Fast Effective Clustering for Graphs and Document Collections

William W. Cohen

Machine Learning Dept. and Language Technologies Institute
School of Computer Science
Carnegie Mellon University

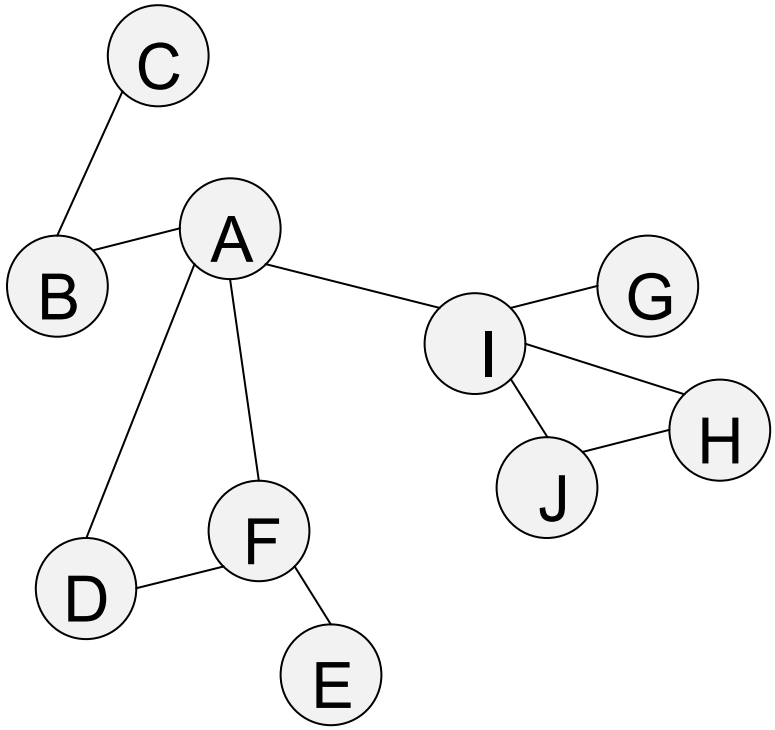
Joint work with: Frank Lin

Outline

- Background on spectral clustering
- "Power Iteration Clustering"
  - Motivation
  - Experimental results
- Analysis: PIC vs spectral methods
- PIC for sparse bipartite graphs
  - Motivation & Method
  - Experimental Results

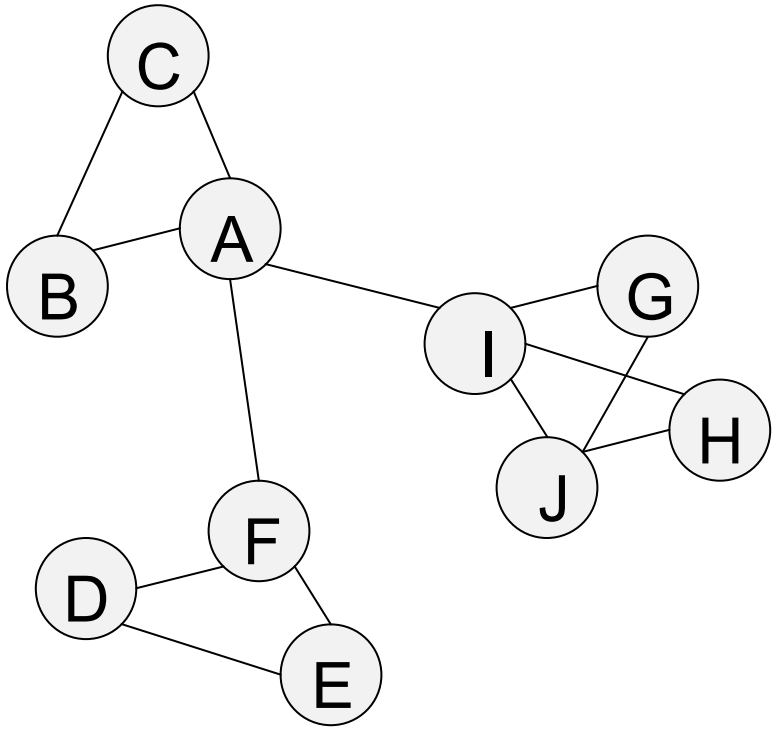
Spectral Clustering: Graph = Matrix

	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	_



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

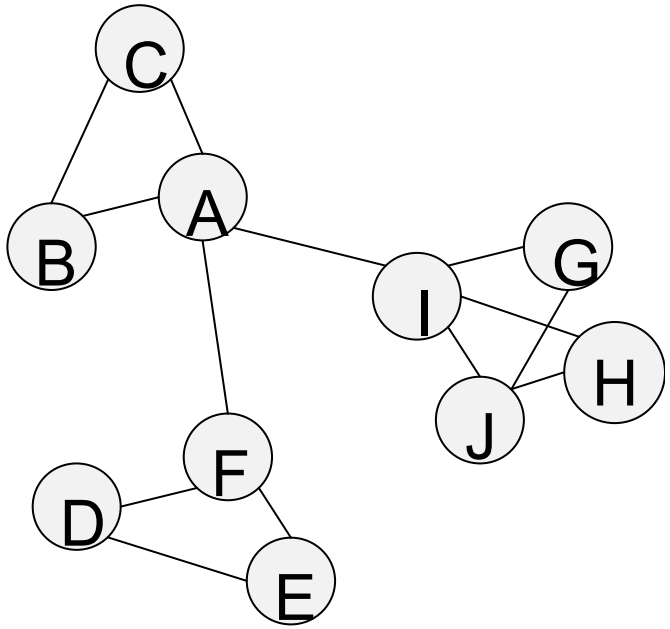
Spectral Clustering: Graph = Matrix
Vector = Node → Weight

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	



M

Spectral Clustering: Graph = Matrix

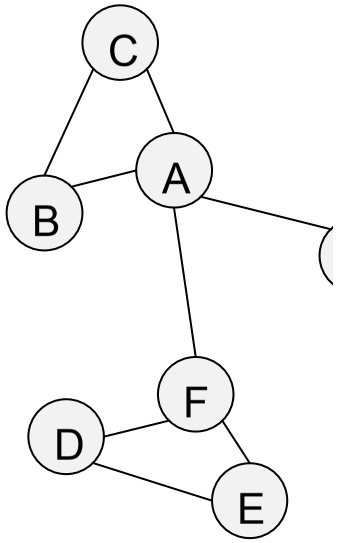
$M*v_1 = v_2$ "propogates weights from neighbors"

$$M * v_1 = v_2$$

	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"

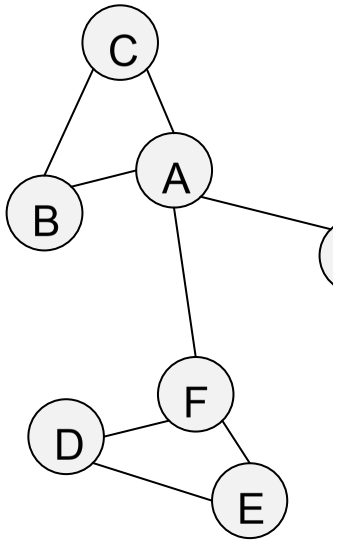
W: normalized so columns sum to 1

$$W * v_1 = v_2$$

	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	

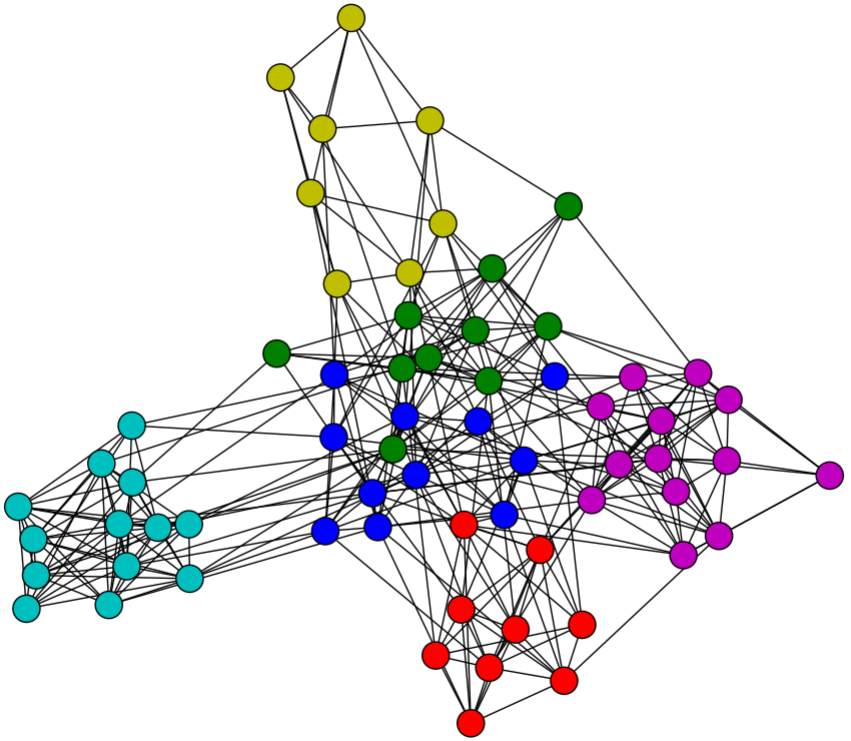


Spectral Clustering: Graph = Matrix

$W^*v_1 = v_2$ "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?

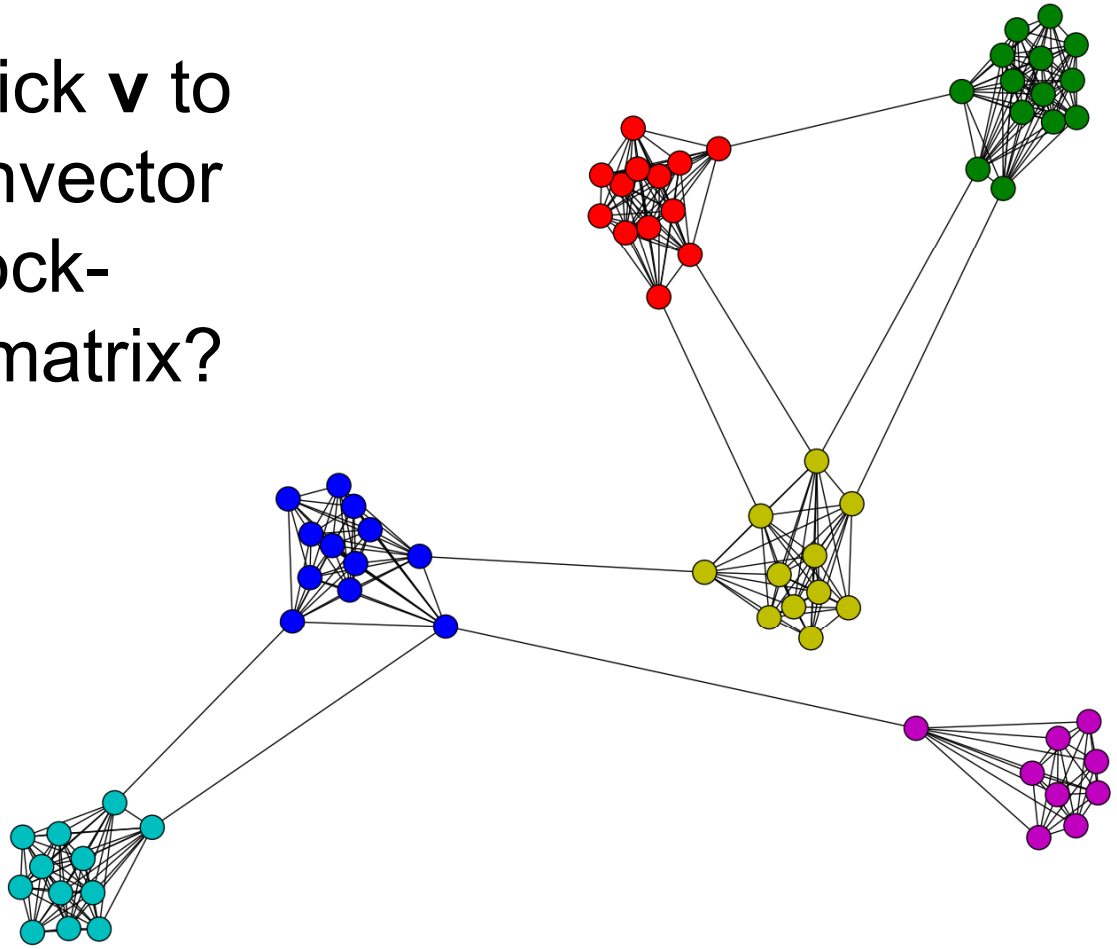


Spectral Clustering: Graph = Matrix

W*v_1 = v_2 "propogates weights from neighbors"

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

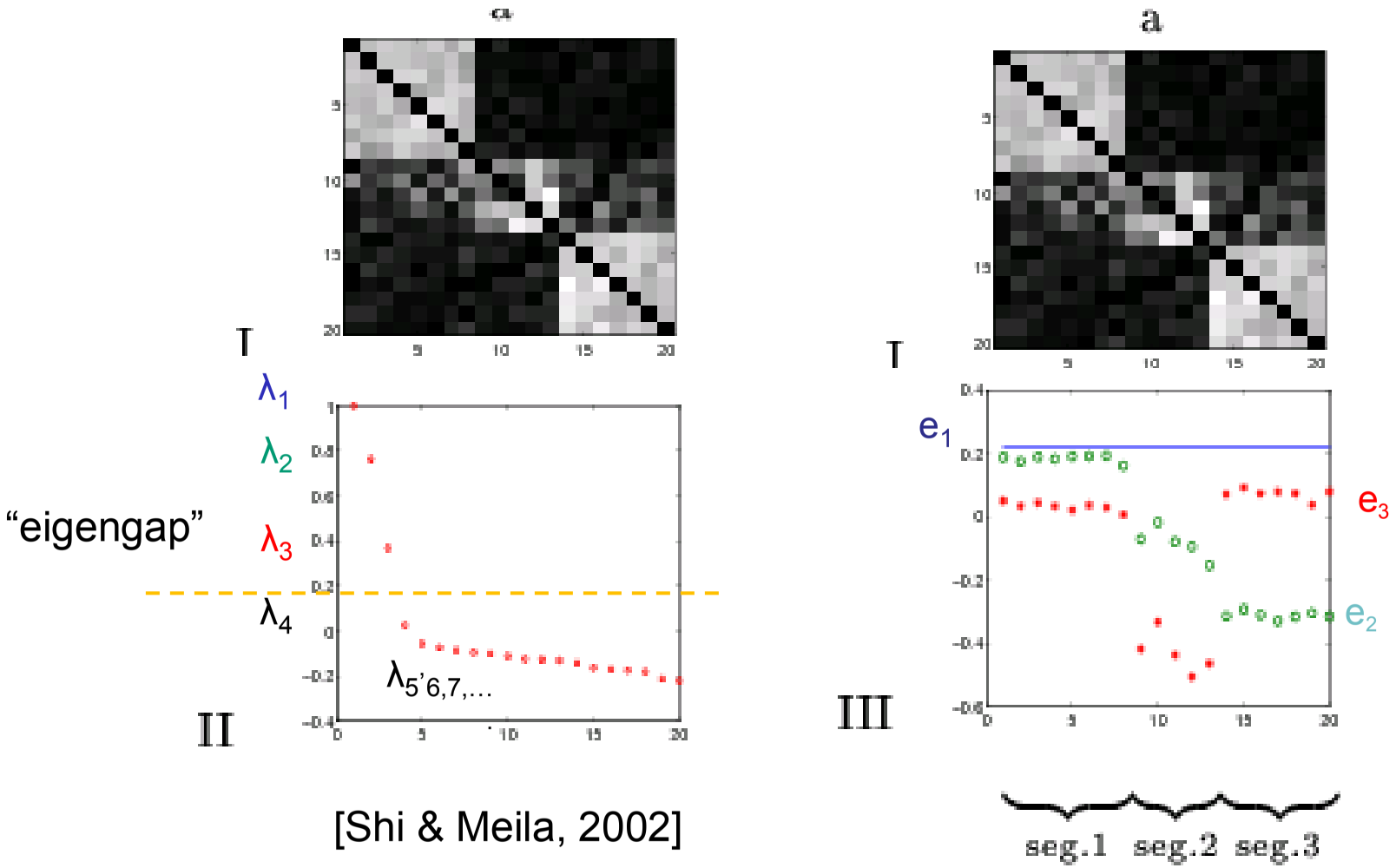
How do I pick **v** to be an eigenvector for a block-stochastic matrix?



Spectral Clustering: Graph = Matrix

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

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

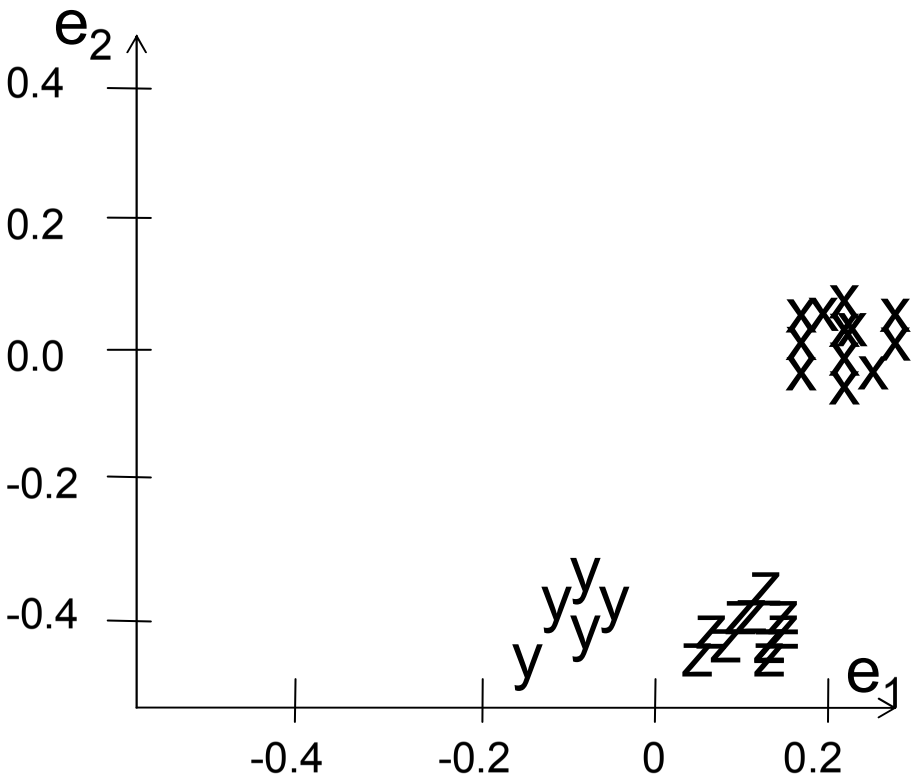


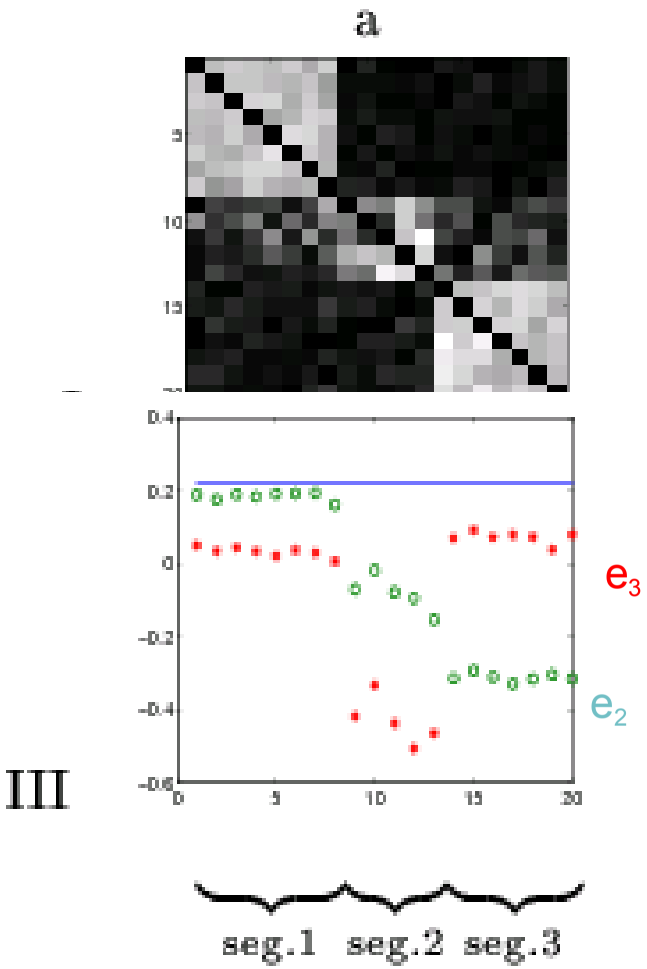
[Shi & Meila, 2002]

Spectral Clustering: Graph = Matrix

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

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



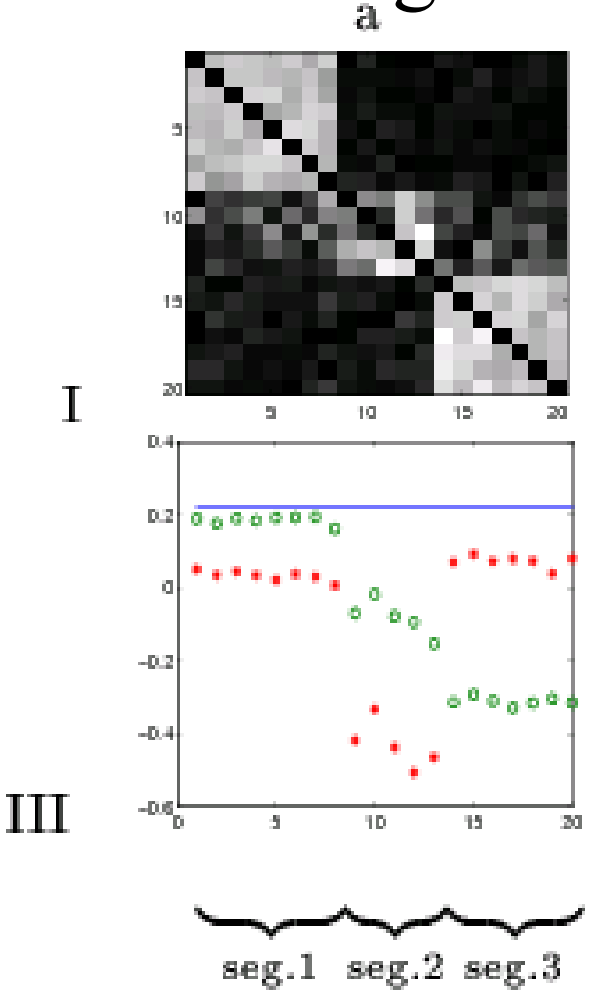


[Shi & Meila, 2002]

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:

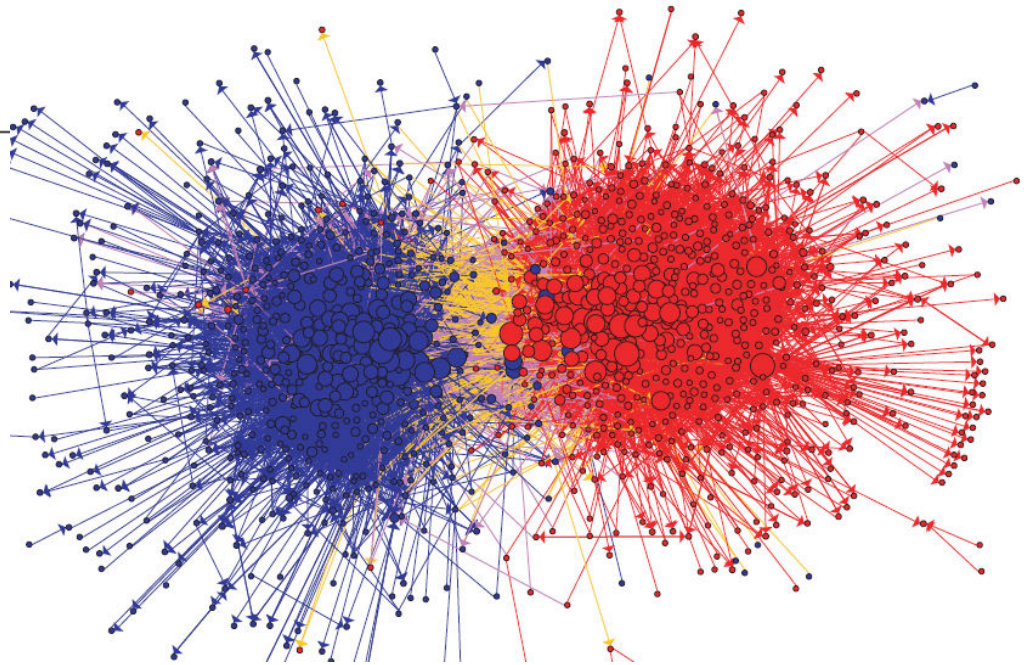
- Find the top *k+1* eigenvectors $\mathbf{v}_1,\ldots,\mathbf{v}_{k+1}$
- Discard the "top" one
- Replace every node *a* with *k*-dimensional vector $x_a = \langle \mathbf{v}_2(a),\ldots,\mathbf{v}_{k+1}(a) \rangle$
- Cluster with *k*-means

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		
20ngC	3	0.613	0.621		
20ngD	4	0.469	0.432		
Average	-	0.716	0.663		



Adamic & Glance
"Divided They Blog:…" 2004

Spectral Clustering: Graph = Matrix

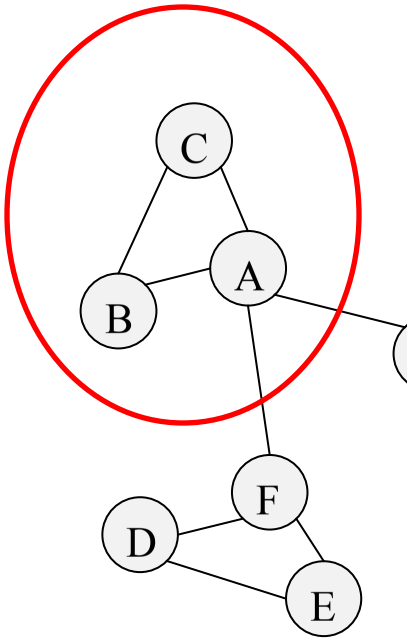
$M*v_1 = v_2$ "propogates weights from neighbors"

$$M * v_1 = v_2$$

	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	5
B	6
C	5
D	
E	
F	
G	
H	
I	
J	

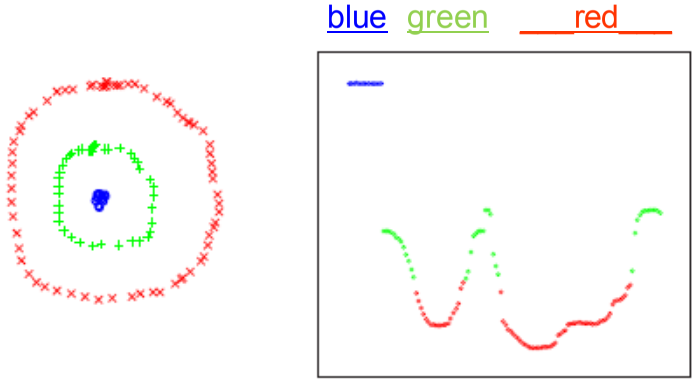


Repeated averaging with neighbors as a clustering method

- 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, …

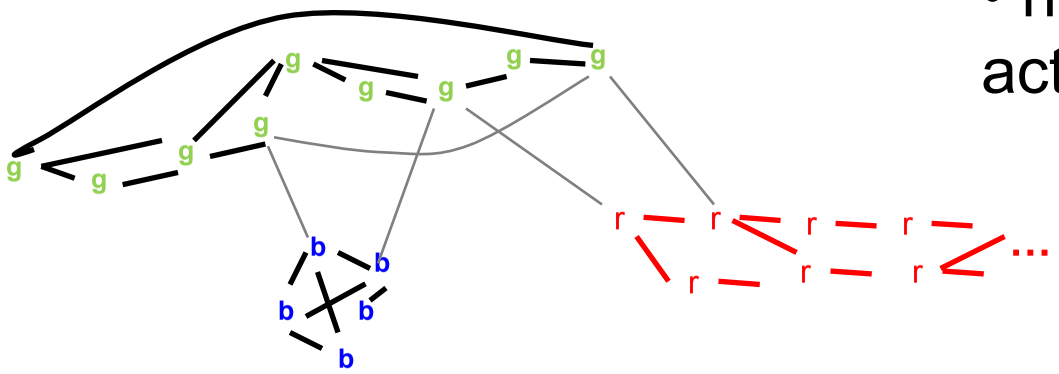
- Variants widely used for *semi-supervised* learning
  - clamping of labels for nodes with known labels
- Without clamping, will converge to constant v^∞
- What are the *dynamics* of this process?

Repeated averaging with neighbors on a sample problem…

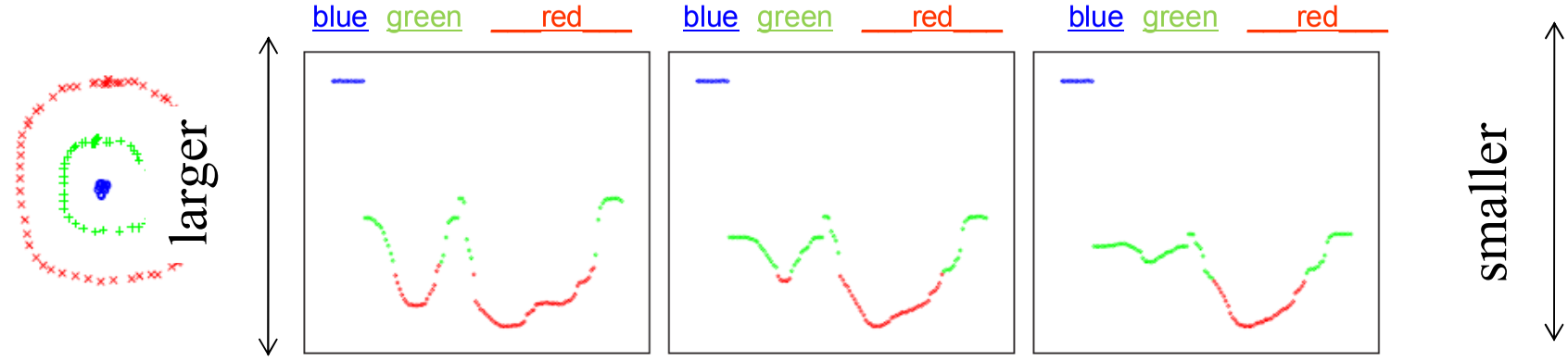


(a) 3Circles PIC result (b) Embedding at $t = 10$

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

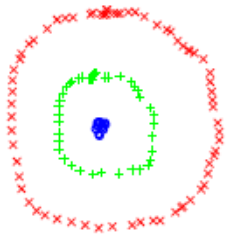


Repeated averaging with neighbors on a sample problem…

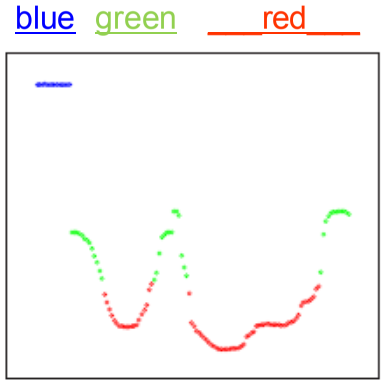


(a) 3Circles PIC result (b) Embedding at $t = 10$ (c) Embedding at $t = 50$ (d) Embedding at $t = 100$

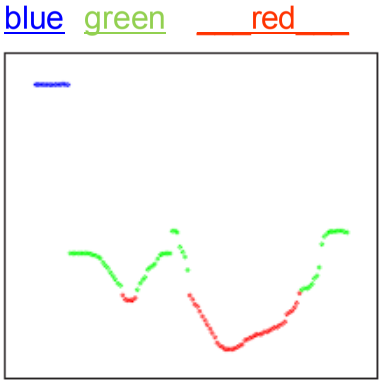
Repeated averaging with neighbors on a sample problem…



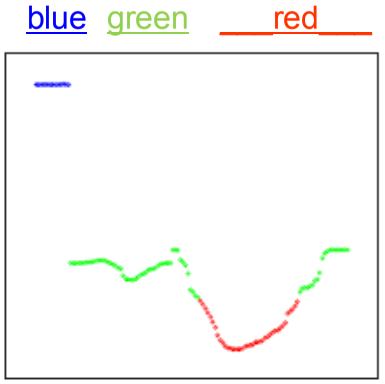
(a) 3Circles PIC result



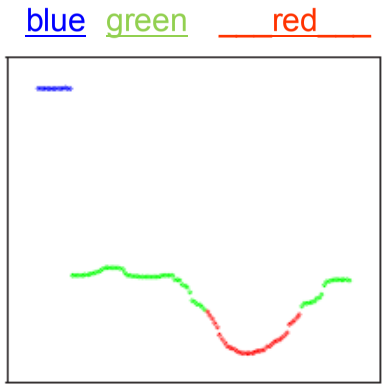
(b) Embedding at $t = 10$



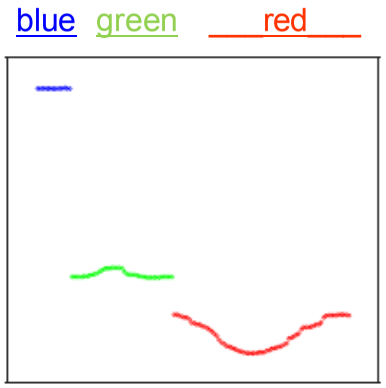
(c) Embedding at $t = 50$



(d) Embedding at $t = 100$

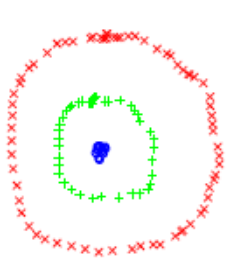


(e) Embedding at $t = 200$

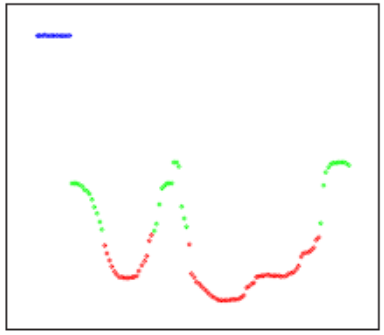


(f) Embedding at $t = 400$

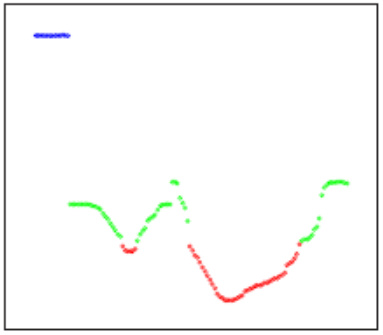
Repeated averaging with neighbors on a sample problem…



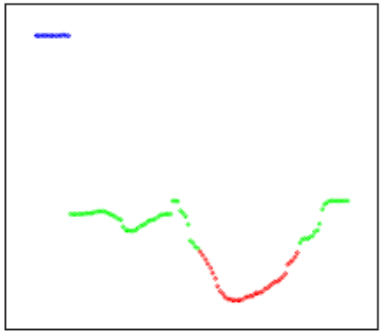
(a) 3Circles PIC result



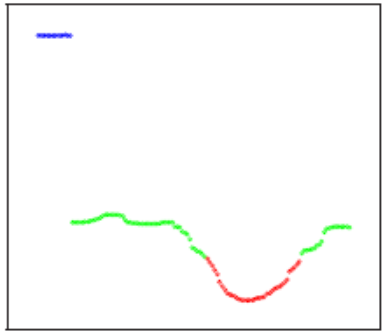
(b) Embedding at $t = 10$



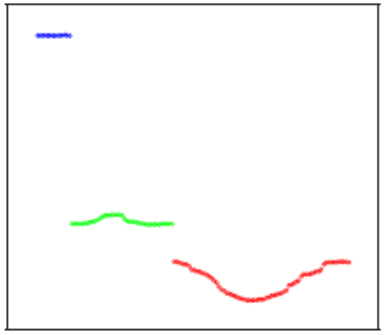
(c) Embedding at $t = 50$



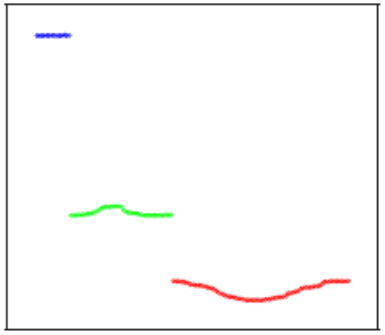
(d) Embedding at $t = 100$



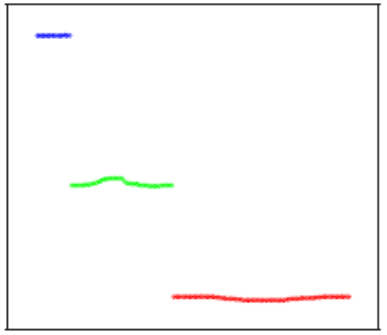
(e) Embedding at $t = 200$



(f) Embedding at $t = 400$



(g) Embedding at $t = 600$



(h) Embedding at $t = 1000$

very small

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!

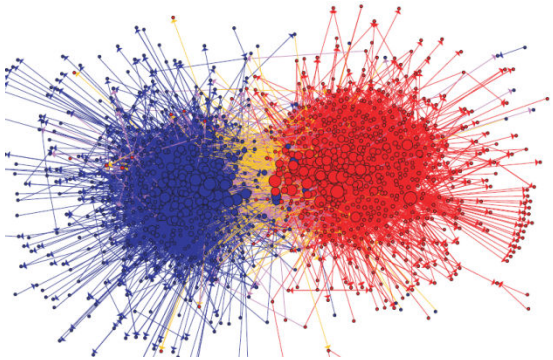
Experiments

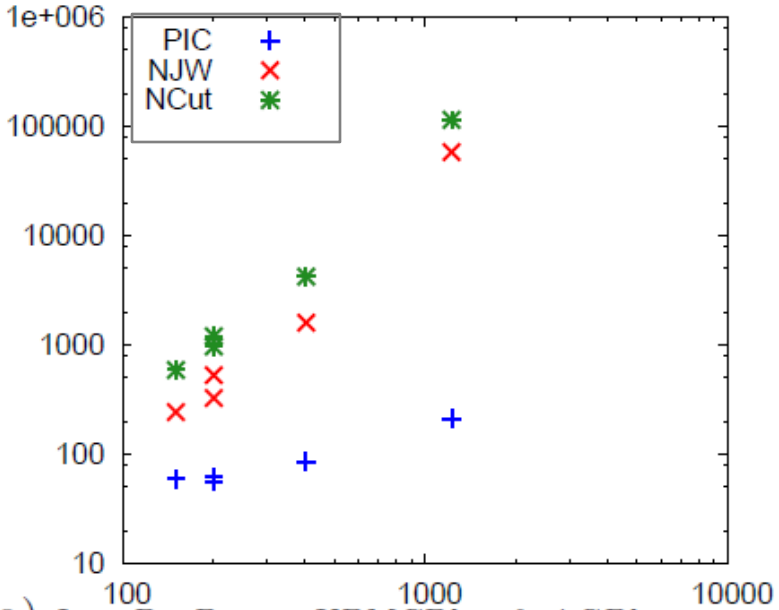
- "Network" problems: natural graph structure
  - PolBooks: 105 political books, 3 classes, linked by copurchaser
  - UMBCBlog: 404 political blogs, 2 classes, blogroll links
  - AGBlog: 1222 political blogs, 2 classes, blogroll links
- "Manifold" problems: cosine distance between classification instances
  - 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

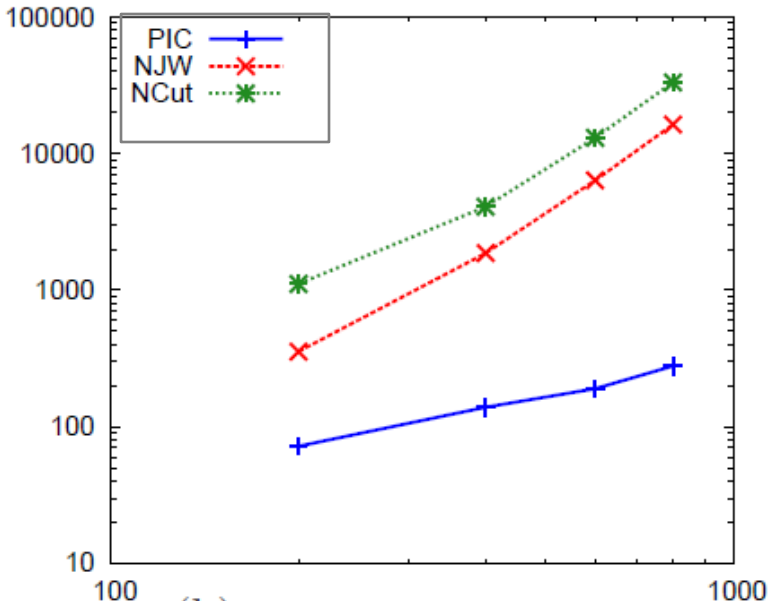
		NCut		NJW		PIC	
Dataset	**k**	**Accuracy**	**Macro-F1**	**Accuracy**	**Macro-F1**	**Accuracy**	**Macro-F1**
Iris	3	0.673	0.570	0.807	0.806	0.980	0.980
PenDigits01	2	1.000	1.000	1.000	1.000	1.000	1.000
PenDigits17	2	0.755	0.753	0.755	0.754	0.755	0.753
UBMCBlog	2	0.953	0.953	0.953	0.953	0.948	0.948
AGBlog	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.

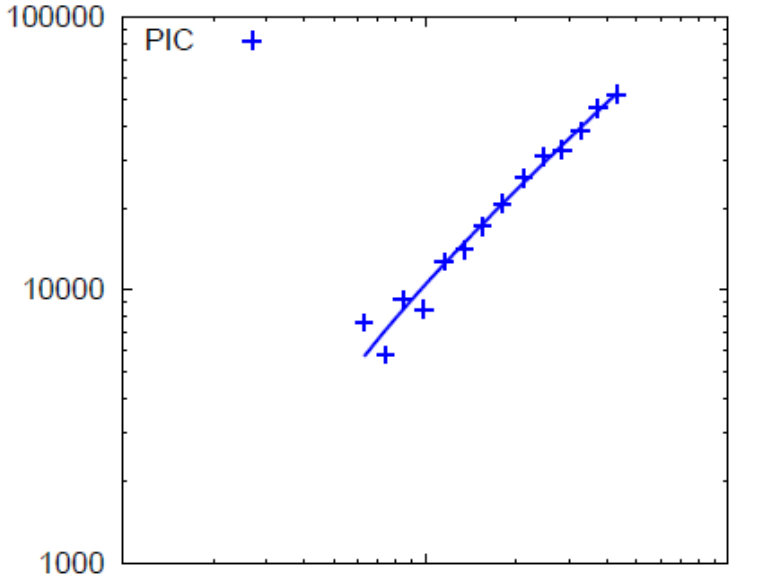




(a) Iris, PenDigits, UBMCBlog & AGBlog



(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

Analysis: why is this working?

eigenvectors $\mathbf{e}_1, \ldots, \mathbf{e}_n$

eigenvalues $\lambda_1, \ldots, \lambda_n$,

$\mathbf{s}_a = \langle \mathbf{e}_1(a), \ldots, \mathbf{e}_k(a) \rangle$,

$$spec(a,b) \equiv \|\mathbf{s}_a - \mathbf{s}_b\|_2 = \sqrt{\sum_{i=2}^{k} (\mathbf{e}_i(a) - \mathbf{e}_i(b))^2}$$

$$pic^t(\mathbf{v}^0; a, b) \equiv |\mathbf{v}^t(a) - \mathbf{v}^t(b)|$$

Analysis: why is this working?

$$\text{eigenvectors } \mathbf{e}_1, \ldots, \mathbf{e}_n \qquad \text{eigenvalues } \lambda_1, \ldots, \lambda_n,$$

$$\mathbf{s}_a = \langle \mathbf{e}_1(a), \ldots, \mathbf{e}_k(a) \rangle,$$

$$pic^t(\mathbf{v}^0; a, b) \equiv |\mathbf{v}^t(a) - \mathbf{v}^t(b)|$$

$$\begin{aligned}\mathbf{v}^t &= W\mathbf{v}^{t-1} = W^2\mathbf{v}^{t-2} = \ldots = W^t\mathbf{v}^0 \\ &= c_1 W^t \mathbf{e}_1 + c_2 W^t \mathbf{e}_2 + \ldots + c_n W^t \mathbf{e}_n \\ &= c_1 \lambda_1^t \mathbf{e}_1 + c_2 \lambda_2^t \mathbf{e}_2 + \ldots + c_n \lambda_n^t \mathbf{e}_n\end{aligned}$$

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

Analysis: why is this working?

$$\text{eigenvectors } \mathbf{e}_1, \ldots, \mathbf{e}_n \qquad \text{eigenvalues } \lambda_1, \ldots, \lambda_n,$$

$$\mathbf{s}_a = \langle \mathbf{e}_1(a), \ldots, \mathbf{e}_k(a) \rangle,$$

$$spec(a,b) \equiv \|\mathbf{s}_a - \mathbf{s}_b\|_2 = \sqrt{\sum_{i=2}^{k} (\mathbf{e}_i(a) - \mathbf{e}_i(b))^2}$$

L2 distance

$$pic^t(a,b) = |[\mathbf{e}_1(a) - \mathbf{e}_1(b)]c_1\lambda_1^t|$$

scaling?

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

differences might cancel?

"noise" terms

Analysis: why is this working?

- If
  - eigenvectors $e_2,\ldots,e_k$ are approximately piecewise constant on blocks;
  - $\lambda_2,\ldots,\lambda_k$ are "large" and $\lambda_{k+1},\ldots$ are "small";
    - **e.g., if matrix is block-stochastic**
  - the c_i's for v^0 are bounded;
  - for any a,b from distinct blocks there is at least one e_i with $e_i(a)-e_i(b)$ "large"
- Then exists an R so that
  - spec(a,b) small $\Leftrightarrow$ R*pic(a,b) small

Analysis: why is this working?

$$\text{eigenvectors } \mathbf{e}_1, \ldots, \mathbf{e}_n \quad \text{eigenvalues } \lambda_1, \ldots, \lambda_n,$$

$$\mathbf{s}_a = \langle \mathbf{e}_1(a), \ldots, \mathbf{e}_k(a) \rangle,$$

$$spec(a,b) \equiv \|\mathbf{s}_a - \mathbf{s}_b\|_2 = \sqrt{\sum_{i=2}^{k} (\mathbf{e}_i(a) - \mathbf{e}_i(b))^2}$$

$$pic^t(a,b) = \left| [\mathbf{e}_1(a) - \mathbf{e}_1(b)] c_1 \lambda_1^t \right.$$

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

- Sum of differences vs sum-of-squared differences
- **"soft" eigenvector selection**

Dataset	k	Purity	NMI	RI
Iris	3	0.6733	0.7235	0.7779
PenDigits01	2	**1.0000**	**1.0000**	**1.0000**
PenDigits17	2	**0.7550**	**0.2066**	**0.6301**
PolBooks	3	0.8476	0.5745	0.8447
UBMCBlog	2	**0.9530**	**0.7488**	**0.9104**
AGBlog	2	0.5205	0.0060	0.5006
20ngA	2	**0.9600**	**0.7594**	**0.9232**
20ngB	2	0.5050	0.0096	0.5001
20ngC	3	0.6183	0.3295	0.6750
20ngD	4	0.4750	0.2385	0.6312
Average		0.7308	0.4596	0.7393

Ncut with top *k* eigenvectors

Ncut with top 10 eigenvectors: *weighted*

Table 2. Clustering performance of eigenvalue-weighted NCut on several real datasets. For all measures a higher number means better clustering. Bold numbers are the highest in its row.

		uniform weights			e_i weighted by λ_i			e_i weighted by λ_i^{15}		
Dataset	k	Purity	NMI	RI	Purity	NMI	RI	Purity	NMI	RI
Iris	3	0.6667	0.6507	0.7254	**0.9800**	**0.9306**	**0.9741**	**0.9800**	**0.9306**	**0.9741**
PenDigits01	2	0.7000	0.2746	0.5800	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
PenDigits17	2	0.7000	0.1810	0.5800	**0.7550**	**0.2066**	**0.6301**	**0.7550**	**0.2066**	**0.6301**
PolBooks	3	0.4857	0.1040	0.4413	**0.8476**	0.5861	**0.8514**	0.8381	**0.5936**	0.8453
UBMCBlog	2	0.9505	0.7400	0.9059	0.9505	0.7400	0.9059	**0.9530**	**0.7488**	**0.9104**
AGBlog	2	0.9493	0.7143	0.9037	**0.9509**	**0.7223**	**0.9066**	0.9501	0.7175	0.9051
20ngA	2	0.5600	0.0685	0.5072	**0.9600**	**0.7594**	**0.9232**	0.9450	0.7005	0.8961
20ngB	2	0.7125	0.2734	0.5903	**0.9450**	**0.7042**	**0.8961**	0.5050	0.0096	0.5001
20ngC	3	0.6867	0.3866	0.6546	**0.6617**	0.3772	**0.7025**	0.6350	**0.4719**	0.6784
20ngD	4	0.4763	0.2365	0.6368	0.4875	0.2555	0.6425	**0.5263**	**0.2906**	**0.7129**
Average		0.6888	0.3630	0.6525	**0.8538**	**0.6282**	**0.8432**	0.8087	0.5670	0.8052

Dataset	k	Purity	NMI	RI
Iris	3	0.6733	0.7235	0.7779
PenDigits01	2	**1.0000**	**1.0000**	**1.0000**
PenDigits17	2	**0.7550**	**0.2066**	**0.6301**
PolBooks	3	0.8476	0.5745	0.8447
UBMCBlog	2	**0.9530**	**0.7488**	**0.9104**
AGBlog	2	0.5205	0.0060	0.5006
20ngA	2	**0.9600**	**0.7594**	**0.9232**
20ngB	2	0.5050	0.0096	0.5001
20ngC	3	0.6183	0.3295	0.6750
20ngD	4	0.4750	0.2385	0.6312
Average		0.7308	0.4596	0.7393

Purity	NMI	RI
0.9800	**0.9306**	**0.9741**
1.0000	**1.0000**	**1.0000**
0.7550	**0.2066**	**0.6301**
0.8667	**0.6234**	**0.8603**
0.9480	0.7193	**0.9014**
0.9574	**0.7465**	**0.9185**
0.9600	**0.7594**	**0.9232**
0.8700	**0.5230**	**0.7738**
0.6933	**0.4450**	**0.7363**
0.5825	**0.3133**	**0.7149**
0.8613	**0.6267**	**0.8433**

Table 2. Clustering performance of eigenvalue-weighted NCut on several real datasets. For all measures a higher number means better clustering. Bold numbers are the highest in its row.

		uniform weights			e_i weighted by λ_i			e_i weighted by λ_i^{15}		
Dataset	k	Purity	NMI	RI	Purity	NMI	RI	Purity	NMI	RI
Iris	3	0.6667	0.6507	0.7254	**0.9800**	**0.9306**	**0.9741**	**0.9800**	**0.9306**	**0.9741**
PenDigits01	2	0.7000	0.2746	0.5800	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
PenDigits17	2	0.7000	0.1810	0.5800	**0.7550**	**0.2066**	**0.6301**	**0.7550**	**0.2066**	**0.6301**
PolBooks	3	0.4857	0.1040	0.4413	**0.8476**	0.5861	**0.8514**	0.8381	**0.5936**	0.8453
UBMCBlog	2	0.9505	0.7400	0.9059	0.9505	0.7400	0.9059	**0.9530**	**0.7488**	**0.9104**
AGBlog	2	0.9493	0.7143	0.9037	**0.9509**	**0.7223**	**0.9066**	0.9501	0.7175	0.9051
20ngA	2	0.5600	0.0685	0.5072	**0.9600**	**0.7594**	**0.9232**	0.9450	0.7005	0.8961
20ngB	2	0.7125	0.2734	0.5903	**0.9450**	**0.7042**	**0.8961**	0.5050	0.0096	0.5001
20ngC	3	0.6867	0.3866	0.6546	**0.6617**	0.3772	**0.7025**	0.6350	**0.4719**	0.6784
20ngD	4	0.4763	0.2365	0.6368	0.4875	0.2555	0.6425	**0.5263**	**0.2906**	**0.7129**
Average		0.6888	0.3630	0.6525	**0.8538**	**0.6282**	**0.8432**	0.8087	0.5670	0.8052

Summary of results so far

- Both PIC and Ncut embed each graph node in a space where distance is meaningful
- Distances in "PIC space" and Eigenspace are closely related
  - At least for many graphs suited to spectral clustering
- PIC does "soft" selection of eigenvectors
  - Strong eigenvalues give high weights
- PIC gives comparable-quality clusters
  - But is *much* faster

Outline

- Background on spectral clustering
- "Power Iteration Clustering"
  - Motivation
  - Experimental results
- Analysis: PIC vs spectral methods
- **PIC for sparse bipartite graphs**
  - "Lazy" Distance Computation
  - "Lazy" Normalization
  - Experimental Results

Motivation: Experimental Datasets 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
  - …

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: $D=A*\mathbf{1}$

1 is a column vector of 1's

- 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: $A = N^{-1}F^TFN^{-1}$

$\langle u,v\rangle$=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 **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

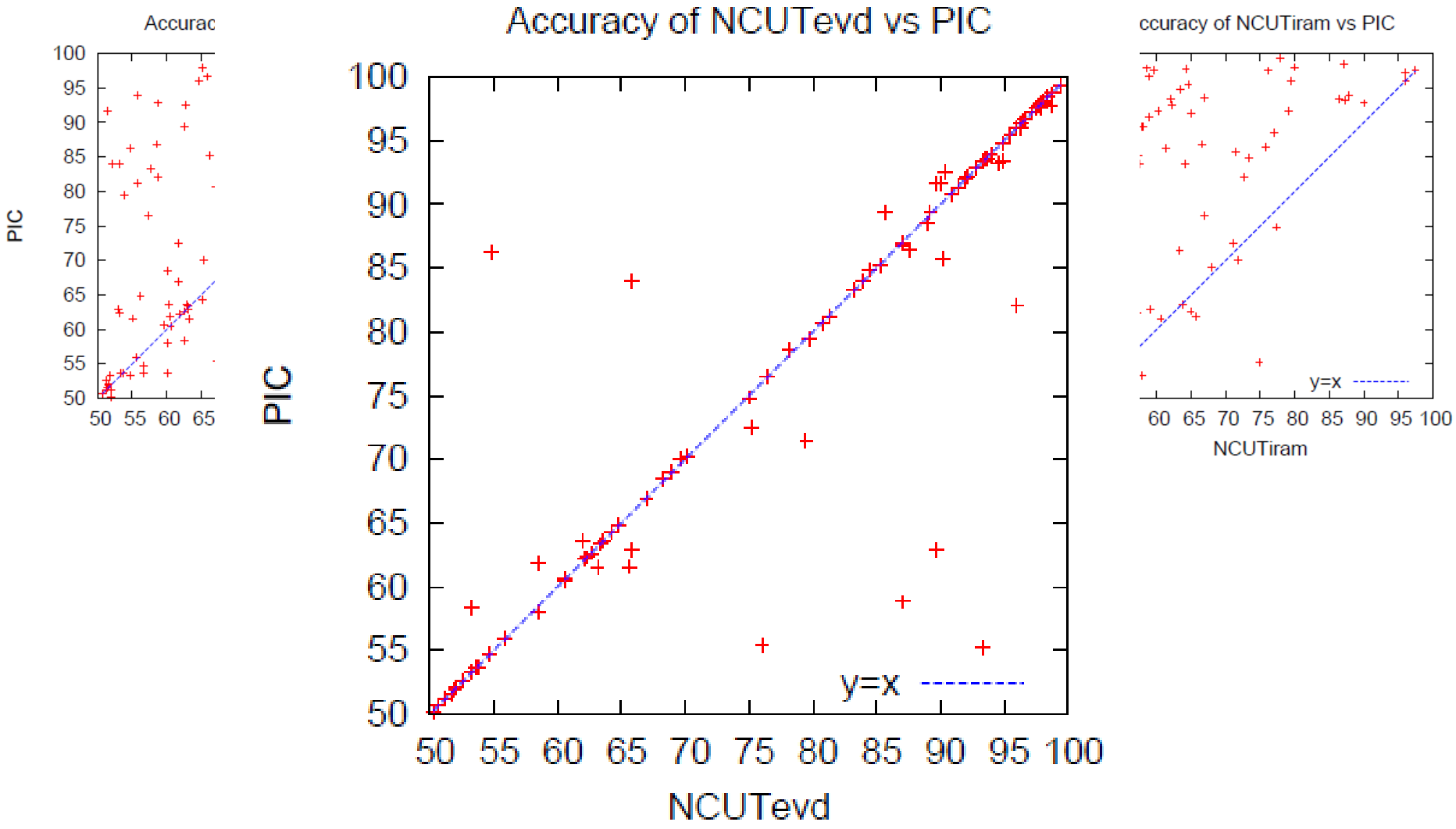
- RCV1 text classification dataset
  - 800k + newswire stories
  - Category labels from *industry* vocabulary
  - Took single-label documents and categories with at least 500 instances
  - Result: 193,844 documents, 103 categories
- Generated 100 random category pairs
  - Each is all documents from two 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

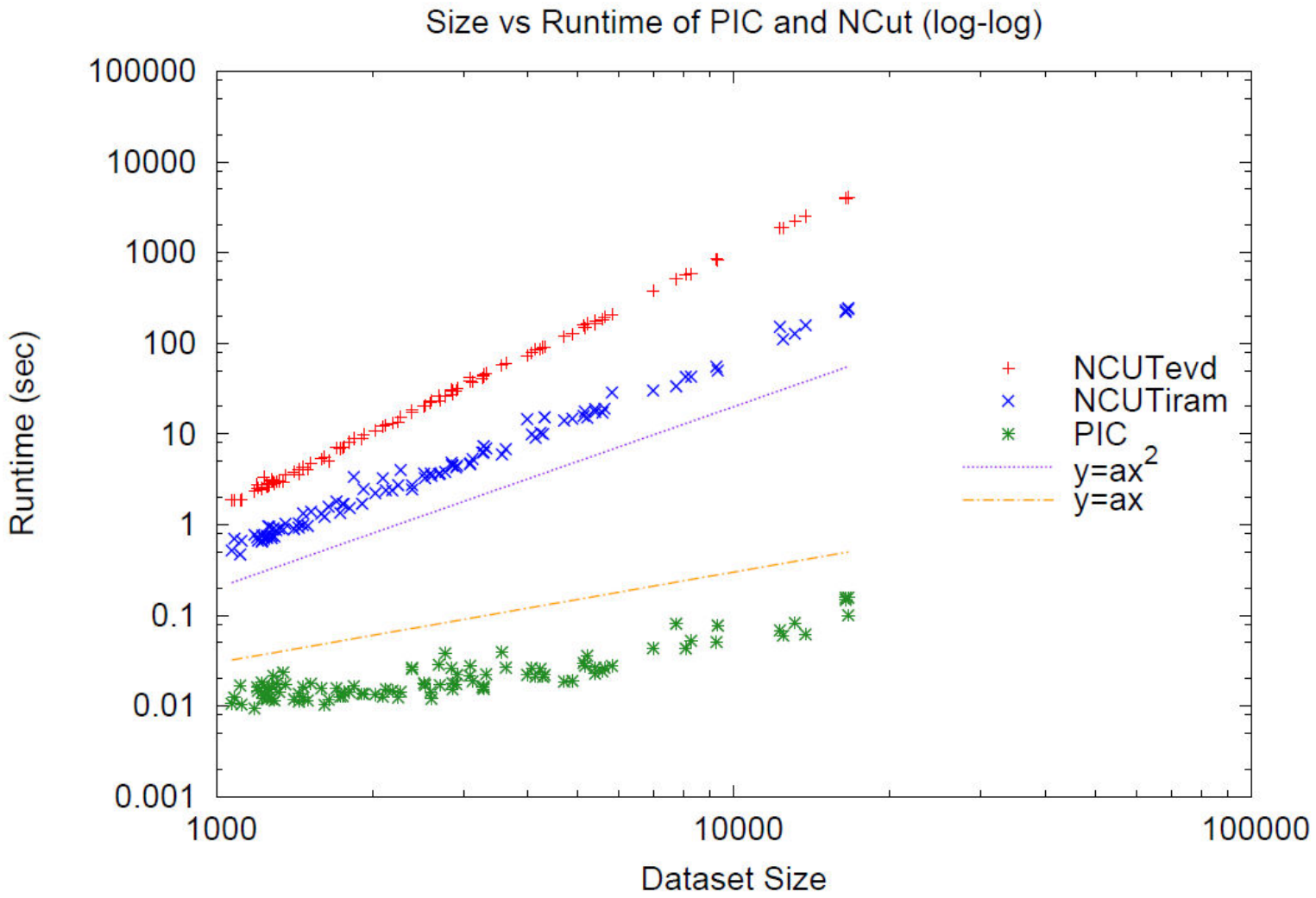
	ACC-Avg	NMI-Avg
baseline	57.59	-
k-means	69.43	0.2629
NCUTevd	**77.55**	**0.3962**
NCUTiram	61.63	0.0943
PIC	**76.67**	**0.3818**

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

Results

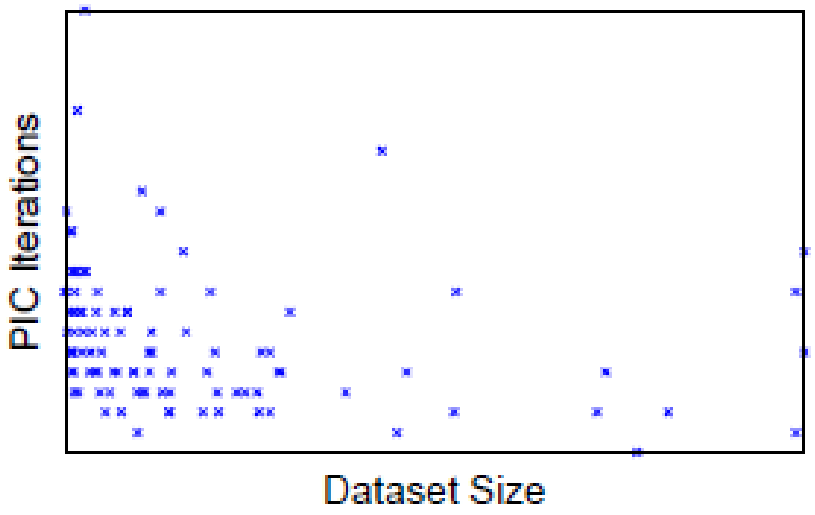

Accurac
PIC
100
95
90
85
80
75
70
65
60
55
50
50 55 60 65
Accuracy of NCUTevd vs PIC
PIC
NCUTevd
y=x
50 55 60 65 70 75 80 85 90 95 100
ccuracy of NCUTiram vs PIC
y=x
60 65 70 75 80 85 90 95 100
NCUTiram

Results

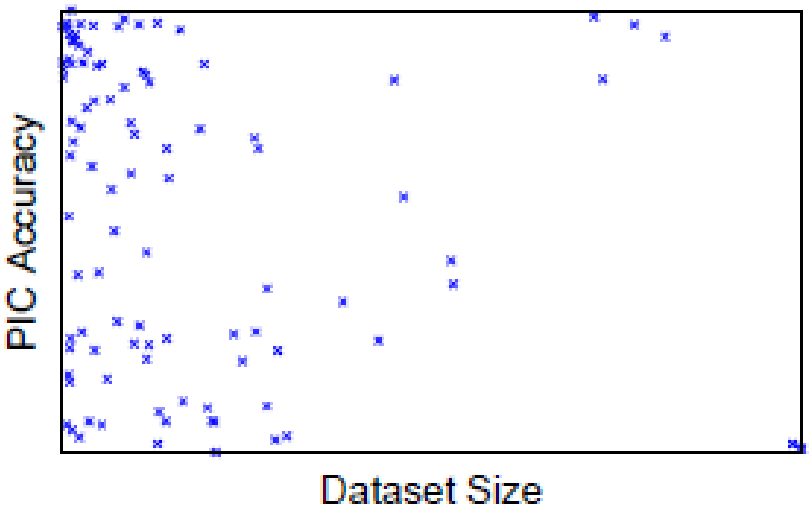

Size vs Runtime of PIC and NCut (log-log)
Runtime (sec)
100000
10000
1000
100
10
1
0.1
0.01
0.001
1000
10000
100000
Dataset Size
NCUTevd
NCUTiram
PIC
y=ax^2
y=ax

Results

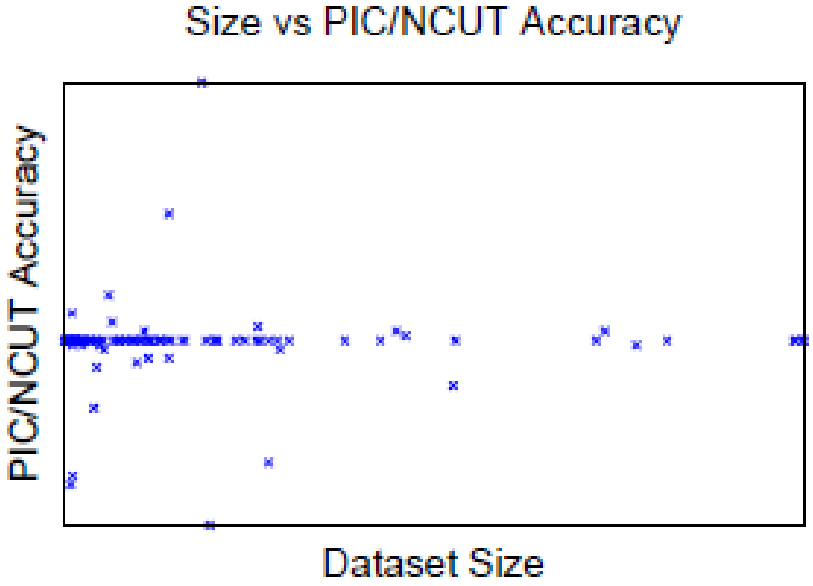
- Linear run-time implies *constant* number of iterations
- Number of iterations to "acceleration-convergence" is hard to analyze:
  - Faster than a single complete run of power iteration to convergence
  - On our datasets
    - 10-20 iterations is typical
    - 30-35 is exceptional



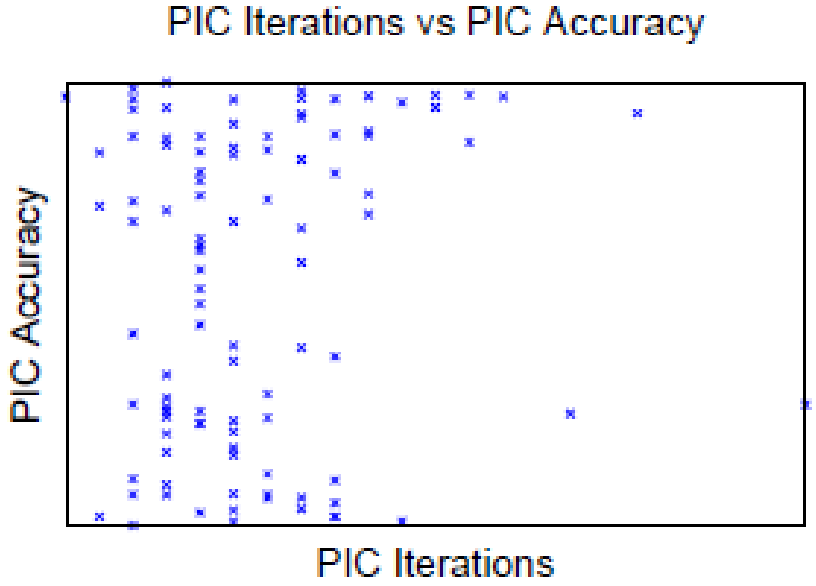
(a) $R^2 = 0.0424$



(b) $R^2 = 0.0552$



(c) $R^2 = 0.0007$



(d) $R^2 = 0.0134$

Thanks to…

- NIH/NIGMS
- NSF
- Microsoft LiveLabs
- Google