Introducción a la Bioinformática:

Data Redundancy and Clustering

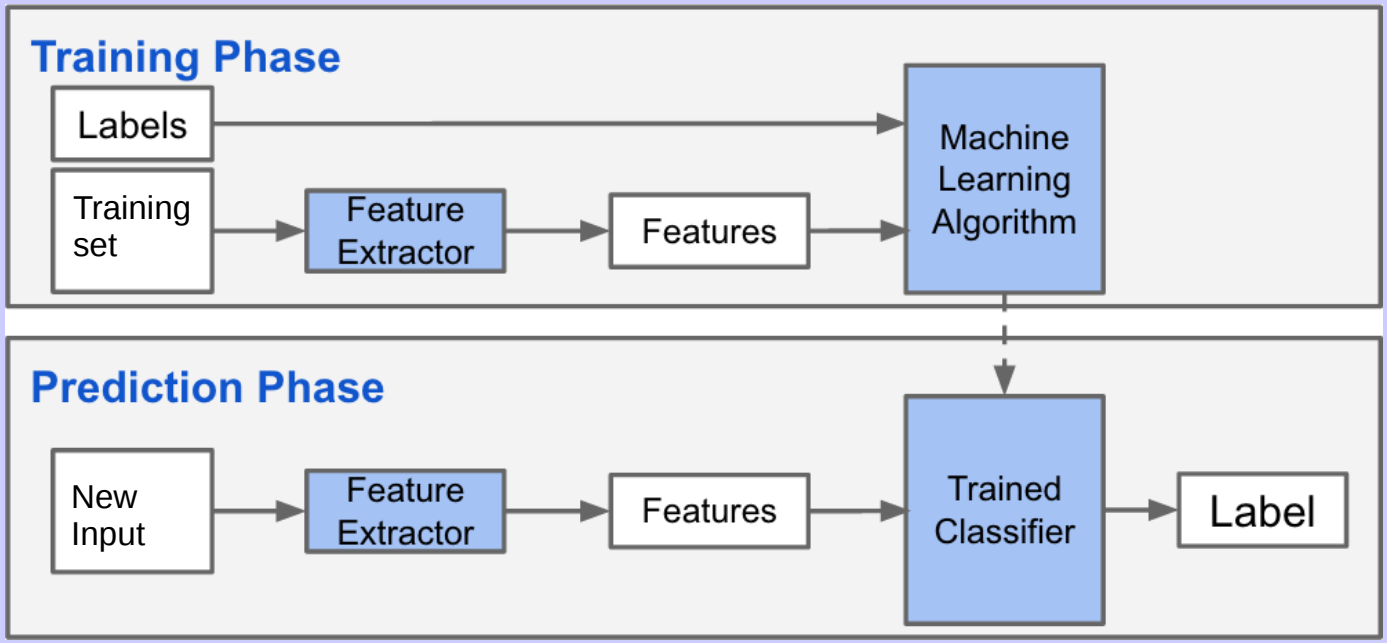
Luis Garreta

Doctorado en Ingeniería
Pontificia Universidad Javeriana – Cali

April 8, 2017

A Machine Learning Algorithm

A machine-learning algorithm is shown a training set, which is a collection of training examples called instances. After learning from the training set, the learning algorithm is presented with additional input vectors, and the algorithm should generalize, that is to decide what the output value should be.



Training a Bioinformatical Method

One very important initial step is **to generate representative sets**:

- If the training data set used to train an algorithm have **many very similar data** examples:
  - It will **not be trained in an optimal manner**.
- The reason for this is first of all that:
  - The algorithm will focus on **learning the data that are repeated**
  - and thereby get a **lower ability to generalize**.
- And, secondly:
  - The **performance** of the prediction method will be **overestimated**, since the data in the **training and test sets will be very alike.**

Generating a representative set from a data set is therefore a very important

Benefits of Reducing Training Dataset

- Reducing the size of the dataset can result in:
  - Avoid noisy and redundant data
  - Increasing capabilities and generalization properties
  - Reducing space complexity of the classification problem
  - Decreasing the required computational time

It is often advisable to reduce original training set by selecting the most representative information

Two approaches for Reducing Data Sets

Data reduction can be achieved by selecting instances and by selecting features.

Selecting Instances (Instance Reduction)

- Becomes especially important in case of **large data sets**.
- Storage and complexity constraints become **computationally expensive**.
- A **variety of methods** has been so far proposed:
  - No single approach can be considered as superior,
  - Nor guaranteeing satisfactory results

Selecting Features

- **Remove features that are irrelevant** for classification results.
- When relevant features are unknown **a priori many features are introduced** with a view to better represent the domain.
  - Many are **irrelevant** from the point of view of classification results
  - Introduces **noise** to the data mining analysis
- Resulting in **negative influence**:
  - on the accuracy, and
  - on the required learning time of the classifier.
- Computational complexity of the learning process increases.

The **number of instances needed** to assure the required classification accuracy **grows** exponentially with the number or irrelevant features

Algorithms for Reducing Data Sets

- Algorithms for Clustering Instances
  - Agglomerative: Hierarchical
  - Partitioning: K-Means
  - Greedy: CD-Hit
- Algorithms for Selecting Instances:
  - Hobohm and Sander's Algorithm for making a representative set.
  - CD-Hit: Ultrafast protein/nucleotides sequence clustering program.
- Algorithms for Selecting Features:
  - Principal Component Analysis (PCA) for reducing dimensionality

Hierarchical Clustering Algorithm

S. C. Johnson (1967) "Hierarchical Clustering Schemes"

Steps:

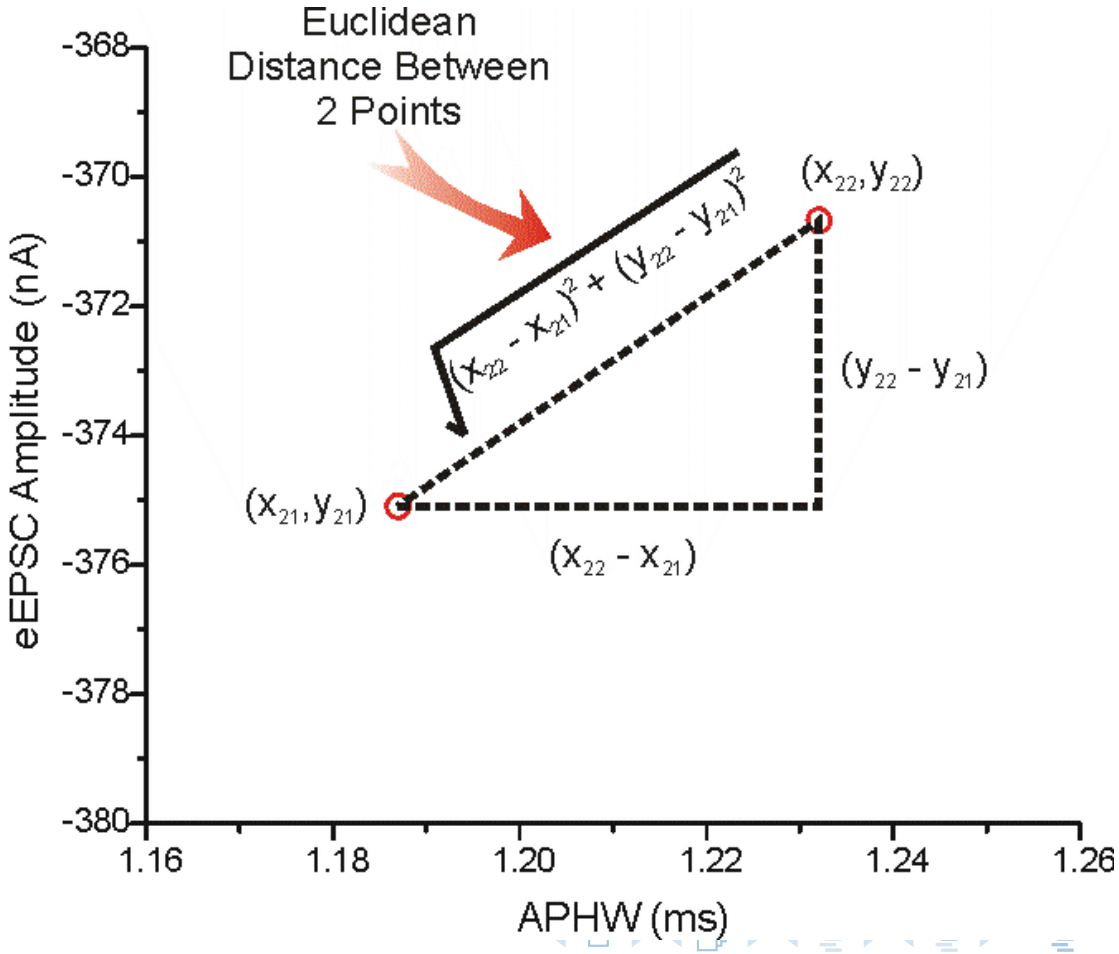
Let X = $x_1, \ldots, x_n$ be the set of data points.

1. Start with each item $x_1, \ldots, x_n$ in its own cluster $c_1, \ldots, c_n$
2. Let the distances (similarities) between the clusters equal the distances (similarities) between the items they contain.
3. Repeat:
   - 3.1 Merge the closest pair of clusters (c_i, c_j) into a single cluster, so that now you have one less cluster.
   - 3.2 Compute distance (similarities) between new cluster and each of the old clusters

Euclidean Distance

There are many metrics to calculate a distance between 2 points: p(x1, y1) and q(x2, y2) in xy-plane.

- Euclidean
- Manhattan
- Chebyshev



Computing Distances in Hierarchical Clustering

- Computing distances can be done in different ways:
  - Single-linkage
  - Complete-linkage
  - Average-linkage (= UPGMA)
  - Others (from R hclust):
    - “"ward.D"”
    - “"ward.D2"”,
    - “"mcquitty"” (= WPGMA),
    - “"median"” (= WPGMC) or “"centroid"” (= UPGMC).

Common Distance Linkages

Single Linkage

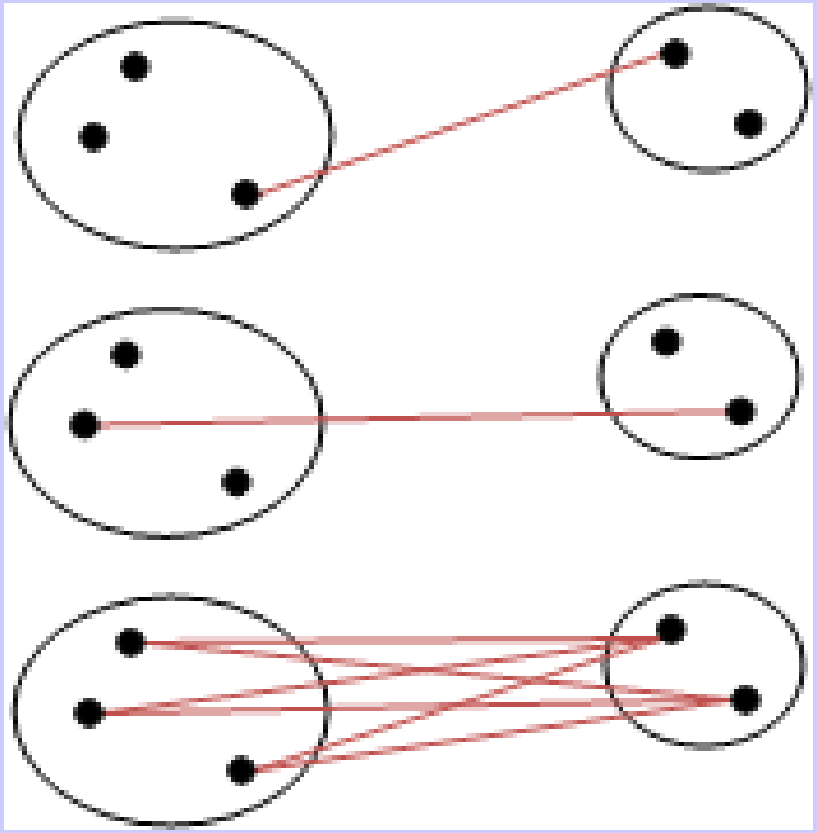
$$d(x,x') = \min_{x \in C_i, x' \in C_j} d(x,x')$$

Complete Linkage

$$d(x,x') = \max_{x \in C_i, x' \in C_j} d(x,x')$$

Average Linkage

$$d(x,x') = \frac{\sum x \in C_i, x' \in C_j \, d(x,x')}{|C_i| . |C_j|}$$



Example Hierarchical Clustering

Clustering of distances in miles between U.S. cities

Input distance Matrix:

	BOS	NY	DC	MIA	CHI	SEA	SF	LA	DEN
BOS	0	206	429	1504	963	2976	3095	2979	1949
NY	206	0	233	1308	802	2815	2934	2786	1771
DC	429	233	0	1075	671	2684	2799	2631	1616
MIA	1504	1308	1075	0	1329	3273	3053	2687	2037
CHI	963	802	671	1329	0	2013	2142	2054	996
SEA	2976	2815	2684	3273	2013	0	808	1131	1307
SF	3095	2934	2799	3053	2142	808	0	379	1235
LA	2979	2786	2631	2687	2054	1131	379	0	1059
DEN	1949	1771	1616	2037	996	1307	1235	1059	0

Example Hierarchical Clustering

After merging BOS with NY:

	BOS/NY	DC	MIA	CHI	SEA	SF	LA	DEN
BOS/NY	0	223	1308	802	2815	2934	2786	1771
DC	223	0	1075	671	2684	2799	2631	1616
MIA	1308	1075	0	1329	3273	3053	2687	2037
CHI	802	671	1329	0	2013	2142	2054	996
SEA	2815	2684	3273	2013	0	808	1131	1307
SF	2934	2799	3053	2142	808	0	379	1235
LA	2786	2631	2687	2054	1131	379	0	1059
DEN	1771	1616	2037	996	1307	1235	1059	0

Example Hierarchical Clustering

After merging DC with BOS-NY:

	BOS/NY/DC	MIA	CHI	SEA	SF	LA	DEN
BOS/NY/DC	0	1075	671	2684	2799	2631	1616
MIA	1075	0	1329	3273	3053	2687	2037
CHI	671	1329	0	2013	2142	2054	996
SEA	2684	3273	2013	0	808	1131	1307
SF	2799	3053	2142	808	0	379	1235
LA	2631	2687	2054	1131	379	0	1059
DEN	1616	2037	996	1307	1235	1059	0

Example Hierarchical Clustering

After merging SF with LA:

	BOS/NY/DC/	MIA	CHI	SEA	SF/LA	DEN
BOS/NY/DC	0	1075	671	2684	2631	1616
MIA	1075	0	1329	3273	2687	2037
CHI	671	1329	0	2013	2054	996
SEA	2684	3273	2013	0	808	1307
SF/LA	2631	2687	2054	808	0	1059
DEN	1616	2037	996	1307	1059	0

Example Hierarchical Clustering
After merging CHI with BOS/NY/DC:

	BOS/NY/DC/CHI	MIA	SEA	SF/LA	DEN
BOS/NY/DC/CHI	0	1075	2013	2054	996
MIA	1075	0	3273	2687	2037
SEA	2013	3273	0	808	1307
SF/LA	2054	2687	808	0	1059
DEN	996	2037	1307	1059	0

Example Hierarchical Clustering

After merging SEA with SF/LA:

	BOS/NY/DC/CHI	MIA	SF/LA/SEA	DEN
BOS/NY/DC/CHI	0	1075	2013	996
MIA	1075	0	2687	2037
SF/LA/SEA	2054	2687	0	1059
DEN	996	2037	1059	0

Example Hierarchical Clustering

After merging DEN with BOS/NY/DC/CHI:

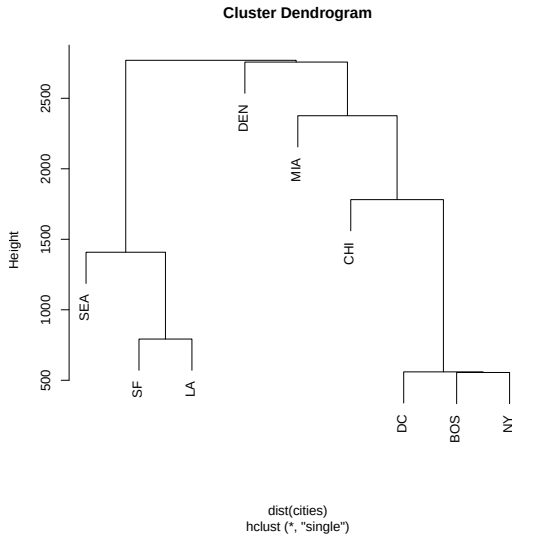
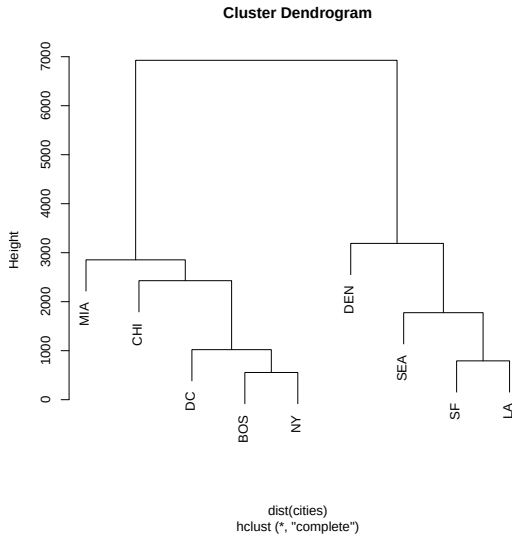
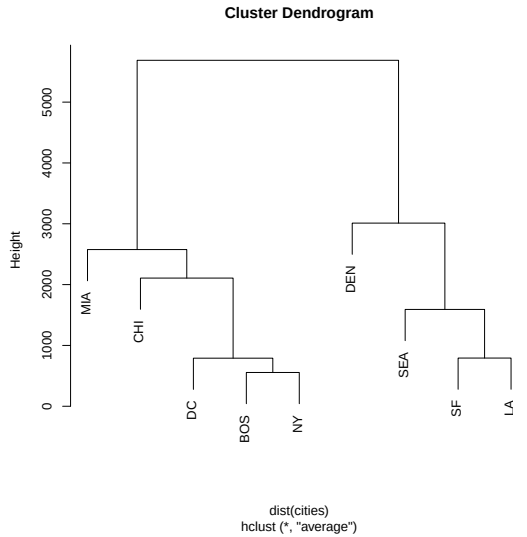
	BOS/NY/DC/CHI/DEN	**MIA**	**SF/LA/SEA**
BOS/NY/DC/CHI/DEN	0	1075	1059
MIA	1075	0	2687
SF/LA/SEA	1059	2687	0

Example Hierarchical Clustering

After merging SF/LA/SEA with BOS/NY/DC/CHI/DEN:

	BOS/NY/DC/CHI/ DEN/SF/LA/SEA	**MIA**
BOS/NY/DC/CHI/ DEN/SF/LA/SEA	0	1075
MIA	1075	0

Dendograms using different distance methods

Single	Complete	Average
		

Hierarchical Clustering Considerations

Advantages

1. No apriori information about the number of clusters required.
2. Easy to implement and gives best result in some cases.

Hierarchical Clustering Considerations

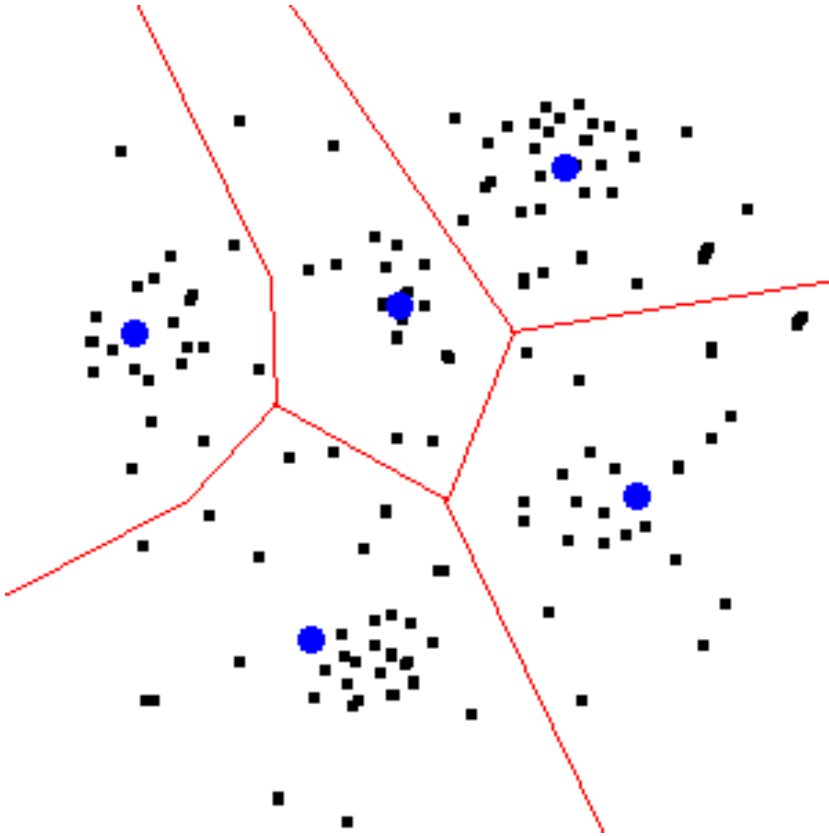
Disadvantages

1. Algorithm can never undo what was done previously.
2. Time complexity of at least $O(n^2 logn)$ is required, where 'n' is the number of data points.
3. Based on the type of distance matrix chosen for merging different algorithms can suffer with one or more of the following:
   - 3.1 Sensitivity to noise and outliers
   - 3.2 Breaking large clusters
   - 3.3 Difficulty handling different sized clusters and convex shapes
4. No objective function is directly minimized

Sometimes it is difficult to identify the correct number of clusters by the dendogram.

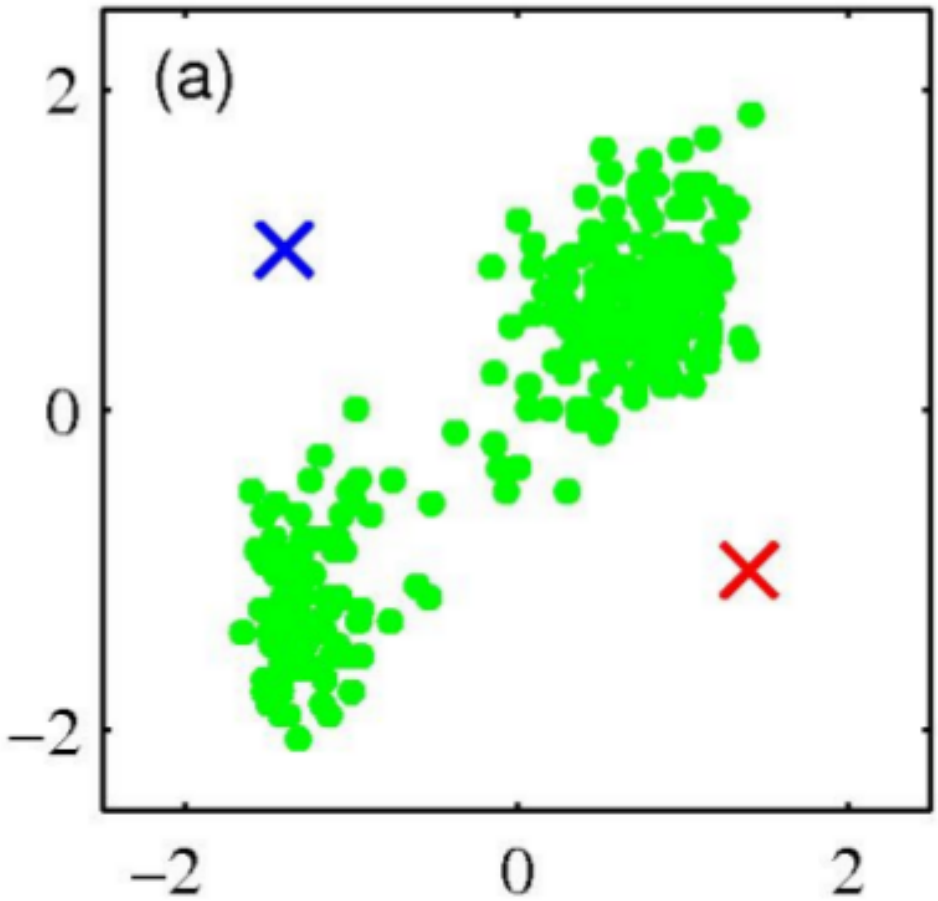
K-means Properties

- One of the most important partitioning Method
- Different concept than the hierarchical clustering
- Not based on distance measures (e.g. Euclidean)
- Instead, it uses the **whitin-cluster variation**
  - Try to form homogenous clusters
  - Segmenting the data
  - Minimizing the Within-cluster variation
- Then, selection of distance measure is not needed



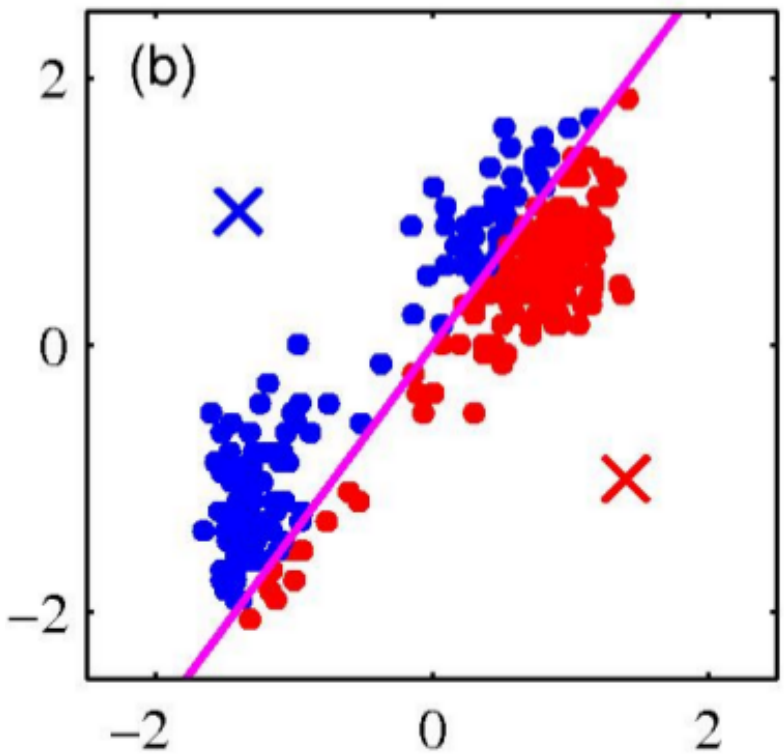
K-means Example:
Define initial K clusters

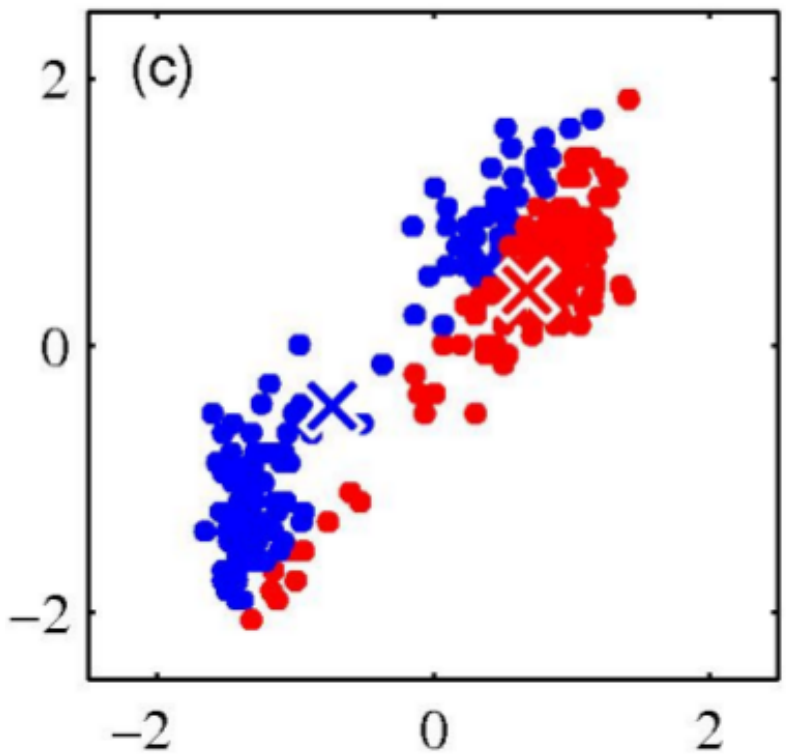
Pick K random points as cluster centers (means)



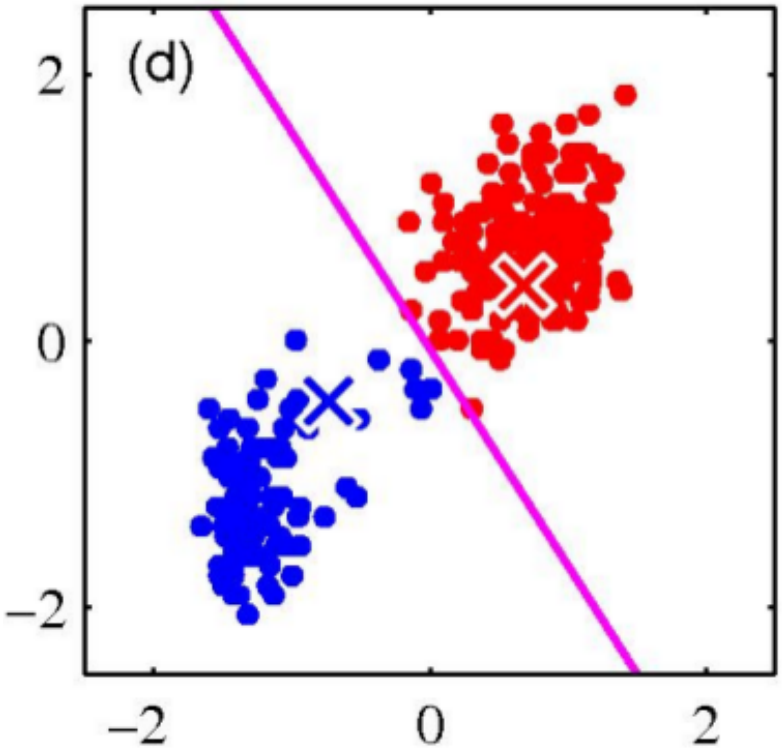
K-means Example: *Iterative Step1 and Step2*

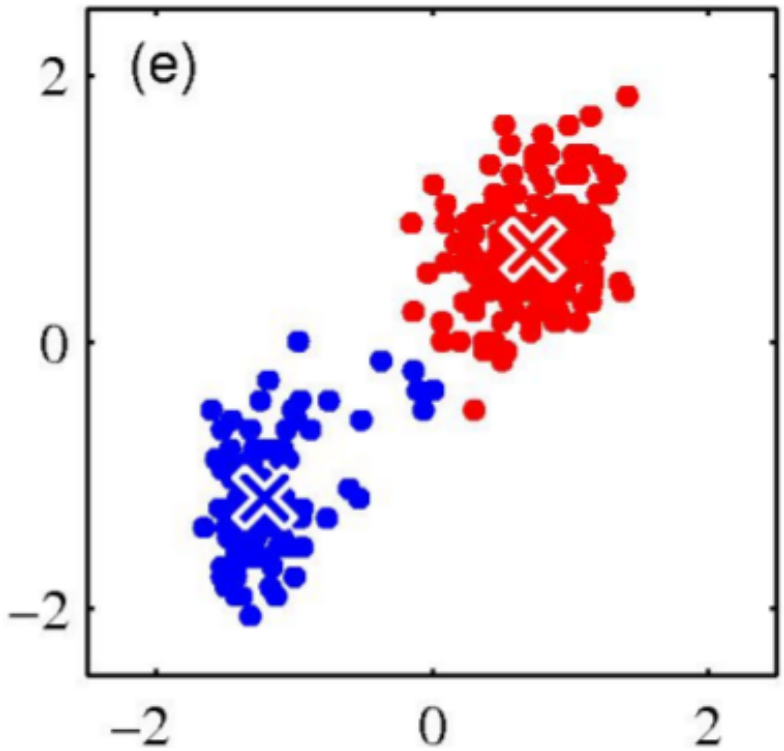
Step1	Step2
Assign data points to closest cluster center	Change the cluster center to the average of the assigned points



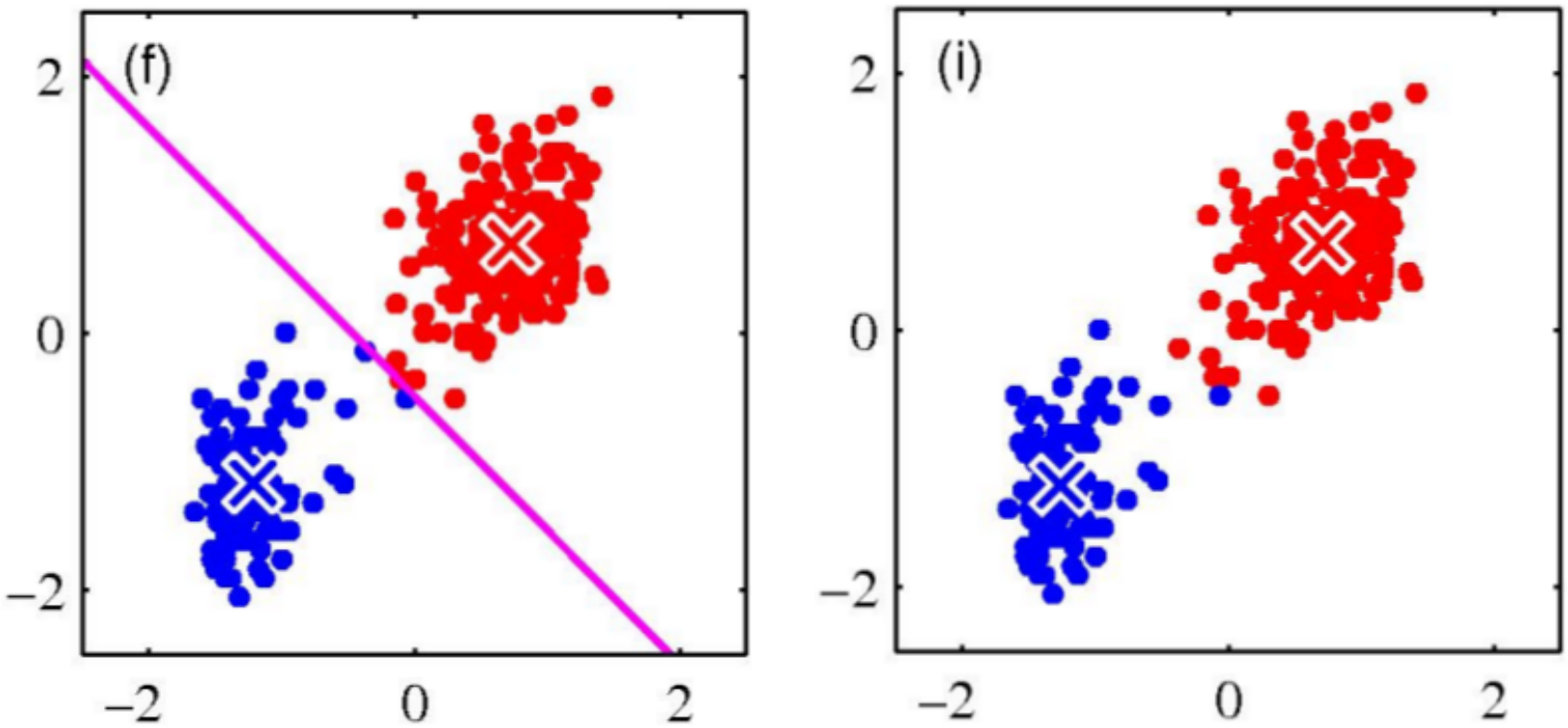


K-means Example:
Repeat until convergence





K-means Example:
Repeat until convergence



K-means Algorithm: *Parameters*

- K-means partitions a set of N points into K clusters
- Each cluster is represented with a mean (a centroid o "k-means")
- **Input**:
  - A set **V** with **N** points ($v_1, v_2, \ldots, v_n$), and
  - The desired number of clusters ***K,*** and
  - A distance measure between any two points $d(v, w)$
- **Output:**
  - A set ***X*** of ***K*** cluster centers that minimize the squared error distortion ***D(V,X)*** over all possible choices of ***X:***

$$D(V, X) = \frac{1}{N} \sum_{i=1}^{N} \min_k d^2(v_i, x_k)$$

  - and the labels $l_1, \ldots, l_k$ for every data point ($v_1, v_2, \ldots, v_n$)

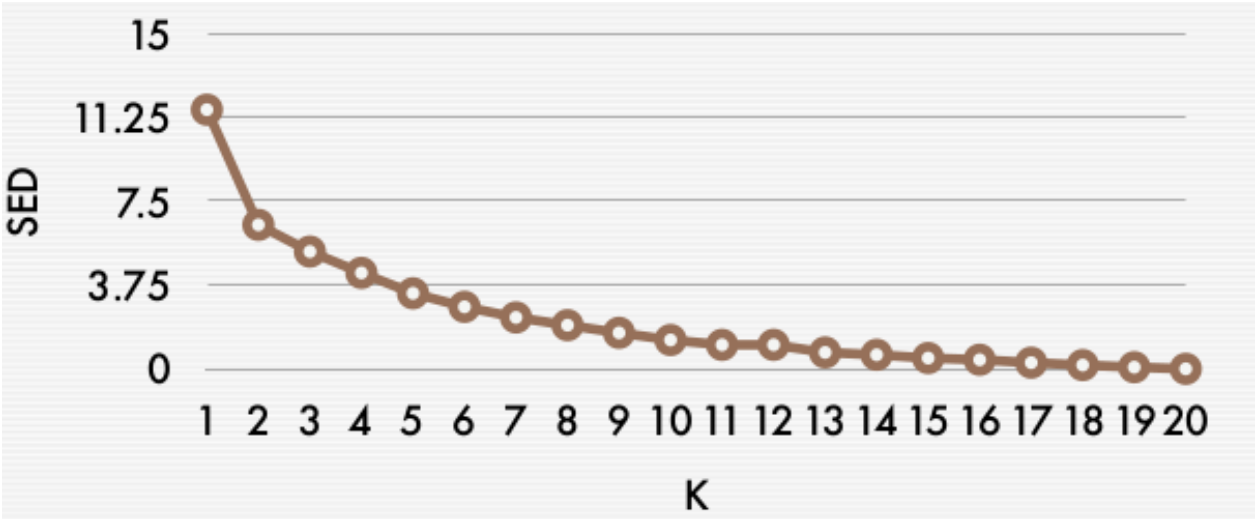
K-means Algorithm:

Lloyd's Algorithm

1. Arbitrarily assign the K cluster centers:
   (This can significantly influence the outcome)
2. **while** cluster centers keep changing
   - 2.1 Compute the distance from each data point to the current cluster center $C_i, 1 \leq i \leq K$ and assign the point to the nearest cluster
   - 2.2 After the assignment of all data points, compute new centers for each cluster by taking the centroid of all the points in that cluster
3. Output cluster centers and assignments

How To Choose K?

- The simplest approach is to start with K=1 and increase K until the squared error distortion (SED) stops decreasing



Representative Datasets

Sequence motifs in the genomes of sripuviruses, curioviruses, hapaviruses and ephemeroviruse

Ephemerovirus

KOTV (U1>U1x) AAAAUUUCAUAACUGGGAGUAAAAAAUUGAAGGAAGAGGAGACUGAAAUCGCAUAGGAUG

KOOLV (U1>U1x) AUAACUUCUCAUCGGGCAACAAACGACUAAAAGAAGAGAAAGAGGAAACUGCUUGAAAUG

BEFV (U1>U1x) GGAUUAUAUUGAUUAUAUUACUGGGAAUUCUUGCAAUUAGGAUA-100-CAUAAAAUG

BRMV (U1>U1x) AAGGUCGGAUCAGGUUGGGAUUUUGGAUUAUCUUUAUAAUACU-125-AUUAGAAUG

Hapavirus

GLOV (U1>U1x) AGUUGCAAAAUACCUGUCAUGGGAUGAAGCGAACAGGCUGUUCAAACUAGUAGAAUGA

GLOV (G>Gx) UUGACCCCCCAUCAGUAGAGAGGCGCAAAGGCAGCAGUGUAUUUUCUCAUUACUAAUG

FLAV (G>Gx) ACACCCCCUCAUGGGAAAGGAACUGGGGUCAAAUACUUUGAUUAUUAAUG

HPV (G>Gx) ACACCCCCUCAUGGGAAAGGAAAUGGGGUCAAGUAUUUUGAAUACUAAUG

MANV (G>Gx) UCAAACUGCUCAUGGGAAGGAAAGUGGUUUUUUUUGACUUUGUAUAAACAAUG

MQOV (G>Gx) CAGUGGCUCGUGGGAAUCAAGGCCAUCCGUUCUUUUCUUUCUGAUG

KAMV (G>Gx) GAGGGCUCAUGGGAAAGACUUAGCCCAUAAUUACUUUCAAUACUGAUG

MOSV (G>Gx) AAGGCCCAUGGGAAAAGUCAAGGCCACAGUUACUUCCAGUACUAAUG

LJAV (G>Gx) AAAGUUGCUCAUGGGAAGGAAGGCAACUAUGCAGAUUUUAUGUAAUG

Curiovirus

ARUV (U1>U1x) AGGAGGCGCAUAUUGAGACCUAGCCCUCCUCCA-60-CAGGUAUAAUG

Sripuvirus

NIAV (M>Mx) UACUGAGGAGUGGGAAAUCACGUACUCAGAUCUGUGCAAGACCAUGA

SRIV (M>Mx) UACUGAGGCGUGGGAAAAAUUCUACUCAGAUGUGCAGAAGACCAUGA

CHOV (M>Mx) CUUUCCAAUUUGAGAUUCAAAAGAAUAAGAUUUUAAUAUCUCCUAAAUGA

True Motifs or Noisy Sequences?

Rank	Match Score	Redundant Motif	P-value	log P-value	% of Targets	% of Background
1	0.918		1e-1776	-4089.766852	26.30%	4.60%
2	0.873		1e-1711	-3941.421170	25.85%	4.62%
3	0.844		1e-968	-2231.146991	25.56%	7.71%
4	0.616		1e-259	-597.025749	12.81%	5.44%
5	0.662		1e-233	-537.315538	13.40%	6.12%
6	0.795		1e-222	-512.488031	22.69%	13.20%
7	0.874		1e-148	-341.450152	20.88%	13.25%

Representative Data Sets

- In sequence analysis **a number of algorithms** exist for selecting a representative subset from a set of data points.
- This is generally done by **keeping only one of two very similar data points**.
- In order to do this a measure for **similarity must be defined** between two data points:
  - e.g., percentage identity, alignment score, or significance of alignment score.
- Hobohm et al. [1992] have presented two algorithms for making a representative set from a list of data points D.

Hobohm1 Algorithm

- Fast
- Requires a prior sorting of data

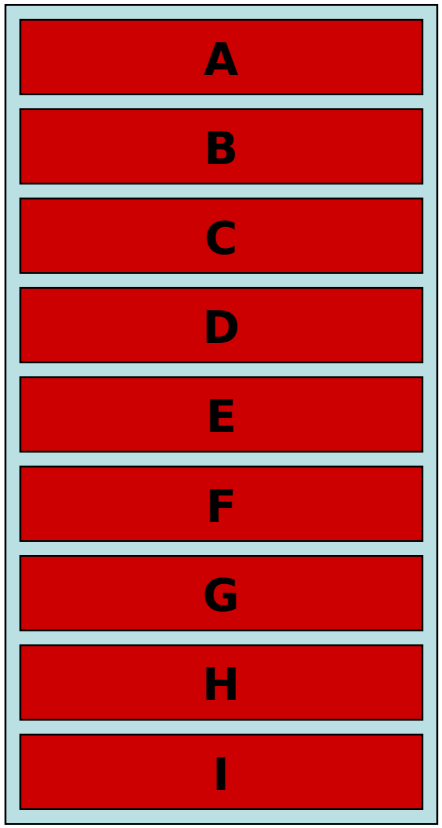
Algorithm:

Repeat for all data points on the list D:

- Add next data point in D to list of non-redundant data points N if it is not similar to any of the elements already on the list.

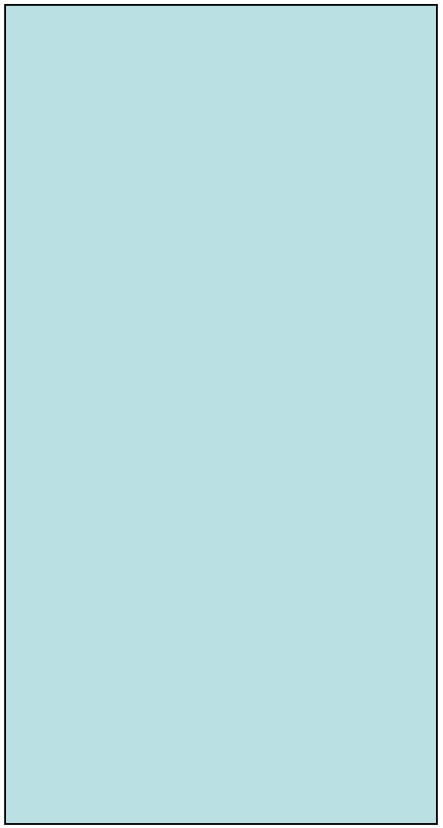
Hobbon1: Start with an empty and sorted list

Input data - sorted list

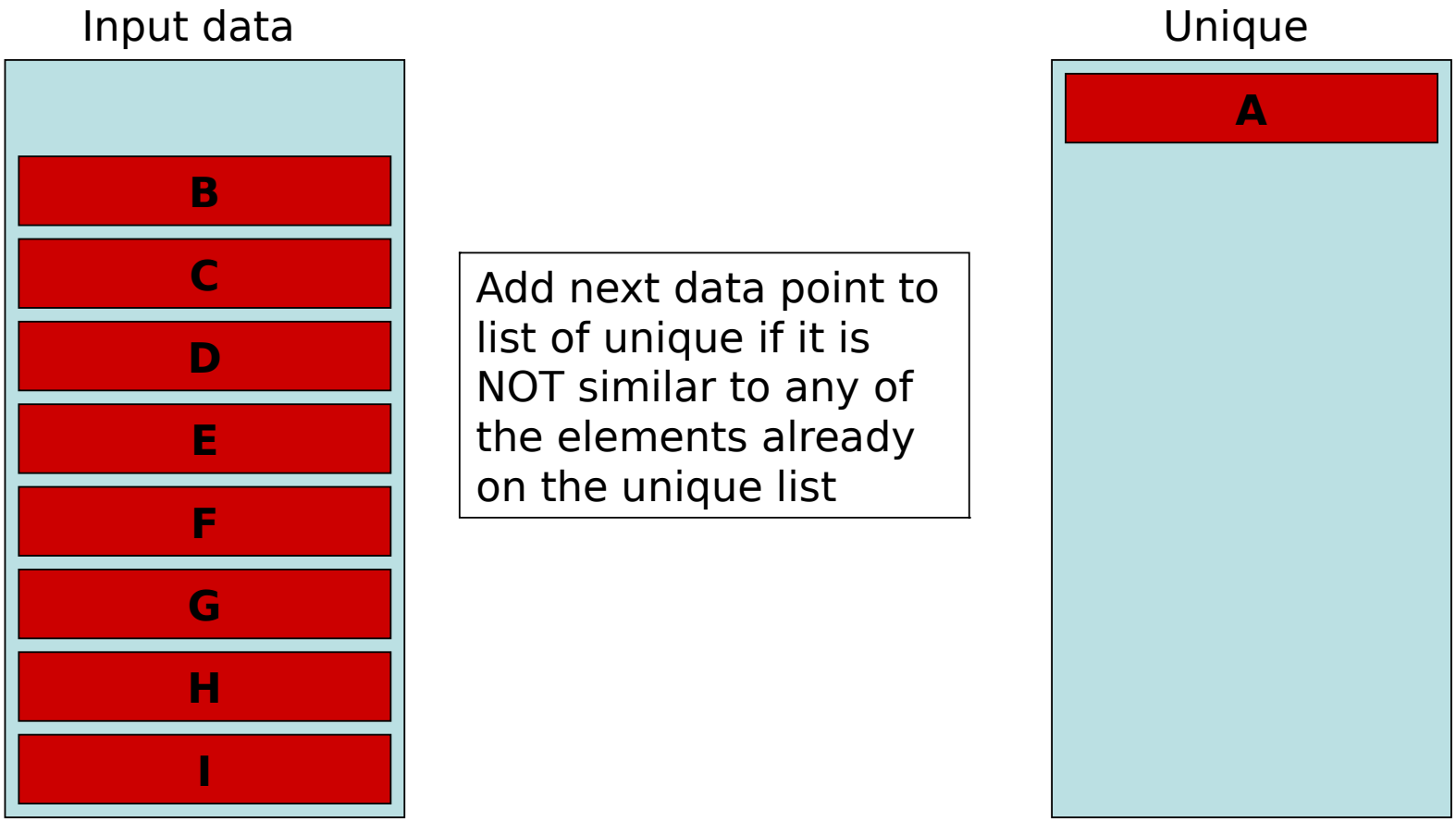


Add next data point to list of unique if it is NOT similar to any of the elements already on the unique list

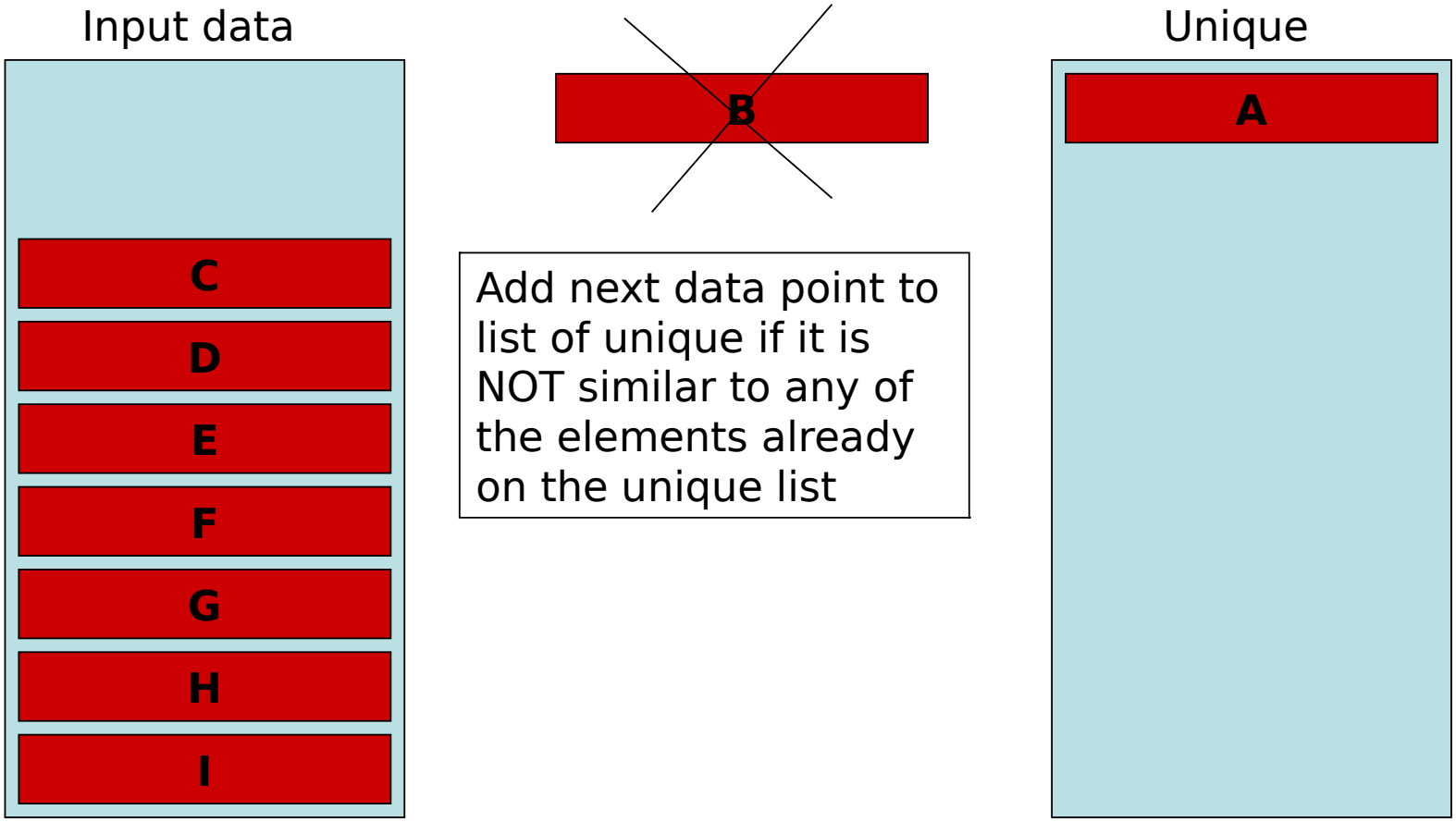
Unique



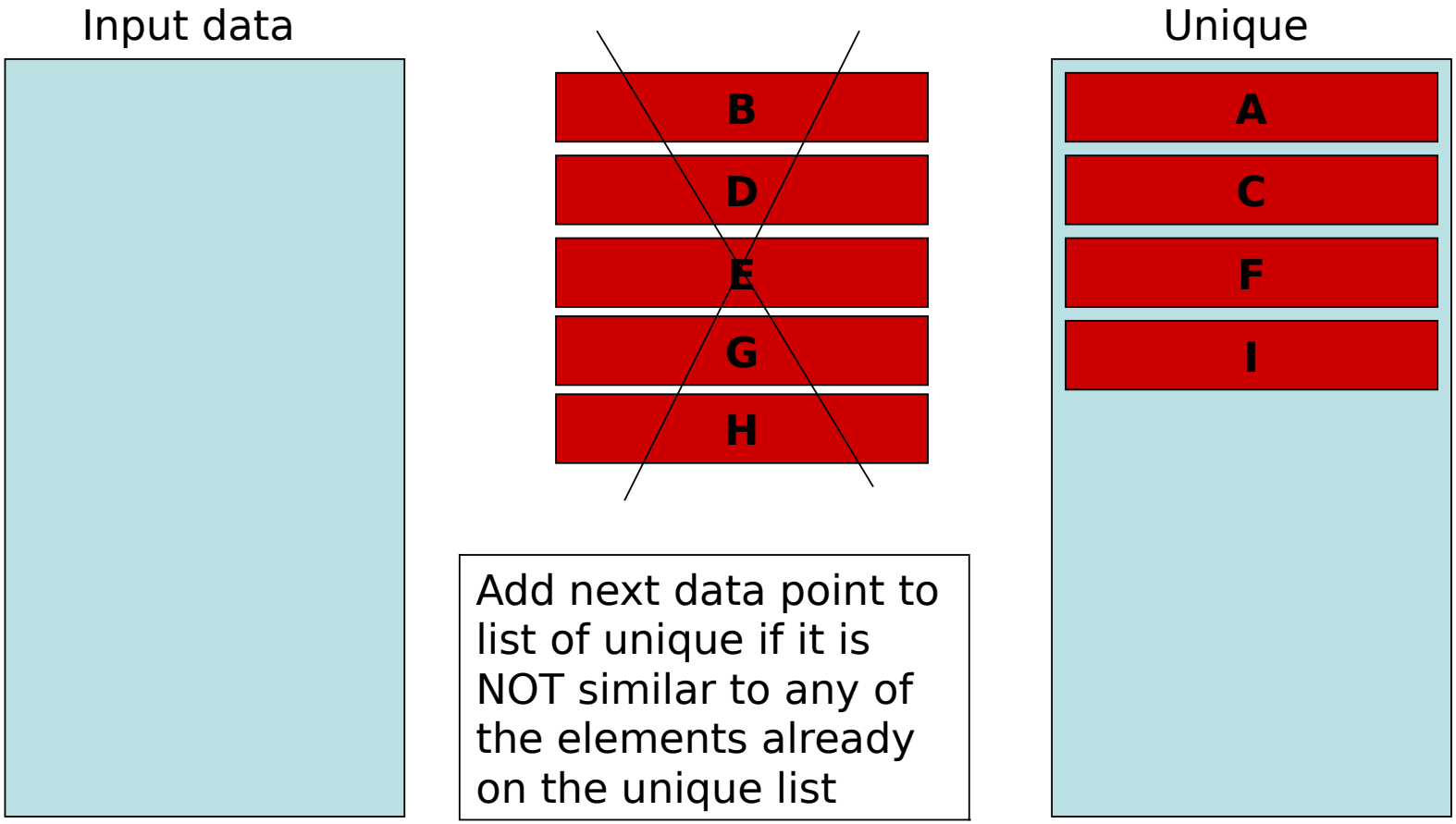
Hobbon1: Add First Element Without Any Comparison



Hobbon1: Add Next One Comparing with the First One



Hobbon1: Add the Other Ones and Compare



Hobbon1 Considerations

- Before applying the algorithm, **the data points can be sorted** according to some **property**.
- So, **maximizing the average value of this property** in the selected set:
  - Points higher on the list have **less chance of being filtered out**.
- The property can, e.g., be chosen to be the quality of the experimental determination of the data point.

CD-Hit: Fast Clustering for Large Datasets (Databases)

Drawbacks of Clustering Approaches for Large Datasets

- In Bioinformatics:
  - Size of datasets commonly seen in current research (millions to billions of sequences)
- Previous clustering approaches:
  - Require the computation of all pairwise distances between a set of sequences,
  - i.e. the running time is proportional to n^2 , where n is the number of sequences.
- Given the large size of bioinformatics datasets:
  - Such algorithms are impractical!

Greedy Clustering

- An alternative is provided by greedy clustering algorithms, exemplified by CD-hit (Li et al.)

CD-Hit Overview

- CDHIT is a program commonly **used to cluster nucleotide/protein sequences**.
- It is **used routinely by NCBI** to get rid of redundant sequences in the NR (non-redundant) database.
- **It is extremely fast** compared to a traditional all vs all blast and subsequent pair-wise clustering.
- **CDHIT doesn't use dynamic programming** to determine sequence similarity.
  - That's probably the biggest reason for it's speed.
  - It looks strictly at exact sequence **identity of k-mers**.

CD-Hit: *The basic algorithm*

1. Sort the sequences in decreasing order of their lengths
2. Pick the first sequence not assigned to a cluster :
   - This sequence becomes the center of a new cluster
3. Compare all unasigned sequences to already computed cluster centers:
   - First using a quick k-mer distance approach,
   - Then checking the promising alignments with a full smith-waterman algorithm.
   - If any of the sequence falls above the threshold for similarity, then it is grouped together with the first sequence.
4. Repeat from 2.

The output cluster sequences are the longest sequence out of each cluster group.

Exercise

Implements de CD-Hit Algorithm
Resources:

- CD-Hit is explained at:
  `http://blog.nextgenetics.net/?e=26`
- Some sequences for testing at:
  `http://github.com/lgarreta/bioinformatica/04-clustering/`