Divide-and-Merge Methodology for Clustering

D. Cheng, R. Kannan, S. Vempala, and G. Wang

Presented by: Arturo Donate

Outline

- Intro to data clustering
- Eigencluster algorithm
- Web search engine example
- Analysis
- Conclusion

Data Clustering

- Classification/labeling
- Input: data composed of elements
- Output: classification of elements into groups with similar objects
- Distance measure
- Machine learning, data mining, pattern recognition, statistical analysis, etc...

Cluster distance

- Distance between clusters given by distance measure
- Several measures available
  - euclidean
  - manhattan
  - mahalanobis

Data Clustering

- Hierarchical
  - Tree
  - Divisive vs Agglomerative
- Partitional
  - K-Means
- Spectral

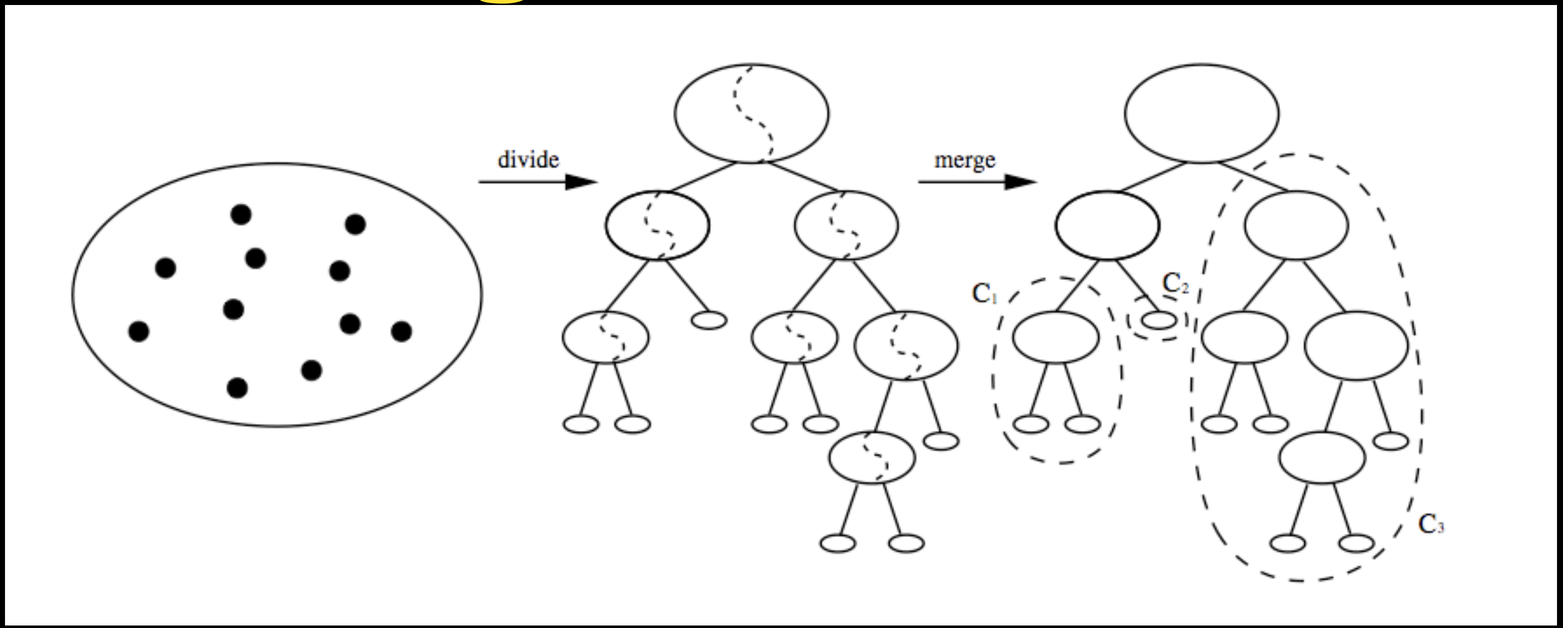
K-Means

- Randomized centroids (K groups)
- Object membership determined by distance to centroids
- Centroid location recalculated
- Repeated until convergence
- Fuzzy c-Means, QT clustering, etc...

Eigencluster

- Clustering algorithm using "divide-and-merge" approach
- Published in Journal of the ACM, 2004
- Combination of clustering approaches
- Used for web searches, but can be applied to any clustering problem
  - http://www-math.mit.edu/cluster/

Eigencluster



- Divide and Merge methodology
- Phase 1: divide data
- Phase 2: merge divided data

Divide Phase

- Create hierarchical clustering of data (tree)
- Input: set of objects w/ distances
- Algorithm recursively divides sets until singletons
- Output: tree with singleton leaves
    - internal nodes represent subsets
- Authors suggest spectral clustering

Spectral Clustering

- Input matrix A has objects as rows
- Uses similarity matrix (AAT)
  - similarity given by dot product:

$$a \cdot b = \sum_{i=1}^{n} a_i b_i = a_1 b_1 \times a_2 b_2 \times \ldots \times a_n b_n$$

  - sparse
    - knn, etc

Spectral Clustering

- Normalize sparse matrix
- Calculate second eigenvector
  - eigenvector defines “cut” on original matrix
  - cut based on: sign, mean, median, etc...

Divide Phase

- Main idea
  - divide an initial cluster into sub-clusters using spectral clustering
    - compute 2nd eigenvector of similarity matrix via power method
    - find best cut in n-1 possible cuts

Divide Phase

- Definitions:
  - Let $\rho \in R^n$ be a vector of the row sums of AAT
  - Let $\pi = \frac{1}{\Sigma_i \rho_i} \rho$
  - Let R be a diagonal matrix so that $R_{ii} = \rho_i$

Divide Phase

- Authors propose the use of the spectral algorithm in []
  - second largest eigenvalue of normalized similarity matrix B = R-1AAT
  - For efficiency, eigenvector is computed from symmetric matrix Q = DBD-1
  - Symmetric Q, power method

Divide Phase

- Power method steps:
  - let v be an arbitrary vector orthogonal to πTD-1
  - repeat:
    - normalize v ($v = v / ||v||$)
    - set $v = Qv$
- Converges in O(log n) (proof in paper)

Divide Phase

- Power method
  - used to estimate 2nd largest eigenvector
  - fast matrix-vector multiplication
  - $Q = DR^{-1}AA^{T}D^{-1}$
  - For $v = Qv$, perform four individual sparse matrix-vector multiplications (ie., $v = D^{-1}v$, etc...)

Divide Phase

- Problem:
  - spectral clustering requires normalized similarity matrix (for calculating $\rho \in R^n$)
  - expensive!
  - solution: do not compute explicitly

Divide Phase

- Rewrite row sums as:

$$\rho_i = \sum_{j=1}^{n} A_{(i)} \cdot A_{(j)}$$
$$= \sum_{j=1}^{n} \sum_{k=1}^{m} A_{ik} A_{jk}$$
$$= \sum_{k=1}^{m} A_{ik} \left(\sum_{j=1}^{n} A_{ik} \right)$$

- $\sum_{j=1}^{n} A_{ik}$ does not depend on i, so runtime is O(M), where M is # of nonzero entries

Divide Phase

- Current steps:
  - Let $\rho \in R^n$ be a vector of the row-sums of AA^T
  - Let $\pi = \frac{1}{\Sigma_i \rho_i} \rho$
  - Compute 2nd largest eigenvector v' of $Q = DR^{-1}AA^TD^{-1}$
  - Let $v = D^{-1}v'$, and sort v so $v_i < v_{i+1}$

Divide Phase

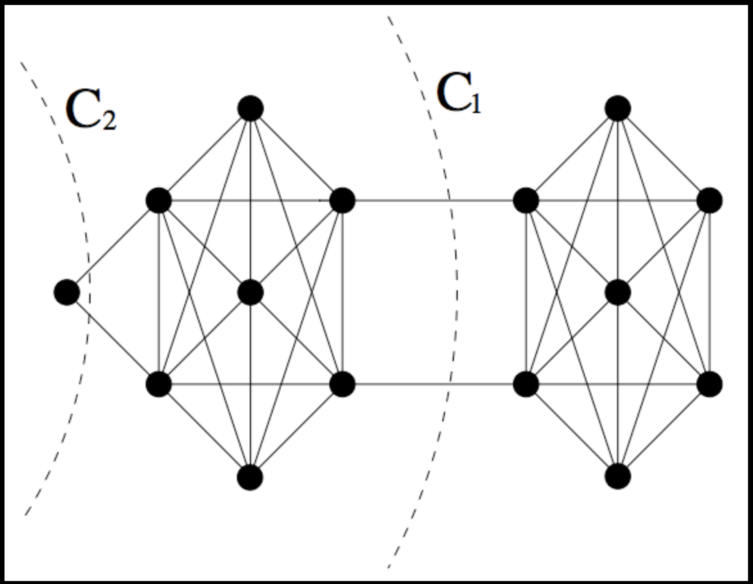
- N-dimensional eigenvector v defines n-1 possible cuts
- Original matrix is sorted according to v, and must be cut
- Find t such that the cut (S, T) = ({1, ..., t}, {t+1, ..., n}) minimizes the conductance across the cut

Divide Phase

- Best cut: min-conductance vs min-cut
  - min-cut = cut with minimum weight across it
    - assumes this means 2 resulting groups are least similar of possible cuts
  - problem: resulting cut may not provide best groups

Divide Phase

- Example: cut C2 may have minimum weight across 2 edges, but cut C1 provides better grouping



Divide Phase

- Conductance:
  - find a cut such that
    $$(S,T) = (\{1,\ldots,t\},\{t+1,\ldots,n\})$$
  - minimize: $$\phi(S,T) = \frac{c(S,T)}{\min(c(S),c(T))}$$
  - where $c(S,T) = \sum_{i\in S, j\in T} A_{(i)} \cdot A_{(j)}$
    $c(S) = C(S,\{1\ldots,n\})$

Divide Phase

- Conductance:
  - helps find cuts with approximately equal size
  - eg., t=2 vs t=n/2
    - t=2 yields large numerator, small denominator
    - larger overall fraction, not minimizing conductance

Divide Phase

- Complete divide algorithm:

Input: An $n \times m$ matrix A.
Output: A tree with the rows of A as leaves.

(1) Let $\rho \in \mathbb{R}^n$ be a vector of the row sums of AA^T, and $\pi = \frac{1}{(\sum_i \rho_i)}\rho$.
(2) Let R, D be diagonal matrices with $R_{ii} = \rho_i$, $D_{ii} = \sqrt{\pi_i}$.
(3) Compute the second largest eigenvector v' of $Q = DR^{-1}AA^T D^{-1}$.
(4) Let $v = D^{-1}v'$, and sort v so that $v_i \le v_{i+1}$.
(5) Find t such that the cut

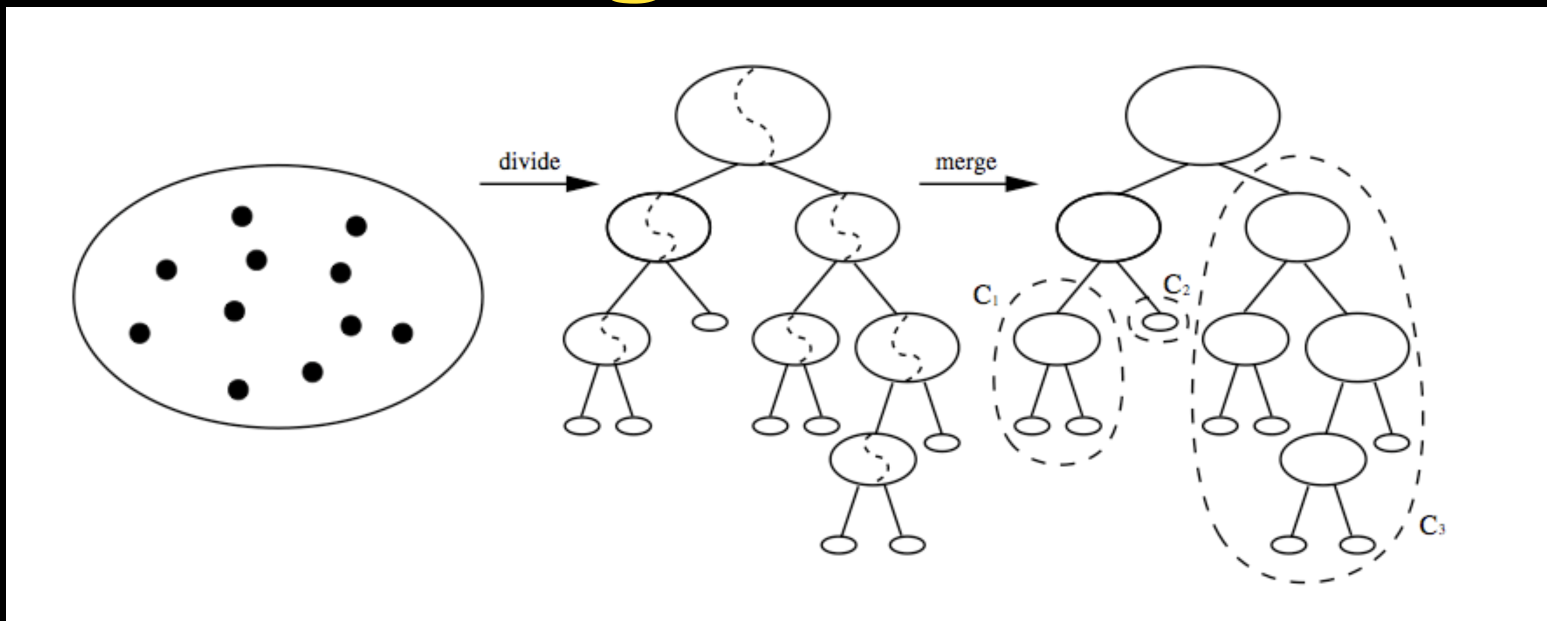
$$(S,T) = (\{1,\ldots,t\},\{t+1,\ldots,n\})$$

minimizes the conductance:

$$\phi(S,T) = \frac{c(S,T)}{\min(c(S),c(T))}$$

where $c(S,T) = \sum_{i \in S, j \in T} A_{(i)} \cdot A_{(j)}$, and $c(S) = C(S,\{1\ldots,n\})$.
(6) Let $\hat{A}_S$, $\hat{A}_T$ be the submatrices of A. Recurse (Steps 1-5) on $\hat{A}_S$ and $\hat{A}_T$.

Merge Phase



- Applied to tree produced by divide phase
- Idea: find best classification produced by divide phase

Merge Phase

- Input: hierarchical tree T
- Output: partition $C_1, \ldots, C_k$ where C_i is a node in T
- Dynamic program to evaluate objective function g
- Bottom up traversal: OPT for interior nodes computed by merging OPT in C_l, C_r

Merge Phase

- Properties of tree T:
  - each node is a subset of objects
  - L,R children form partition for parent
  - Clustering: subset S of nodes in T s.t. every leaf node is covered (leaf-root path encounters exactly 1 node in S)

Merge Phase

- Objective function g
  - describes optimal merge
  - choice of g may vary, crucial!
  - note: $g(C_{OPT})$ may not be OPT clustering
    - choice of g
    - OPT may not respect tree

Merge Phase

- K-Means objective function
  - k-clustering minimizing sum of squared distances of the pts in each cluster to the centroid p_i

$$g(\{C_1, \ldots, C_k\}) = \sum_i \sum_{u \in C_i} d(u, p_i)^2$$

  - pi = mean of points in a cluster Ci
  - NP-Hard!

Merge Phase

- K-Means objective functions
  - Let OPT-TREE(C,i) be optimal tree-respecting clustering for C with i clusters
  - OPT-TREE(C,1) = {C}
  - OPT-TREE(C,i) = OPT-TREE(C_l,j) ∪ OPT-TREE(C_r,i-j)
  - where j = $\text{argmin}_{i \le j < i}$ g(OPT-TREE(C_l,j) ∪ OPT-TREE(C_r,i-j))

Merge Phase

- Compute OPT clustering for leaf nodes first
- Interior nodes computed efficiently via dynamic programming
- OPT-TREE(root, k) gives optimal clustering of data
  - root = root node of divide phase tree

Merge Phase

- Min-diameter objective function
  - k-clustering minimizing max diameter
    - diameter - max distance between pair of objects in C_i
  - defined as:

$$g(\{C_1, \ldots, C_k\}) = \max_i \mathsf{diam}(C_i)$$

Merge Phase

- Min-sum objective function
  - minimize sum of pairwise distances within C_i
  - computed via dynamic program

$$g(\{C_1, \ldots, C_k\}) = \sum_{i=1}^{k} \sum_{u,v \in C_i} d(u, v)$$

  - approximation algorithms exist, but not useful in practice

Merge Phase

- Correlation clustering objective function
  - $G = \{V, E\}$; for each $e_i \in E$, e_i is red (similar vertices) or blue (dissimilar vertices)
  - find partition maximizing red edges within cluster, and blue edges between clusters

$$g(\{C_1 \ldots C_k\}) = \sum_i |\{(u,v) \in R \cap C_i\}| + \frac{1}{2}|\{(u,v) \in B : u \in C_i, v \in U \setminus C_i\}|$$

Merge Phase

- Time complexity
  - Divide
  - Merge
    - choice of g
    - iterations

Web Search

- Sample implementation: web search
- Typical search engine: linear rank
  - fails to show inherent correlation when ambiguity is present (ie, “Mickey” - Rooney, Mantle, Mouse, ...)

Web Search

- Input query: retrieve 400 results from Google
  - title, location, snippet
- Construct document-term matrix:

D_1 = "You like Bob"
D_2 = "You hate hate Bob"

	You	like	hate	Bob
D_1	1	1	0	1
D_2	1	0	2	1

Web Search

- Divide phase - spectral algorithm
- Merge phase - relaxed correlation clustering
  - similar to correlation clustering but relaxed components α, β remove dependency on predefined k

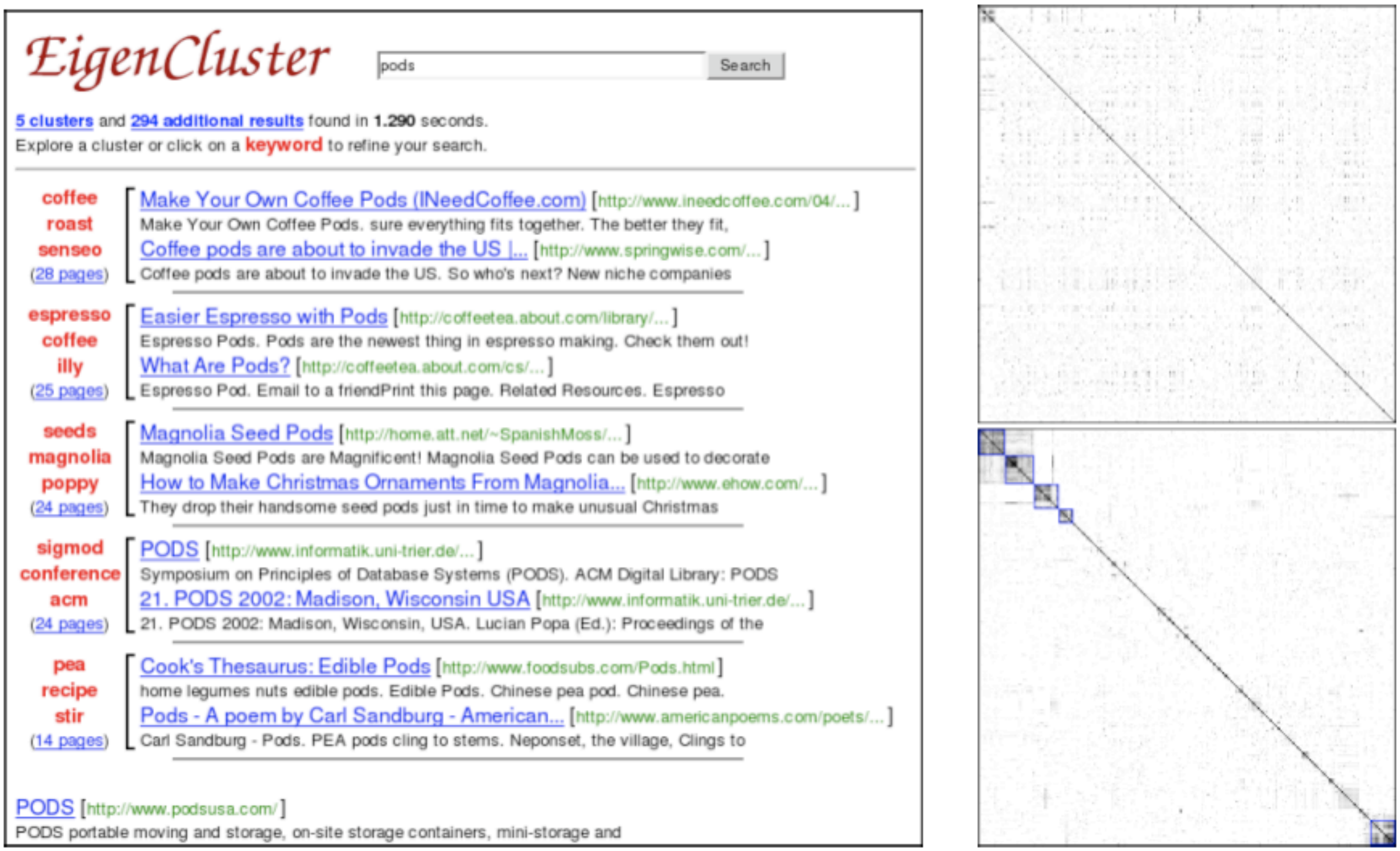
Web Search

- Relaxed correlation objective function:

$$\sum_i \alpha \left(\sum_{u,v \in C_i} 1 - A_{(u)} \cdot A_{(v)} \right) + \beta \left(\sum_{u \in C_i, v \notin C_i} A_{(u)} \cdot A_{(v)} \right)$$

- first component: dissimilarity within cluster
- second component: similarity failed to be captured
- eigencluster: $\alpha = 0.2$, $\beta = 0.8$

Web Search

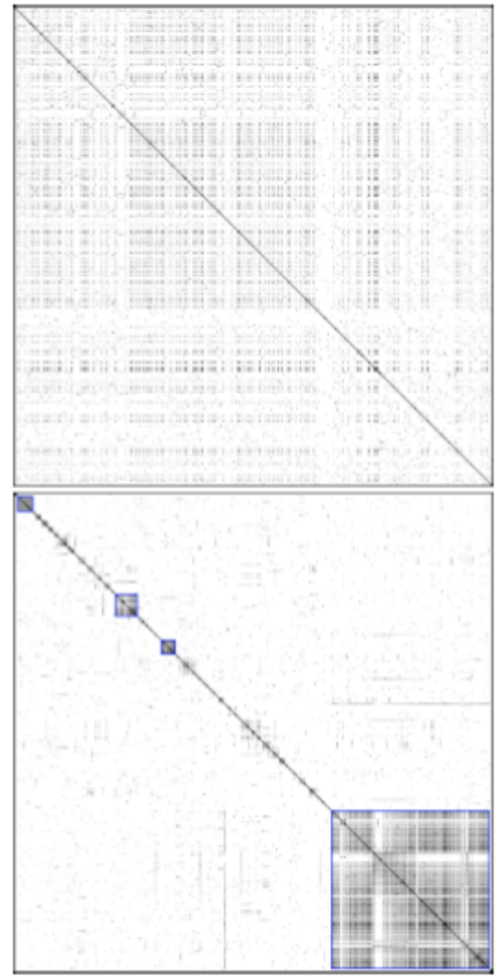


(a) Query: pods

(b) Before/after: pods

Web Search

(c) Query: mickey



(d) Before/after: mickey

Analysis

- Experiment on Boley dataset
  - 185 web pages
  - 10 classes
  - different objective functions
  - quality of results measured by entropy
    - randomness within cluster
    - lower value

Web Search

	J1	J2	J3	J4	J5	J6	J7	J8	J9	J10	J11
k-means	1.00	0.93	0.88	0.81	1.00	0.93	0.84	0.83	0.95	0.71	1.07
min-sum	0.98	0.93	0.88	0.78	0.98	0.92	0.84	0.83	0.94	0.71	1.10
min-diam	1.04	1.10	0.96	1.04	1.10	1.00	1.05	1.23	1.24	0.83	1.16
best in tree	0.98	0.93	0.88	0.78	0.96	0.91	0.84	0.83	0.92	0.71	1.05

- k-means and min-sum typically outperform min-diam
- 7 of 11 - k-means or min-sum found best possible clustering

```
Terminal — bash — 95×31
arturombp:eigencluster arturodonate$ ./main data2 2
8 vectors read, each with 2 elements

A:
    1    45
   87     5
   32     1
    9    51
   61    11
    2    43
   98    10
   10    89

pi: 0.0759555 0.176584 0.0635704 0.101306 0.136139 0.0746009 0.206096 0.165747

R:
11775      0      0      0      0      0      0      0
    0  27375      0      0      0      0      0      0
    0      0   9855      0      0      0      0      0
    0      0      0  15705      0      0      0      0
    0      0      0      0  21105      0      0      0
    0      0      0      0      0  11565      0      0
    0      0      0      0      0      0  31950      0
    0      0      0      0      0      0      0  25695

D:
0.2756      0      0      0      0      0      0      0
    0 0.42022      0      0      0      0      0      0
    0      0 0.25213      0      0      0      0      0
    0      0      0 0.31829      0      0      0      0
    0      0      0      0 0.36897      0      0      0
```

Terminal — bash — 95×31

```
    0     0     0     0     0     0     0 25695

D:
0.2756     0     0     0     0     0     0     0
    0 0.42022     0     0     0     0     0     0
    0     0 0.25213     0     0     0     0     0
    0     0     0 0.31829     0     0     0     0
    0     0     0     0 0.36897     0     0     0
    0     0     0     0     0 0.27313     0     0
    0     0     0     0     0     0 0.45398     0
    0     0     0     0     0     0     0 0.40712

Q:
0.17206 0.017378 0.007148 0.16943 0.03527 0.16599 0.028253 0.23082
0.017378 0.27741 0.1698 0.050061 0.22308 0.021862 0.28998 0.049582
0.007148 0.1698 0.10401 0.027249 0.13611 0.010023 0.17729 0.025702
0.16943 0.050061 0.027249 0.17077 0.060969 0.16406 0.062142 0.23043
0.03527 0.22308 0.13611 0.060969 0.18204 0.038085 0.23445 0.0682350.03527 0.22308 0.13611 0.060
0.16599 0.021862 0.010023 0.16406 0.038085 0.16022 0.032566 0.22316
0.028253 0.28998 0.17729 0.062142 0.23445 0.032566 0.30372 0.065265
0.23082 0.049582 0.025702 0.23043 0.068235 0.22316 0.065265 0.31216

EVECTOR:  -0.2756 -0.420219 -0.252132 -0.318286 -0.368971 -0.273132 -0.453978 -0.407121

ORDER:  0 5 3 7 4 1 2 6

A (SORTED):
    1    45
    2    43
    9    51
   10    89
```

Terminal — bash — 95×31

```
A (SORTED):
    1    45
    2    43
    9    51
   10    89
   61    11
   87     5
   98    10
   32     1

CONDUCTANCE: 0.827941 0.880868 0.683908 0.534404 0.740679 0.571273 0.696275

SPLIT: 3

S:
    1    45
    2    43
    9    51
   10    89

S_mean:
5.5 57

T:
   61    11
   87     5
   32     1
   98    10

T_mean:
69.5 6.75
```

Conclusion

- clustering based on divide-merge
- idea: divide groups data hierarchically, merge finds best cluster within
- divide phase: spectral clustering
- merge: objective functions