Data-Intensive Distributed Computing

CS 431/631 451/651 (Winter 2019)

Part 6: Data Mining (4/4)

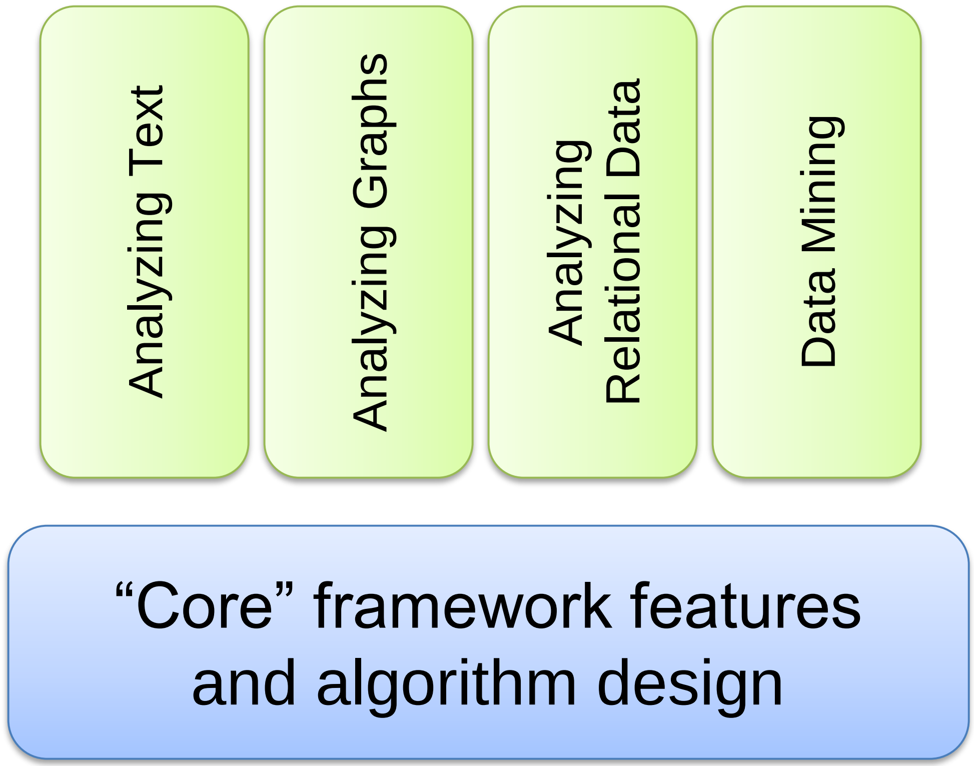
March 12, 2019

Adam Roegiest

Kira Systems

These slides are available at http://roegiest.com/bigdata-2019w/

Structure of the Course



Theme: Similarity

How similar are two items? How “close” are two items?

Equivalent formulations: large distance = low similarity

Lots of applications!

Problem: find similar items

Offline variant: extract all similar pairs of objects from a large collection

Online variant: is this object similar to something I’ve seen before?

Last time!

Problem: arrange similar items into clusters

Offline variant: entire static collection available at once

Online variant: objects incrementally available

Today!

Clustering Criteria

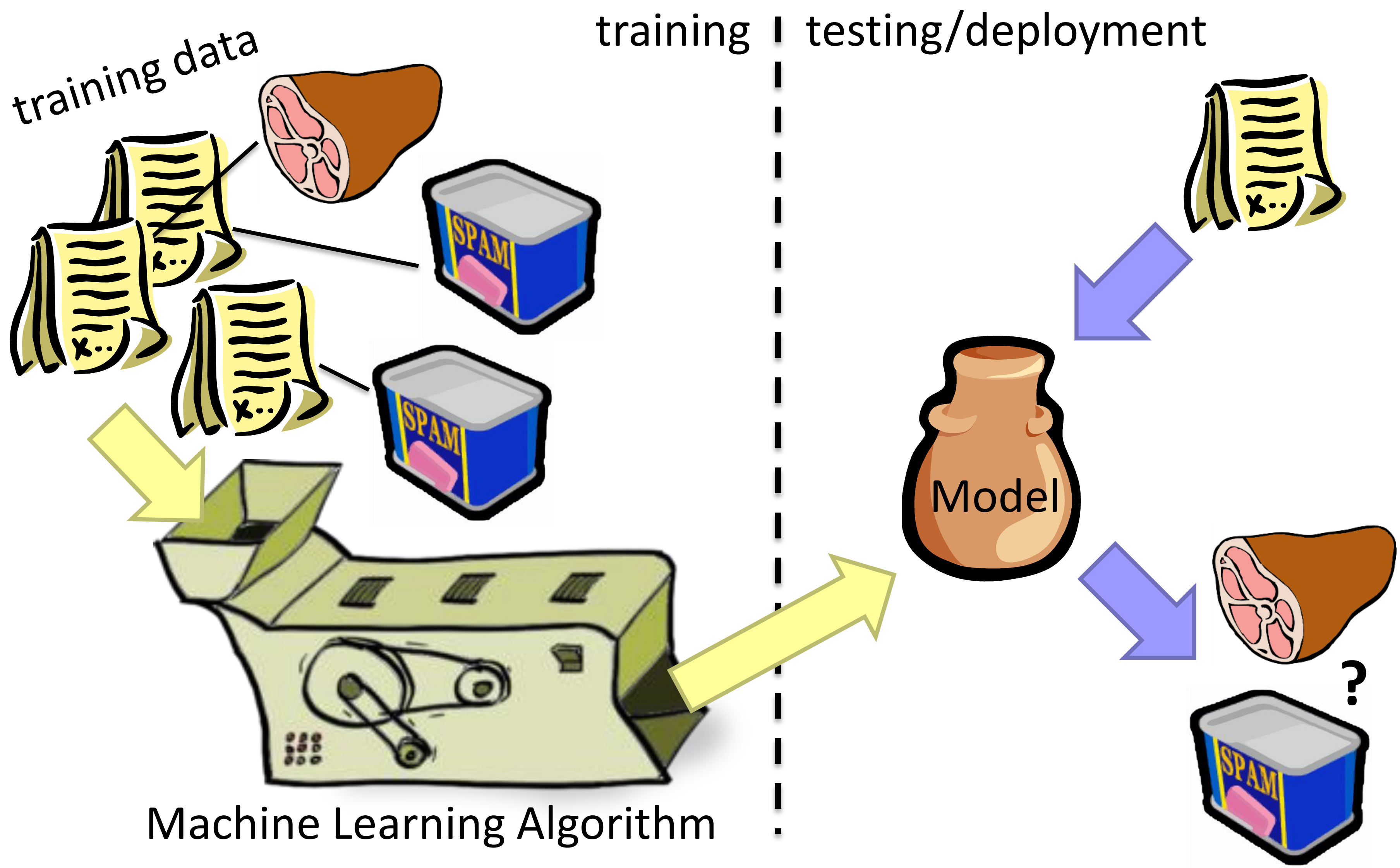
How to form clusters?

High similarity (low distance) between items in the same cluster

Low similarity (high distance) between items in different clusters

Cluster labeling is a separate (difficult) problem!

Supervised Machine Learning



Unsupervised Machine Learning

If supervised learning is function induction…
what's unsupervised learning?

Learning something about the inherent structure of the data

What's it good for?

Applications of Clustering

Clustering images to summarize search results

Clustering customers to infer viewing habits

Clustering biological sequences to understand evolution

Clustering sensor logs for outlier detection

Evaluation

How do we know how well we're doing?

Classification

Nearest neighbor search

Clustering

Inherent challenges of unsupervised techniques!

Clustering

Source: Wikipedia (Star cluster)

Clustering

Specify distance metric

Jaccard, Euclidean, cosine, etc.

Compute representation

Shingling, tf.idf, etc.

Apply clustering algorithm

Distance Metrics

Source: www.flickr.com/photos/thiagoalmeida/250190676/

Distance Metrics

1. Non-negativity:

$$\mathrm{d}(\mathrm{x}, \mathrm{y}) \geq 0$$

2. Identity:

$$\mathrm{d}(\mathrm{x}, \mathrm{y}) = 0 \iff \mathrm{x} = \mathrm{y}$$

3. Symmetry:

$$\mathrm{d}(\mathrm{x}, \mathrm{y}) = \mathrm{d}(\mathrm{y}, \mathrm{x})$$

4. Triangle Inequality

$$\mathrm{d}(\mathrm{x}, \mathrm{y}) \leq \mathrm{d}(\mathrm{x}, \mathrm{z}) + \mathrm{d}(\mathrm{z}, \mathrm{y})$$

Distance: Jaccard

Given two sets A, B

Jaccard similarity:

$$\mathrm{J}(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

$$\mathrm{d}(A, B) = 1 - \mathrm{J}(A, B)$$

Distance: Norms

Given $\mathrm{x} = [x_1, x_2, \ldots x_n]$
$\mathrm{y} = [y_1, y_2, \ldots y_n]$

Euclidean distance (L_2-norm) $$\mathrm{d}(\mathrm{x}, \mathrm{y}) = \sqrt{\sum_{i=0}^{n} (x_i - y_i)^2}$$

Manhattan distance (L_1-norm) $$\mathrm{d}(\mathrm{x}, \mathrm{y}) = \sum_{i=0}^{n} |x_i - y_i|$$

L_r-norm $$\mathrm{d}(\mathrm{x}, \mathrm{y}) = \left[\sum_{i=0}^{n} |x_i - y_i|^r\right]^{1/r}$$

Distance: Cosine

Given $$\mathrm{x} = [x_1, x_2, \ldots x_n]$$ $$\mathrm{y} = [y_1, y_2, \ldots y_n]$$

Idea: measure distance between the vectors

$$\cos\theta = \frac{\mathrm{x} \cdot \mathrm{y}}{|\mathrm{x}||\mathrm{y}|}$$

Thus:

$$\mathrm{sim}(\mathrm{x}, \mathrm{y}) = \frac{\sum_{i=0}^{n} x_i y_i}{\sqrt{\sum_{i=0}^{n} x_i^2}\sqrt{\sum_{i=0}^{n} y_i^2}}$$

$$\mathrm{d}(\mathrm{x}, \mathrm{y}) = 1 - \mathrm{sim}(\mathrm{x}, \mathrm{y})$$

Advantages over others?

Representations

Representations

(Text)

Unigrams (i.e., words)

Shingles = *n*-grams

At the word level

At the character level

Feature weights

boolean

tf.idf

BM25

…

Representations

(Beyond Text)

For recommender systems:

Items as features for users

Users as features for items

For graphs:

Adjacency lists as features for vertices

For log data:

Behaviors (clicks) as features

Clustering Algorithms

Agglomerative (bottom-up)

Divisive (top-down)

K-Means

Gaussian Mixture Models

Hierarchical Agglomerative Clustering

Start with each object in its own cluster

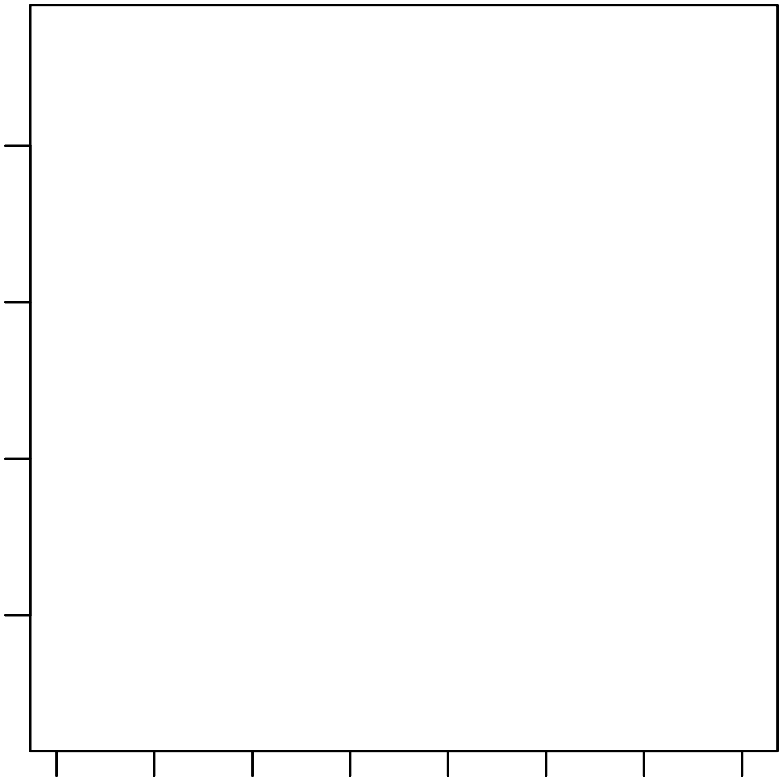
Until there is only one cluster:

Find the two clusters c_i and c_j, that are most similar

Replace c_i and c_j with a single cluster $c_i \cup c_j$

The history of merges forms the hierarchy

HAC in Action



Step 1: {1}, {2}, {3}, {4}, {5}, {6}, {7}

Step 2: {1}, {2, 3}, {4}, {5}, {6}, {7}

Step 3: {1, 7}, {2, 3}, {4}, {5}, {6}

Step 4: {1, 7}, {2, 3}, {4, 5}, {6}

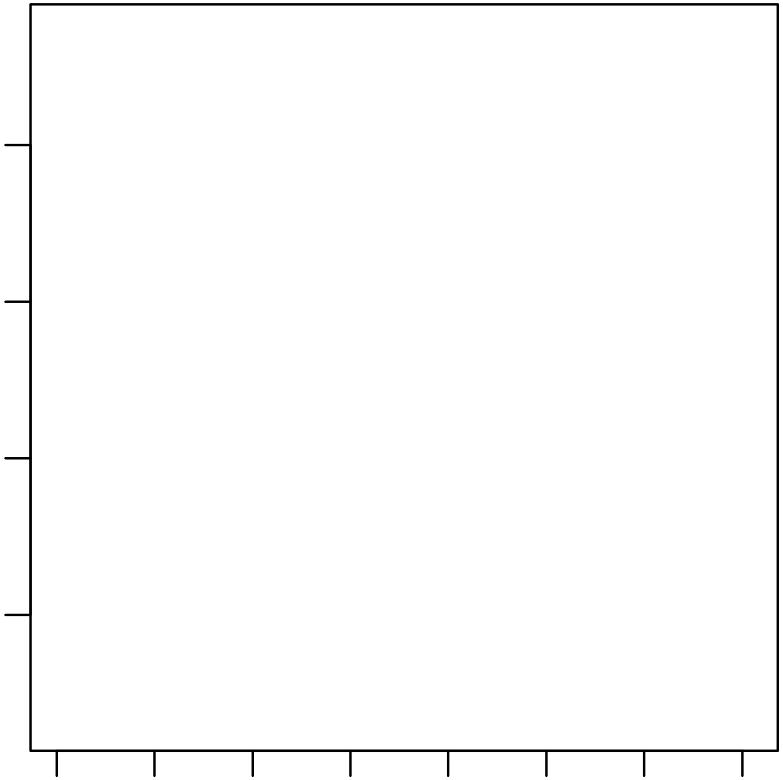
Step 5: {1, 7}, {2, 3, 6}, {4, 5}

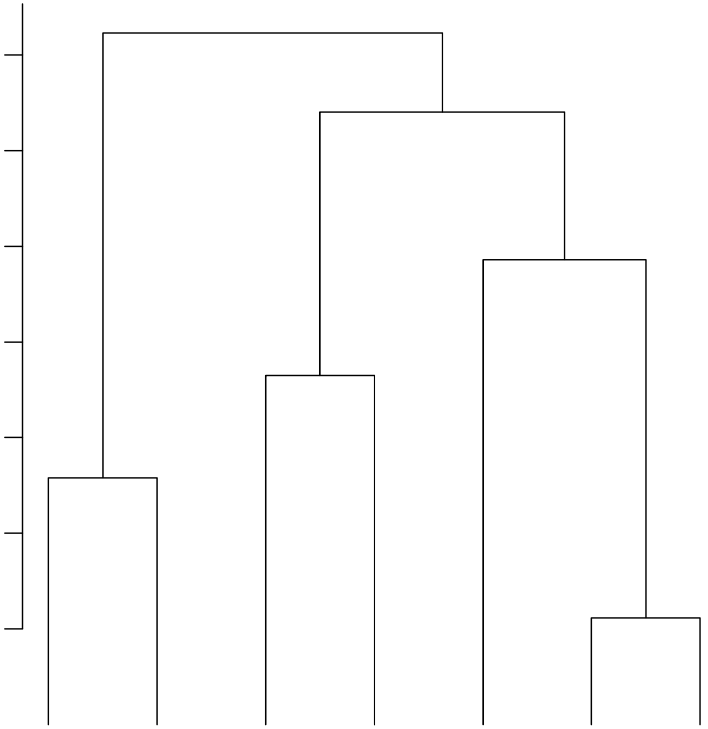
Step 6: {1, 7}, {2, 3, 4, 5, 6}

Step 7: {1, 2, 3, 4, 5, 6, 7}

Source: Slides by Ryan Tibshirani

Dendrogram





Source: Slides by Ryan Tibshirani

Cluster Merging

Which two clusters do we merge?

What's the similarity between two clusters?

Single Linkage: similarity of two most similar members

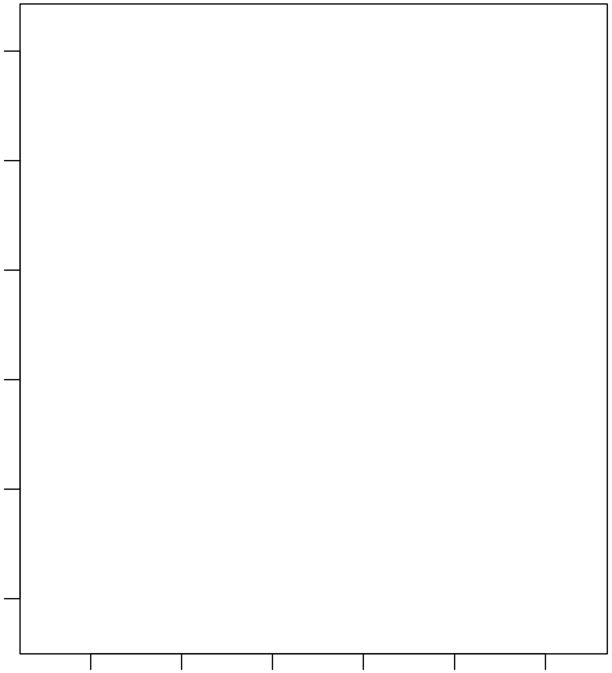
Complete Linkage: similarity of two least similar members

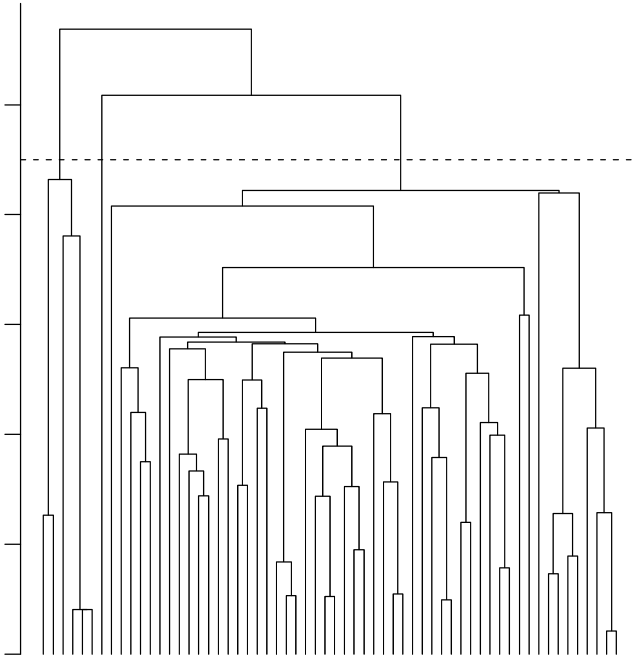
Average Linkage: average similarity between members

Single Linkage

Uses maximum similarity (min distance) of pairs:

$$d_{\text{single}}(G, H) = \min_{i \in G,\, j \in H} d_{ij}$$



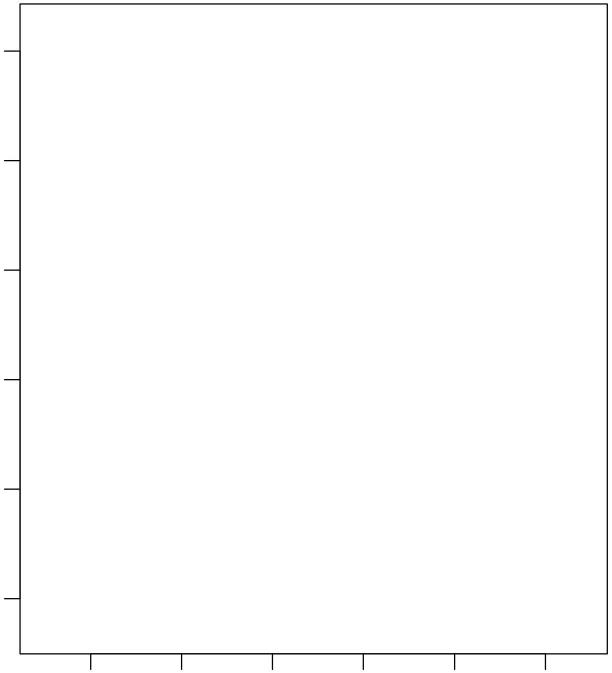


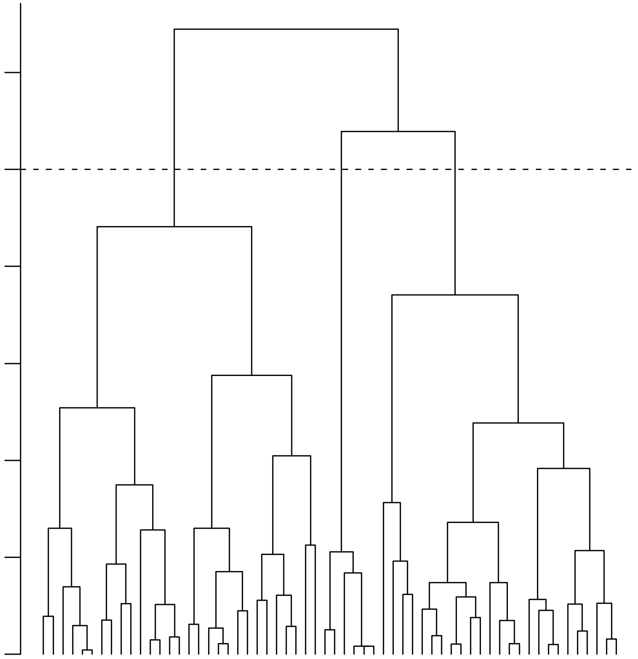
Source: Slides by Ryan Tibshirani

Complete Linkage

Uses minimum similarity (max distance) of pairs:

$$d_{\text{complete}}(G, H) = \max_{i \in G,\, j \in H} d_{ij}$$



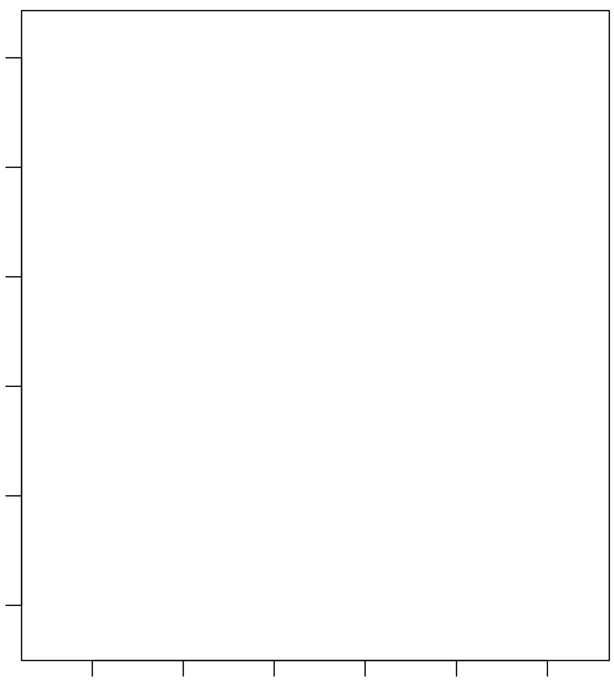


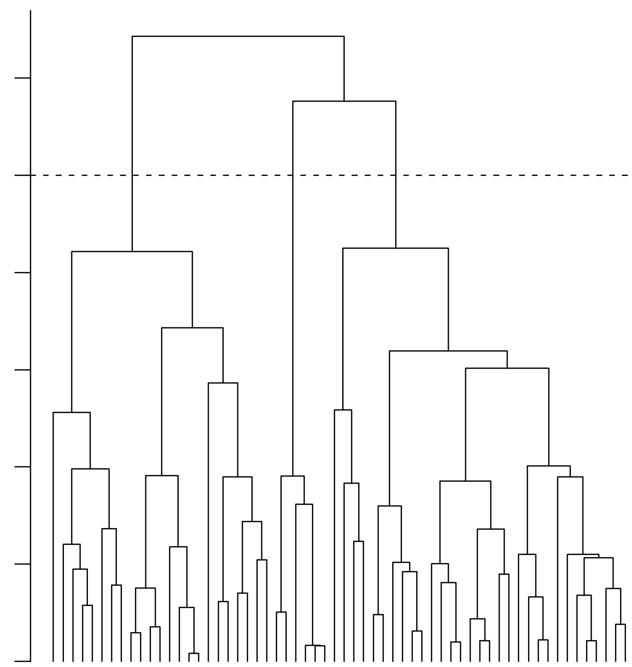
Source: Slides by Ryan Tibshirani

Average Linkage

Uses average of all pairs:

$$d_{\text{average}}(G,H) = \frac{1}{n_G \cdot n_H} \sum_{i \in G,\, j \in H} d_{ij}$$





Source: Slides by Ryan Tibshirani

Link Functions

Single linkage:

Uses maximum similarity (min distance) of pairs

Weakness: “straggly” (long and thin) clusters due to *chaining effect*

Clusters may not be compact

Complete linkage:

Uses minimum similarity (max distance) of pairs

Weakness: *crowding effect* – points closer to other clusters than own cluster

Clusters may not be far apart

Average linkage:

Uses average of all pairs

Tries to strike a balance – compact and far apart

Weakness: similarity more difficult to interpret

MapReduce Implementation

What's the inherent challenge?

Practicality as in-memory final step

Clustering Algorithms

Agglomerative (bottom-up)

Divisive (top-down)

K-Means

Gaussian Mixture Models

K-Means Algorithm

Select k random instances $\{s_1, s_2, \ldots s_k\}$ as initial centroids

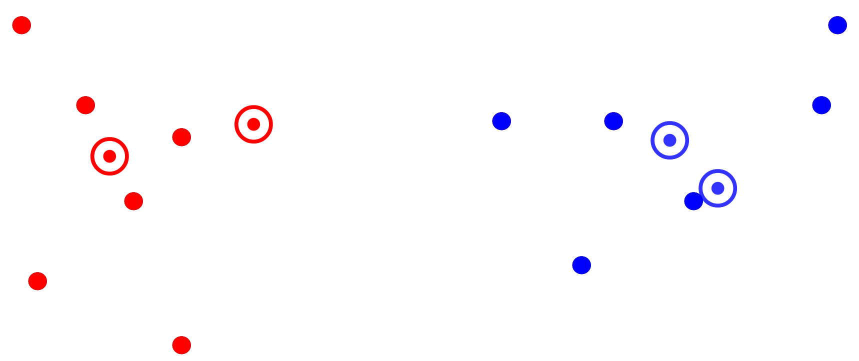
Iterate:

Assign each instance to closest centroid

Update centroids based on assigned instances

$$\mu(c) = \frac{1}{|c|} \sum_{\mathrm{x} \in c} \mathrm{x}$$

K-Means Clustering Example



Pick seeds

Reassign clusters

Compute centroids

Reassign clusters

Compute centroids

Reassign clusters

Converged!

Basic MapReduce Implementation

```
class Mapper  {
  def setup() = {
    clusters = loadClusters()
  }

  def map(id: Int, vector: Vector) = {
    emit(clusters.findNearest(vector), vector)
  }
}

class Reducer {
  def reduce(clusterId: Int, values: Iterable[Vector]) = {
    for (vector <- values) {
      sum += vector
      cnt += 1
    }
    emit(clusterId, sum/cnt)
  }
}
```

$$\mu(c) = \frac{1}{|c|} \sum_{\mathrm{x} \in c} \mathrm{x}$$

Basic MapReduce Implementation

Conceptually, what's happening?

Given current cluster assignment, assign each vector to closest cluster

Group by cluster

Compute updated clusters

What's the cluster update?

Computing the mean!

Remember IMC and other optimizations?

What about Spark?

Implementation Notes

Standard setup of iterative MapReduce algorithms

Driver program sets up MapReduce job

Waits for completion

Checks for convergence

Repeats if necessary

Must be able keep cluster centroids in memory

With large *k*, large feature spaces, potentially an issue

Memory requirements of centroids grow over time!

Variant: *k*-medoids

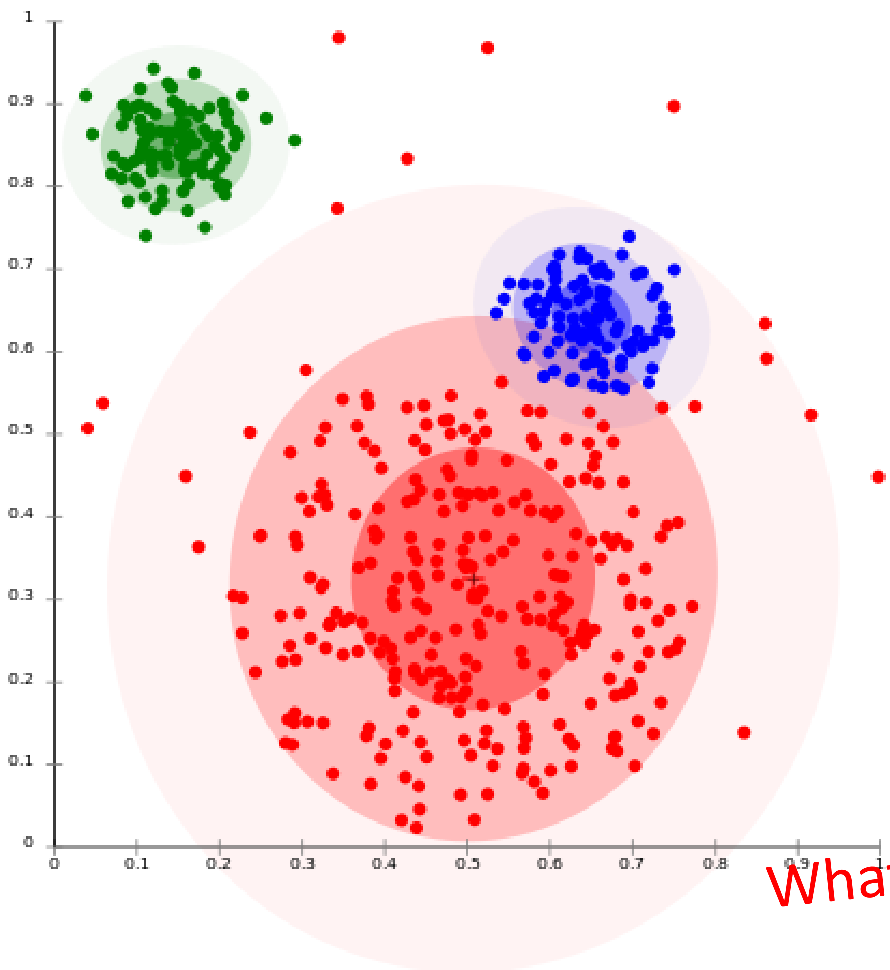
How do you select initial seeds?

How do you select *k*?

Clustering w/ Gaussian Mixture Models

Model data as a mixture of Gaussians

Given data, recover model parameters



What's with models?

Source: Wikipedia (Cluster analysis)

Gaussian Distributions

Univariate Gaussian (i.e., Normal):

$$p(x;\mu,\sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x-\mu)^2\right)$$

A random variable with such a distribution we write as:

$$x \sim \mathcal{N}(\mu, \sigma^2)$$

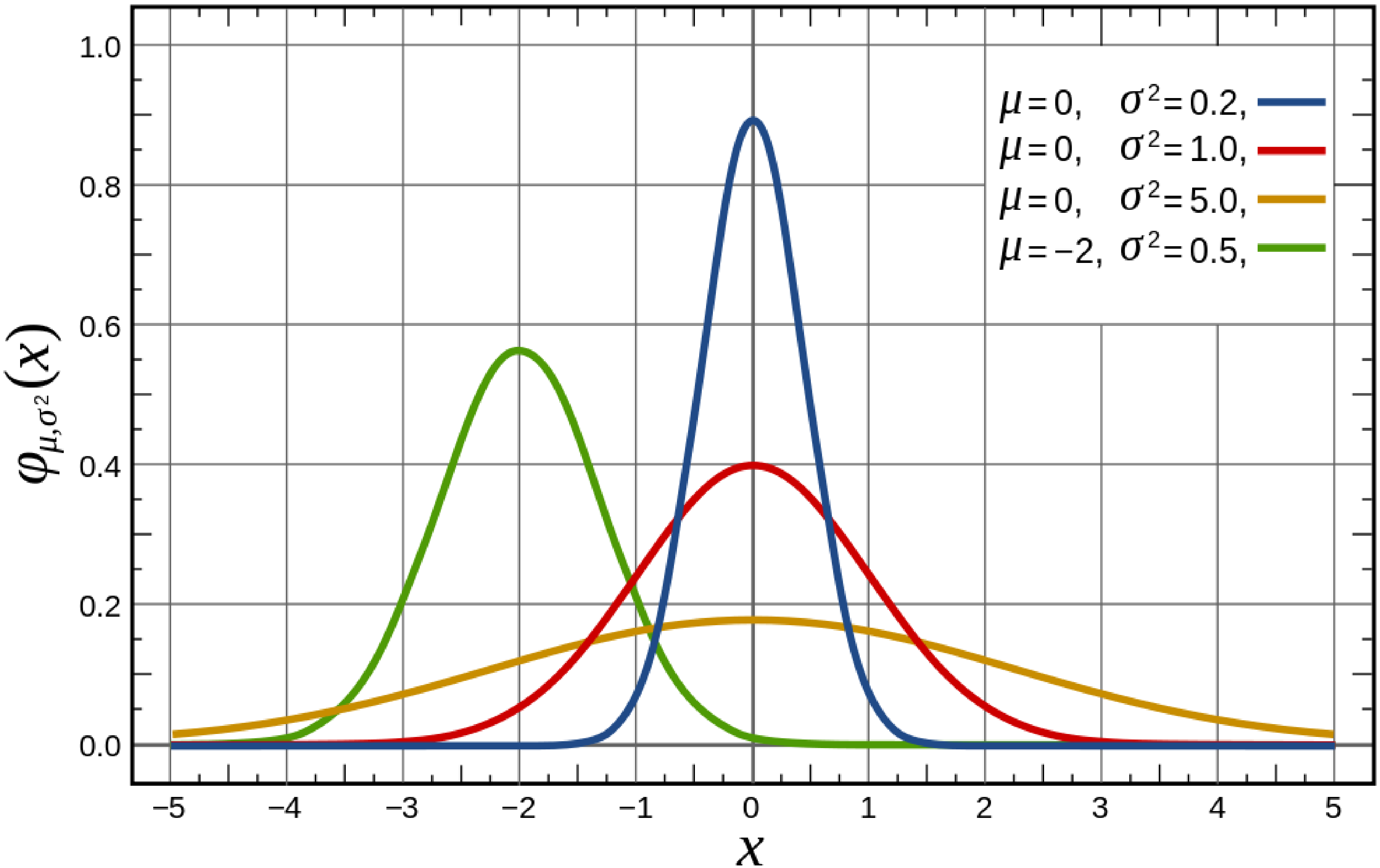
Multivariate Gaussian:

$$p(\mathrm{x};\mu,\Sigma) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\mathrm{x}-\mu)^T\Sigma^{-1}(\mathrm{x}-\mu)\right)$$

A random variable with such a distribution we write as:

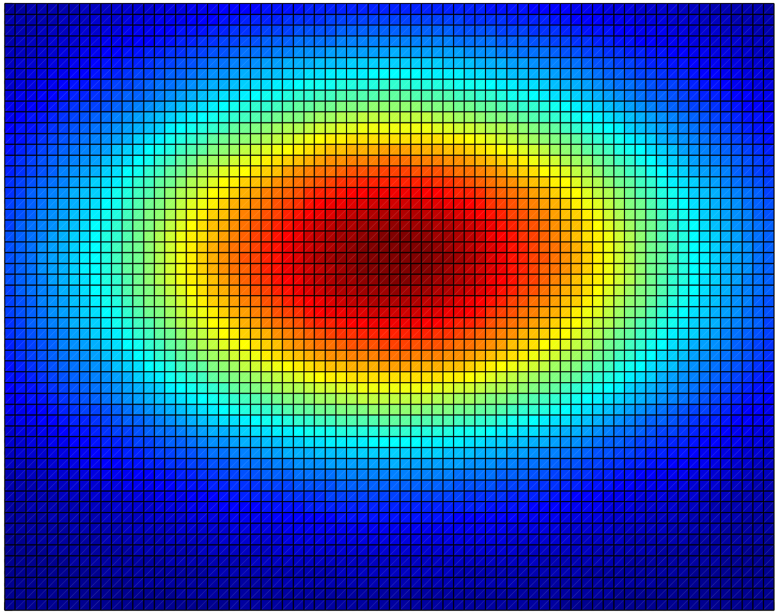
$$\mathrm{x} \sim \mathcal{N}(\mu, \Sigma)$$

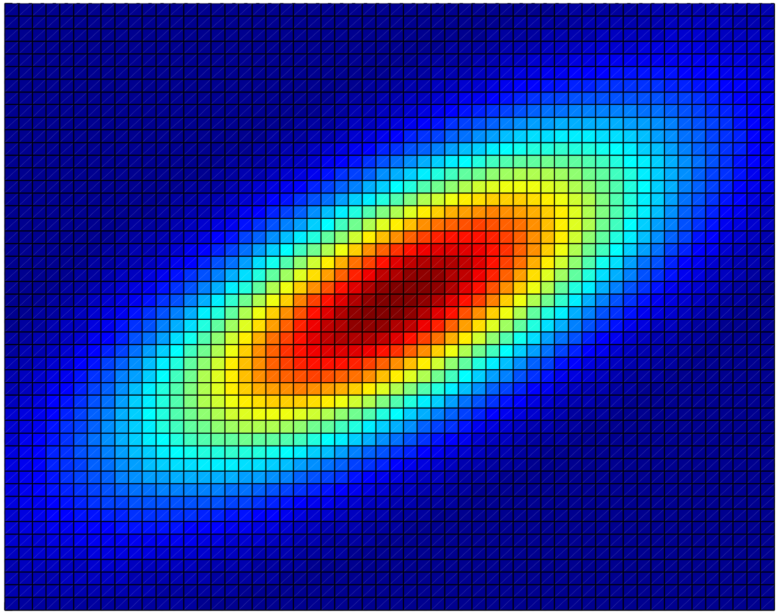
Univariate Gaussian



Source: Wikipedia (Normal Distribution)

Multivariate Gaussians





$$\mu = \begin{bmatrix} 3 \\ 2 \end{bmatrix} \quad \Sigma = \begin{bmatrix} 25 & 0 \\ 0 & 9 \end{bmatrix}$$

$$\mu = \begin{bmatrix} 3 \\ 2 \end{bmatrix} \quad \Sigma = \begin{bmatrix} 10 & 5 \\ 5 & 5 \end{bmatrix}$$

Source: Lecture notes by Chuong B. Do (IIT Delhi)

Gaussian Mixture Models

Model Parameters

Number of components: K

"Mixing" weight vector: π

For each Gaussian, mean and covariance matrix: $\mu_{1:K}$ $\Sigma_{1:K}$

The generative story?
(yes, that's a technical term)

Problem: Given the data, recover the model parameters

Varying constraints on co-variance matrices

Spherical vs. diagonal vs. full

Tied vs. untied

Learning for Simple Univariate Case

Problem setup:

Given number of components: K

Given points: $x_{1:N}$

Learn parameters: $\pi, \mu_{1:K}, \sigma^2_{1:K}$

Model selection criterion: maximize likelihood of data

Introduce indicator variables:

$$z_{n,k} = \begin{cases} 1 & \text{if } x_n \text{ is in cluster } k \\ 0 & \text{otherwise} \end{cases}$$

Likelihood of the data:

$$p(x_{1:N}, z_{1:N,1:K} | \mu_{1:K}, \sigma^2_{1:K}, \pi)$$

EM to the Rescue!

We're faced with this:

$$p(x_{1:N}, z_{1:N,1:K} | \mu_{1:K}, \sigma^2_{1:K}, \pi)$$

It'd be a lot easier if we knew the z's!

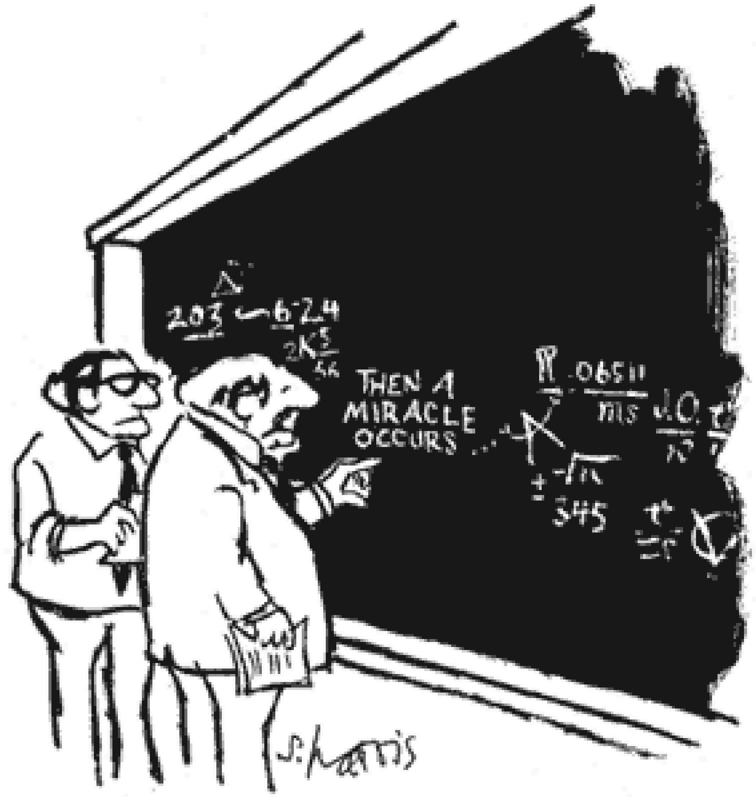
Expectation Maximization

Guess the model parameters

E-step: Compute posterior distribution over latent (hidden) variables given the model parameters

M-step: Update model parameters using posterior distribution computed in the E-step

Iterate until convergence



"I THINK YOU SHOULD BE MORE EXPLICIT HERE IN STEP TWO."

EM for Univariate GMMs

Initialize: $\pi, \mu_{1:K}, \sigma^2_{1:K}$

Iterate:

E-step: compute expectation of *z* variables

$$\mathbb{E}[z_{n,k}] = \frac{\mathcal{N}(x_n | \mu_k, \sigma^2_k) \cdot \pi_k}{\sum_{k'} \mathcal{N}(x_n | \mu_{k'}, \sigma^2_{k'}) \cdot \pi_{k'}}$$

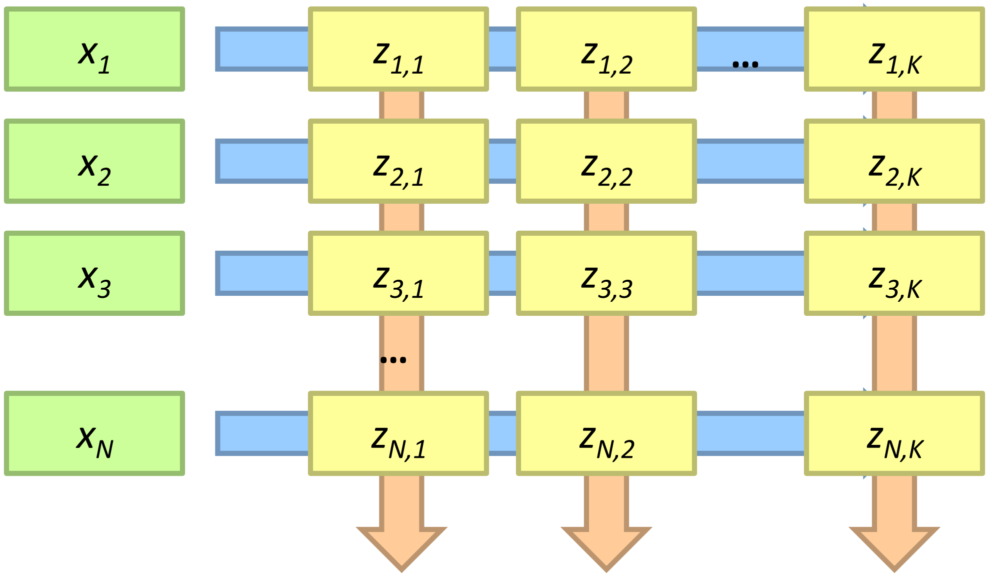
M-step: compute new model parameters

$$\pi_k = \frac{1}{N} \sum_n z_{n,k}$$

$$\mu_k = \frac{1}{\sum_n z_{n,k}} \sum_n z_{n,k} \cdot x_n$$

$$\sigma^2_k = \frac{1}{\sum_n z_{n,k}} \sum_n z_{n,k} ||x_n - \mu_k||^2$$

MapReduce Implementation



Map

$$\mathbb{E}[z_{n,k}] = \frac{\mathcal{N}(x_n|\mu_k, \sigma_k^2) \cdot \pi_k}{\sum_{k'} \mathcal{N}(x_n|\mu_{k'}, \sigma_{k'}^2) \cdot \pi_{k'}}$$

Reduce

$$\pi_k = \frac{1}{N} \sum_n z_{n,k}$$

$$\mu_k = \frac{1}{\sum_n z_{n,k}} \sum_n z_{n,k} \cdot x_n$$

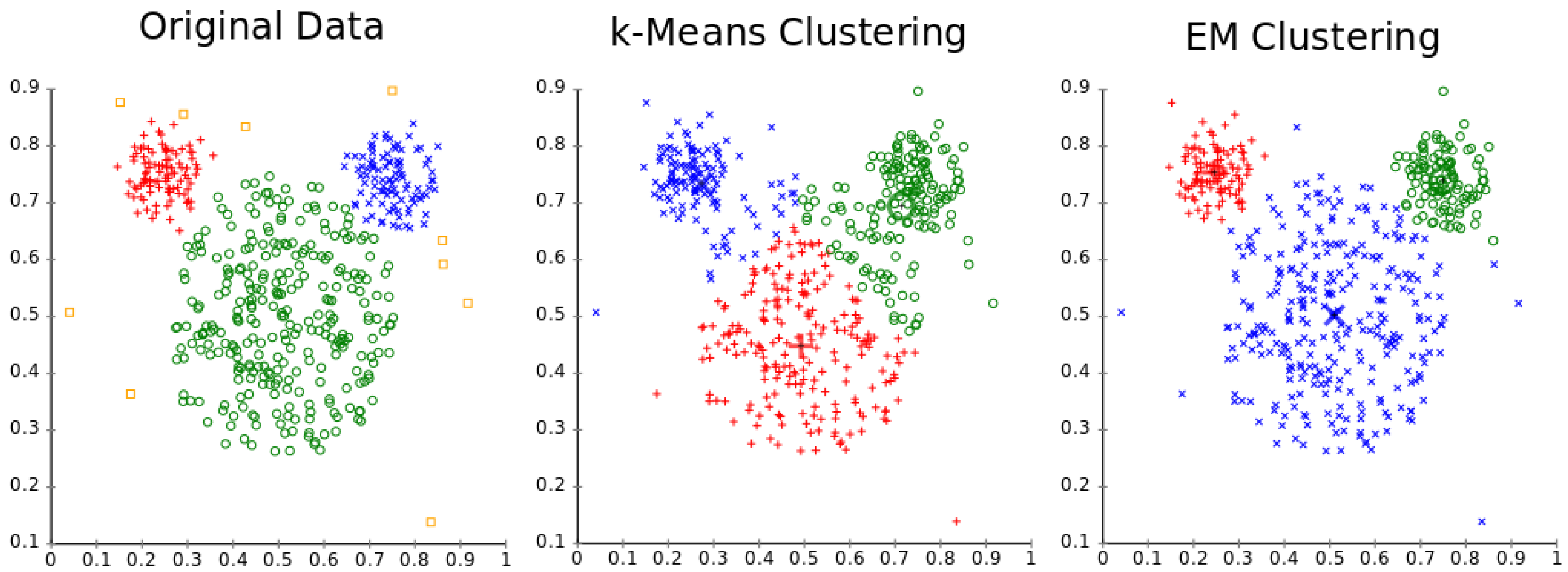
$$\sigma_k^2 = \frac{1}{\sum_n z_{n,k}} \sum_n z_{n,k} ||x_n - \mu_k||^2$$

What about Spark?

K-Means vs. GMMs

	K-Means	GMM
Map	Compute distance of points to centroids	E-step: compute expectation of *z* indicator variables
Reduce	Recompute new centroids	M-step: update values of model parameters

Different cluster analysis results on "mouse" data set:



Source: Wikipedia (*k*-means clustering)


Source: Wikipedia (Japanese rock garden)