Distance-based Outliers and Robust Space Transformations

Raymond Ng
Dept. of Computer Science, UBC

(Joint work with Ed Knorr and Ruben Zamar)

Focus of Our Work

- To efficiently identify meaningful outliers in large, multidimensional datasets

- 3 main parts:

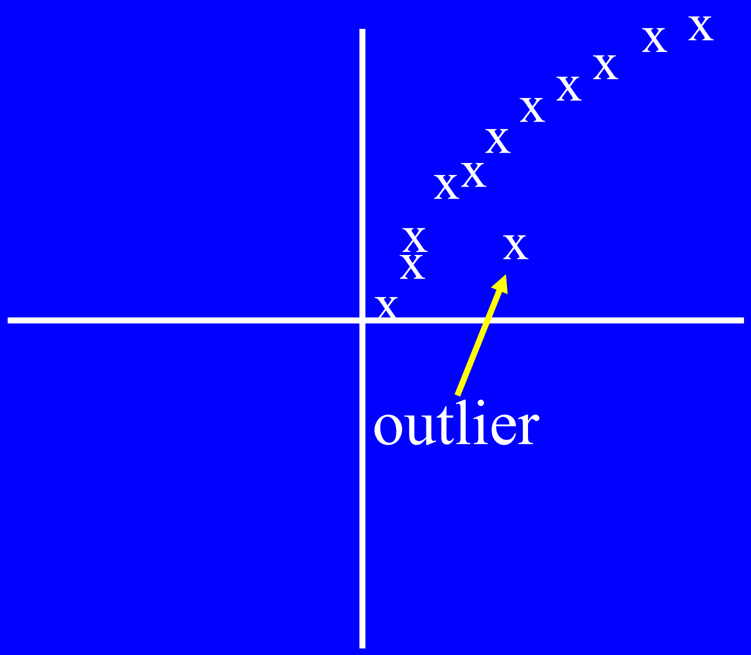
1. Outlier *identification*
2. Outlier *explanation*
3. ** Outlier *generalization* (i.e., statistical distances vs. Euclidean distances)

Motivation of Our Work

- numerous techniques treat outliers as second-class citizens, i.e., how to get the job done in spite of the outliers
- in our work, outliers are first-class citizens as valuable discovered knowledge
- "*one person's noise is another person's signal*"
- valuable for surveillance applications and other monitoring tasks

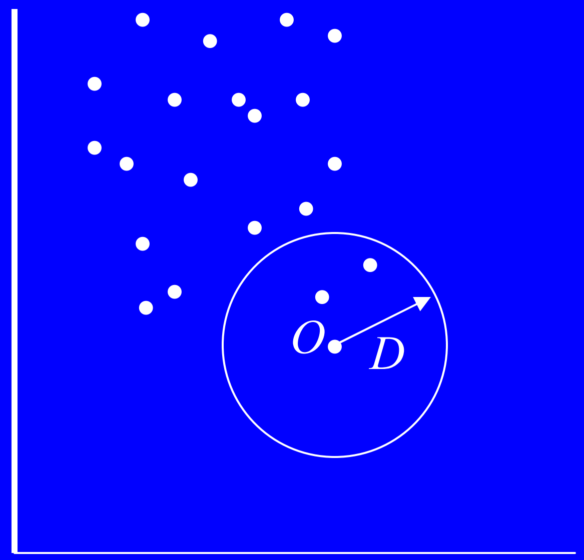
Intuitive Notion of Outliers

- An *outlier* is an object which differs *sufficiently* from a great majority of the other objects
- "One of these things is not like the others …" [Sesame Street, circa 1975]



DB-Outliers (Distance-Based Outliers)

- Formally:
  - Object *O* in dataset *T* is a DB(*p*,*D*)-outlier if at least fraction *p* of the objects in *T* are > distance *D* from *O*
  - e.g., DB(0.99,5) => 99% of points are > 5 units distance away



Existing Outlier Detection Techniques

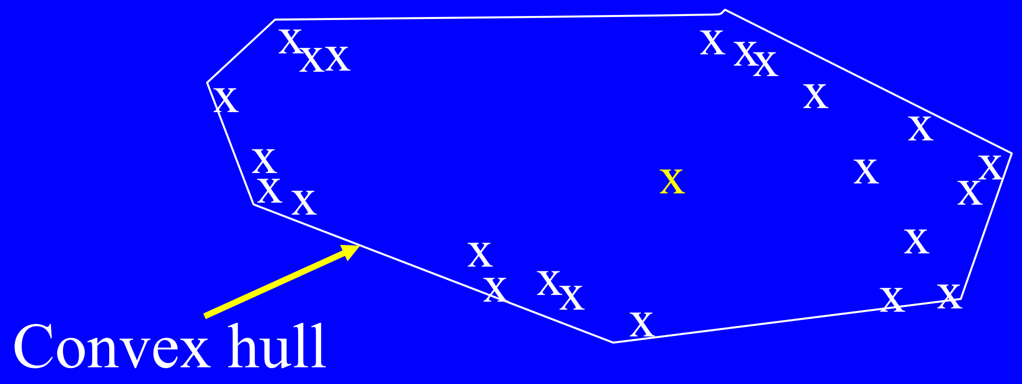
- Visual-Based (low dimensional only)
  - Boxplot (1-D), Scatterplot (2-D), Spin Plot (3-D)
  - Time-consuming, subjective
- Distribution-Based
  - Statistical discordancy tests (e.g., [BL94])
    - Requires Prior Knowledge of Distribution, # of Outliers, Types of Outliers, Mostly Univariate
    - Subject to Masking and Swamping

Existing Techniques: Depth-Based Methods

- Peeling, Depth Contours [PS88],[RR96], [JKN98]
- Regression Depth [vK99]
- Idea: *Shallow layers are more likely to contain outliers*
  - Note: median is found at deepest layer
- High complexity; only suitable for small k, the dimensionality of the space

Extreme Points as Outliers?

- What if outliers occur in middle of data rather than at extremes?
  - "extreme" points (lots!) appear on convex hull



Part 1: Overview of DB-outlier Identification [KN98]

Salient Features of DB-Outliers

- non-parametric
- need not be extreme points
- algorithmically, quadratic wrt k, the dimensionality
  - particularly suitable for large values of k
  - can handle many non-standard applications, e.g., video survelliance -> 2-D spatio trajectories

More About Algorithms

- an optimized cell-based algorithm
  - linear wrt the number of objects
  - suitable only for small values of k
- handle the complication when the entire data set cannot fit in main memory
  - guarantee at most 3 passes over the data

Part 2: Overview of Outlier Explanation [KN99]

Forms of Explanation

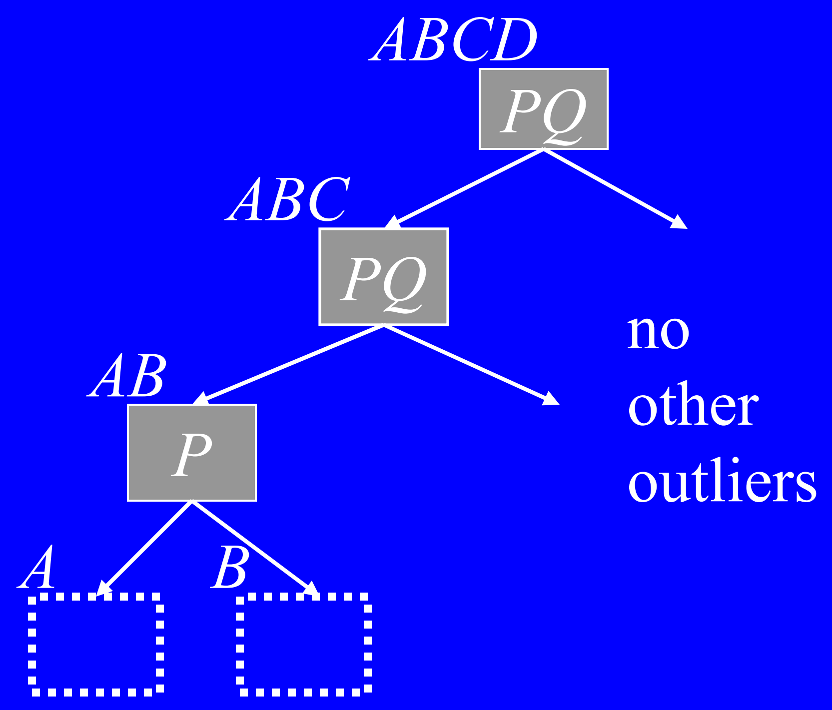
- We provide **intensional knowledge** of <u>specific</u> forms, namely, *structural* intensional knowledge:
  - *Which sets of dimensions explain the uniqueness of the outliers?*
  - *How can one outlier <u>be compared with</u> another?*
- We introduce the notions of **strongest** and **weak** outliers, and how to compute them efficiently

Strongest Outliers

Suppose P is an outlier in space A_P. Then ...

1. P is a **strongest** outlier in a space A_P if *no* outlier exists in *any* subspace of A_P
2. P is a **trivial** outlier in superspaces of A_P

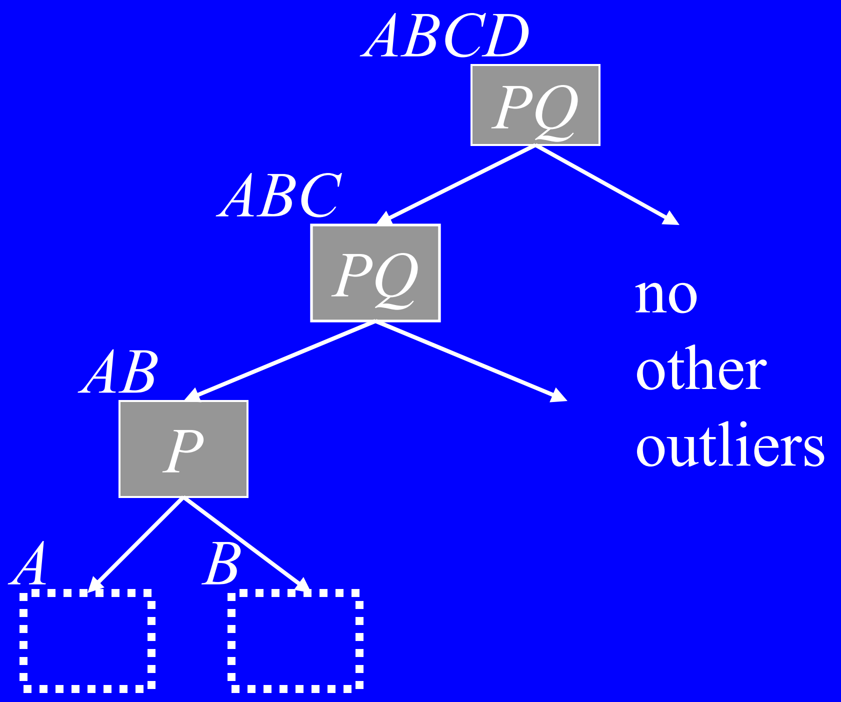
Example



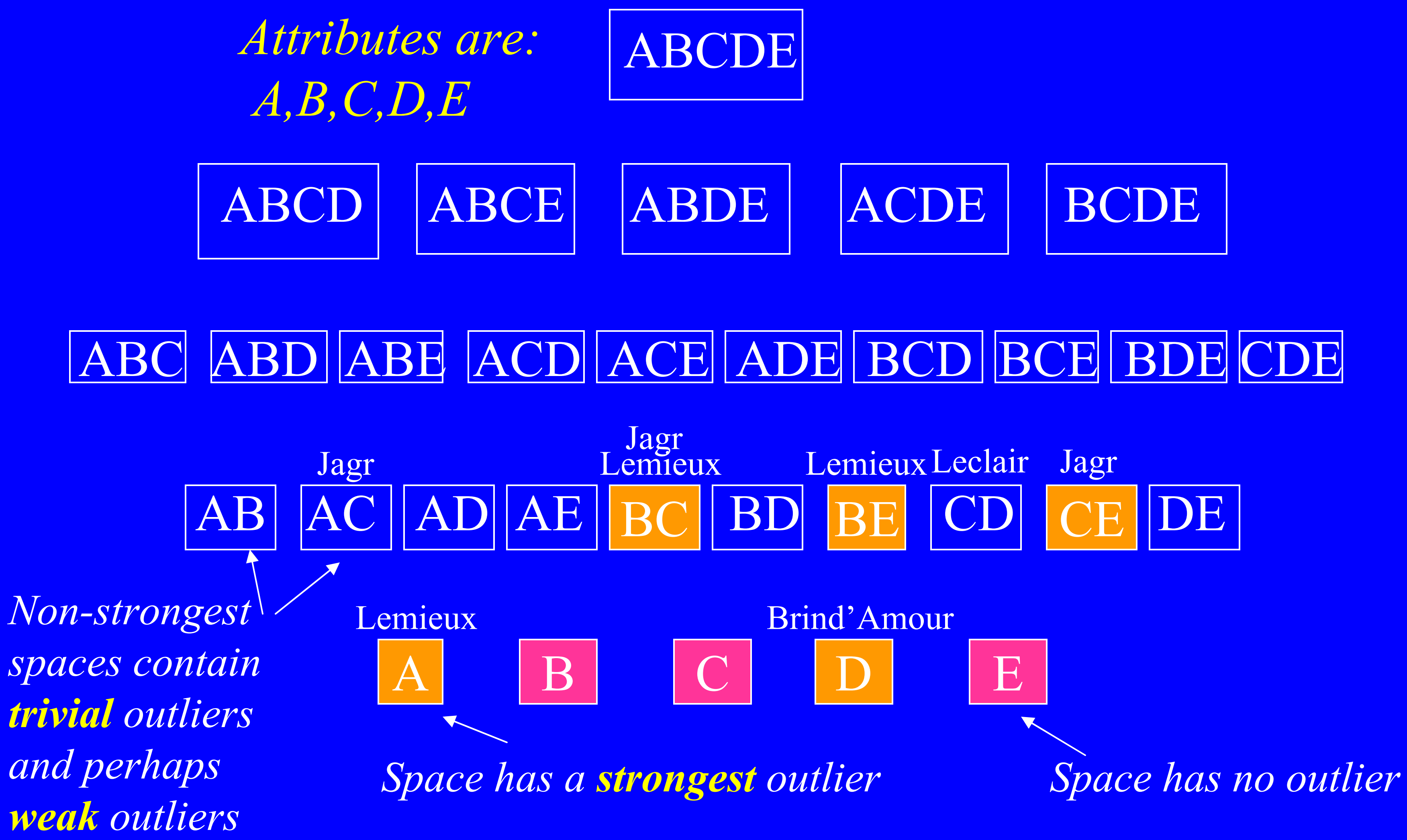
Weak Outliers

3. Q is a **weak** outlier in A_P if Q is neither strongest nor trivial

Example



Intensional Knowledge in a Lattice



Part 3: Robust Space Transformations [KNZ01]

General Comments

- Distance-based operations assume (weighted) Euclidean *k*-D space … not always correct!
- Data mining applications (e.g., clustering, nearest-neighbour search, outlier detection) often neglect to deal with differing scale, variability, correlation, and outliers in datasets
  - need to "fairly" compare attributes to get meaningful results

"So, what is an appropriate space?"

Motivating Example

- Consider a dataset of 3-tuples, each containing measurements for these attributes for adolescents aged 13-19:
  1. Systolic blood pressure (in mm Hg., μ=120)
  2. Body temperature (in degrees Celsius, μ=37)
  3. Age (μ=16)

- Are distance comparisons meaningful?

Simple "Fixes"

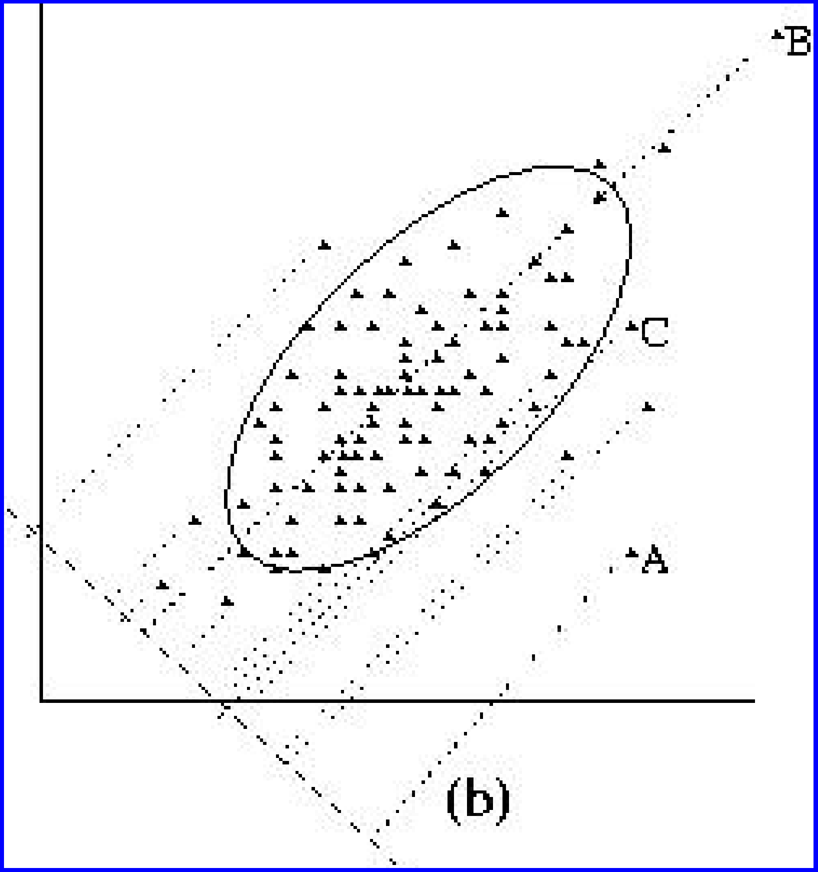
- Normalize the ranges (e.g., map each attribute into the range [0,1])
  - But outliers can seriously skew the range!
- Use Weighted Euclidean
  - But how do we find appropriate weights?
- Standardize the ranges (e.g., map each observation x to $(x-\mu)/\sigma$)
  - better, but outliers can still dominate and skew range
- Our solution: use a <u>robust</u> space transformation, namely Donoho-Stahel Estimator (DSE)

Estimators, DSE Properties

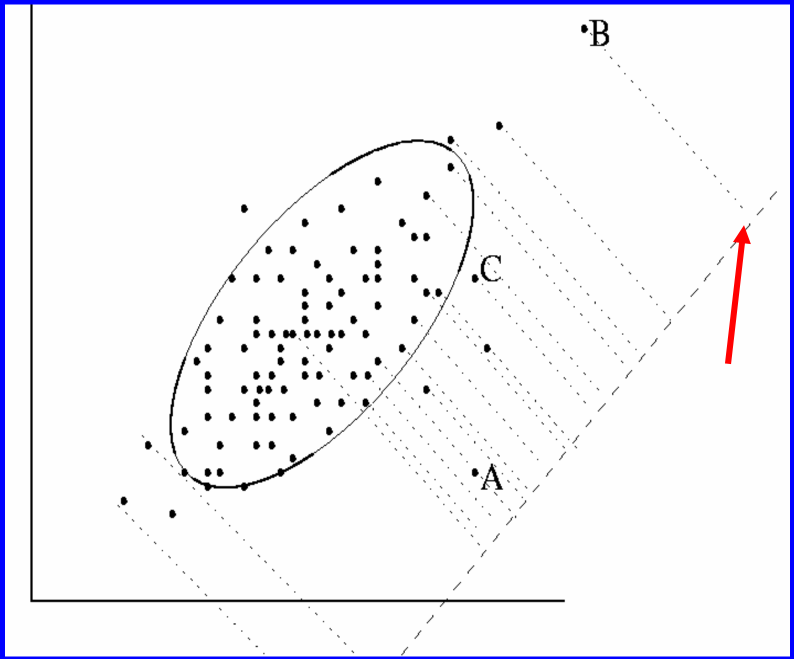
- Other robust estimators:
  - Minimum Volume Ellipsoid (MVE)
  - Minimum Covariance Determinant (MCD)
  - Fast MCD (FMCD)
  - References: [RL87], [RvD99], [MZ01]
- DSE properties:
  - affine equivariance, small bias, intuitively appealing, scales relatively well

DSE: Projection Vectors

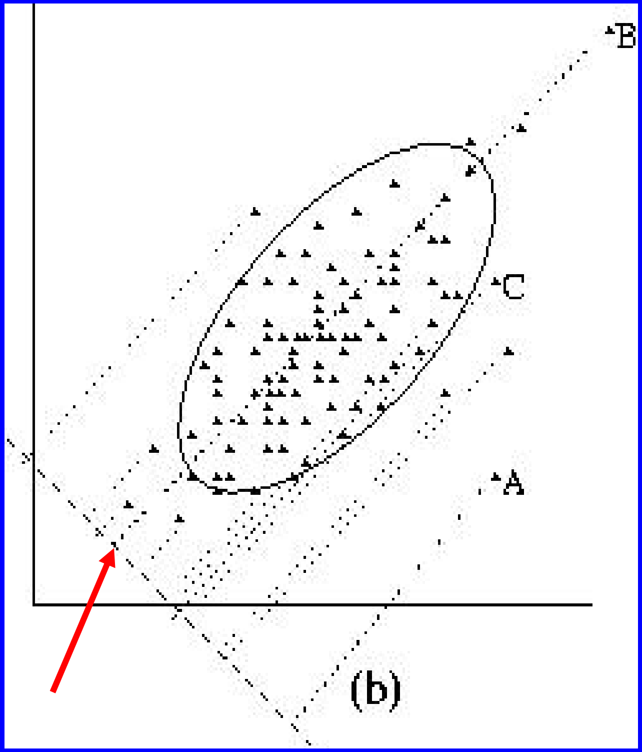
- Points are projected onto *projection vectors*
- Find out which points are outlying on the projection vector



Projecting points onto different projection vectors (dashed lines)



B is outlying, but *not* A, C



B is not outlying here

Skeleton Algorithm for the DSE Scatter Matrix

- For each projection vector selected
  - Project all *N* points onto it
  - Compute each point's "outlyingness" value
  - Keep track of each point's largest outlyingness value (across all projection vectors)
- Compute the robust covariance matrix by downweighting each point according to:
  - its largest outlyingness value
  - a weighting function

Key question (later): What is a good set of projection vectors to use?

(1) Fixed-angle Algorithm

- Proposed independently by Donoho and Stahel in early 1980's
- Idea: Exhaustively try a fixed increment
- Very CPU intensive: $O(a^{k-1}\,k\,N)$
  - a = number of angles tested
  - e.g., 75-85 hours of CPU time in 5-D for N=100,000 tuples, using a 10-degree increment
- Yields a high quality estimator … but the following algorithms achieve a finer balance between efficiency and quality

(2) Subsampling Algorithm

- Proposed by Stahel
- Uses projection vectors orthogonal to axes of hyperellipsoid
- Also CPU intensive: $O(m\,k^3 + k^2 N)$
  - m is number of subsamples desired
  - e.g., for 5-D, with 95% chance of getting at least one "good" subsample, $m = 47$

(3) Pure-random Algorithm

- Randomly pick projection vectors from the unit hypersphere
- $O(r\ k\ N)$ where r = # of random projections
- Can be long-running, but can also give very good results (if lucky)
  - e.g., 5-D, 100K points: 5-10 minutes of CPU time for 90% recall

(4) Hybrid-random Algorithm

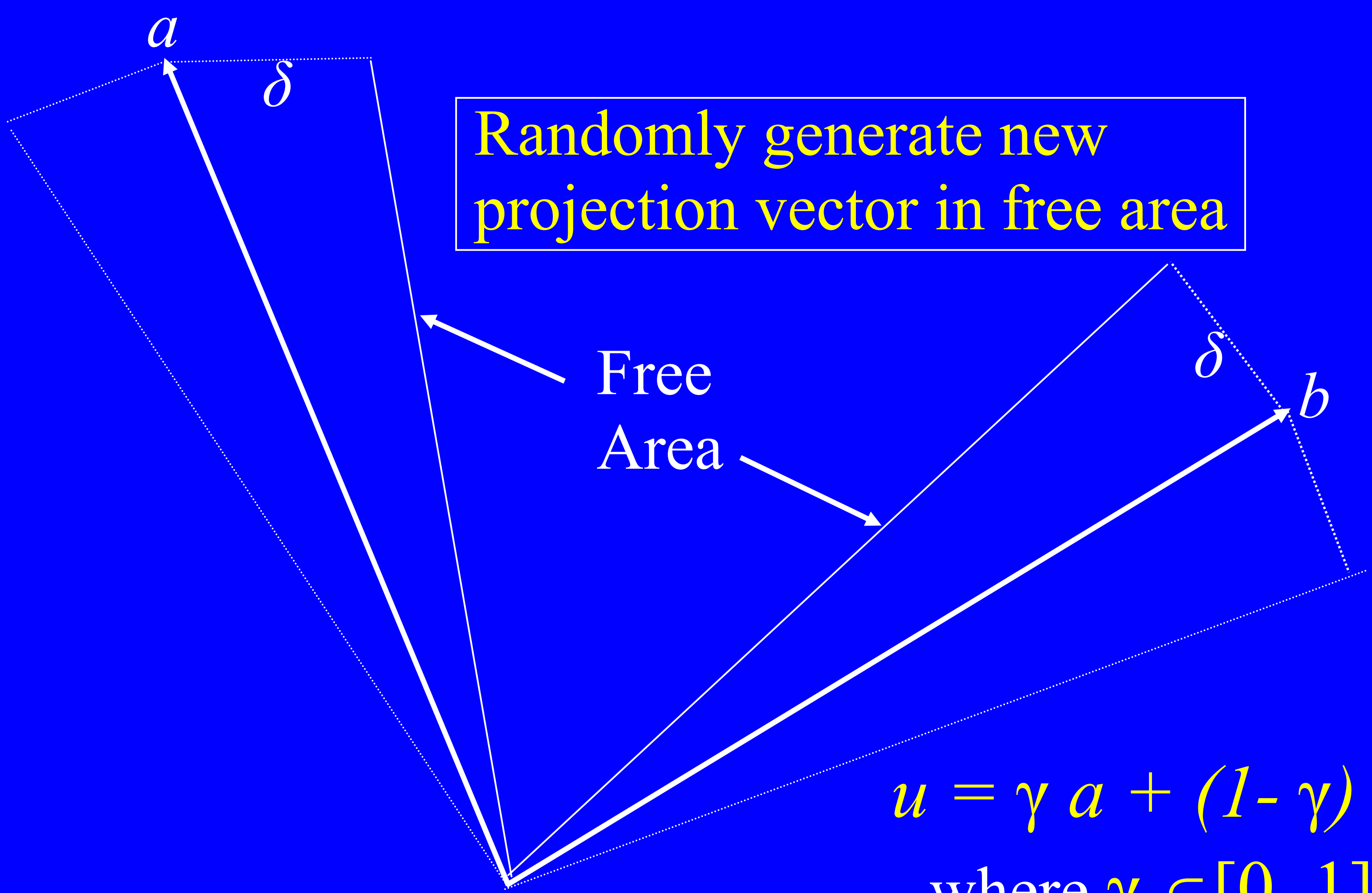
- Our own algorithm
- Combines properties of Subsampling and Pure-random algorithms
- 2 phases make up the *grid* (set of proj. vectors):

1. Use a small number of subsamples (e.g., $m/2$) to start the grid, plus the k eigenvectors
2. Set a buffer zone around each grid vector and randomly generate new vectors outside of all zones

   (Projection vectors too close to each other yield similar results)

2-D Example

Randomly generate new projection vector in free area



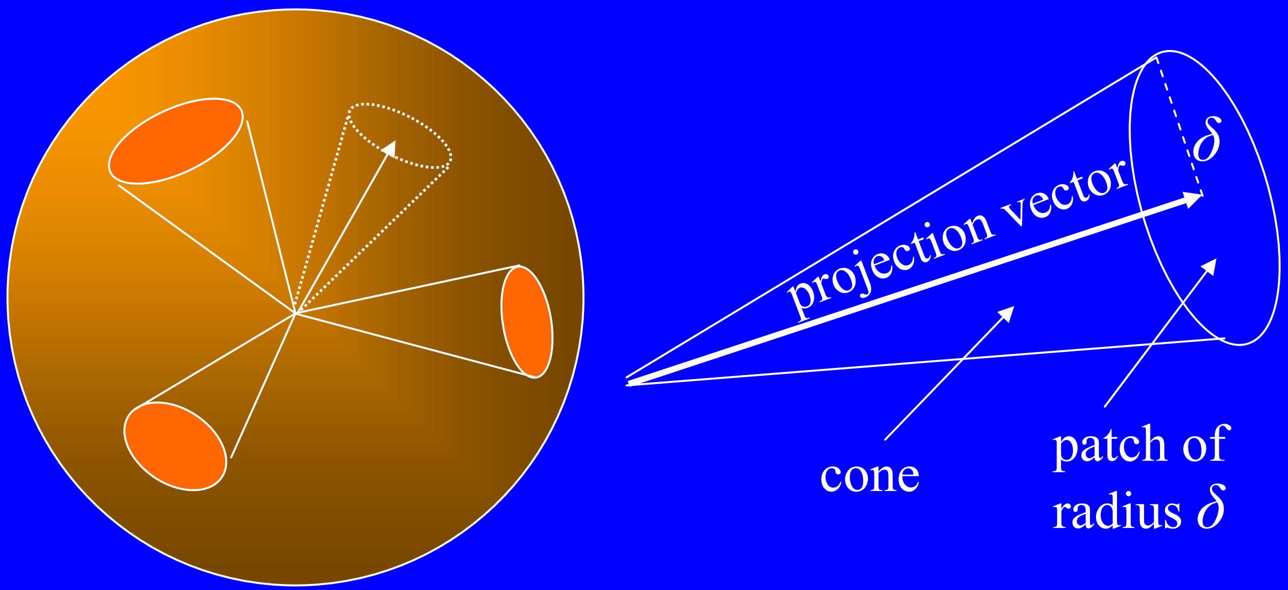
$$u = \gamma\, a + (1 - \gamma)\, b$$

where $\gamma \in [0, 1]$

but avoid cone collisions

3-D Example of Projection Vectors, Cones, and Patches

- Randomly pick 2 existing grid vectors and create a new projection vector randomly between them (avoid colliding with existing cones)



Experimental Results

Experimental Setup

- Outlier detection application
- Datasets range in size from 1K to 200K, and 3-D to 10-D, real-life and synthetic datasets
  - Can use sampling for DSE for large datasets
- We report the median of 3 runs, for the randomized cases

Executive Summary

- Fixed-angle Algorithm (Worst Performer):
  - Can be several orders of magnitude longer than others
  - But, its exhaustive search provides a “guarantee” of quality of estimator
- Subsampling:
  - Typically fast to return an estimator of modest quality
  - May take a long time to return a higher quality estimator

Executive Summary, cont.

- Pure-random:
  - If lucky, can be very competitive with Hybrid-random
  - Otherwise, can be several orders of magnitude longer
- Hybrid-random (Best Performer):
  - Combines best features of:
    - Subsampling (for quickly building the grid, thus providing a good starting point)
    - Pure-Random (for greater speed in improving the quality)

CPU Time for Similar, High Precision and Recall (e.g., 95%)

Algorithm	100,000 Tuples in 5-D	~1,000 Tuples in 5-D	~1,000 Tuples in 10-D
Hybrid-Random	196 sec.	6 sec.	5 sec.
Subsampling	Hours	15 sec.	6 sec.
Pure-Random	423 sec.	140 sec.	710 sec.
Fixed-angle	Hours	302 sec.	Hours

Further Details

- See papers [KNZ01] for:
  - Details of algorithms, including complexity analysis
  - Comments on parameters (e.g., number of subsamples)
  - Examples of NHL outliers with and without a robust space transformation:
    - [without] - Hockey players who get a lot of penalties (e.g., Brad May, Chris Simon) may dominate other attributes
    - [with] - Players who do not necessarily have extreme values, but have unusual combinations of values (e.g., Jan Caloun, Joe Mullen)

Ongoing and Future Work

- Other datasets from non-hockey domains:
  - NASDAQ daily data
  - Mutual fund data from major Wall Street brokerage
  - Education datasets (labs, midterms, finals)
- Other improvements and optimizations
  - e.g., Analytic determination of "best" patch size, δ
- Compare our DSE results to other estimators (MCD, Fast MCD)

Take-Home Message

- We can efficiently identify meaningful outliers in large, multidimensional datasets.

- Outlier detection is a worthwhile data mining activity.

Cell-Based Algorithm

- Handles disk-resident data
  - Also, algorithm for memory-resident data
- Idea of cell-based approach:
  - **Quantize** tuples into cells
  - **Prune** cells that can't be outliers
- Wherever possible, do cell-by-cell processing, rather than tuple-by-tuple!
- $O(m\, c^k\, k^{k/2} + N)$

A 2-D Cell-Structure

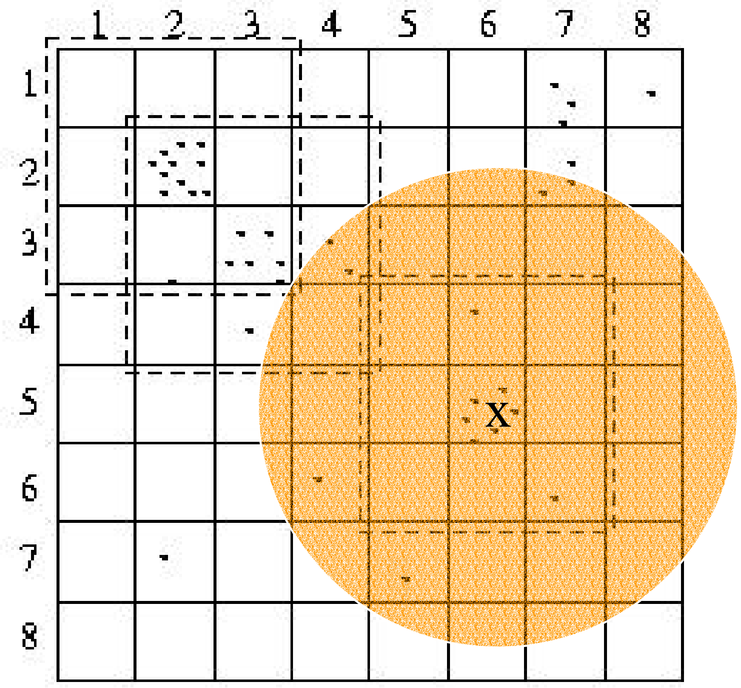
- Cell length l =

  $\boldsymbol{D / (2 \sqrt{k})}$
- Diagonal = $\boldsymbol{D/2}$
- Layer 1 is

  one cell thick
- Layer 2 is

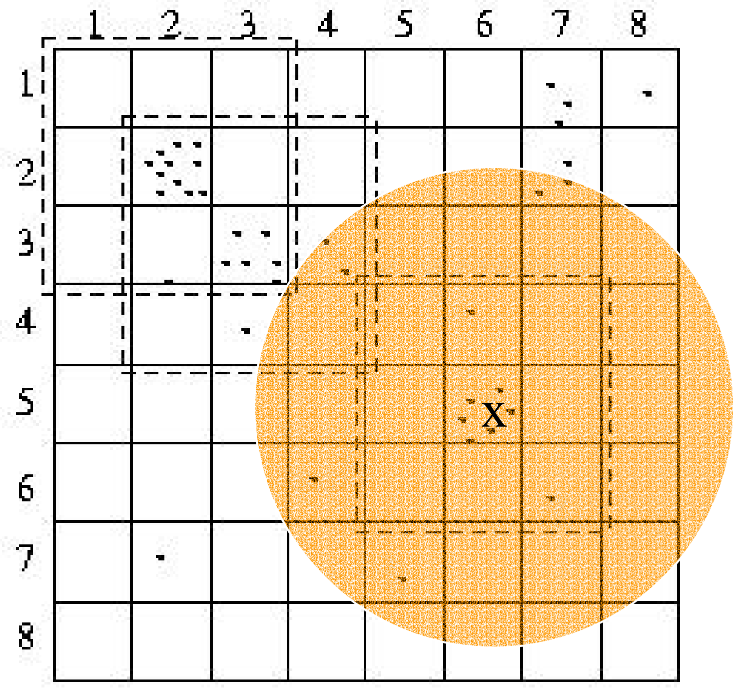
  $\lceil \boldsymbol{2 \sqrt{k} - 1} \rceil$

  cells thick



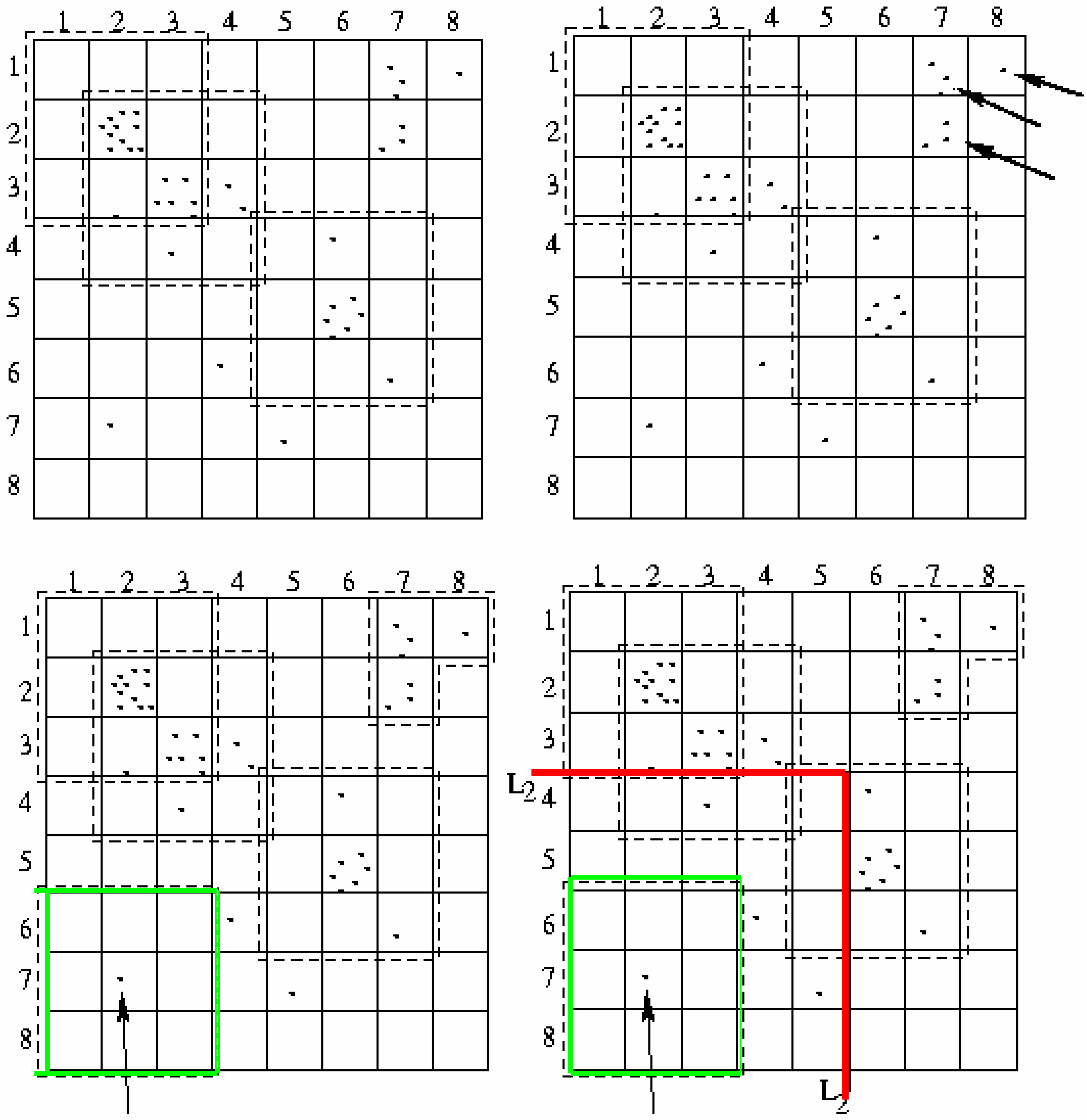
2-D Cell-Structure, cont.

- If > *M* objects in a cell *C*, then *none* of those objects is an outlier
- If > *M* objects in *C* ∪ {Layer 1}, then no obj. in C is an outlier
- If ≤ *M* objects in *C* ∪ {Layer 1}∪{Layer 2}, then *all* objects in *C* are outliers



M=4

No More Than 4 Pts. in the D-nbhd of an outlier



4 Phases of I/O in Cell-Based Algorithm

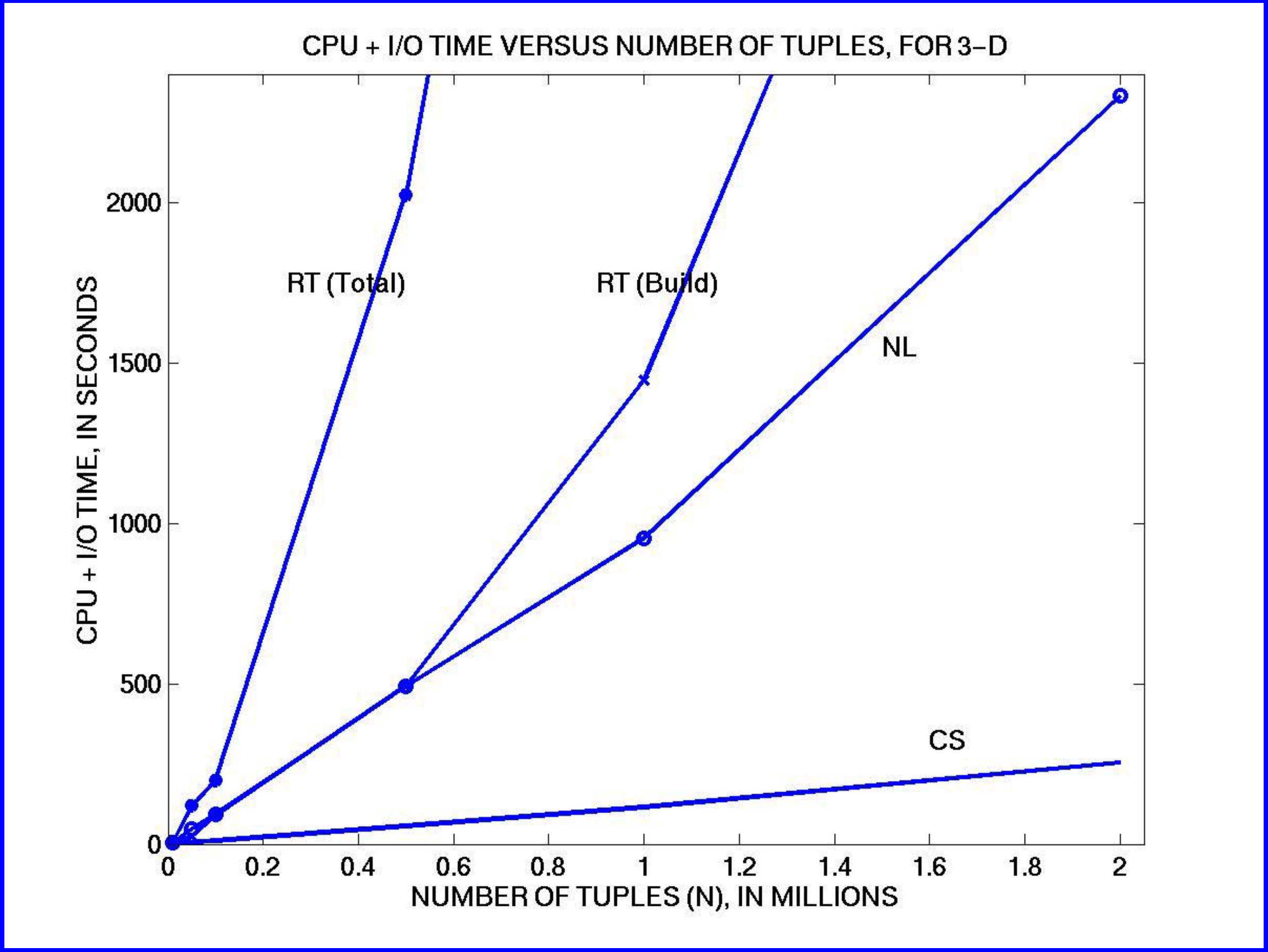
1. Read all pages (quantization)
2. Read **Class I** pages (pages containing some *white* tuples)
3. Read **Class II** pages (pages containing only *non-white* tuples, needed for tuple-by-tuple comparisons)
4. Repeat [2]

4 Phases of I/O, cont.

Key Points:

- Class I and Class II pages are mutually exclusive
- Each page is guaranteed to be read no more than 3 times

How Total Time Scales with *N* for 3-D Disk-Resident Datasets



Experimental Results (in seconds)

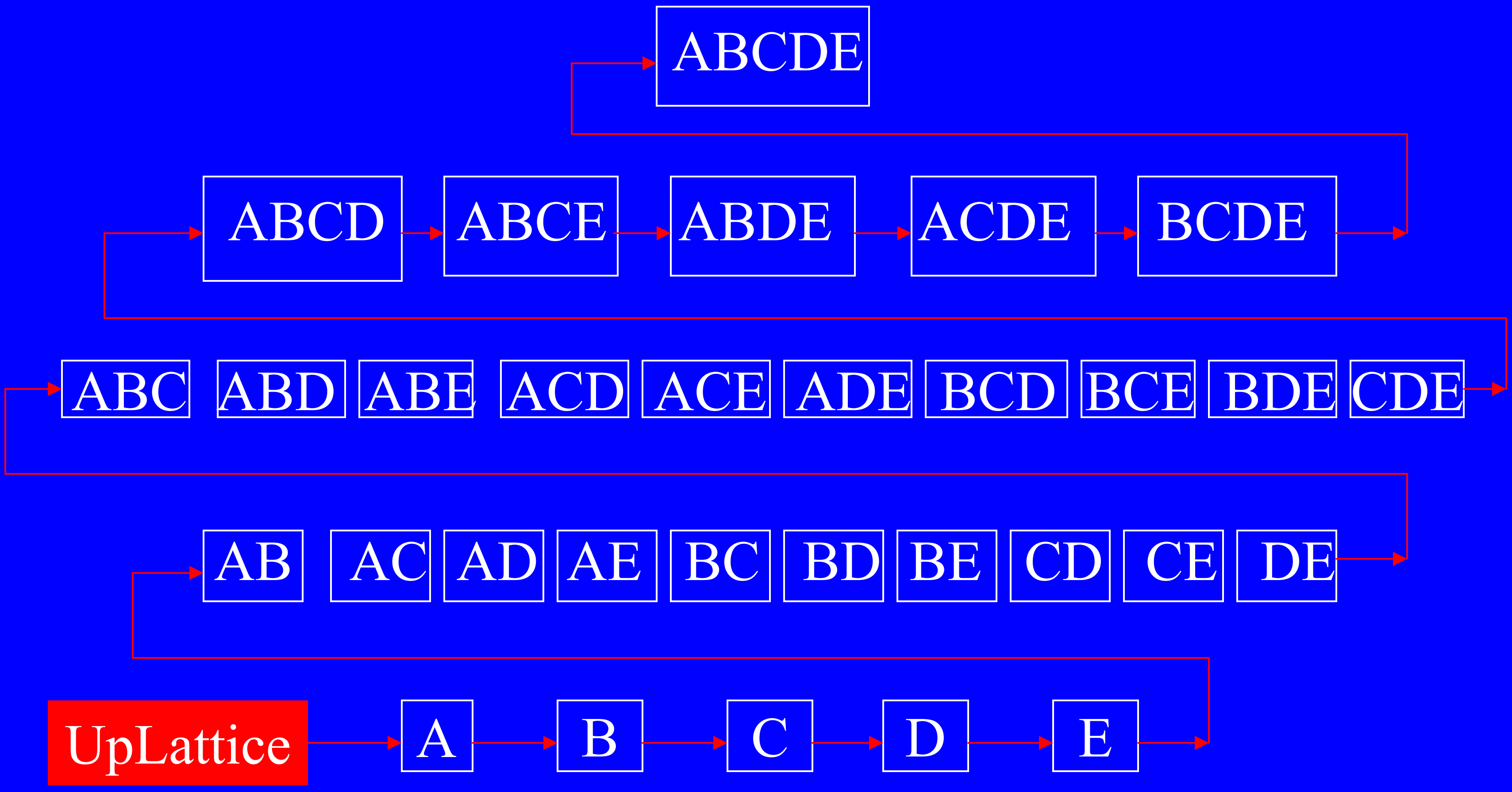
- If k ≤ 4, use cell-based alg.; else use NL alg.

N	3-D CS	3-D NL	4-D CS	4-D NL	5-D CS	5-D NL
500,000	57	491	114	224	695	148
2,000,000	254	2332	607	1421	>>2147	1556
5,000,000	497	4811	1140	3651		

Computing Intensional Knowledge

I/O Savings Realized Due to Sharing

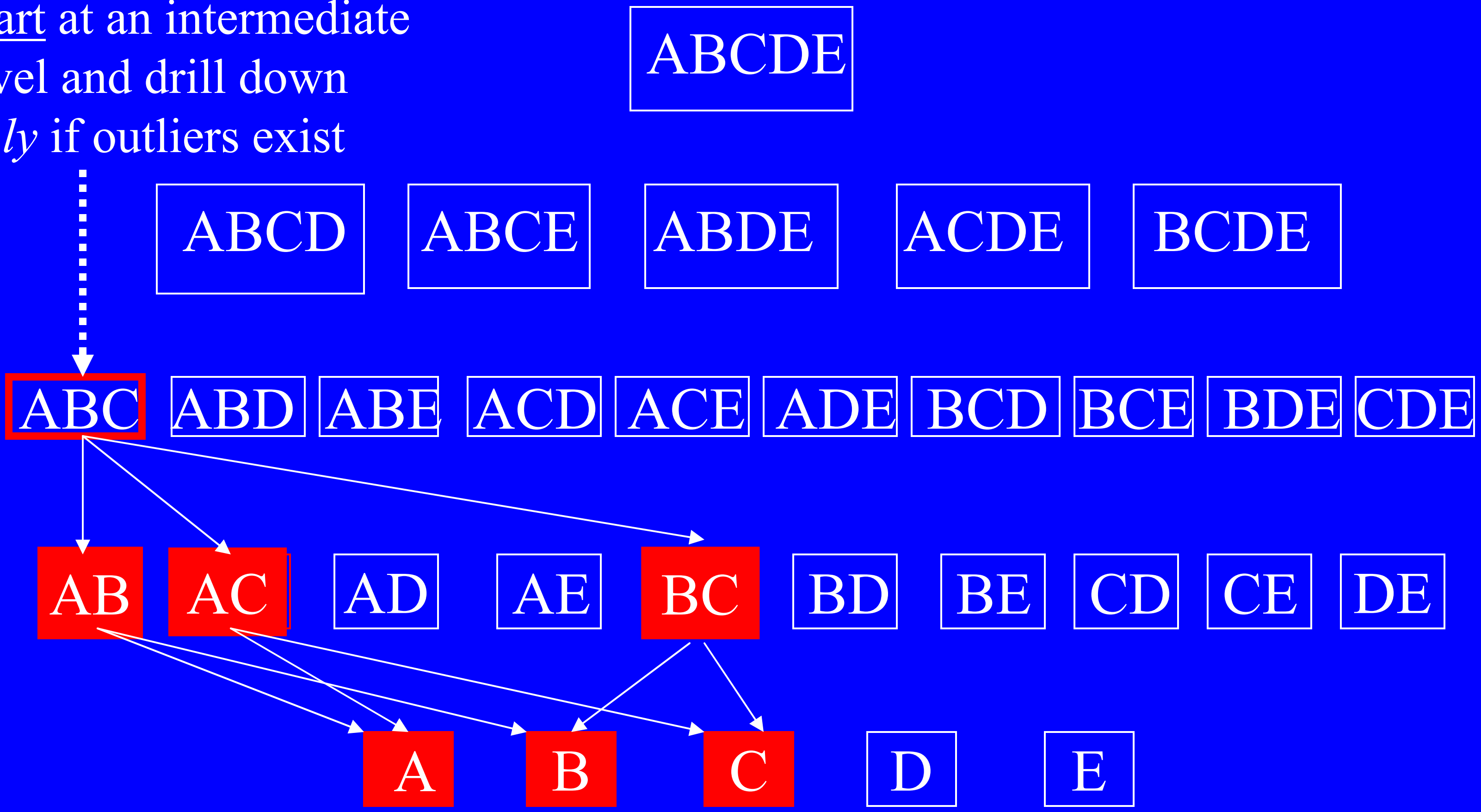
Strategy 1: Algorithm UpLattice

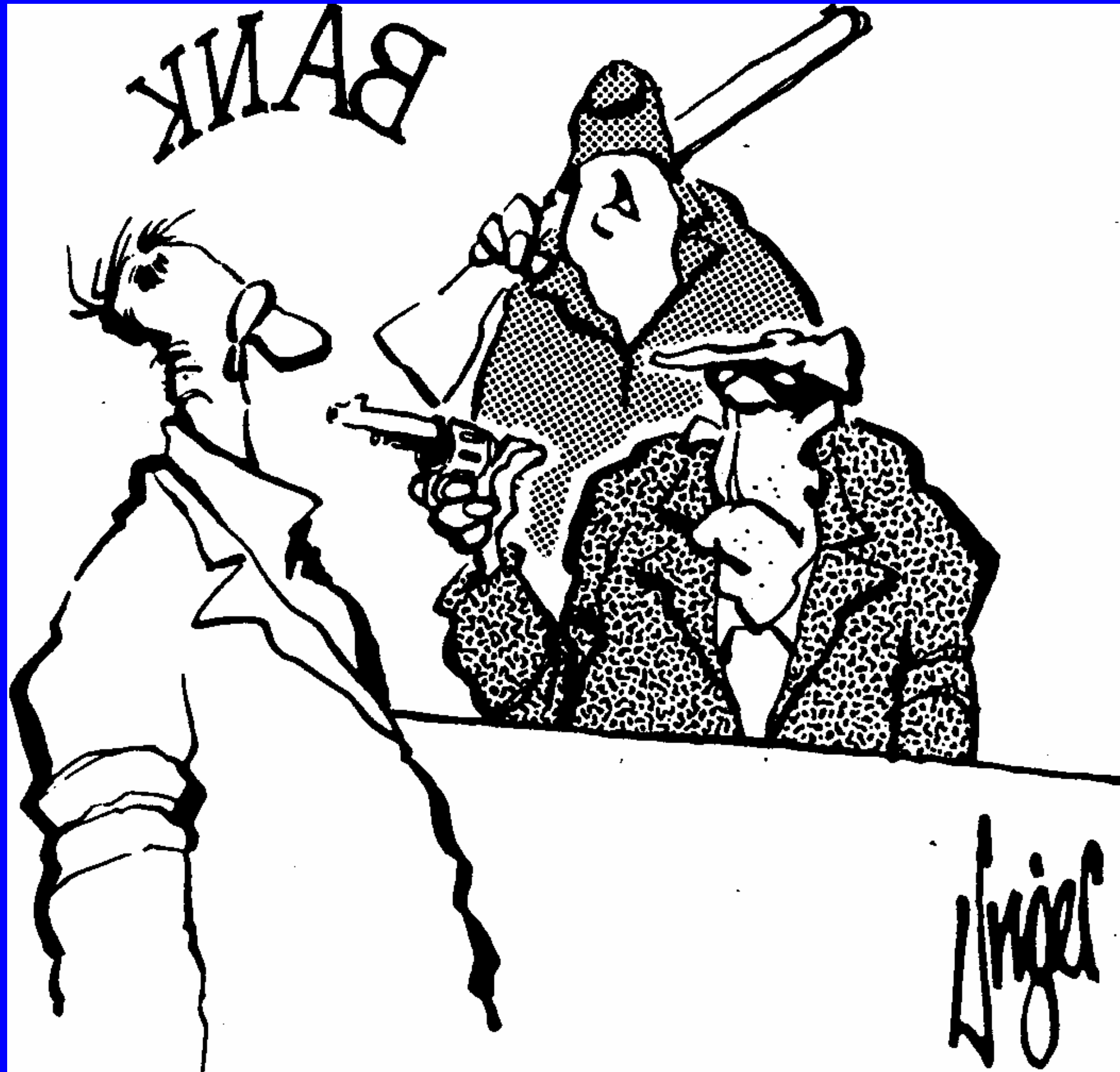


(bottom-up, level-wise)

Strategy 2: JumpLattice with **DrillDown**

<u>Start</u> at an intermediate level and drill down *only* if outliers exist





"I'm gonna pay my electricity bill while I'm here."

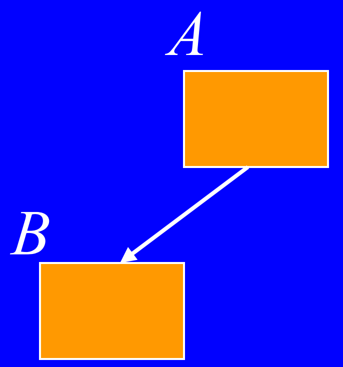
Strategy 3: **Path** (Grouped) Processing

- P is a white tuple in B $\Rightarrow$ P is a white tuple in A

$$\therefore\ WT_B \subseteq WT_A$$

- The set of <u>Class I pages</u> needed to process both spaces simultaneously is given by:

$$PgI(WT_A \cup WT_B) = PgI(WT_A)$$



… and the combined set of <u>Class II pages</u>:

$$PgII(WT_A - WT_B, A) \cup PgII(WT_B, B)$$

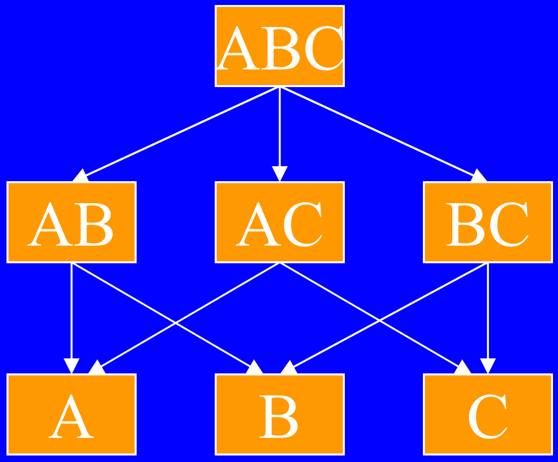
Strategy 4: **Semi-Lattice** (Grouped) Processing

- Combined set of Class I pages: $PgI(WT_{ABC})$
- ... of Class II pages:

$PgII(WT_A, A) \cup \ldots \cup PgII(WT_C, C) \cup \mathrm{PgII}(WT_{AB} - WT_A - WT_B, AB) \cup \ldots \cup PgII(WT_{ABC} - WT_{AB} - \ldots - WT_A - \ldots), ABC)$

Example:

Space *ABC* is top-element for attributes A, B, & C



Summary of I/O Sharing (Path and Semi-Lattice)

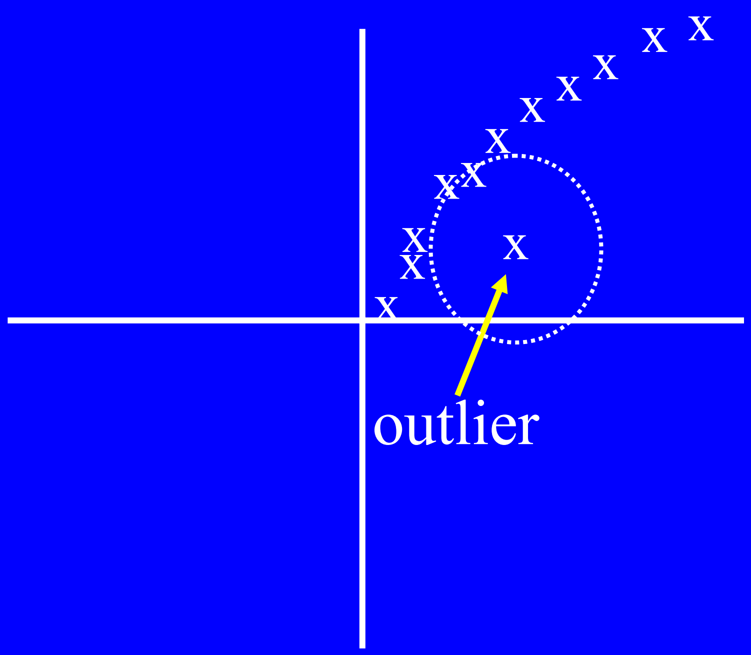
- Have similar performance for most scenarios
- Usually better than **UpLattice** or **DrillDown**
- Both benefit from shared processing when finding top-*u* non-trivial outliers
  - 65-75% savings in I/O than if each space is handled separately (i.e., no sharing)
- Overkill if no outliers exist (esp. Semi-Lattice, which needs more memory than Path)

Robust Statistics

- *Robust* algorithms are able to *accommodate* (i.e., minimize the impact of) outliers
- Outliers can radically affect distance-based operations
  - Consider mean μ vs. median *M:*
    - single outlier can greatly affect μ
    - single outlier is unlikely to change *M* by much

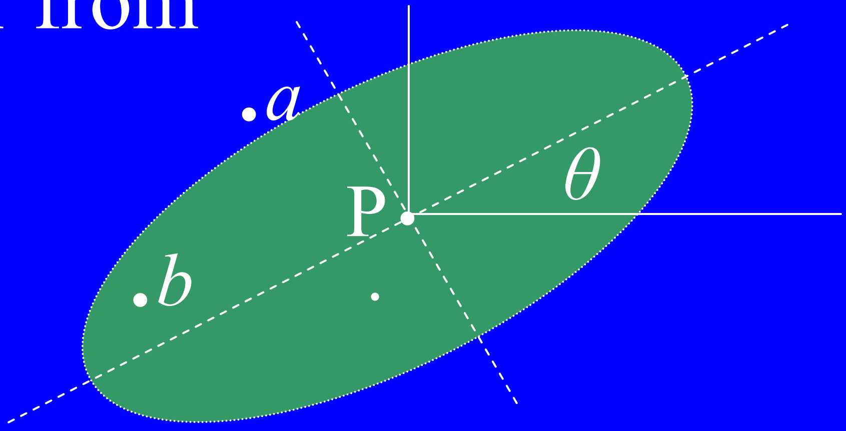
Outliers and *D*-Neighbourhoods

- Is the notion of a D-neighbourhood meaningful if the attributes have different scale, variability, and correlation?



Statistical Distances

- In the presence of variability, differing scales, and correlation, all δ-neighbours lie within an ellipse (hyperellipsoid)
  - Correlation => ellipse is rotated by θ
- Figure: *a* is further from P than *b* is



Quantifying Location and Scatter

- We seek robust estimates of *location* (center of cloud of points) and *scatter* (variability)
- In 1-D, μ and σ^2 are scalars; in k-D, this extends to:
  - $\boldsymbol{\mu}$: a vector of k scalars
  - $\boldsymbol{\Sigma}$: a symmetric k x k matrix of covariances, where:
    - entry ij is the *covariance* of attributes Y_i and Y_j
- Covariance of two random variables is a measure of their joint variability (or degree of association)

Introduction to Donoho-Stahel Estimator (DSE)

DSE Properties

Key Properties for Distance-based Operations

1. Euclidean property
   - Can use Euclidean distances after transformation
     - Results in overall efficiency (e.g., [KN98])
2. Stability property
   - Particularly important for database operations because of frequent updates
     - Addition and/or deletion of n_0 points does not affect DSE much

Precision and Recall

- Use *precision* and *recall* [S83] to evaluate quality of results
  - Let A = answer set (outliers returned by a test)
  - Let B = target set of “actual” outliers given by a suitably fine Fixed-angle interval
- Define:

  (1) *Precision* = % of outliers in A that are in B

  (2) *Recall* = % of outliers in B that are in A

DSE Algorithms: Selection of Projection Vectors

Conclusions

Conclusions: Identifying *DB*-Outliers

- We gave 2 kinds of algorithms for identifying distance-based outliers in large, *disk-resident* datasets:
  - **Cell-based**: $O(m\, c^k\, k^{k/2} + N)$, best for $k \leq 4$
  - **Nested-loop**: $O(k\, N^2)$, best for $k \geq 5$

Conclusions: Computing Intensional Knowledge

- We provided a notion of strength: strongest, weak, and trivial outliers
- We presented 4 strategies for finding *non-trivial* outliers:
  - UpLattice
  - JumpLattice with DrillDown
  - JumpLattice with Path
  - JumpLattice with Semi-Lattice
- Path is our recommended strategy
- Recommend entry level k=3

Conclusions: Robust Space Transformations

- Must account for scale, variability, correlation, and outliers in many data mining applications
  - Use robust statistics to improve quality and meaningfulness of results
- We recommend DSE; it possesses:
  - Euclidean property
  - Stability property

Conclusions: DSE

- Use Hybrid-random with 400-1000 patches, depending on level of recall desired
  - Suggested Default: $\delta = 0.1581$; patches = 1000
- Hybrid-Random can provide excellent DSE:
  - in 1-3 minutes for 100,000 tuples in 5-D
  - in 5 seconds for 855 tuples in 10-D