8th Annual Benefit Concert

Friday, December 1st

7:30 - 10:00 PM

(Doors open at 7:00 PM)
Pittsburgh Friends Meeting House
4836 Ellsworth Avenue, 15213

An evening of music with

Smokestack Lightning

and special guests Raging Grannies, Chie Togami, Penny Anderson, Chuck Bowen and Sarah Bowen-Salio

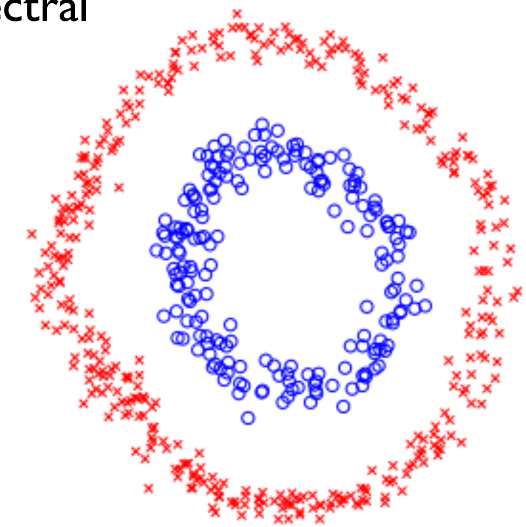
---

Donation: $15 ($6 students/unemployed)

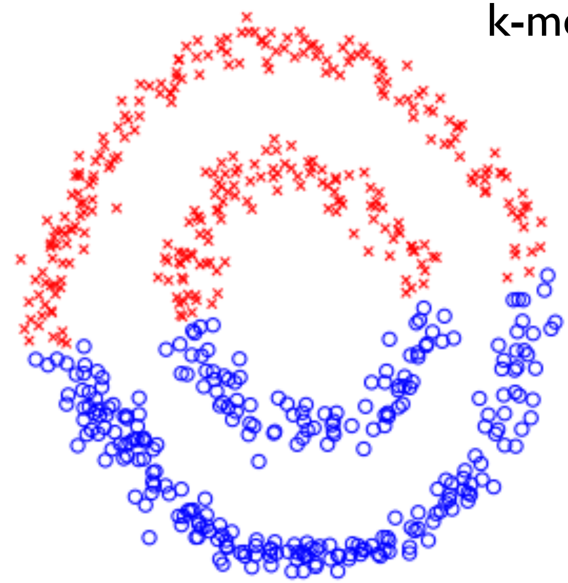
Bake Sale & Refreshments
Benefit for Casa San Jose & Pittsburghers for Public Transit

Spectral Clustering

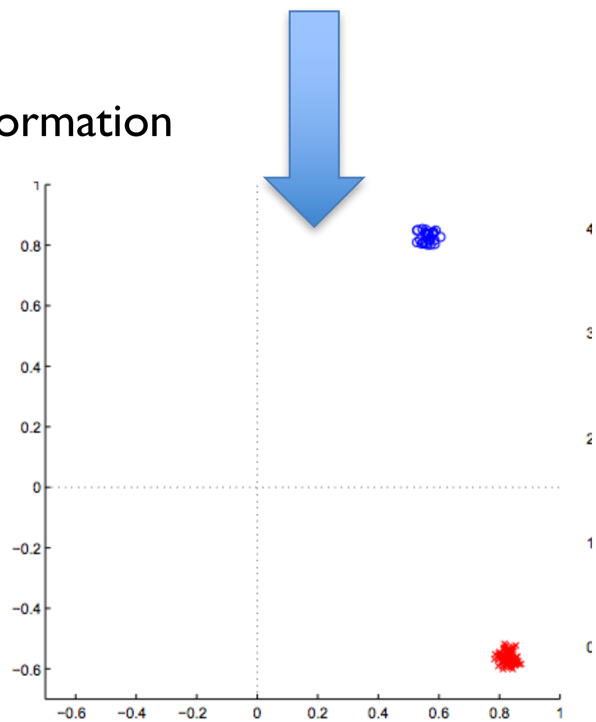
spectral

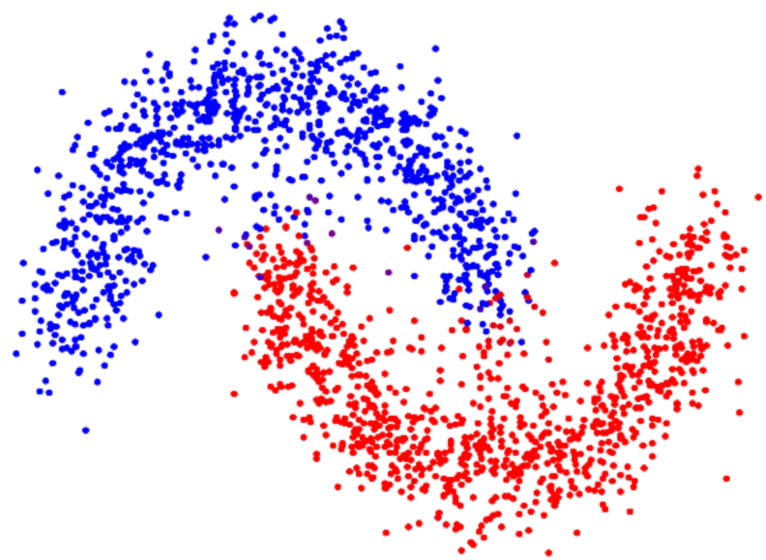


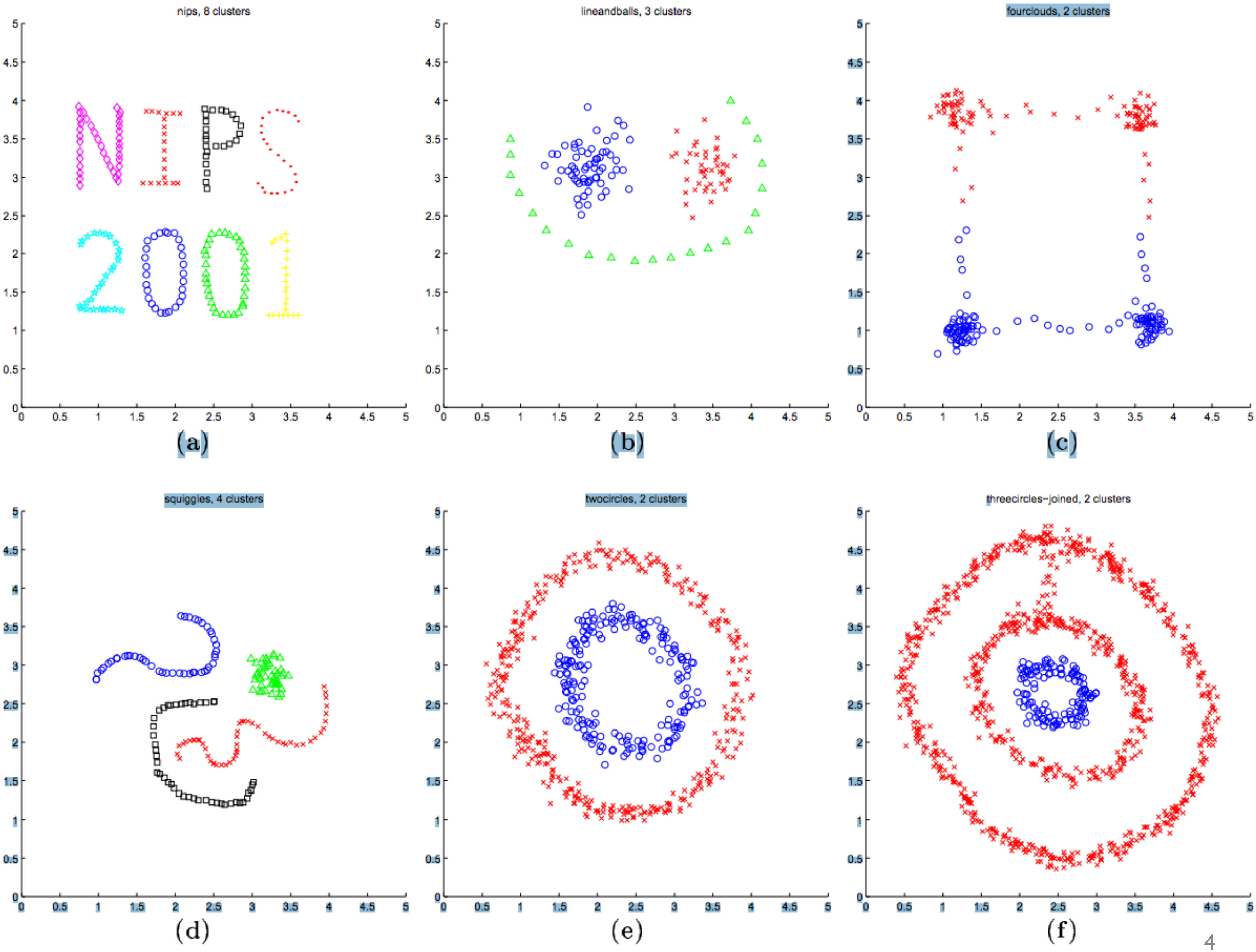
k-means



after transformation





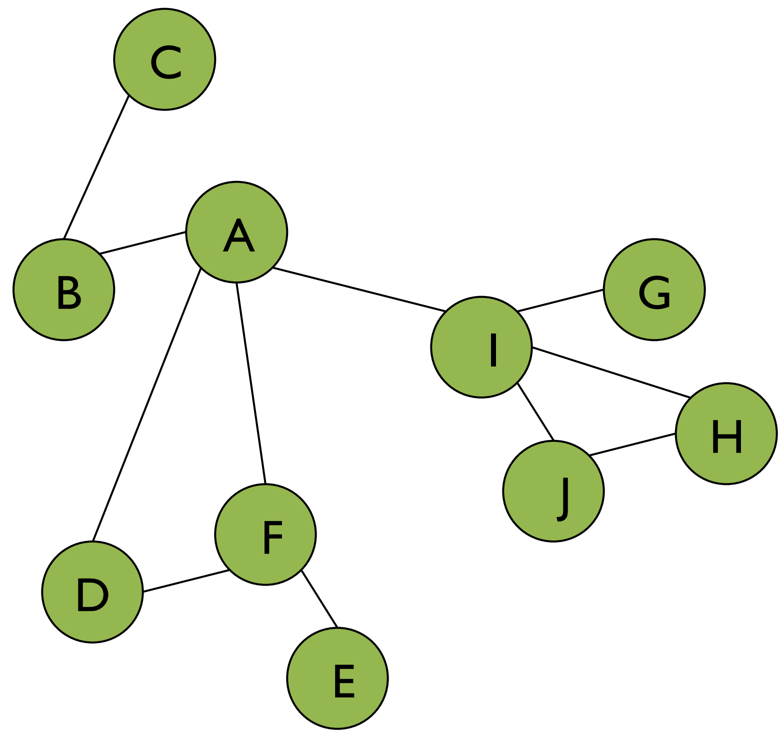

nips, 8 clusters
lineandballs, 3 clusters
fourclouds, 2 clusters
squiggles, 4 clusters
twocircles, 2 clusters
threecircles-joined, 2 clusters

(a) (b) (c)

(d) (e) (f)

Spectral Clustering: 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	1
I							1	1		1
J								1	1	

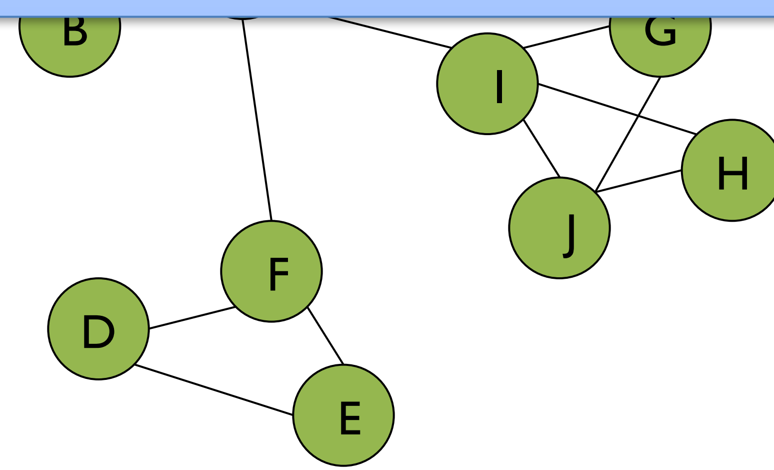


Spectral Clustering: Graph = Matrix
Transitively Closed Components = "Blocks"

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

sometimes called a **block-stochastic matrix:**

- each node has a latent "block"
- fixed probability qi for links between elements of block i
- fixed probability q0 for links between elements of different blocks



Of course we can't see the "blocks" unless the nodes are sorted by cluster…

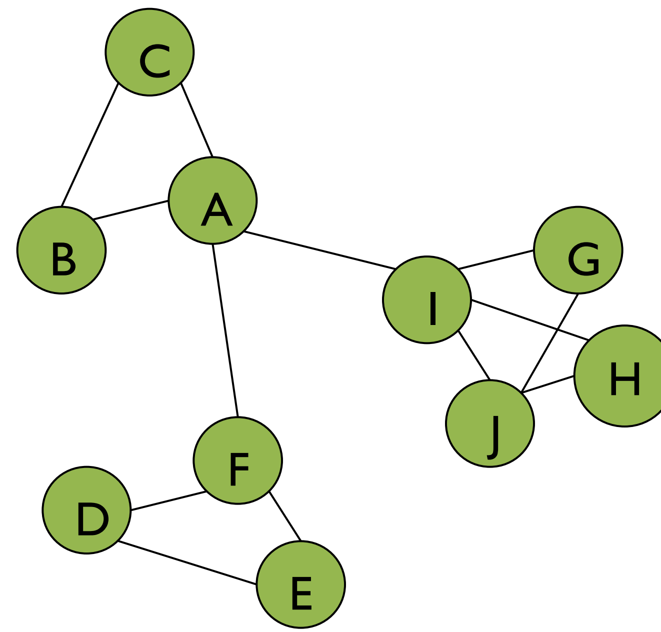
Spectral Clustering: Graph = Matrix
Vector = Node → Weight

M

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

v

	A
A	3
B	2
C	3
D	
E	
F	
G	
H	
I	
J	



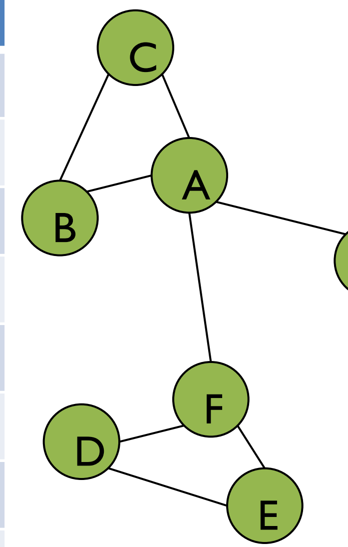
Spectral Clustering: Graph = Matrix
$M^*v_1 = v_2$ "propogates weights from neighbors"

$$M \ * \ v_1 = v_2$$

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

A	3
B	2
C	3
D	
E	
F	
G	
H	
I	
J	

A	2*1+3*1+0*1
B	3*1+3*1
C	3*1+2*1
D	
E	
F	
G	
H	
I	
J	



M

Spectral Clustering: Graph = Matrix
$W*v_1 = v_2$ "propogates weights from neighbors"

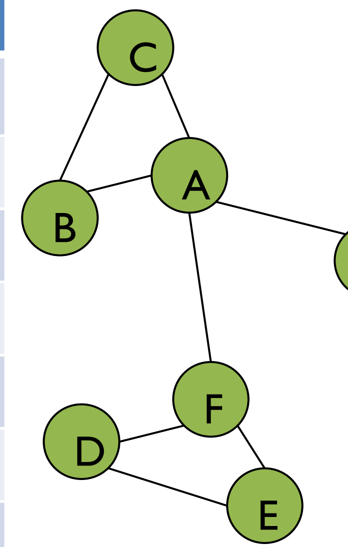
W: normalized so columns sum to 1

$$W * v_1 = v_2$$

	A	B	C	D	E	F	G	H	I	J
A	_	.5	.5			.3				
B	.3	_	.5							
C	.3	.5	_							
D				_	.5	.3				
E				.5	_	.3				
F	.3			.5	.5	_				
G							_		.3	.3
H								_	.3	.3
I							.5	.5	_	.3
J							.5	.5	.3	_

A	3
B	2
C	3
D	
E	
F	
G	
H	
I	
J	

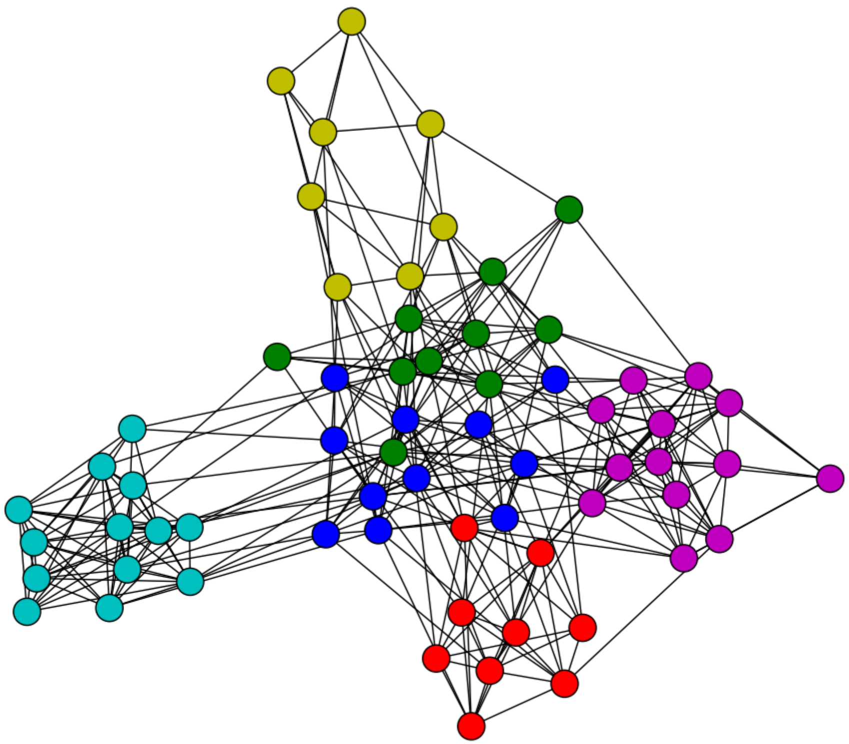
A	2*.5+3*.5+0*.3
B	3*.3+3*.5
C	3*.33+2*.5
D	
E	
F	
G	
H	
I	
J	



Spectral Clustering: Graph = Matrix
$W^*v_1 = v_2$ "propogates weights from neighbors"

$$\mathbf{W} \cdot \mathbf{v} = \lambda \mathbf{v} : \mathbf{v} \text{ is an eigenvector with eigenvalue } \lambda$$

Q: How do I pick **v** to be an eigenvector for a block-stochastic matrix?

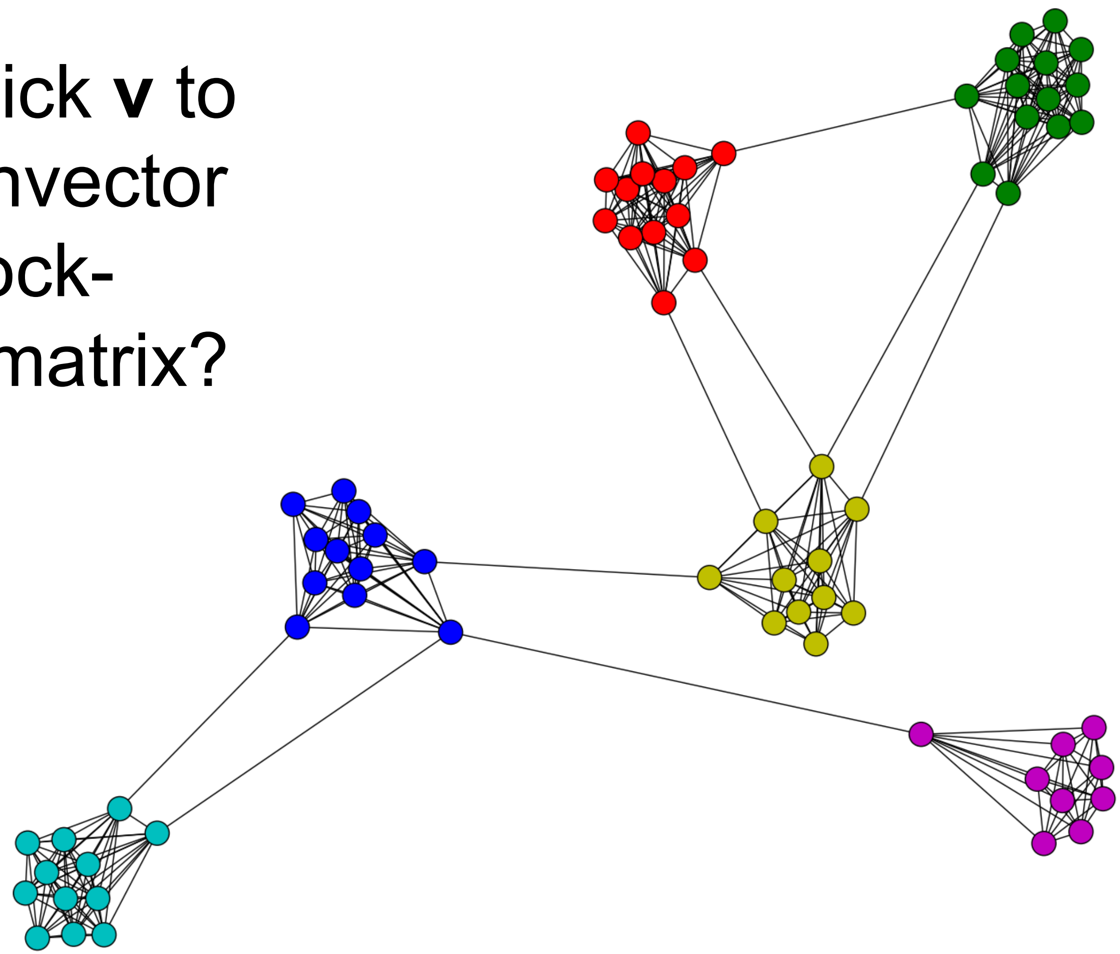


Spectral Clustering: Graph = Matrix

$W^*v_1 = v_2$ "propogates weights from neighbors"

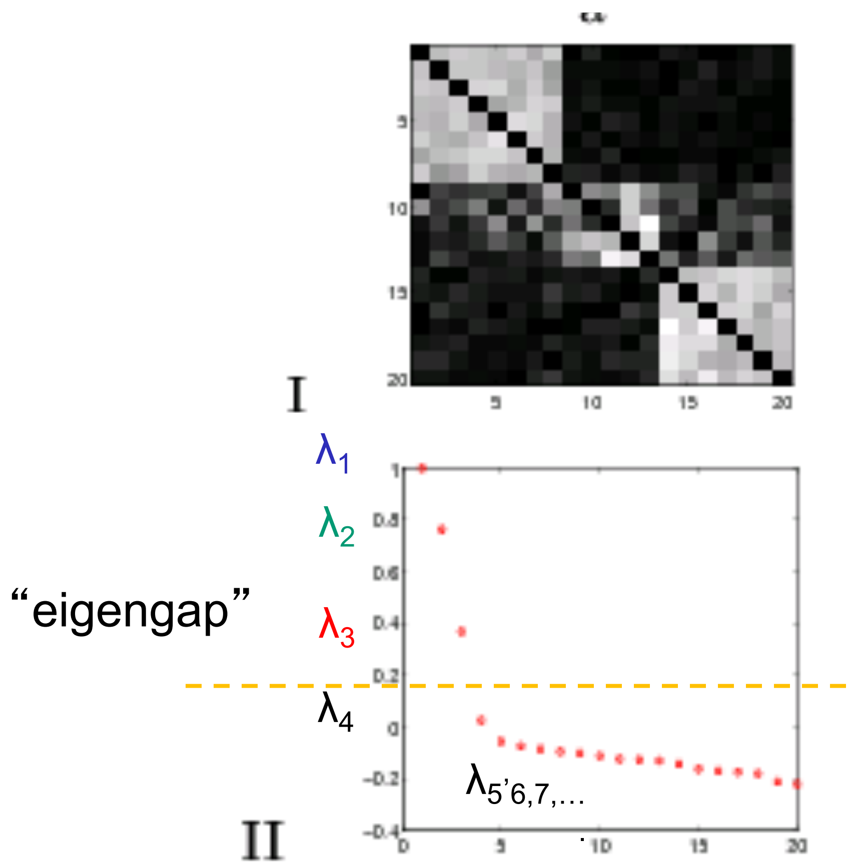
$$\mathbf{W} \cdot \mathbf{v} = \lambda \mathbf{v} : \mathbf{v} \text{ is an eigenvector with eigenvalue } \lambda$$

How do I pick **v** to be an eigenvector for a block-stochastic matrix?

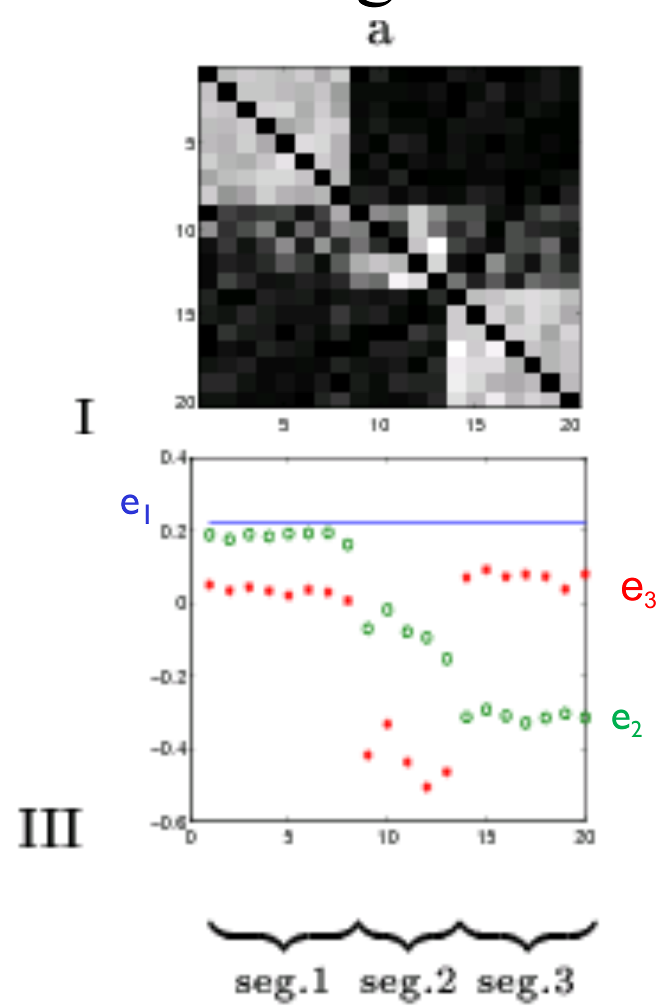


Spectral Clustering: Graph = Matrix
$W^*v_1 = v_2$ "propogates weights from neighbors"

$$\mathbf{W} \cdot \mathbf{v} = \lambda \mathbf{v} : \mathbf{v} \text{ is an eigenvector with eigenvalue } \lambda$$

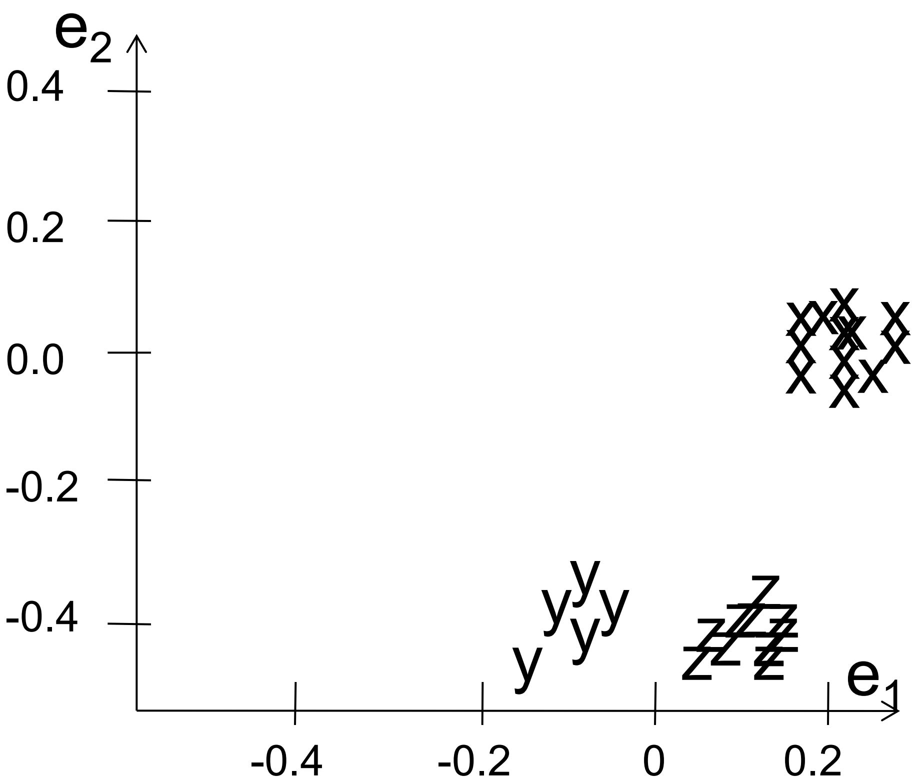


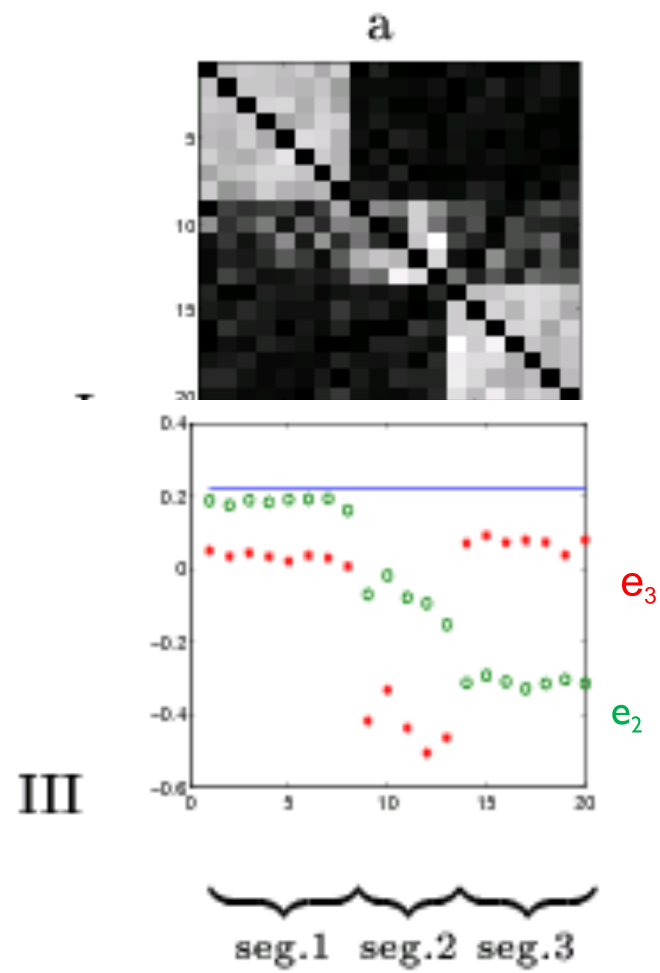
[Shi & Meila, 2002]



Spectral Clustering: Graph = Matrix
$W^*v_1 = v_2$ “propogates weights from neighbors”

$$\mathbf{W} \cdot \mathbf{v} = \lambda \mathbf{v} : \mathbf{v} \text{ is an eigenvector with eigenvalue } \lambda$$



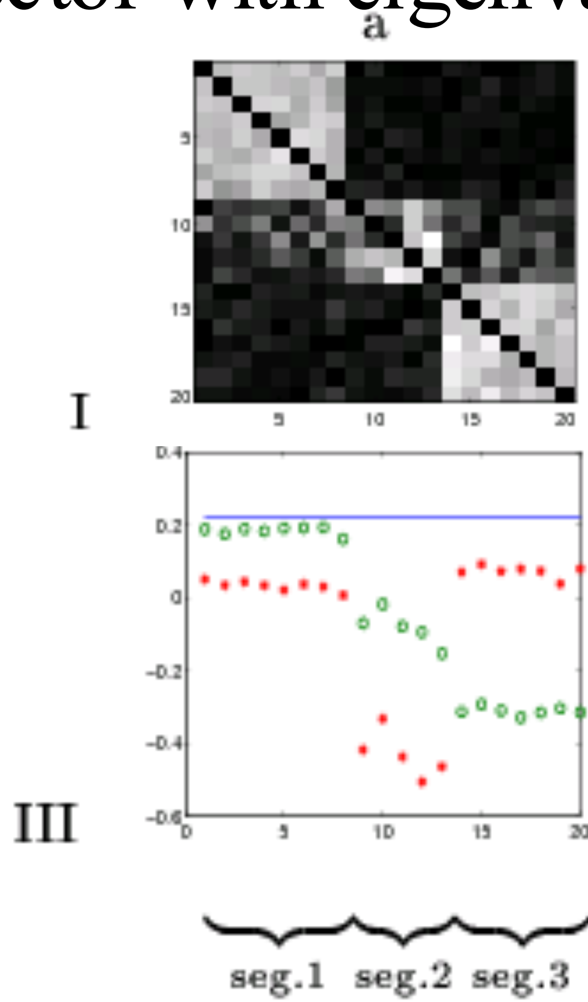


[Shi & Meila, 2002]

Spectral Clustering: Graph = Matrix
$W^*v_1 = v_2$ "propogates weights from neighbors"

$$\mathbf{W} \cdot \mathbf{v} = \lambda \mathbf{v} : \mathbf{v} \text{ is an eigenvector with eigenvalue } \lambda$$

If W is connected but roughly block diagonal with *k* blocks then
- the top eigenvector is a constant vector
- the next *k* eigenvectors are roughly piecewise constant with "pieces" corresponding to blocks



Spectral Clustering: Graph = Matrix
$W^*v_1 = v_2$ "propogates weights from neighbors"

$$\mathbf{W} \cdot \mathbf{v} = \lambda \mathbf{v} : \mathbf{v} \text{ is an eigenvector with eigenvalue } \lambda$$

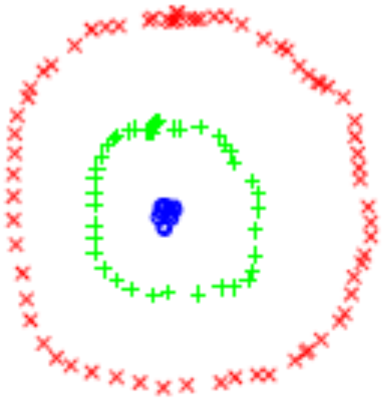
If **W** is connected but roughly block diagonal with *k* blocks then

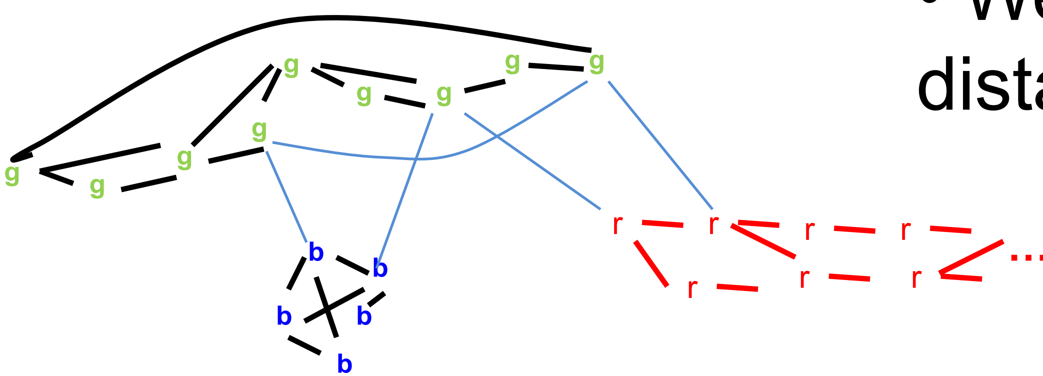
- the "top" eigenvector is a constant vector
- the next *k* eigenvectors are roughly piecewise constant with "pieces" corresponding to blocks

Spectral clustering:

- Find the top *k+1* eigenvectors $\mathbf{v}_1,\ldots,\mathbf{v}_{k+1}$
- Discard the "top" one
- Replace every node *a* with *k*-dimensional vector $x_a = \langle \mathbf{v}_2(a),\ldots,\mathbf{v}_{k+1}(a) \rangle$
- Cluster with *k*-means

Back to the 2-D examples





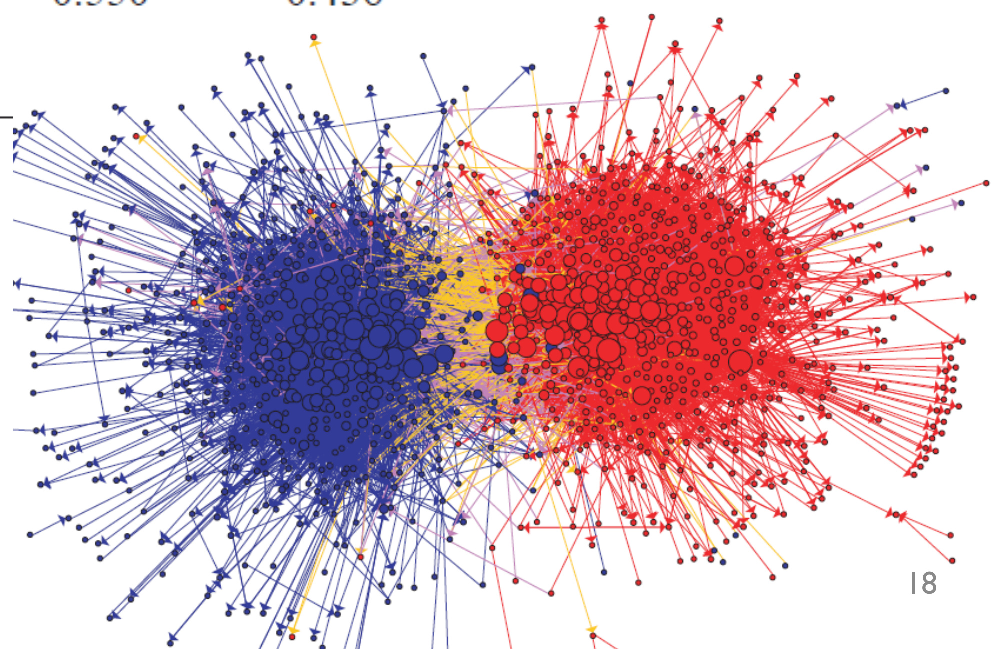
- Create a **graph**.
- Each 2D point is a vertex
- Add edges connecting all points in the 2-D initial space to all other points
- Weight of edge is distance

Spectral Clustering: Pros and Cons

- Elegant, and well-founded mathematically
- Tends to avoid local minima
  - Optimal solution to relaxed version of mincut problem (Normalized cut, aka NCut)
- Works quite well when relations are approximately transitive (like similarity, social connections)
- Expensive for very large datasets
  - Computing eigenvectors is the bottleneck
  - Approximate eigenvector computation not always useful
- Noisy datasets sometimes cause problems
  - Picking number of eigenvectors and k is tricky
  - "Informative" eigenvectors need not be in top few
  - Performance can drop suddenly from good to terrible

Experimental results:
best-case assignment of class labels to clusters

		NCut		NJW	
Dataset	**k**	**Accuracy**	**Macro-F1**	**Accuracy**	**Macro-F1**
Iris	3	0.673	0.570	0.807	0.806
PenDigits01	2	1.000	1.000	1.000	1.000
PenDigits17	2	0.755	0.753	0.755	0.754
UBMCBlog	2	0.953	0.953	0.953	0.953
AGBlog	2	0.520	0.342	0.520	0.342
20ngA	2	0.955	0.955	0.955	0.955
20ngB	2	0.505	0.344	0.550	0.436
20ngC	3	0.613	0.621		
20ngD	4	0.469	0.432		
Average	-	0.716	0.663		



Spectral clustering as Optimization

Spectral Clustering: Graph = Matrix

$W^*v_1 = v_2$ "propogates weights from neighbors"

$$\mathbf{W} \cdot \mathbf{v} = \lambda \mathbf{v} : \mathbf{v} \text{ is an eigenvector with eigenvalue } \lambda$$

- A = adjacency matrix
- D = diagonal normalizer for A (# outlinks)
- $W = A D^{-1}$
- smallest eigenvecs of D-A are largest eigenvecs of A
- smallest eigenvecs of I-W are largest eigenvecs of W

Suppose each *y(i)=+1* or *-1*:

- Then *y* is a cluster indicator that splits the nodes into two
- what is $\mathbf{y}^T(D-A)\mathbf{y}$?

$$\mathbf{y}^T(D-A)\mathbf{y} = \mathbf{y}^T D\mathbf{y} - \mathbf{y}^T A\mathbf{y} = \sum_i d_i y_i^2 - \sum_{i,j} a_{i,j} y_i y_j$$

$$= \frac{1}{2}\left[2\sum_i d_i y_i^2 - 2\sum_{i,j} a_{i,j} y_i y_j\right]$$

$$= \frac{1}{2}\left[\sum_i \left(\sum_j a_{ij}\right) y_i^2 + \sum_j \left(\sum_i a_{ij}\right) y_j^2 - 2\sum_{i,j} a_{i,j} y_i y_j\right]$$

$$= \frac{1}{2}\left[\sum_{i,j} a_{ij} y_i^2 + \sum_{i,j} a_{ij} y_j^2 - 2\sum_{i,j} a_{i,j} y_i y_j\right]$$

$$= \frac{1}{2}\left[\sum_{i,j} a_{i,j}(y_i - y_j)^2\right]$$

= size of CUT(**y**)

Same as smoothness criterion used in SSL but now there are no seeds

$$\mathbf{y}^T(I-W)\mathbf{y} = \text{size of NCUT}(\mathbf{y})$$

NCUT: roughly minimize ratio of transitions between classes vs transitions within classes

Spectral Clustering: Graph = Matrix

$W^*v_1 = v_2$ "propogates weights from neighbors"

$$\mathbf{W} \cdot \mathbf{v} = \lambda \mathbf{v} : \mathbf{v} \text{ is an eigenvector with eigenvalue } \lambda$$

- smallest eigenvecs of D-A are largest eigenvecs of A
- smallest eigenvecs of I-W are largest eigenvecs of W

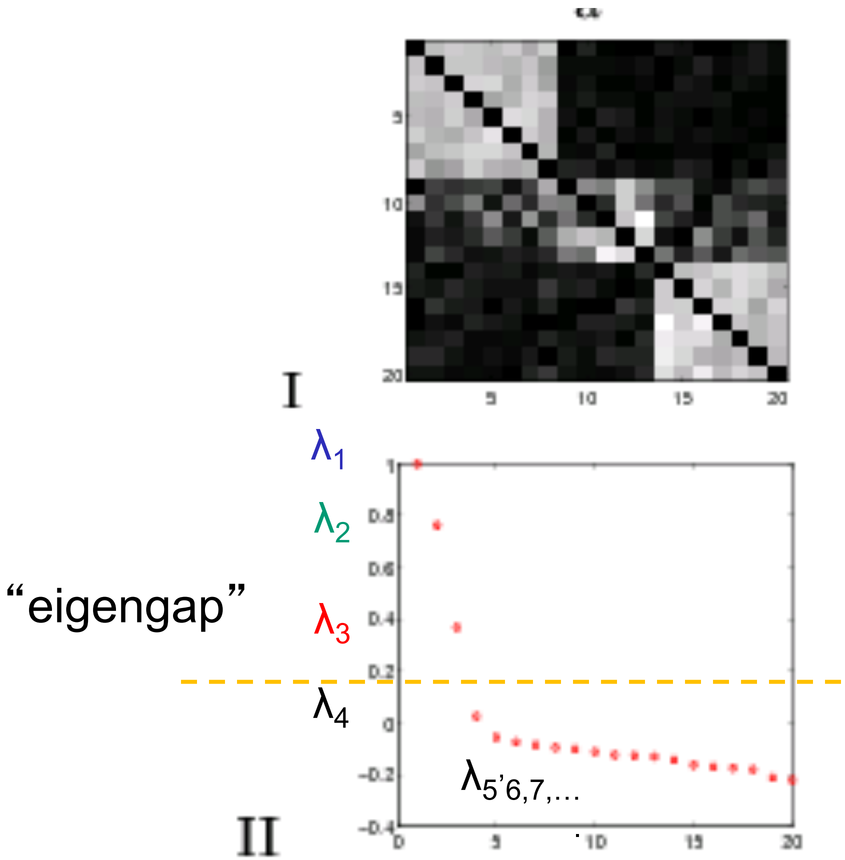
Suppose each *y(i)=+1* or *-1*:

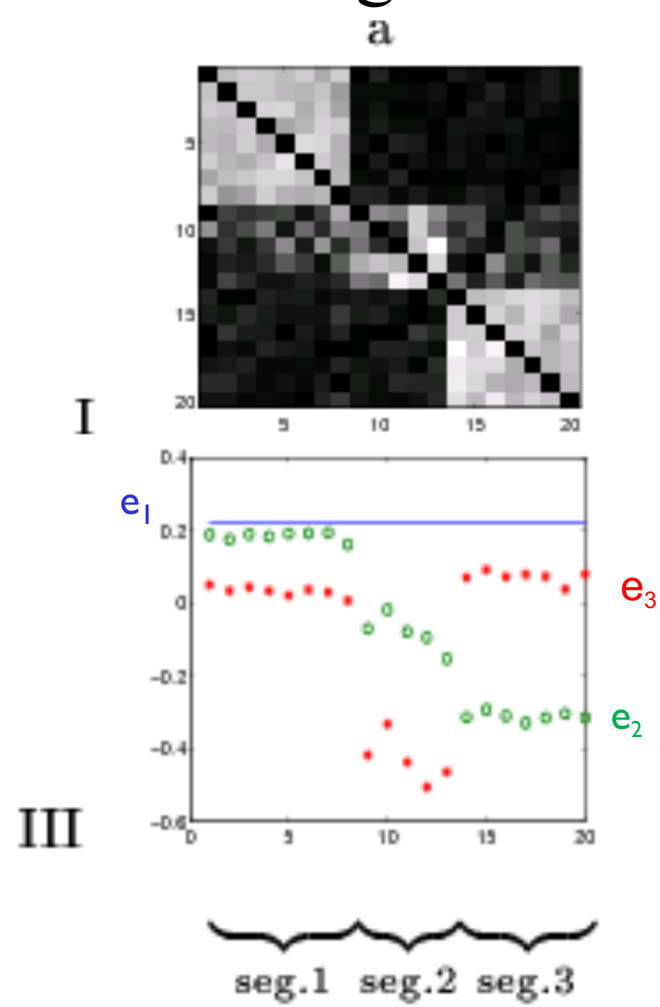
- Then *y* is a cluster indicator that cuts the nodes into two
- what is $\mathbf{y}^T(D-A)\mathbf{y}$? <u>The cost of the graph cut defined by **y**</u>
- what is $\mathbf{y}^T(I-W)\mathbf{y}$? <u>Also a cost of a graph cut defined by **y**</u>
- How to minimize it?
    - Turns out: to minimize $\mathbf{y}^T X \mathbf{y} / (\mathbf{y}^T\mathbf{y})$ find *smallest* eigenvector of X
    - But: this will not be +1/-1, so it's a "relaxed" solution

Spectral Clustering: Graph = Matrix

$W^*v_1 = v_2$ "propogates weights from neighbors"

$$\mathbf{W}\cdot\mathbf{v} = \lambda\mathbf{v} : \mathbf{v} \text{ is an eigenvector with eigenvalue } \lambda$$





[Shi & Meila, 2002]

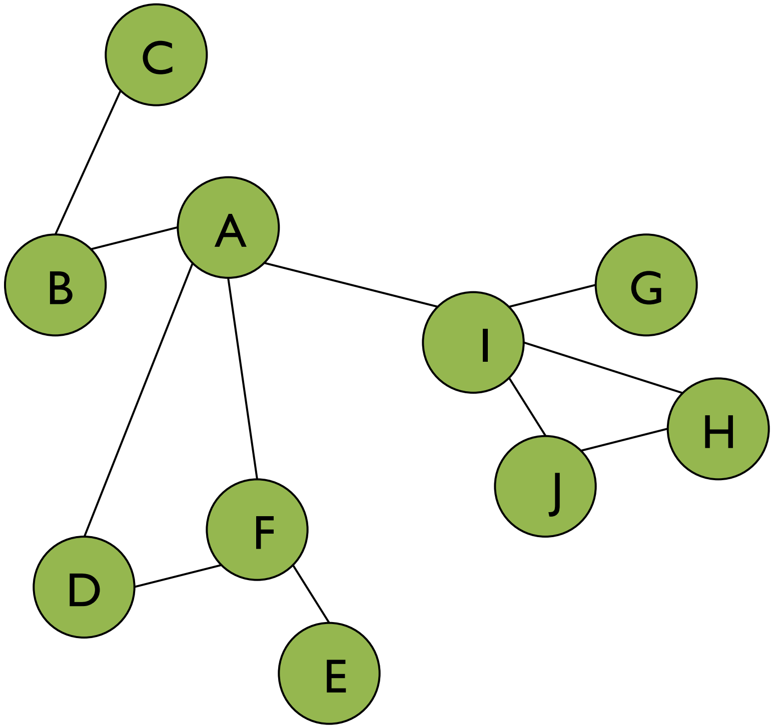
Another way to think about spectral clustering….

- Most normal people think about spectral clustering like that - as relaxed optimization
- …me, not so much
- I like the connection to "averaging"…because…. it leads to a nice fast variation of spectral clustering called *power iteration clustering*

Unsupervised Learning on Graphs with Power Iteration Clustering

Spectral Clustering: 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	1
I							1	1		1
J								1	1	

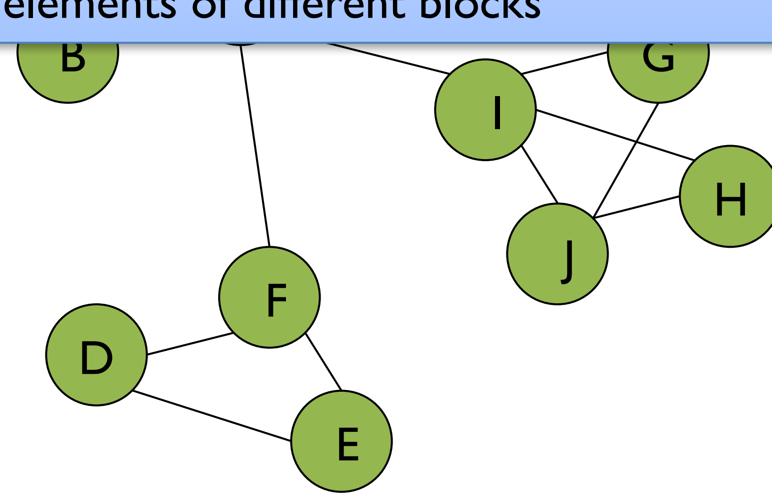


Spectral Clustering: Graph = Matrix
Transitively Closed Components = "Blocks"

sometimes called a **block-stochastic matrix:**

- each node has a latent "block"
- fixed probability qi for links between elements of block i
- fixed probability q0 for links between elements of different blocks

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



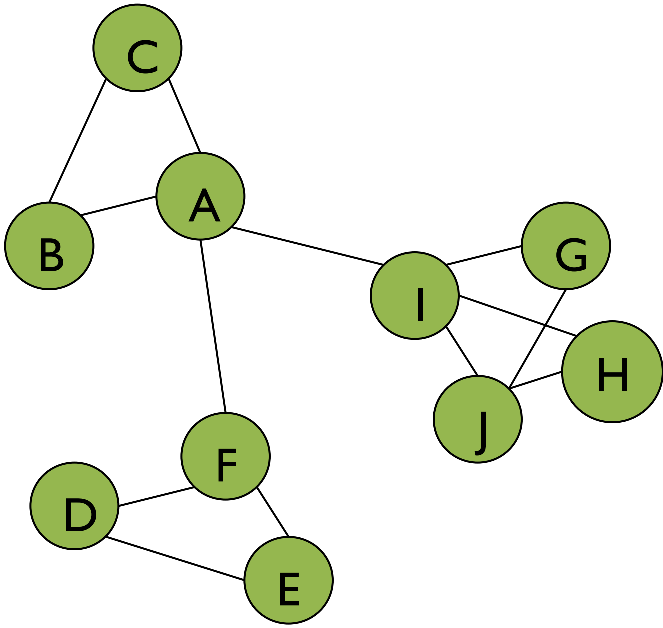
Of course we can't see the "blocks" unless the nodes are sorted by cluster…

Spectral Clustering: Graph = Matrix
Vector = Node → Weight

M **v**

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

	A
A	3
B	2
C	3
D	
E	
F	
G	
H	
I	
J	



M

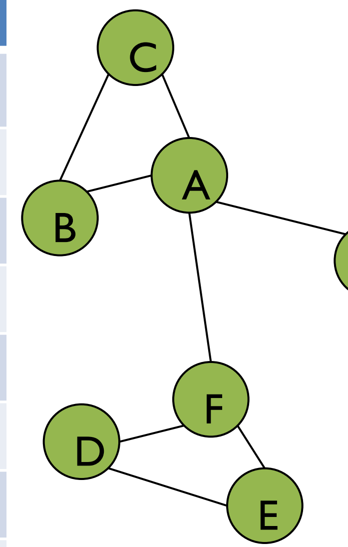
Spectral Clustering: Graph = Matrix
$M*v_1 = v_2$ “propogates weights from neighbors”

$$M \quad * \quad v_1 \quad = \quad v_2$$

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

A	3
B	2
C	3
D	
E	
F	
G	
H	
I	
J	

A	2*1+3*1+0*1
B	3*1+3*1
C	3*1+2*1
D	
E	
F	
G	
H	
I	
J	



Spectral Clustering: Graph = Matrix
$W^*v_1 = v_2$ "propogates weights from neighbors"

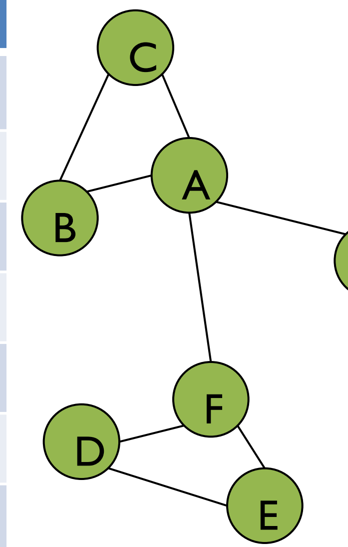
W: normalized so columns sum to 1

$$W \; * \; v_1 \; = \; v_2$$

	A	B	C	D	E	F	G	H	I	J
A	_	.5	.5			.3				
B	.3	_	.5							
C	.3	.5	_							
D				_	.5	.3				
E				.5	_	.3				
F	.3			.5	.5	_				
G							_		.3	.3
H								_	.3	.3
I							.5	.5	_	.3
J							.5	.5	.3	_

A	3
B	2
C	3
D	
E	
F	
G	
H	
I	
J	

A	2*.5+3*.5+0*.3
B	3*.3+3*.5
C	3*.33+2*.5
D	
E	
F	
G	
H	
I	
J	

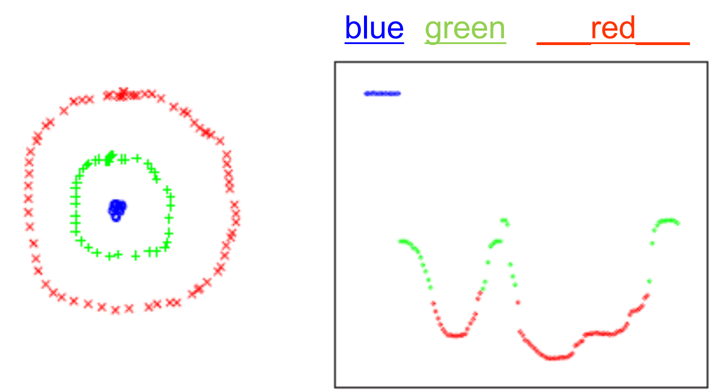


Repeated averaging with neighbors as a clustering method

- Pick a vector v^0 (maybe at random)
- Compute $v^1 = Wv^0$
  - i.e., replace $v^0[x]$ with *weighted average* of $v^0[y]$ for the neighbors y of x
- Plot $v^1[x]$ for each x
- Repeat for v^2, v^3, …

- Variants widely used for *semi-supervised* learning
  - HF/CoEM/wvRN - average + clamping of labels for nodes with known labels
- Without clamping, will converge to constant v^t
- What are the *dynamics* of this process?

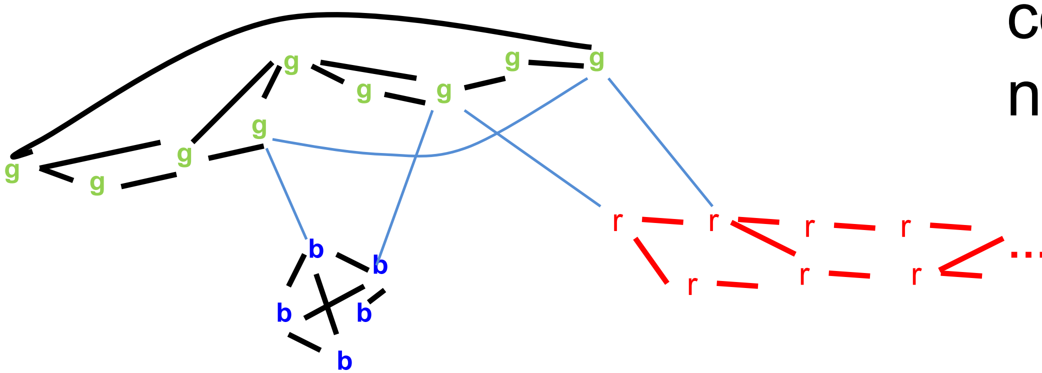
Repeated averaging with neighbors on a sample problem…



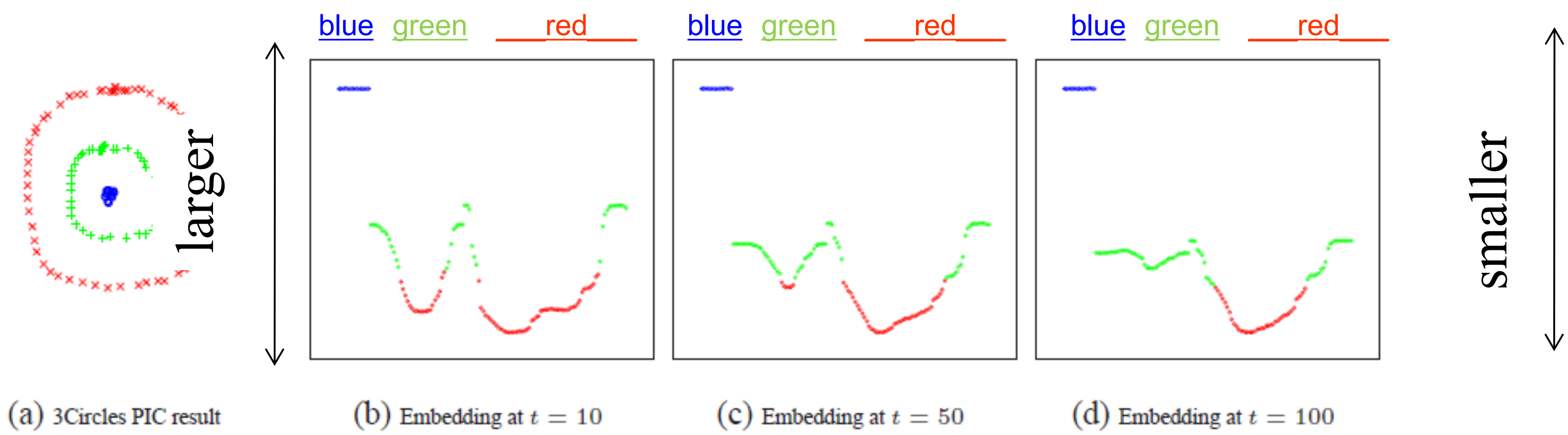
(a) 3Circles PIC result

(b) Embedding at $t = 10$

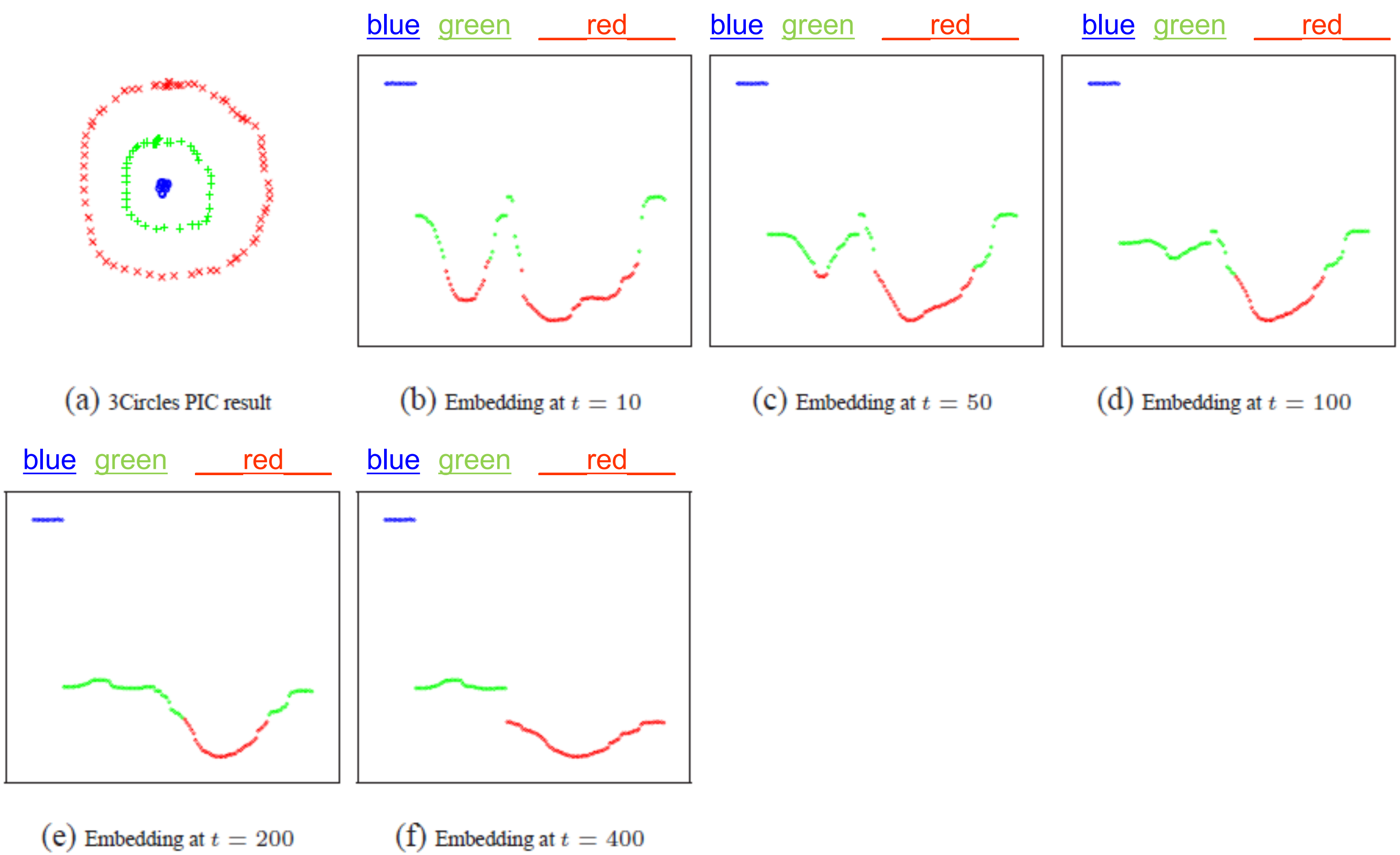
- Create a graph, connecting all points in the 2-D initial space to all other points
  - Weighted by distance
- Run power iteration (averaging) for 10 steps
- Plot node id x vs $v^{10}(x)$
  - nodes are ordered and colored by actual cluster number



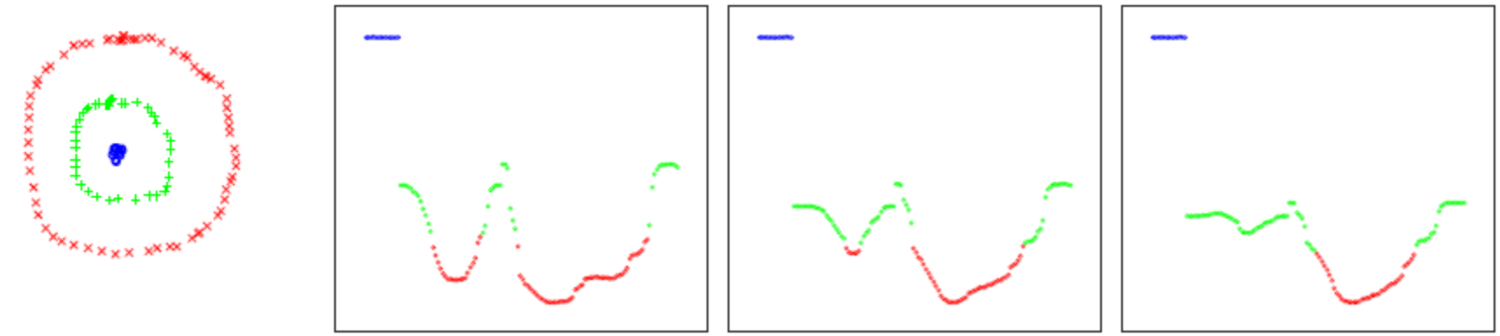
Repeated averaging with neighbors on a sample problem…



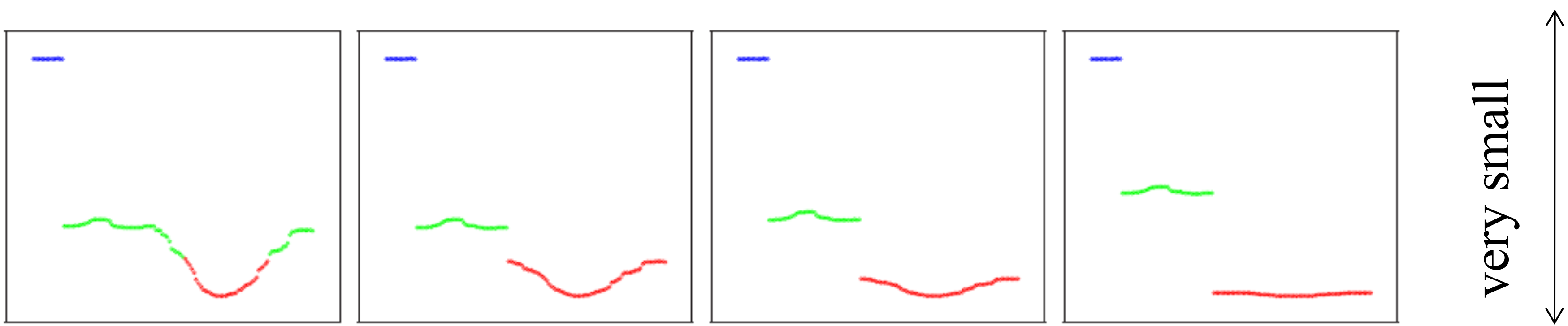
Repeated averaging with neighbors on a sample problem…



Repeated averaging with neighbors on a sample problem…



(a) 3Circles PIC result (b) Embedding at $t = 10$ (c) Embedding at $t = 50$ (d) Embedding at $t = 100$



(e) Embedding at $t = 200$ (f) Embedding at $t = 400$ (g) Embedding at $t = 600$ (h) Embedding at $t = 1000$

PIC: Power Iteration Clustering

run power iteration (repeated averaging w/ neighbors) with early stopping

1. Pick an initial vector $\mathbf{v^0}$.
2. Set $\mathbf{v^{t+1}} \leftarrow \frac{W\mathbf{v^t}}{\|W\mathbf{v^t}\|_1}$ and $\delta^{t+1} \leftarrow |\mathbf{v^{t+1}} - \mathbf{v^t}|$.
3. Increment t and repeat above step until $|\delta^t - \delta^{t-1}| \simeq \mathbf{0}$.
4. Use k-means to cluster points on $\mathbf{v^t}$ and return clusters $C_1, C_2, ..., C_k$.

- V^0: random start, or "degree matrix" D, or …
- Easy to implement and efficient
- Very easily parallelized

- Experimentally, often **better than** traditional spectral methods

- Surprising since the embedded space is 1-dimensional!

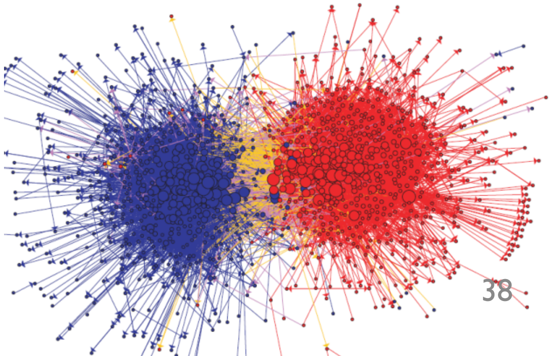
Experiments

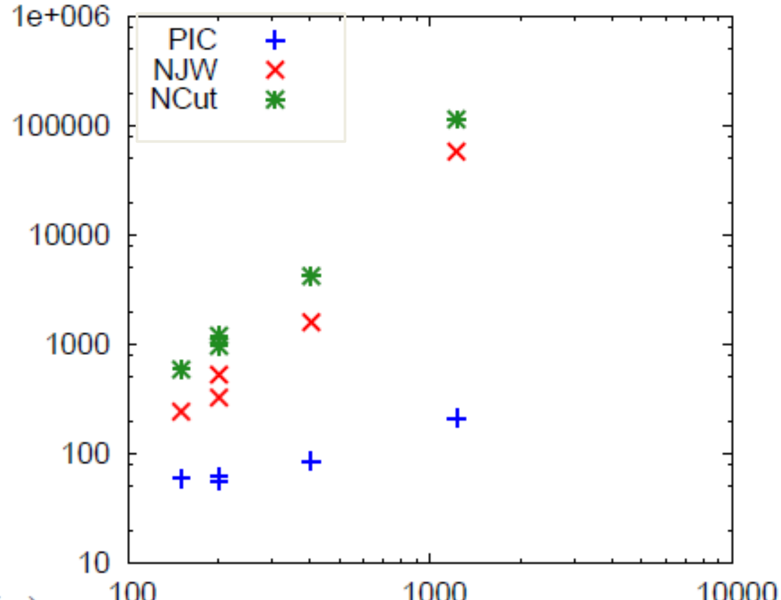
- "Network" problems: natural graph structure
  - PolBooks: 105 political books, 3 classes, linked by copurchaser
  - UMBCBlog: 404 political blogs, 2 classes, blogroll links
  - AGBlog: 1222 political blogs, 2 classes, blogroll links
- "Manifold" problems: cosine distance between classification instances
  - Iris: 150 flowers, 3 classes
  - PenDigits01,17: 200 handwritten digits, 2 classes (0-1 or 1-7)
  - 20ngA: 200 docs, misc.forsale vs soc.religion.christian
  - 20ngB: 400 docs, misc.forsale vs soc.religion.christian
  - 20ngC: 20ngB + 200 docs from talk.politics.guns
  - 20ngD: 20ngC + 200 docs from rec.sport.baseball

Experimental results:
best-case assignment of class labels to clusters

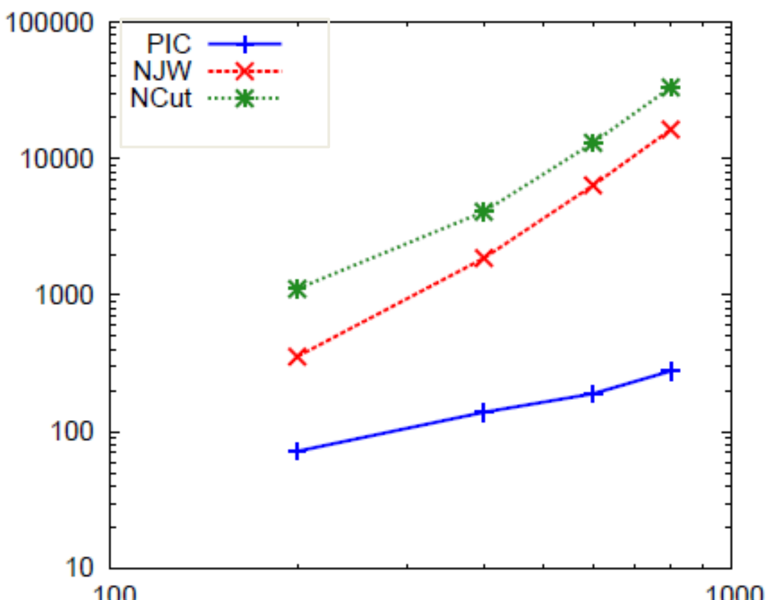
Dataset	k	NCut Accuracy	NCut Macro-F1	NJW Accuracy	NJW Macro-F1	PIC Accuracy	PIC Macro-F1
Iris	3	0.673	0.570	0.807	0.806	0.980	0.980
PenDigits01	2	1.000	1.000	1.000	1.000	1.000	1.000
PenDigits17	2	0.755	0.753	0.755	0.754	0.755	0.753
UBMCBlog	2	0.953	0.953	0.953	0.953	0.948	0.948
AGBlog	2	0.520	0.342	0.520	0.342	0.957	0.957
20ngA	2	0.955	0.955	0.955	0.955	0.960	0.960
20ngB	2	0.505	0.344	0.550	0.436	0.905	0.904
20ngC	3	0.613	0.621	0.635	0.639	0.737	0.730
20ngD	4	0.469	0.432	0.535	0.534	0.580	0.570
Average	-	0.716	0.663	0.746	0.713	0.869	0.867

Table 1: Clustering performance of PIC and spectral clustering algorithms on several real datasets.

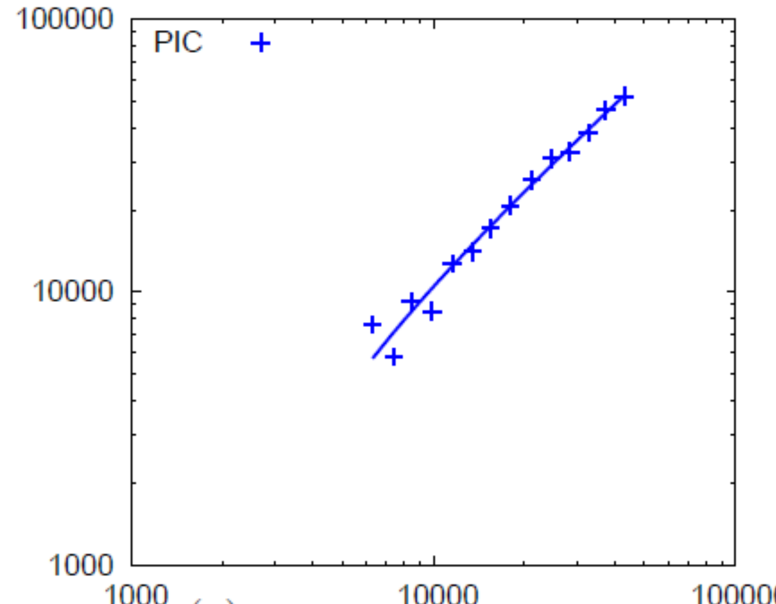




(a) Iris, PenDigits, UBMCBlog & AGBlog



(b) 20 Newsgroups



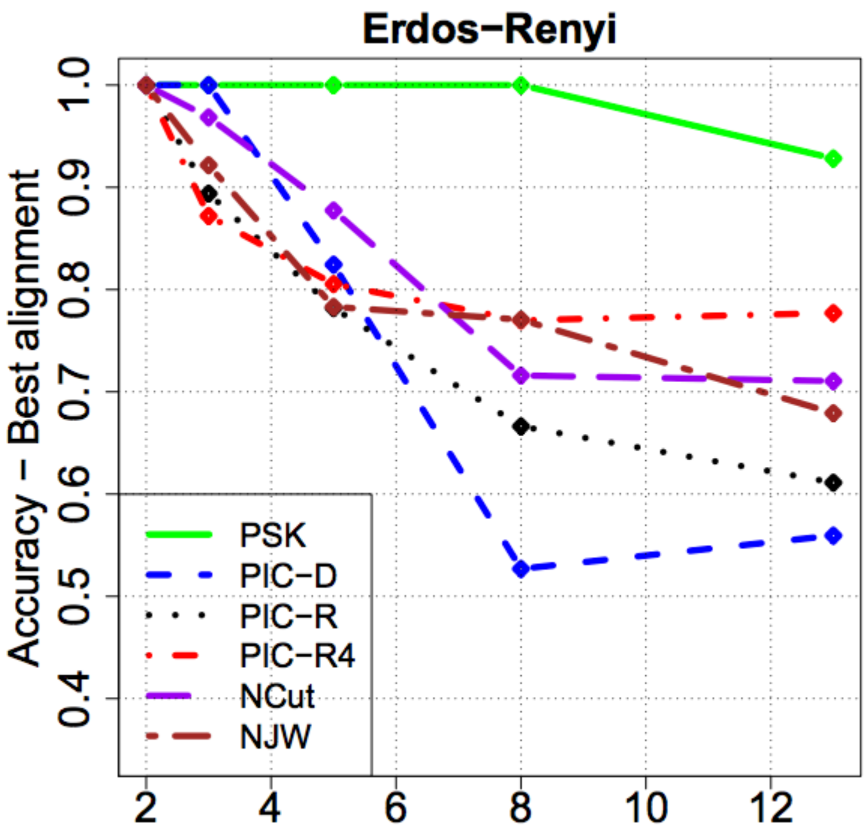
(c) Gene Citation

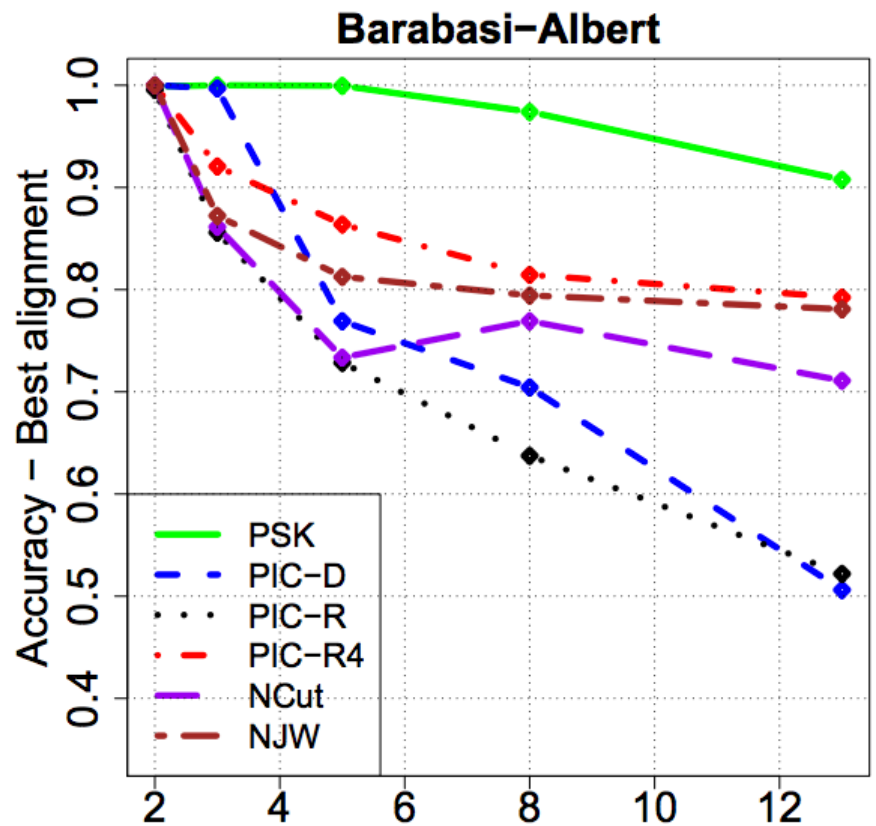
Experiments: run time and scalability

Dataset	Size	NCut Runtime	NJW Runtime	PIC Runtime	PIC Iterations
Iris	150	589	242	59	6
PenDigits01	200	965	326	56	6
PenDigits17	200	1197	528	62	6
UBMCBlog	404	4205	1589	85	21
AGBlog	1222	114821	58145	211	34
20ngA	200	1113	355	72	15
20ngB	400	4085	1864	139	13
20ngC	600	13070	6383	190	13
20ngD	800	33191	16295	278	11

Time in millisec

More experiments





Varying the number of clusters for PIC and PIC4 (starts with random 4-d point rather than a random 1-d point).

More experiments

Table 1: Dataset Statistics (N/E/C indicates Nodes / Edges / Clusters)

(a) Social network

Dataset	N/E/C	Dataset	N/E/C
karate	34 / 156 / 2	umbc	404 / 4764 / 2
polbooks	105 / 882 / 3	mgemail	280 / 1344 / 55
dolphin	62 / 318 / 2	citeseer	2114 / 7396 / 6
football	115 / 1226 / 10	cora	2485 / 10138 / 7
msp	4324 / 37254 / 2		
ag	1222 / 33428 / 2		
senate	98 / 9506 / 2		

(b) Author disambiguation

Dataset	N/E/C	Dataset	N/E/C
jsmith	4120 / 21452 / 30	jrobinson	686 / 2846 / 12
akumar	801 / 2476 / 14	ktanaka	827 / 2758 / 10
cchen	424 / 1558 / 16	mbrown	579 / 2112 / 13
djohnson	1381 / 5344 / 15	mmiller	2106 / 9918 / 12
jmartin	424 / 1558 / 16	jlee	5820 / 23110 / 100
agupta	2485 /10208 / 26	ychen	5472 / 25584 / 71
mjones	961 / 3450 / 13	slee	5963 / 23086 / 86

More “real” network datasets from various domains

(c) Best alignment: Social networks

Dataset	PSK	PIC_D	PIC_R	PIC_{R4}	NCut	NJW
Karate	**1.00**	0.91	0.93	0.95	0.95	0.95
Dolphin	0.90	**0.98**	0.98	**0.98**	**0.98**	**0.98**
UMBC	0.95	0.93	0.95	0.95	0.95	**0.96**
AG	**0.95**	0.91	0.94	0.94	0.52	0.51
MSP	**0.88**	0.63	0.63	0.63	0.63	0.64
Senate	0.98	**0.99**	**0.99**	**0.99**	**0.99**	**0.99**
PolBook	0.78	0.80	0.81	**0.83**	0.82	0.80
Football	**0.76**	0.47	0.51	0.66	0.72	0.67
MGEmail	0.28	0.39	0.40	**0.64**	0.59	0.56
CiteSeer	0.33	0.51	0.48	**0.55**	0.48	0.52
Cora	**0.47**	0.46	0.40	0.45	0.29	0.42
Average	0.75	0.73	0.73	**0.78**	0.72	0.73

(d) Best alignment: Author disambiguation

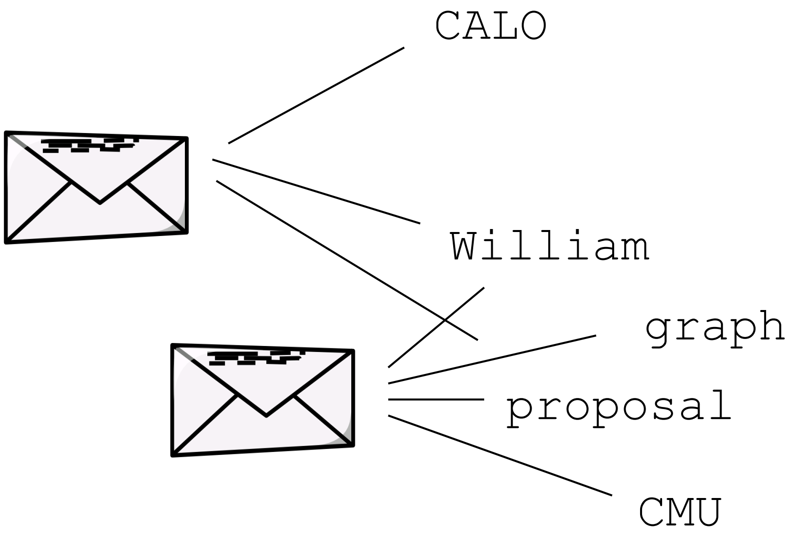
Dataset	PSK	PIC_D	PIC_R	PIC_{R4}	NCut	NJW
AGupta	0.13	0.26	0.24	**0.37**	0.26	0.34
AKumar	0.20	0.29	0.31	0.37	0.35	**0.40**
CChen	0.30	0.43	0.44	**0.53**	0.24	0.50
DJohnson	0.15	0.24	0.33	0.46	**0.47**	0.35
JLee	0.11	0.20	0.23	**0.41**	0.17	0.39
JMartin	0.28	0.42	0.43	**0.53**	0.25	0.49
JRobinson	0.26	0.37	0.42	**0.49**	0.26	0.48
JSmith	0.11	0.22	0.21	0.41	0.31	**0.42**
KTanaka	0.19	0.36	0.41	0.45	**0.45**	0.43
MBrown	0.21	0.35	0.41	**0.52**	0.47	0.50
MJones	0.19	0.29	0.34	0.38	**0.38**	0.35
MMiller	0.14	0.30	0.41	0.52	0.52	**0.53**
SLee	0.08	0.19	0.23	**0.41**	0.23	0.39
YChen	0.10	0.23	0.28	**0.47**	0.23	0.46
Average	0.18	0.30	0.34	**0.45**	0.33	0.43

LEARNING ON GRAPHS FOR NON-GRAPH DATASETS

Why I'm talking about graphs

- Lots of large data *is* graphs
  - Facebook, Twitter, citation data, and other *social* networks
  - The web, the blogosphere, the semantic web, Freebase, Wikipedia, Twitter, and other *information* networks
  - Text corpora (like RCV1), large datasets with discrete feature values, and other *bipartite* networks
    - nodes = documents or words
    - links connect document → word or word → document
  - Computer networks, biological networks (proteins, ecosystems, brains, …), …
  - Heterogeneous networks with multiple types of nodes
    - people, groups, documents

Simplest Case: Bi-partite Graphs



Motivation: Experimental Datasets `Used for PIC are…

- "Network" problems: natural graph structure
  - PolBooks: 105 political books, 3 classes, linked by copurchaser
  - UMBCBlog: 404 political blogs, 2 classes, blogroll links
  - AGBlog: 1222 political blogs, 2 classes, blogroll links
  - Also: Zachary's karate club, citation networks, ...
- "Manifold" problems: cosine distance between **all pairs** of classification instances

  Gets expensive fast

  - Iris: 150 flowers, 3 classes
  - PenDigits01,17: 200 handwritten digits, 2 classes (0-1 or 1-7)
  - 20ngA: 200 docs, misc.forsale vs soc.religion.christian
  - 20ngB: 400 docs, misc.forsale vs soc.religion.christian
  - …

Spectral Clustering: Graph = Matrix

$A^*v_1 = v_2$ “propogates weights from neighbors”

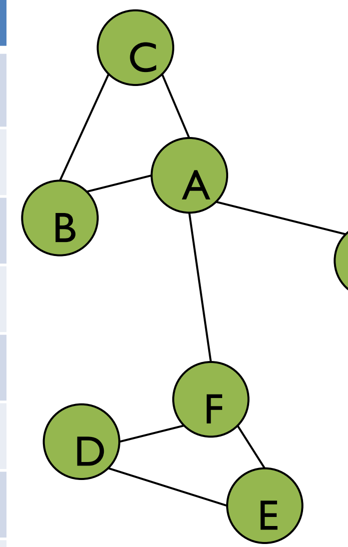
$$A \;*\; v_1 = v_2$$

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

M

A	3
B	2
C	3
D	
E	
F	
G	
H	
I	
J	

A	2*1+3*1+0*1
B	3*1+3*1
C	3*1+2*1
D	
E	
F	
G	
H	
I	
J	



Spectral Clustering: Graph = Matrix
$W^*v_1 = v_2$ "propogates weights from neighbors"

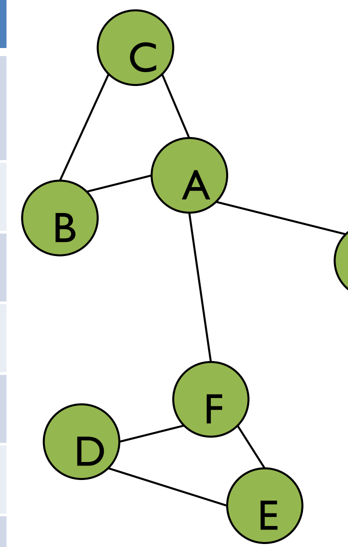
W: normalized so columns sum to 1

$$W \ * \ v_1 \ = \ v_2$$

	A	B	C	D	E	F	G	H	I	J
A	_	.5	.5			.3				
B	.3	_	.5							
C	.3	.5	_							
D				_	.5	.3				
E				.5	_	.3				
F	.3			.5	.5	_				
G							_		.3	.3
H								_	.3	.3
I							.5	.5	_	.3
J							.5	.5	.3	_

A	3
B	2
C	3
D	
E	
F	
G	
H	
I	
J	

A	2*.5+3*.5+0*.3
B	3*.3+3*.5
C	3*.33+2*.5
D	
E	
F	
G	
H	
I	
J	



$$W = D^{-1} {}^* A$$

$$D[i,i]=1/degree(i)$$

Lazy computation of distances and normalizers

- Recall PIC's update is
  - $v^t = W * v^{t-1} = = D^{-1}A * v^{t-1}$
  - ...where D is the [diagonal] degree matrix: $D=A*\mathbf{1}$
- My favorite distance metric for text is length-normalized TFIDF:
  - Def'n: $A(i,j)=\langle v_i,v_j\rangle/||v_i||*||v_j||$
  - Let $N(i,i)=||v_i||$ ... and $N(i,j)=0$ for $i!=j$
  - Let $F(i,k)$=TFIDF weight of word w_k in document v_i
  - Then: $A = N^{-1}F^TFN^{-1}$

1 is a column vector of 1's

$\langle u,v\rangle$=inner product
$||u||$ is L2-norm

Lazy computation of distances and normalizers

Equivalent to using TFIDF/cosine on all pairs of examples but requires only *sparse* matrices

- 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 $A = N^{-1}F^TFN^{-1}$
  - Let $D(i,i)= N^{-1}F^TFN^{-1}*\mathbf{1}$ where $\mathbf{1}$ is an all-1's vector
    - Computed as $D=N^{-1}(F^T(F(N^{-1}*\mathbf{1})))$ for efficiency
  - New update:
    - $v^t = D^{-1}A * v^{t-1} = D^{-1} N^{-1}F^TFN^{-1} *v^{t-1}$

Experimental results

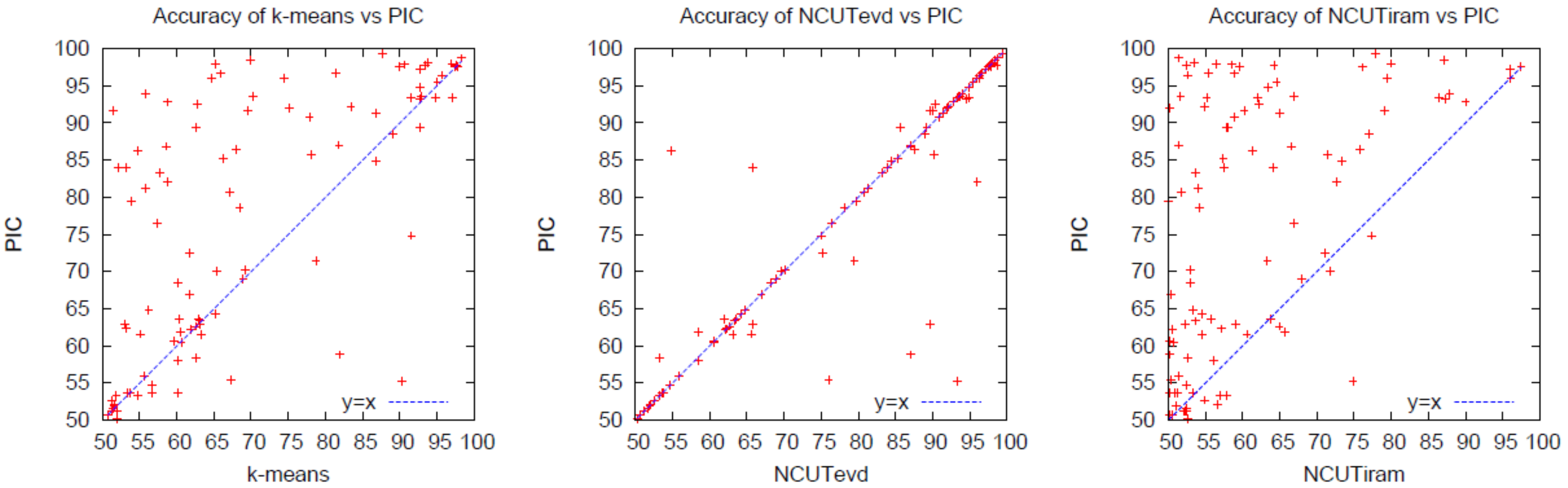
- RCV1 text classification dataset
  - 800k + newswire stories
  - Category labels from *industry* vocabulary
  - Took single-label documents and categories with at least 500 instances
  - Result: 193,844 documents, 103 categories
- Generated 100 random category pairs
  - Each is all documents from two categories
  - Range in size and difficulty
  - Pick category 1, with m_1 examples
  - Pick category 2 such that $0.5m_1 < m_2 < 2m_1$

Results

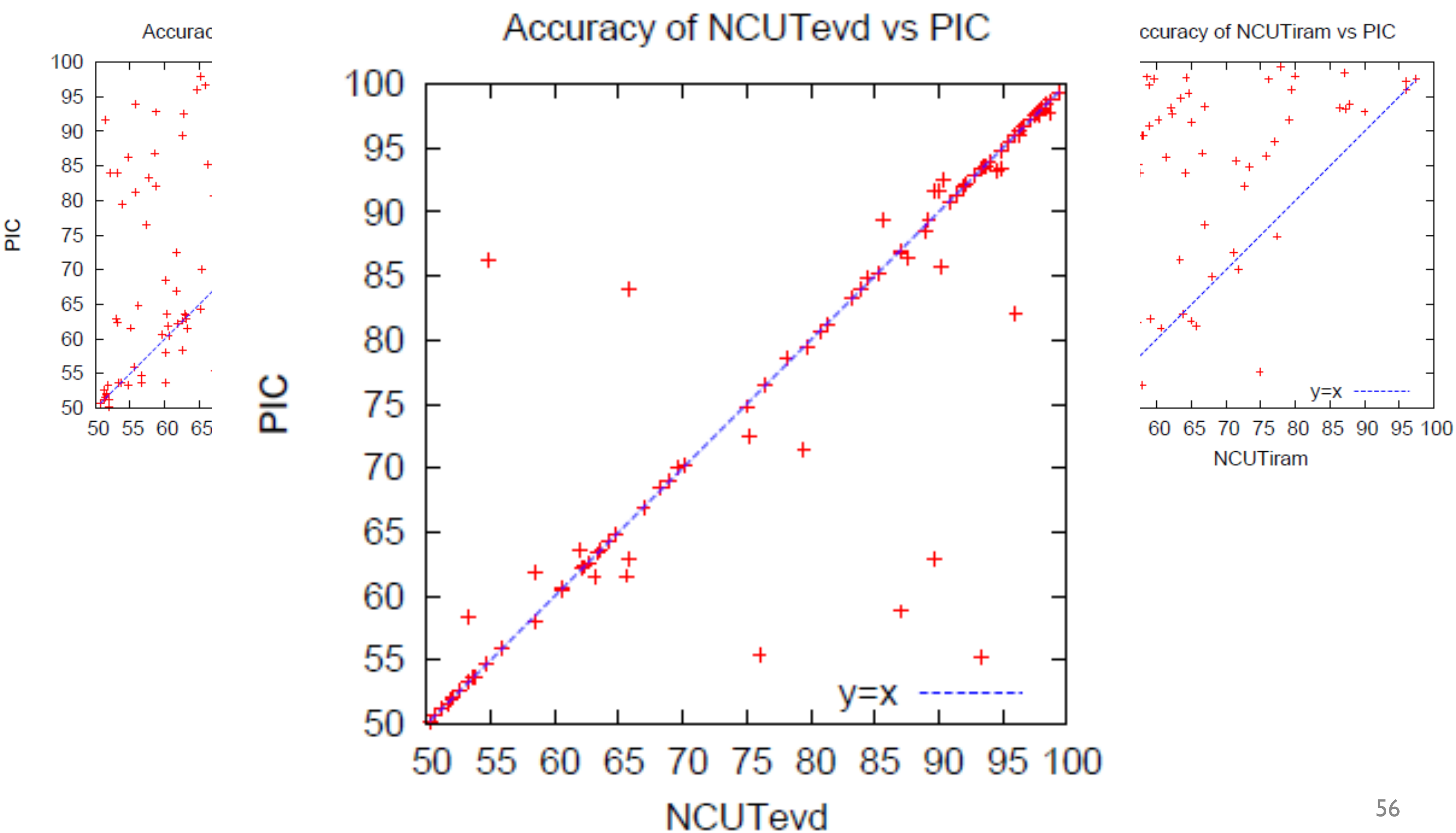
	ACC-Avg	NMI-Avg
baseline	57.59	-
k-means	69.43	0.2629
NCUTevd	**77.55**	**0.3962**
NCUTiram	61.63	0.0943
PIC	**76.67**	**0.3818**

- NCUTevd: Ncut with exact eigenvectors
- NCUTiram: Implicit restarted Arnoldi method
- No stat. signif. diffs between NCUTevd and PIC

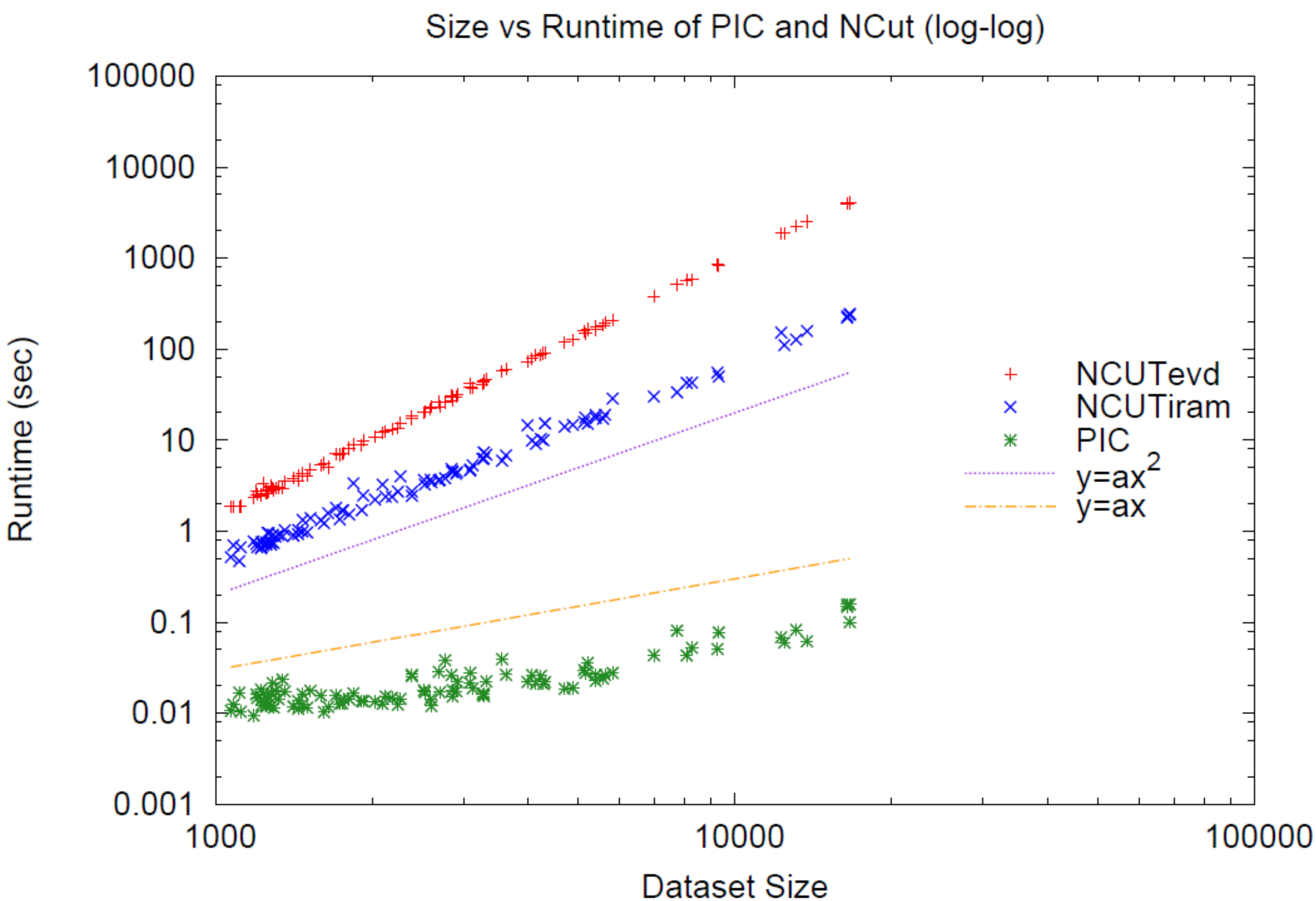
Results


Accuracy of k-means vs PIC
PIC
k-means
y=x
Accuracy of NCUTevd vs PIC
PIC
NCUTevd
y=x
Accuracy of NCUTiram vs PIC
PIC
NCUTiram
y=x

Results


Accurac
PIC
ccuracy of NCUTiram vs PIC
y=x
NCUTiram
Accuracy of NCUTevd vs PIC
PIC
y=x
NCUTevd

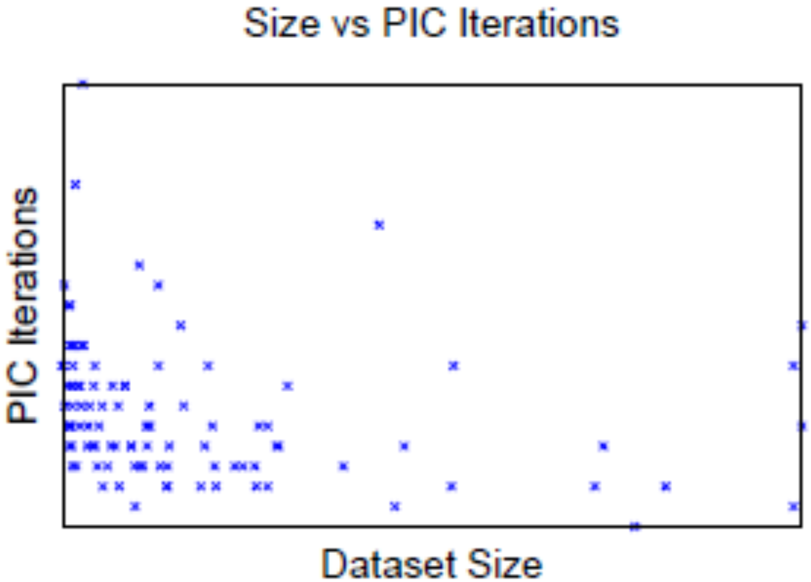
Results



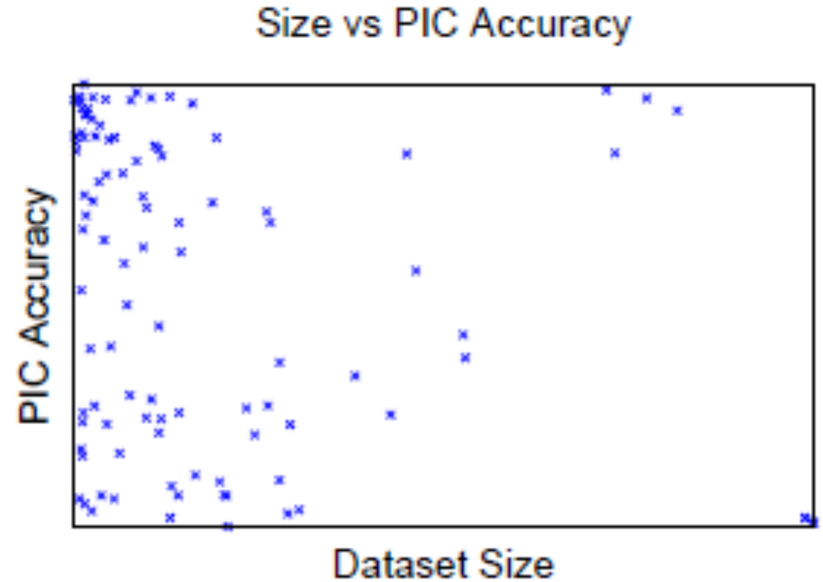
Size vs Runtime of PIC and NCut (log-log)
100000
10000
1000
100
10
1
0.1
0.01
0.001
Runtime (sec)
1000
10000
100000
Dataset Size
NCUTevd
NCUTiram
PIC
y=ax2
y=ax

Results

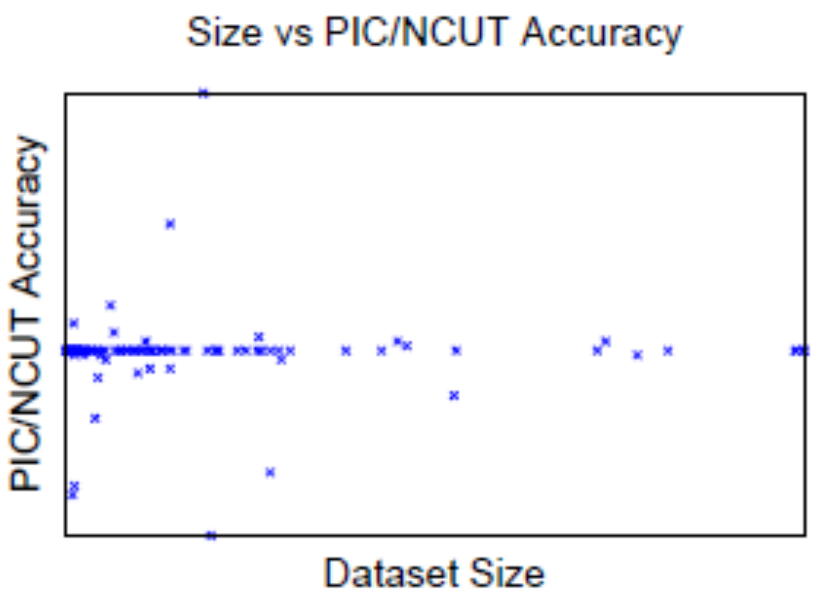
- Linear run-time implies *constant* number of iterations
- Number of iterations to "acceleration-convergence" is hard to analyze:
  - Faster than a single complete run of power iteration to convergence
  - On our datasets
    - 10-20 iterations is typical
    - 30-35 is exceptional



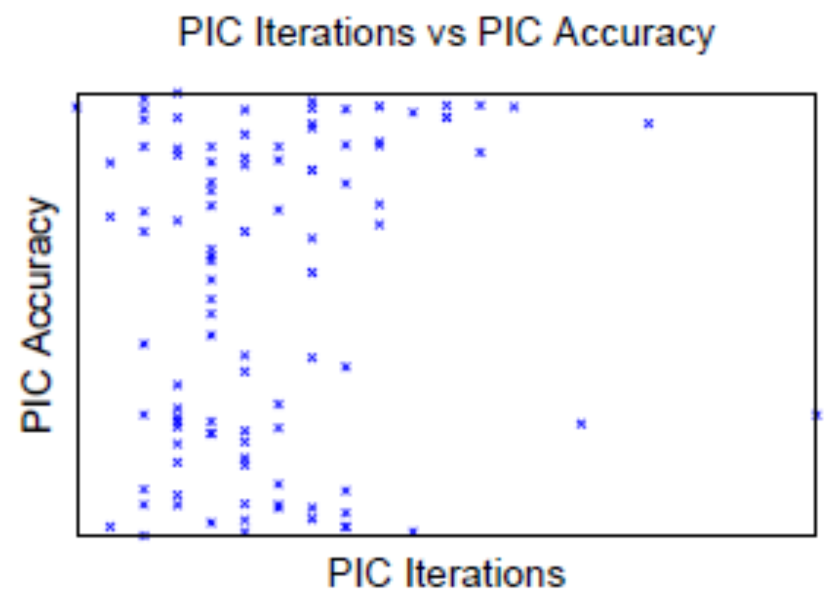
(a) $R^2 = 0.0424$



(b) $R^2 = 0.0552$



(c) $R^2 = 0.0007$



(d) $R^2 = 0.0134$

PIC AND SSL

PIC: Power Iteration Clustering

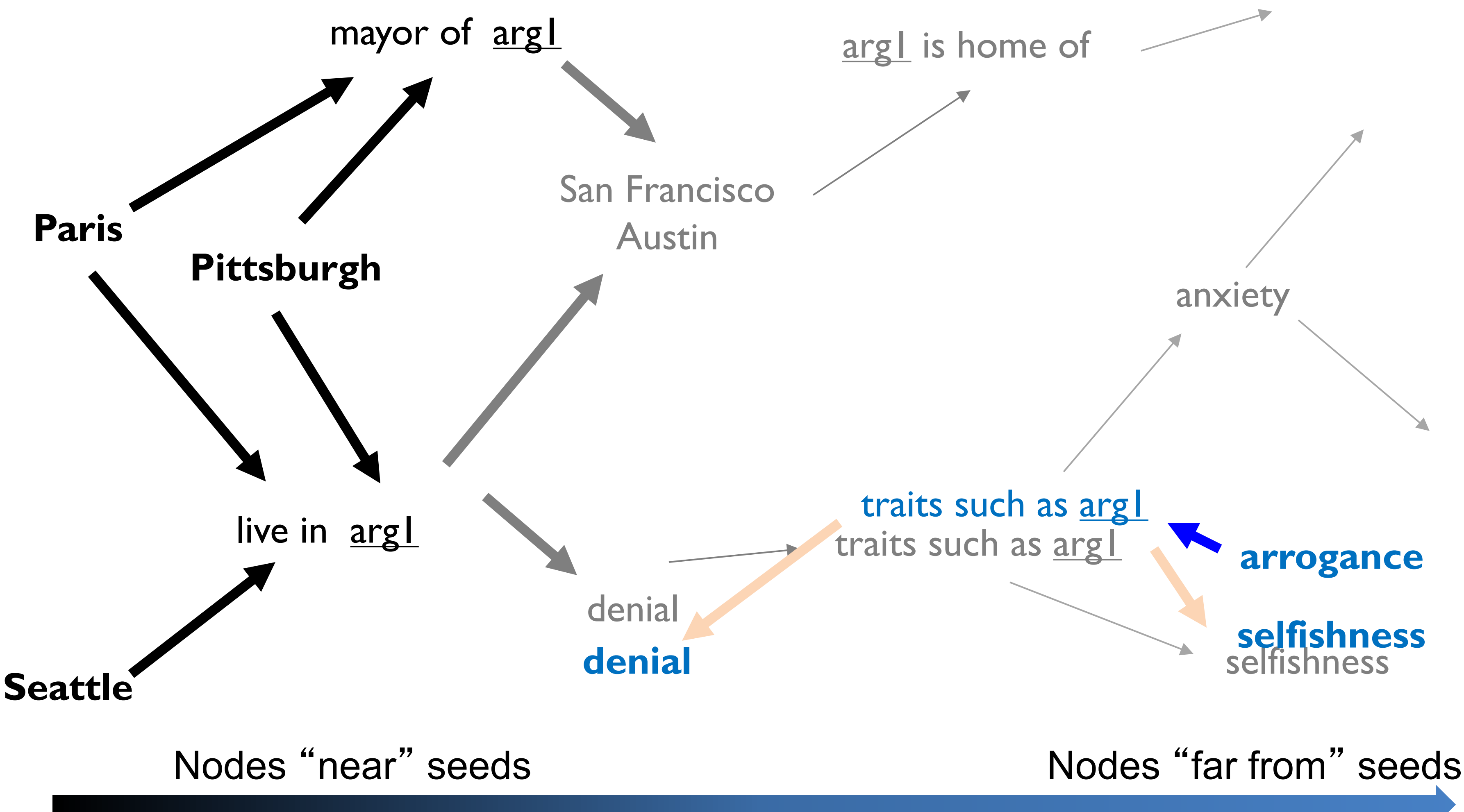
run power iteration (repeated averaging w/ neighbors) with early stopping

1. Pick an initial vector $\mathbf{v^0}$.
2. Set $\mathbf{v^{t+1}} \leftarrow \frac{W\mathbf{v^t}}{\|W\mathbf{v^t}\|_1}$ and $\delta^{t+1} \leftarrow |\mathbf{v^{t+1}} - \mathbf{v^t}|$.
3. Increment t and repeat above step until $|\delta^t - \delta^{t-1}| \simeq \mathbf{0}$.
4. Use k-means to cluster points on $\mathbf{v^t}$ and return clusters $C_1, C_2, ..., C_k$.

Harmonic Functions/CoEM/wvRN

1. Pick an initial vector $\mathbf{v}^0$.
2. Set $\mathbf{v}^{t+1} \leftarrow \frac{W\mathbf{v}^t}{\cancel{\|W\mathbf{v}^t\|_1}}$ then replace $\mathbf{v}^{t+1}$(i) with seed values +1/-1 for labeled data
3. Increment t and repeat above step for 5-10 iterations
4. Classify data using final values from **v**

Implicit Manifolds on the NELL datasets



Using the Manifold Trick for SSL

Name	20NG	RCV1	City	44Cat
Instances	19K	194K	88K	9,846K
Features	61K	47K	99K	8,622K
NZF	2M	11M	21M	121M
Cats	20	103	1	44
Type	doc	doc	NP	NP
Manifold	cosine	cosine	bipart	bipart
Input Size	39MB	198MB	330MB	2GB
IM Size	40MB	207MB	335MB	2.4GB
EM Size	5.6GB	*540GB	*80GB	*4TB

Table 1: Dataset comparison. *NZF* **is the total number of non-zero feature values and** *Cats* **is the number of categories.** *Type* **is the dataset type, where** *doc* **and** *NP* **correspond to document collection and noun phrase-context data, respectively.** *Manifold* **is the choice of manifold for the dataset, where** *cosine* **and** *bipart* **refers to cosine similarity and bipartite graph walk, respectively.** *Input Size* **is the MATLAB memory requirement for the original sparse feature matrix;** *IM Size* **is the total memory requirement for using the implicit manifold, including the feature matrix;** *EM Size* **is the memory requirement for constructing a explicit manifold. * indicates that the memory requirement is estimated using random sampling and extrapolation.**

Using the Manifold Trick for SSL

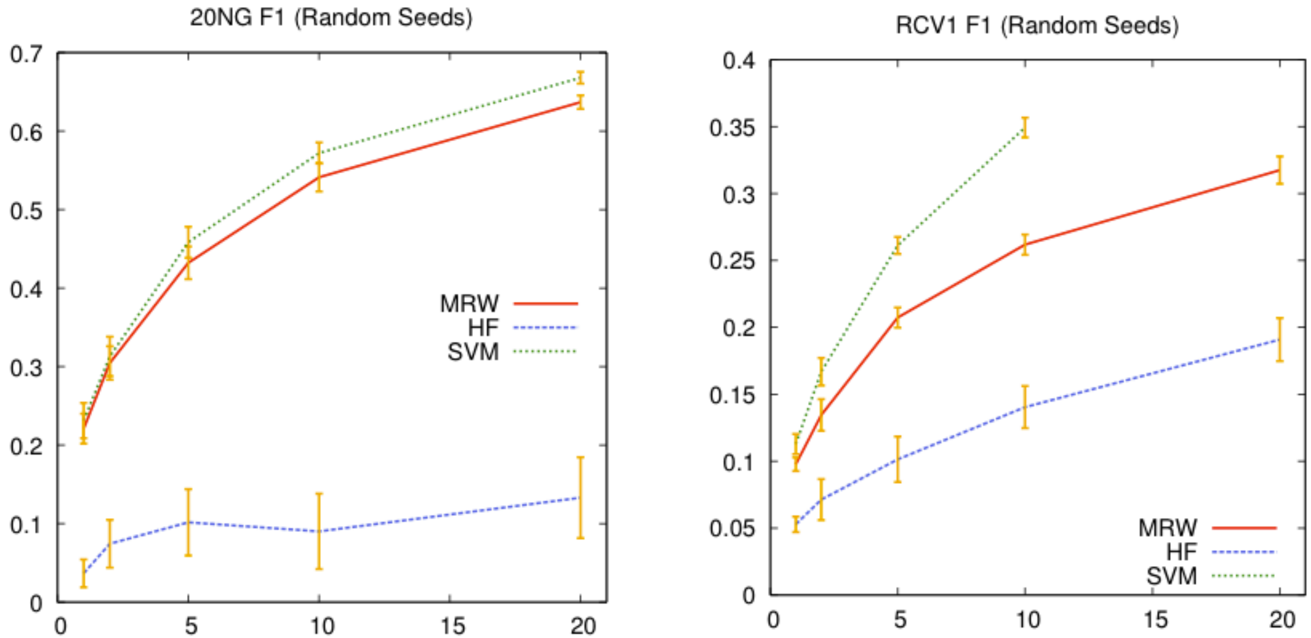


Figure 1: F1 scores on the 20NG and RCV1 datasets. The x-axis indicates the number of labeled instances and the y-axis indicates the macro-averaged F1 score. Vertical lines indicate standard deviation (over 20 trials for 20NG and 10 for RCV1) using randomly selected seed labels.

Using the Manifold Trick for SSL

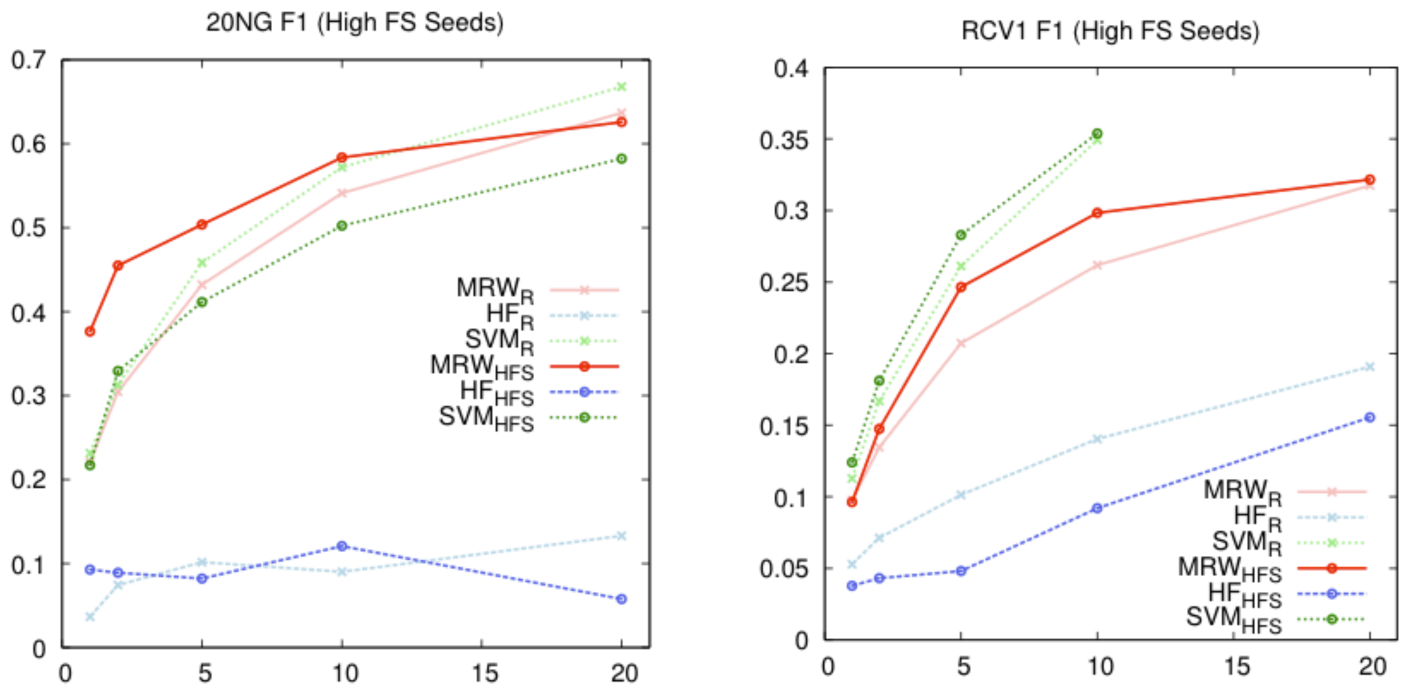


Figure 2: F1 scores on the 20NG and RCV1 datasets using preferred (high feature weight sum) seeds. Subscript HFS indicates result using high feature-sum seeds and R indicates result using random seeds—included for comparison.

Using the Manifold Trick for SSL

Method Manifold	SVM -	HF inner	MRW inner	HF bipart	MRW bipart
NDCG	0.0263	0.0402	0.0405	0.0406	**0.0408**
AP	0.0208	0.6728	0.7067	0.7130	**0.7389**
P@10%	0.0123	0.8732	0.8926	0.8796	**0.9094**
P@20%	0.0143	0.8698	0.8991	0.8941	**0.9162**
P@30%	0.0168	0.8773	0.9093	0.9036	**0.9116**
P@40%	0.0199	0.8574	0.8957	0.9118	**0.9179**
P@50%	0.0210	0.8227	0.8647	0.8832	**0.9038**
P@60%	0.0236	0.7591	0.7990	0.8093	**0.8307**
P@70%	0.0265	0.6337	0.6743	0.6805	**0.7189**
P@80%	0.0267	0.4131	0.4533	0.5087	**0.5297**
P@90%	0.0272	0.1927	0.2155	0.2521	**0.2926**
P@100%	0.0274	0.0275	0.0279	0.0280	**0.0289**

Table 2: City dataset result. Boldfaced font indicates the highest number in a row. *inner* refers to the inner product manifold and *bipart* refers to the bipartite graph walk manifold. Note that HF with bipart is equivalent to co-EM as used in [11]

Using the Manifold Trick for SSL

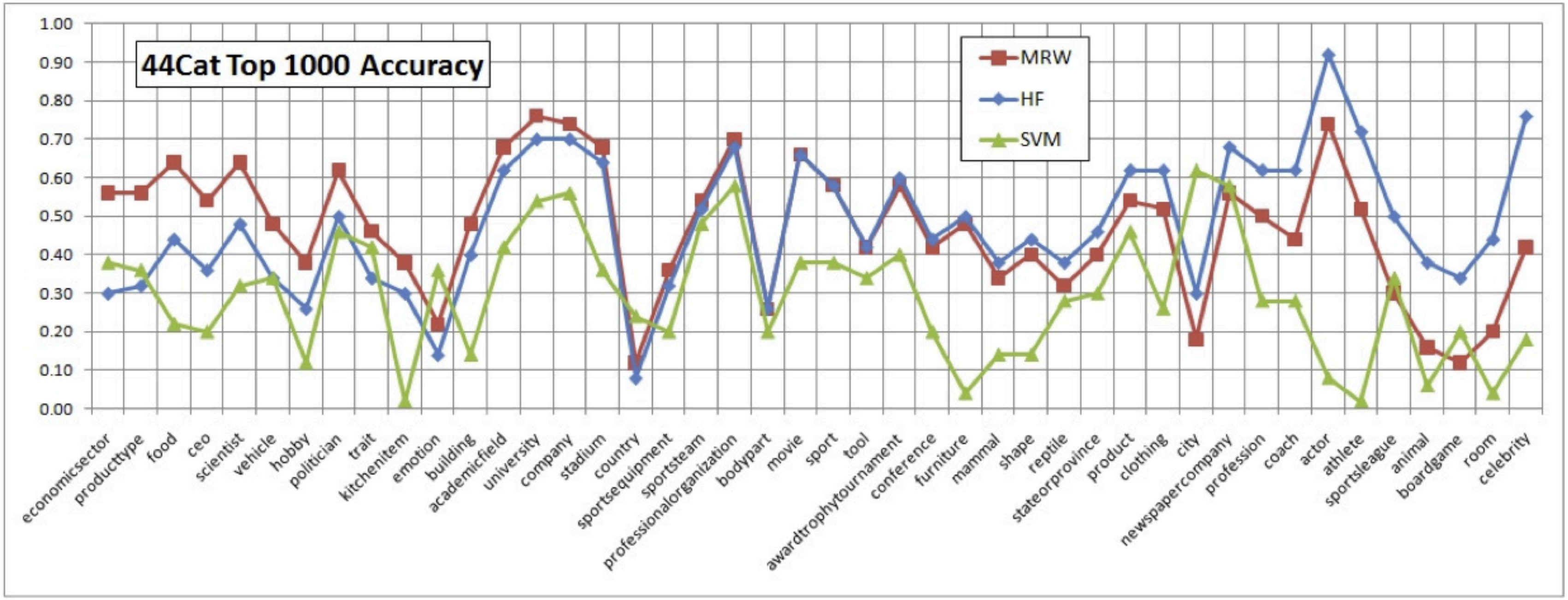


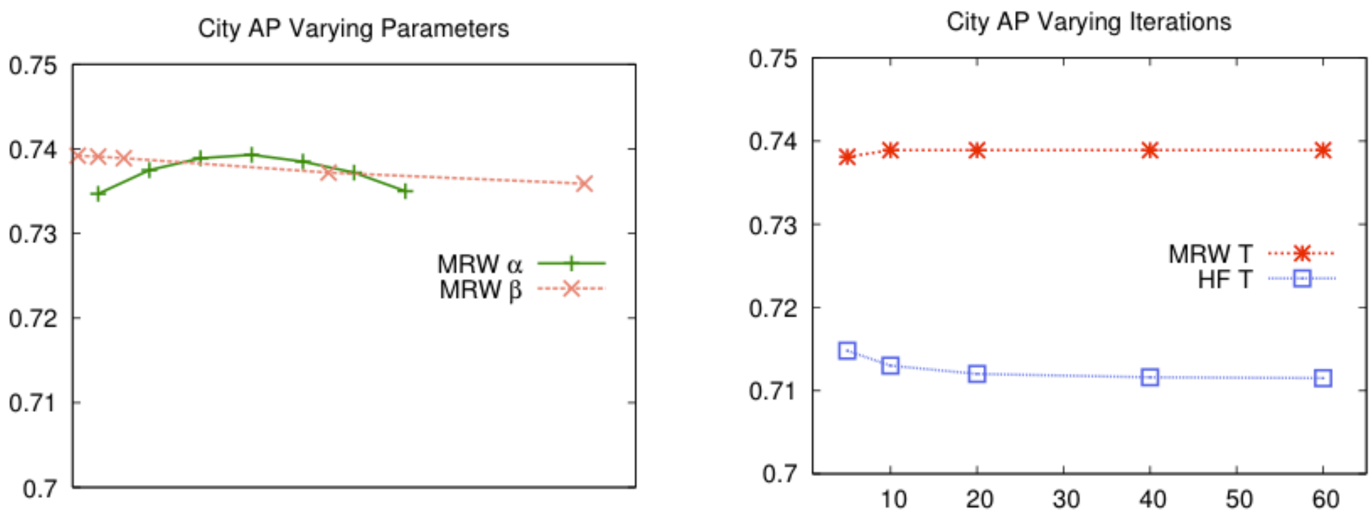
Figure 4: Sampled per-category accuracies of the top 1000 retrieved NPs on the 44Cat dataset. The categories are ordered from left to right according to the difference between the MRW accuracy and HF accuracy, from the high to low.

Using the Manifold Trick for SSL

A smoothing trick:

$$V^{t+1} \leftarrow (1 - \alpha - \beta) S D^{-1} V^t + \alpha R + \beta(\mathbf{1}/n)$$

Using the Manifold Trick for SSL



(a) α and β (b) the # of iterations T

Figure 3: Parameter sensitivity. The x-axis correspond to parameter values and the y-axis shows average precisions. α ranges from 0.05 to 0.65, β ranges from 0.0001 to 0.01; the number of iterations T are indicated below x-axes.