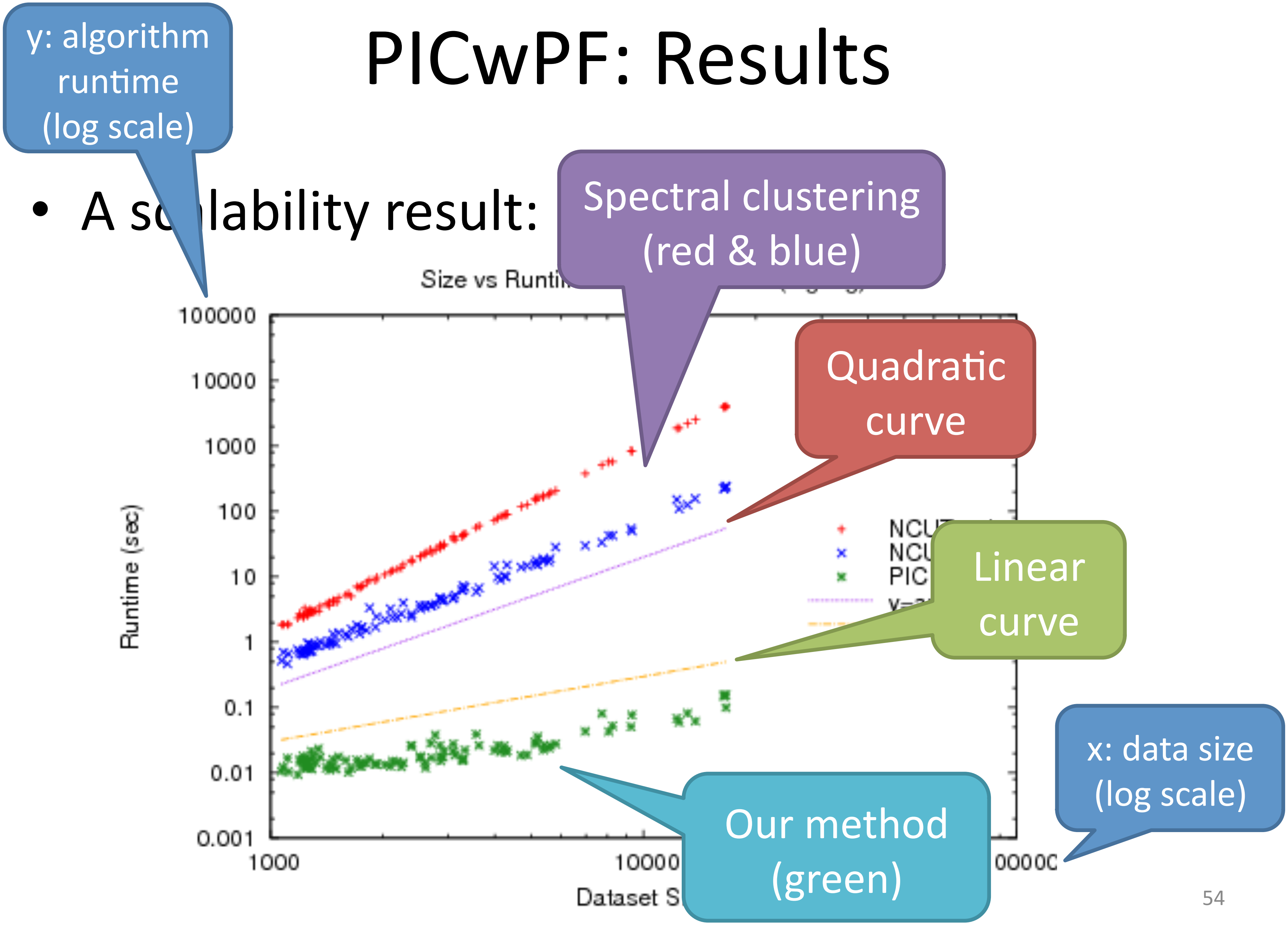
Randomized sampling methods are also popular

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

Method	Time	Space
IRAM	$(O(m^3)+(O(nm)+O(e))\times O(m\text{-}k))\times(\#\ restart)$	$O(e)+O(nm)$
PIC	$O(e)x(\#\ iterations)$	$O(e)$

PICwPF: Results

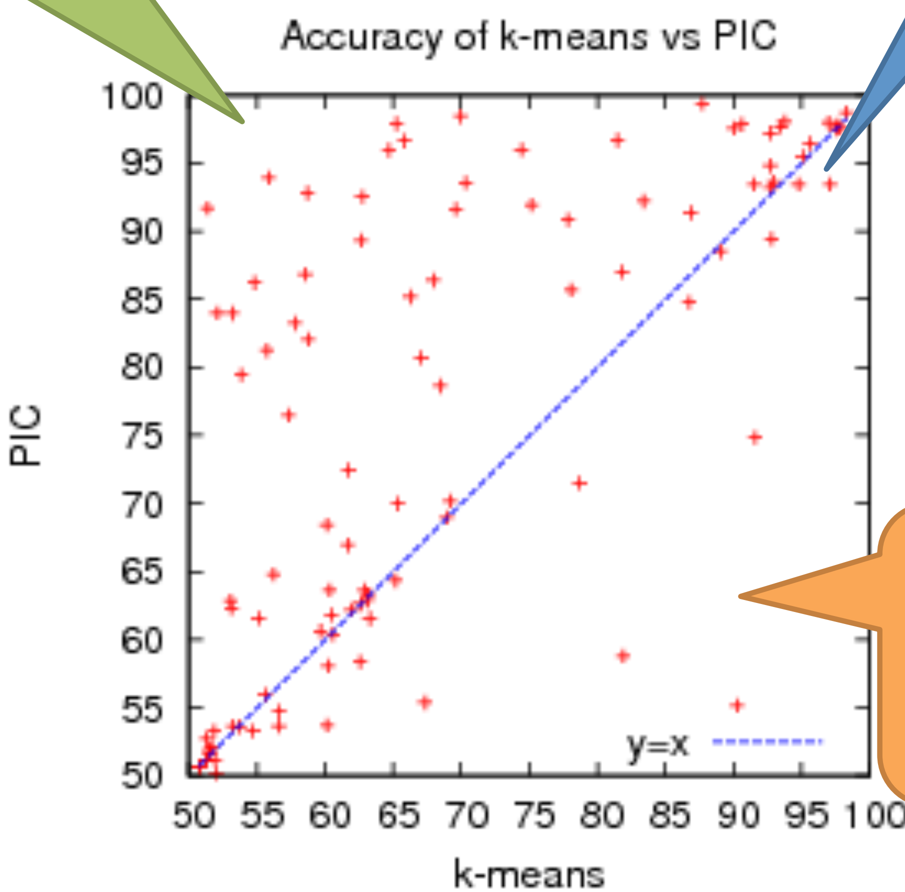
- A scalability result:



CwPF: Results

Upper triangle: PIC wins

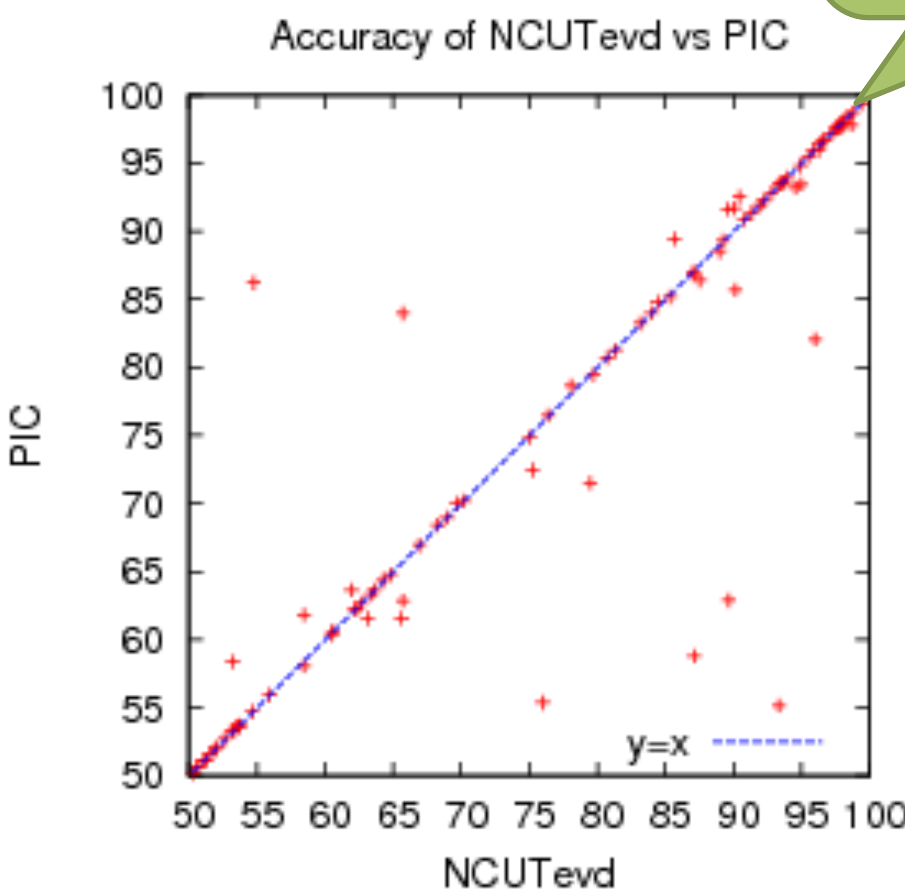
Each point represents the accuracy result from a dataset



Lower triangle: k-means wins

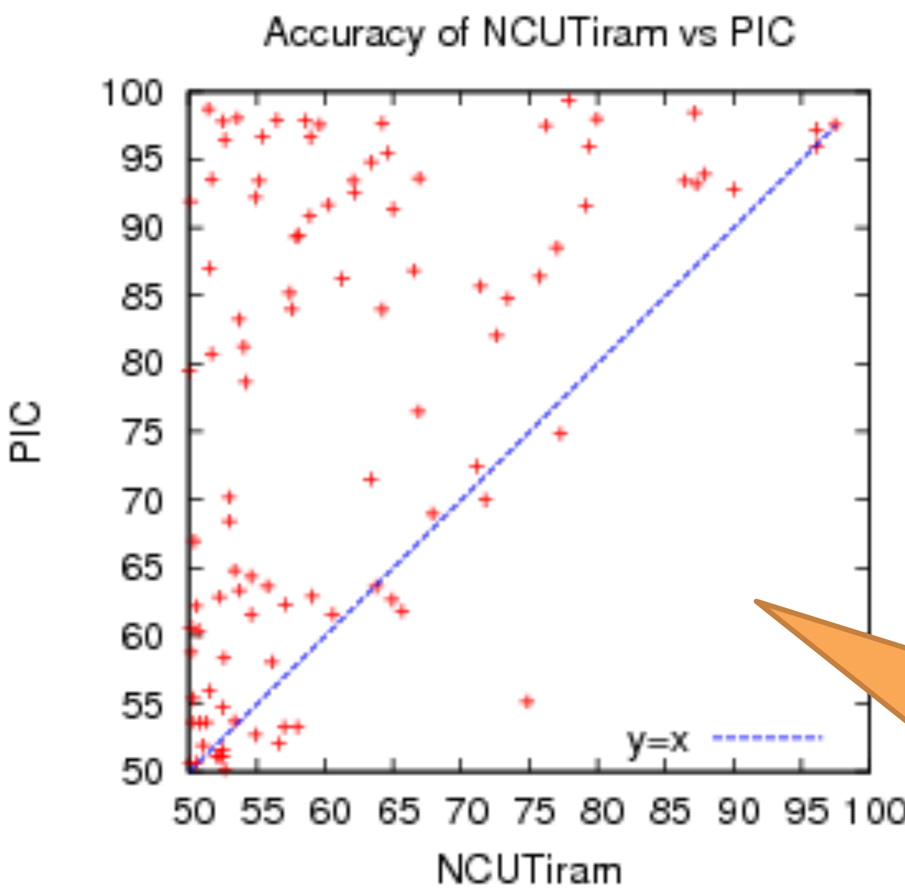
PICwPF: Results

Two methods have almost the same behavior



Overall, one method not statistically significantly better than the other

PICwPF: Results

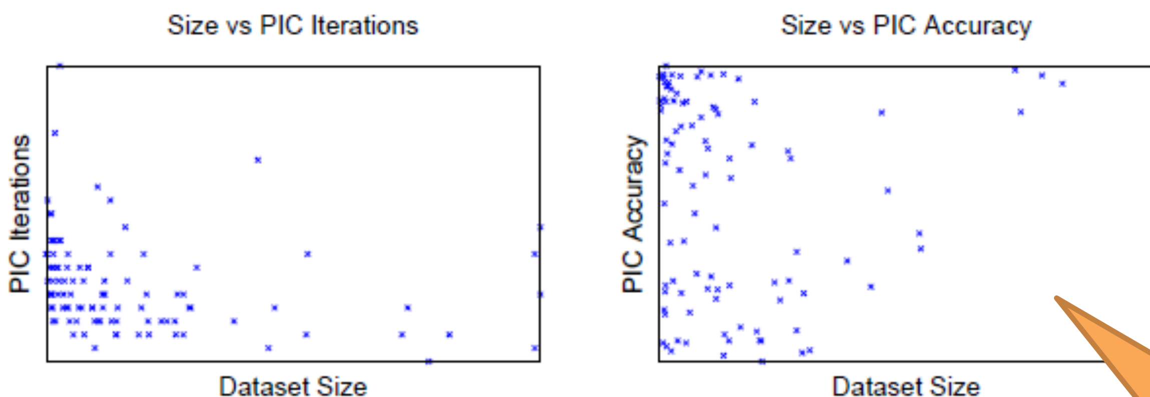


Lesson: Approximate eigen-computation methods may require expertise to work well

Not sure why NCUTiram did not work as well as NCUTevd

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?



(a) $R^2 = 0.0424$ (b) $R^2 = 0.0552$

Correlation statistic (0=none, 1=correlated)

Correlation plot

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

PICwPF: Related Wo

Not O(n) time methods

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

Still require O(n^2) construction in general

Not O(n) space methods

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**