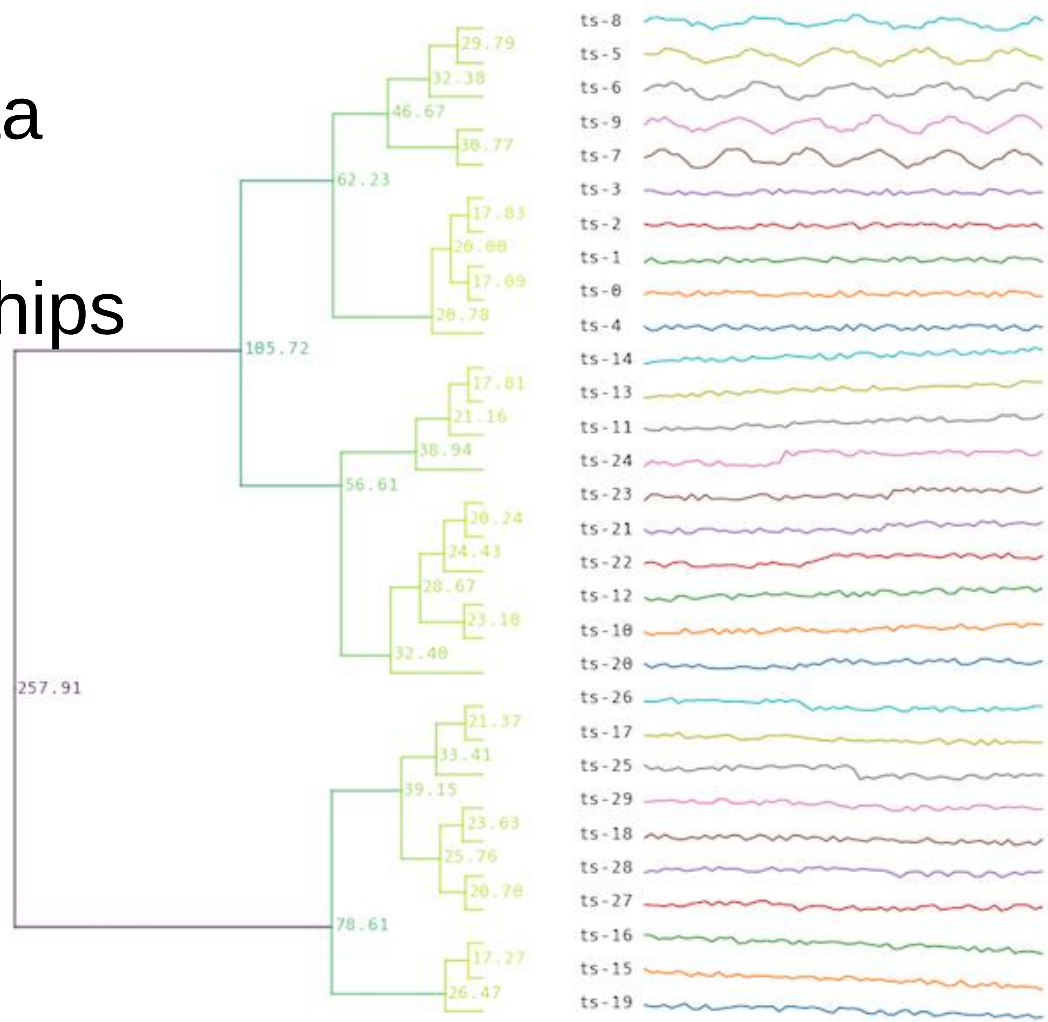
HIERARCHICAL CLUSTERING

Jesse Davis

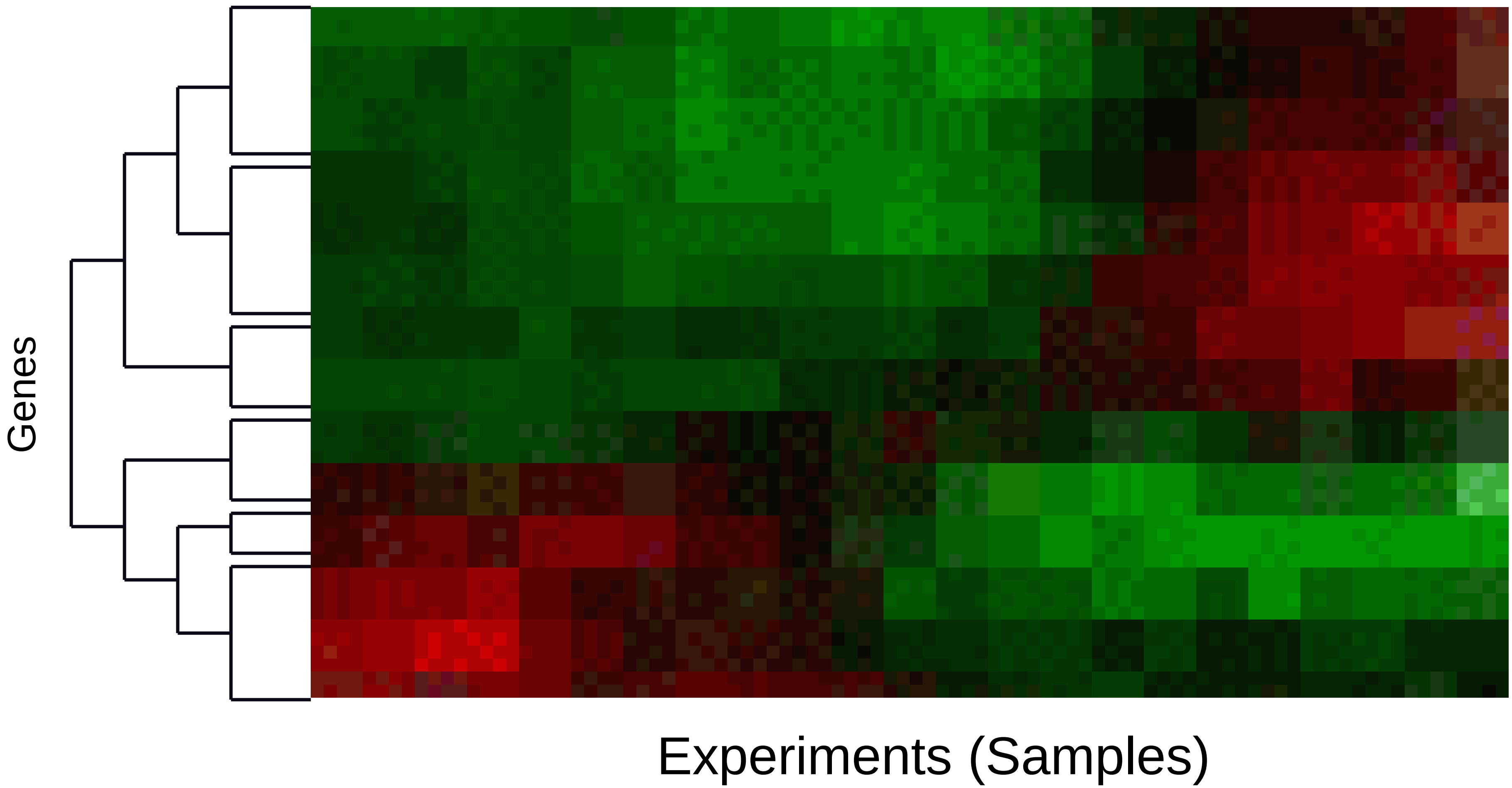
Hierarchical Clustering

- Model hierarchical structure in the data
- Captures relationships among clusters



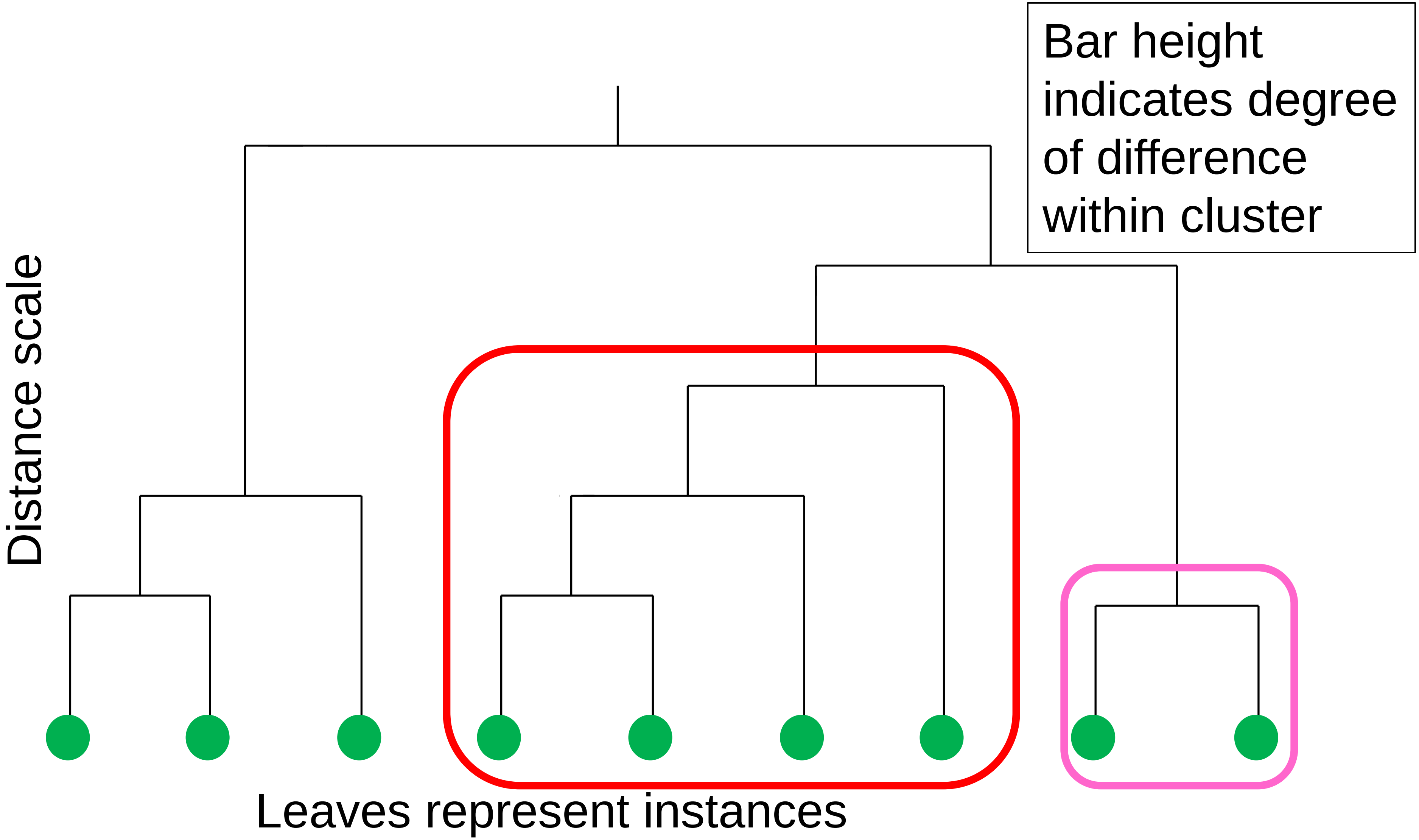
Example: Gene Expression

(Green = up-regulated, Red = down-regulated)



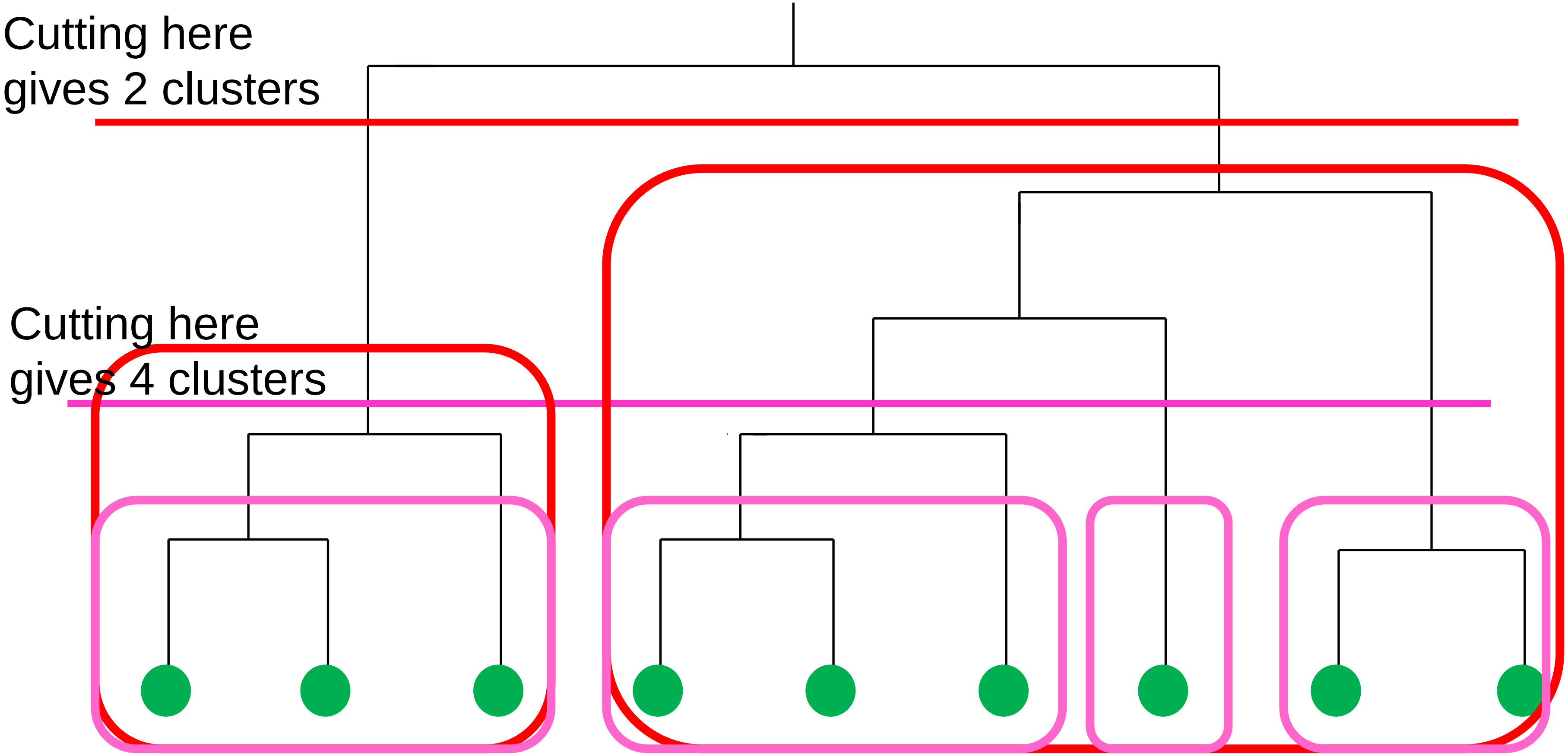
Basics

Hierarchical Clustering: Dendogram



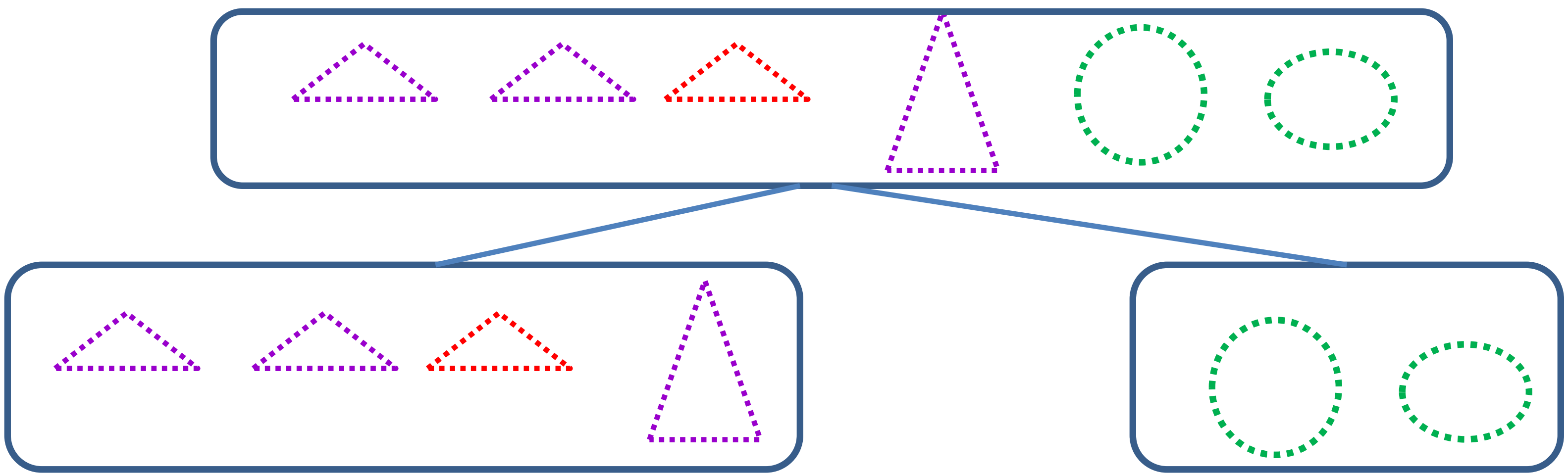
Note: Partitional Clustering from a Hierarchical Clustering

Can generate a partitional clustering from a hierarchical clustering by “cutting” the tree at some level

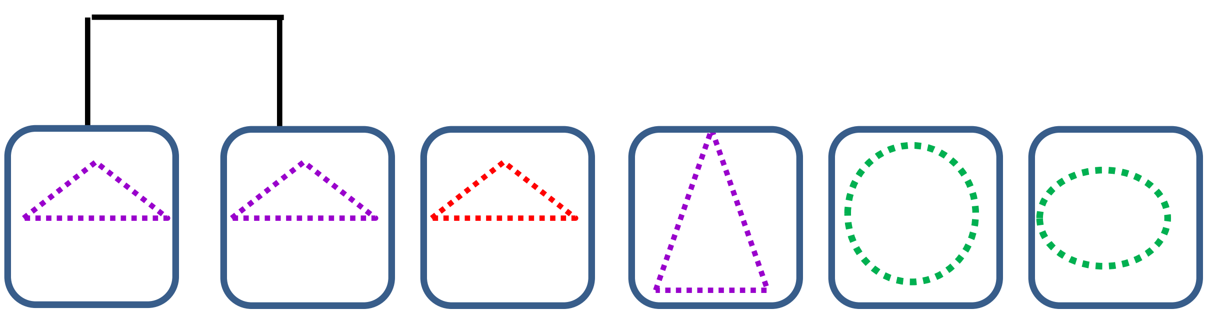


Hierarchical Clustering Approaches

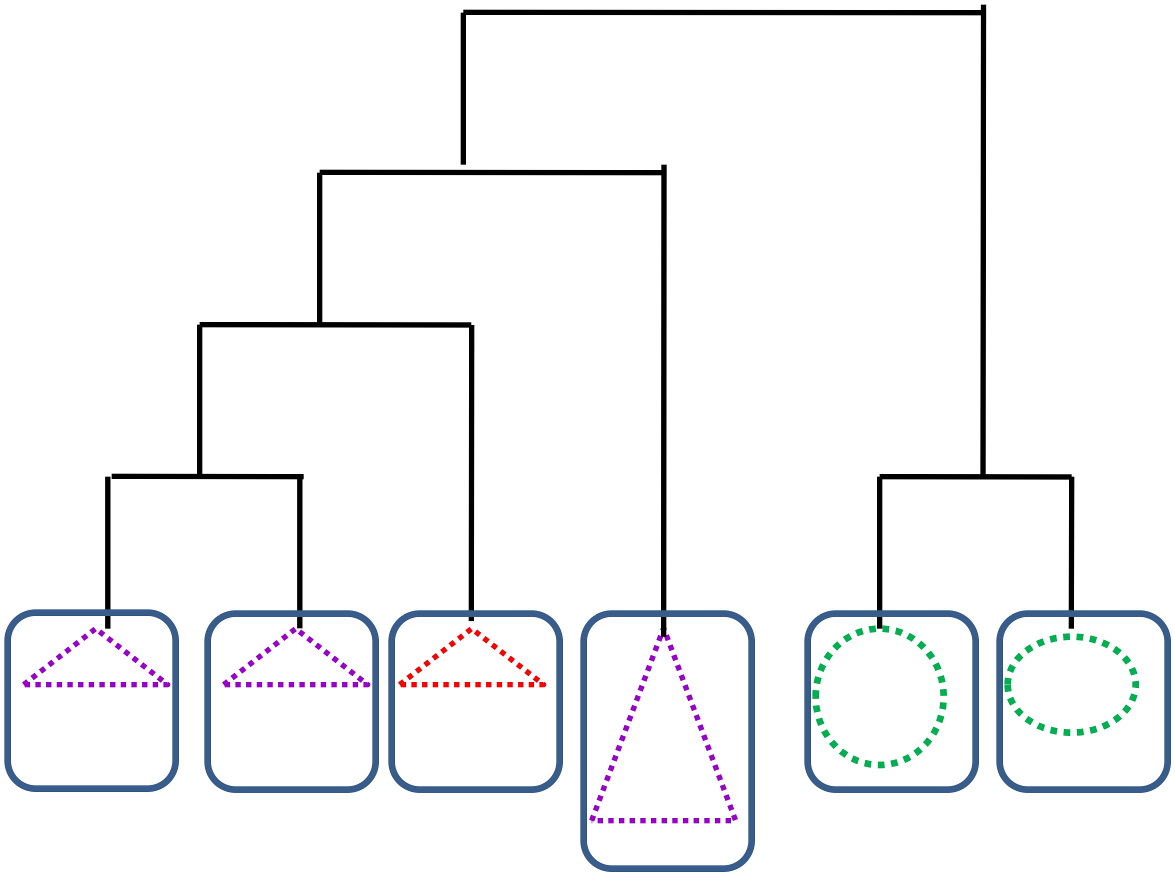
- Top-down or divisive



- Bottom-up or agglomerative



Bottom-Up Example



Bottom-Up Hierarchical Clustering

Given: instances $x_1,\ldots,x_n$

For i = 1 to n, $c_i = \{x_i\}$

$C = \{c_1,\ldots,c_n\}$

j = n

While |C| > 1

  j = j+1

  $(c_a,c_b) = \text{argmin dist}(c_a,c_b)$

  $c_j = c_a \cup c_b$

  add node to tree joining a and b

  $C = (C - \{c_a,c_b\}) \cup c_j$

Return tree with root node j

Key question: Measuring distance

Distance Matrix

$$D = \begin{bmatrix} 0 & & & & \\ d(2,1) & 0 & & & \\ d(3,1) & d(3,2) & 0 & & \\ \vdots & \vdots & \vdots & \ddots & \\ d(n,1) & d(n,2) & d(n,3) & \cdots & 0 \end{bmatrix}$$

Initial Distance Matrix for a Data Set with 5 Examples

1) Form five clusters, one for each example
2) Compute pairwise distance between initial clusters (=pairwise distance between examples)

	1	2	3	4	5
1	**0**				
2	**7.4**	**0**			
3	**0.7**	**7.1**	**0**		
4	**7.3**	**0.3**	**7.0**	**0**	
5	**0.5**	**6.9**	**0.6**	**6.8**	**0**

Find Two Closest Clusters

	1	2	3	4	5
1	**0**				
2	**7.4**	**0**			
3	**0.7**	**7.1**	**0**		
4	**7.3**	**0.3**	**7.0**	**0**	
5	**0.5**	**6.9**	**0.6**	**6.8**	**0**

Smallest value in matrix

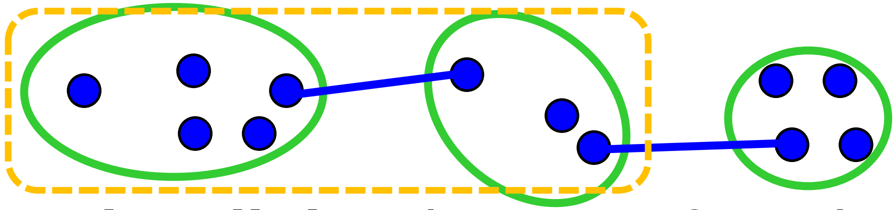
Update Distance Matrix

	1	(2,4)	3	5
1	**0**			
(2,4)	**?**	**0**		
3	**0.7**	**?**	**0**	
5	**0.5**	**?**	**0.6**	**0**

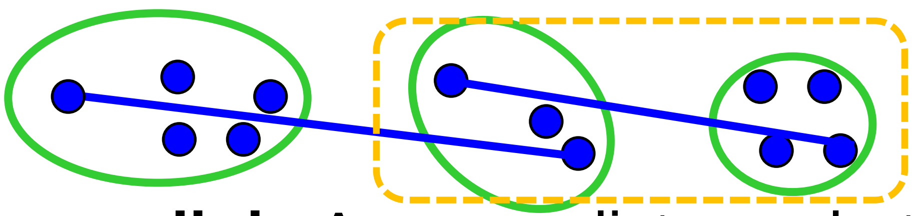
Question: What is the distance between the new cluster (2,4) and the other three clusters?

Measuring the Distance Between Two Clusters

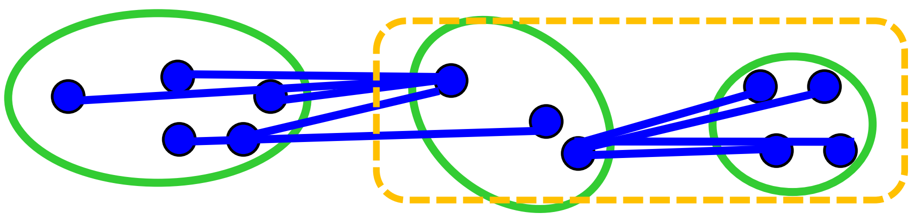
- **Single link:** Distance of two most similar instances: dist(c_u, c_v) = min{dist(a, b) | a∈c_u, b∈c_v}



- **Complete link:** Distance of two least similar instances: dist(c_u, c_v) = max{dist(a, b) | a∈c_u, b∈c_v}



- **Average link:** Average distance between instances: dist(c_u, c_v) = avg{dist(a, b) | a∈c_u, b∈c_v}



Efficient Distance Updates

- If we merged and c_u and c_v into c_j, we can determine distance to every other cluster:
  - Single link:

$$dist(c_j, c_k) = \min(dist(c_u, c_k), dist(c_v, c_k))$$

  - Complete link:

$$dist(c_j, c_k) = \max(dist(c_u, c_k), dist(c_v, c_k))$$

  - Average link:

$$dist(c_j, c_k) = \frac{|c_u| * dist(c_u, c_k) + |c_v| * dist(c_v, c_k)}{|c_u| + |c_v|}$$

Note: The linkage choice is a hyper parameter for the bottom-up clustering algorithm

Illustrative Example Updates for Each Linkage Criteria

Single

	2	4	(2,4)
1	7.4	7.3	**7.3**
3	7.1	7.0	**7.0**
5	6.9	6.8	**6.8**

min

Complete

	2	4	(2,4)
1	7.4	7.3	**7.4**
3	7.1	7.0	**7.1**
5	6.9	6.8	**6.9**

max

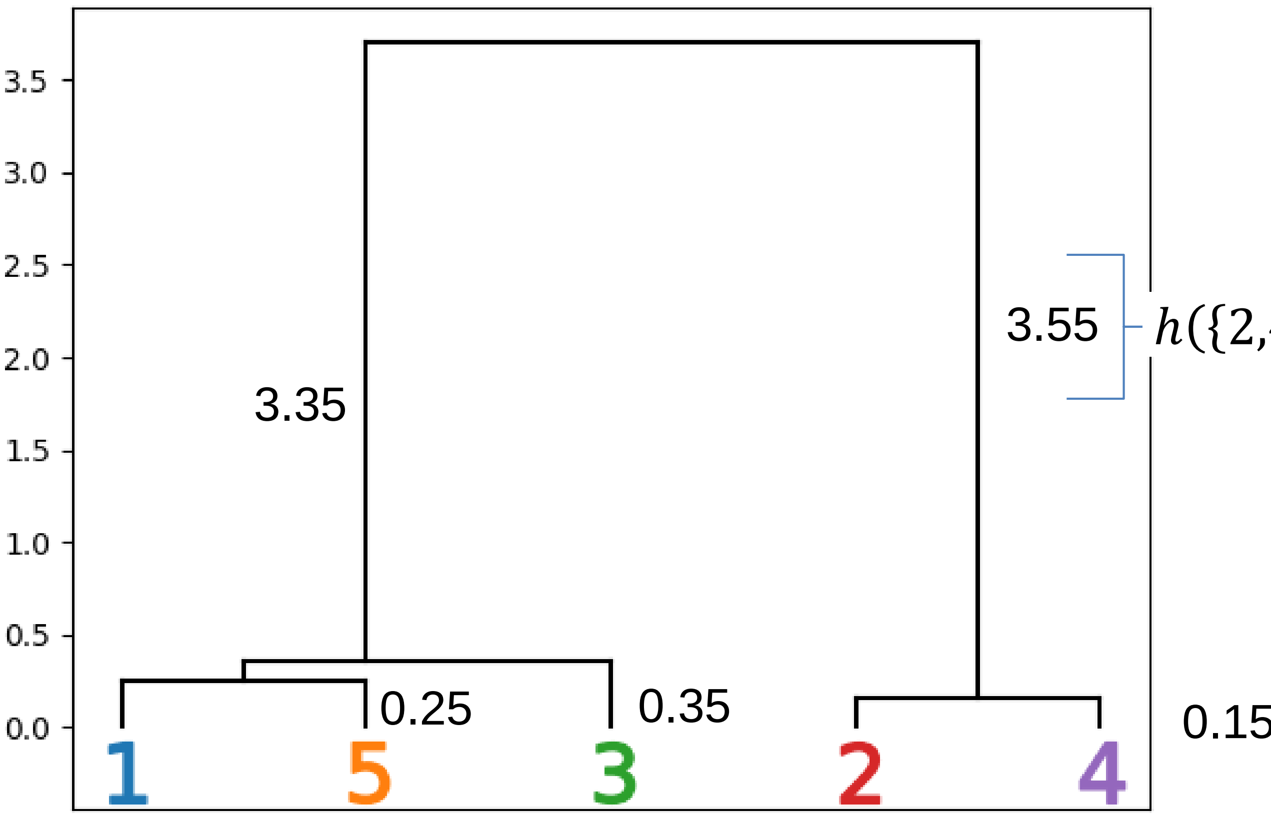
Average

	2	4	(2,4)
1	7.4	7.3	**7.35**
3	7.1	7.0	**7.05**
5	6.9	6.8	**6.85**

(Weighted) mean

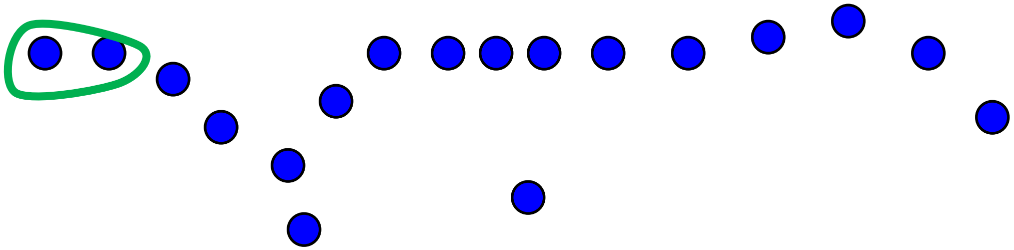
Complete Link Dendogram for Sample Dataset

$$\text{height} = \mathrm{h}(c_j, c_k) = \frac{dist(c_j, c_k)}{2}$$

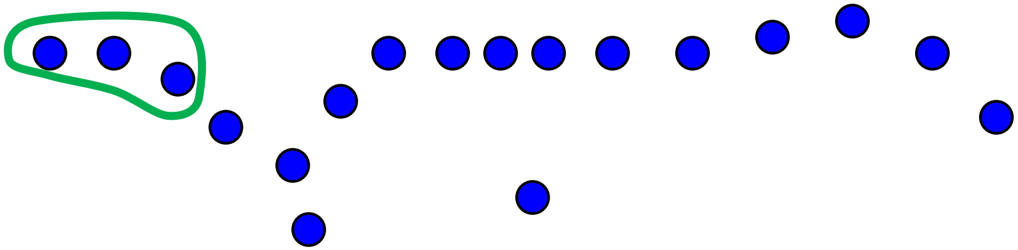


$$h(\{2,4\},\{1,5,3\}) - h\ (\{2\},\{4\})$$

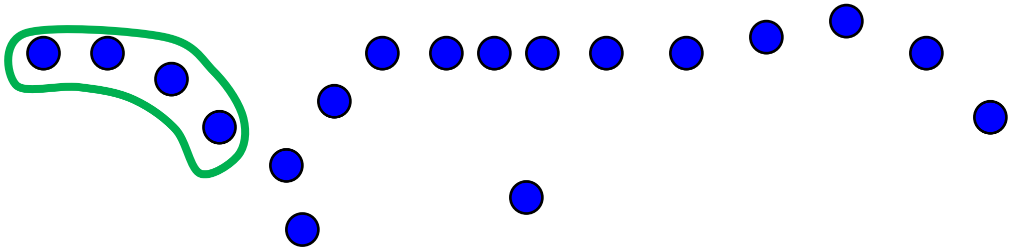
Single Link: Chaining



Single Link: Chaining

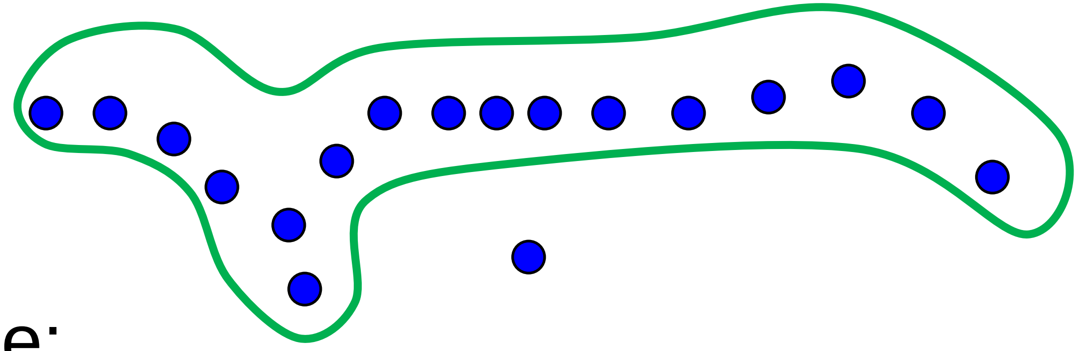


Single Link: Chaining



Single Link

- Chaining:



- Bottom line:
  - Simple, fast
  - Often low quality

Complete Link Hierarchical Summary

- Complexity: $O(n^3)$
  - $O(n^2)$ to build initial similarity matrix
  - $O(n)$ for the merges
- Fast algorithm: Requires $O(n^2)$ space
- Bottom line
  - Typically much faster than $O(n^3)$
  - Often good quality
  - No Chaining

Advanced Hierarchical Clustering

Other Hierarchical Clustering Methods

- Weaknesses of agglomerative clustering methods
  - **Do not scale well:** time complexity of at least $O(n^2)$, where n is the number of total objects
  - Can never undo what was done previously

- Integration of hierarchical with distance-based clustering
  - BIRCH: uses CF-tree and incrementally adjusts the quality of sub-clusters
  - CURE: selects well-scattered points from the cluster and then shrinks them towards the center of the cluster by a specified fraction

BIRCH: Balanced Iterative Reducing and Clustering using Hierarchies

- Incrementally construct a Clustering Feature (CF) tree
  - **Phase 1:** Scan DB to build an initial in-memory CF tree (each node: #points, sum, sum of squares)
  - **Phase 2:** use an arbitrary clustering algorithm to cluster the leaf nodes of the CF-tree

- ***Scales linearly*****:** Finds a good clustering with a single scan

- ***Weaknesses:*** handles only numeric data, sensitive to order of data records

Definitions

- **Centroid:** $\mu = \frac{1}{N}\sum_{i=1}^{N} x_i$

Average along each dimension

- **Radius:** Average distance from member points to cluster centroid

$$R = \sqrt{\frac{\sum_{i=1}^{N}\sum_{j=1}^{d} (x_{ij} - \mu_j)^2}{N}}$$

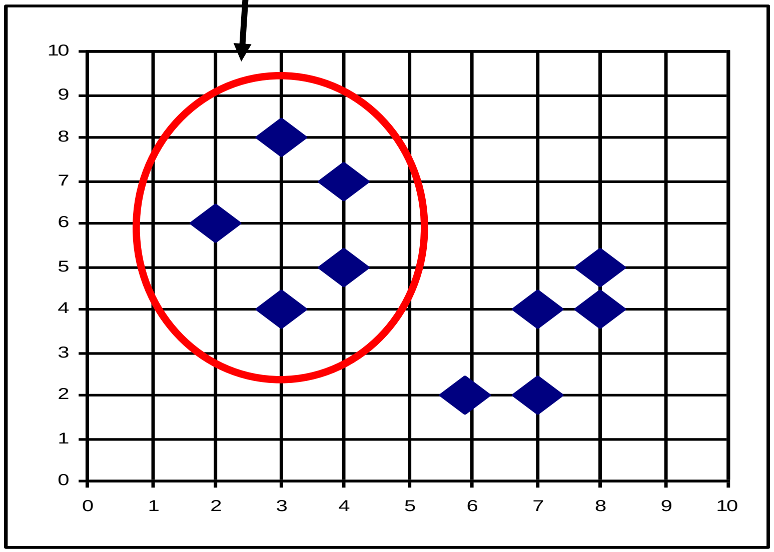
  - Captures tightness of cluster around centroid
  - Note: Math and verbal definition not 100% aligned, but these come directly from the paper

Cluster Feature Vector

- Given: $X_1, \ldots, X_n$, data points in a cluster where each with d-dimensions
- We define $\mathrm{CF} = (\mathrm{N}, \mathrm{LS}, \mathrm{SS})$, where
  - N: Number of data points
  - LS_j: $\sum_{i=1}^{n} x_{ij}$
  - SS_j: $\sum_{i=1}^{n} x_{ij}^2$
- Note: CFs are additive!
  - E.g., $\mathrm{CF}_1 + \mathrm{CF}_2 = (\mathrm{N}_1{+}\mathrm{N}_2, \mathrm{LS}_1{+}\mathrm{LS}_2, \mathrm{SS}_1{+}\mathrm{SS}_2)$

Cluster Feature Example

CF = (5, (16,30), (54,190))



(3,4)
(2,6)
(4,5)
(4,7)
(3,8)

$LS_x = 3 + 2 + 4 + 4 + 3 = 16$

$LS_y = 4 + 6 + 5 + 7 + 8 = 30$

$SS_x = 3^2 + 2^2 + 4^2 + 4^2 + 3^2 = 54$

$SS_y = 4^2 + 6^2 + 5^2 + 7^2 + 8^2 = 190$

Comments

- $2d + 1$ values represent any number of points
  - d = number of dimensions
- **Average in each dimension j:** LS_j / N
- **Variance in each dimension *j*:** $(SS_j / N) - (LS_j / N)^2$
  - To get standard deviation take square root
- Can also compute the radius (next slide)

Radius Derivation

$$R = \sqrt{\frac{\sum_{i=1}^{N}\sum_{j=1}^{d}\,(x_{ij}-\mu_j)^2}{N}} = \sqrt{\frac{\sum_{j=1}^{d}\sum_{i=1}^{N}(x_{ij}^2\; - 2x_{ij}\mu_j + \mu_j^2)}{N}}$$

Definition of SS

Definition of LS

$$= \sqrt{\frac{\sum_{j=1}^{d}(\sum_{i=1}^{N} x_{ij}^2) - 2\mu_j\,(\sum_{i=1}^{N} x_{ij}) + (N\mu_j^2))}{N}}$$

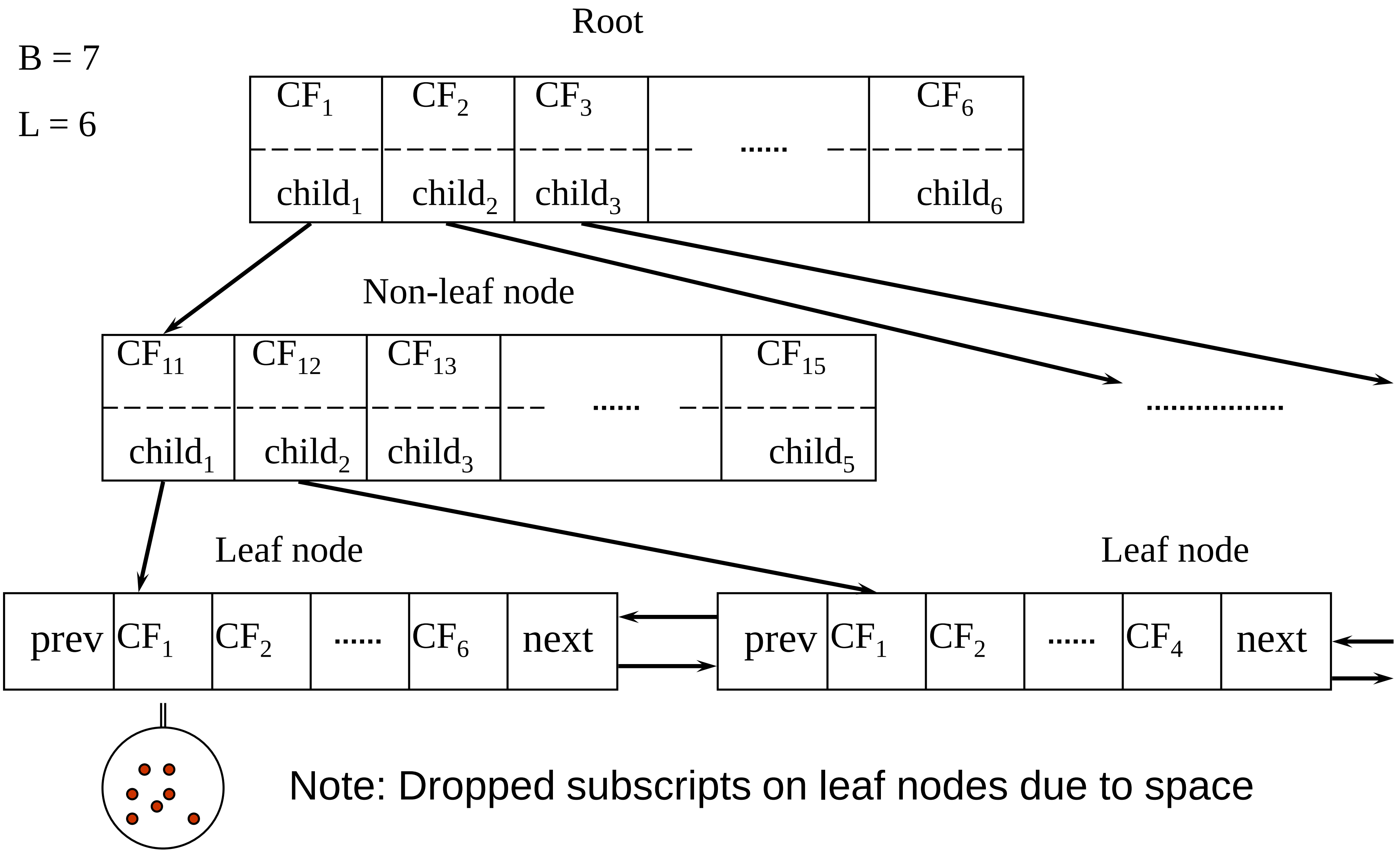
Definition of centroid

$$= \sqrt{\frac{\sum_{j=1}^{d} SS_j - 2\,\frac{LS_j}{N}\,LS_j + N\left(\frac{LS_j}{N}\right)^2}{N}}$$

Cluster Feature Tree

- A CF-tree is a height-balanced tree with two parameters:
  - Branching factor (non leaf nodes B, leaf nodes, L)
  - Threshold T
- Each non leaf node has the form [CF_i, $child_i$]
- Each leaf node has CF
  - Set of CFs
  - Two pointers: prev and next
- Radius of a subcluster under a leaf node can not exceed the threshold T

CF Tree



Note: Dropped subscripts on leaf nodes due to space

CF-Tree Construction

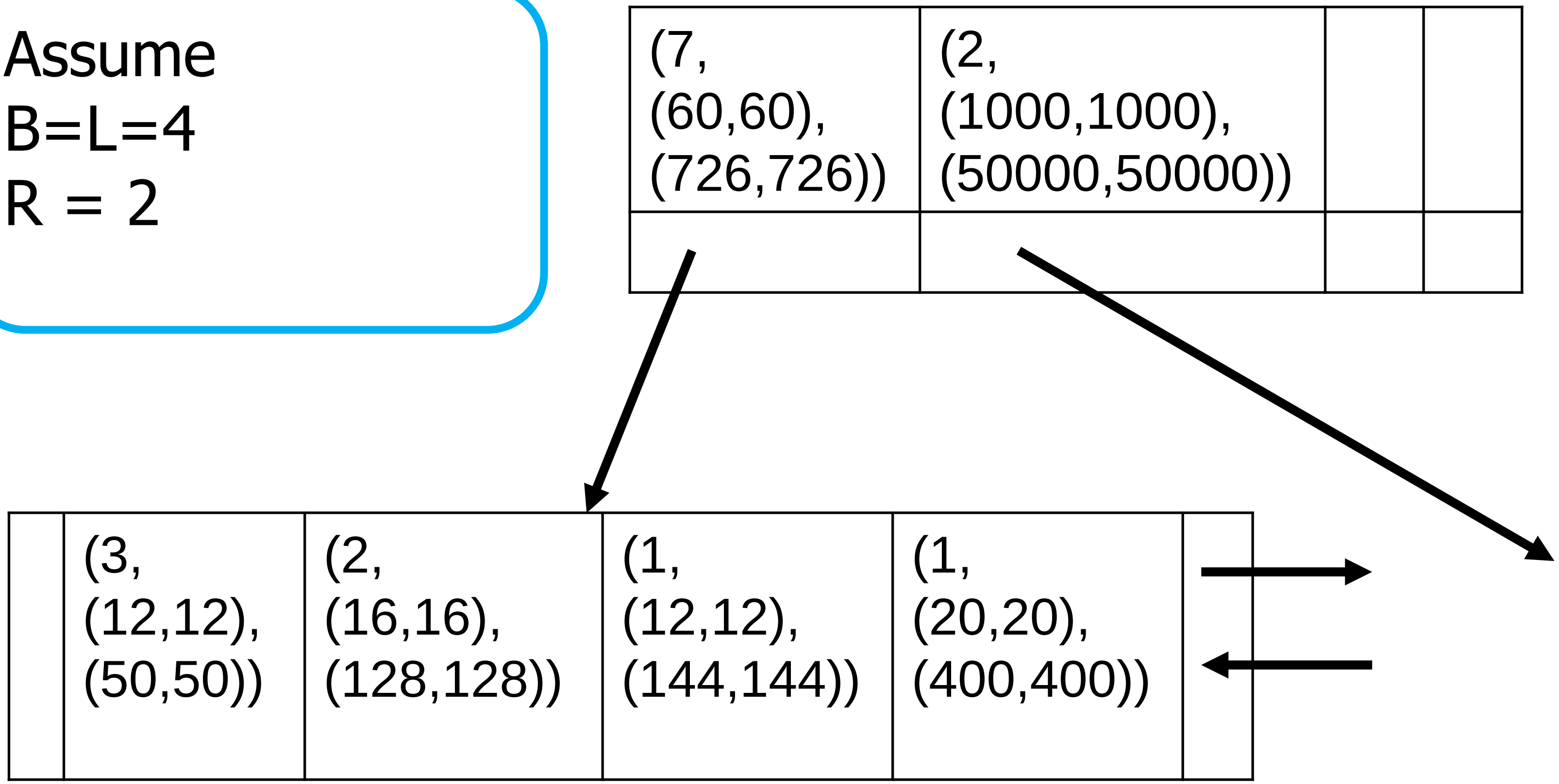
- Scan data set and insert the incoming data instances into the CF tree one by one
- Each instance is inserted into the closest subcluster under a leaf node
- If insertion causes subcluster radius to exceed threshold, then create new subcluster

CF-Tree Construction

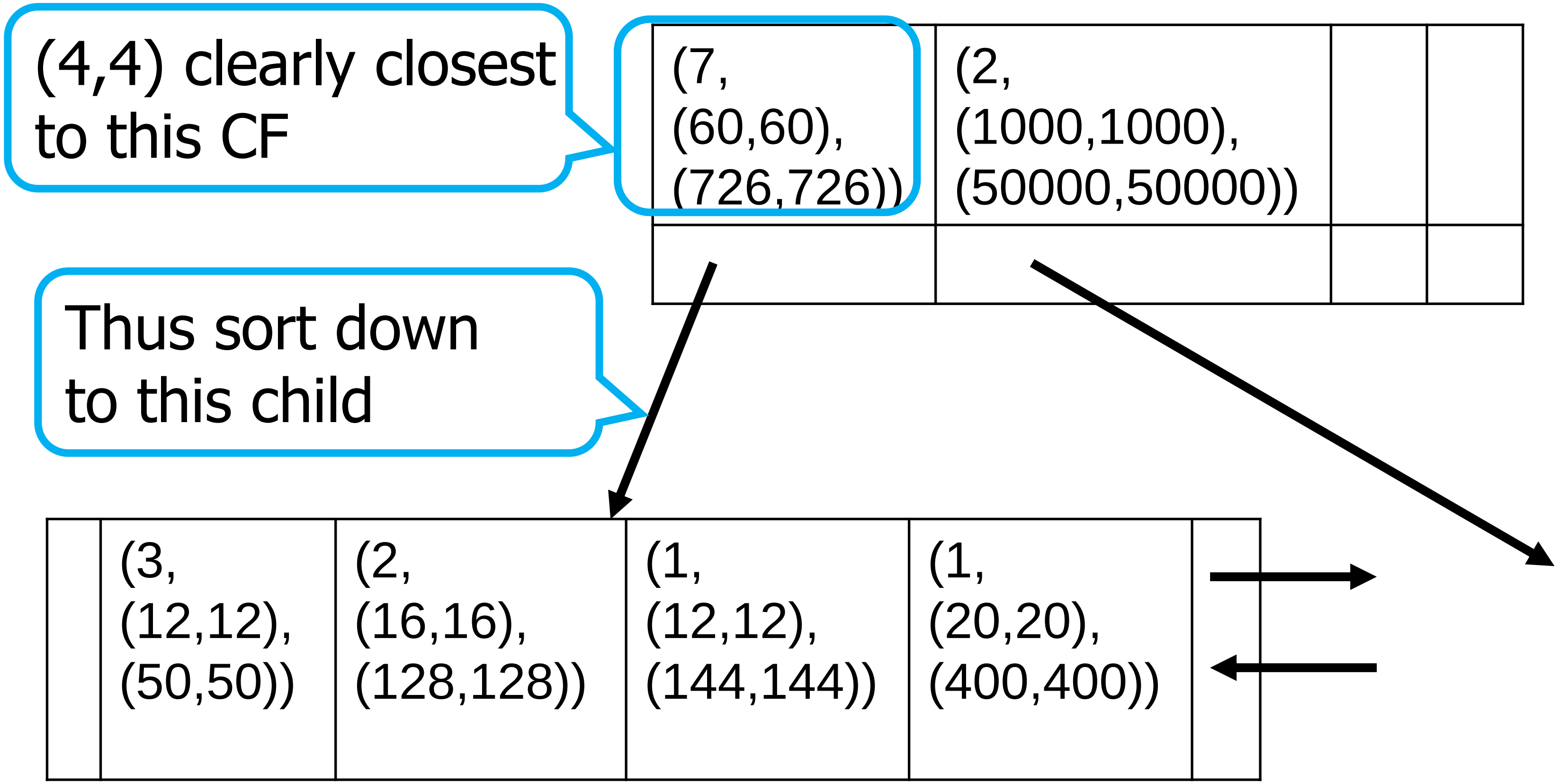
- The new subcluster may cause its parent to exceed branching factor
- If so, split leaf node
  - Identifying the pair of subclusters with largest inter-cluster distance
  - Divide by proximity to these two subclusters
- If this split clause non-leaf node to exceed branching factor, then recursively split
- If the root node is split, then the height of the CF tree is increased by one

Example: Insert (4,4) into CF Tree

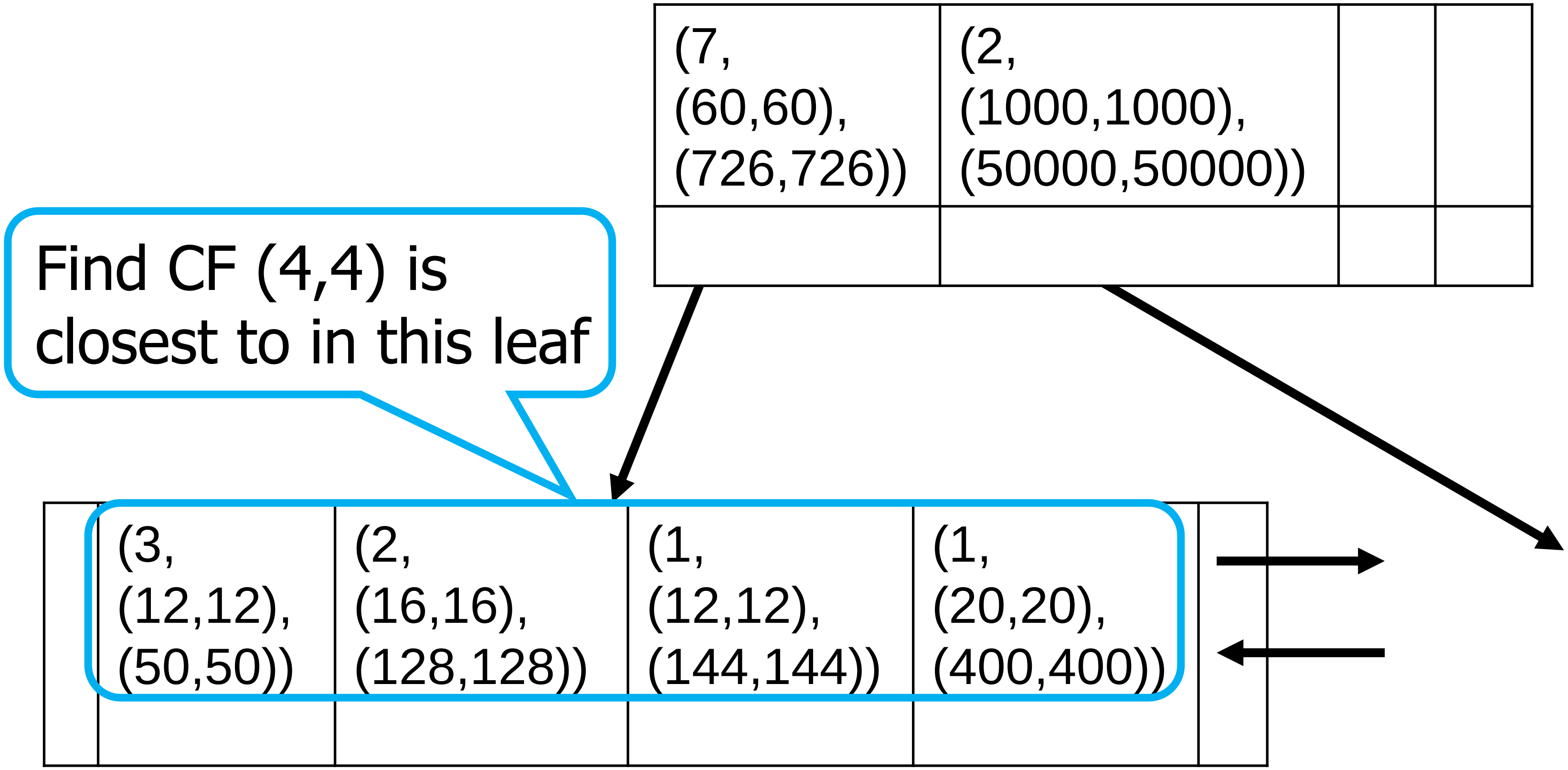
Assume
B=L=4
R = 2



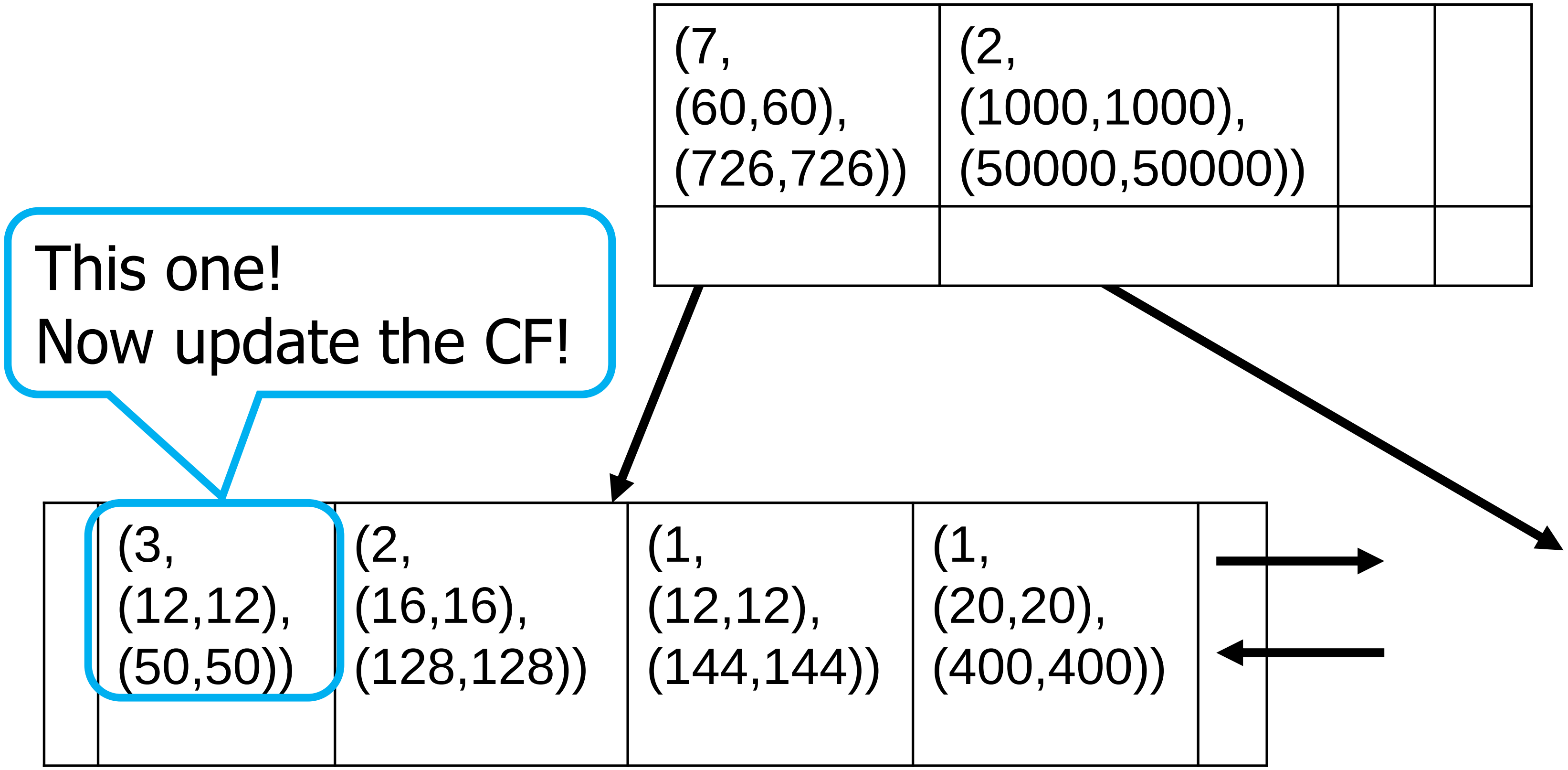
Example: Insert (4,4) into CF Tree



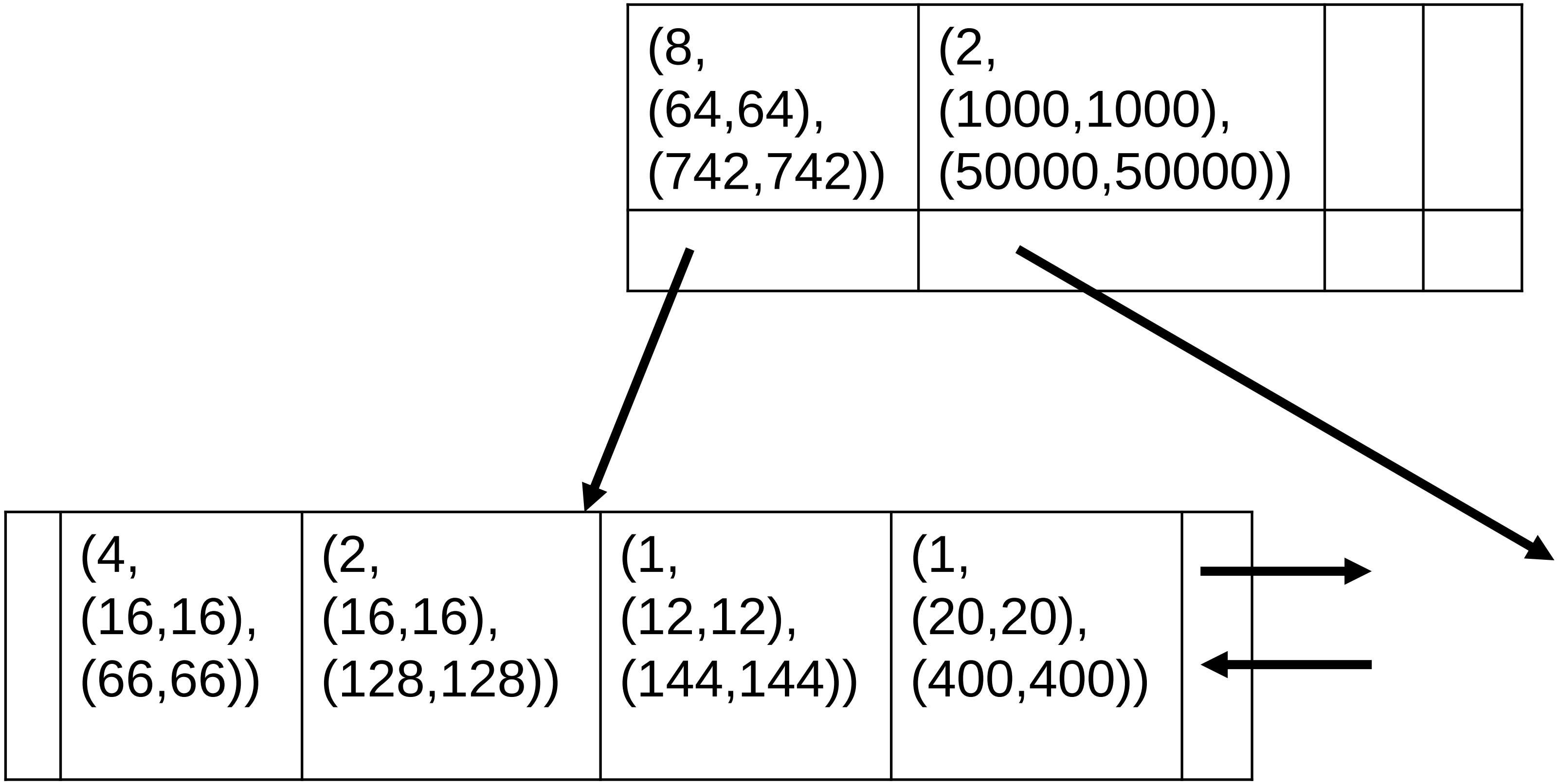
Example: Insert (4,4) into CF Tree



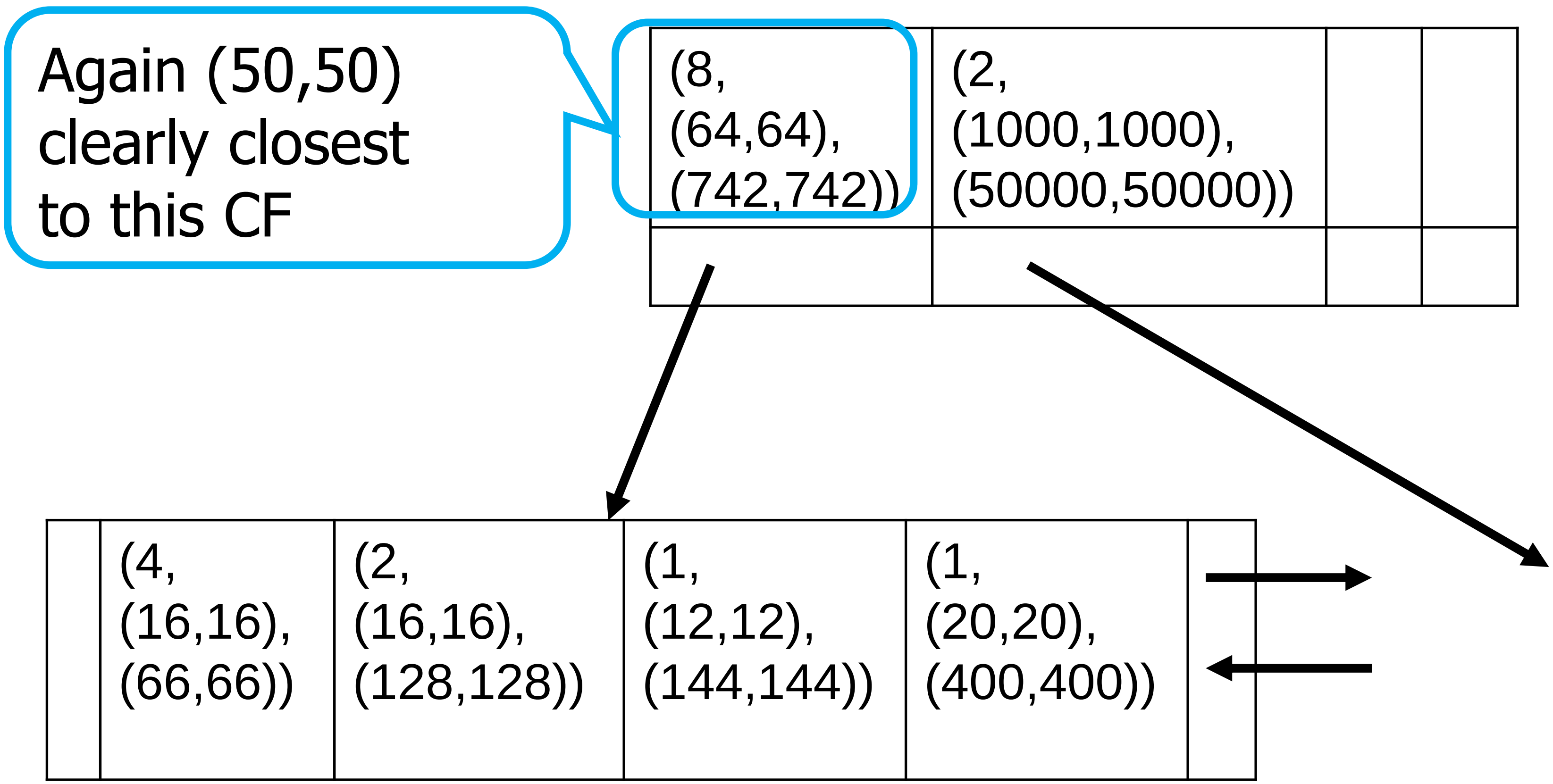
Example: Insert (4,4) into CF Tree



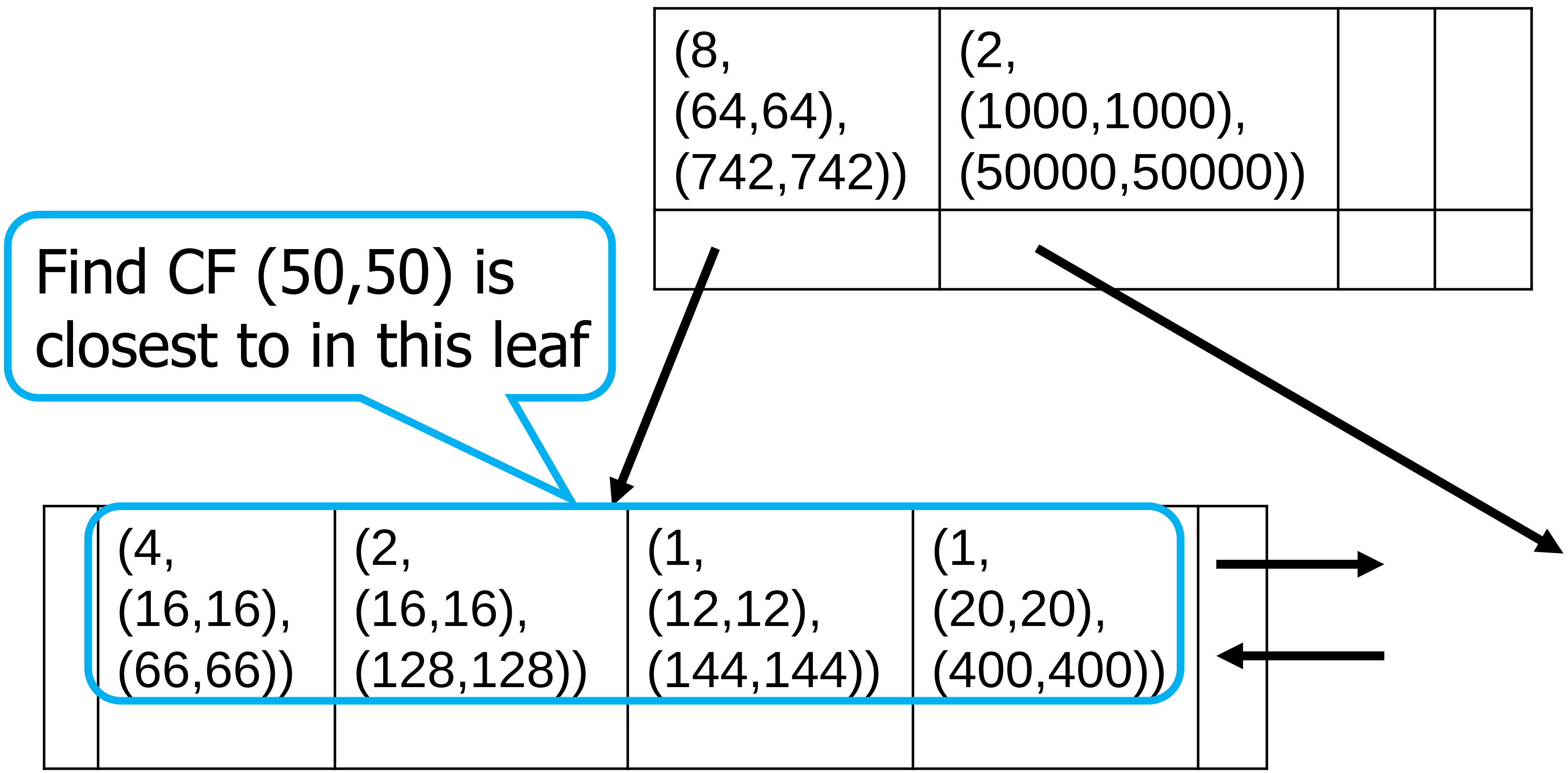
Example: Result After Inserting (4,4) into CF Tree



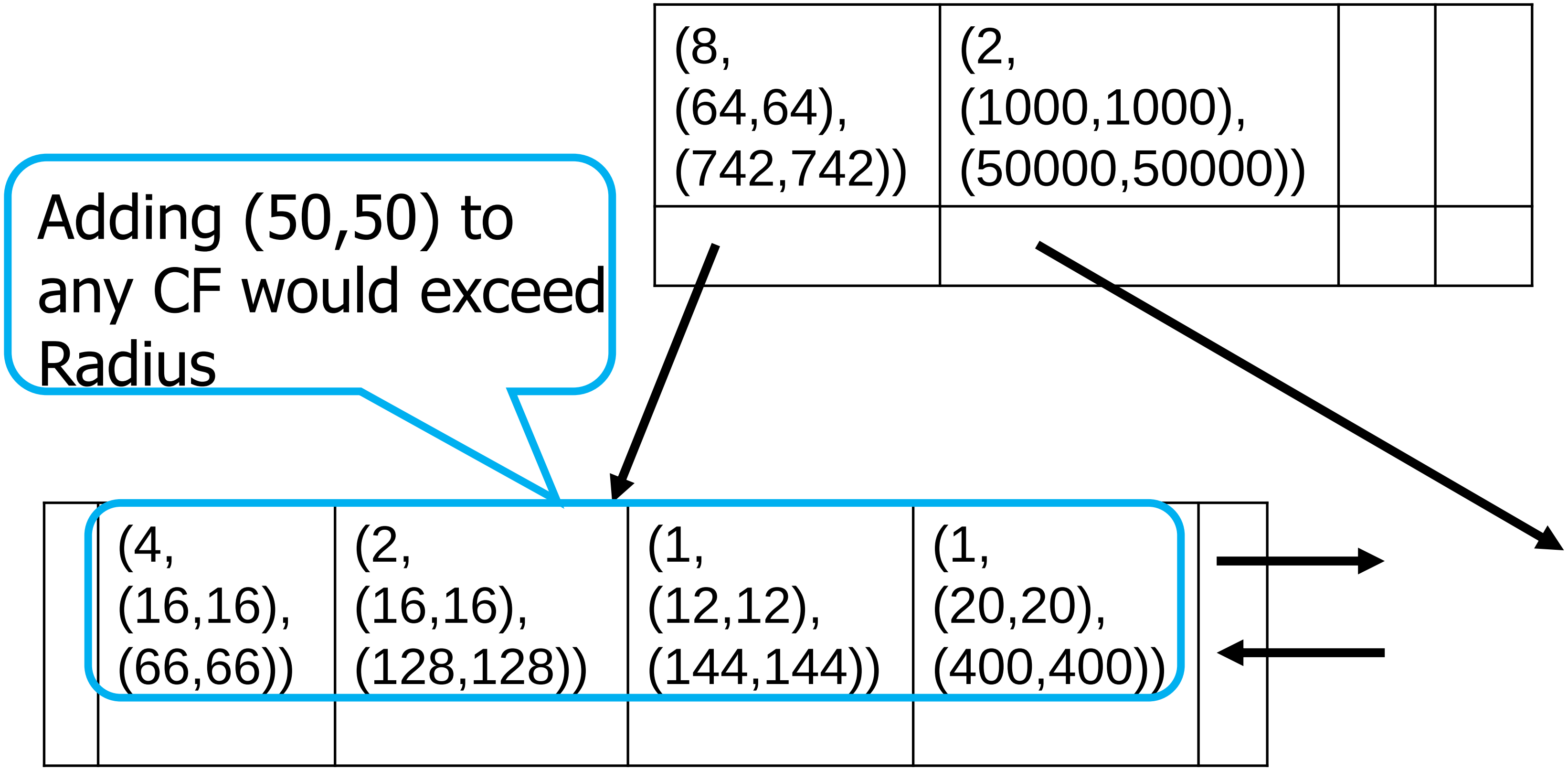
Now Insert (50,50) into CF Tree



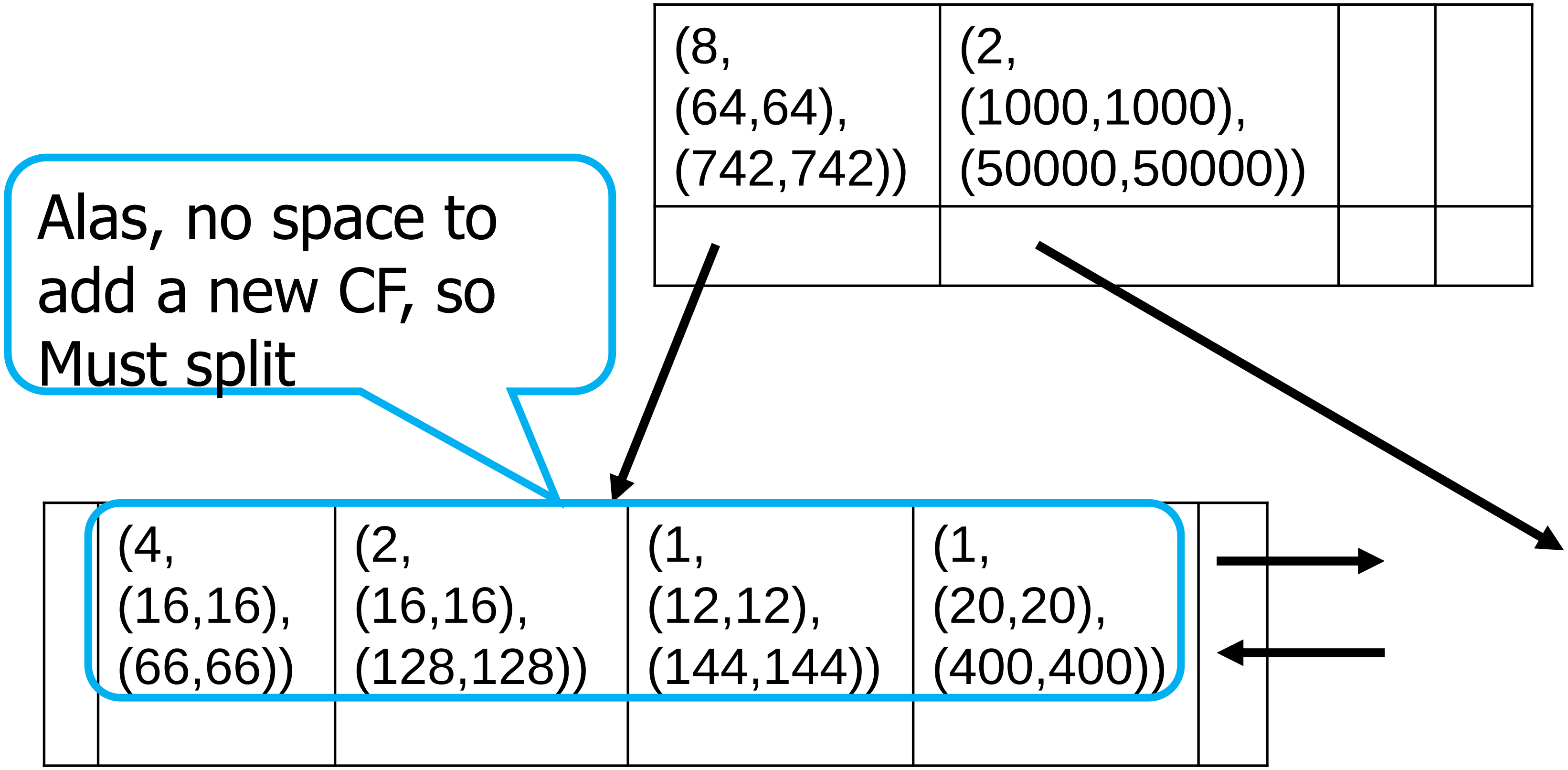
Now Insert (50,50) into CF Tree



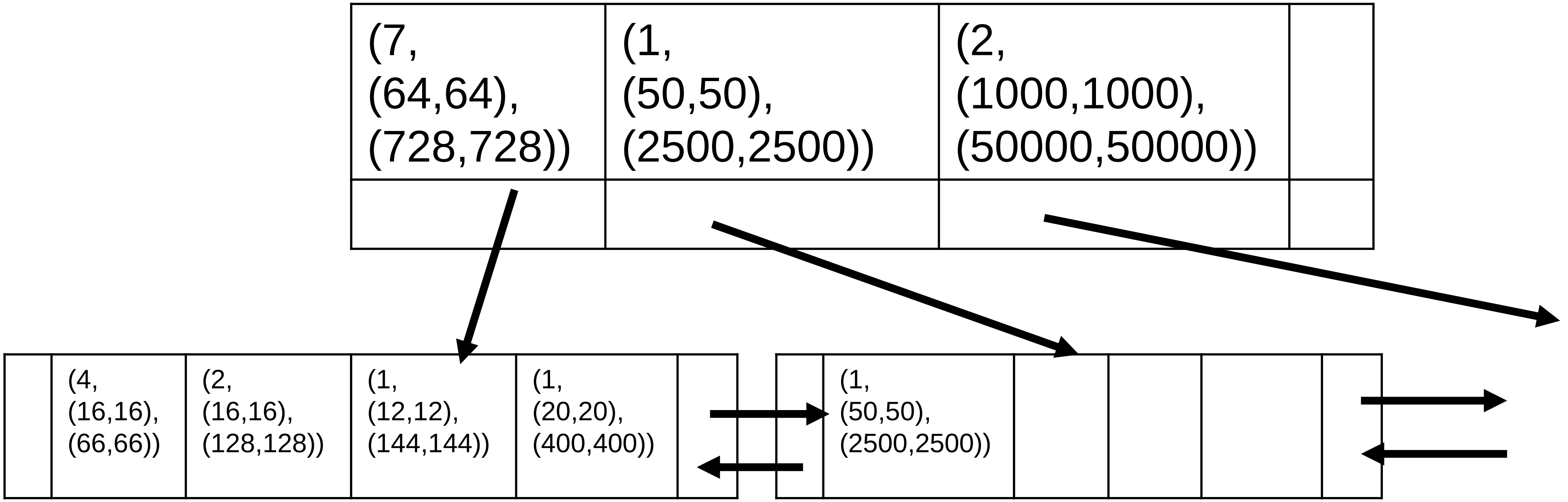
Now Insert (50,50) into CF Tree



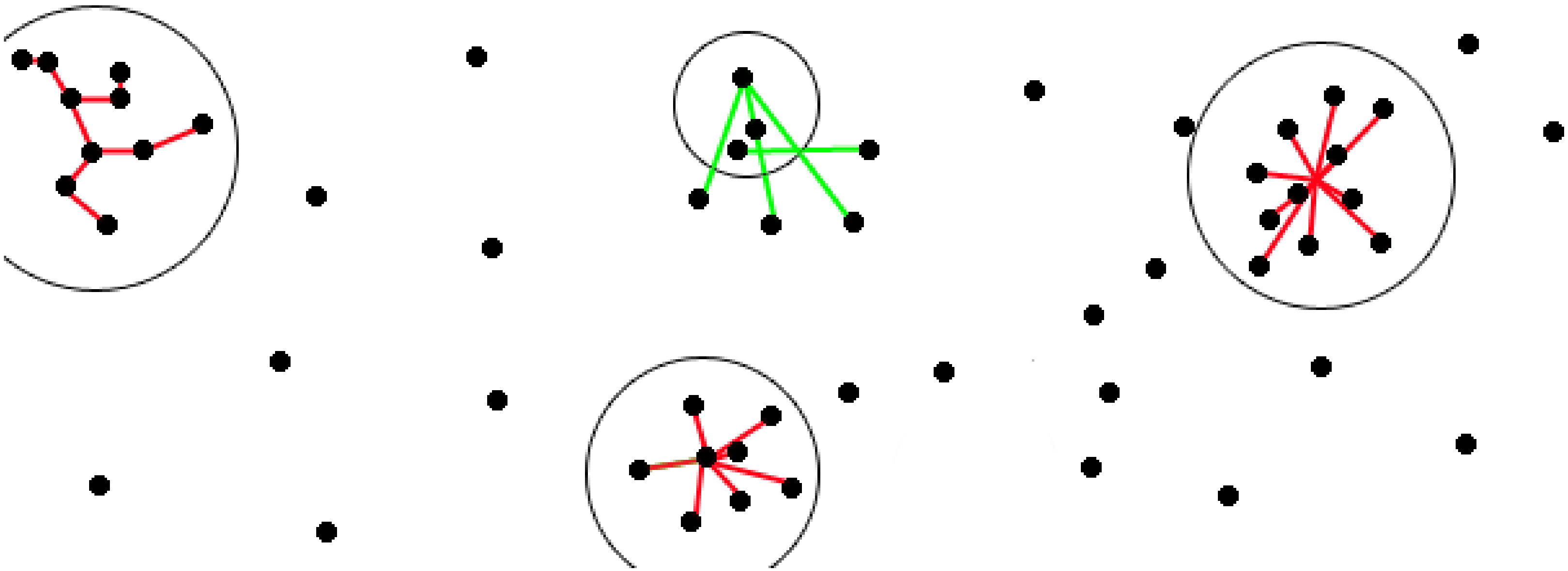
Now Insert (50,50) into CF Tree



After Inserting (50,50) into CF Tree



Traditional Algorithms

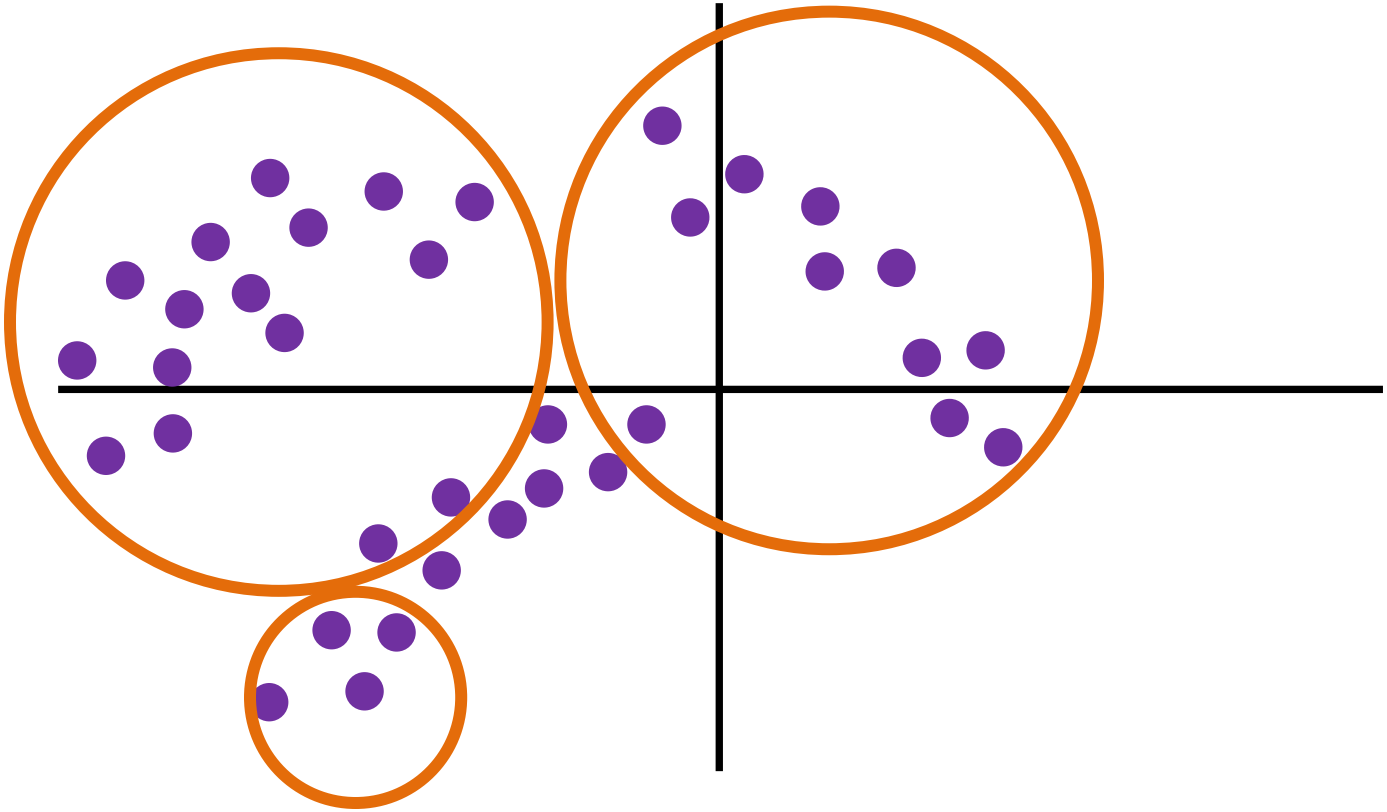


All-Points Based | Centroid Based

d_{min}, d_{max} | d_{avg}, d_{mean}

What Would BIRCH Do?

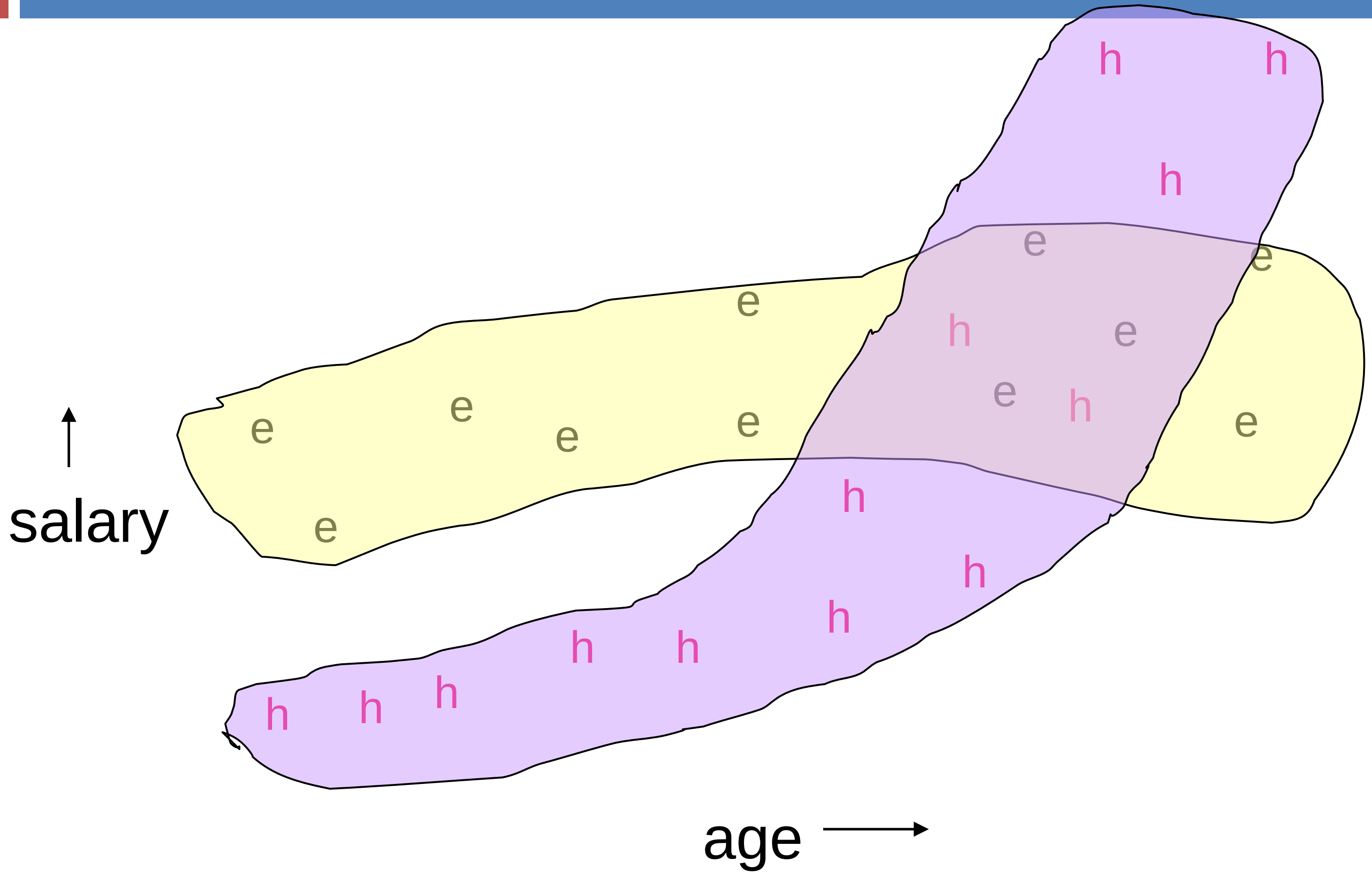
- **BIRCH assumes:**
  - Clusters are normally distributed in each dimension
  - Axes are fixed: Ellipses at an angle are ***not*** ***OK***



Clustering Using Representatives (CURE)

- Cluster definition: Set of representative points
  - Enables clusters of differing shapes
- Requires an Euclidean space
- Two-pass (hierarchical) clustering approach
  - Pass 1: Clustering of subset of data to pick “representative” points
  - Pass 2: Assign all points to clusters

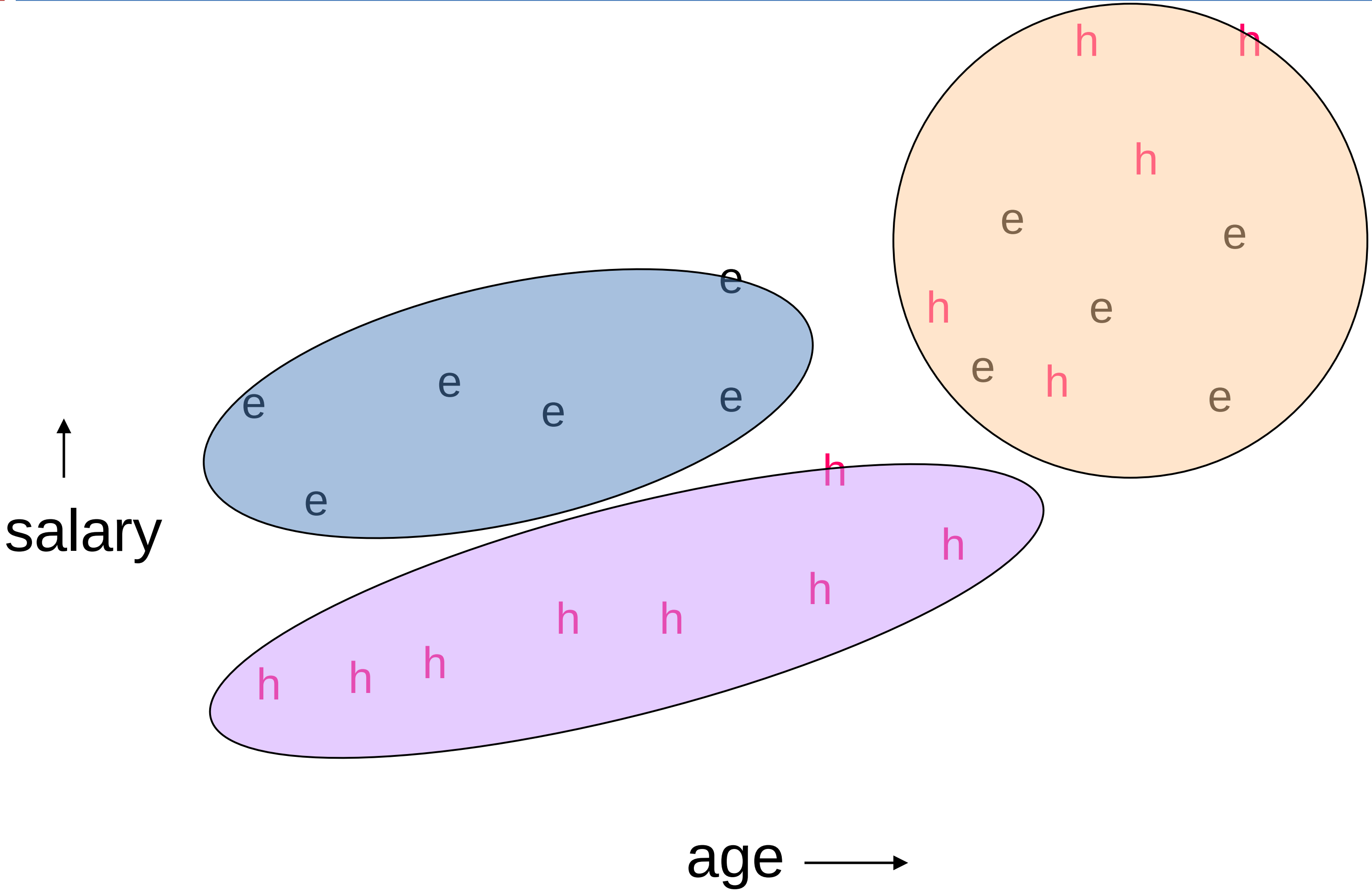
Example: Stanford Salaries



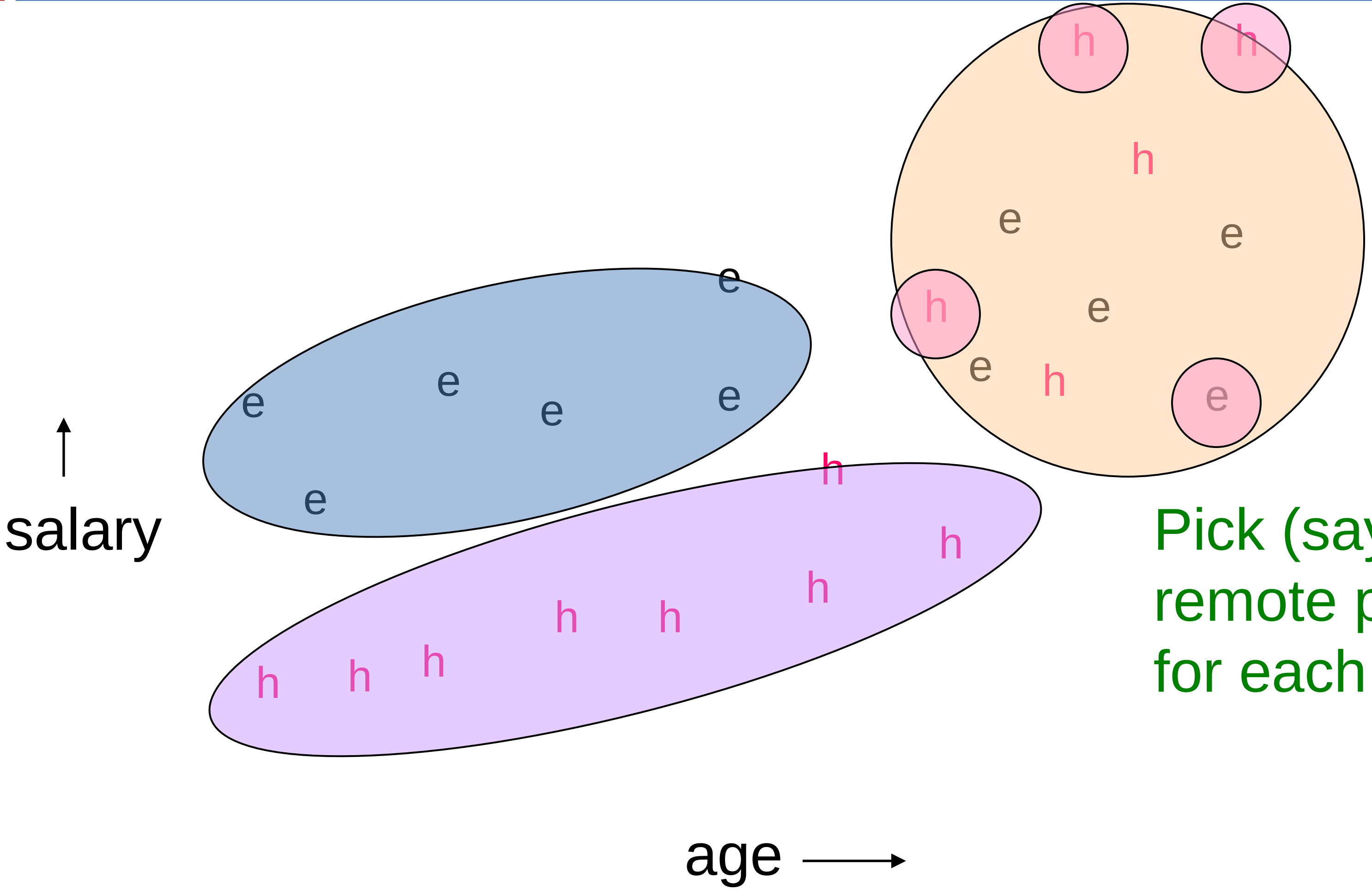
Pass 1

- Randomly sample of data that fits in memory
- Find initial clusters: Hierarchically cluster the data sample
- For each cluster, pick representative points
  - Select subset of points, as dispersed as possible to represent cluster
  - Move these points towards cluster center (e.g., shrink 20% towards mean)

Example: Initial Clusters

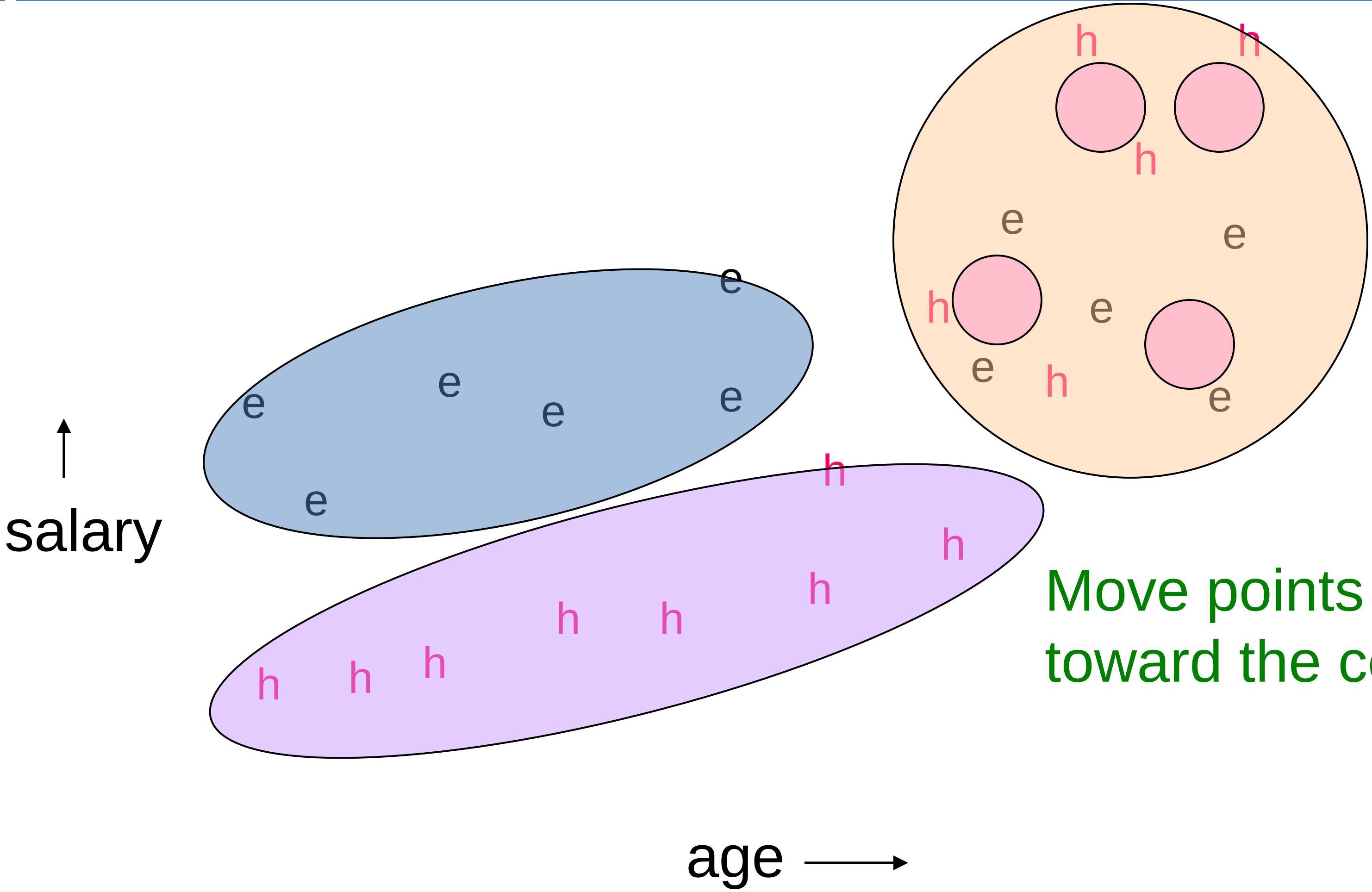


Example: Pick Dispersed Points



Pick (say) 4 remote points for each cluster.

Example: Pick Dispersed Points

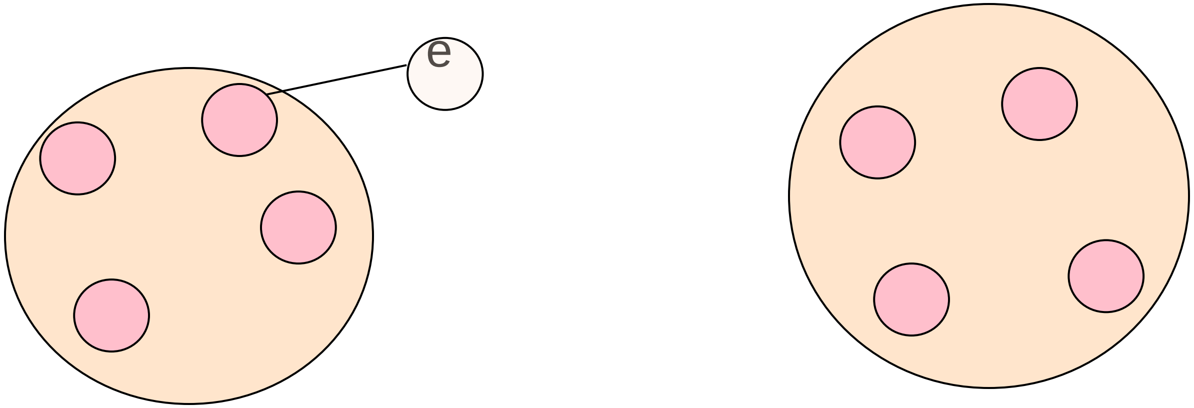


Move points (say) 20% toward the centroid.

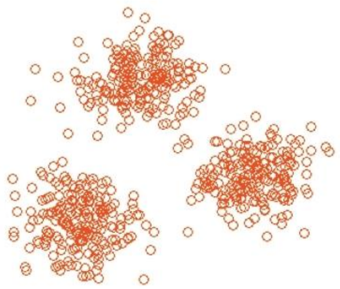
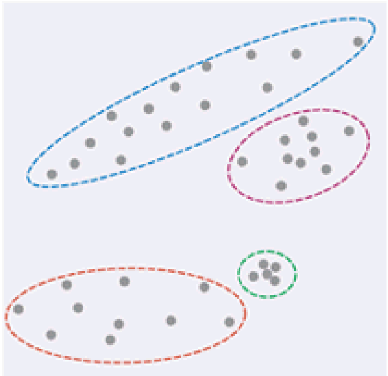
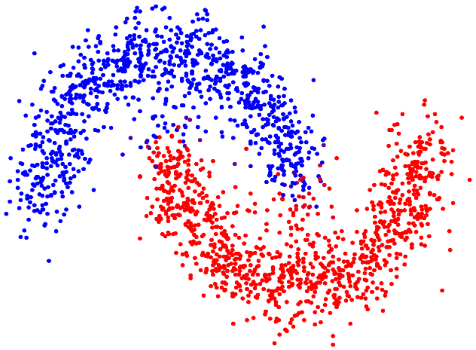
Pass 2

- Scan entire data set

- Assign each example e to "closest" cluster
  - Standard metric determines closest
  - Done by finding representative with smallest distance to e



BIRCH vs. CURE Summary

BIRCH		CURE
		
Fixed axes, normally distributed in each dimension	Rotated axes	Non-ellipsoid shape

Summary

- Hierarchical clustering models hierarchical structure among the examples
- Typically learned in a bottom-up manner
  - Linkage and distance are key parameters
  - Scalability is a key concern
- Advanced algorithms improve efficiency
- Shapes that can be represented are algorithm dependent

Questions?

- CURE slides from MMDS.org