How Did Yeast Become a Wine-Maker?

Clustering Algorithms

Phillip Compeau and Pavel Pevzner

Bioinformatics Algorithms: An Active Learning Approach

How Did Yeast Become a Wine Maker?

- Which Yeast Genes Are Responsible for Wine Brewing?
- Clustering as an optimization problem
- The Lloyd algorithm for *k*-means clustering
- From Hard to Soft Clustering
- From Coin Flipping to *k*-means Clustering
- Expectation Maximization
- Soft *k*-means Clustering
- Hierarchical Clustering

Which Domesticated Animal Is Next?

30,000 BC

10,000 BC

8,000 BC

8,000 BC

4,000 BC

Which Domesticated Animal Is Next?

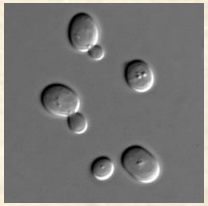
30,000 BC

10,000 BC

8,000 BC

8,000 BC

4,000 BC

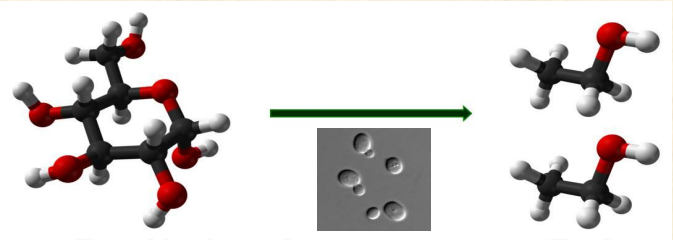


An Armenian cave, scientists discovered a 6,000 year-old winery

Why Do Winemakers Store Grapes in Sealed Barrels?

Yeast lives on grapevines and can convert glucose into **ethanol**.

glucose $C_6H_{12}O_6$



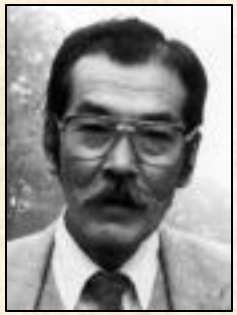
ethanol $2\ C_2H_5OH$

When the glucose runs out, yeast *inverts* its metabolism, and ethanol becomes its new food .

This change in metabolism (the **diauxic shift**) can only occur in the presence of oxygen.

Which genes are responsible for the diauxic shift?

How Did Yeast Invent the Diauxic Shift?



Susumu Ohno: two hypotheses with different fates.

Random Breakage Model: genomic architectures are shaped by rearrangements that occur randomly.

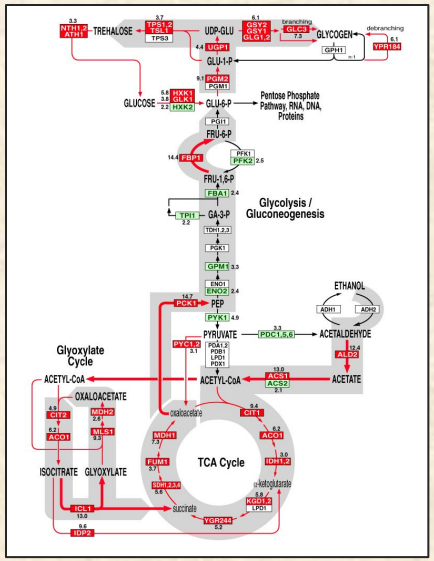
was embraced by biologists until it was refuted in 2003

Whole Genome Duplication Model: Big leaps in evolution would have been impossible without whole genome duplications.

first met with skepticism but proven 30 years later

What Does It Take to Convert Glucose into Ethanol and Back?

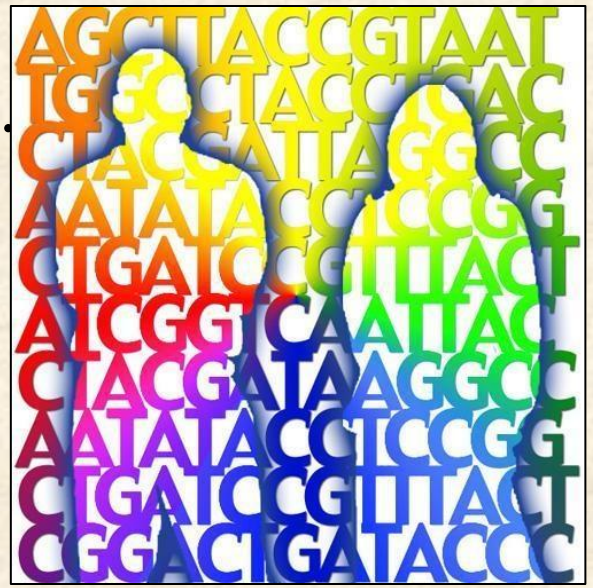
- Imagine the time when the first fruit-bearing plants evolved.
- The first species to metabolize glucose would have had an enormous evolutionary advantage.



- But metabolizing glucose — let alone ethanol — is not simple.
- It required creating new **metabolic pathways** with many genes working together.

Whole Genome Duplications Enables Evolutionary Breakthroughs

- Ohno argued that a WGD would provide a platform for such a revolutionary innovation, since every duplicated gene would have two copies.

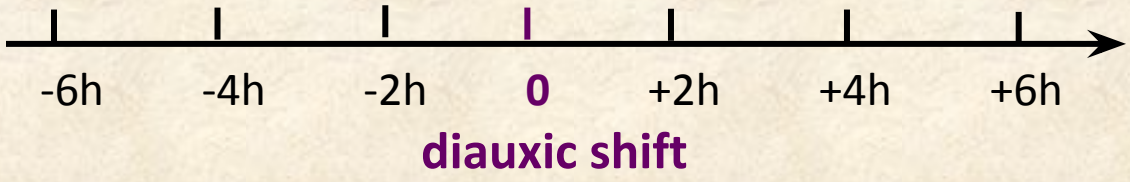


- One copy would be free to evolve without compromising the gene's existing function.

- Another copy would perform the existing function.

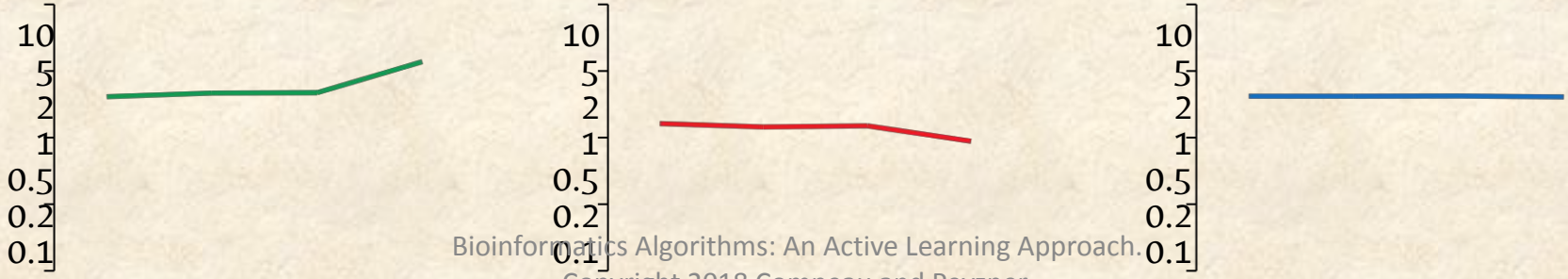
Measuring 3 Genes at 7 Checkpoints

Measure expression of various yeast genes at 7 checkpoints:



YLR258W	1.1	1.4	1.4	3.7	4.0	10.0	5.9
YPL012W	1.1	0.8	0.9	0.4	0.3	0.1	0.1
YPR055W	1.1	1.1	1.1	1.1	1.1	1.1	1.1

e_{ij} = **expression level** of gene i at checkpoint j



Switching to Logarithms of Expression Levels

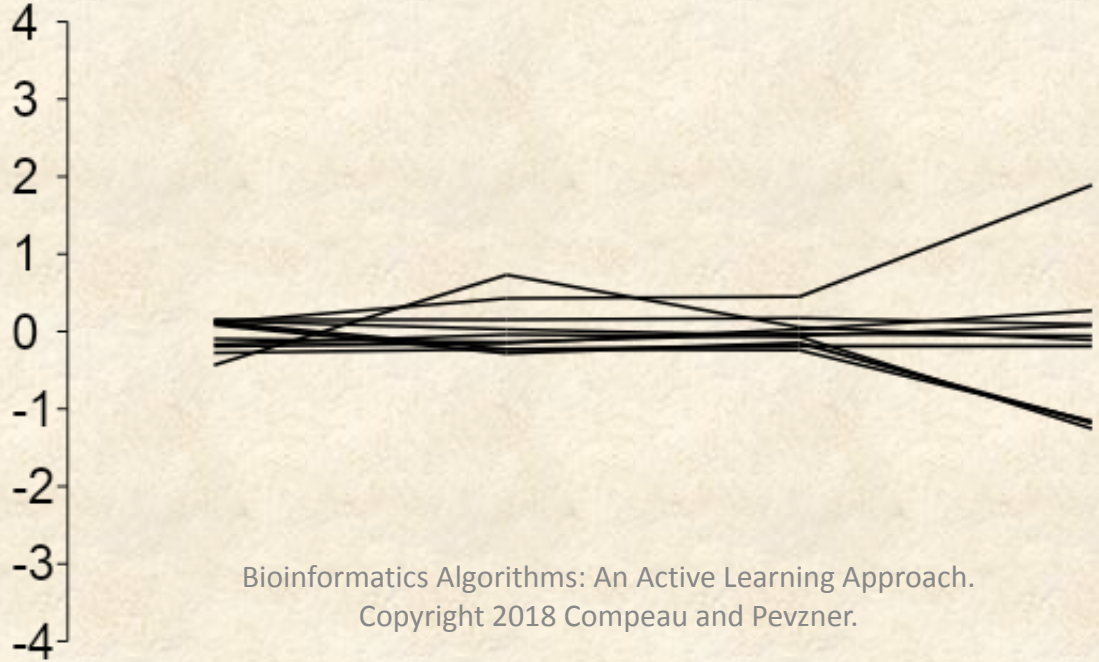
YLR258W	1.1	1.4	1.4	3.7	4.0	10.0	5.9
YPL012W	1.1	0.8	0.9	0.4	0.3	0.1	0.1
YPR055W	1.1	1.1	1.1	1.1	1.1	1.1	1.1

taking logarithms (base-2)

YLR258W	0.1	0.4	0.5	1.9	2.0	3.3	2.6
YPL012W	0.1	-0.3	-0.2	-1.2	-1.6	-3.0	-3.1
YPR055W	0.2	0.2	0.2	0.1	0.1	0.1	0.1

Gene Expression Matrix

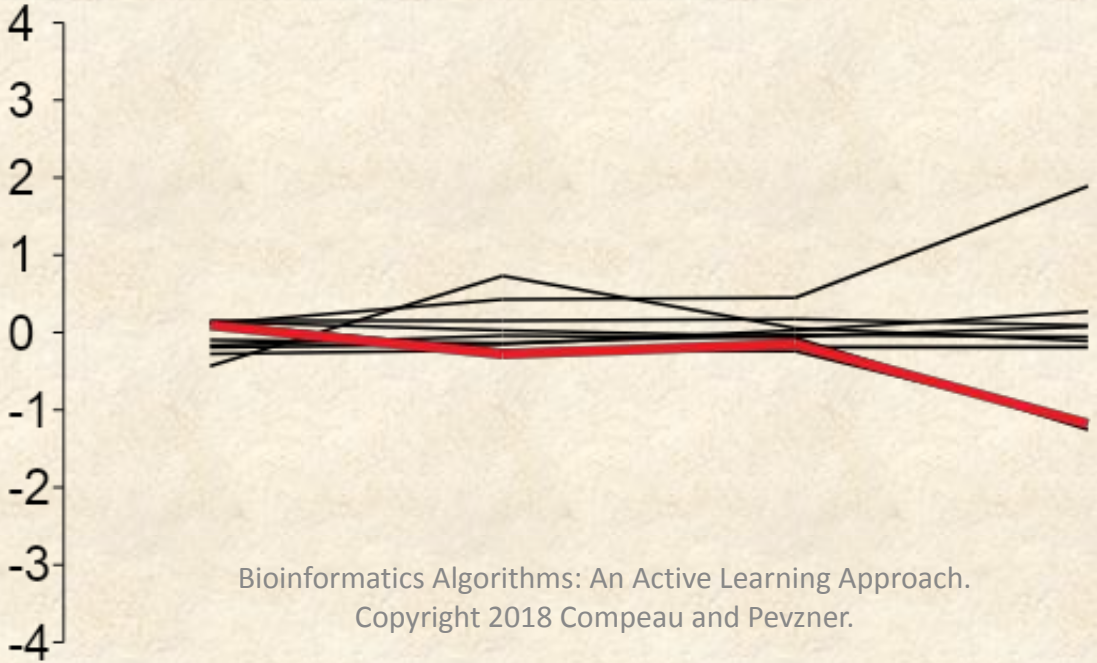
YLR361C	0.14	0.03	-0.06	0.07	-0.01	-0.06	-0.01
YMR290C	0.12	-0.23	-0.24	-1.16	-1.40	-2.67	-3.00
YNR065C	-0.10	-0.14	-0.03	-0.06	-0.07	-0.14	-0.04
YGR043C	-0.43	-0.73	-0.06	-0.11	-0.16	3.47	2.64
YLR258W	0.11	0.43	0.45	1.89	2.00	3.32	2.56
YPL012W	0.09	-0.28	-0.15	-1.18	-1.59	-2.96	-3.08
YNL141W	-0.16	-0.04	-0.07	-1.26	-1.20	-2.82	-3.13
YJL028W	-0.28	-0.23	-0.19	-0.19	-0.32	-0.18	-0.18
YKL026C	-0.19	-0.15	0.03	0.27	0.54	3.64	2.74
YPR055W	0.15	0.15	0.17	0.09	0.07	0.09	0.07



Gene Expression Matrix

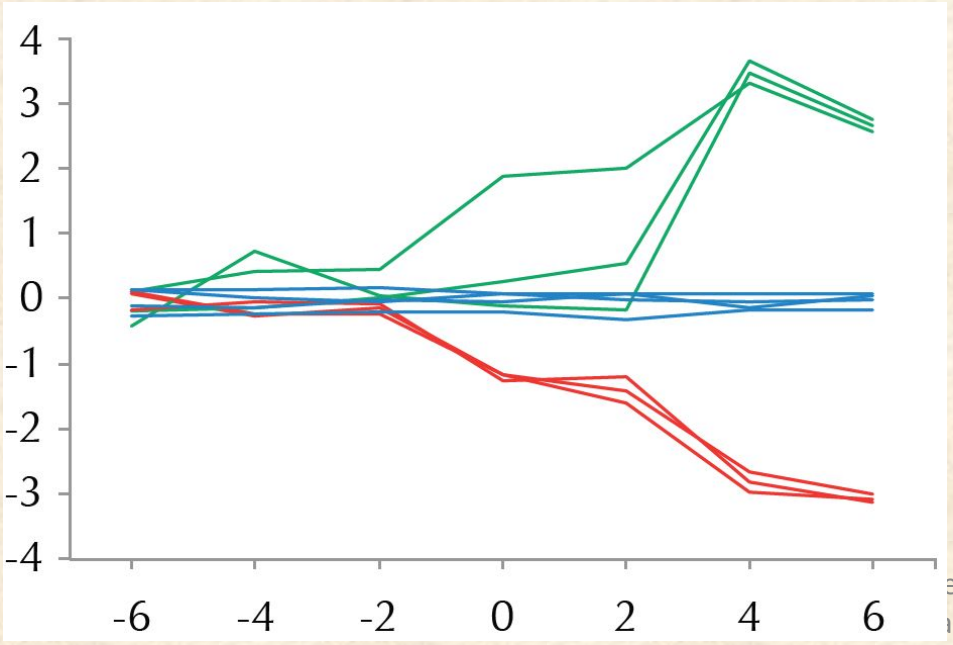
YLR361C	0.14	0.03	-0.06	0.07	-0.01	-0.06	-0.01
YMR290C	0.12	-0.23	-0.24	-1.16	-1.40	-2.67	-3.00
YNR065C	-0.10	-0.14	-0.03	-0.06	-0.07	-0.14	-0.04
YGR043C	-0.43	-0.73	-0.06	-0.11	-0.16	3.47	2.64
YLR258W	0.11	0.43	0.45	1.89	2.00	3.32	2.56
YPL012W	0.09	-0.28	-0.15	-1.18	-1.59	-2.96	-3.08
YNL141W	-0.16	-0.04	-0.07	-1.26	-1.20	-2.82	-3.13
YJL028W	-0.28	-0.23	-0.19	-0.19	-0.32	-0.18	-0.18
YKL026C	-0.19	-0.15	0.03	0.27	0.54	3.64	2.74
YPR055W	0.15	0.15	0.17	0.09	0.07	0.09	0.07

gene expression vector



Gene Expression Matrix

YLR361C	0.14	0.03	-0.06	0.07	-0.01	-0.06	-0.01
YMR290C	0.12	-0.23	-0.24	-1.16	-1.40	-2.67	-3.00
YNR065C	-0.10	-0.14	-0.03	-0.06	-0.07	-0.14	-0.04
YGR043C	-0.43	-0.73	-0.06	-0.11	-0.16	3.47	2.64
YLR258W	0.11	0.43	0.45	1.89	2.00	3.32	2.56
YPL012W	0.09	-0.28	-0.15	-1.18	-1.59	-2.96	-3.08
YNL141W	-0.16	-0.04	-0.07	-1.26	-1.20	-2.82	-3.13
YJL028W	-0.28	-0.23	-0.19	-0.19	-0.32	-0.18	-0.18
YKL026C	-0.19	-0.15	0.03	0.27	0.54	3.64	2.74
YPR055W	0.15	0.15	0.17	0.09	0.07	0.09	0.07



e Learning Approach.
nd Pevzner.

Gene Expression Matrix

YLR361C	0.14	0.03	-0.06	0.07	-0.01	-0.06	-0.01
YMR290C	0.12	-0.23	-0.24	-1.16	-1.40	-2.67	-3.00
YNR065C	-0.10	-0.14	-0.03	-0.06	-0.07	-0.14	-0.04
YGR043C	-0.43	-0.73	-0.06	-0.11	-0.16	3.47	2.64
YLR258W	0.11	0.43	0.45	1.89	2.00	3.32	2.56
YPL012W	0.09	-0.28	-0.15	-1.18	-1.59	-2.96	-3.08
YNL141W	-0.16	-0.04	-0.07	-1.26	-1.20	-2.82	-3.13
YJL028W	-0.28	-0.23	-0.19	-0.19	-0.32	-0.18	-0.18
YKL026C	-0.19	-0.15	0.03	0.27	0.54	3.64	2.74
YPR055W	0.15	0.15	0.17	0.09	0.07	0.09	0.07

1997: Joseph deRisi measured expression of 6,400 yeast genes at 7 checkpoints before and after the diauxic shift.

6,400 x 7 gene expression matrix

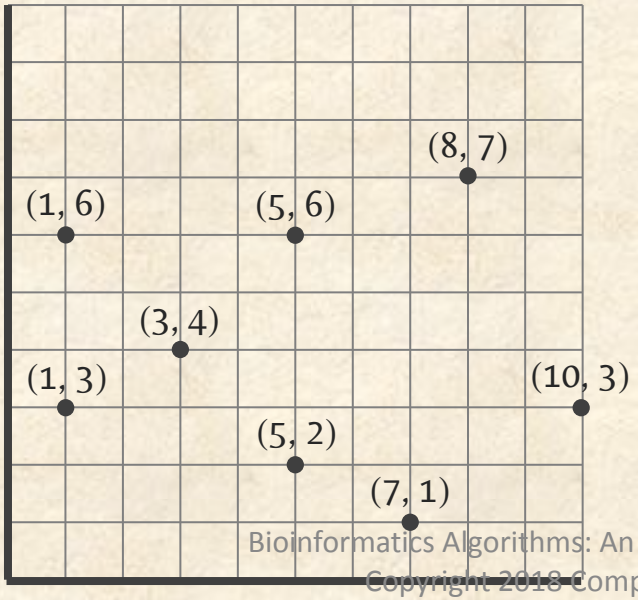
Goal: partition all yeast genes into clusters so that:

- genes in the *same* cluster have similar behavior
- genes in *different* clusters have different behavior

Genes as Points in Multidimensional Space

YLR361C	0.14	0.03	-0.06	0.07	-0.01	-0.06	-0.01
YMR290C	0.12	-0.23	-0.24	-1.16	-1.40	-2.67	-3.00
YNR065C	-0.10	-0.14	-0.03	-0.06	-0.07	-0.14	-0.04
YGR043C	-0.43	-0.73	-0.06	-0.11	-0.16	3.47	2.64
YLR258W	0.11	0.43	0.45	1.89	2.00	3.32	2.56
YPL012W	0.09	-0.28	-0.15	-1.18	-1.59	-2.96	-3.08
YNL141W	-0.16	-0.04	-0.07	-1.26	-1.20	-2.82	-3.13
YJL028W	-0.28	-0.23	-0.19	-0.19	-0.32	-0.18	-0.18
YKL026C	-0.19	-0.15	0.03	0.27	0.54	3.64	2.74
YPR055W	0.15	0.15	0.17	0.09	0.07	0.09	0.07

n x *m*
gene expression
matrix



n points in
m-dimensional space

How Did Yeast Become a Wine Maker?

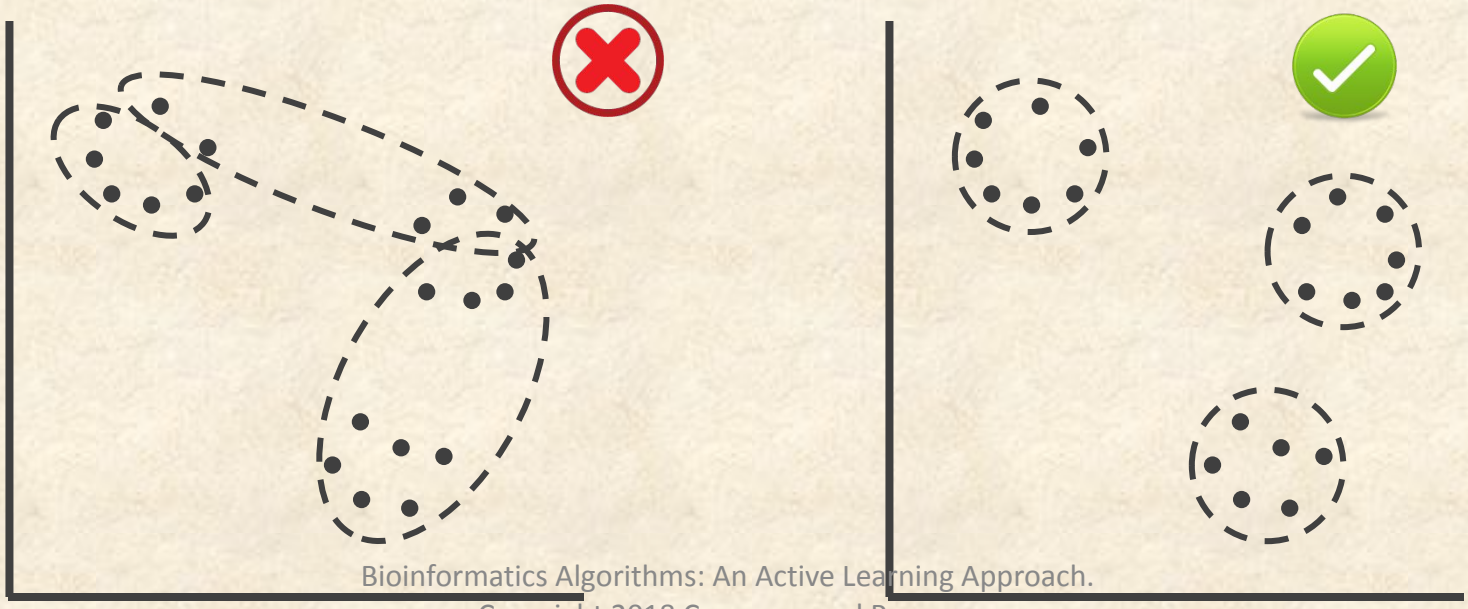
- Which Yeast Genes Are Responsible for Wine Brewing?
- Clustering as an optimization problem
- The Lloyd algorithm for *k*-means clustering
- From Hard to Soft Clustering
- From Coin Flipping to *k*-means Clustering
- Expectation Maximization
- Soft *k*-means Clustering
- Hierarchical Clustering

Toward a Computational Problem

Good Clustering Principle: Elements within the same cluster are closer to each other than elements in different clusters.

Toward a Computational Problem

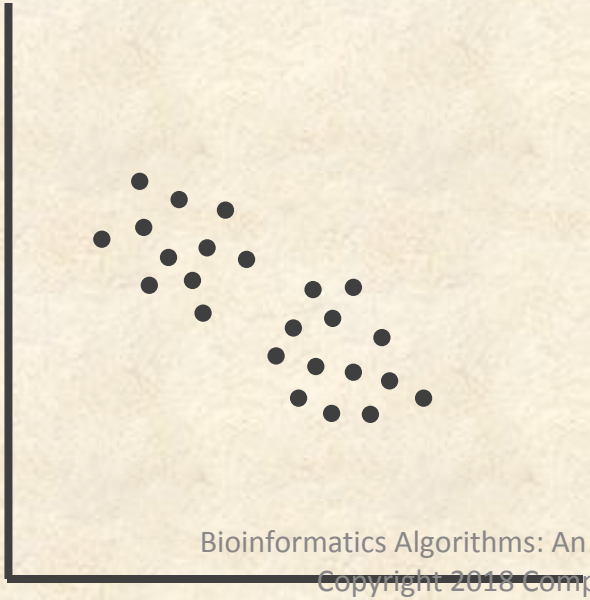
- distance between elements in the same cluster < Δ
- distance between elements in different clusters > Δ



Clustering Problem

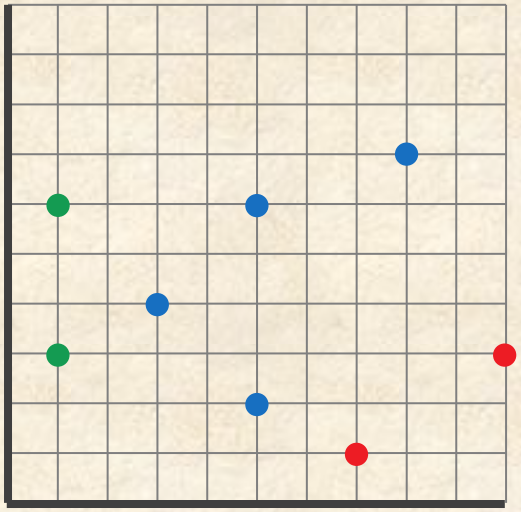
Clustering Problem: *Partition a set of expression vectors into clusters.*

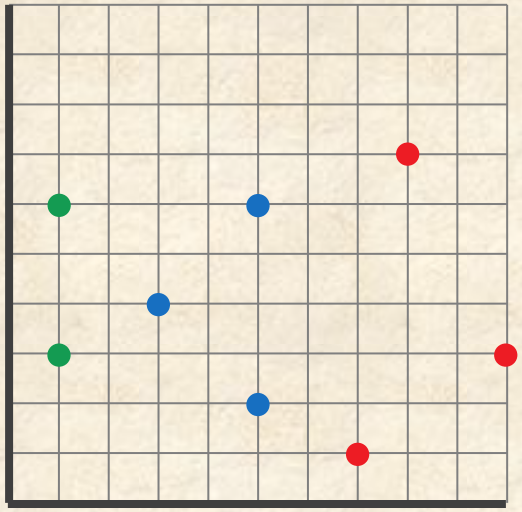
- **Input**: A collection of *n* vectors and an integer *k*.
- **Output**: Partition of *n* vectors into *k* disjoint clusters satisfying the Good Clustering Principle.

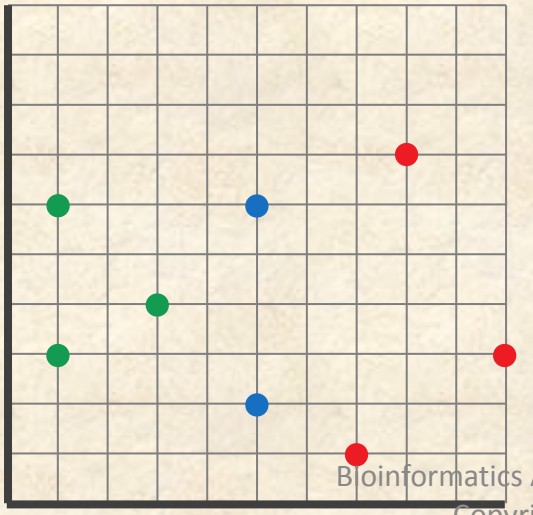


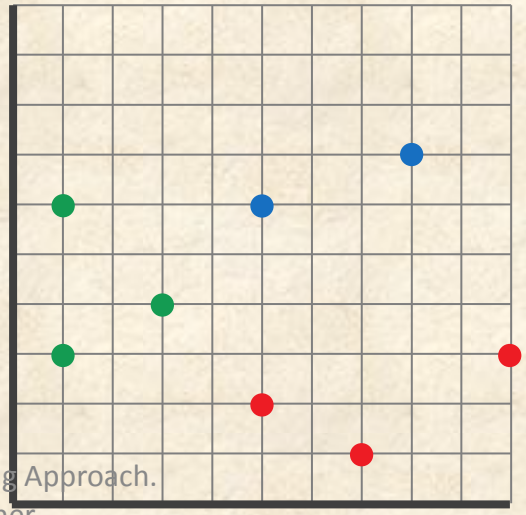
Any partition into two clusters **does not** satisfy the Good Clustering Principle!

STOP and Think: What is the "best" partition into three clusters?









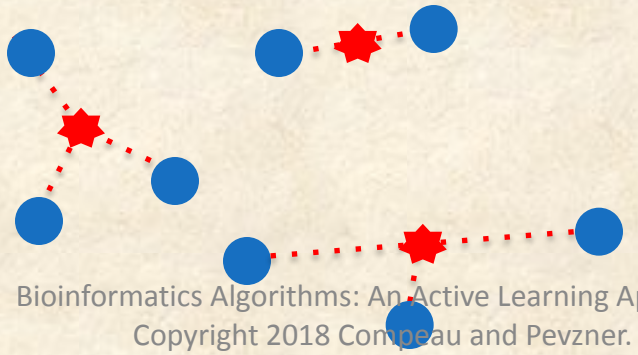
Clustering as Finding Centers

Goal: partition a set *Data* into *k* clusters.

Clustering as Finding Centers

Goal: partition a set *Data* into *k* clusters.

Equivalent goal: find a set of *k* points *Centers* that will serve as the "centers" of the *k* clusters in *Data*.

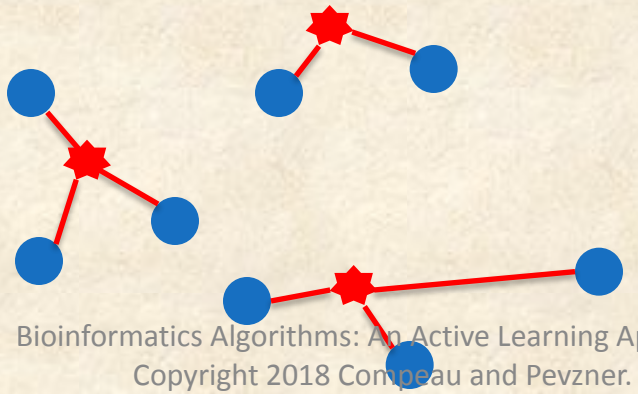


Clustering as Finding Centers

Goal: partition a set *Data* into *k* clusters.

Equivalent goal: find a set of *k* points *Centers* that will serve as the "centers" of the *k* clusters in *Data* and will minimize some notion of distance from *Centers* to *Data* .

What is the "distance" from *Centers* to *Data*?



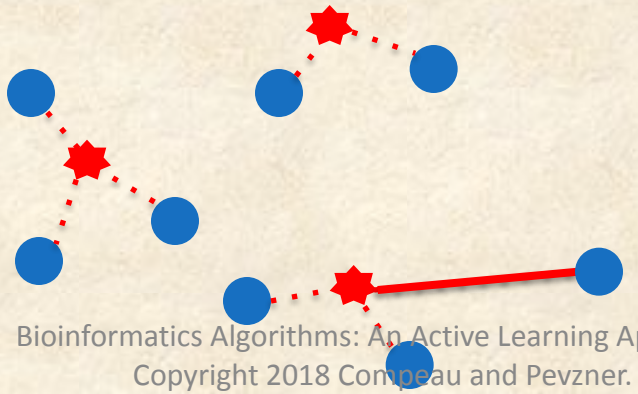
Distance from a *Single* *DataPoint* to *Centers*

The distance from *DataPoint* in *Data* to *Centers* is the distance from *DataPoint* to the closest center:

$$d(DataPoint, Centers) = \min_{\text{all points } x \text{ from } Centers} d(DataPoint, x)$$

Distance from *Data* to *Centers*

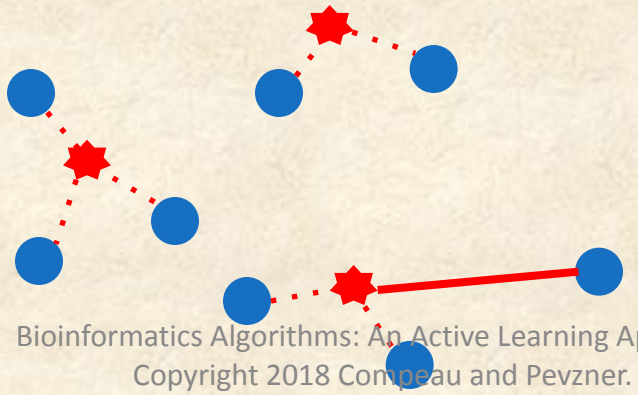
$$MaxDistance(Data, Centers) = \max_{\text{all points } DataPoint \text{ from } Data} d(DataPoint, Centers)$$



k-Center Clustering Problem

***k*-Center Clustering Problem.** Given a set of points *Data*, find *k* centers minimizing *MaxDistance*(*Data*, *Centers*).

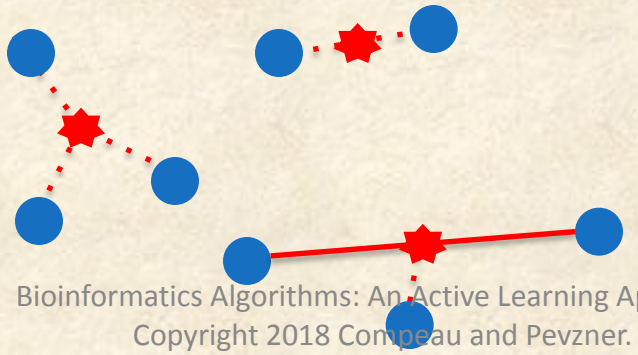
- **Input:** A set of points *Data* and an integer *k*.
- **Output:** A set of *k* points *Centers* that minimizes *MaxDistance*(*DataPoints*, *Centers*) over all possible choices of *Centers*.



k-Center Clustering Problem

***k*-Center Clustering Problem.** Given a set of points *Data*, find *k* centers minimizing *MaxDistance*(*Data*, *Centers*).

- **Input:** A set of points *Data* and an integer *k*.
- **Output:** A set of *k* points *Centers* that minimizes *MaxDistance*(*DataPoints*, *Centers*) over all possible choices of *Centers*.



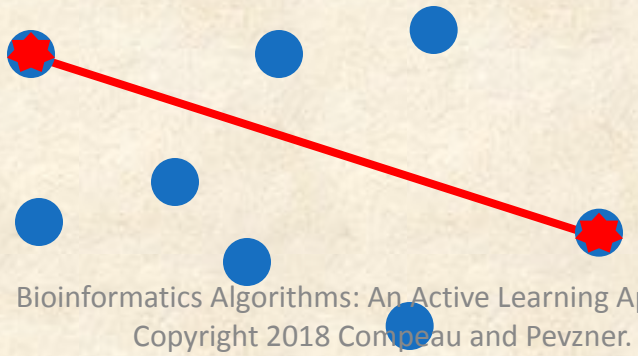
k-Center Clustering Problem

***k*-Center Clustering Problem.** Given a set of points *Data*, find *k* centers minimizing *MaxDistance*(*Data*, *Centers*).

- **Input:** A set of points *Data* and an integer *k*.
- **Output:** A set of *k* points *Centers* that minimizes *MaxDistance*(*DataPoints*, *Centers*) over all possible choices of *Centers*.

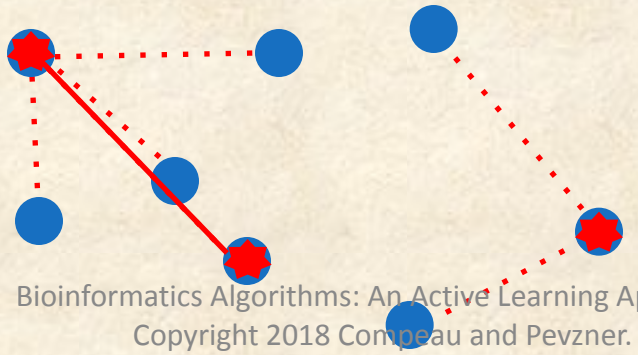
k-Center Clustering Heuristic

FarthestFirstTraversal(*Data*, *k*)
  Centers ← the set consisting of a single *DataPoint* from *Data*
  while *Centers* have fewer than *k* points
    DataPoint ← a point in *Data* maximizing *d*(*DataPoint*, *Centers*)
      among all data points
    add *DataPoint* to *Centers*



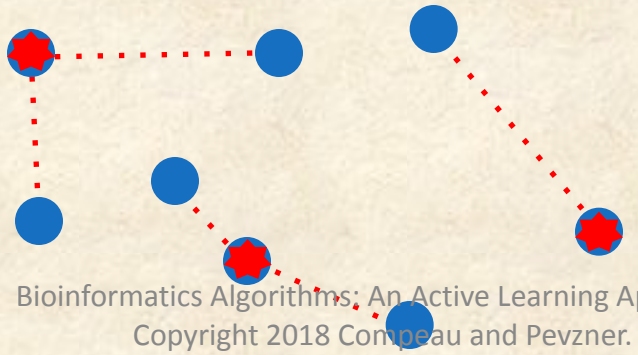
k-Center Clustering Heuristic

FarthestFirstTraversal(*Data*, *k*)
  Centers ← the set consisting of a single *DataPoint* from *Data*
  while *Centers* have fewer than *k* points
    DataPoint ← a point in *Data* maximizing *d*(*DataPoint*, *Centers*)
      among all data points
    add *DataPoint* to *Centers*



k-Center Clustering Heuristic

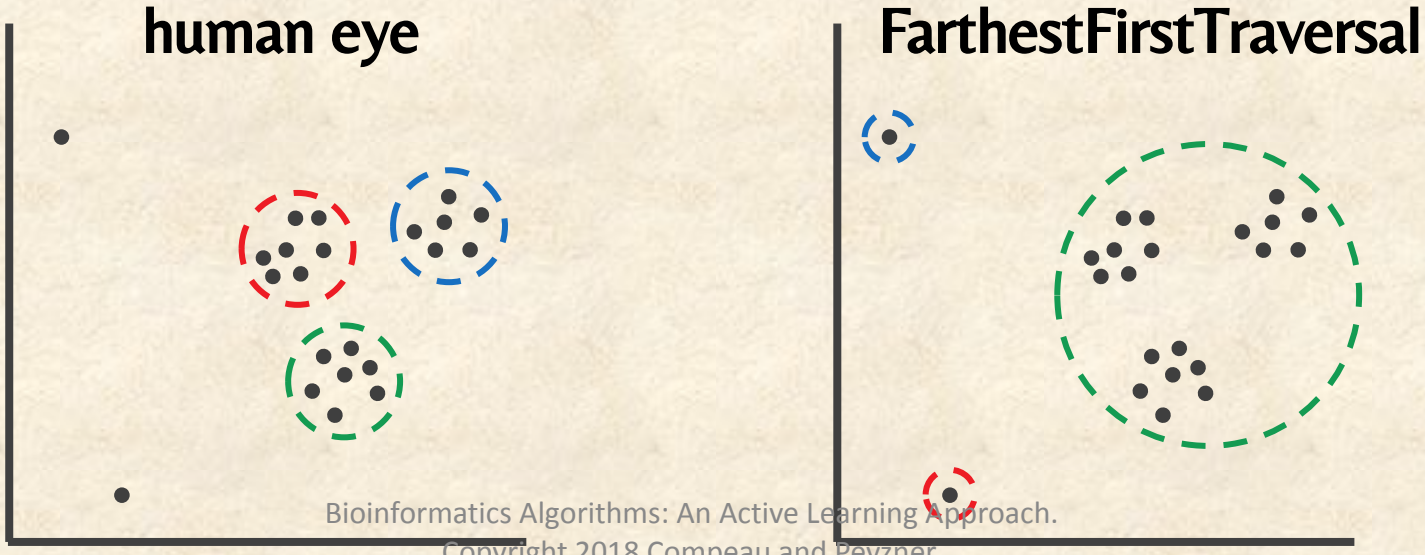
FarthestFirstTraversal(*Data*, *k*)
  Centers ← the set consisting of a single *DataPoint* from *Data*
  while *Centers* have fewer than *k* points
    DataPoint ← a point in *Data* maximizing *d*(*DataPoint*, *Centers*)
      among all data points
    add *DataPoint* to *Centers*



What Is Wrong with **FarthestFirstTraversal**?

FarthestFirstTraversal selects *Centers* that minimize *MaxDistance*(*Data*, *Centers*).

But biologists are interested in **typical** rather than **maximum** deviations, since maximum deviations may represent **outliers** (experimental errors).



Modifying the Objective Function

The **maximal distance** between *Data* and *Centers*:

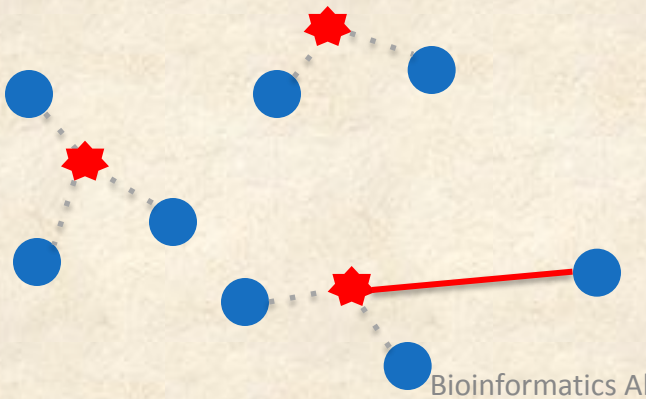
$$MaxDistance(Data, Centers) = \max_{DataPoint \text{ from } Data} d(DataPoint, Centers)$$

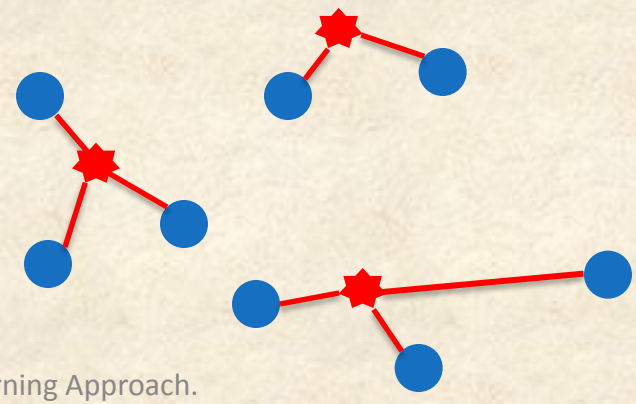
The **squared error distortion** between *Data* and *Centers*:

$$Distortion(Data, Centers) = \sum_{DataPoint \text{ from } Data} d(DataPoint, Centers)^2/n$$

A single data point contributes to *MaxDistance*

All data points contribute to *Distortion*





k-Means Clustering Problem

***k*-Center Clustering Problem:**

Input: A set of points *Data* and an integer *k*.

Output: A set of *k* points *Centers* that minimizes

MaxDistance(*DataPoints*,*Centers*)

over all choices of *Centers*.

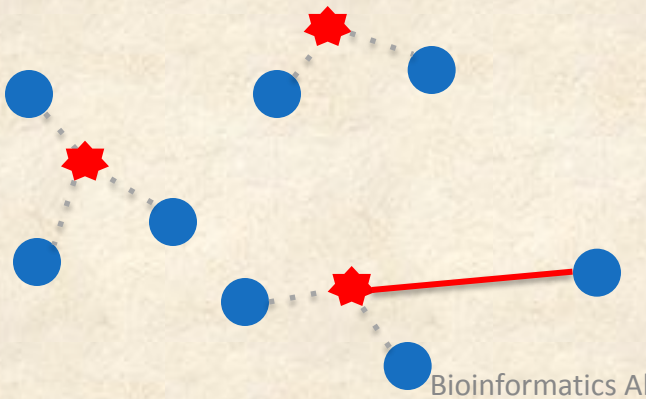
***k*-Means Clustering Problem:**

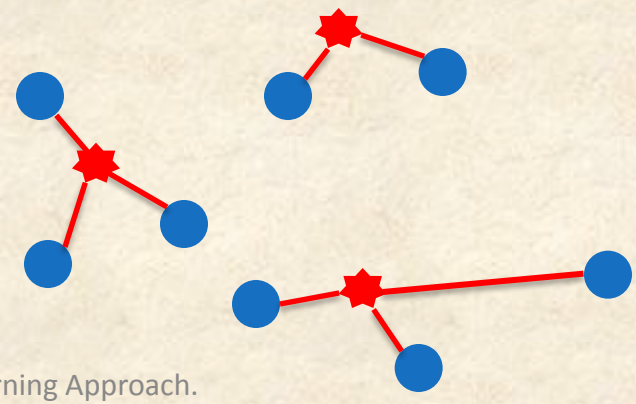
Input: A set of points *Data* and an integer *k*.

Output: A set of *k* points *Centers* that minimizes

Distortion(*Data*,*Centers*)

over all choices of *Centers*.





k-Means Clustering Problem

***k*-Center Clustering Problem:**

Input: A set of points *Data* and an integer *k*.

Output: A set of *k* points *Centers* that minimizes

MaxDistance(*DataPoints*,*Centers*)

over all choices of *Centers*.

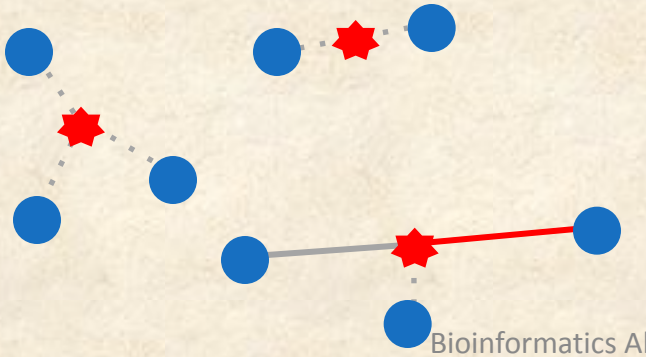
***k*-Means Clustering Problem:**

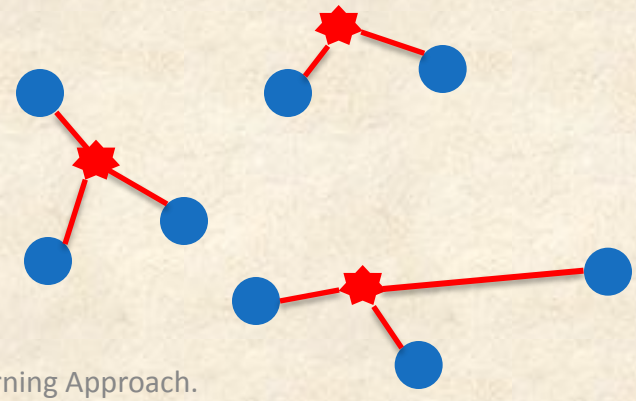
Input: A set of points *Data* and an integer *k*.

Output: A set of *k* points *Centers* that minimizes

Distortion(*Data*,*Centers*)

over all choices of *Centers*.





k-Means Clustering Problem

***k*-Center Clustering Problem:**

Input: A set of points *Data* and an integer *k*.

Output: A set of *k* points *Centers* that minimizes

MaxDistance(*DataPoints*,*Centers*)

over all choices of *Centers*.

***k*-Means Clustering Problem:**

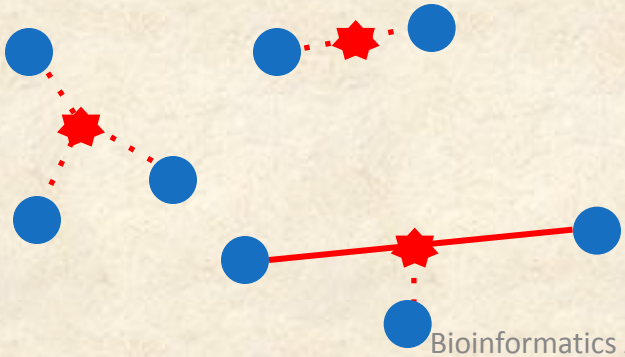
Input: A set of points *Data* and an integer *k*.

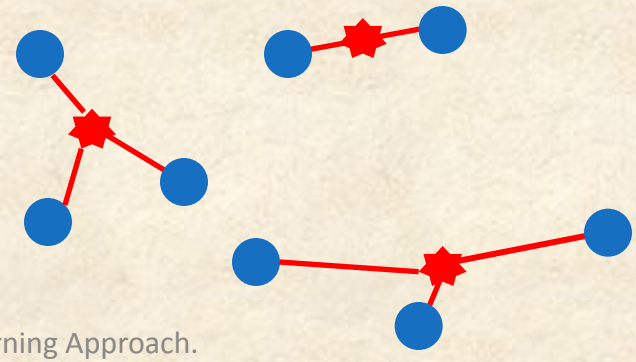
Output: A set of *k* points *Centers* that minimizes

Distortion(*Data*,*Centers*)

over all choices of *Centers*.

NP-Hard for $k > 1$



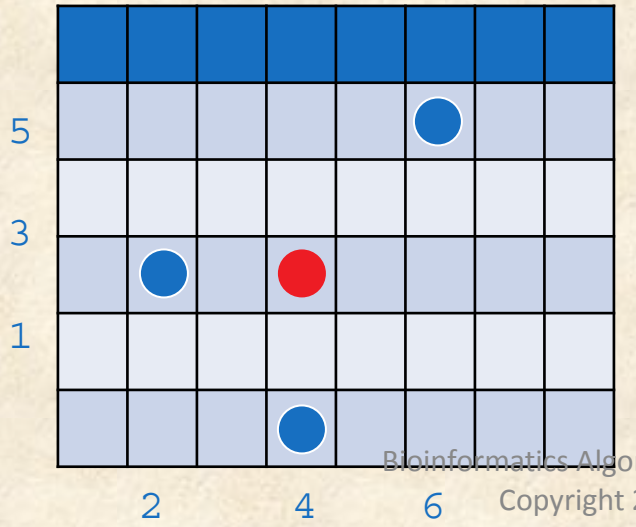


k-Means Clustering for *k* = 1

Center of Gravity Theorem: The center of gravity of points *Data* is the only point solving the 1-Means Clustering Problem.

The **center of gravity** of points *Data* is

$\sum_{\text{all points } DataPoint \text{ in } Data} DataPoint$ / #points in *Data*

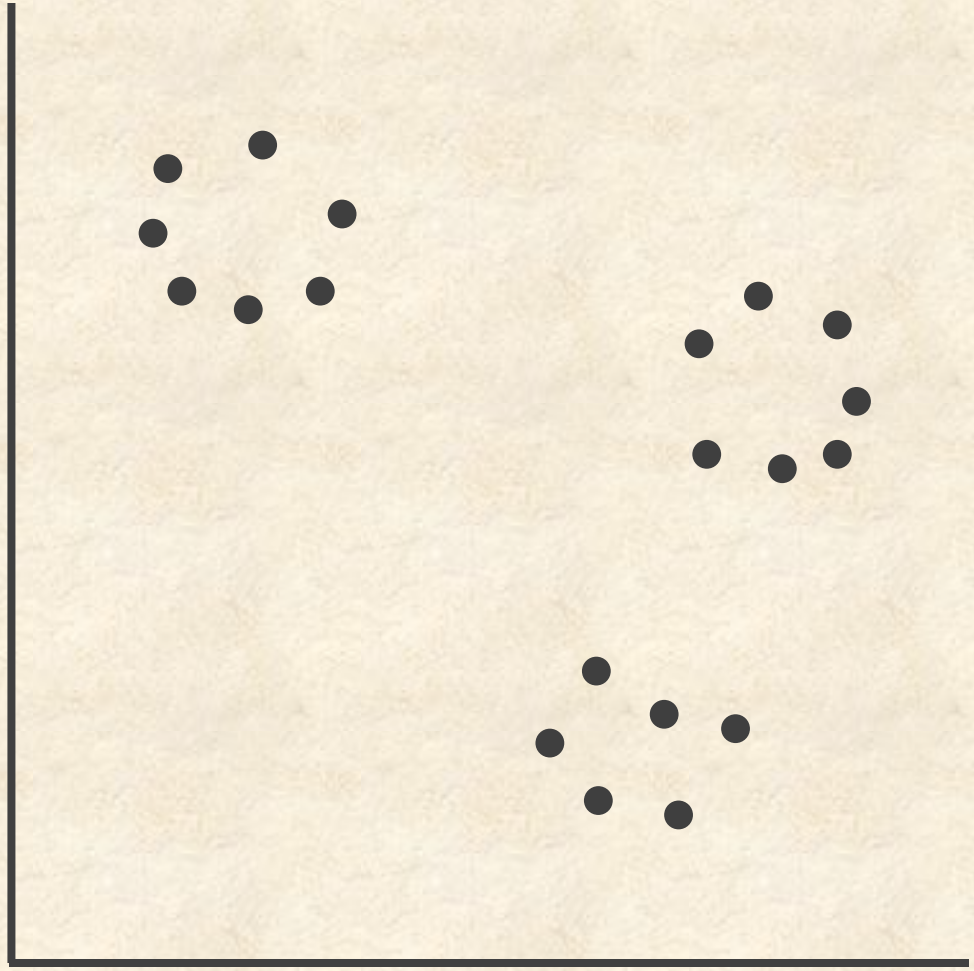


i-th coordinate of the center of gravity = the average of the *i*-th coordinates of datapoints:
((2+4+6)/3, (3+1+5)/3) = (4, 3)

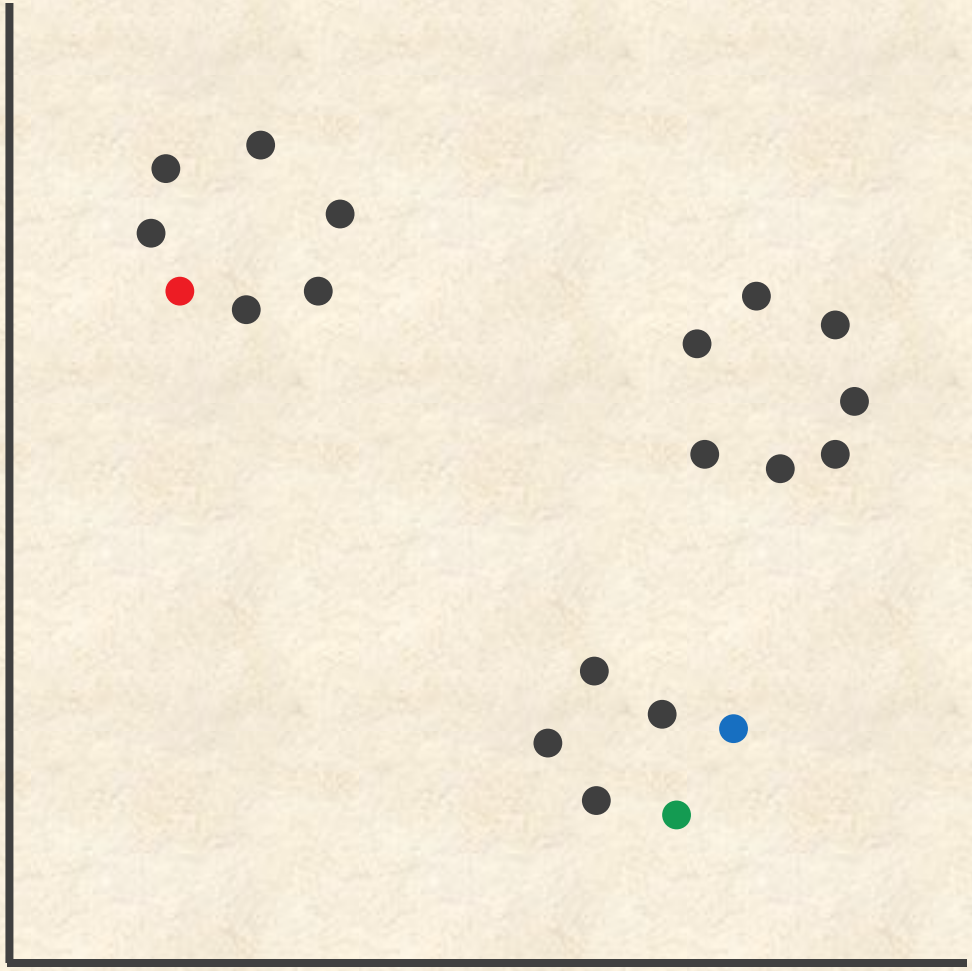
How Did Yeast Become a Wine Maker?

- Which Yeast Genes Are Responsible for Wine Brewing?
- Clustering as an optimization problem
- The Lloyd algorithm for *k*-means clustering
- From Hard to Soft Clustering
- From Coin Flipping to *k*-means Clustering
- Expectation Maximization
- Soft *k*-means Clustering
- Hierarchical Clustering

The Lloyd Algorithm in Action

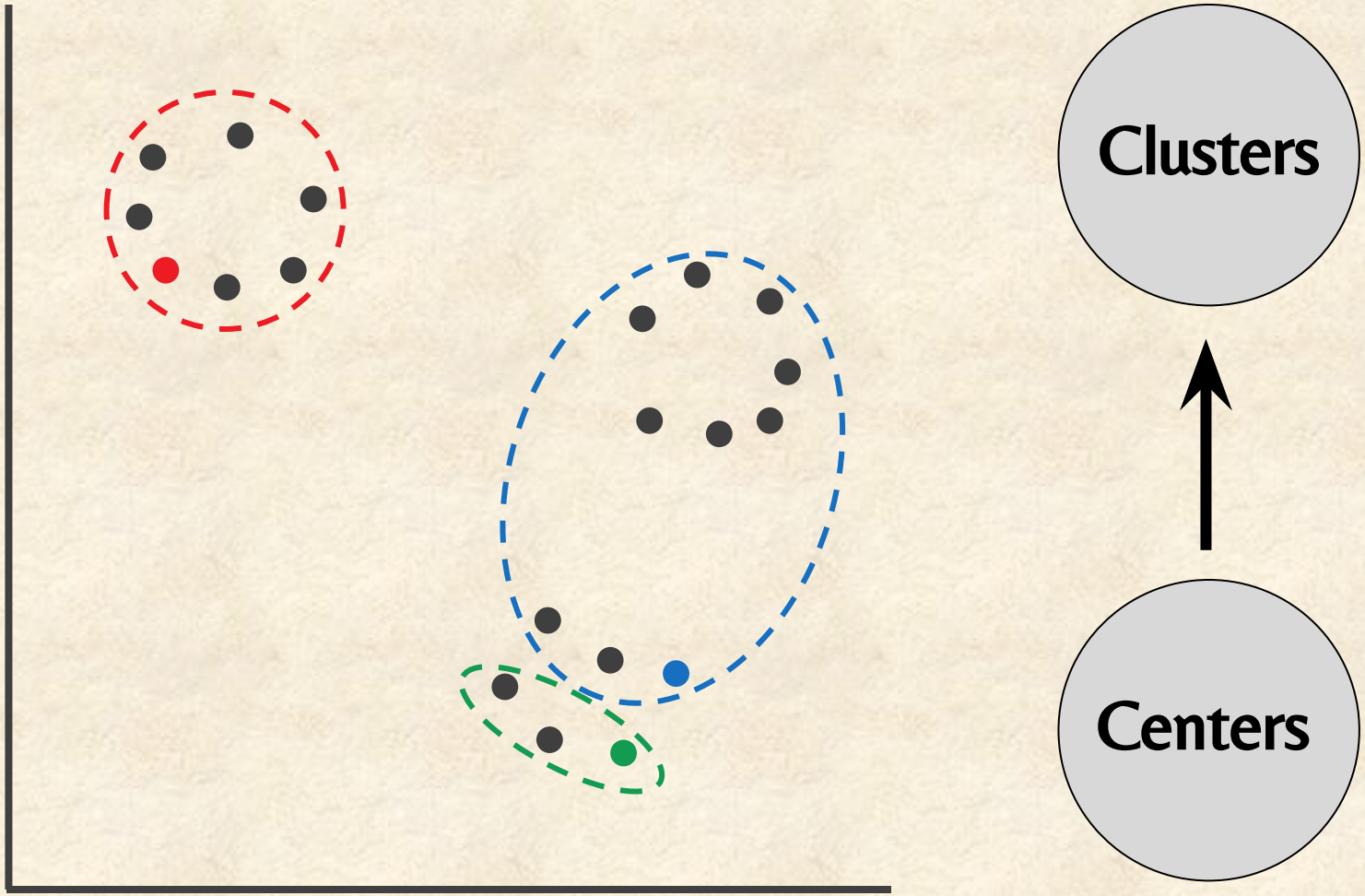


The Lloyd Algorithm in Action



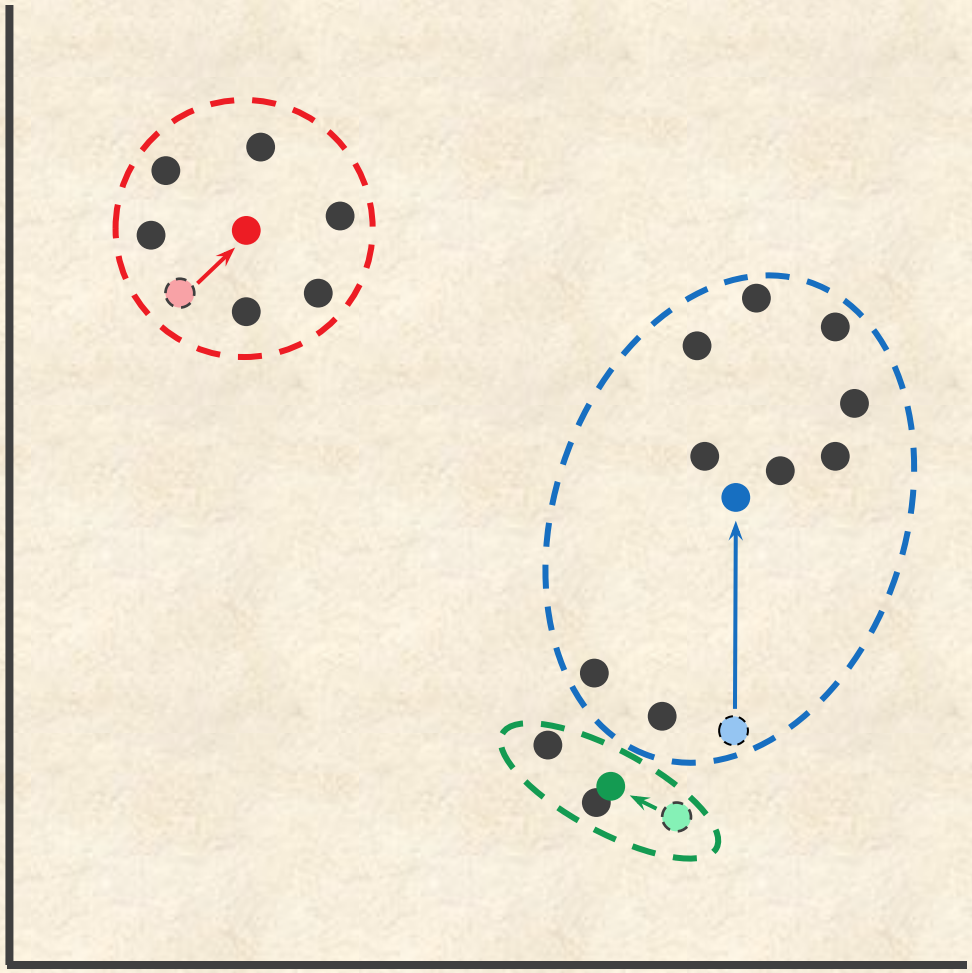
Select *k* arbitrary data points as *Centers*

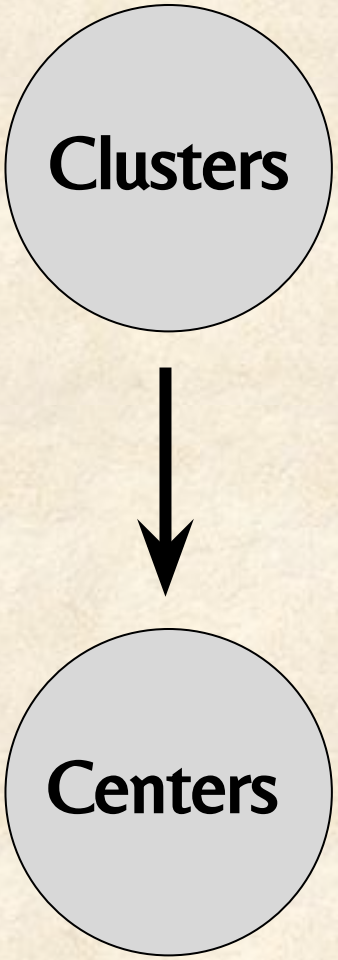
The Lloyd Algorithm in Action



assign each data point to its nearest center

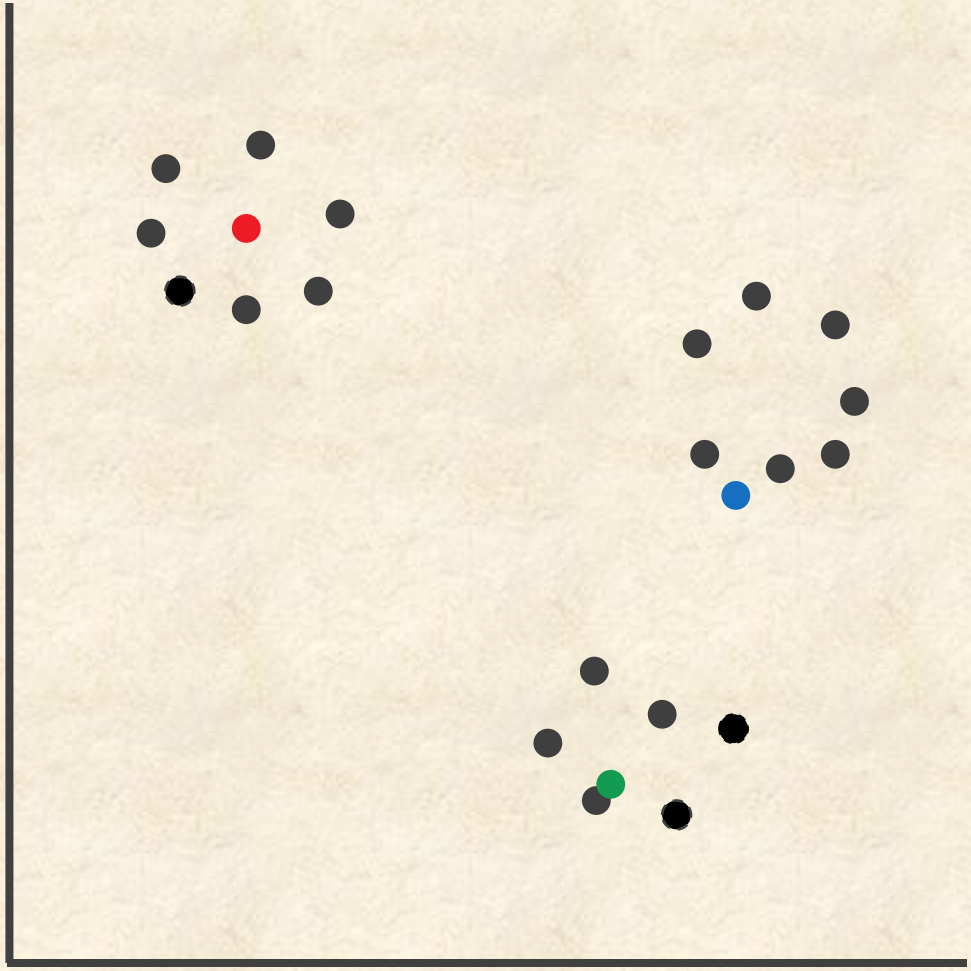
The Lloyd Algorithm in Action





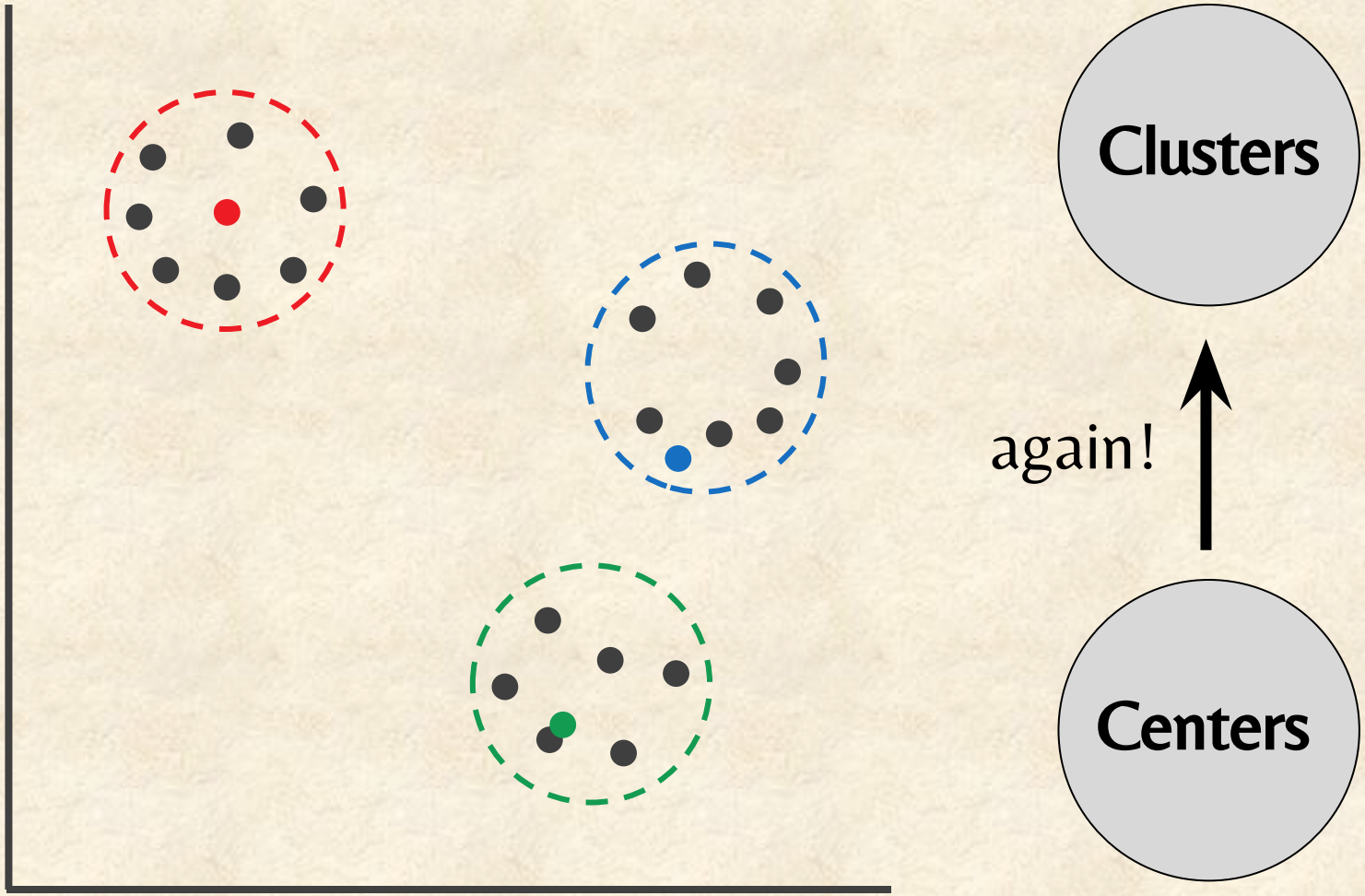
new centers □ clusters' centers of gravity

The Lloyd Algorithm in Action



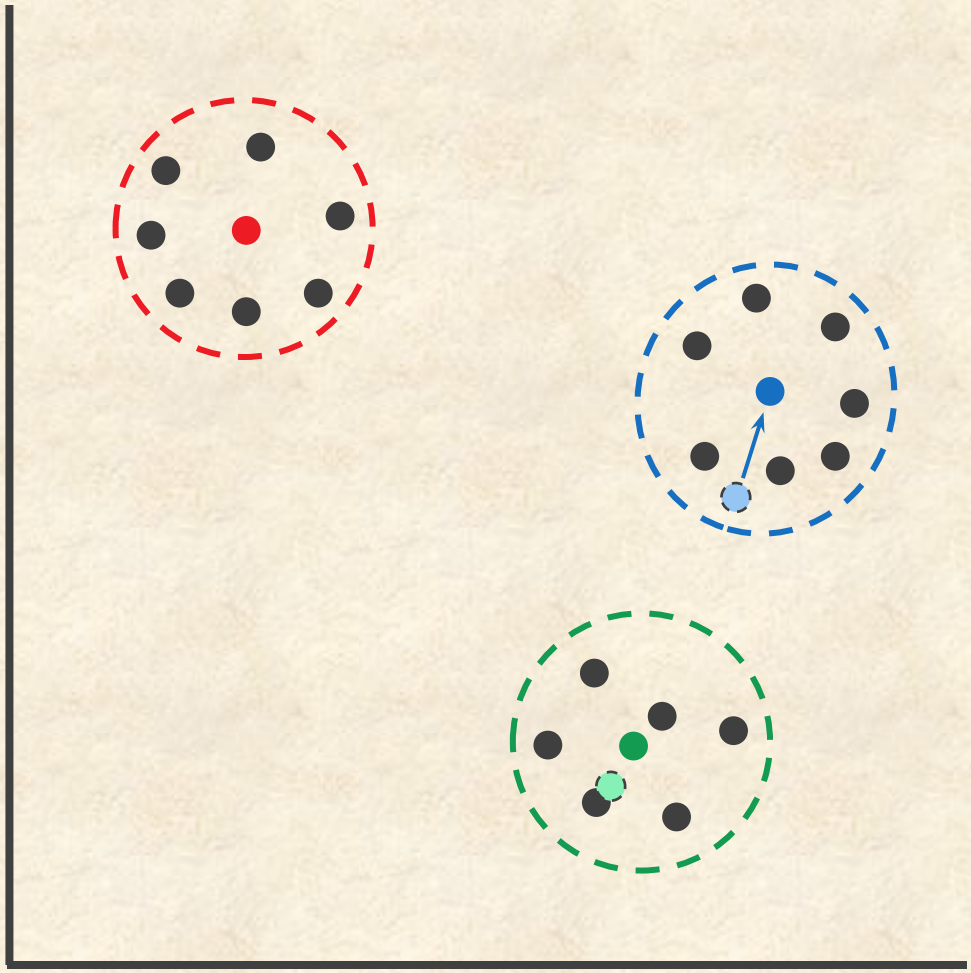
STOP and Think: What do we do now?

The Lloyd Algorithm in Action



assign each data point to its nearest center

The Lloyd Algorithm in Action



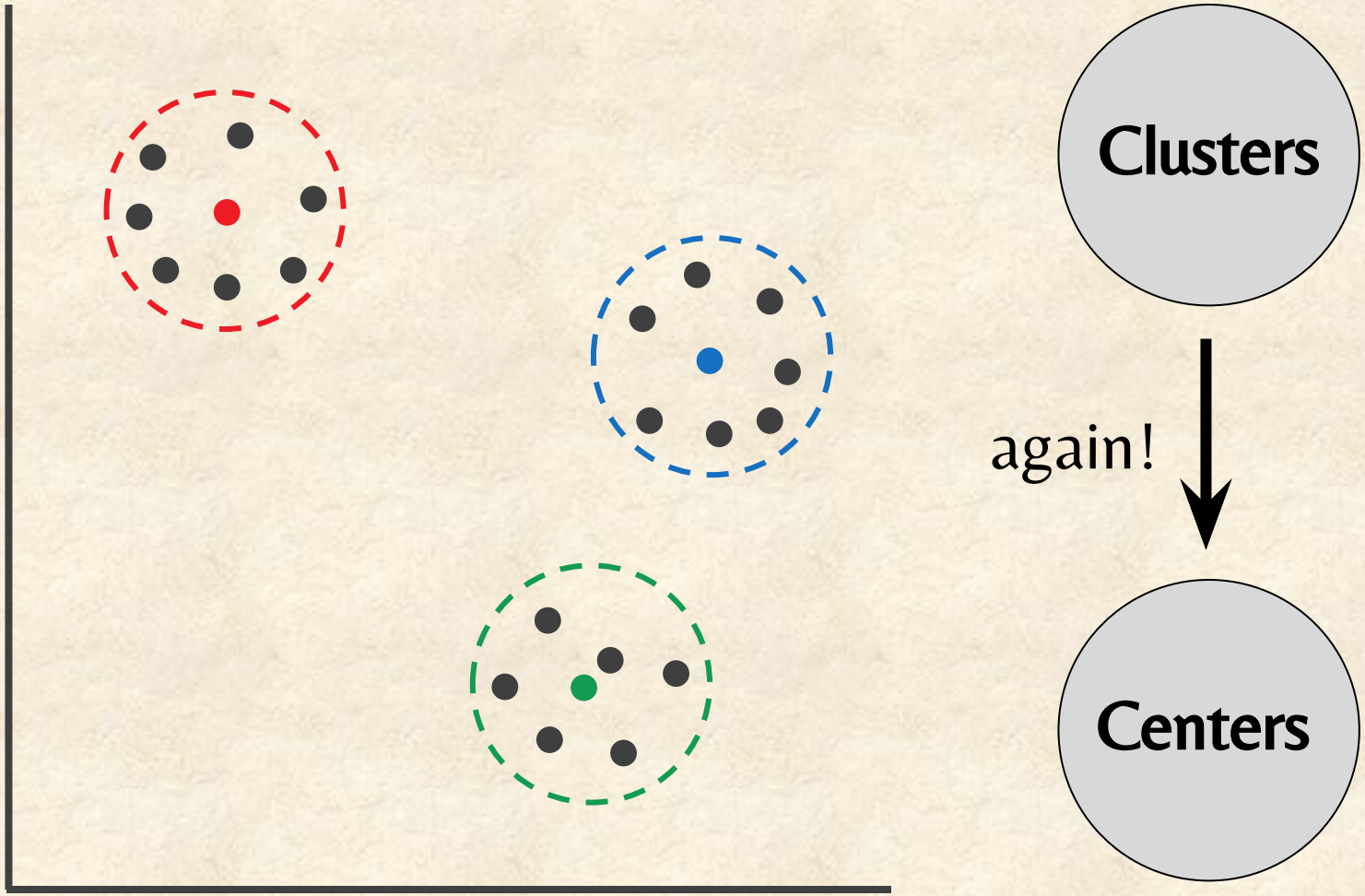
Clusters

again!

Centers

new centers □ clusters' centers of gravity

The Lloyd Algorithm in Action



assign each data point to its nearest center

Lloyd Algorithm

Select *k* arbitrary data points as *Centers* and then iteratively perform the following steps:

- **Centers to Clusters**: Assign each data point to the cluster corresponding to its nearest center (ties are broken arbitrarily).
- **Clusters to Centers**: After the assignment of data points to *k* clusters, compute new centers as clusters' center of gravity.

The Lloyd Algorithm

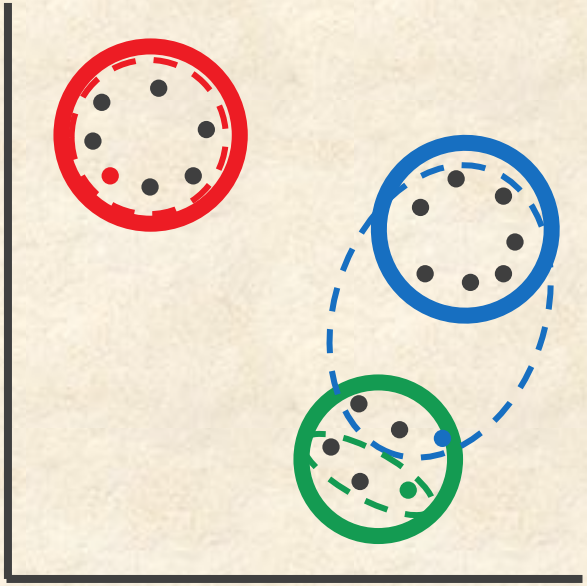
Select *k* arbitrary data points as *Centers* and then iteratively performs the following two steps:

- **Centers to Clusters**: Assign each data point to the cluster corresponding to its nearest center (ties are broken arbitrarily).
- **Clusters to Centers**: After the assignment of data points to *k* clusters, compute new centers as clusters' center of gravity.

The Lloyd algorithm terminates when the centers stop moving (**convergence**).

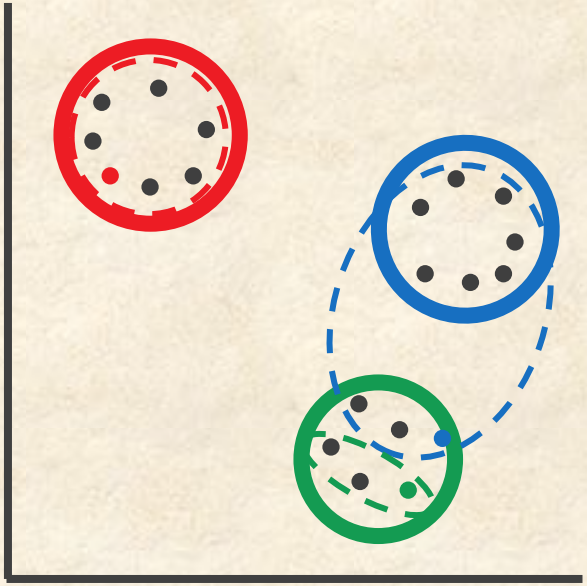
Must the Lloyd Algorithm Converge?

- If a data point is assigned to a new center during the **Centers to Clusters** step:
  - the squared error distortion is reduced because this center must be closer to the point than the previous center was.

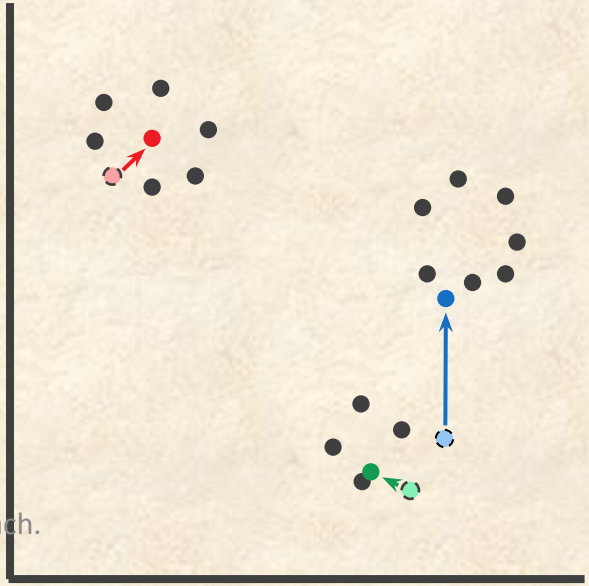


Must the Lloyd Algorithm Converge?

- If a data point is assigned to a new center during the **Centers to Clusters** step:
  - the squared error distortion is reduced because this center must be closer to the point than the previous center was.

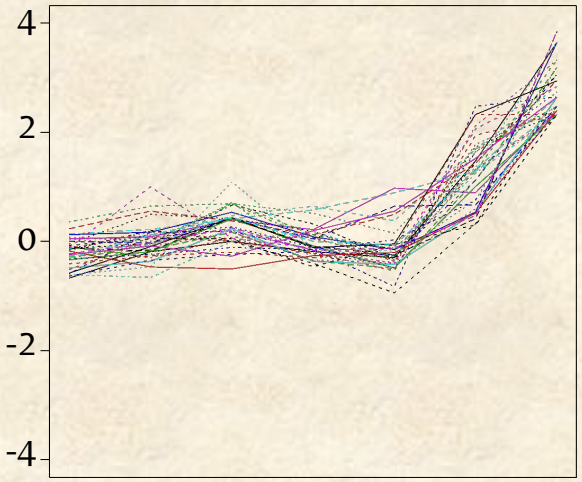


- If a center is moved during the **Clusters to Centers** step:
  - the squared error distortion is reduced since the center of gravity is the *only point* minimizing the distortion (the Center of Gravity Theorem).

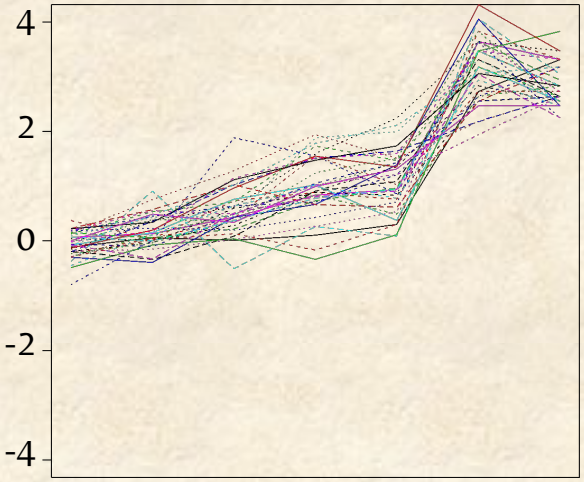


Clustering 230 Yeast Genes

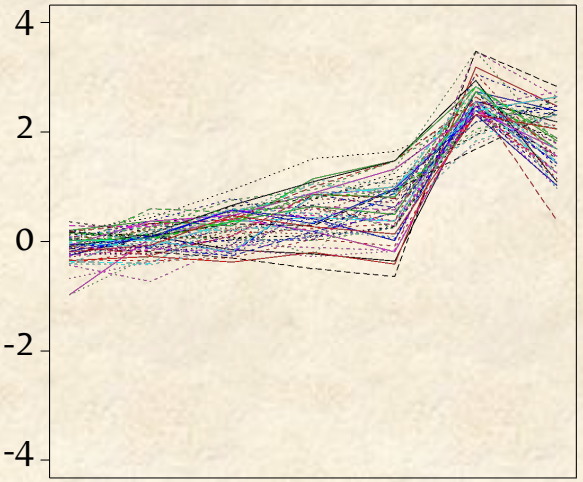
Cluster 1



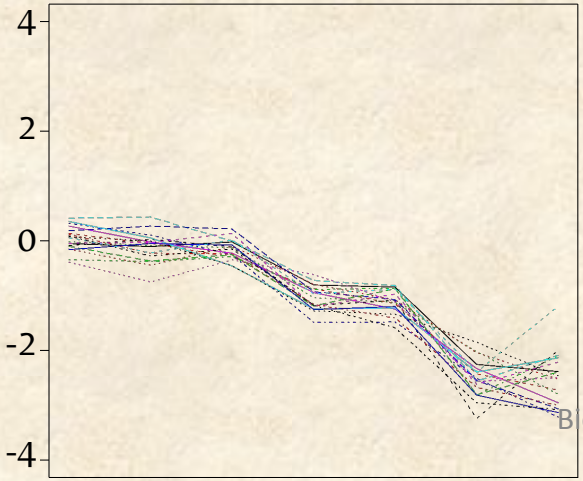
Cluster 2



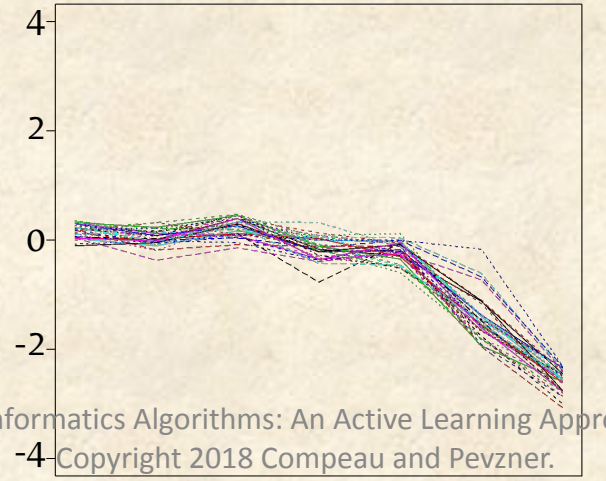
Cluster 3



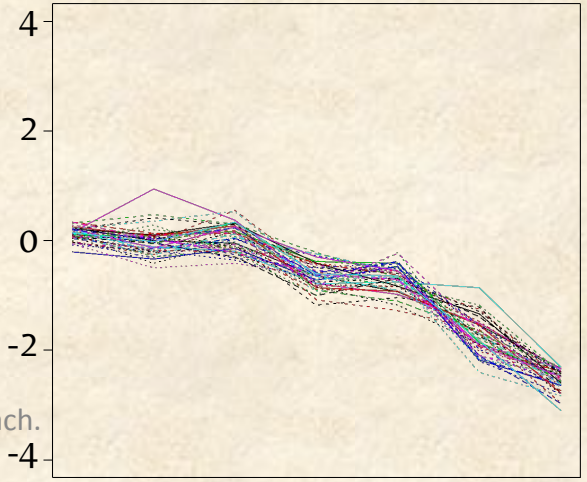
Cluster 4



Cluster 5

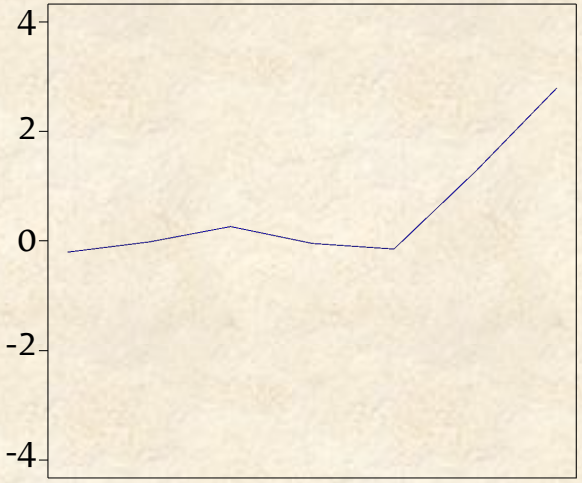


Cluster 6

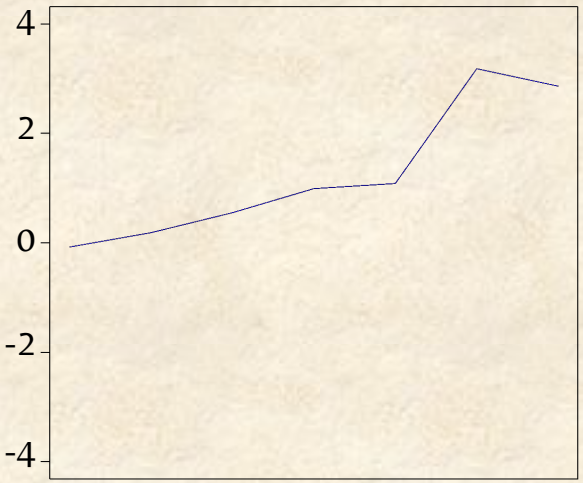


Clustering 230 Yeast Genes

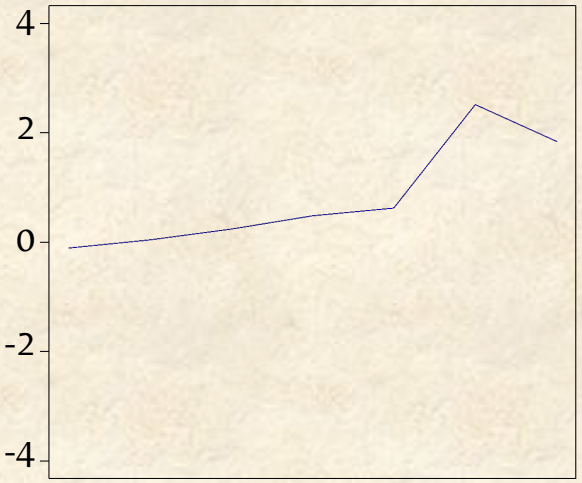
Cluster 1



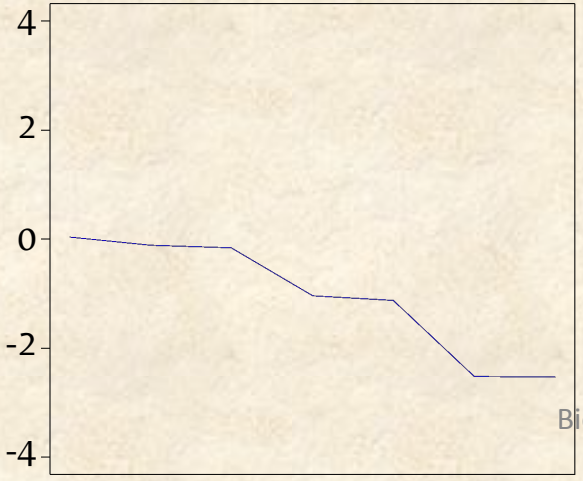
Cluster 2



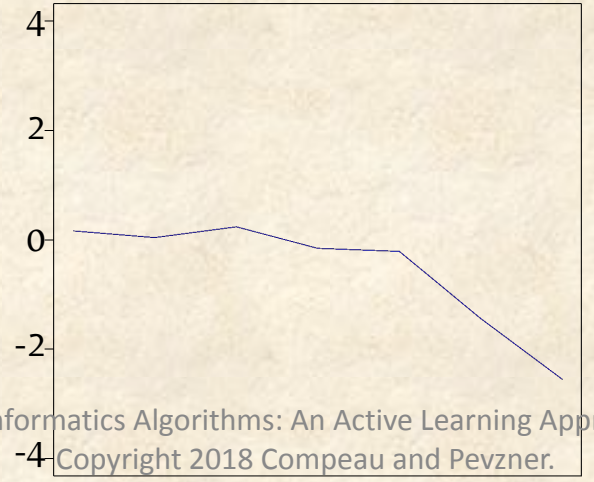
Cluster 3



Cluster 4

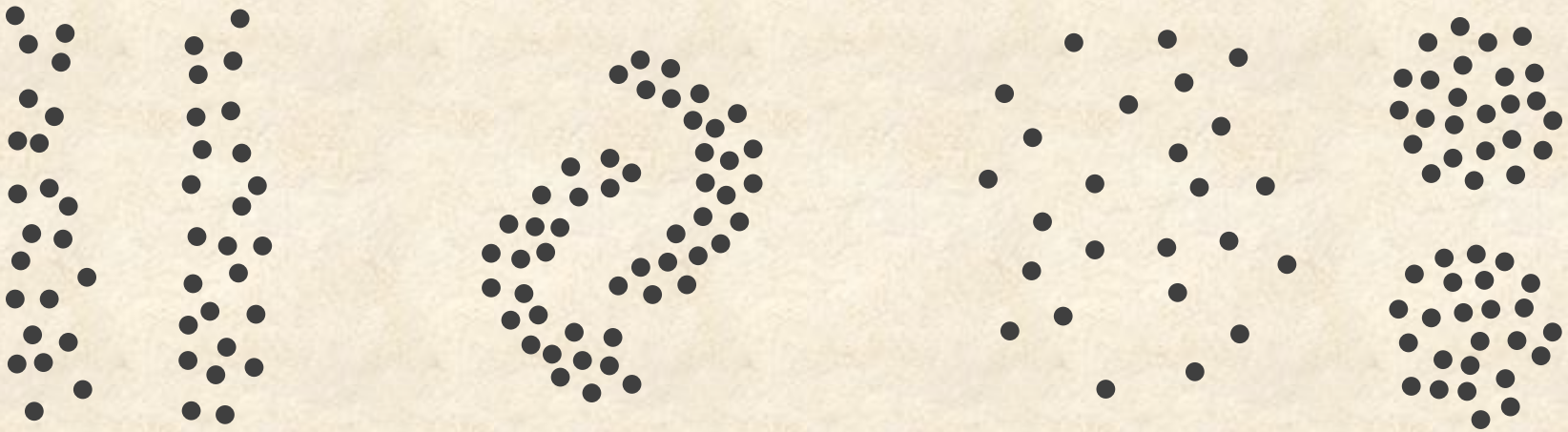


Cluster 5



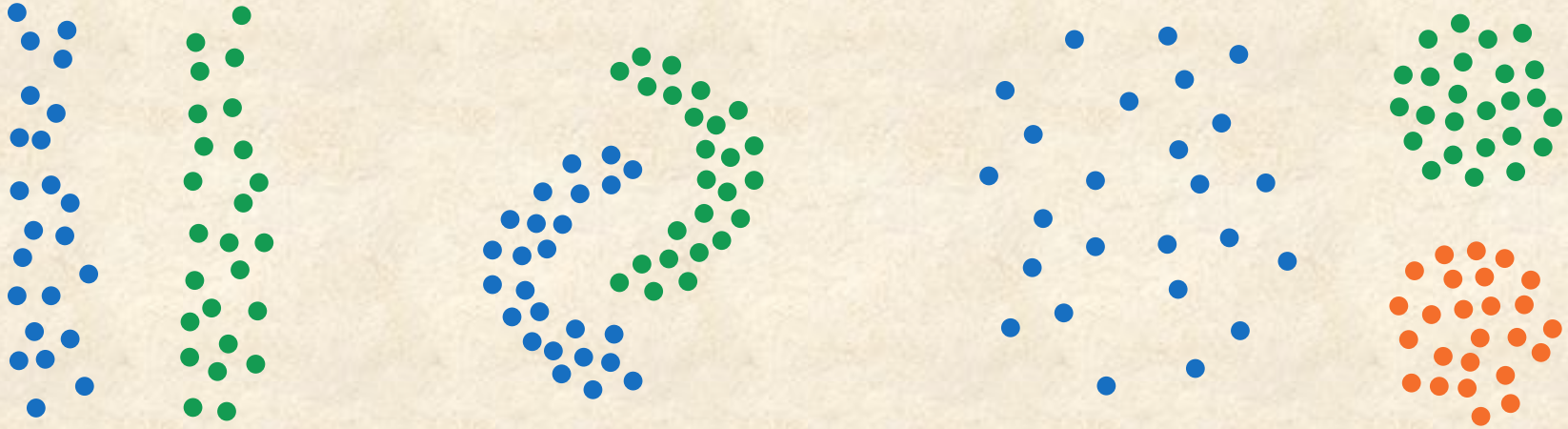
Cluster 6

k-means Clustering vs. the Human Eye



STOP and Think: How would you cluster these three sets of points?

k-means Clustering vs. the Human Eye



STOP and Think: How would the *Lloyd algorithm* cluster these sets of points?

How Did Yeast Become a Wine Maker?

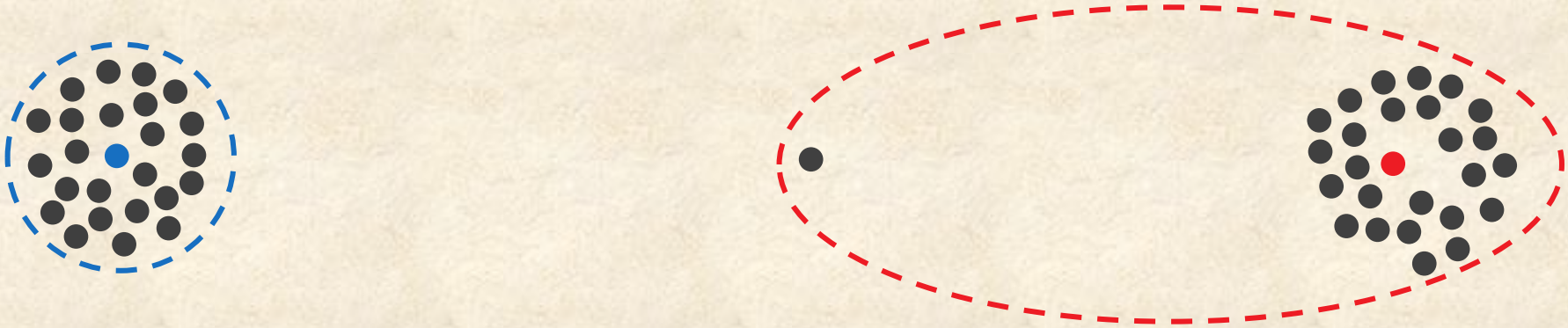
- Which Yeast Genes Are Responsible for Wine Brewing?
- Clustering as an optimization problem
- The Lloyd algorithm for *k*-means clustering
- From Hard to Soft Clustering
- From Coin Flipping to *k*-means Clustering
- Expectation Maximization
- Soft *k*-means Clustering
- Hierarchical Clustering

Soft vs. Hard Clustering

Midpoint: A point approximately halfway between two clusters.

Soft vs. Hard Clustering

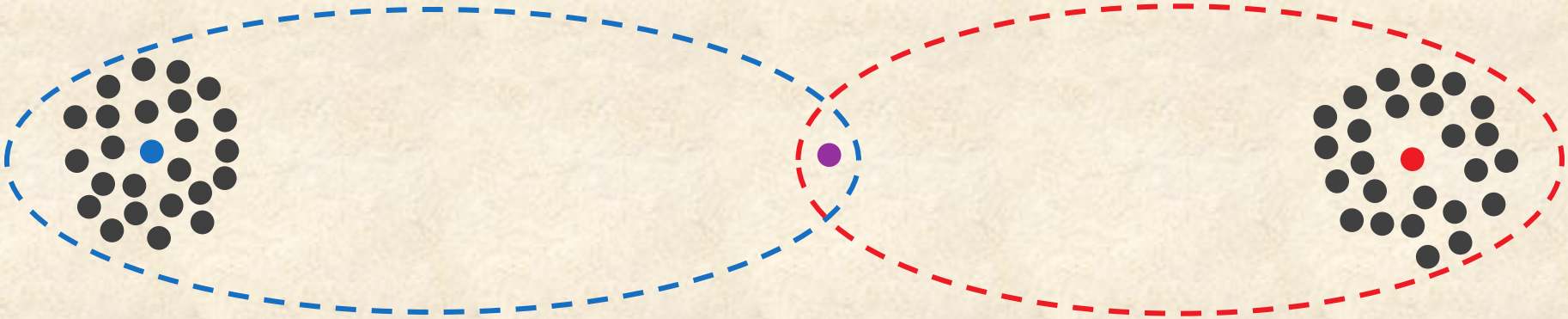
- The Lloyd algorithm assigns the midpoint either to the red or to the blue cluster.
  - "**hard**" assignment of data points to clusters.



- Can we color the midpoint half-red and half-blue?
  - "**soft**" assignment of data points to clusters.

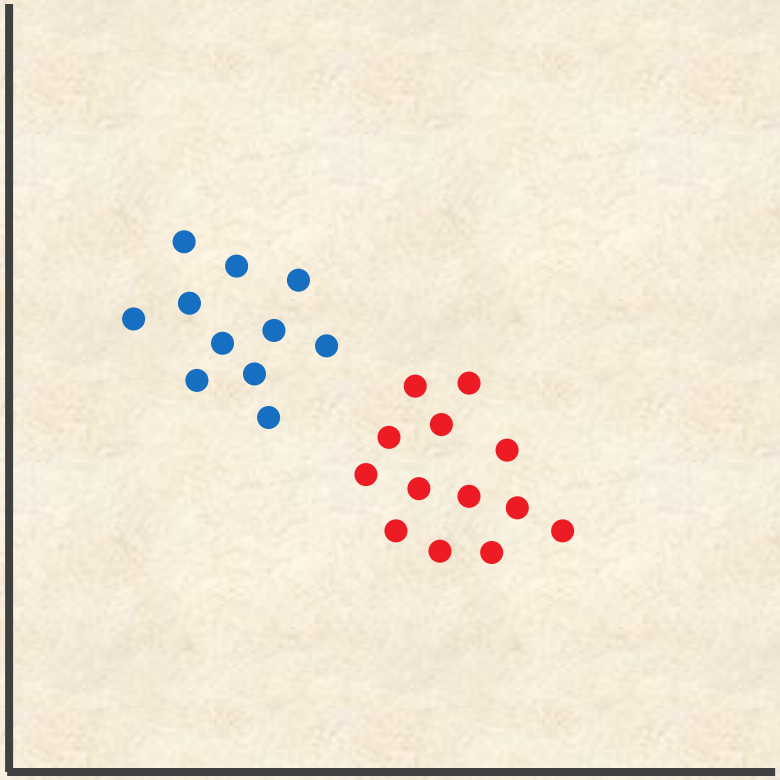
Soft vs. Hard Clustering

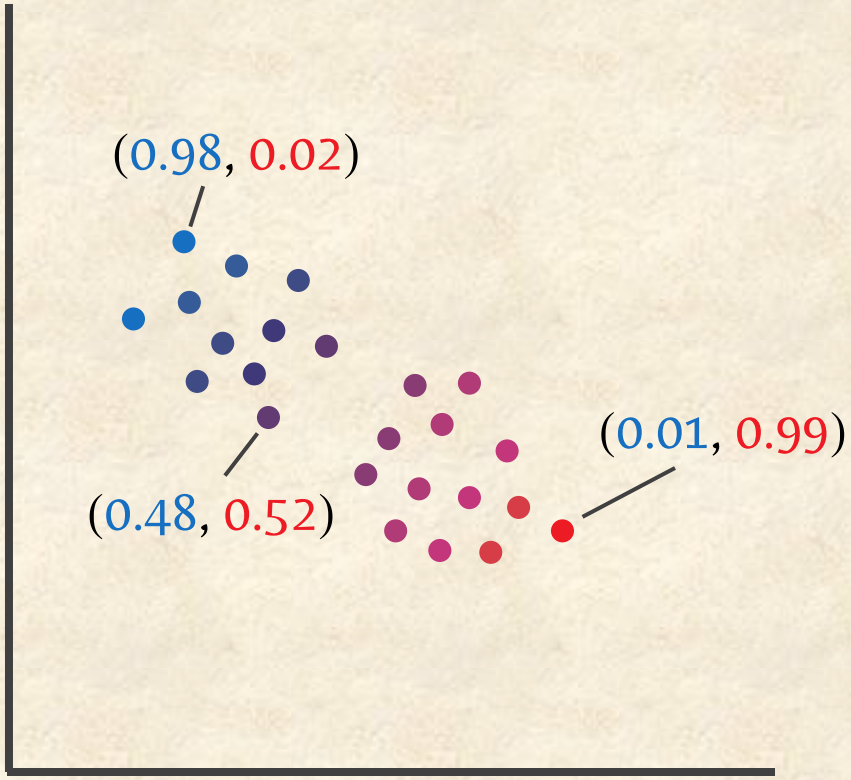
- The Lloyd algorithm assigns the midpoint either to the red or to the blue cluster.
  - "**hard**" assignment of data points to clusters.



- Can we color the midpoint half-red and half-blue?
  - "**soft**" assignment of data points to clusters.

Soft vs. Hard Clustering





Hard choices: points are colored red or blue depending on their cluster membership.

Soft choices: points are assigned "red" and "blue" *responsibilities* r_{blue} and r_{red} ($r_{blue} + r_{red} = 1$)

How Did Yeast Become a Wine Maker?

- Which Yeast Genes Are Responsible for Wine Brewing?
- Clustering as an optimization problem
- The Lloyd algorithm for *k*-means clustering
- From Hard to Soft Clustering
- From Coin Flipping to *k*-means Clustering
- Expectation Maximization
- Soft *k*-means Clustering
- Hierarchical Clustering

Flipping One Biased Coins

- We flip a loaded coin with an **unknown bias θ** (probability that the coin lands on heads).
- The coin lands on heads ***i* out of *n*** times.
- For each bias, we can compute the probability of the resulting sequence of flips.

Probability of generating the given sequence of flips is

$$\Pr(\text{sequence}|\theta) = \theta^{i} * (1-\theta)^{n-i}$$

This expression is minimized at **$\theta = i/n$** (most likely bias)

Flipping Two Biased Coins

A *B*

	Data
HTTTHTTHTH	0.4
HHHHTHHHHH	0.9
HTHHHHHTHH	0.8
HTTTTTHHTT	0.3
THHHTHHHTH	0.7

Goal: estimate the probabilities θ_A and θ_B

If We Knew Which Coin Was Used in Each Sequence...

	Data	*HiddenVector*
HTTTHTTHTH	0.4	1
HHHHTHHHHH	0.9	0
HTHHHHHTHH	0.8	0
HTTTTTHHTT	0.3	1
THHHTHHHTH	0.7	0

Goal: estimate *Parameters* = (θ_A,θ_B) when *HiddenVector* is given

If We Knew Which Coin Was Used in Each Sequence...

Data	*HiddenVector*
HTTTHTTHTH	0.4	1
HHHHTHHHHH	0.9	0
HTHHHHHTHH	0.8	0
HTTTTTHHTT	0.3	1
THHHTHHHTH	0.7	0

θ_A = fraction of heads generated in all flips with coin A =
(4+3) / (10+10) = (0.4+0.3) / 2 = 0.35

If We Knew Which Coin Was Used in Each Sequence...

Data	*HiddenVector*
HTTTHTTHTH	0.4	1
HHHHTHHHHH	0.9	0
HTHHHHHTHH	0.8	0
HTTTTTHHTT	0.3	1
THHHTHHHTH	0.7	0

θ_A = fraction of heads generated in all flips with coin A =
(4+3) / (10+10) = (0.4+0.3) / 2 = 0.35

θ_B = fraction of heads generated in all flips with coin B =
(9+8+7) / (10+10+10) = (0.9+0.8+0.7) / (1+1+1) = 0.80

Parameters as a Dot-Product

	Data	*HiddenVector*	*Parameters*=(θ_A, θ_B)
HTTTHTTHTH	**0.4**	* **1**	
HHHHTHHHHH	0.9	* 0	
HTHHHHHTHH	0.8	* 0	(0.35, 0.80)
HTTTTTHHTT	**0.3**	* **1**	
THHHTHHHTH	0.7	* 0	

θ_A = fraction of heads generated in all flips with coin A =

= (4+3) / (10+10) = (0.4+0.3) / 2 = 0.35

(0.4*1+0.9*0+0.8*0+0.3*1+0.7*0)/ (1+0+0+1+0) = 0.35

$\sum_{\text{all data points } i} Data_i * HiddenVector_i \;/\; \sum_{\text{all data points } i} HiddenVector_i = 0.35$

Data * *HiddenVector* / (1,**1**,..., **1**)***HiddenVector* =0.35

1 refers to a vector (1,1, ... ,1) consisting of all 1s

Parameters as a Dot-Product

Data		HiddenVector	Parameters=(θ_A, θ_B)
HTTTHTTHTH	0.4	* 1	
HHHHTHHHHH	0.9	* 0	
HTHHHHHTHH	0.8	* 0	(0.35, 0.80)
HTTTTTHHTT	0.3	* 1	
THHHTHHHTH	0.7	* 0	

θ_B = fraction of heads generated in all flips with coin *B*

= (9+8+7) / (10+10+10) = (0.9+0.8+0.7) /(1+1+1) = 0.80

(0.5*0+0.9*1+0.8*1+0.4*0+0.7*1) / (0+1+1+0+1) = 0.80

$\sum_{\text{all points } i} Data_i * (1- HiddenVector_i) / \sum_{\text{all points } i} (1- HiddenVector_i) =$

Data * (1-*HiddenVector*) / **1** * (1 - *HiddenVector*)

Parameters as a Dot-Product

	Data		*HiddenVector*	*Parameters*=(θ_A, θ_B)
HTTT**H**TT**H**T**H**	0.4	*	1	
HHHHT**HHHHH**	0.9	*	0	
HTHHHHHT**HH**	0.8	*	0	(0.35, 0.80)
HTTTTT**HH**TT	0.3	*	1	
T**HHH**T**HHH**T**H**	0.7	*	0	

θ_A = fraction of heads generated in all flips with coin *A*

= (0.4+0.3)/2=0.35

= *Data* * *HiddenVector* / **1** * *HiddenVector*

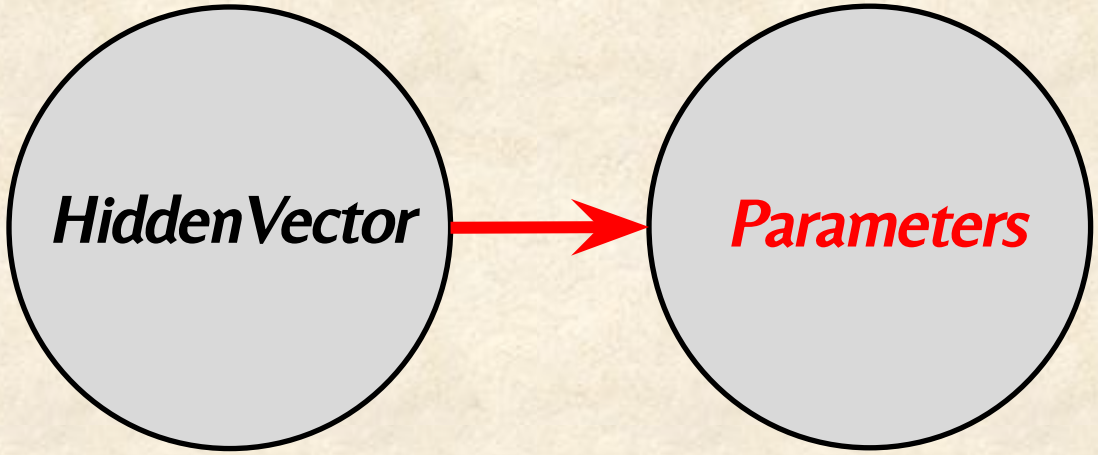
θ_B = fraction of heads generated in all flips with coin *B*

= (0.9+0.8+0.7)/3=0.80

= *Data* * (**1**-*HiddenVector*) / **1** * (**1** - *HiddenVector*)

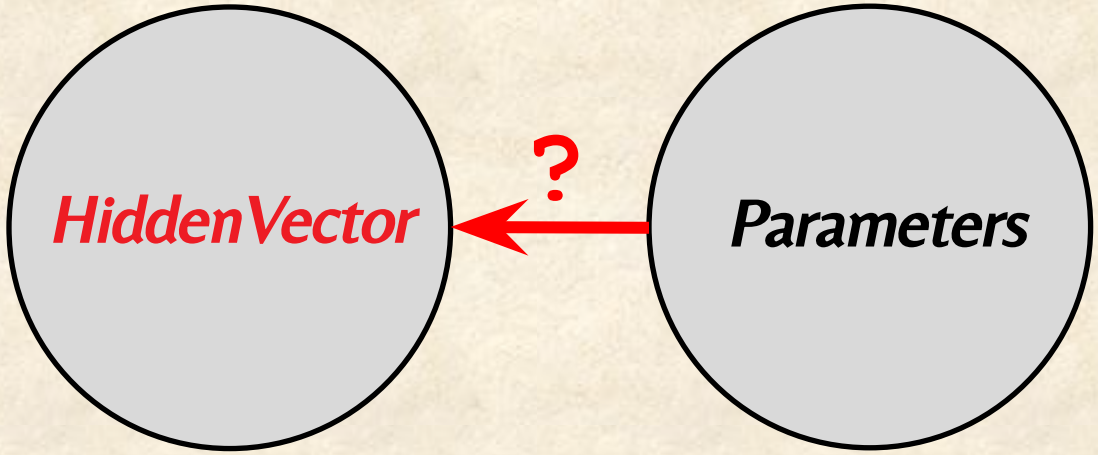
Data, HiddenVector, Parameters

Data	HiddenVector	Parameters=(θ_A, θ_B)
0.4	1	
0.9	0	
0.8	0	→ (0.35, 0.80)
0.3	1	
0.7	0	



Data, HiddenVector, Parameters

Data	HiddenVector	Parameters=(θ_A, θ_B)
0.4	?	
0.9	?	
0.8	?	← (0.35, 0.80)
0.3	?	
0.7	?	



From *Data & Parameters to HiddenVector*

Data	***HiddenVector***	***Parameters*=(θ_A, θ_B)**
0.4	?	
0.9	?	
0.8	?	← (0.35, 0.80)
0.3	?	
0.7	?	

STOP and Think: Which coin is more likely to generate the 1st sequence (with 4 H)?

$\Pr(1^{st}\text{ sequence}|\theta_A)=\theta_A^4(1-\theta_A)^6 = 0.35^4 \bullet 0.65^6 \simeq 0.00113$ **>**

$\Pr(1^{st}\text{ sequence}|\theta_B)=\theta_B^4(1-\theta_B)^6 = 0.80^4 \bullet 0.20^6 \simeq 0.00003$

From *Data & Parameters to HiddenVector*

Data	***HiddenVector***	***Parameters*=(θ_A, θ_B)**
0.4	1	
0.9	?	
0.8	?	← (0.35, 0.80)
0.3	?	
0.7	?	

STOP and Think: Which coin is more likely to generate the 1st sequence (with 4 H)?

$\Pr(1^{st}\ \text{sequence}|\theta_A) = \theta_A^4 (1-\theta_A)^6 = 0.35^4 \bullet 0.65^6 \simeq 0.00113 >$
$\Pr(1^{st}\ \text{sequence}|\theta_B) = \theta_B^4 (1-\theta_B)^6 = 0.80^4 \bullet 0.20^6 \simeq 0.00003$

From *Data* & *Parameters* to *HiddenVector*

Data	*HiddenVector*	*Parameters*=(θ_A, θ_B)
0.4	1	
0.9	?	
0.8	?	← (0.35, 0.80)
0.3	?	
0.7	?	

STOP and Think: Which coin is more likely to generate the 2nd sequence (with 9 H)?

$\Pr(2^{nd}\text{ sequence}|\theta_A) = \theta_A^9 (1-\theta_A)^1 = 0.35^9 \bullet 0.65^1 \simeq 0.00005$ **<**

$\Pr(2^{nd}\text{ sequence}|\theta_B) = \theta_B^9 (1-\theta_B)^1 = 0.80^9 \bullet 0.20^1 \simeq 0.02684$

From *Data & Parameters* to *HiddenVector*

Data	*HiddenVector*	*Parameters*=(θ_A, θ_B)
0.4	1	
0.9	0	
0.8	?	← (0.35, 0.80)
0.3	?	
0.7	?	

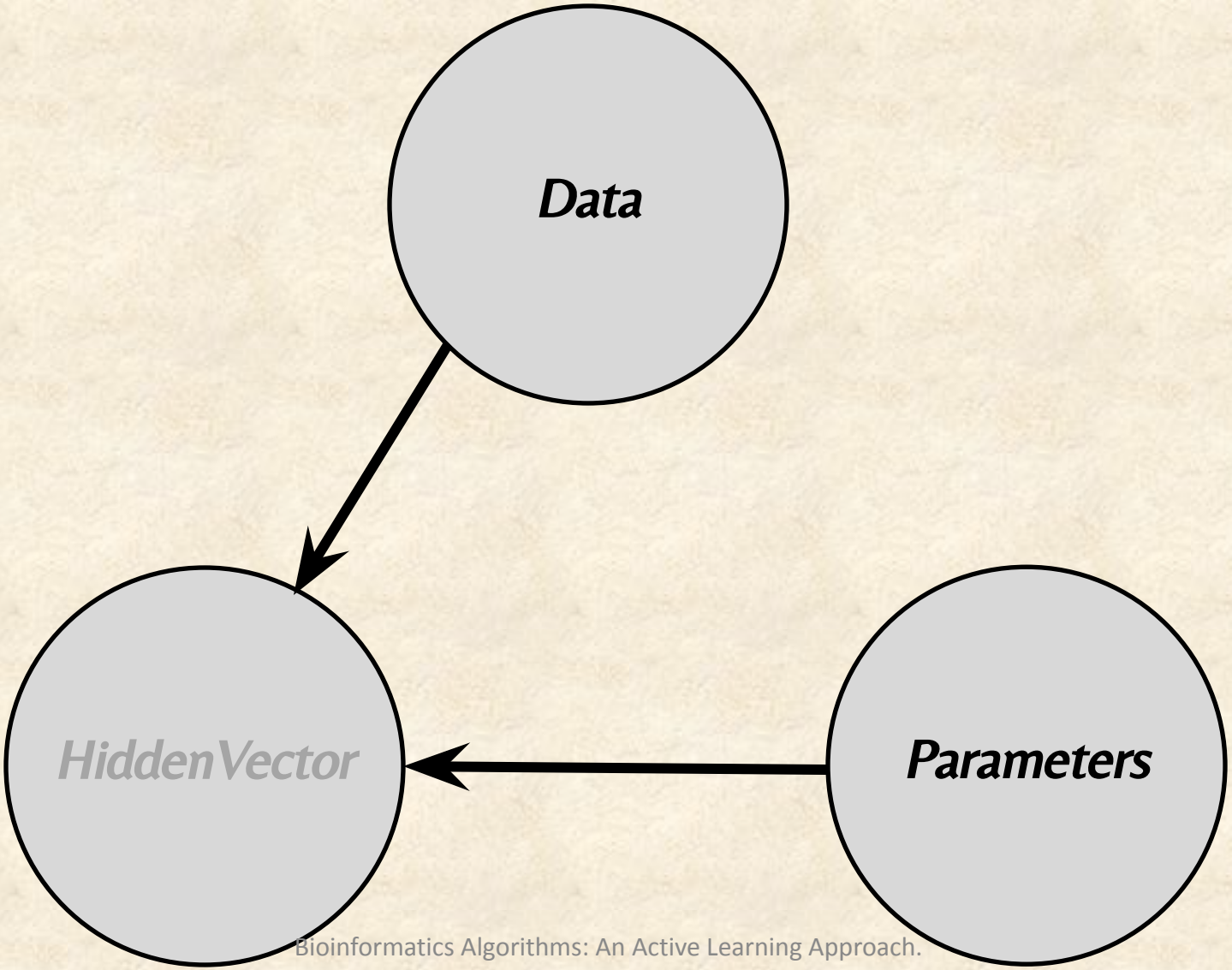
STOP and Think: Which coin is more likely to generate the 2nd sequence (with 9 H)?

$\Pr(2^{nd}\text{ sequence}|\theta_A) = \theta_A^{9}(1-\theta_A)^{1} = 0.35^{9} \bullet 0.65^{1} \simeq 0.00005 <$
$\Pr(2^{nd}\text{ sequence}|\theta_B) = \theta_B^{9}(1-\theta_B)^{1} = 0.80^{9} \bullet 0.20^{1} \simeq 0.02684$

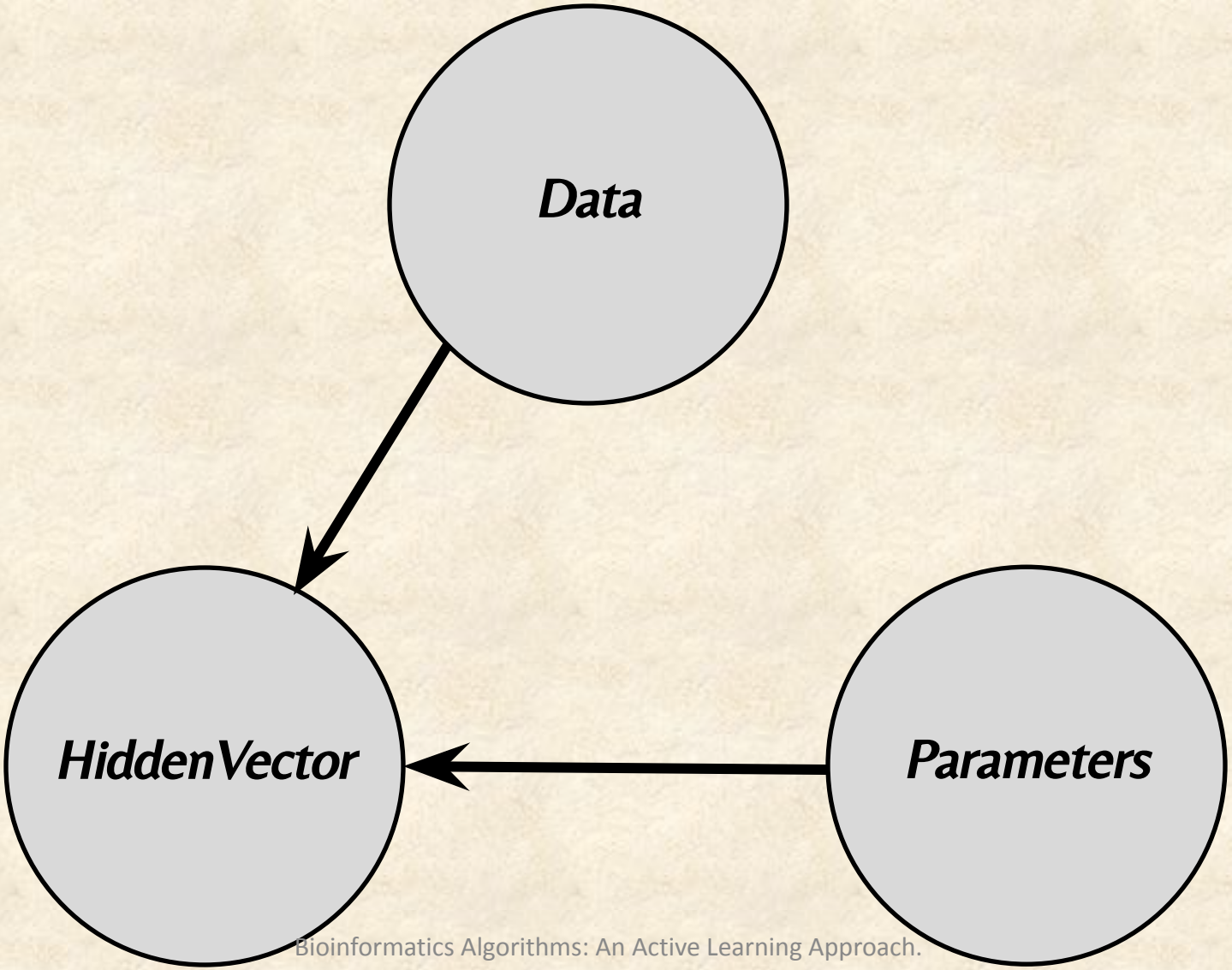
HiddenVector Reconstructed!

Data	*HiddenVector*	*Parameters*=(θ_A, θ_B)
0.4	1	
0.9	0	
0.8	0	← (0.35, 0.80)
0.3	1	
0.7	0	

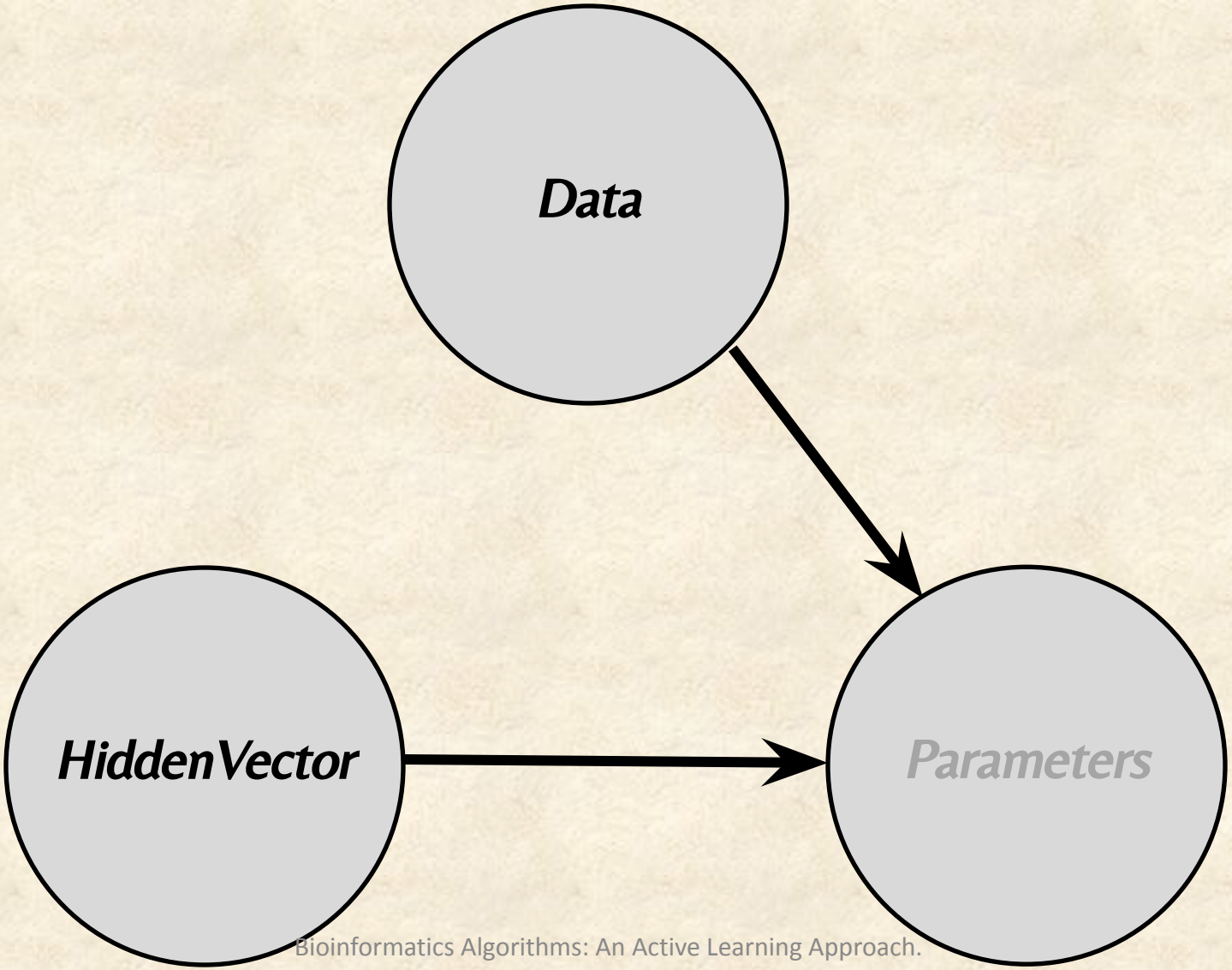
Reconstructing *HiddenVector* and *Parameters*



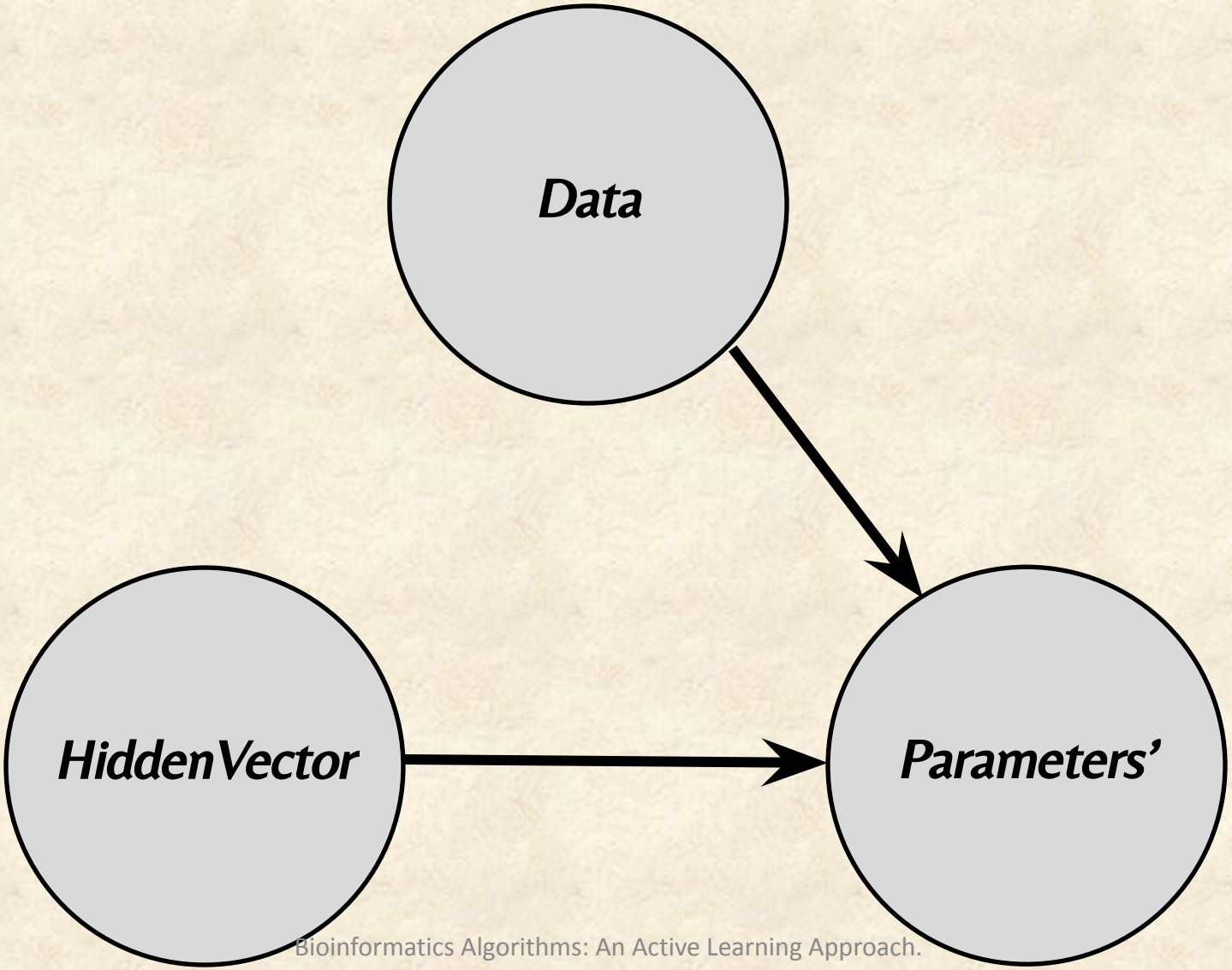
Reconstructing *HiddenVector* and *Parameters*



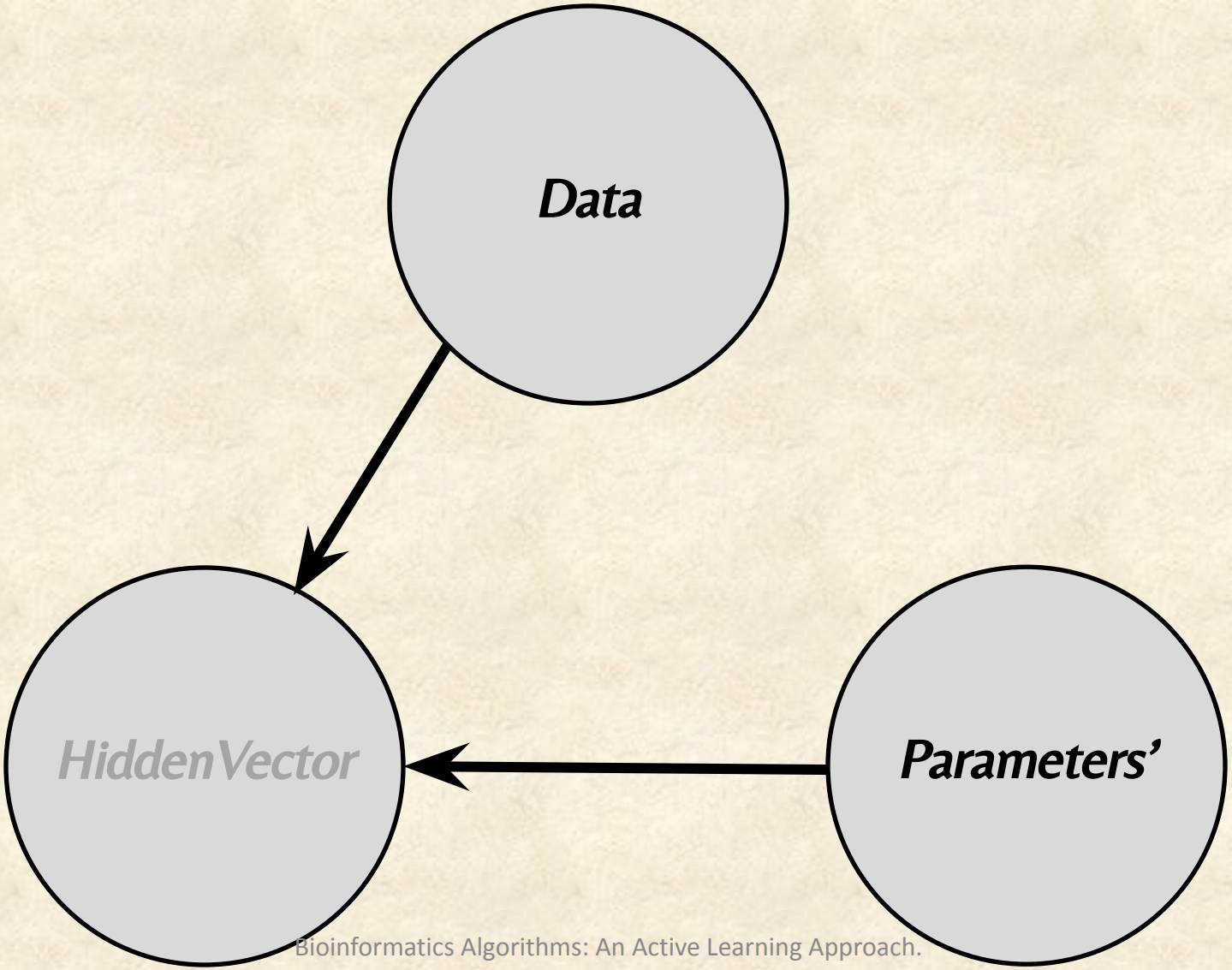
Reconstructing *HiddenVector* and *Parameters*



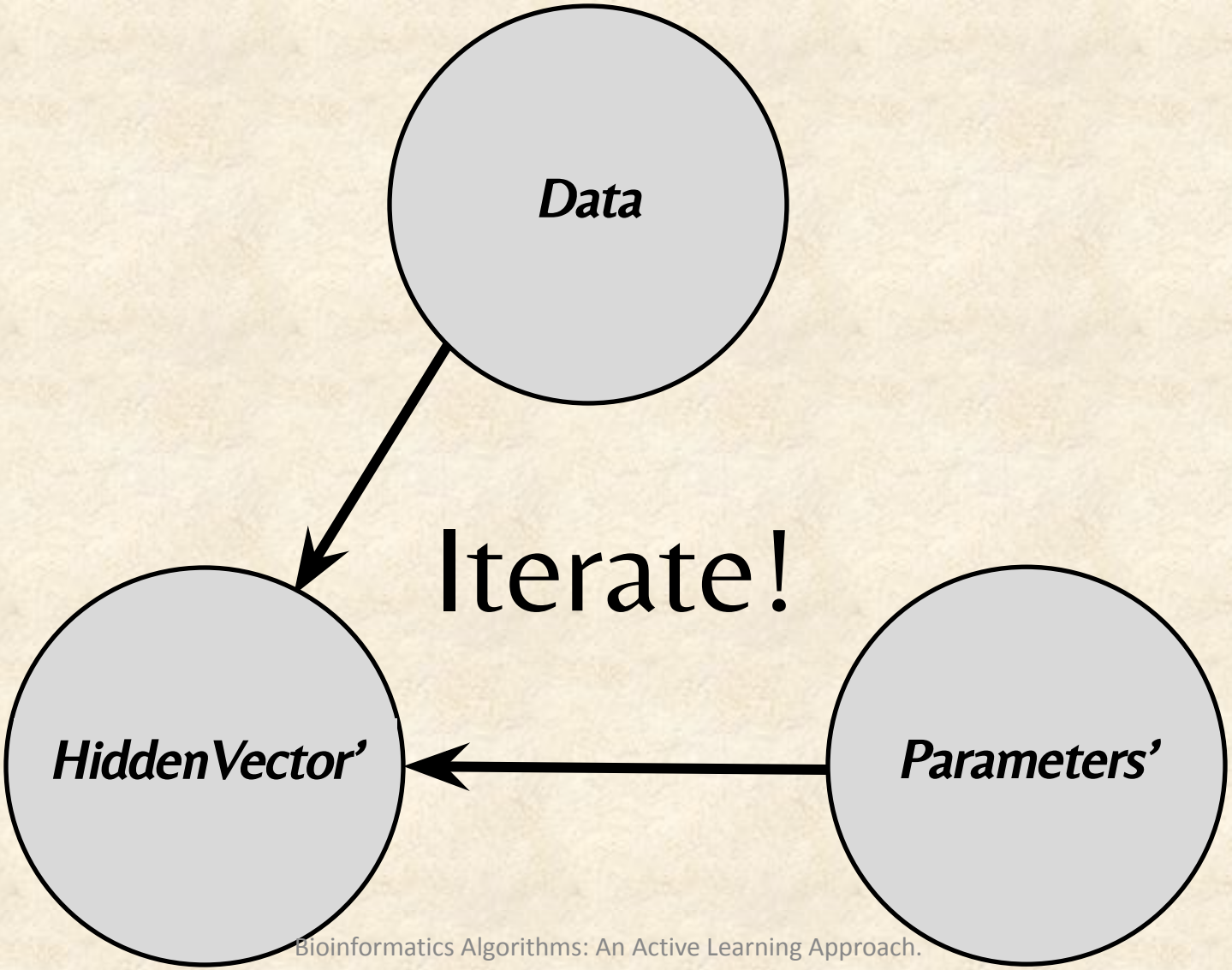
Reconstructing *HiddenVector* and *Parameters*



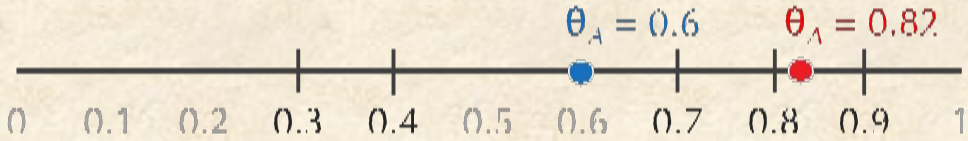
Reconstructing *HiddenVector* and *Parameters*



Reconstructing *HiddenVector* and *Parameters*

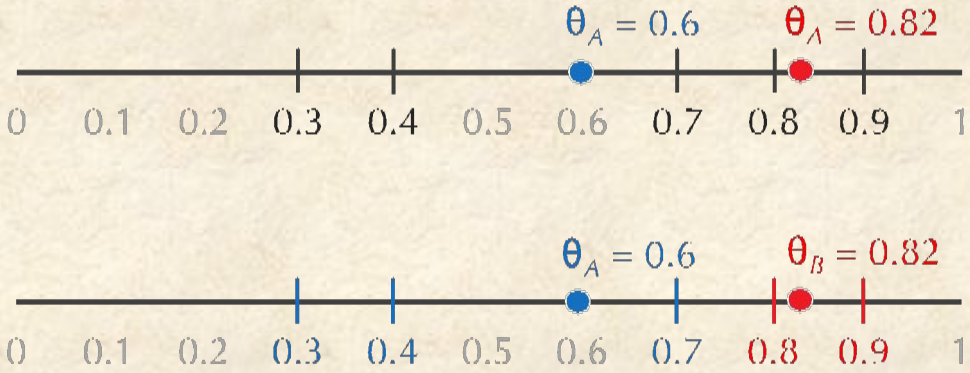


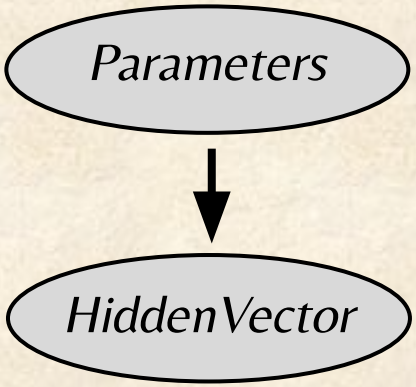
What does this algorithm remind you of?



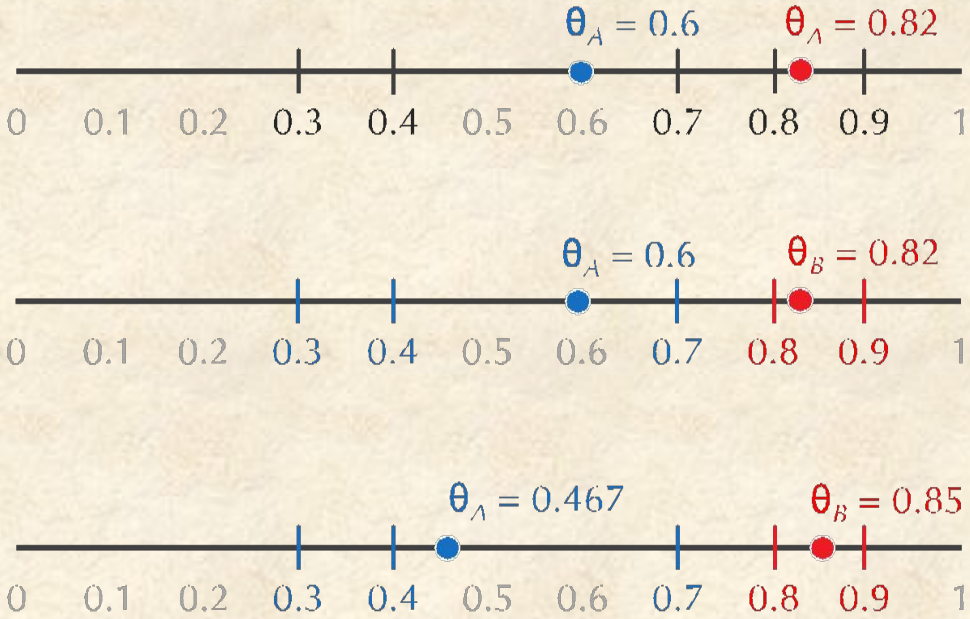
Parameters

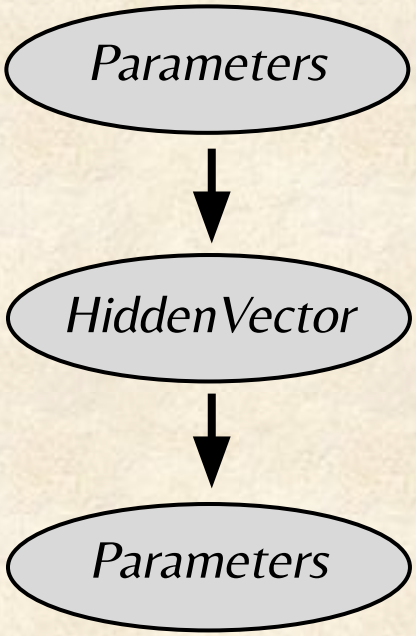
What does this algorithm remind you of?



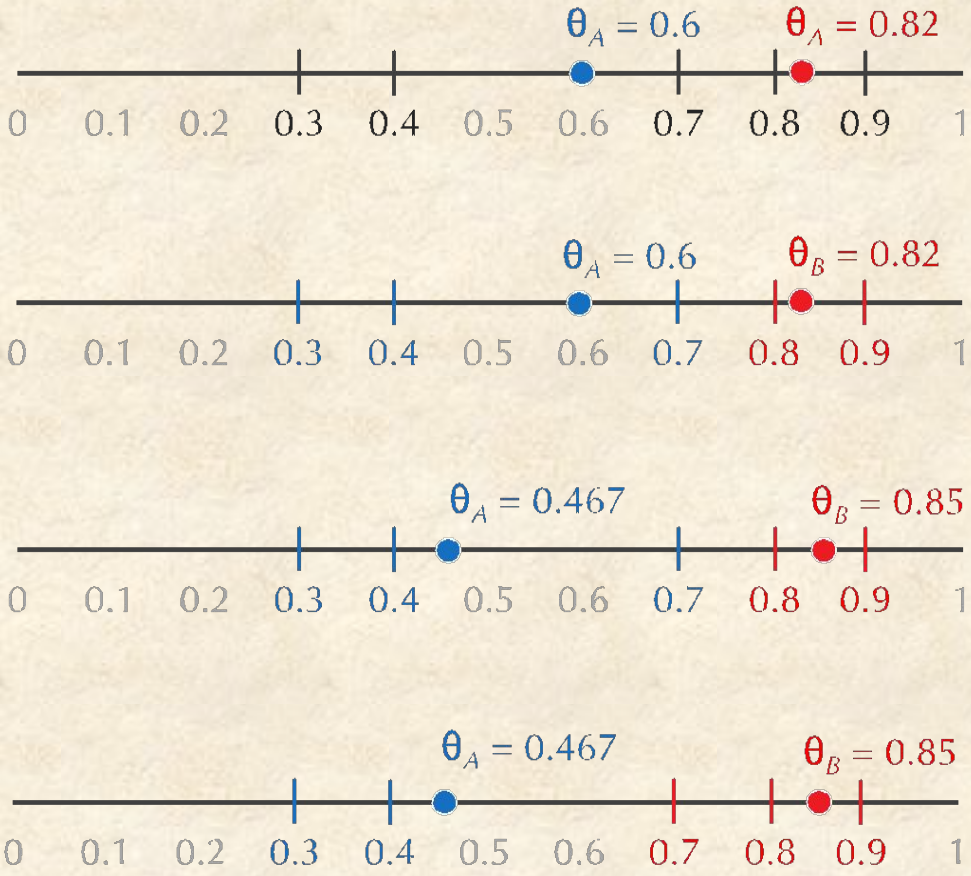


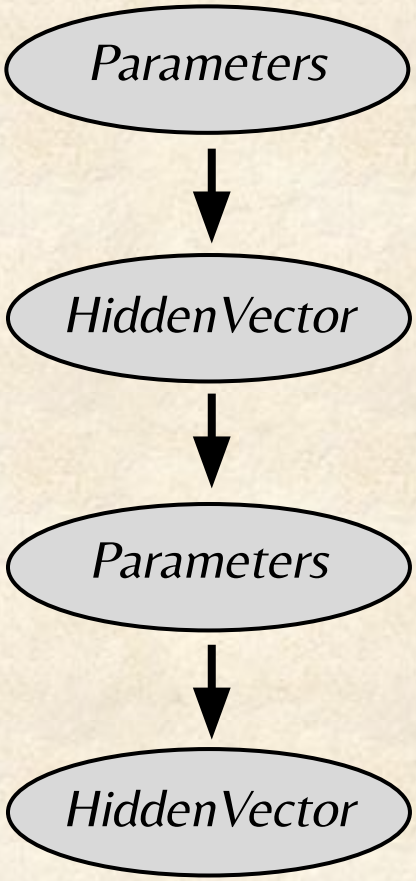
What does this algorithm remind you of?



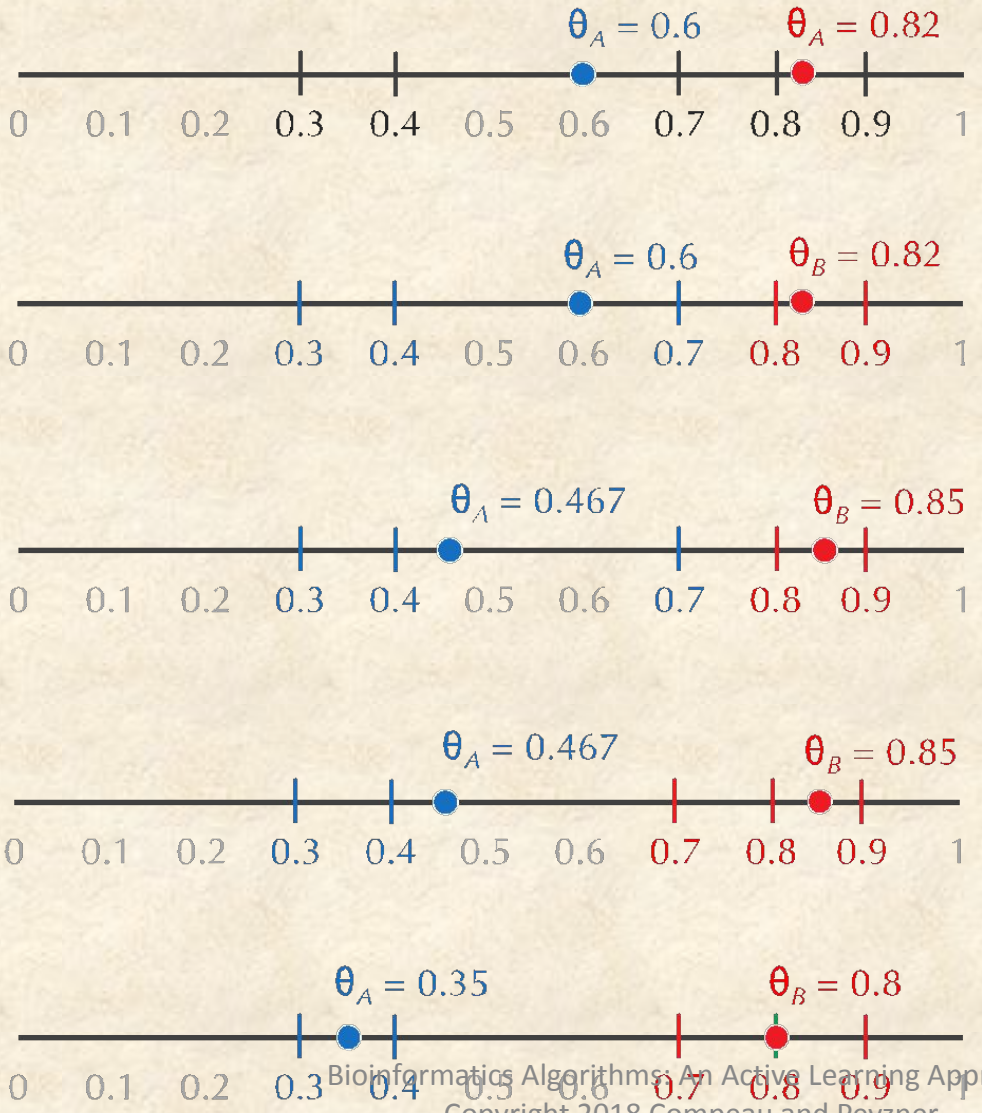


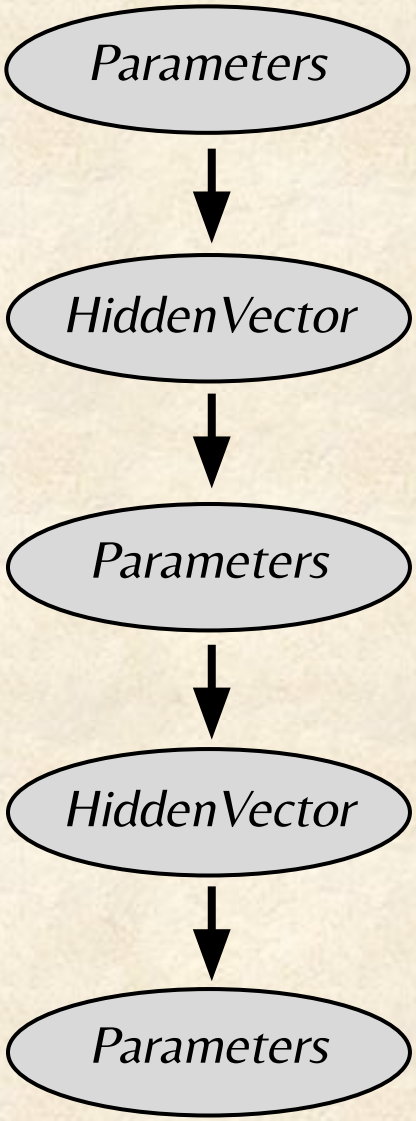
What does this algorithm remind you of?

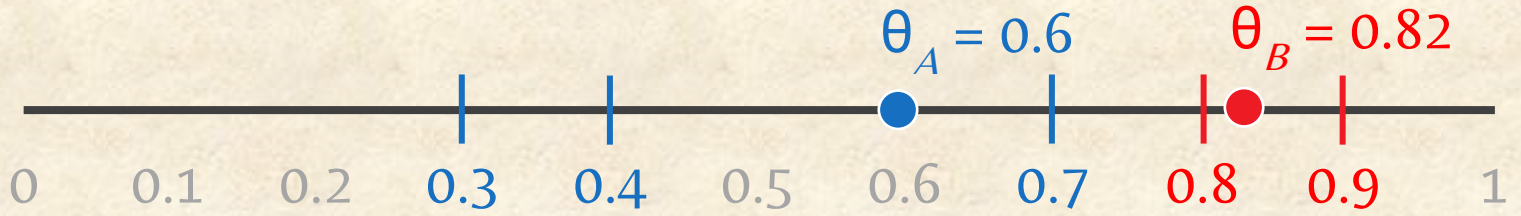


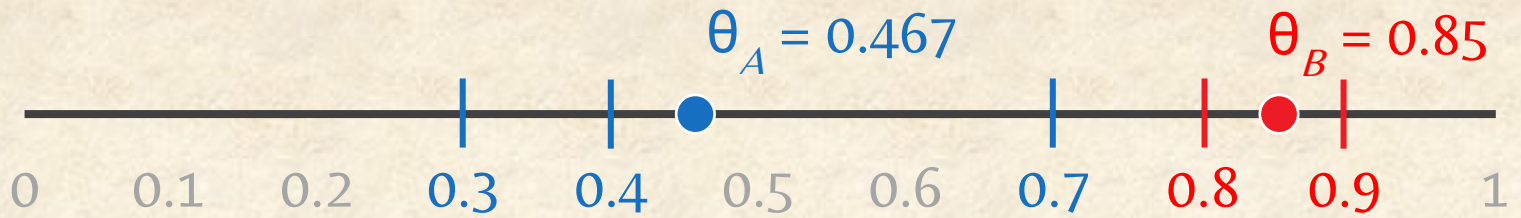


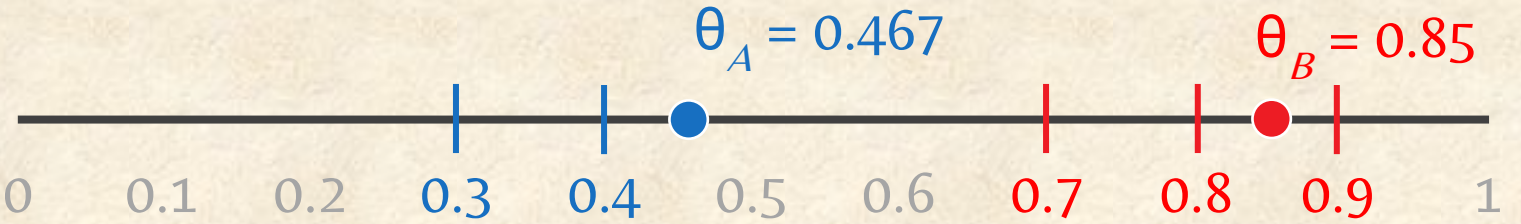
What does this algorithm remind you of?









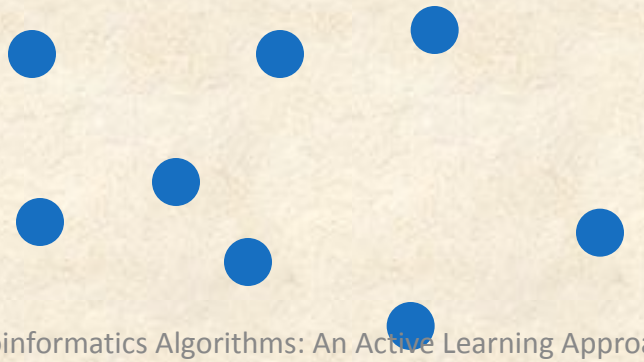


θA = 0.35 θB = 0.8

From Coin Flipping to *k*-Means Clustering: Where Are *Data, HiddenVector,* and *Parameters*?

Data: data points $Data = (Data_1,...,Data_n)$

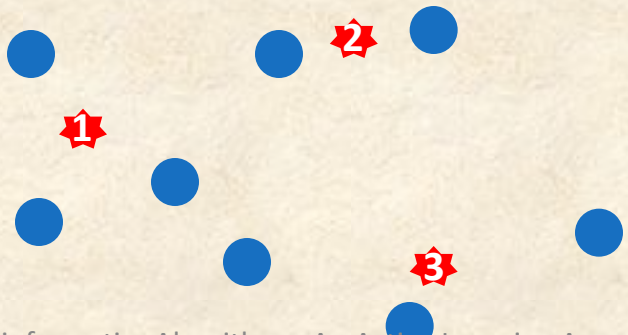
Parameters: $Centers = (Center_1,...,Center_k)$



From Coin Flipping to k-means Clustering: Where Are *Data*, *HiddenVector*, and *Parameters*?

Data: data points $Data = (Data_1,...,Data_n)$

Parameters: $Centers = (Center_1,...,Center_k)$

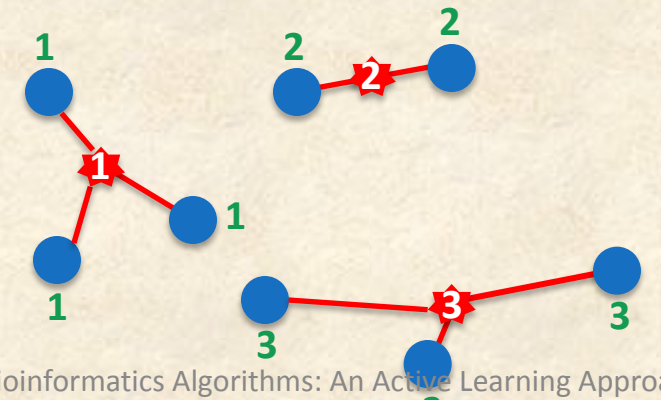


From Coin Flipping to k-means Clustering: Where Are *Data*, *HiddenVector*, and *Parameters*?

Data: data points $Data = (Data_1,...,Data_n)$

Parameters: $Centers = (Center_1,...,Center_k)$

HiddenVector: assignments of data points to k centers (n-dimensional vector with coordinates varying from 1 to k).



How Did Yeast Become a Wine Maker?

- Which Yeast Genes Are Responsible for Wine Brewing?
- Clustering as an optimization problem
- The Lloyd algorithm for *k*-means clustering
- From Hard to Soft Clustering
- From Coin Flipping to *k*-means Clustering
- Expectation Maximization
- Soft *k*-means Clustering
- Hierarchical Clustering

Coin Flipping and Soft Clustering

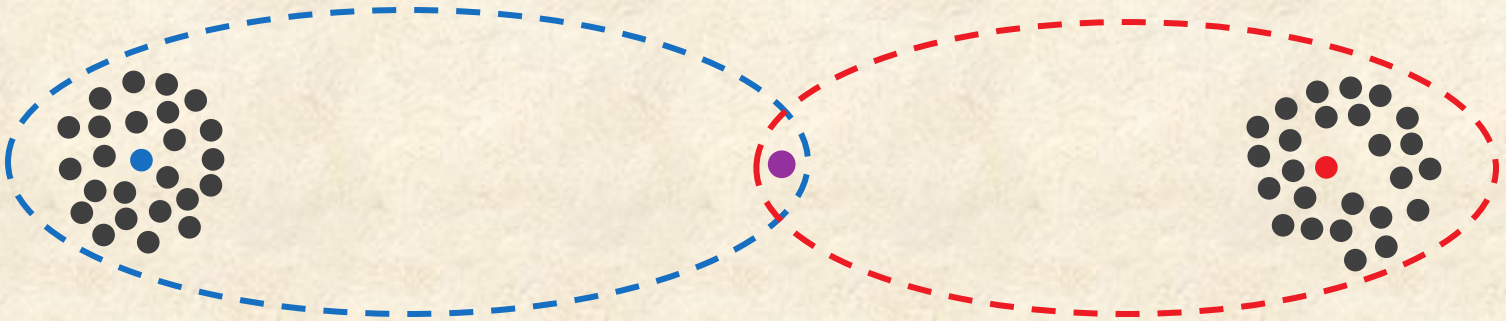
- **Coin flipping**: how would you select between coins A and B if $\Pr(\text{sequence}|\theta_A) = \Pr(\text{sequence}|\theta_B)$?

- ***k*-means clustering:** what cluster would you assign a data point it to if it is a midpoint of centers C_1 and C_2?

Coin Flipping and Soft Clustering

- **Coin flipping**: how would you select between coins A and B if $\Pr(\text{sequence}|\theta_A) = \Pr(\text{sequence}|\theta_B)$?

- ***k*-means clustering:** what cluster would you assign a data point it to if it is a midpoint of centers C_1 and C_2?



Soft assignments: assigning C_1 and C_2 "responsibility" =0.5 for a midpoint.

Memory Flash:
From *Data & Parameters to HiddenVector*

Data	***HiddenVector***	***Parameters =*** (θ_A, θ_B)
0.4	?	
0.9	?	
0.8	?	← (0.60, 0.82)
0.3	?	
0.7	?	

STOP and Think: Which coin is more likely to have generated the first sequence (with 4 H)?

$\Pr(1^{st}\text{ sequence}|\theta_A) = \theta_A^5 (1-\theta_A)^5 = 0.60^4 \bullet 0.40^6 \simeq 0.000531 >$
$\Pr(1^{st}\text{ sequence}|\theta_B) = \theta_B^5 (1-\theta_B)^5 = 0.82^4 \bullet 0.18^6 \simeq 0.000015$

Memory Flash:
From *Data & Parameters to HiddenVector*

Data	***HiddenVector***	***Parameters =*** (θ_A, θ_B)
0.4	1	
0.9	?	
0.8	?	← (0.60, 0.82)
0.3	?	
0.7	?	

STOP and Think: Which coin is more likely to have generated the first sequence (with 4 H)?

$\Pr(1^{st}\text{ sequence}|\theta_A) = \theta_A^5 (1-\theta_A)^5 = 0.60^4 \bullet 0.40^6 \simeq 0.000531 >$
$\Pr(1^{st}\text{ sequence}|\theta_B) = \theta_B^5 (1-\theta_B)^5 = 0.82^4 \bullet 0.18^6 \simeq 0.000015$

From *Data & Parameters* to ***HiddenMatrix***

Data	***HiddenMatrix***	***Parameters =*** (θ_A, θ_B)
0.4	0.97 0.03	
0.9	?	
0.8	?	← (0.60, 0.82)
0.3	?	
0.7	?	

What are the **responsibilities** of coins for this sequence?

$\Pr(1^{st}\ \text{sequence}|\theta_A) \simeq 0.000531 >$
$\Pr(1^{st}\ \text{sequence}|\theta_B) \simeq 0.000015$

$0.000531 / (0.000531 + 0.000015) \simeq 0.97$
$0.000015 / (0.000531 + 0.000015) \simeq 0.03$

From *Data & Parameters* to ***HiddenMatrix***

Data	***HiddenMatrix***	***Parameters =*** (θ_A, θ_B)
0.4	0.97 0.03	
0.9	0.12 0.88	
0.8	?	← (0.60, 0.82)
0.3	?	
0.7	?	

What are the responsibilities of coins for the 2nd sequence?

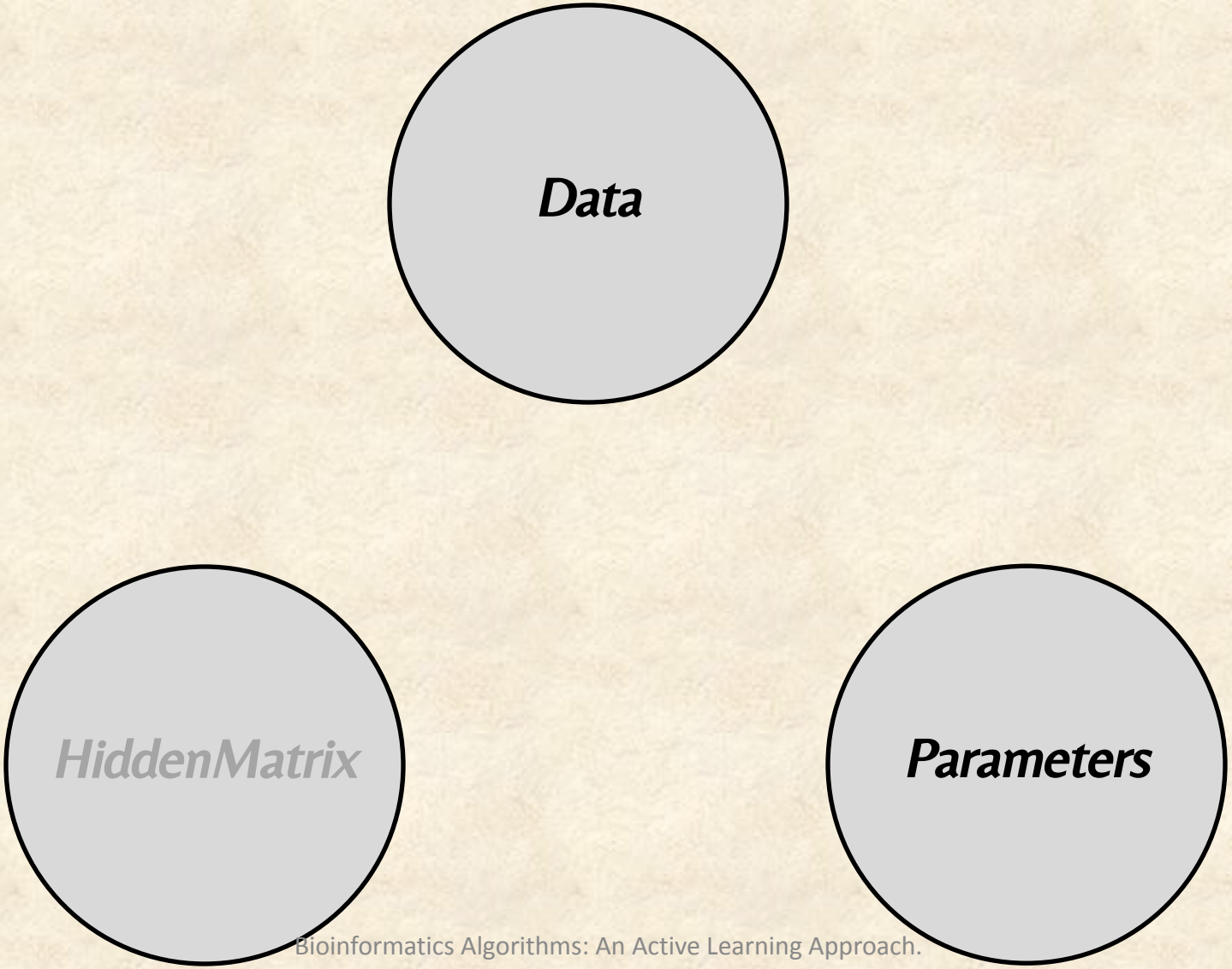
$\Pr(2^{nd}\text{ sequence}|\theta_A) \propto 0.0040 <$
$\Pr(2^{nd}\text{ sequence}|\theta_B) \propto 0.0302$

$0.0040 / (0.0040 + 0.0302) = 0.12$
$0.0342 / (0.0040 + 0.0342) = 0.88$

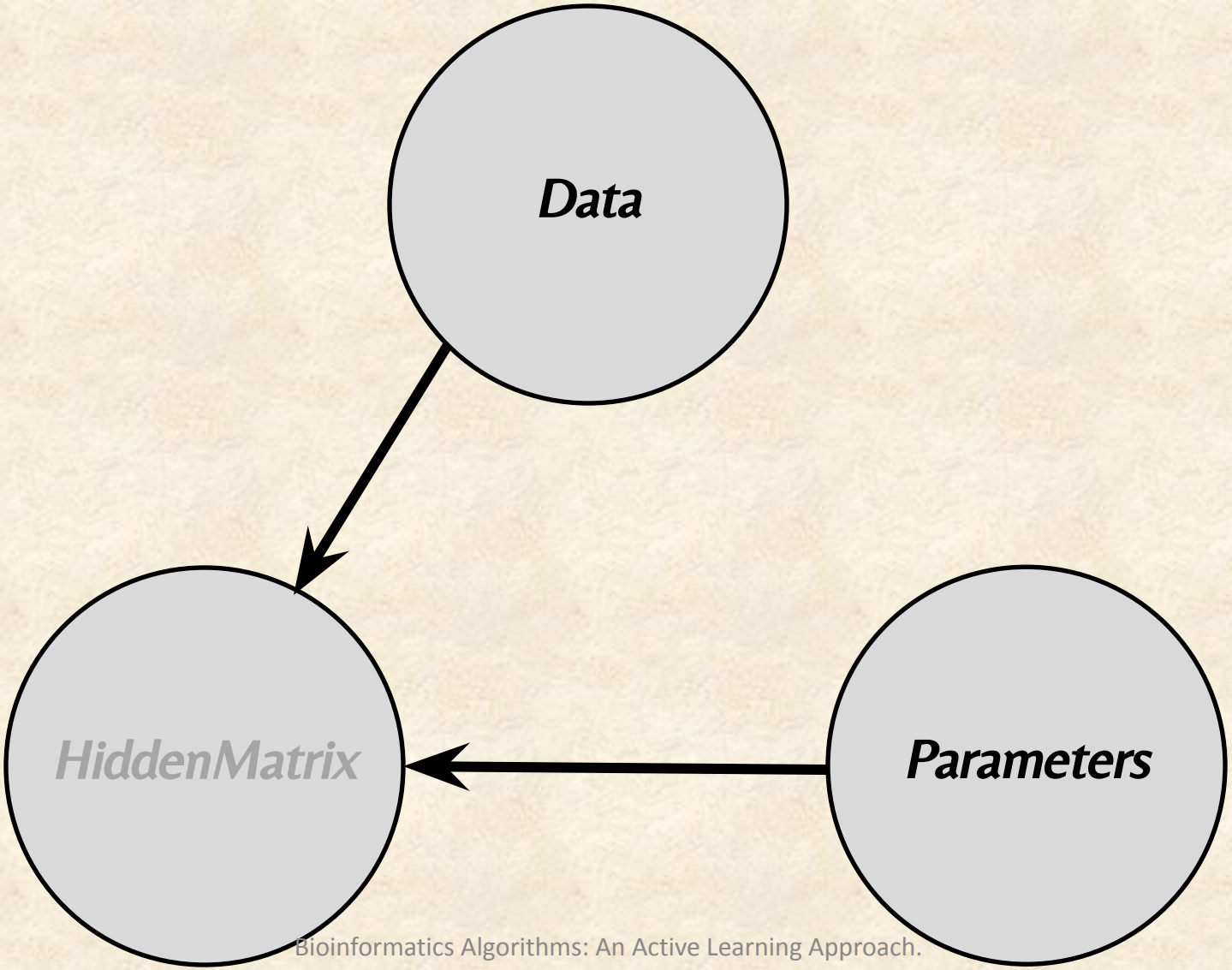
HiddenMatrix Reconstructed!

Data	*HiddenMatrix*		*Parameters* = (θ_A, θ_B)
0.4	0.97	0.03	
0.9	0.12	0.88	
0.8	0.29	0.71	← (0.60, 0.82)
0.3	0.99	0.01	
0.7	0.55	0.45	

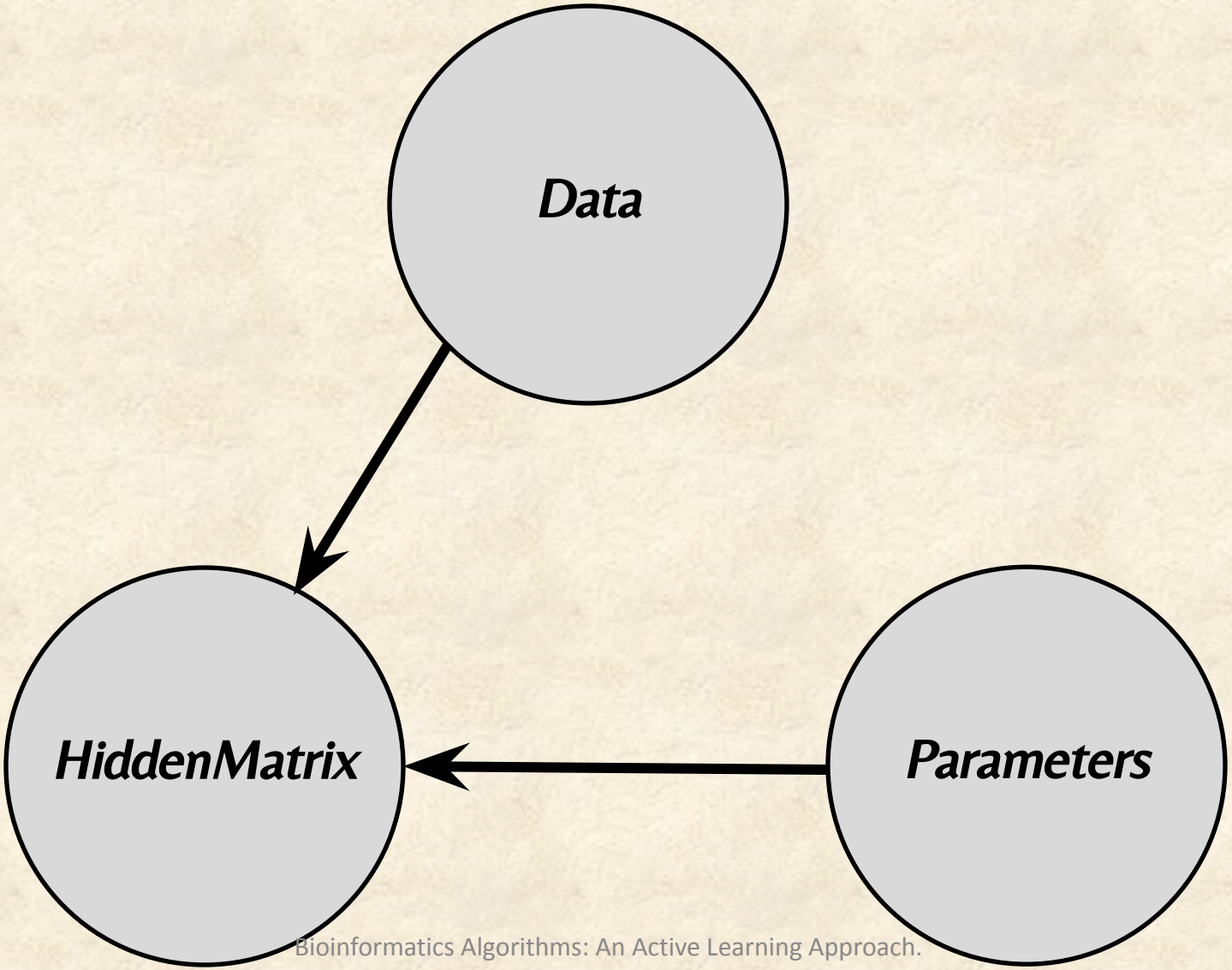
Expectation Maximization Algorithm



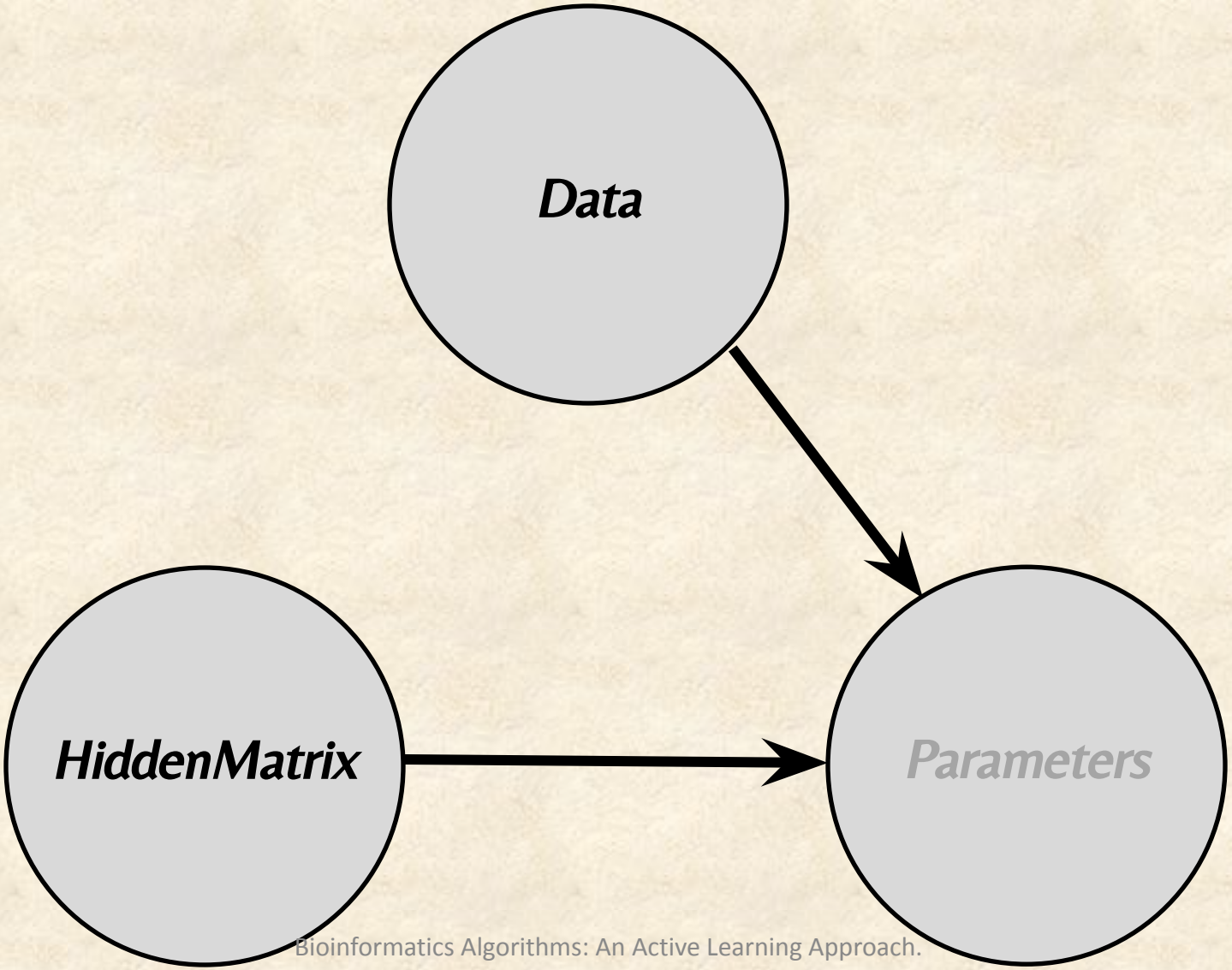
E-step



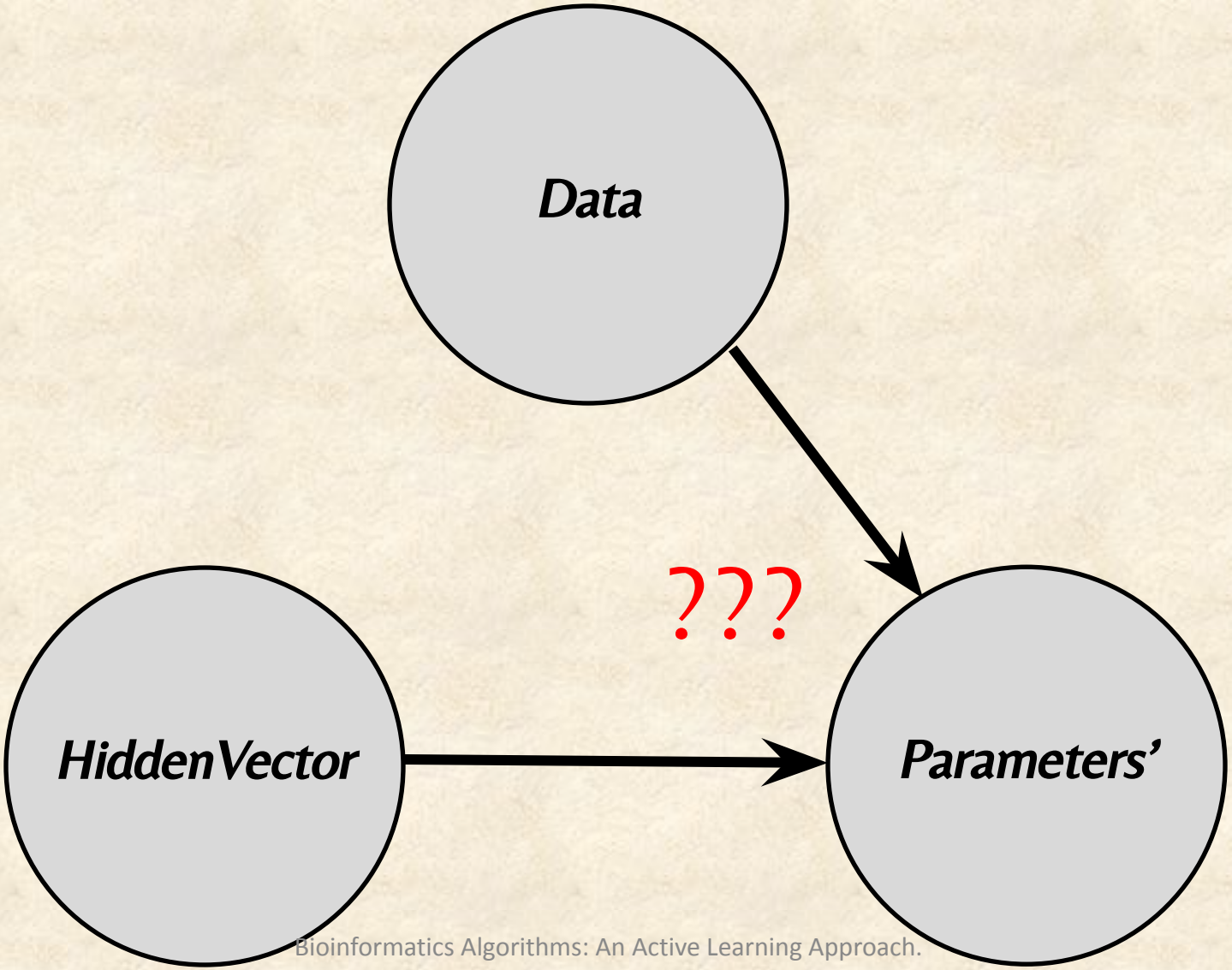
E-step



M-step



M-step



Memory Flash: Dot Product

Data		HiddenVector	Parameters=(θ_A, θ_B)
HTTTHTTHTH	0.4	* 1	
HHHHTHHHHH	0.9	* 0	
HTHHHHHTHH	0.8	* 0	???
HTTTTTHHTT	0.3	* 1	
THHHTHHHTH	0.7	* 0	

θ_A = *Data* * *HiddenVector* / **1** * *HiddenVector*

θ_B = *Data* * (**1**-*HiddenVector*) / **1** * (**1**-*HiddenVector*)

From *Data* & *HiddenMatrix* to ***Parameters***

Data	HiddenVector	Parameters=(θ_A,θ_B)
HTTTHTTHTH	0.4	1
HHHHTHHHHH	0.9	0
HTHHHHHTHH	0.8	0
HTTTTTHHTT	0.3	1
THHHTHHHTH	0.7	0

θ_A = *Data* * *HiddenVector* / **1** * *HiddenVector*

θ_B = *Data* * (**1**-*HiddenVector*) / **1** * (**1**-*HiddenVector*)

HiddenVector = (1 0 0 1 0)

STOP and Think: What is *HiddenMatrix* corresponding to this *HIddenVector*?

From *Data* & *HiddenMatrix* to ***Parameters***

	Data	***HiddenVector***	***Parameters*=(θ_A,θ_B)**
HTTTHTTHTH	**0.4**	**1**	
HHHHTHHHHH	**0.9**	**0**	
HTHHHHHTHH	**0.8**	**0**	
HTTTTTHHTT	**0.3**	**1**	
THHHTHHHTH	**0.7**	**0**	

θ_A = *Data* * *HiddenVector* / **1** * *HiddenVector*

θ_A = *Data* * 1st row of *HiddenMatrix* / **1***1st row of *HiddenMatrix*

θ_B = *Data* * (**1**-*HiddenVector*) / **1** * (**1**-*HiddenVector*)

θ_B = *Data* * 2nd row of *HiddenMatrix* / **1***2nd row of *HiddenMatrix*

HiddenVector = (1 0 0 1 0)

Hidden Matrix = 1 0 0 1 0 = *HiddenVector*
0 1 1 0 1 = **1** - *HiddenVector*

From *Data* & *HiddenMatrix* to ***Parameters***

	Data	***HiddenMatrix***	***Parameters*=(θ_A,θ_B)**
HTTT**H**TT**H**T**H**	0.4	0.97	0.03
HHHHT**HHHHH**	0.9	0.12	0.88
HT**HHHHH**T**HH**	0.8	0.29	0.71
HTTTTT**HH**TT	0.3	0.99	0.01
T**HHH**T**HHH**T**H**	0.7	0.55	0.45

θ_A = *Data* * *HiddenVector* / **1** * *HiddenVector*

θ_A = *Data* * 1st row of *HiddenMatrix* / **1***1st row of *HiddenMatrix*

θ_B = *Data* * (**1**-*HiddenVector*) / **1** * (**1**-*HiddenVector*)

θ_B = *Data* * 2nd row of *HiddenMatrix* / **1***2nd row of *HiddenMatrix*

HiddenVector = (1 0 0 1 0)

Hidden Matrix = .97 .03 .29 .99 .55
.03 .97 .71 .01 .45

How Did Yeast Become a Wine Maker?

- Which Yeast Genes Are Responsible for Wine Brewing?
- Clustering as an optimization problem
- The Lloyd algorithm for *k*-means clustering
- From Hard to Soft Clustering
- From Coin Flipping to *k*-means Clustering
- Expectation Maximization
- Soft *k*-means Clustering
- Hierarchical Clustering

From *HiddenVector* to *HiddenMatrix*

Data: data points $Data = \{Data_1, \ldots, Data_n\}$

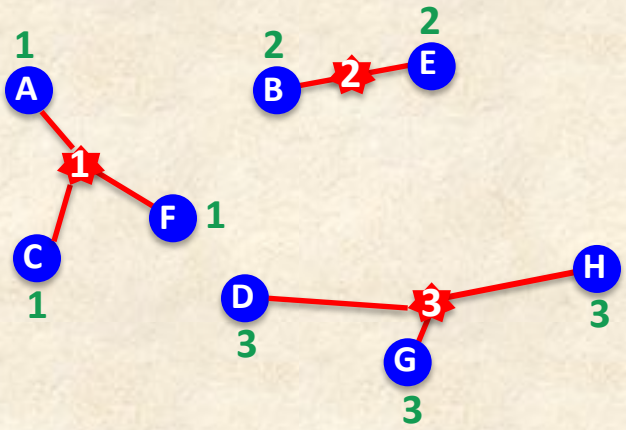
Parameters: $Centers = \{Center_1, \ldots, Center_k\}$

HiddenVector: assignments of data points to centers

	A	B	C	D	E	F	G	H
HiddenVector	1	2	1	3	2	1	3	3

1	1	0	1	0	0	1	0	0
2	0	**1**	0	0	**1**	0	0	0
3	0	0	0	**1**	0	0	**1**	**1**

HiddenMatrix



From *HiddenVector* to *HiddenMatrix*

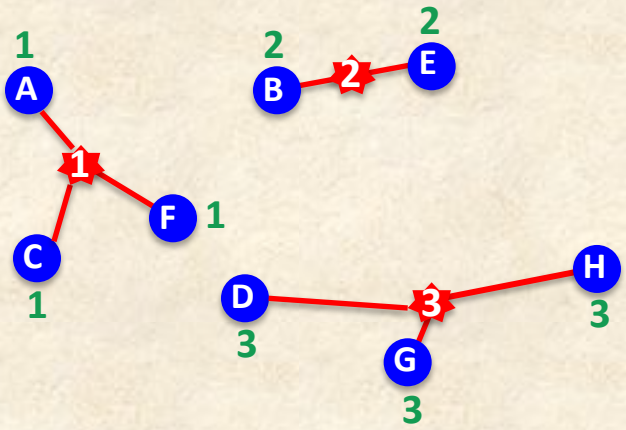
Data: data points $Data = \{Data_1, \ldots, Data_n\}$

Parameters: $Centers = \{Center_1, \ldots, Center_k\}$

$HiddenMatrix_{i,j}$: responsibility of center *i* for data point *j*

HiddenMatrix

	A	B	C	D	E	F	G	H
1	0.7	0	1	0	0	1	0	0
2	0.2	1	0	0	1	0	0	0
3	0.1	0	0	1	0	0	1	1



From *HiddenVector* to *HiddenMatrix*

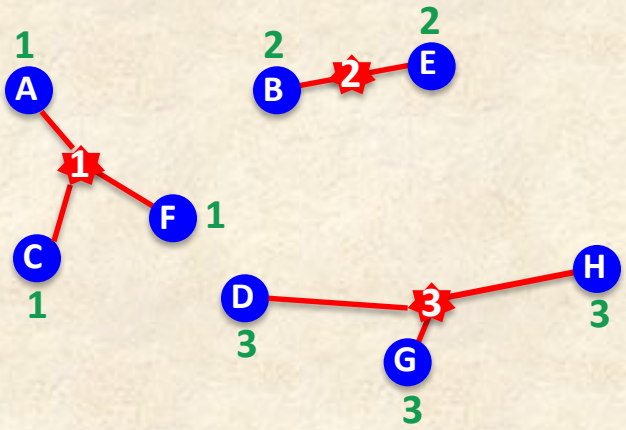
Data: data points $Data = \{Data_1, \ldots, Data_n\}$

Parameters: $Centers = \{Center_1, \ldots, Center_k\}$

$HiddenMatrix_{i,j}$: responsibility of center *i* for data point *j*

HiddenMatrix

	A	B	C	D	E	F	G	H
1	0.70	0.15	0.73	0.40	0.15	0.80	0.05	0.05
2	0.20	0.80	0.17	0.20	0.80	0.10	0.05	0.20
3	0.10	0.05	0.10	0.40	0.05	0.10	0.90	0.75



Responsibilities and the Law of Gravitation

planets

stars

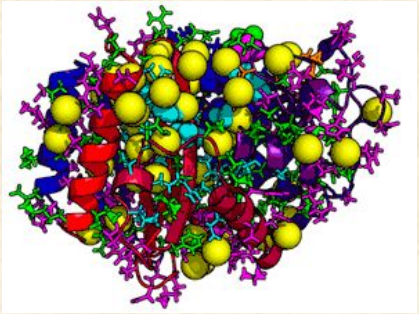
0.70	0.15	0.73	0.40	0.15	0.80	0.05	0.05
0.20	0.80	0.17	0.20	0.80	0.10	0.05	0.20
0.10	0.05	0.10	0.40	0.05	0.10	0.90	0.75

responsibility of star *i* for a planet *j* is proportional to the pull (Newtonian law of gravitation):

$$Force_{i,j} = 1/distance(Data_j, Center_i)^2$$

$$HiddenMatrix_{ij} := Force_{i,j} / \sum_{\text{all centers } j} Force_{i,j}$$

Responsibilities and Statistical Mechanics



data points

centers

0.70	0.15	0.73	0.40	0.15	0.80	0.05	0.05
0.20	0.80	0.17	0.20	0.80	0.10	0.05	0.20
0.10	0.05	0.10	0.40	0.05	0.10	0.90	0.75

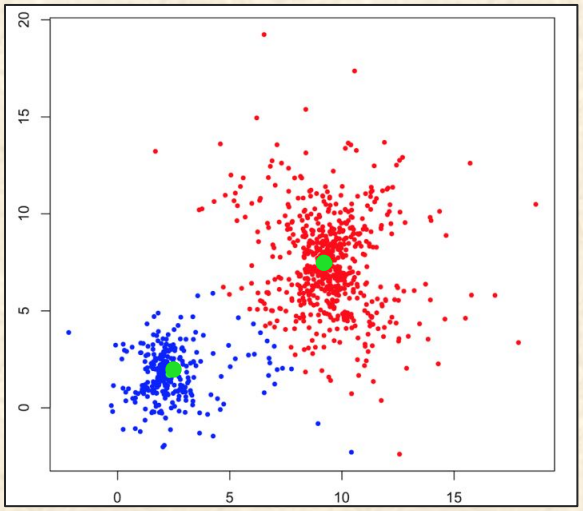
responsibility of center *i* for a data point *j* is proportional to

$$Force_{i,j} = e^{-\beta \cdot \text{distance}(Data_j,\ Center_i)}$$

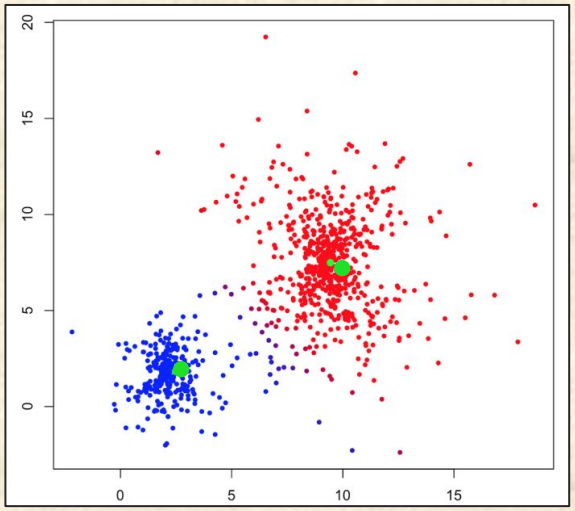
where β is a **stiffness parameter**.

$$HiddenMatrix_{ij} := Force_{i,j} \,/\, \sum_{\text{all centers } j} Force_{i,j}$$

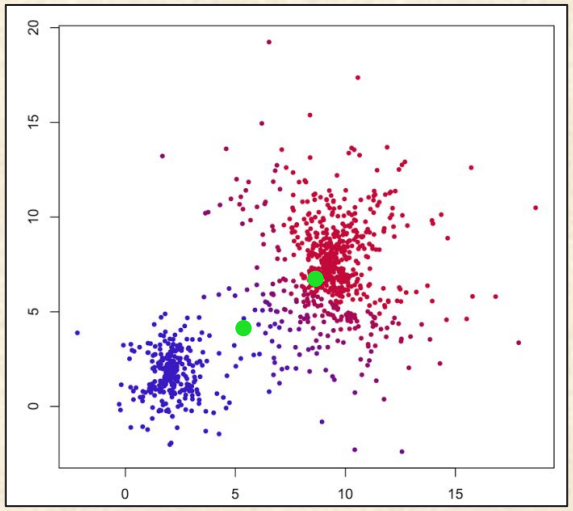
How Does Stiffness Affect Clustering?



Hard *k*-means clustering



Soft *k*-means clustering (stiffness β=1)



Soft *k*-means clustering (stiffness β= 0.3)

How Did Yeast Become a Wine Maker?

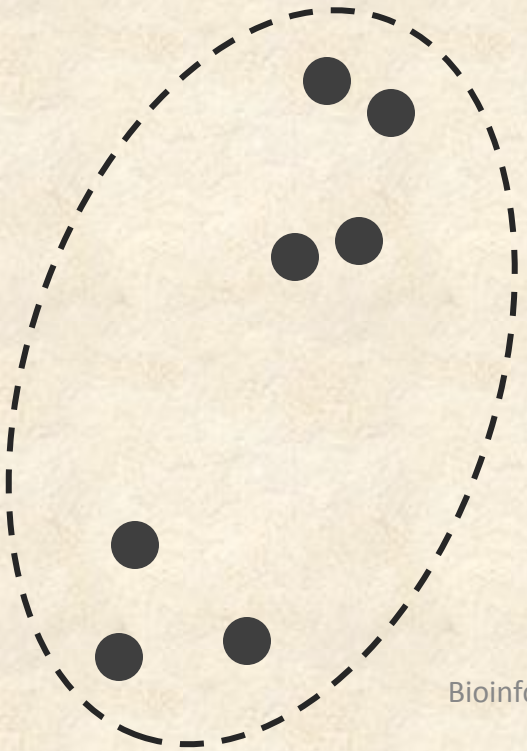
- What Yeast Genes Are Responsible for Wine Brewing?
- Clustering as an optimization problem
- The Lloyd algorithm for *k*-means clustering
- From Hard to Soft Clustering
- From Coin Flipping to *k*-means Clustering
- Expectation Maximization
- Soft *k*-means Clustering
- **Hierarchical Clustering**

Stratification of Clusters

Clusters often have **subclusters**, which have subsubclusters, and so on.

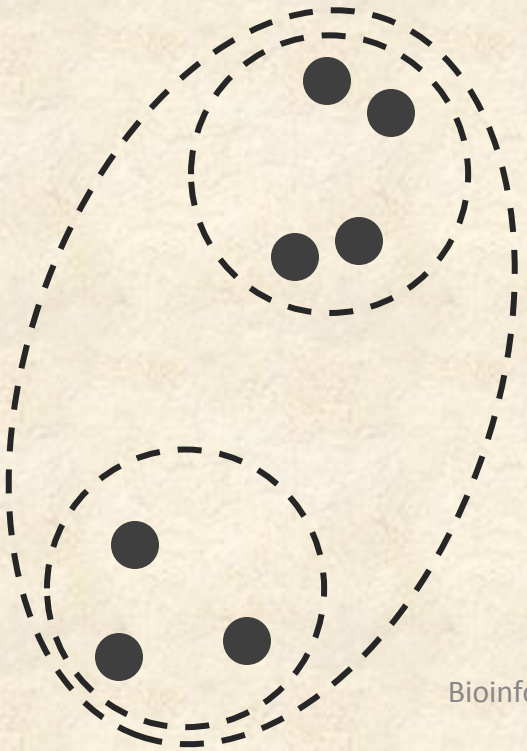
Stratification of Clusters

Clusters often have **subclusters**, which have subsubclusters, and so on.



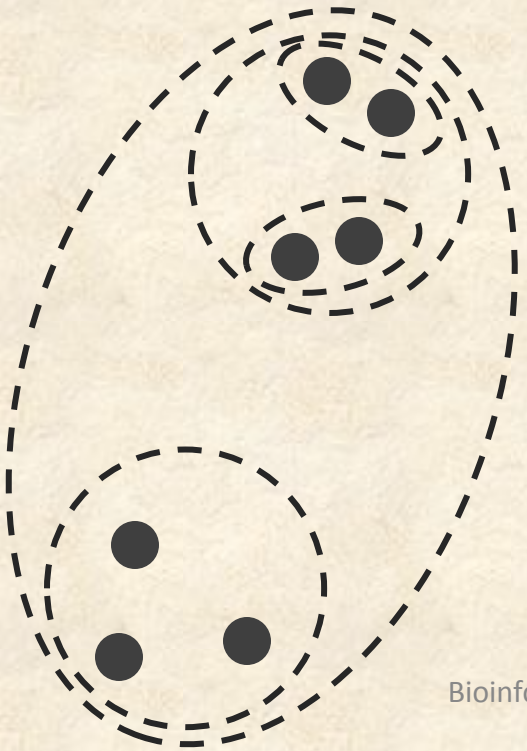
Stratification of Clusters

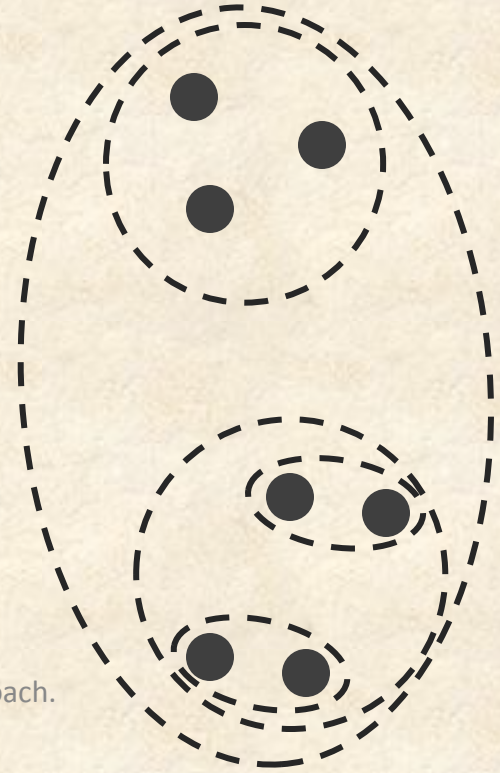
Clusters often have **subclusters**, which have subsubclusters, and so on.



Stratification of Clusters

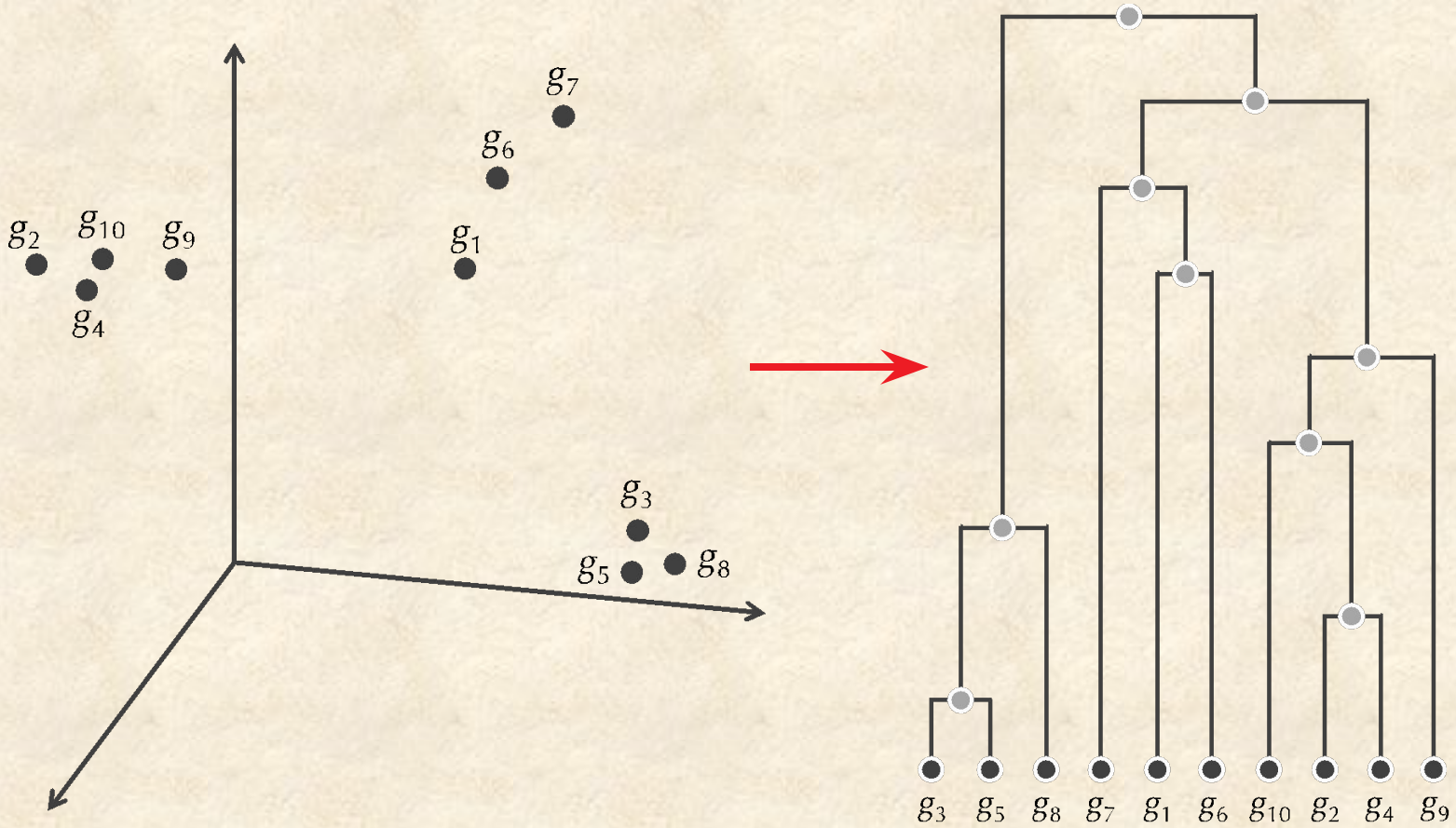
Clusters often have **subclusters**, which have sub-subclusters, and so on.





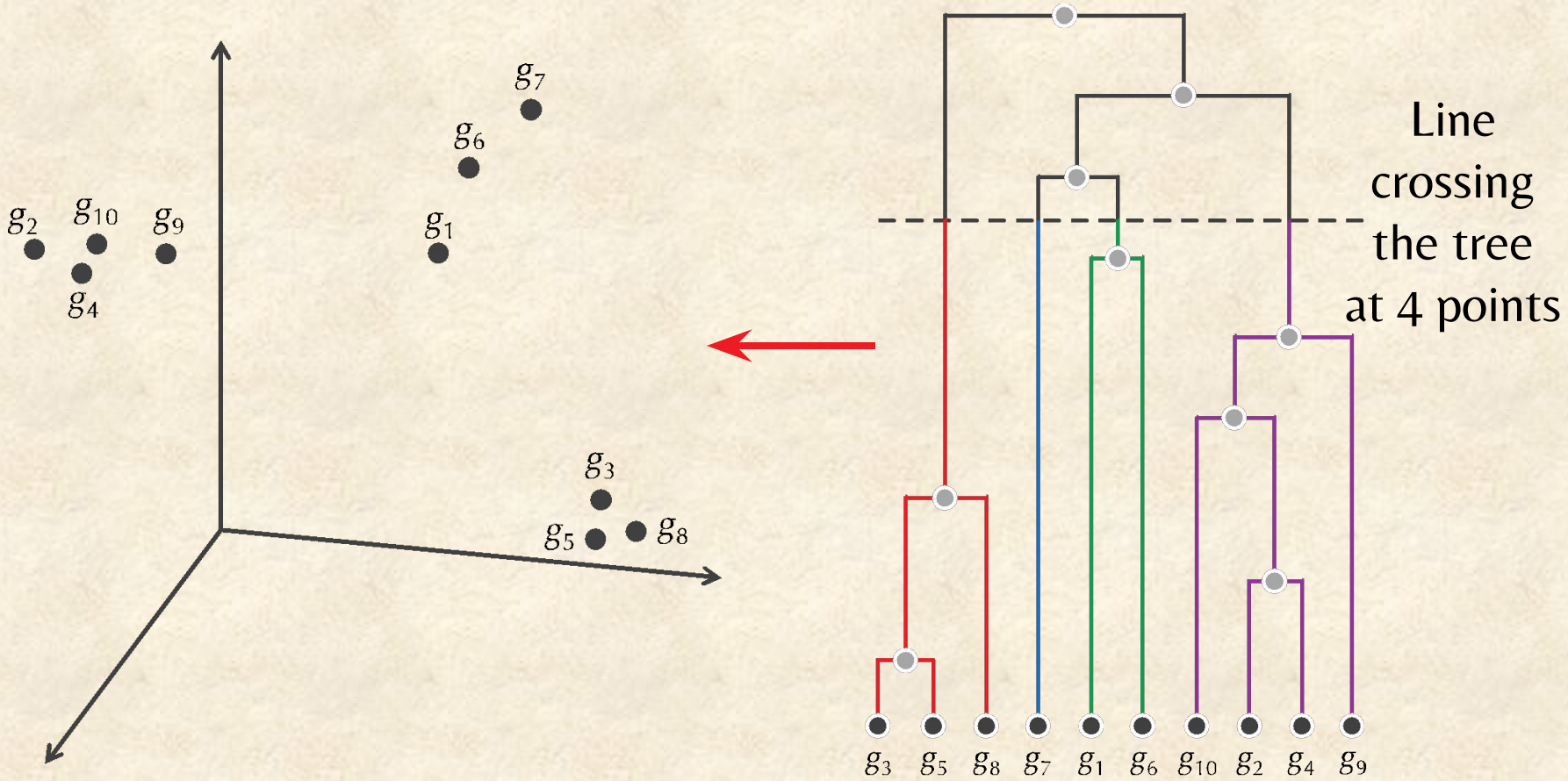
From Data to a Tree

To capture stratification, the **hierarchical clustering** algorithm organizes *n* data points into a tree.



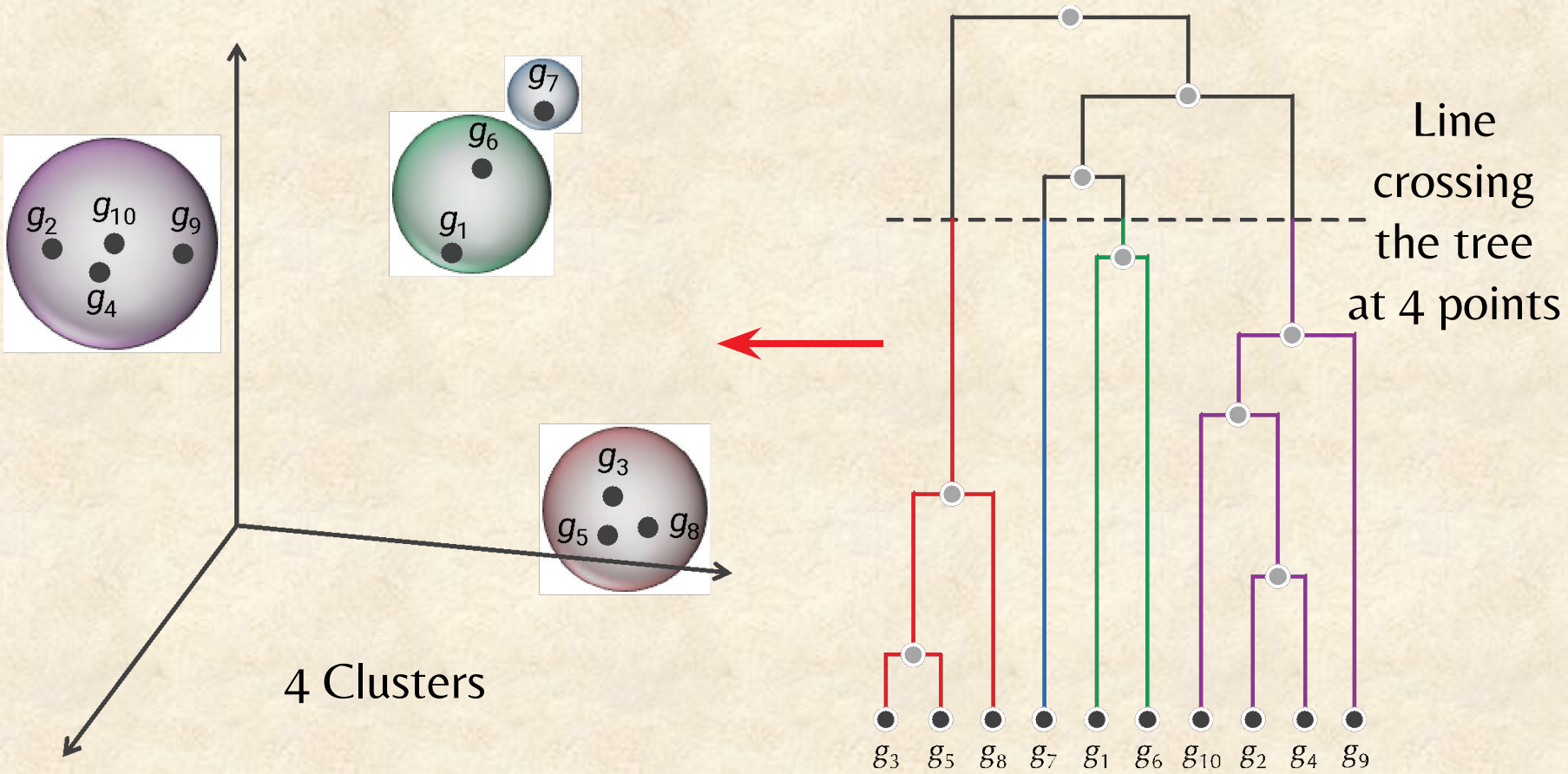
From a Tree to a Partition into 4 Clusters

To capture stratification, the **hierarchical clustering** algorithm organizes *n* data points into a tree.



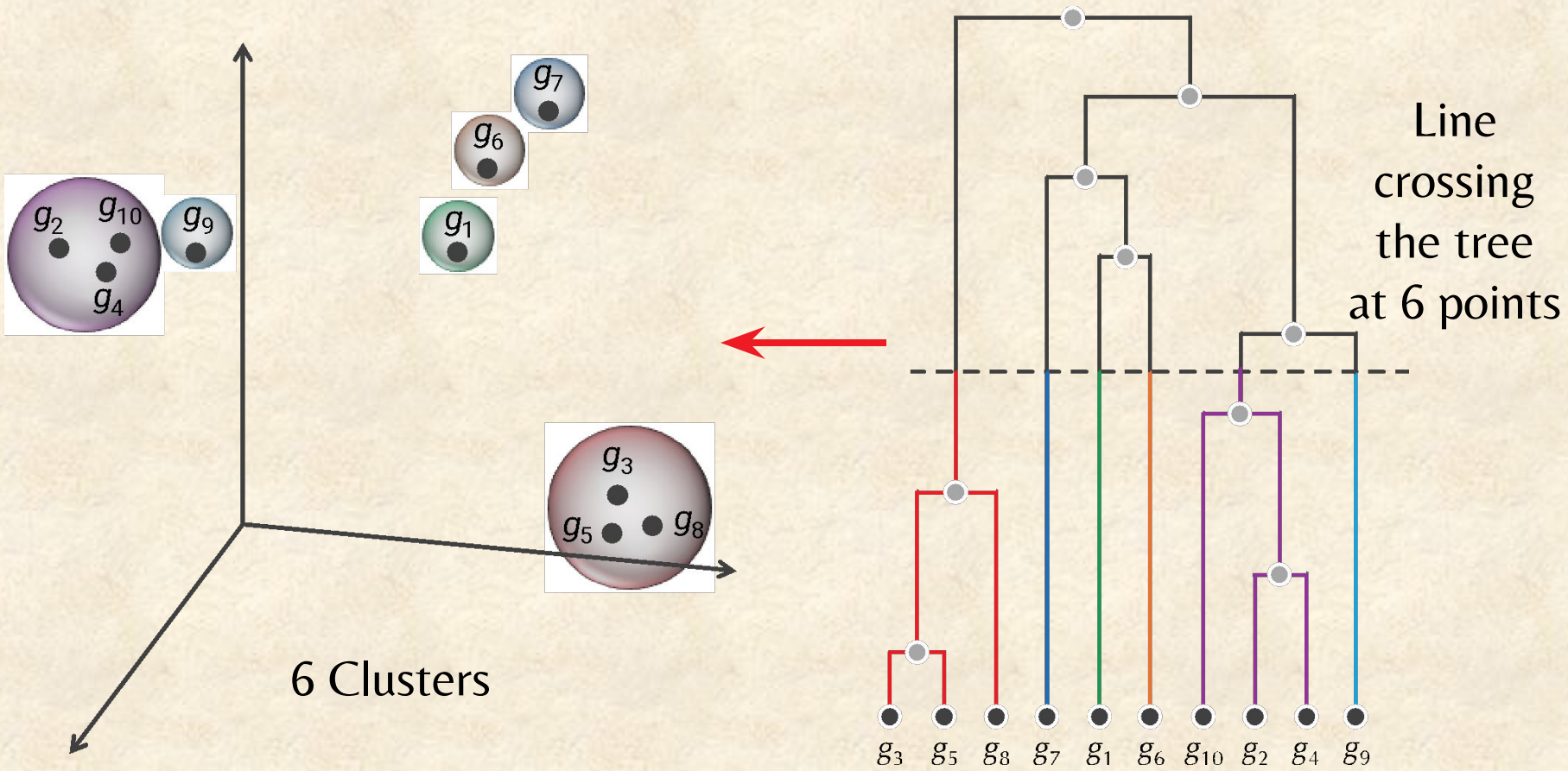
From a Tree to a Partition into 4 Clusters

To capture stratification, the **hierarchical clustering** algorithm first organizes *n* data points into a tree.



From a Tree to a Partition into 6 Clusters

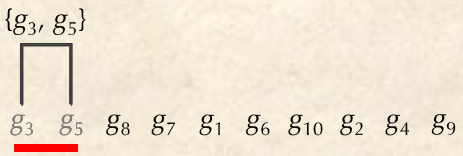
To capture stratification, the **hierarchical clustering** algorithm first organizes *n* data points into a tree.



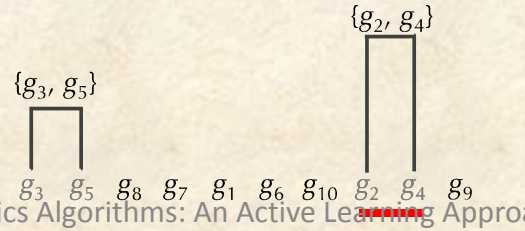
Constructing the Tree

- Hierarchical clustering starts from a transformation of *n* x *m* expression matrix into *n* x *n* **similarity matrix** or **distance matrix.**
- E.g., the similarity between expression vectors can be defined as their dot product or **Pearson correlation** coefficient.

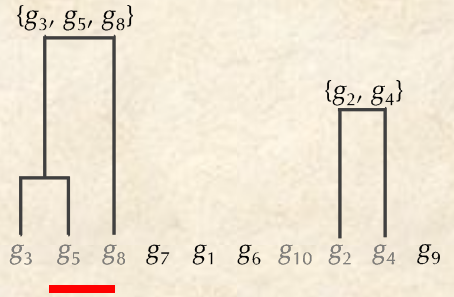
	g_1	g_2	g_3	g_4	g_5	g_6	g_7	g_8	g_9	g_{10}
g_1	0.0	8.1	9.2	7.7	9.3	2.3	5.1	10.2	6.1	7.0
g_2	8.1	0.0	12.0	0.9	12.0	9.5	10.1	12.8	2.0	1.0
g_3	9.2	12.0	0.0	11.2	**0.7**	11.1	8.1	1.1	10.5	11.5
g_4	7.7	0.9	11.2	0.0	11.2	9.2	9.5	12.0	1.6	1.1
g_5	9.3	12.0	**0.7**	11.2	0.0	11.2	8.5	1.0	10.6	11.6
g_6	2.3	9.5	11.1	9.2	11.2	0.0	5.6	12.1	7.7	8.5
g_7	5.1	10.1	8.1	9.5	8.5	5.6	0.0	9.1	8.3	9.3
g_8	10.2	12.8	1.1	12.0	1.0	12.1	9.1	0.0	11.4	12.4
g_9	6.1	2.0	10.5	1.6	10.6	7.7	8.3	11.4	0.0	1.1
g_{10}	7.0	1.0	11.5	1.1	11.6	8.5	9.3	12.4	1.1	0.0



	g_1	g_2	g_3, g_5	g_4	g_6	g_7	g_8	g_9	g_{10}
g_1	0.0	8.1	9.2	7.7	2.3	5.1	10.2	6.1	7.0
g_2	8.1	0.0	12.0	**0.9**	9.5	10.1	12.8	2.0	1.0
g_3, g_5	9.2	12.0	0.0	11.2	11.1	8.1	1.0	10.5	11.5
g_4	7.7	**0.9**	11.2	0.0	9.2	9.5	12.0	1.6	1.1
g_6	2.3	9.5	11.1	9.2	0.0	5.6	12.1	7.7	8.5
g_7	5.1	10.1	8.1	9.5	5.6	0.0	9.1	8.3	9.3
g_8	10.2	12.8	1.0	12.0	12.1	9.1	0.0	11.4	12.4
g_9	6.1	2.0	10.5	1.6	7.7	8.3	11.4	0.0	1.1
g_{10}	7.0	1.0	11.5	1.1	8.5	9.3	12.4	1.1	0.0

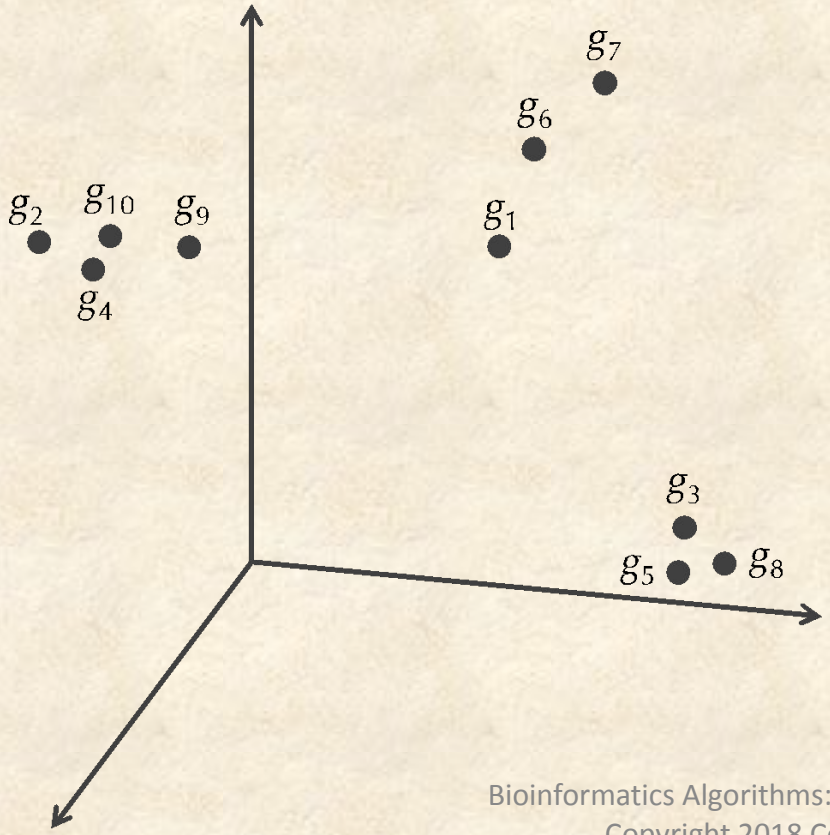


	g_1	g_2, g_4	g_3, g_5	g_6	g_7	g_8	g_9	g_{10}
g_1	0.0	7.7	9.2	2.3	5.1	10.2	6.1	7.0
g_2, g_4	7.7	0.0	11.2	9.2	9.5	12.0	1.6	1.0
g_3, g_5	9.2	11.2	0.0	11.1	8.1	**1.0**	10.5	11.5
g_6	2.3	9.2	11.1	0.0	5.6	12.1	7.7	8.5
g_7	5.1	9.5	8.1	5.6	0.0	9.1	8.3	9.3
g_8	10.2	12.0	**1.0**	12.1	9.1	0.0	11.4	12.4
g_9	6.1	1.6	10.5	7.7	8.3	11.4	0.0	1.1
g_{10}	7.0	1.0	11.5	8.5	9.3	12.4	1.1	0.0



Constructing the Tree

Hierarchical clustering starts from a transformation of *n* x *m* expression matrix into *n* x *n* **similarity matrix** or **distance matrix.**

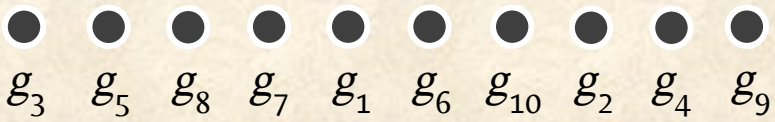


Distance Matrix

	g_1	g_2	g_3	g_4	g_5	g_6	g_7	g_8	g_9	g_{10}
g_1	0.0	8.1	9.2	7.7	9.3	2.3	5.1	10.2	6.1	7.0
g_2	8.1	0.0	12.0	0.9	12.0	9.5	10.1	12.8	2.0	1.0
g_3	9.2	12.0	0.0	11.2	0.7	11.1	8.1	1.1	10.5	11.5
g_4	7.7	0.9	11.2	0.0	11.2	9.2	9.5	12.0	1.6	1.1
g_5	9.3	12.0	0.7	11.2	0.0	11.2	8.5	1.0	10.6	11.6
g_6	2.3	9.5	11.1	9.2	11.2	0.0	5.6	12.1	7.7	8.5
g_7	5.1	10.1	8.1	9.5	8.5	5.6	0.0	9.1	8.3	9.3
g_8	10.2	12.8	1.1	12.0	1.0	12.1	9.1	0.0	11.4	12.4
g_9	6.1	2.0	10.5	1.6	10.6	7.7	8.3	11.4	0.0	1.1
g_{10}	7.0	1.0	11.5	1.1	11.6	8.5	9.3	12.4	1.1	0.0

Constructing the Tree

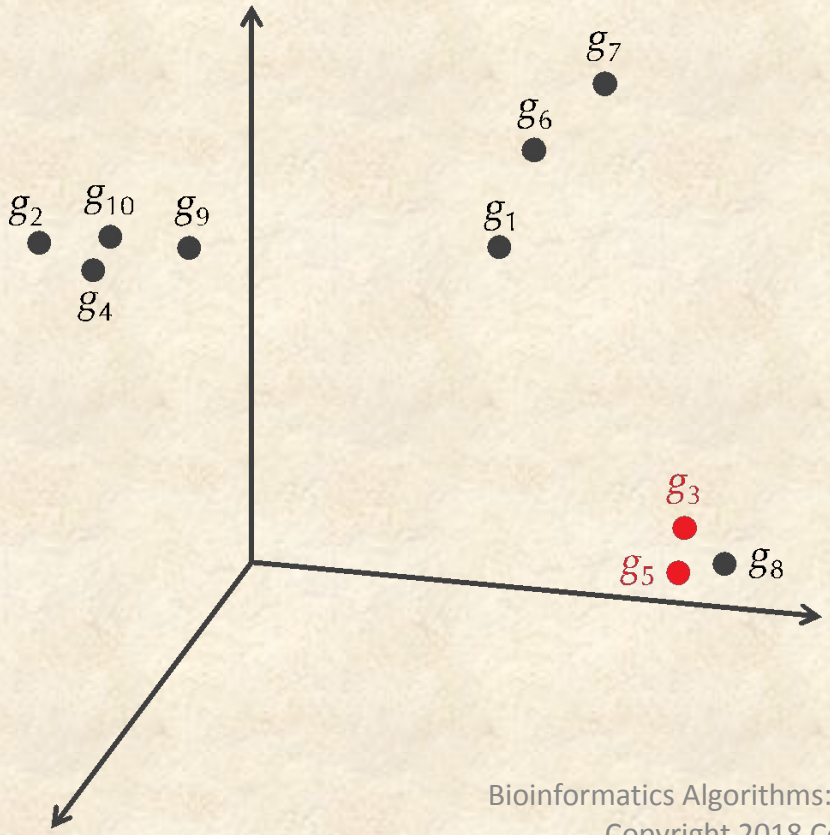
form a node (single-element cluster) for every gene



Distance Matrix

	g_1	g_2	g_3	g_4	g_5	g_6	g_7	g_8	g_9	g_{10}
g_1	0.0	8.1	9.2	7.7	9.3	2.3	5.1	10.2	6.1	7.0
g_2	8.1	0.0	12.0	0.9	12.0	9.5	10.1	12.8	2.0	1.0
g_3	9.2	12.0	0.0	11.2	0.7	11.1	8.1	1.1	10.5	11.5
g_4	7.7	0.9	11.2	0.0	11.2	9.2	9.5	12.0	1.6	1.1
g_5	9.3	12.0	0.7	11.2	0.0	11.2	8.5	1.0	10.6	11.6
g_6	2.3	9.5	11.1	9.2	11.2	0.0	5.6	12.1	7.7	8.5
g_7	5.1	10.1	8.1	9.5	8.5	5.6	0.0	9.1	8.3	9.3
g_8	10.2	12.8	1.1	12.0	1.0	12.1	9.1	0.0	11.4	12.4
g_9	6.1	2.0	10.5	1.6	10.6	7.7	8.3	11.4	0.0	1.1
g_{10}	7.0	1.0	11.5	1.1	11.6	8.5	9.3	12.4	1.1	0.0

Constructing the Tree

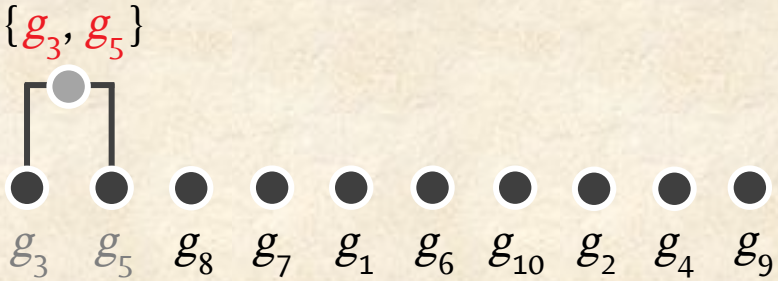


Distance Matrix

	g_1	g_2	g_3	g_4	g_5	g_6	g_7	g_8	g_9	g_{10}
g_1	0.0	8.1	9.2	7.7	9.3	2.3	5.1	10.2	6.1	7.0
g_2	8.1	0.0	12.0	0.9	12.0	9.5	10.1	12.8	2.0	1.0
g_3	9.2	12.0	0.0	11.2	**0.7**	11.1	8.1	1.1	10.5	11.5
g_4	7.7	0.9	11.2	0.0	11.2	9.2	9.5	12.0	1.6	1.1
g_5	9.3	12.0	0.7	11.2	0.0	11.2	8.5	1.0	10.6	11.6
g_6	2.3	9.5	11.1	9.2	11.2	0.0	5.6	12.1	7.7	8.5
g_7	5.1	10.1	8.1	9.5	8.5	5.6	0.0	9.1	8.3	9.3
g_8	10.2	12.8	1.1	12.0	1.0	12.1	9.1	0.0	11.4	12.4
g_9	6.1	2.0	10.5	1.6	10.6	7.7	8.3	11.4	0.0	1.1
g_{10}	7.0	1.0	11.5	1.1	11.6	8.5	9.3	12.4	1.1	0.0

Constructing the Tree

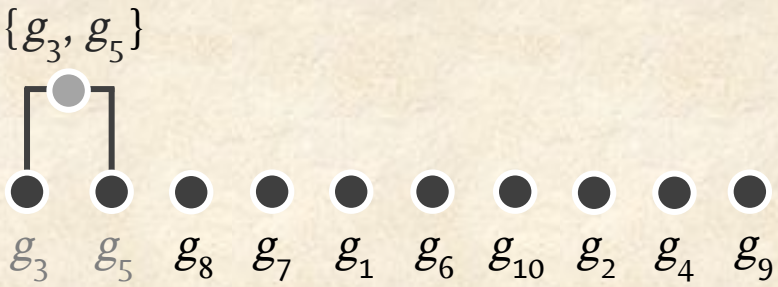
Identify the two closest clusters and merge them.



	g_1	g_2	g_3	g_4	g_5	g_6	g_7	g_8	g_9	g_{10}
g_1	0.0	8.1	9.2	7.7	9.3	2.3	5.1	10.2	6.1	7.0
g_2	8.1	0.0	12.0	0.9	12.0	9.5	10.1	12.8	2.0	1.0
g_3	9.2	12.0	0.0	11.2	**0.7**	11.1	8.1	1.1	10.5	11.5
g_4	7.7	0.9	11.2	0.0	11.2	9.2	9.5	12.0	1.6	1.1
g_5	9.3	12.0	0.7	11.2	0.0	11.2	8.5	1.0	10.6	11.6
g_6	2.3	9.5	11.1	9.2	11.2	0.0	5.6	12.1	7.7	8.5
g_7	5.1	10.1	8.1	9.5	8.5	5.6	0.0	9.1	8.3	9.3
g_8	10.2	12.8	1.1	12.0	1.0	12.1	9.1	0.0	11.4	12.4
g_9	6.1	2.0	10.5	1.6	10.6	7.7	8.3	11.4	0.0	1.1
g_{10}	7.0	1.0	11.5	1.1	11.6	8.5	9.3	12.4	1.1	0.0

Constructing the Tree

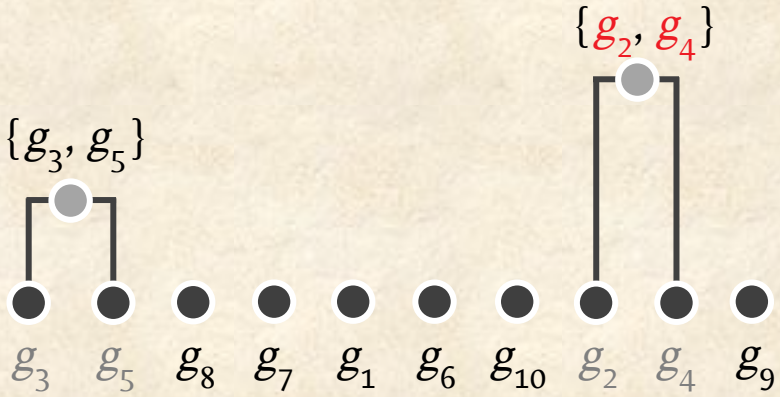
Recompute the distance between two clusters as average distance between elements in the cluster.



	g_1	g_2	g_3, g_5	g_4	g_6	g_7	g_8	g_9	g_{10}
g_1	0.0	8.1	9.2	7.7	2.3	5.1	10.2	6.1	7.0
g_2	8.1	0.0	12.0	0.9	9.5	10.1	12.8	2.0	1.0
g_3, g_5	9.2	12.0	0.0	11.2	11.1	8.1	1.0	10.5	11.5
g_4	7.7	0.9	11.2	0.0	9.2	9.5	12.0	1.6	1.1
g_6	2.3	9.5	11.1	9.2	0.0	5.6	12.1	7.7	8.5
g_7	5.1	10.1	8.1	9.5	5.6	0.0	9.1	8.3	9.3
g_8	10.2	12.8	1.0	12.0	12.1	9.1	0.0	11.4	12.4
g_9	6.1	2.0	10.5	1.6	7.7	8.3	11.4	0.0	1.1
g_{10}	7.0	1.0	11.5	1.1	8.5	9.3	12.4	1.1	0.0

Constructing the Tree

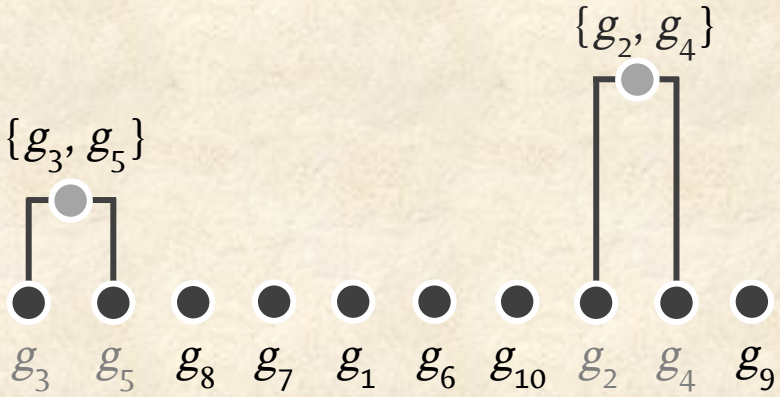
Identify the two closest clusters and merge them.



	g_1	g_2	g_3, g_5	g_4	g_6	g_7	g_8	g_9	g_{10}
g_1	0.0	8.1	9.2	7.7	2.3	5.1	10.2	6.1	7.0
g_2	8.1	0.0	12.0	**0.9**	9.5	10.1	12.8	2.0	1.0
g_3, g_5	9.2	12.0	0.0	11.2	11.1	8.1	1.0	10.5	11.5
g_4	7.7	0.9	11.2	0.0	9.2	9.5	12.0	1.6	1.1
g_6	2.3	9.5	11.1	9.2	0.0	5.6	12.1	7.7	8.5
g_7	5.1	10.1	8.1	9.5	5.6	0.0	9.1	8.3	9.3
g_8	10.2	12.8	1.0	12.0	12.1	9.1	0.0	11.4	12.4
g_9	6.1	2.0	10.5	1.6	7.7	8.3	11.4	0.0	1.1
g_{10}	7.0	1.0	11.5	1.1	8.5	9.3	12.4	1.1	0.0

Constructing the Tree

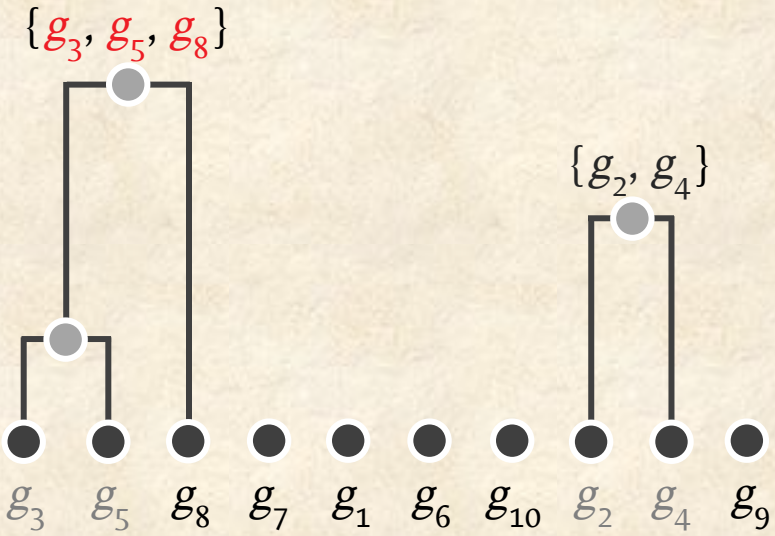
Recompute the distance between two clusters (as average distance between elements in the cluster).



	g_1	g_2, g_4	g_3, g_5	g_6	g_7	g_8	g_9	g_{10}
g_1	0.0	7.7	9.2	2.3	5.1	10.2	6.1	7.0
g_2, g_4	7.7	0.0	11.2	9.2	9.5	12.0	1.6	1.0
g_3, g_5	9.2	11.2	0.0	11.1	8.1	1.0	10.5	11.5
g_6	2.3	9.2	11.1	0.0	5.6	12.1	7.7	8.5
g_7	5.1	9.5	8.1	5.6	0.0	9.1	8.3	9.3
g_8	10.2	12.0	1.0	12.1	9.1	0.0	11.4	12.4
g_9	6.1	1.6	10.5	7.7	8.3	11.4	0.0	1.1
g_{10}	7.0	1.0	11.5	8.5	9.3	12.4	1.1	0.0

Constructing the Tree

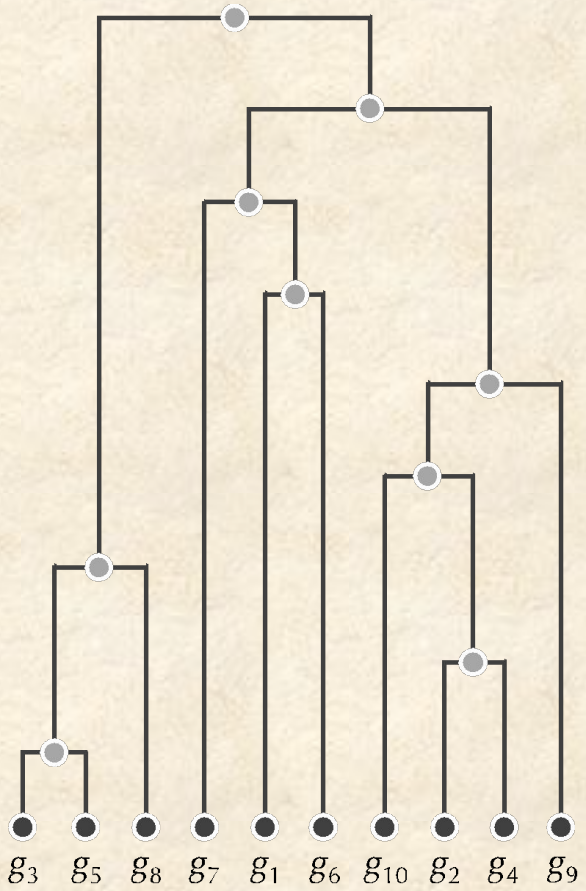
Identify the two closest clusters and merge them.



	g_1	g_2, g_4	g_3, g_5	g_6	g_7	g_8	g_9	g_{10}
g_1	0.0	7.7	9.2	2.3	5.1	10.2	6.1	7.0
g_2, g_4	7.7	0.0	11.2	9.2	9.5	12.0	1.6	1.0
g_3, g_5	9.2	11.2	0.0	11.1	8.1	**1.0**	10.5	11.5
g_6	2.3	9.2	11.1	0.0	5.6	12.1	7.7	8.5
g_7	5.1	9.5	8.1	5.6	0.0	9.1	8.3	9.3
g_8	10.2	12.0	1.0	12.1	9.1	0.0	11.4	12.4
g_9	6.1	1.6	10.5	7.7	8.3	11.4	0.0	1.1
g_{10}	7.0	1.0	11.5	8.5	9.3	12.4	1.1	0.0

Constructing the Tree

Iterate until all elements form a single cluster (root).



Constructing a Tree from a Distance Matrix *D*

HierarchicalClustering (*D, n*)
  Clusters ← *n* single-element clusters labeled 1 to *n*
  T ← a graph with the *n* isolated nodes labeled 1 to *n*
  while there is more than one cluster
    find the two closest clusters C_i and C_j
    merge C_i and C_j into a new cluster C_{new} with $|C_i| + |C_j|$ elements
    add a new node labeled by cluster C_{new} to *T*
    connect node C_{new} to C_i and C_j by directed edges
    remove the rows and columns of *D* corresponding to C_i and C_j
    remove C_i and C_j from *Clusters*
    add a row and column to *D* for the cluster C_{new} by computing
      $D(C_{new}, C)$ for each cluster *C* in *Clusters*
    add C_{new} to *Clusters*
  assign root in *T* as a node with no incoming edges
  return *T*

Constructing a Tree from a Distance Matrix *D*

HierarchicalClustering (*D*, *n*)
  Clusters ← *n* single-element clusters labeled 1 to *n*
  T ← a graph with the *n* isolated nodes labeled 1 to *n*
  while there is more than one cluster
    find the two closest clusters C_i and C_j
    merge C_i and C_j into a new cluster C_{new} with $|C_i| + |C_j|$ elements
    add a new node labeled by cluster C_{new} to *T*
    connect node C_{new} to C_i and C_j by directed edges
    remove the rows and columns of *D* corresponding to C_i and C_j
    remove C_i and C_j from *Clusters*
    add a row and column to *D* for the cluster C_{new} by computing
      $D(C_{new}, C)$ for each cluster *C* in *Clusters*
    add C_{new} to *Clusters*
  assign root in *T* as a node with no incoming edges
  return *T*

Different Distance Functions Result in Different Trees

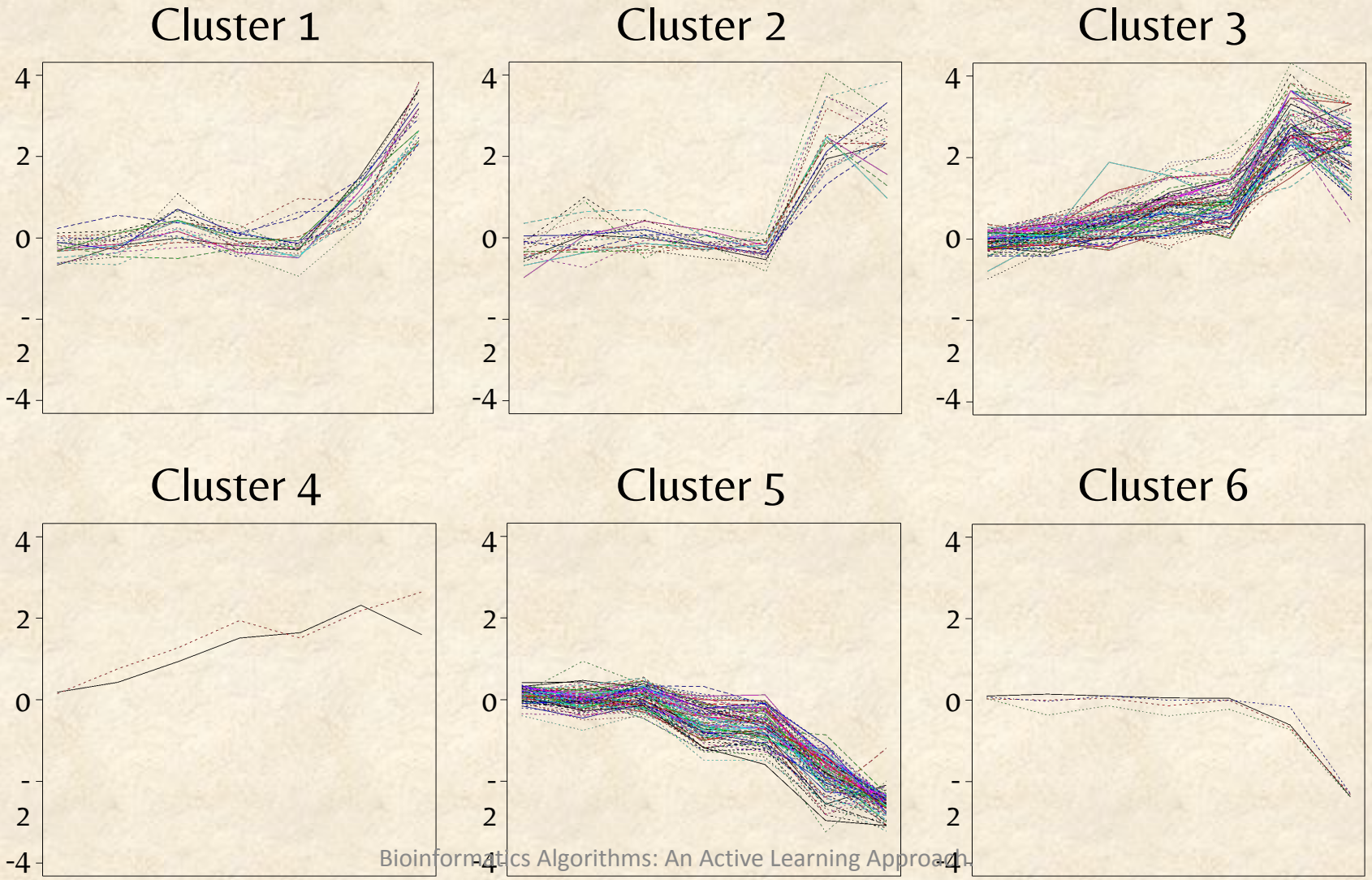
Average distance between elements of two clusters:

$$D_{avg}(C_1, C_2) = (\sum_{\text{all points } i \text{ and } j \text{ in clusters } C1 \text{ and } C2, \text{ respectively}} D_{i,j}) / (|C_1| * |C_2|)$$

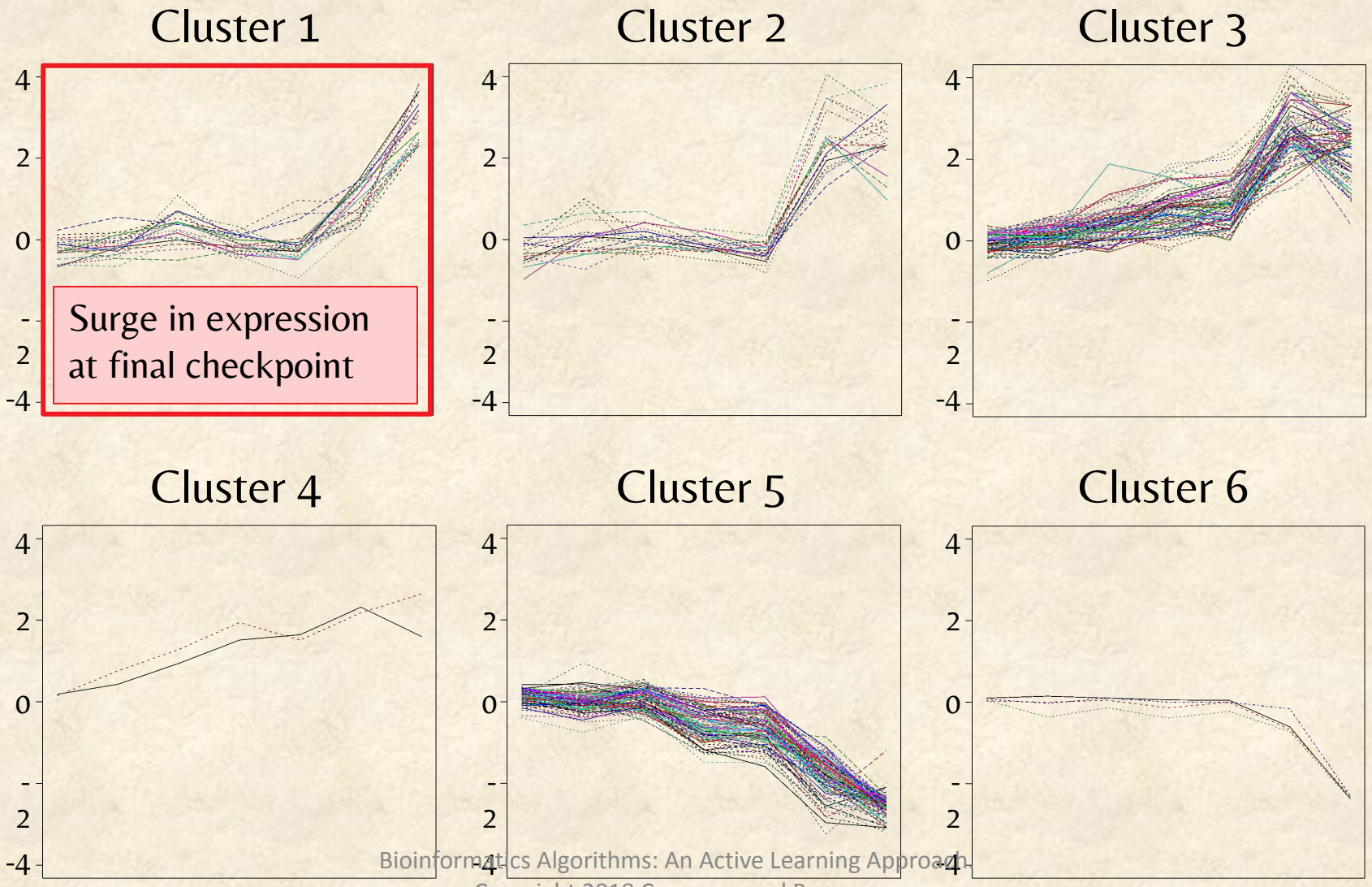
Minimum distance between elements of two clusters:

$$D_{min}(C_1, C_2) = \min_{\text{all points } i \text{ and } j \text{ in clusters } C1 \text{ and } C2, \text{ respectively}} D_{i,j}$$

Clusters Constructed by **HierarchicalClustering**

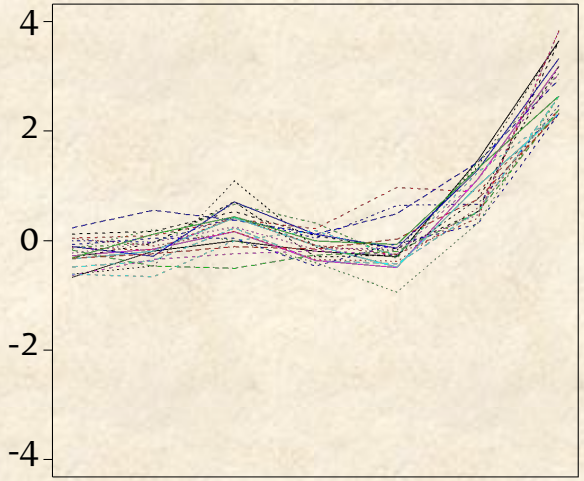


Clusters Constructed by **HierarchicalClustering**



From Gene Expression to Gene Function

- Upstream regions of many genes in this cluster have the **carbon source response element (CSRE)** motif (CATTCATCCG).



- Since yeast prefers glucose over ethanol as an energy source, genes responsible for metabolizing “less tasty” ethanol are repressed in the presence of glucose.

- CSRE activates these genes when the yeast run out of glucose and starts consuming ethanol.