# Principal Direction Partitioning in Data Mining

**Daniel Boley**
**Computer Science and Engineering**
**University of Minnesota**

http://www.cs.umn.edu/~boley/PDDP.html
*Supported in part by NSF*



## Outline

- Practice of Data Mining
- Divisive Partitioning for Unsupervised Clustering
- Related Methods
- Algorithmic Issues – Fast Lanczos Solver
- Experimental Results
- Linear Algebra elsewhere in Data Exploration
- Conclusions and Future Work

# Practice of Data Mining

- Data Explosion
  - Commercial & Gov't databases
  - Scientific data: Space, Satellite, Simulations.
  - WWW had 200 M web pages in 1997, 800 M in 1999.

- Search through commercial transactions:
  - Find patterns in buying habits
  - Predict where to focus marketing efforts

- Organize scientific data
  - Extract & Save only "interesting parts" of PDE simulations
  - Classify many individual data samples (stars, terrains, etc.)

- Aid in searching WWW & organizing what is found.



# Divisive Partitioning for Unsupervised Clustering

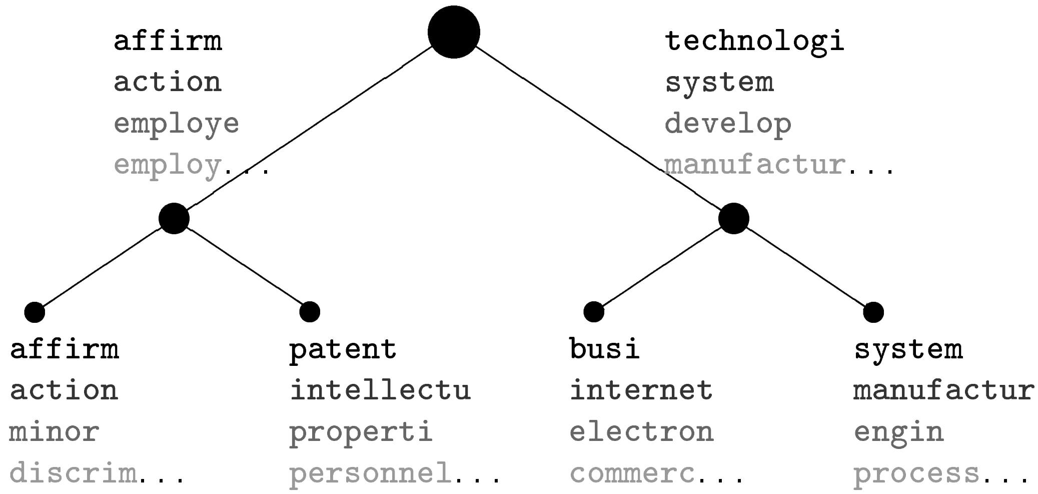
- Unsupervised, as opposed to Supervised:
  - no predefined categories;
  - no previously classified training data;
  - no a-priori assumptions on the number of clusters.

- Top-down Hierarchical:
  - imposes a tree hierarchy on unstructured data;
  - tree is source for some taxomonic information for dataset;
  - tree is generated from the root down.

- Principal Direction Divisive Partitioning
  - operates on real-valued data, even with missing data;
  - embedded in high dimensional Euclidean space;
  - fast & scalable by using efficient Lanczos solver.

# Principal Direction Divisive Partitioning

- Start with root cluster representing all the documents.
- Split the root cluster into two children clusters.
- Recursively split each leaf cluster into two children
- Stop when stopping test satisfied.






# Data Representation for Linear Algebra Methods

- Each document represented by $n$-vector $\mathbf{d}$ of word counts.
- Vectors assembled into Term Frequency Matrix $\mathbf{M} = (\, \mathbf{d}_1 \quad \cdots \quad \mathbf{d}_m \,)$.

| | Quake Risk High | Closes For Snow | Rose Bowl Result | Big 10 Sanctions | Housing Crunch |
|---|---|---|---|---|---|
| berkeley | 1 | 0 | 0 | 0 | 2 |
| stanford | 3 | 0 | 2 | 0 | 2 |
| minnesota | 0 | 2 | 0 | 1 | 0 |
| wisconsin | 0 | 2 | 2 | 1 | 0 |
| ucla | 1 | 0 | 0 | 0 | 1 |
| caltech | 1 | 0 | 1 | 0 | 1 |

- Other attribute valuess can also be used.

# Divisive Partitioning

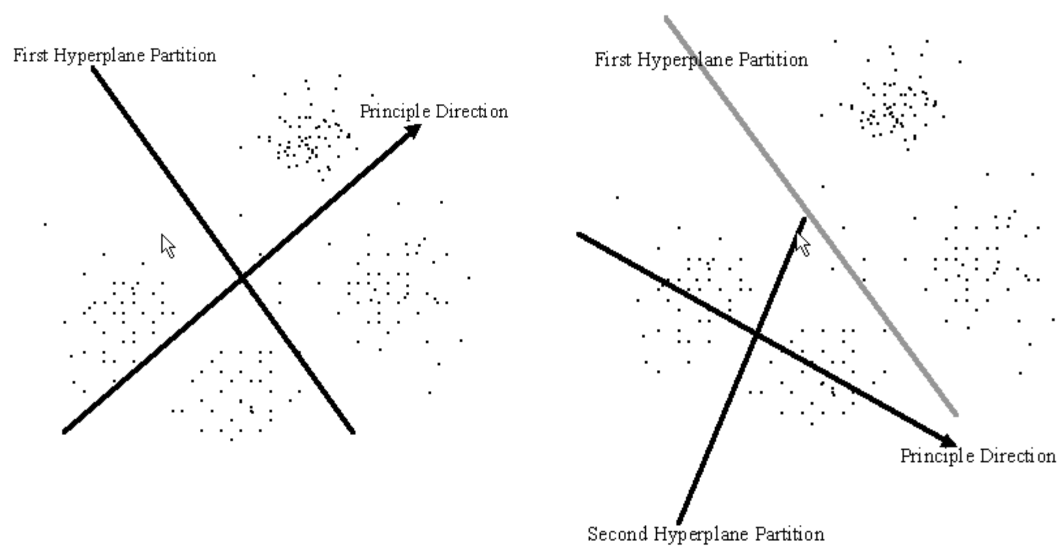
- Each document represented by $n$-vector $\mathbf{d}$ of word counts.
- Each $\mathbf{d}$ scaled to $\|\mathbf{d}\| = 1$ to make independent of document length.
- Vectors assembled into Term Frequency Matrix $\mathbf{M} = (\, \mathbf{d}_1 \quad \cdots \quad \mathbf{d}_m \,)$.

**Splitting Process:**

- Get leading principal direction $\mathbf{u}$ of $\mathbf{M} - \mathbf{w}\mathbf{e}^T$ with SVD, where $\mathbf{w} \triangleq \frac{1}{m}\mathbf{M}\mathbf{e} =$ centroid, $\mathbf{e} \triangleq (1 \cdots 1)^T$.
- Split documents by value of projection $\mathbf{u}^T(\mathbf{d}_j - \mathbf{w})$, $j = 1, 2, \cdots$.
- Repeat recursively on each set of documents.



# Divisive Partitioning - Splitting Step



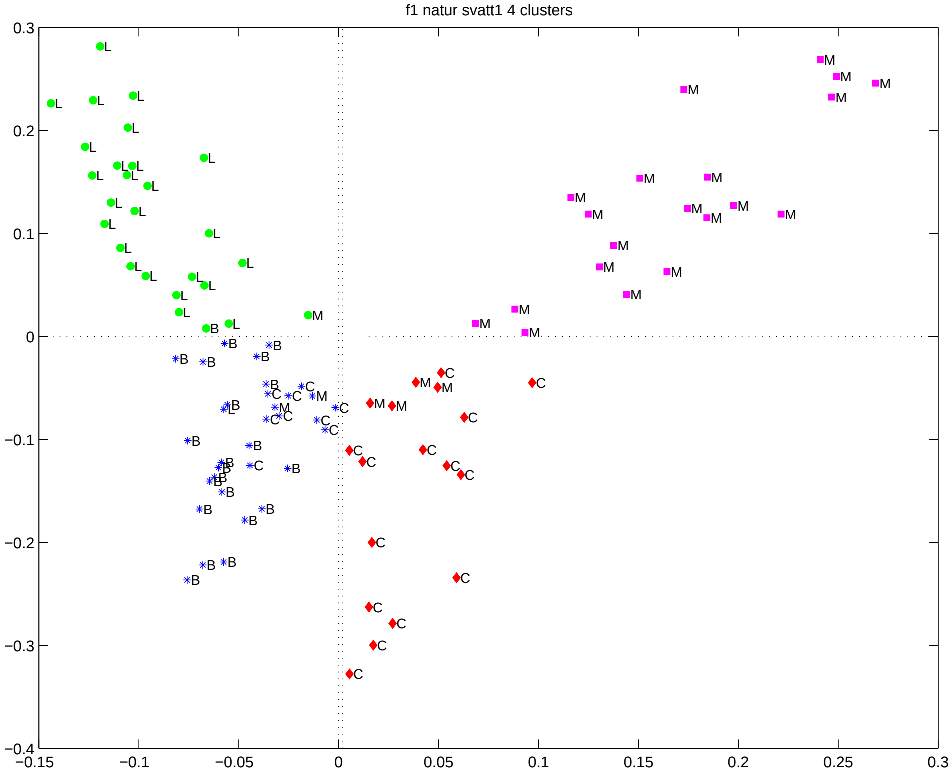

*Two Total Clusters* *Three Total Clusters*

## Divisive Algorithm

0. **Start** with $n \times m$ matrix $\mathbf{M}$ of (scaled) document vectors.
1. **Initialize** Binary Tree with a single Root Node.
2. **For** $c = 2, 3, \ldots,$ **do**
3.     **Select** node K with largest *cluster scatter* value.
4.     **Compute** principal direction $\mathbf{u}$.
5.     **Set** *indices*(L) := indices of the non-positive entries in $\mathbf{u}$.
6.     **Set** *indices*(R) := indices of the positive entries in $\mathbf{u}$.
7.     **Put** documents L into left child, R in to right child.
8.     **Compute** *centroid scatter* of collected cluster centroids.
9. **until** *centroid scatter* exceeds largest *cluster scatter*.
10. **Result:** A binary tree with leaf nodes forming a partitioning of the entire data set.



## Document Clusters



## Related Methods – Principal Component Analysis

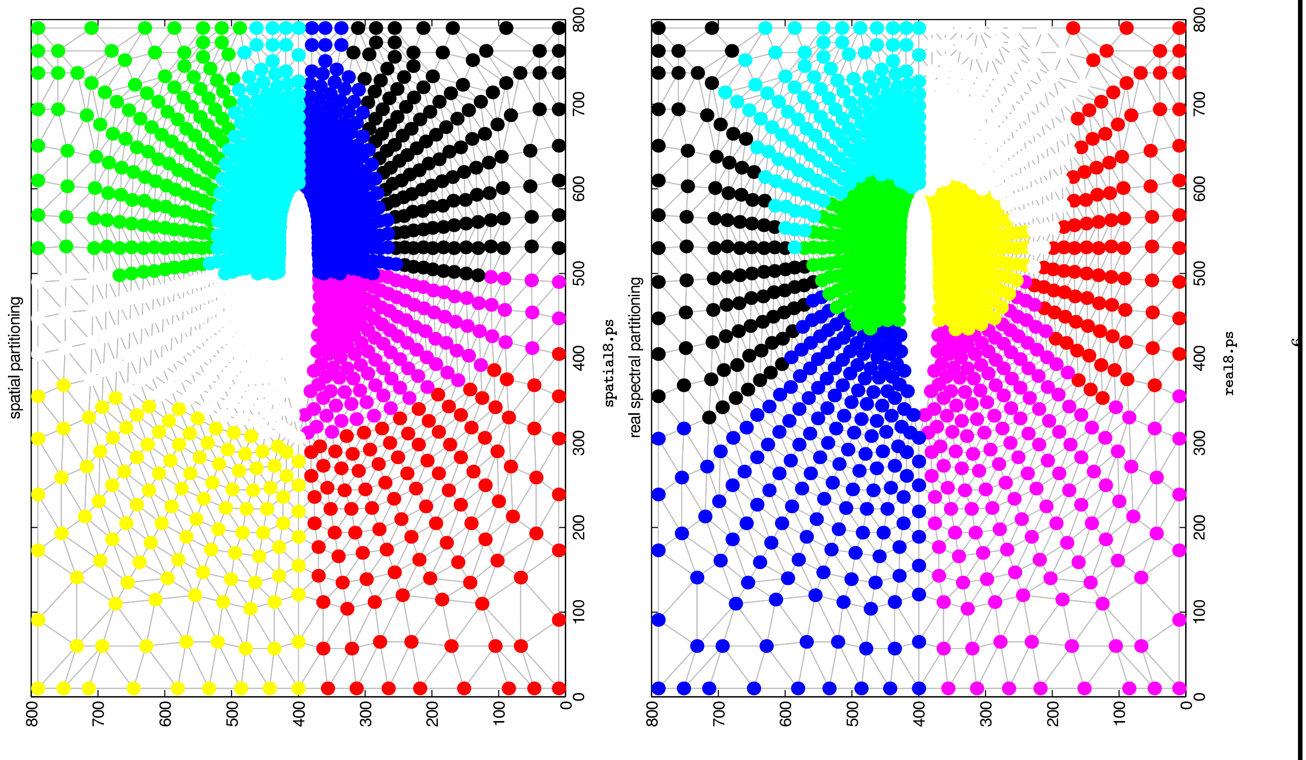
- PCA shifts the documents by their mean: $\mathbf{M} \to \mathbf{M} - \mathbf{e}\mathbf{w}^T$ where $\mathbf{e} = (1 \quad \cdots 1)^T$, $\mathbf{w} =$ centroid.
- Then select best rank $k$ approximation to $\mathbf{M} - \mathbf{e}\mathbf{w}^T$.
- Result: original data represented with fewer degrees of freedom.
- Like LSI, get vectors giving inter-word relationships.
- PDDP computes just first eigenvector.



## Related Methods – Spectral Graph Partitioning

- $\mathbf{A} \triangleq$ Laplacian: diagonal entry $a_{ii} \triangleq$ degree of $v_i$, and $a_{ij} \triangleq -1$ iff there is an edge between vertex $i \Longleftrightarrow$ vertex $j$.
- Smallest eigenvalue is zero; Fiedler vector is eigenvector corresponding to next smallest eigenvalue.
  Split vertices according to sign of Fiedler vector entry.
- Get same split applying PDDP to $sI - A$ for $s > \lambda_{\text{max}}$.
  Same eigenvector algorithm, same convergence rate: eigenvalue distribution much less favorable than for text documents.

# Spatial vs Spectral Graph Partitioning






# Algorithmic Issues – Fast Lanczos Solver

- Total cost dominated by cost of finding principal direction.
- Use efficient sparse matrix eigensolver "Lanczos".
- Matrix used only to form matrix-vector products.
- Convergence depends on distribution of eigenvalues.
- On matrices of word counts from document sets, convergence appears to be fast ($\sim 20$ iterations).
- Cost to find first principal direction:
  Lanczos iters · mat-vec products per iter · cost of mat-vec product
  $\sim 20$ · 2 · fill fraction $\cdot m \cdot n$.
- Subsequent principal directions are cheaper [fewer documents].

## Symmetric Lanczos Recursion

- $\mathbf{A}\mathbf{X}_p = \mathbf{X}_p\mathbf{T}_p + \mathbf{x}_{p+1}\mathbf{e}_p^T t_{p+1,p}$

  where $\mathbf{T}_p = (\, t_{ij}\,)_{p\times p}$, symmetric & tridiagonal,

  and $\mathbf{X}_p = [\mathbf{x}_1, \ldots, \mathbf{x}_p]$ is the $n \times p$ matrix of Lanczos vectors.

**Traditional Termination Condition – use eigenvector:**

- Let $\lambda_p, \mathbf{v}_p$ be leading eigenpair of $\mathbf{T}_p$. Then

$$\mathbf{A}\mathbf{X}_p\mathbf{v}_p = (\mathbf{X}_p\mathbf{T}_p + \mathbf{x}_{p+1}\mathbf{e}_p^T t_{p+1,p})\mathbf{v}_p = \lambda_p\mathbf{X}_p\mathbf{v}_p + \boxed{t_{p+1,p}v_{pp}}\,\mathbf{x}_{p+1},$$

- Stop when $\boxed{t_{p+1,p}v_{pp}}$ is small.

**Simplified Termination Condition – use eigenvalue:**

- Interlacing property implies $\lambda_p \ge \lambda_{p-1}$ in exact arithmetic.
- Stop when $\boxed{\lambda_p \le \lambda_{p-1}}$, or alternatively when $|\lambda_p - \lambda_{p-1}|$ is small.



## Lanczos Algorithm

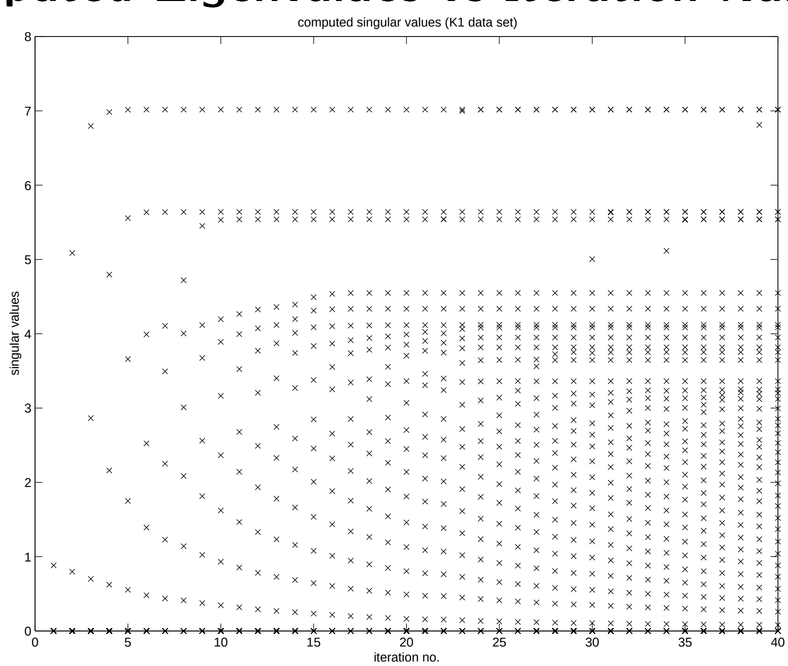
0. **Start** with $m \times m$ symmetric matrix $\mathbf{A}$ and starting vector $\mathbf{x}_1$.
1. **For** $p = 1, 2, 3, \ldots$ **do**
2.     **Set** $\hat{\mathbf{x}} = \mathbf{A}\mathbf{x}_p$      *mat-vec product: most costly step*
3.     **If** $p > 1$, **set** $\hat{\mathbf{x}} = \hat{\mathbf{x}} - t_{p-1,p}\,\mathbf{x}_{p-1}$
4.     **Set** $t_{pp} = \mathbf{x}_p^T\hat{\mathbf{x}}$
5.     **Set** $\lambda_p = \max\{\mathrm{eig}(\mathbf{T})\}$      *no eigenvector needed here*
6.     **If** $\lambda_p \le \lambda_{p-1}$, **set** $p = p - 1$; **break**
7.     **Set** $\hat{\mathbf{x}} = \hat{\mathbf{x}} - t_{pp}\mathbf{x}_p$
8.     **Set** $t_{p+1,p} = t_{p,p+1} = \|\hat{\mathbf{x}}\|$
9.     **If** $t_{p+1,p} \le \texttt{tol}$, **break**
10.     **Set** $\mathbf{x}_{p+1} = \hat{\mathbf{x}}\,/\,t_{p+1,p}$
11. **Set** $\mathbf{w} = [\mathbf{x}_1, \ldots, \mathbf{x}_p] \times$ [leading eigenvector of $\mathbf{T}$]
12. **Result:** eigenpair $\lambda_p, \mathbf{w}$.

## Adapt Lanczos Algorithm – Choices

- Low accuracy needed: use $\mathbf{A} \triangleq \mathbf{M}\mathbf{M}^T$ or $\mathbf{M}^T\mathbf{M}$ for simplicity.
- No reorthogonalization to get speed.
- Spurious eigenvalues always in interior – can ignore.
- Simple "eigenvalue only" stopping test.
- Could use Sturm sequences to get leading eigenvalue fast (or other recent fast solver)
- Save computation of eigenvectors until end.
- Lanczos vectors used only at end for eigenvectors.



## Computed Eigenvalues vs Iteration Number



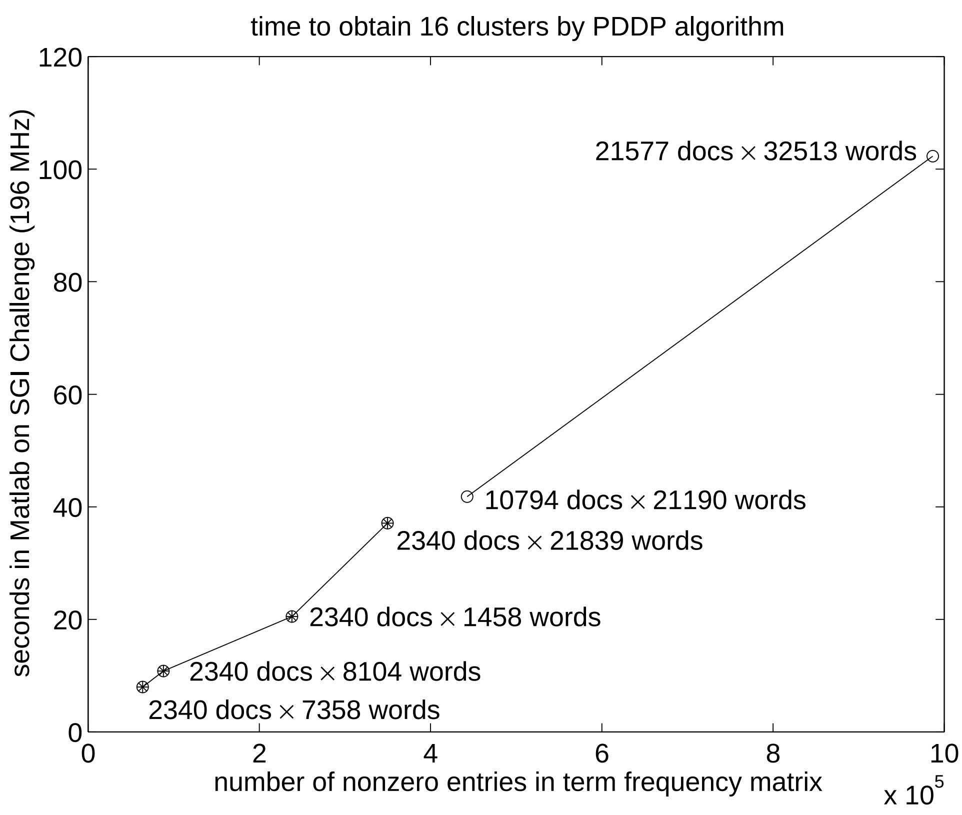

SVplot.eps

## Experimental Results: Document Test Sets

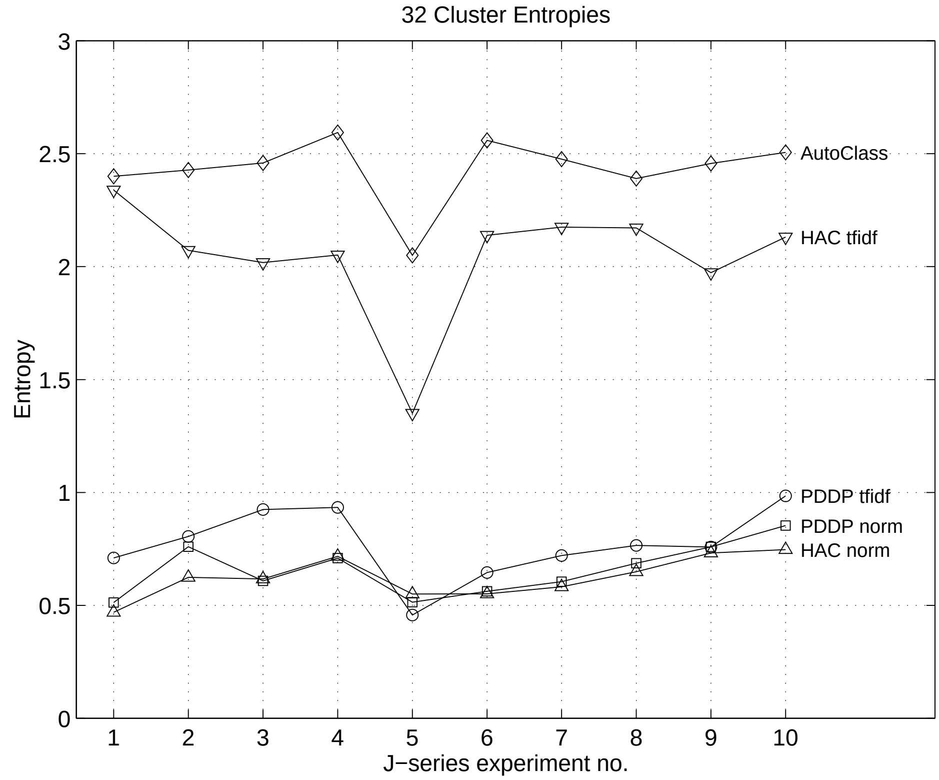
| *Exp #* | *Term Frequency Matrix Size* *F-series* | *J-series* | *K-series* | *Selection Criteria* |
|---|---|---|---|---|
| 1 | 98 × 5623 | 185 × 10536 | 2340 × 21839 | all words |
| 2 | 98 × 619 | 185 × 946 | 2340 × 7358 | quantile filtering |
| 3 | 98 × 1239 | 185 × 1763 | 2340 × 8104 | top 20+ words |
| 4 | 98 × 1432 | 185 × 2951 | | top 5+ words plus emphasized words |
| 5 | 98 × 399 | 185 × 449 | 2340 × 1458 | frequent item sets |
| 6 | 98 × 2641 | 185 × 5106 | | all with TF > 1 |
| 7 | 98 × 1004 | 185 × 1328 | | top 20+ & TF > 1 |
| 8 | 98 × 827 | 185 × 1105 | | top 15+ & TF > 1 |
| 9 | 98 × 622 | 185 × 805 | | top 10+ & TF > 1 |
| 10 | 98 × 332 | 185 × 474 | | top 5+ & TF > 1 |
| Reuters-21578 | | | 21577 × 32513 | all documents |
| Reuters-21578 | | | 10794 × 21190 | docs w/ topic labels |



## Speed on Text Documents



# Quality on Text Documents





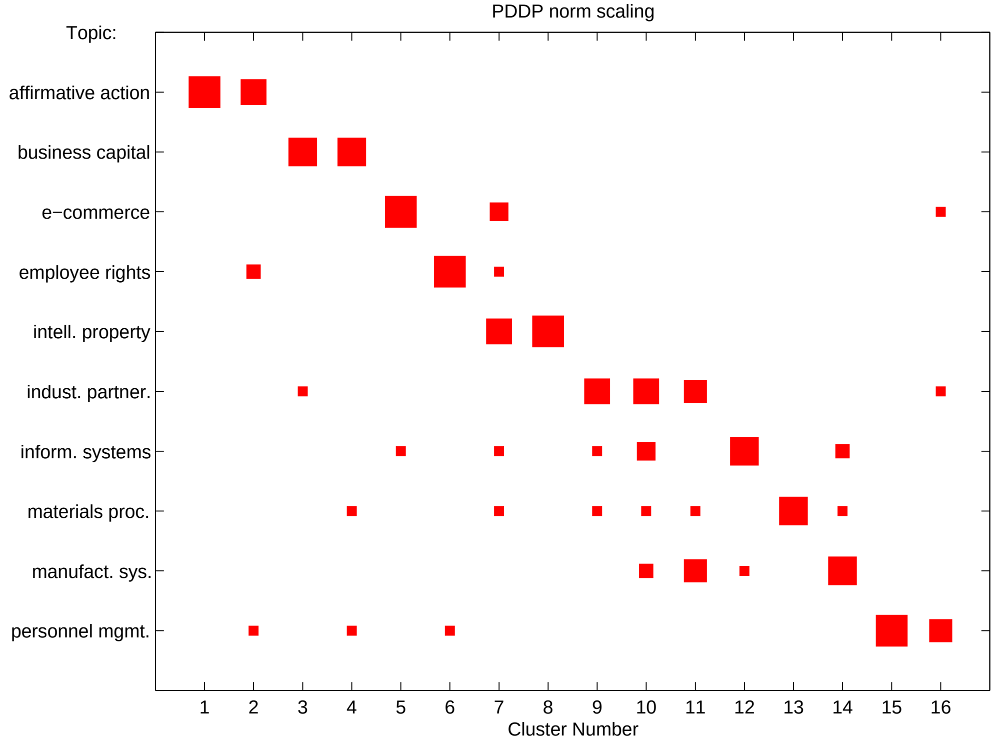

# Cluster Contents

| *cluster:* | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| business | 90 | 0 | 0 | 0 | 7 | 0 | 5 | 12 | 0 | 6 | 0 | 1 | 18 | 3 | 0 | 0 |
| health | 0 | 150 | 166 | 171 | 3 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 |
| politics | 2 | 0 | 0 | 0 | 100 | 1 | 2 | 0 | 0 | 1 | 0 | 2 | 1 | 5 | 0 | 0 |
| sports | 0 | 0 | 0 | 0 | 1 | 62 | 35 | 0 | 0 | 1 | 0 | 0 | 0 | 42 | 0 | 0 |
| techno. | 8 | 0 | 0 | 0 | 0 | 1 | 14 | 24 | 0 | 8 | 0 | 1 | 4 | 0 | 0 | 0 |
| entertain. | 24 | 0 | 0 | 4 | 11 | 4 | 22 | 61 | 135 | 131 | 148 | 159 | 143 | 137 | 204 | 206 |

⇑ *topic*

⇑ *number of documents of each topic in each cluster*

## Cluster Distribution






## Experiment: Search for MBTE on Altavista

Found 222 documents, clustered as follows:

```
      CENTROID WORDS
62:found.serv.request.url.alt.html.fram.pleas.http.fil.de.ww
44:inc.servi.corp.ttm.compan.com.sit.stock.fre.pri.mrq.syste
38:fuel.car.gasolin.vehic.rav.re.com.gas.engin.pri.messag.su
78:wat.mtb.environmental.air.health.gasolin.program.sit.cali
      PRINCIPAL DIRECTION WORDS
62:found.serv.url.request.html.pleas.apach.fil.port.htm.http
44:inc.corp.ttm.ltd.servi.corpor.international.mrq.stock.fin
38:rav.car.fuel.tir.subject.toyota.vehic.driv.wd.engin.com.h
78:wat.mtb.environmental.health.california.air.gasolin.clean
```

## Linear Algebra elsewhere in Data Exploration

- Latent Semantic Indexing (*Anderson, Berry, Dumais, ...*).
  - Find documents best matching a query, by e.g. angle.
  - Replace $\mathbf{M}$ with low rank version to reduce noise.
- Linear Least Squares Fit (*Yang, Chute* – MEDLINE).
  - Have 2nd matrix $\mathbf{N}$ of predefined categories for each document.
  - Train by finding best fit: $\text{minimize}_{\mathbf{W}} \|\mathbf{WM} - \mathbf{N}\|_F$.
- Hub & Authority of Web Pages from Link Structure (*Kleinberg*).
  - Authority/hubness weighted by incoming/outgoing links.
  - Propagate weights, much like simulating a Markov chain.
- Surface matching from images (*Tomasi, Kriegman, ...*).
  - Get leading singular vectors from many images of same surface.
  - Use to match queries (e.g. recognize building, face, etc.).



## Conclusions

- Unsupervised Clustering: get structure on large unstructured datasets.
- PDDP exhibits good scalability properties.
- PDDP generates clusters of high quality,
  comparable to other methods.
- PDDP identifies the distinctive features of the individual clusters.
- PDDP can be applied to non-text data.
- PDDP needs a self-contained, portable implementation.

## Future Work

- Applications:
  - Organize Alcohol Laws for Minn. Health Dept. study
  - Classify speech recognition errors left over after all other processing.
  - Image data: classification or anomaly detection
  - Minnesota Sky Survey.

- Method Development
  - Two principal directions at a time (4-way split?).
  - Re-agglomerate clusters wrongly chopped by hyperplane.
  - Adjust hyperplanes during course of partitioning.
  - Study statistical significance of separation based on direction of maximal variance.
  - Handle datasets too big to fit in memory.