Clustering IV

Outline

- Impossibility theorem for clustering
- Density-based clustering and subspace clustering
- Bi-clustering or co-clustering

General form of impossibility results

- Define a set of simple **axioms** (properties) that a computational task should satisfy

- Prove that ***there does not exist an algorithm*** that can simultaneously satisfy all the axioms → **impossibility**

Computational task: clustering

- A ***clustering function*** operates on a set **X** of **n** points. **X = {1,2,…,n}**

- Distance function **d: X** x **X → R** with **d(i,j)≥0**, **d(i,j)=d(j,i)**, and **d(i,j)=0** only if **i=j**

- Clustering function **f**: **f(X,d) = Γ**, where **Γ** is a ***partition*** of **X**

Axiom 1: Scale invariance

- For **a>0**, distance function ***ad*** has values ***(ad)(i,j)=ad(i,j)***

- For any **d** and for any **a>0** we have **f(d) = f(ad)**

- The clustering function should not be sensitive to the changes in the units of distance measurement – should not have a built-in “length scale”

Axiom 2: Richness

- The ***range*** of **f** is equal to ***the set of partitions*** of **X**

- For any **X** and any partition **Γ** of **X**, there is a distance function on **X** such that **f(X,d) = Γ**.

Axiom 3: Consistency

- Let **Γ** be a partition of **X**
- **d, d'** two distance functions on **X**
- **d'** is a **Γ**-transformation of **d**, if
  - For all **i,jϵ X** in the ***same cluster*** of **Γ**, we have **d'(i,j)≤d(i,j)**
  - For all **i,jϵ X** in ***different clusters*** of **Γ**, we have **d'(i,j)≥d(i,j)**
- **Consistency:** if **f(X,d)= Γ** and d' is a **Γ**-transformation of **d**, then **f(X,d')= Γ**.

Axiom 3: Consistency

- **Intuition:** Shrinking distances between points inside a cluster and expanding distances between points in different clusters does not change the result

Examples

- Single-link agglomerative clustering
- Repeatedly merge clusters whose closest points are at minimum distance
- Continue until a stopping criterion is met
  - ***k***-cluster stopping criterion: continue until there are k clusters
  - distance-***r*** stopping criterion: continue until all distances between clusters are larger than r
  - scale-***a*** stopping criterion: let d* be the maximum pairwise distance; continue until all distances are larger than ad*

Examples (cont.)

- Single-link agglomerative clustering with ***k***-cluster stopping criterion does not satisfy richness axiom

- Single-link agglomerative clustering with distance-***r*** stopping criterion does not satisfy scale-invariance property

- Single-link agglomerative clustering with scale-***a*** stopping criterion does not satisfy consistency property

Centroid-based clustering and consistency

- **k**-centroid clustering:
  - **S** subset of **X** for which $\sum_{i \in X} \min_{j \in S}\{d(i,j)\}$ is minimized
  - Partition of **X** is defined by assigning each element of **X** to the centroid that is the ***closest*** to it

- **Theorem:** for every **k≥2** and for **n** sufficiently large relative to **k**, the **k**-centroid clustering function does not satisfy the consistency property

k-centroid clustering and the consistency axiom

- Intuition of the proof
- Let **$k=2$** and **X** be partitioned into parts **Y** and **Z**
- **$d(i,j) \leq r$** for every **$i,j \in Y$**
- **$d(i,j) \leq \varepsilon$**, with **$\varepsilon<r$** for every **$i,j \in Z$**
- **$d(i,j) > r$** for every **$i \in Y$** and **$j \in Z$**

- Split part **Y** into subparts **Y_1** and **Y_2**
- Shrink distances in **Y_1** appropriately
- What is the result of this shrinking?

Impossibility theorem

- For **$n \geq 2$**, there is no clustering function that satisfies all three axioms of scale-invariance, richness and consistency

Impossibility theorem (proof sketch)

- A partition Γ’ is a refinement of partition Γ, if each cluster C’ϵ Γ’ is included in some set Cϵ Γ

- There is a partial order between partitions: Γ’≤ Γ

- Antichain of partitions: a collection of partitions such that no one is a refinement of others

- Theorem: If a clustering function f satisfies scale-invariance and consistency, then, the range of f is an anti-chain

What does an impossibility result really mean

- Suggests a technical underpinning for the difficulty in unifying the initial, informal concept of clustering
- Highlights basic trade-offs that are inherent to the clustering problem
- Distinguishes how clustering methods resolve these tradeoffs (by looking at the methods not only at an operational level)

Outline

- Impossibility theorem for clustering
- Density-based clustering and subspace clustering
- Bi-clustering or co-clustering

Density-Based Clustering Methods

- Clustering based on density (local cluster criterion), such as density-connected points
- Major features:
  - Discover clusters of arbitrary shape
  - Handle noise
  - One scan
  - Need density parameters as termination condition
- Several interesting studies:
  - <u>DBSCAN:</u> Ester, et al. (KDD'96)
  - <u>OPTICS</u>: Ankerst, et al (SIGMOD'99).
  - <u>DENCLUE</u>: Hinneburg & D. Keim (KDD'98)
  - <u>CLIQUE</u>: Agrawal, et al. (SIGMOD'98)

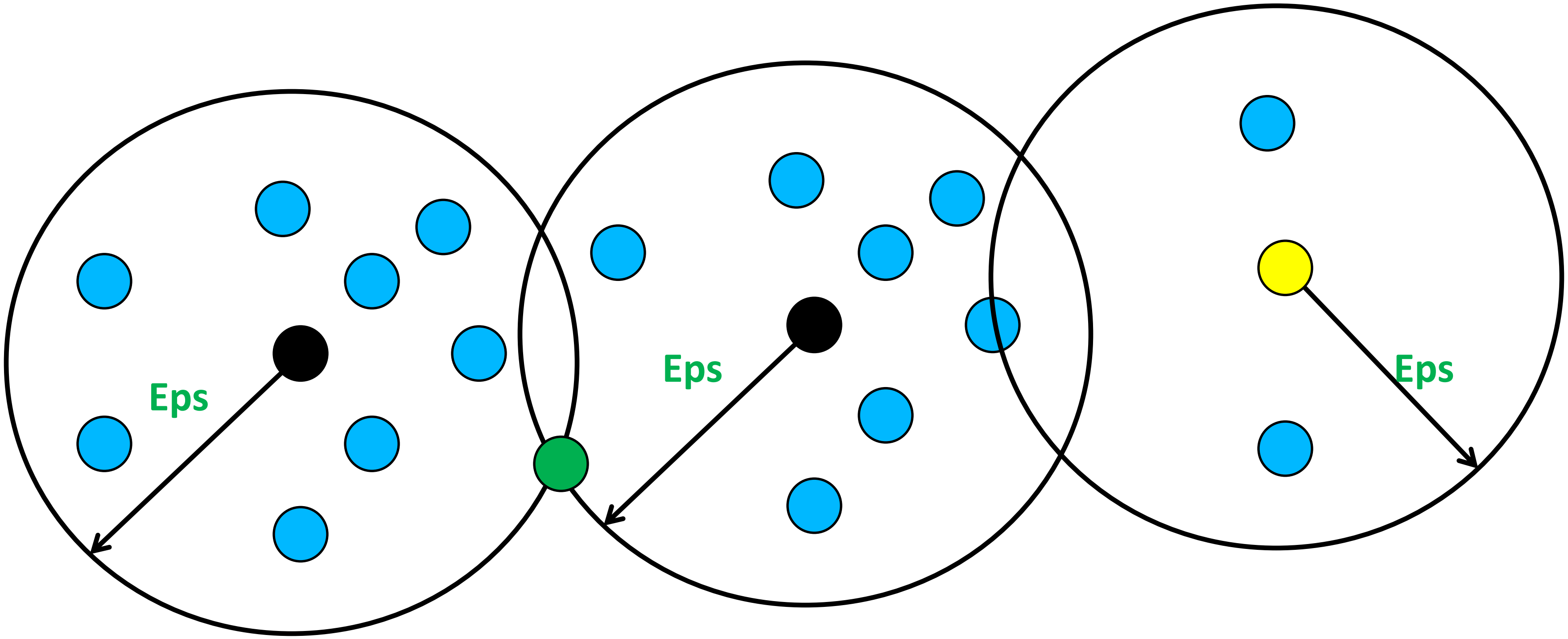
Classification of points in density-based clustering

- **Core points**: Interior points of a density-based cluster. A point **p** is a core point if for distance **Eps** :
  - $|N_{Eps}(p)=\{q \mid dist(p,q) <= \varepsilon\}| \geq MinPts$

- **Border points:** Not a core point but within the neighborhood of a core point (it can be in the neighborhoods of many core points)

- **Noise points:** Not a core or a border point

Core, border and noise points



DBSCAN: The Algorithm

- Label all points as ***core***, ***border***, or ***noise*** points
- Eliminate noise points
- Put an edge between all core points that are within ***Eps*** of each other
- Make each group of connected core points into a separate cluster
- Assign each border point to one of the cluster of its associated core points

Time and space complexity of DBSCAN

- For a dataset **X** consisting of **n** points, the time complexity of DBSCAN is **O(n x time to find points in the Eps-neighborhood)**

- Worst case **$O(n^2)$**

- In low-dimensional spaces **O(nlogn)**; efficient data structures (e.g., ***kd-trees***) allow for efficient retrieval of all points within a given distance of a specified point

Strengths and weaknesses of DBSCAN

- Resistant to noise
- Finds clusters of arbitrary shapes and sizes
- Difficulty in identifying clusters with varying densities
- Problems in high-dimensional spaces; notion of density unclear
- Can be computationally expensive when the computation of nearest neighbors is expensive

Generic density-based clustering on a grid

- Define a set of grid cells
- Assign objects to appropriate cells and compute the density of each cell
- Eliminate cells that have density below a given threshold **τ**
- Form clusters from “***contiguous***” (adjacent) groups of dense cells

Questions

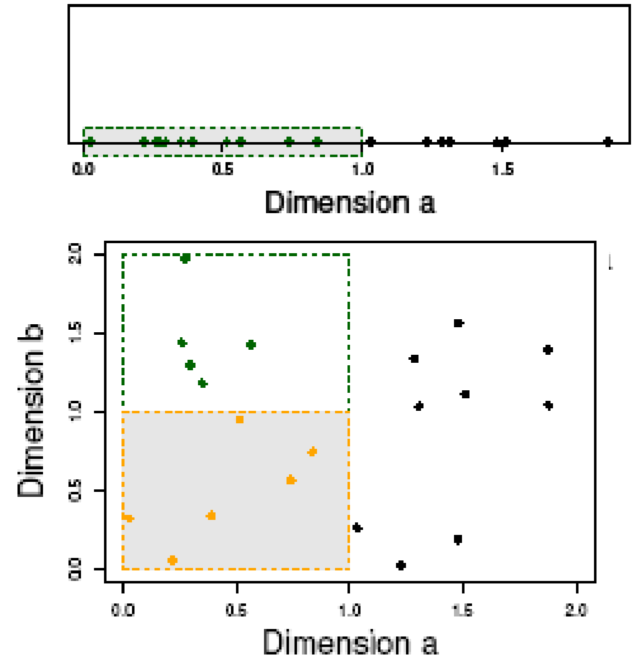
- How do we define the grid?
- How do we measure the density of a grid cell?
- How do we deal with multidimensional data?

Clustering High-Dimensional Data

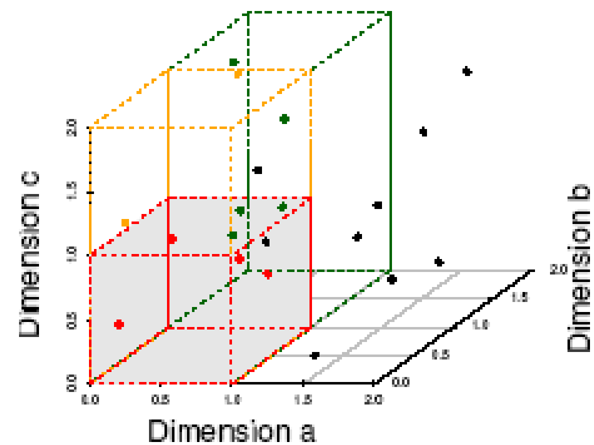
- Clustering high-dimensional data
  - Many applications: text documents, DNA micro-array data
  - Major challenges:
    - Many irrelevant dimensions may mask clusters
    - Distance measure becomes meaningless—due to equi-distance
    - Clusters may exist only in some subspaces
- Methods
  - Feature transformation: only effective if most dimensions are relevant
    - PCA & SVD useful only when features are highly correlated/redundant
  - Feature selection: wrapper or filter approaches
    - useful to find a subspace where the data have nice clusters
  - Subspace-clustering: find clusters in all the possible subspaces
    - CLIQUE

The Curse of Dimensionality

- Data in only one dimension is relatively packed
- Adding a dimension “stretches” the points across that dimension, making them further apart
- Adding more dimensions will make the points further apart—high dimensional data is extremely sparse
- Distance measure becomes meaningless



(b) 6 Objects in One Unit Bin

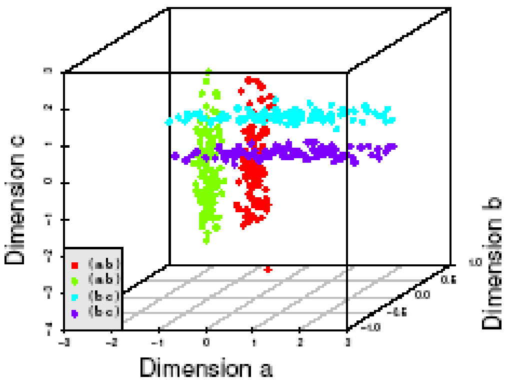


(c) 4 Objects in One Unit Bin

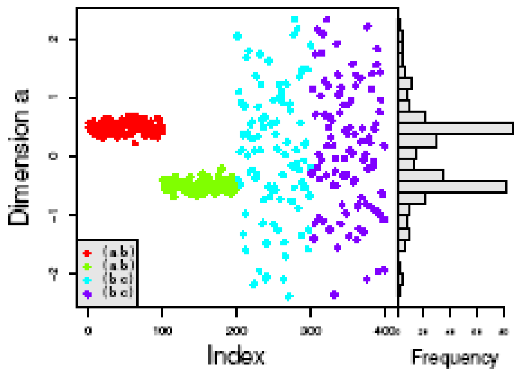
(graphs from Parsons et al. KDD Explorations 2004)

Why Subspace Clustering?

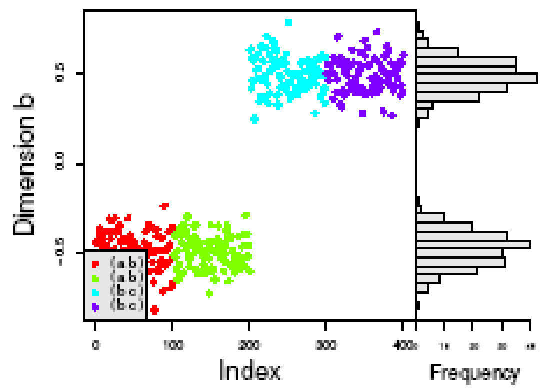
(Parsons et al. SIGKDD Explorations 2004)



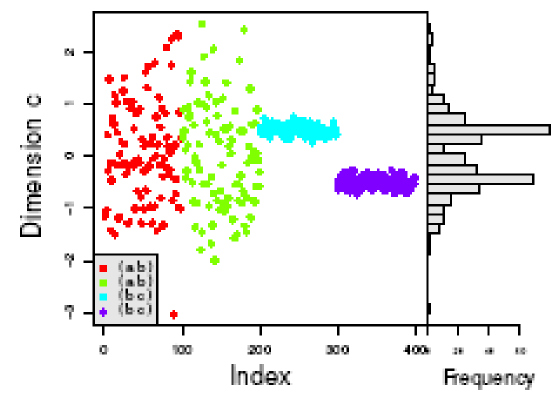
- Clusters may exist only in some subspaces
- ***Subspace-clustering***: find clusters in some of the subspaces



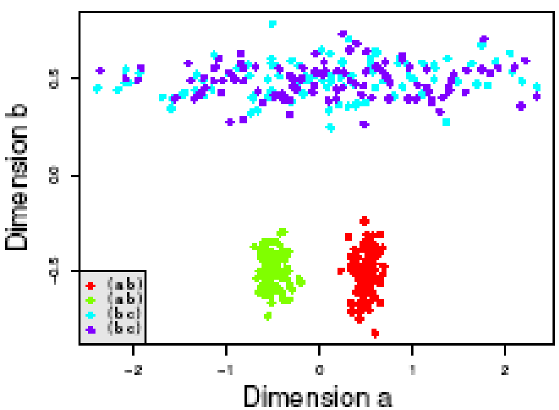
(a) Dimension a



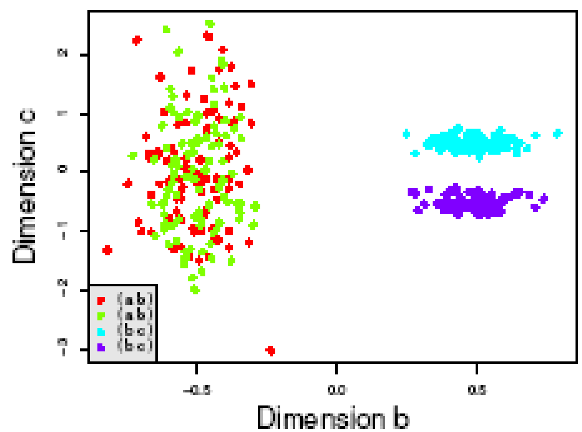
(b) Dimension b



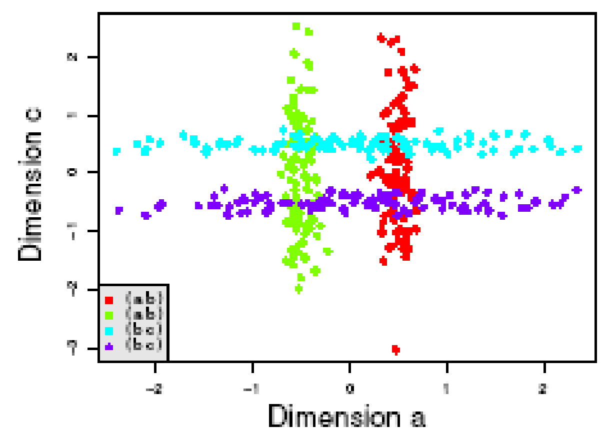
(c) Dimension c



(a) Dims a & b



(b) Dims b & c



(c) Dims a & c

CLIQUE (Clustering In QUEst)

- Agrawal, Gehrke, Gunopulos, Raghavan (SIGMOD'98)
- Automatically identifying subspaces of a high dimensional data space that allow better clustering than original space
- **CLIQUE** can be considered as both density-based and grid-based
  - It partitions each dimension into the same number of equal length interval
  - It partitions an **m**-dimensional data space into non-overlapping rectangular units
  - A unit is ***dense*** if the fraction of total data points contained in the unit exceeds an input threshold **τ**
  - A ***cluster*** is a ***maximal*** set of ***connected dense units*** within a subspace

The CLIQUE algorithm

- Find all dense areas in the 1-dimensional spaces (single attributes)
- **k ← 2**
- **repeat**
  - Generate all candidate dense **k-**dimensional cells from dense **(k-1)-**dimensional cells
  - Eliminate cells that have fewer than **τ** points
  - **k← k+1**
- **until** there are no candidate dense **k**-dimensional cells
- Find clusters by taking the union of all adjacent, high-density cells
- Summarize each cluster using a small set of inequalities that describe the attribute ranges of the cells in the cluster

CLIQUE: Monotonicity property

- "If a set of points forms a density-based cluster in **k**-dimensions (attributes), then the same set of points is also part of a density-based cluster in all possible subsets of those dimensions"

Strengths and weakness of **CLIQUE**

- *automatically* finds subspaces of the highest dimensionality such that high density clusters exist in those subspaces

- *insensitive* to the order of records in input and does not presume some canonical data distribution

- scales *linearly* with the size of input and has good scalability as the number of dimensions in the data increases

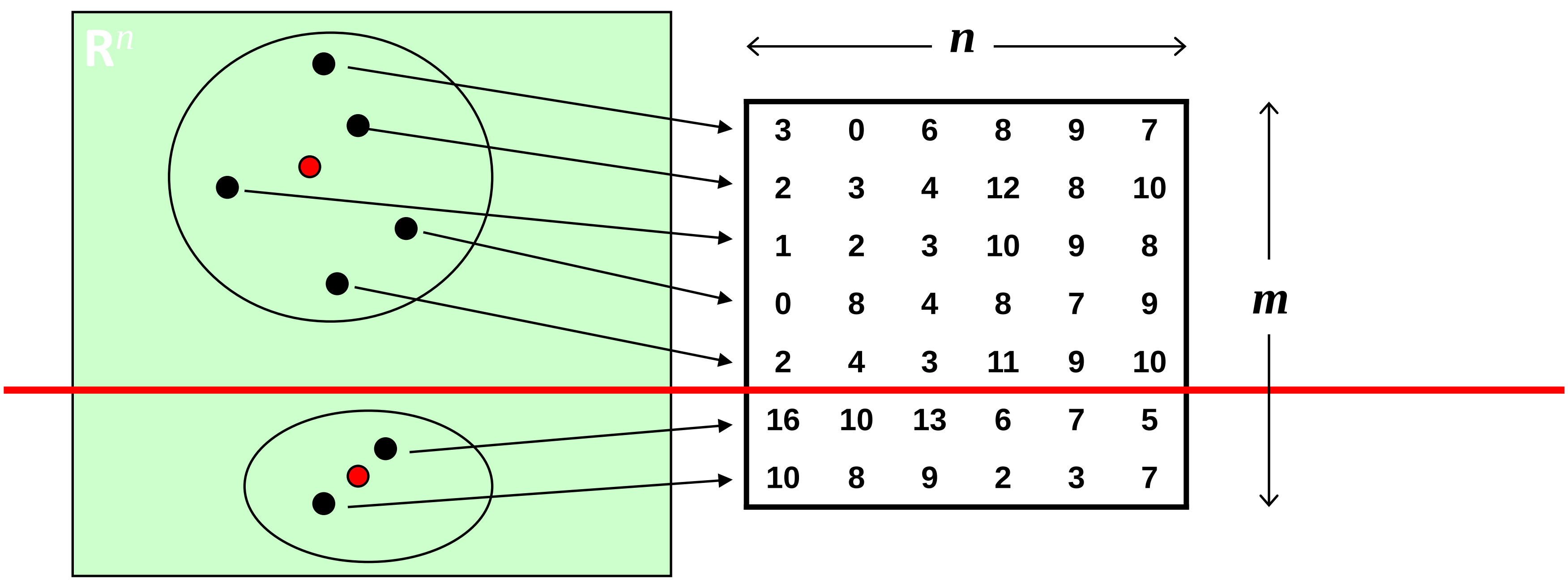
- Its not clear how to define the boundaries of cells in the different dimensions

Outline

- Impossibility theorem for clustering
- Density-based clustering and subspace clustering
- Bi-clustering or co-clustering

Clustering

- m points in $\mathbf{R}^n$
- Group them to k clusters
- Represent them by a matrix $A \in \mathbf{R}^{m \times n}$
  - A point corresponds to a row of A
- **Cluster:** Partition the rows to k



Co-Clustering

- **Co-Clustering: Cluster rows and columns of A simultaneously:**

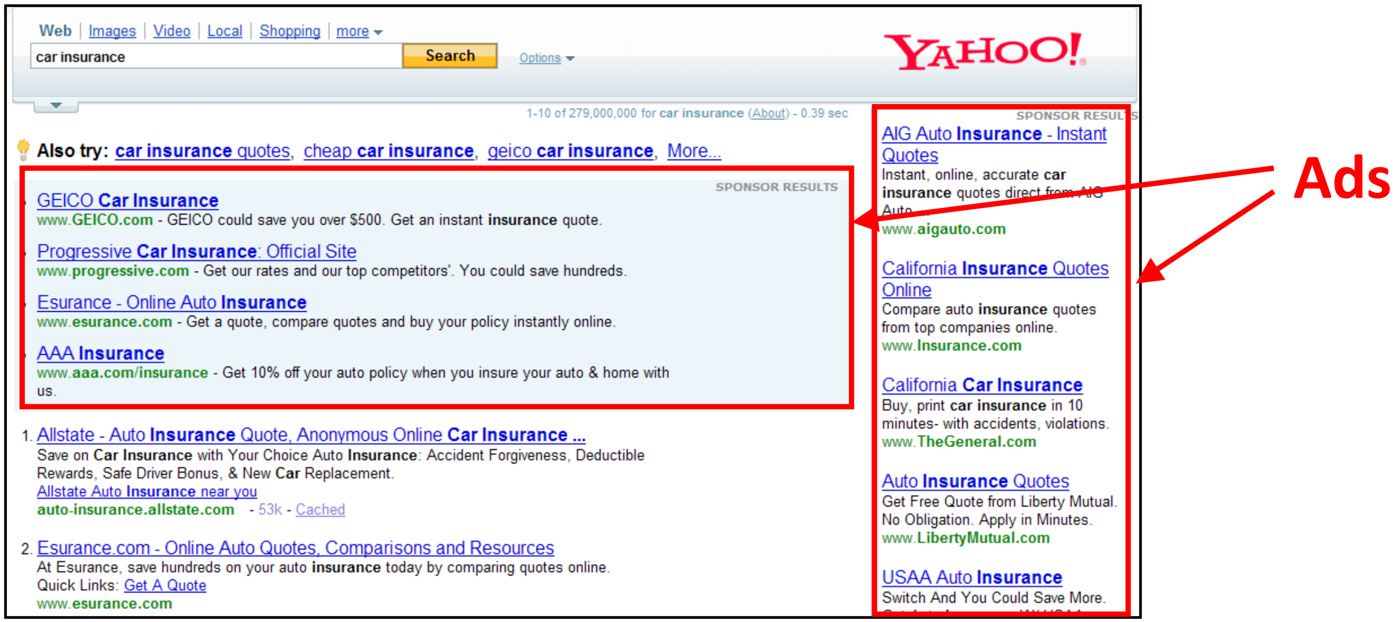
$$\ell = 2$$

$$k = 2 \qquad \left[\begin{array}{ccc|ccc}
3 & 0 & 6 & 8 & 9 & 7 \\
2 & 3 & 4 & 12 & 8 & 10 \\
1 & 2 & 3 & 10 & 9 & 8 \\
0 & 8 & 4 & 8 & 9 & 7 \\
2 & 4 & 3 & 11 & 9 & 10 \\
\hline
16 & 10 & 13 & 6 & 7 & 5 \\
10 & 8 & 9 & 2 & 3 & 7
\end{array}\right]$$

$$A$$

Co-cluster

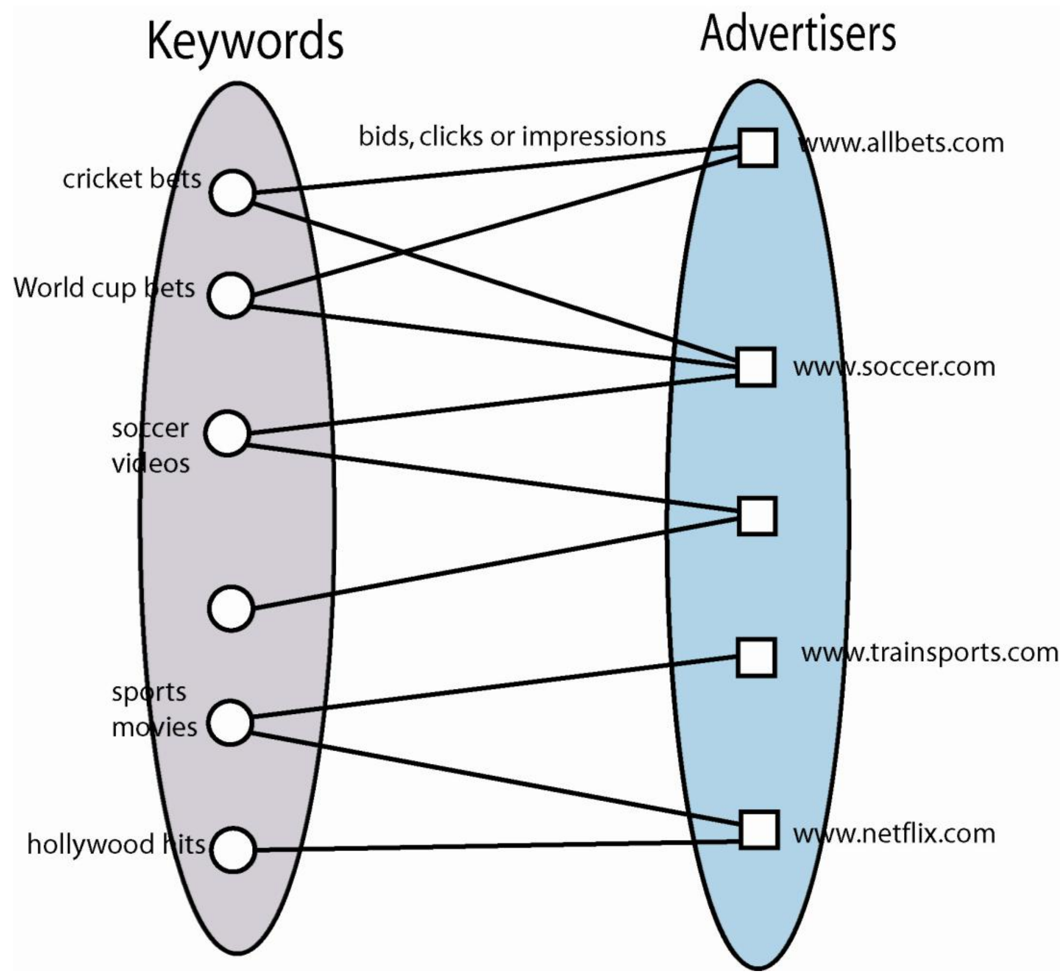
Motivation: Sponsored Search



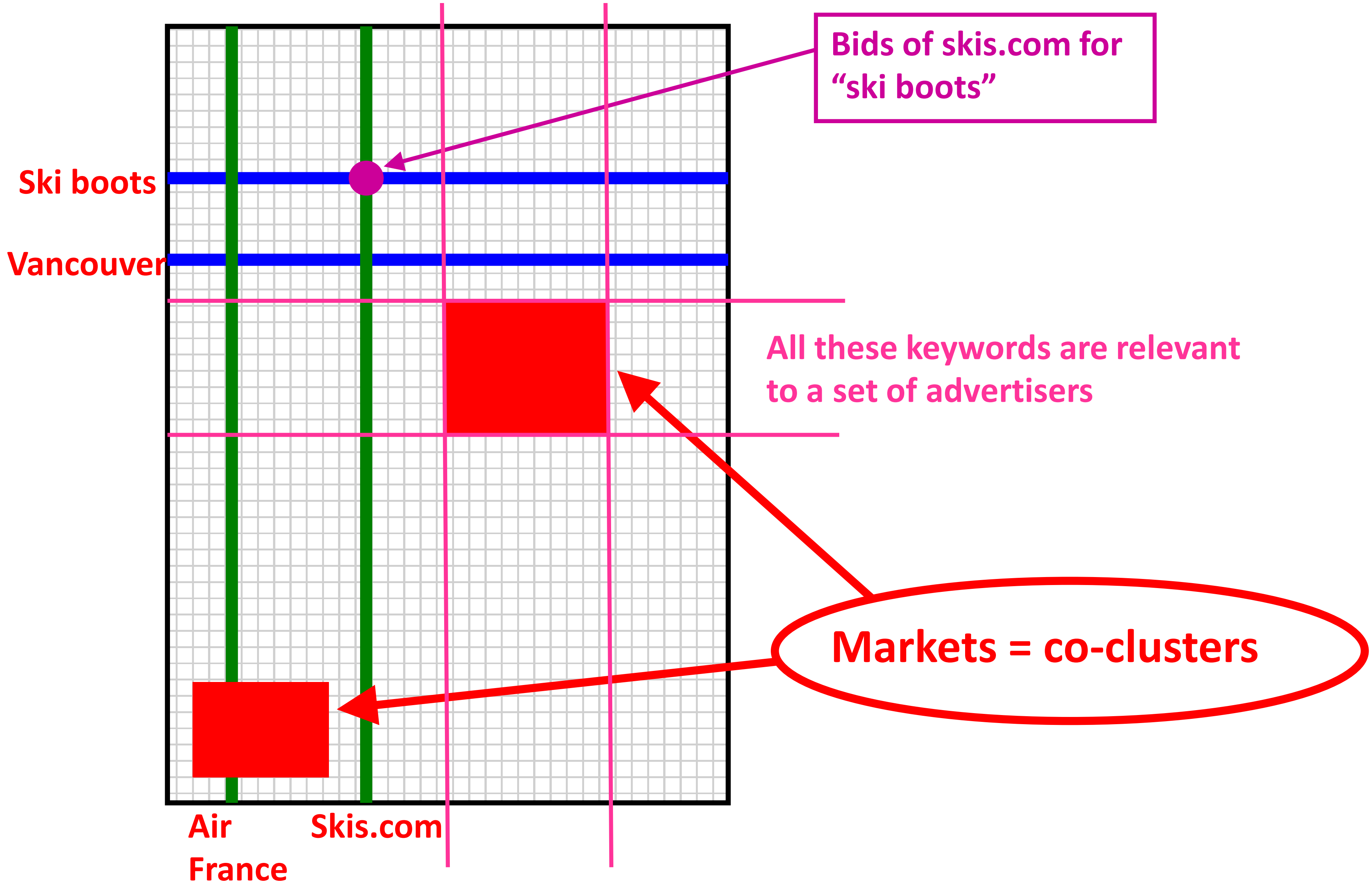
- Advertisers bid on keywords
- A user makes a query
- Show ads of advertisers that are relevant and have high bids
- User clicks or not an ad

Motivation: Sponsored Search

- For every (*advertiser, keyword*) pair we have:
  - Bid amount
  - Impressions
  - # clicks
- Mine information at query time
  - Maximize # clicks / revenue



Co-Clusters in Sponsored Search

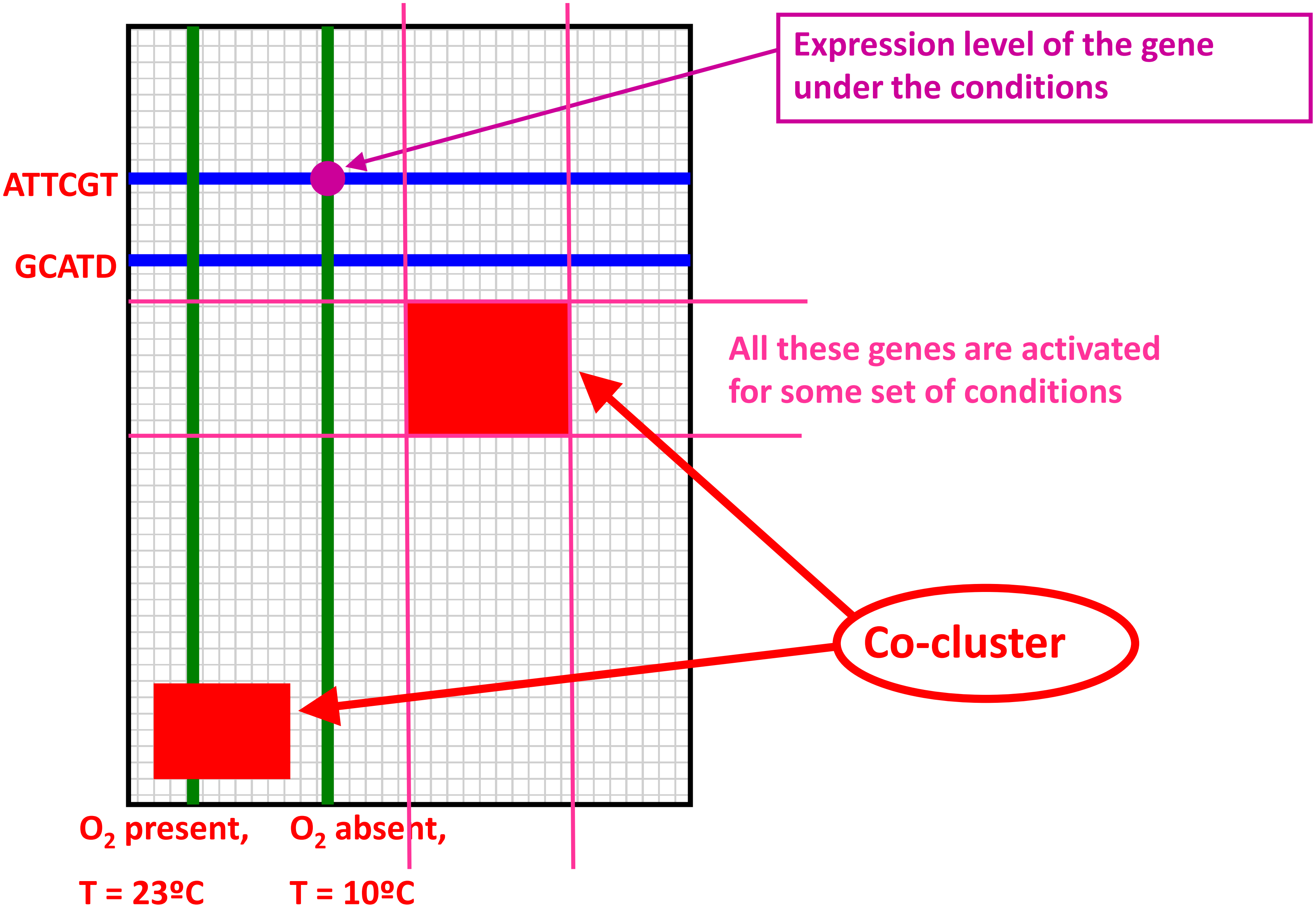


Co-Clustering in Sponsored Search

Applications:

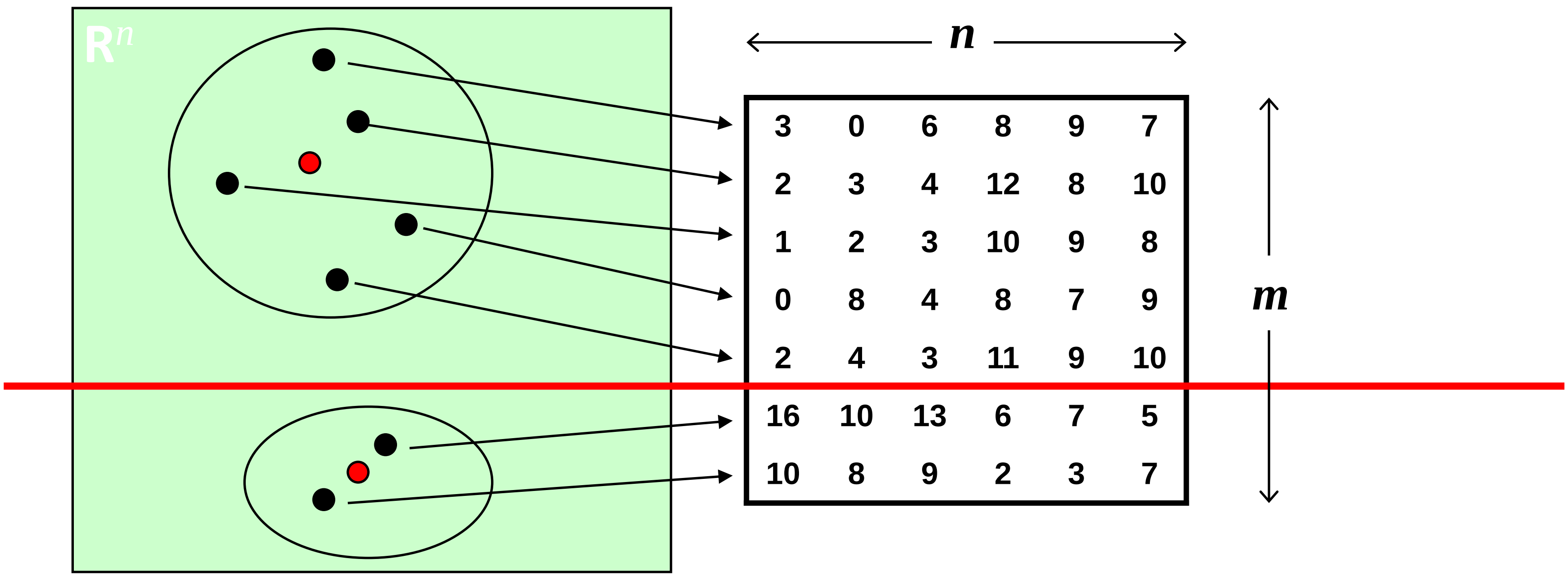
- Keyword suggestion
  - Recommend to advertisers other relevant keywords
- Broad matching / market expansion
  - Include more advertisers to a query
- Isolate submarkets
  - Important for economists
  - Apply different advertising approaches
- Build taxonomies of advertisers / keywords

Co-Clusters in Gene Expression Data

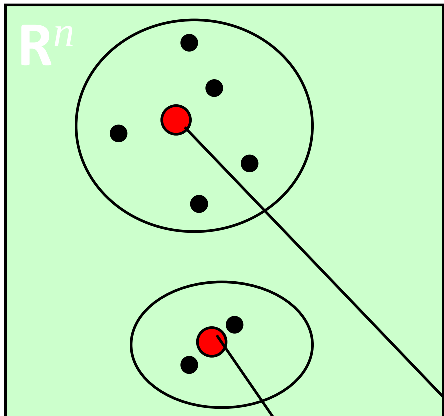


Clustering of the rows

- m points in $\mathbf{R}^n$
- Group them to k clusters
- Represent them by a matrix $A \in \mathbf{R}^{m \times n}$
  - A point corresponds to a row of A
- **Clustering:** Partitioning of the rows into k groups



Clustering of the columns



- n points in $\mathbf{R}^m$
- Group them to k clusters
- Represent them by a matrix $A \in \mathbf{R}^{m \times n}$
  - A point corresponds to a column of A
- **Clustering:** Partitioning of the columns into k groups

3	0	6	8	9	7
2	3	4	12	8	10
1	2	3	10	9	8
0	8	4	8	7	9
2	4	3	11	9	10
16	10	13	6	7	5
10	8	9	2	3	7

m

n

3	3	3	9	9	9
3	3	3	9	9	9
3	3	3	9	9	9
3	3	3	9	9	9
3	3	3	9	9	9
11	11	11	5	5	5
11	11	11	5	5	5

Cost of clustering

3	0	6	8	9	7
2	3	4	12	8	10
1	2	3	10	9	8
0	8	4	8	7	9
2	4	3	11	9	10
16	10	13	6	7	5
10	8	9	2	3	7

Original data points **A**

1.6	3.4	4	9.8	8.4	8.8
1.6	3.4	4	9.8	8.4	8.8
1.6	3.4	4	9.8	8.4	8.8
1.6	3.4	4	9.8	8.4	8.8
1.6	3.4	4	9.8	8.4	8.8
13	9	11	4	5	6
13	9	11	4	5	6

Data representation **A'**

- In **A'** every point in **A** (row or column) is replaced by the corresponding representative (row or column)
- The quality of the clustering is measured by computing distances between the data in the cells of **A** and **A'**.

- **k-means clustering**: **cost =** $\sum_{i=1\ldots n} \sum_{j=1\ldots m} (A(i,j)-A'(i,j))^2$
- **k-median clustering**: **cost =** $\sum_{i=1\ldots n} \sum_{j=1\ldots m} |A(i,j)-A'(i,j)|$

Co-Clustering

- **Co-Clustering:** Cluster rows and columns of $A \in \mathbf{R}^{m \times n}$ simultaneously
- $\boldsymbol{k}$ row clusters, $\boldsymbol{\ell}$ column clusters
- Every cell in **A** is represented by a cell in **A'**
- All cells in the same co-cluster are represented by the same value in the cells of **A'**

3	0	6	8	9	7
2	3	4	12	8	10
1	2	3	10	9	8
0	8	4	8	9	7
2	4	3	11	9	10
16	10	13	6	7	5
10	8	9	2	3	7

Original data **A**

3	3	3	9	9	9
3	3	3	9	9	9
3	3	3	9	9	9
3	3	3	9	9	9
3	3	3	9	9	9
11	11	11	5	5	5
11	11	11	5	5	5

Co-cluster representation **A'**

Co-Clustering Objective Function

3	0	6	8	9	7
2	3	4	12	8	10
1	2	3	10	9	8
0	8	4	8	7	9
2	4	3	11	9	10
16	10	13	6	7	5
10	8	9	2	3	7

3	3	3	9	9	9
3	3	3	9	9	9
3	3	3	9	9	9
3	3	3	9	9	9
3	3	3	9	9	9
11	11	11	5	5	5
11	11	11	5	5	5

- In **A'** every point in **A** (row or column) is replaced by the corresponding representative (row or column)
- The quality of the clustering is measured by computing distances between the data in the cells of **A** and **A'**.

- **k-means Co-clustering**: cost = $\sum_{i=1\ldots n} \sum_{j=1\ldots m} (A(i,j)-A'(i,j))^2$
- **k-median Co-clustering**: cost = $\sum_{i=1\ldots n} \sum_{j=1\ldots m} |A(i,j)-A'(i,j)|$

Some Background

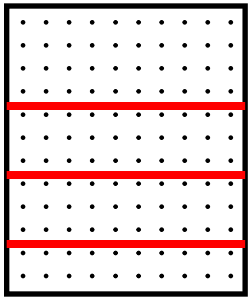
- A.k.a.: biclustering, block clustering, …

- Many objective functions in co-clustering
  - This is one of the easier
  - Others factor out row-column average (priors)
  - Others based on information theoretic ideas (e.g. KL divergence)

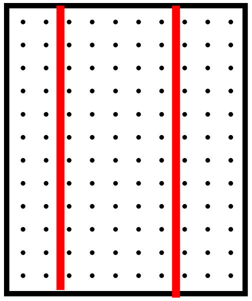
- A lot of existing work, but mostly heuristic
  - k-means style, alternate between rows/columns
  - Spectral techniques

Algorithm

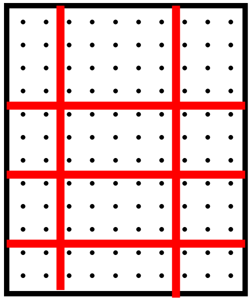
1. Cluster rows of $\boldsymbol{A}$



2. Cluster columns of $\boldsymbol{A}$



3. Combine



Properties of the algorithm

Theorem 1. Algorithm with optimal row/column clusterings is 3-approximation to co-clustering optimum.

Theorem 2. For L_2 distance function, the algorithm with optimal row/column clusterings is a 2-approximation.

Algorithm--details

- Clustering of the **n** rows of **A** assigns every row to a cluster with cluster name **{1,…,k}**
  - $R(i)= r_i$ with $1\leq r_i \leq k$
- Clustering of the **m** columns of **A** assigns every column to a cluster with cluster name **{1,…,ℓ}**
  - $C(j)=c_j$ with $1\leq c_j \leq \ell$
- $A'(i,j) = \{r_i,c_j\}$
- **(i,j)** is in the same co-cluster as **(i',j')** if $A'(i,j)=A'(i',j')$