



# Big Data Infrastructure

CS 489/698 Big Data Infrastructure (Winter 2017)

Week 9: Data Mining (4/4)

March 9, 2017

Jimmy Lin

David R. Cheriton School of Computer Science  
University of Waterloo

These slides are available at <http://lintool.github.io/bigdata-2017w/>



This work is licensed under a Creative Commons Attribution-Noncommercial-Share Alike 3.0 United States  
See <http://creativecommons.org/licenses/by-nc-sa/3.0/us/> for details

# Structure of the Course

Analyzing Text

Analyzing Graphs

Analyzing  
Relational Data

Data Mining

“Core” framework features  
and algorithm design

# 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

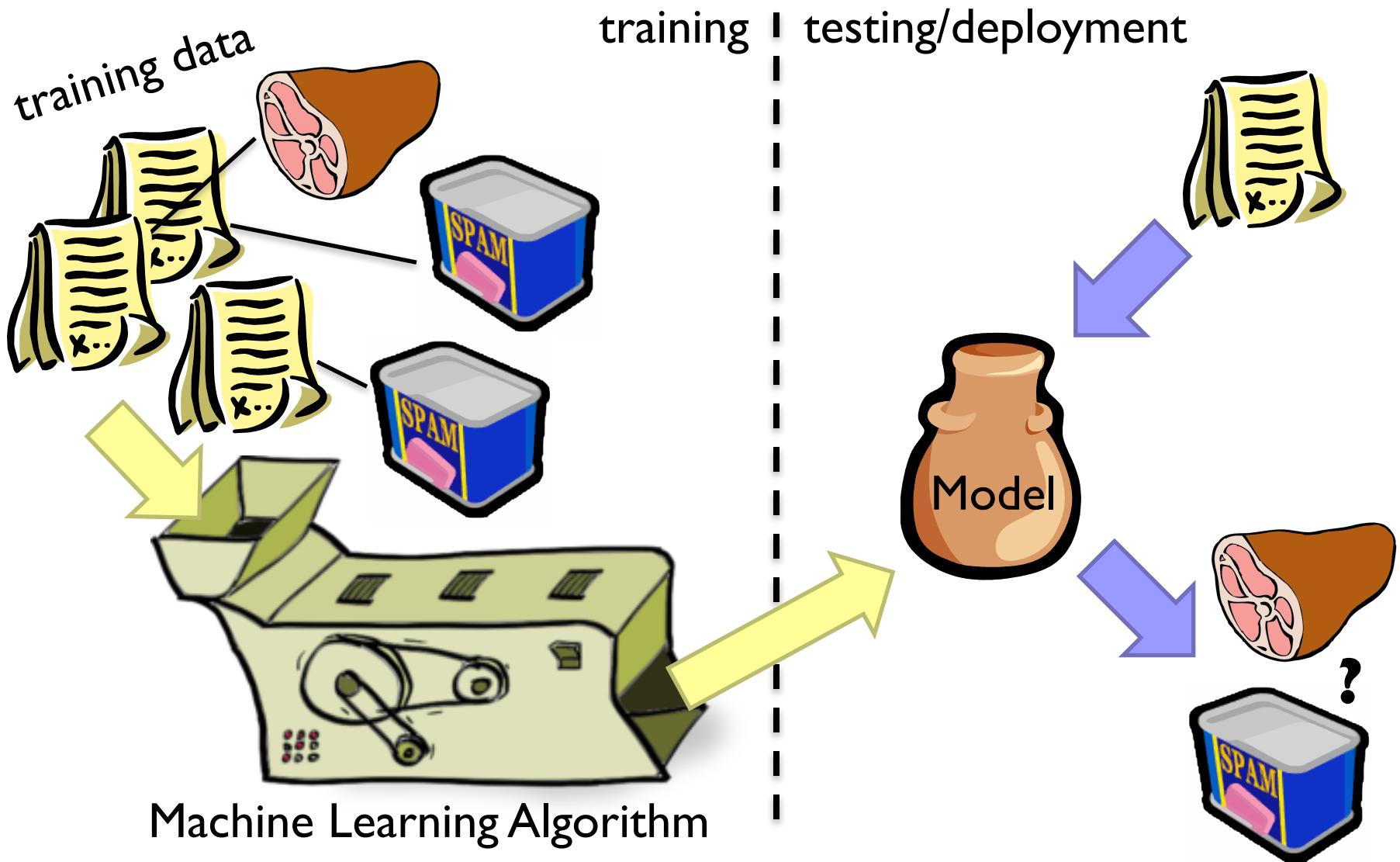
How to form clusters?

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

# Clustering



# Clustering

Specify distance metric  
Jaccard, Euclidean, cosine, etc.

Compute representation  
Shingling, tf.idf, etc.

Apply clustering algorithm

# Distance Metrics



# Distance Metrics

1. Non-negativity:

$$d(x, y) \geq 0$$

2. Identity:

$$d(x, y) = 0 \iff x = y$$

3. Symmetry:

$$d(x, y) = d(y, x)$$

4. Triangle Inequality

$$d(x, y) \leq d(x, z) + d(z, y)$$

# Distance: Jaccard

Given two sets A, B

Jaccard similarity:

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

$$d(A, B) = 1 - J(A, B)$$

# Distance: Norms

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

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

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

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

# Distance: Cosine

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

Idea: measure distance between the vectors

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

Thus:

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

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

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

Hierarchical  
K-Means  
Gaussian Mixture Models

# Hierarchical Agglomerative Clustering

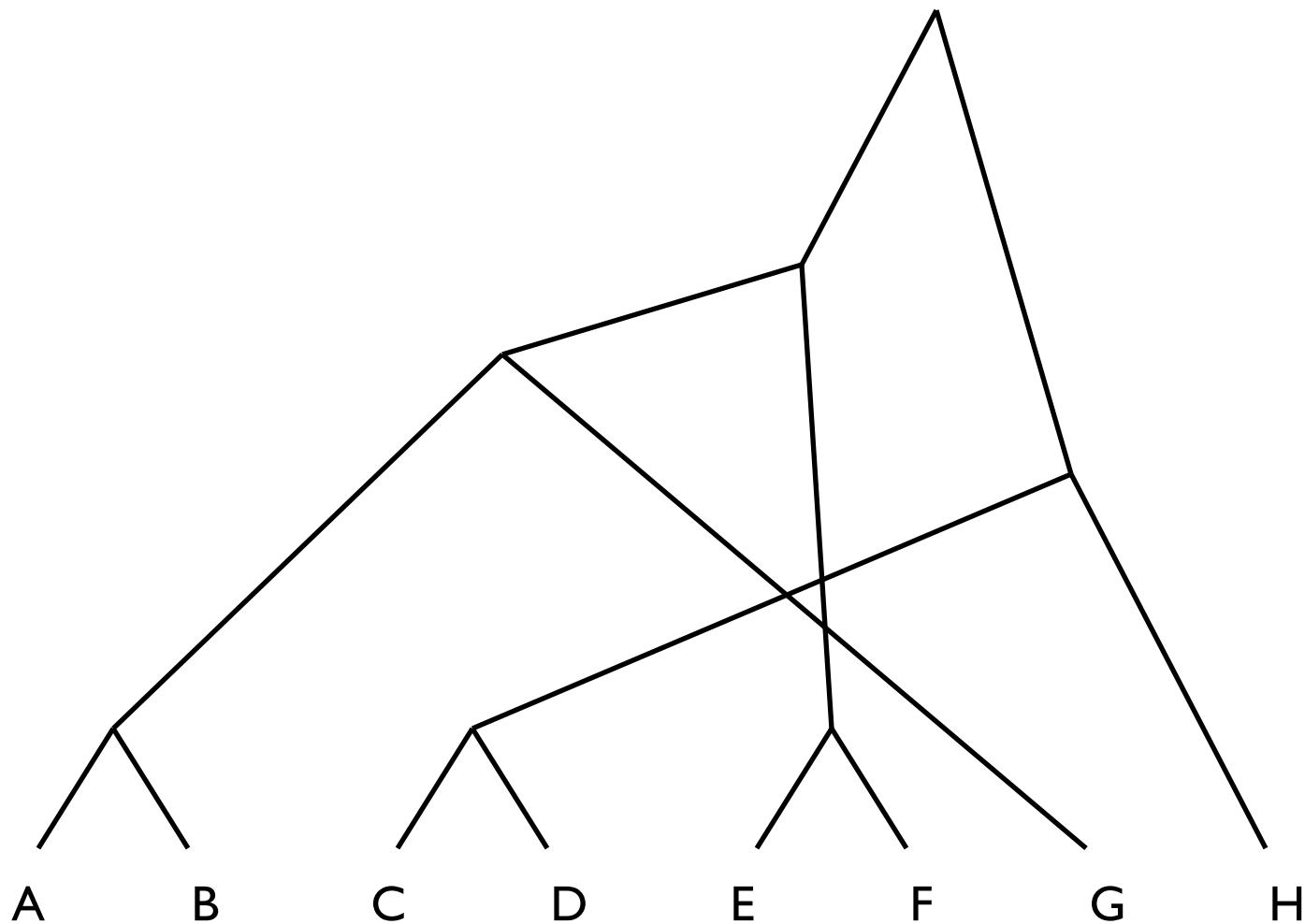
Start with each document in its own cluster

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



# Cluster Merging

Which two clusters do we merge?

What's the similarity between two clusters?

Single Link: similarity of two most similar members

Complete Link: similarity of two least similar members

Group Average: average similarity between members

# Link Functions

Single link:

Uses maximum similarity of pairs:

$$\text{sim}(c_i, c_j) = \max_{x \in c_i, y \in c_j} \text{sim}(x, y)$$

Can result in “straggly” (long and thin) clusters due to *chaining effect*

Complete link:

Use minimum similarity of pairs:

$$\text{sim}(c_i, c_j) = \min_{x \in c_i, y \in c_j} \text{sim}(x, y)$$

Makes more “tight” spherical clusters

# MapReduce Implementation

What's the inherent challenge?  
Practicality as in-memory final step

# K-Means Algorithm

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

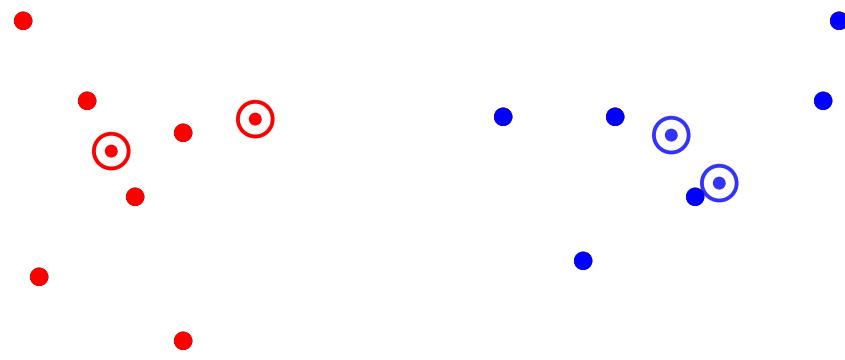
Iterate:

Assign each instance to closest centroid

Update centroids based on assigned instances

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

# K-Means Clustering Example



Pick seeds

Reassign clusters

Compute centroids

Reassign clusters

Compute centroids

Reassign clusters

Converged!

# Basic MapReduce Implementation

```
1: class MAPPER
2:   method CONFIGURE()
3:     c ← LOADCLUSTERS()
4:   method MAP(id i, point p)
5:     n ← NEARESTCLUSTERID(clusters c, point p)
6:     p ← EXTENDPOINT(point p) ← (Just a clever way to keep
7:     EMIT(clusterid n, point p) track of denominator)
1: class REDUCER
2:   method REDUCE(clusterid n, points [p1, p2, ...])
3:     s ← INITPOINTSUM()
4:     for all point p ∈ points do
5:       s ← s + p
6:     m ← COMPUTECENTROID(point s)
7:     EMIT(clusterid n, centroid m)
```

# MapReduce Implementation w/ IMC

```
1: class MAPPER
2:   method CONFIGURE()
3:      $c \leftarrow \text{LOADCLUSTERS}()$ 
4:      $H \leftarrow \text{INITASSOCIATIVEARRAY}()$ 
5:   method MAP(id  $i$ , point  $p$ )
6:      $n \leftarrow \text{NEARESTCLUSTERID}(\text{clusters } c, \text{ point } p)$ 
7:      $p \leftarrow \text{EXTENDPOINT}(\text{point } p)$ 
8:      $H\{n\} \leftarrow H\{n\} + p$ 
9:   method CLOSE()
10:    for all clusterid  $n \in H$  do
11:      EMIT(clusterid  $n$ , point  $H\{n\}$ )
1: class REDUCER
2:   method REDUCE(clusterid  $n$ , points  $[p_1, p_2, \dots]$ )
3:      $s \leftarrow \text{INITPOINTSUM}()$ 
4:     for all point  $p \in \text{points}$  do
5:        $s \leftarrow s + p$ 
6:      $m \leftarrow \text{COMPUTECENTROID}(\text{point } s)$ 
7:     EMIT(clusterid  $n$ , centroid  $m$ )
```

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

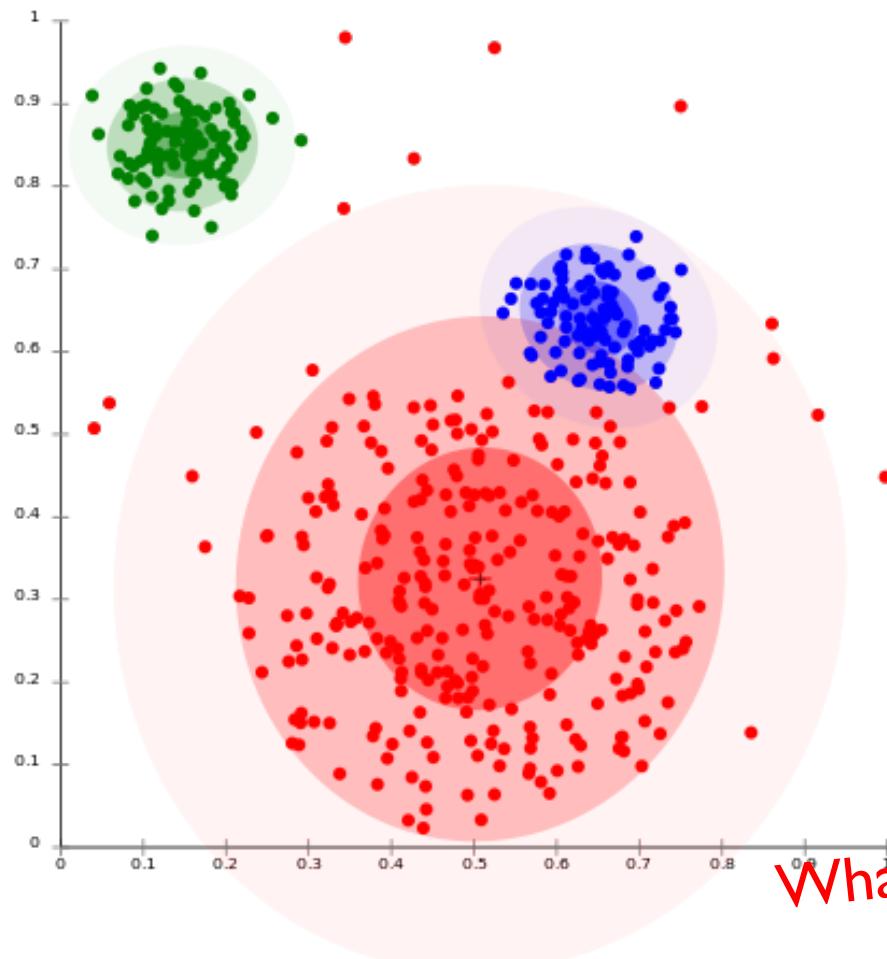
How do you select initial seeds?

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?

# 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:

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

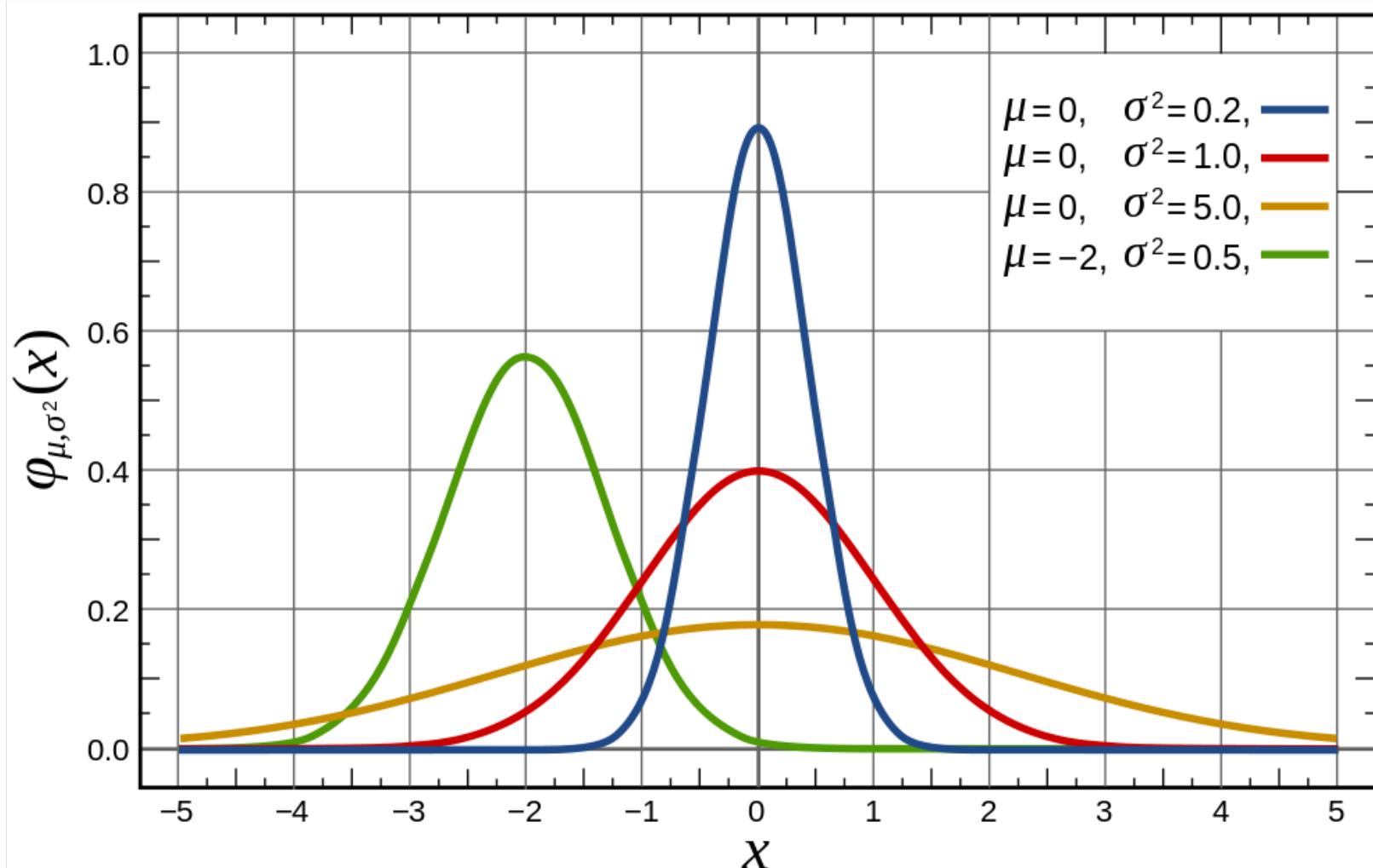
**Multivariate Gaussian:**

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

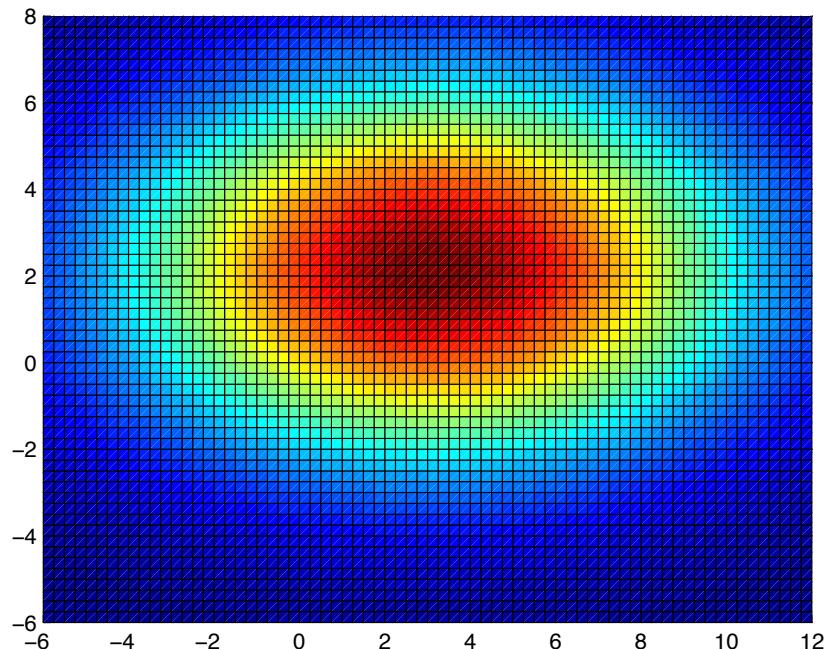
A random variable with such a distribution we write as:

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

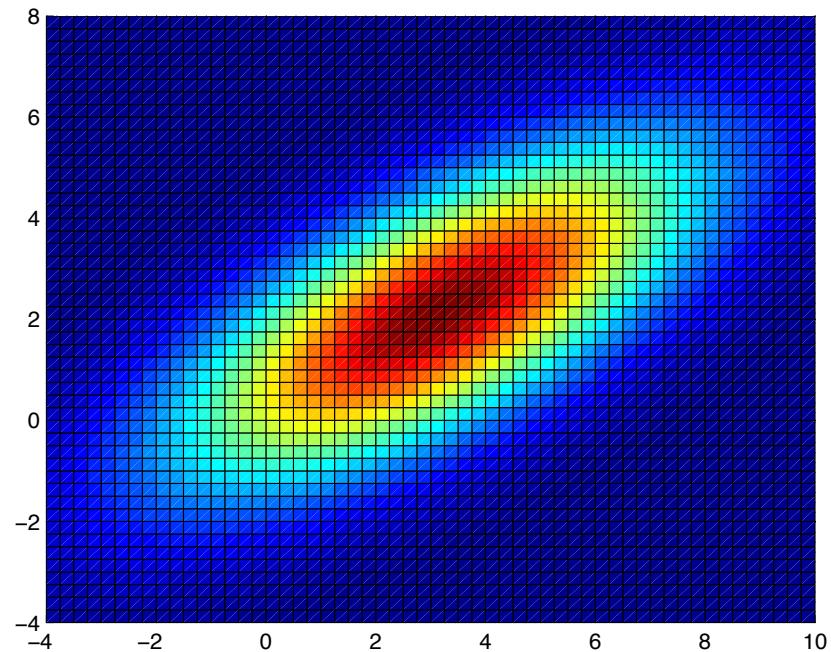
# Univariate Gaussian



# Multivariate Gaussians



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



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

# Gaussian Mixture Models

## Model Parameters

Number of components:  $K$

“Mixing” weight vector:  $\pi$

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:  $\pi, \mu_{1:K}, \sigma_{1:K}^2$

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:

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

# EM to the Rescue!

We're faced with this:

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

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

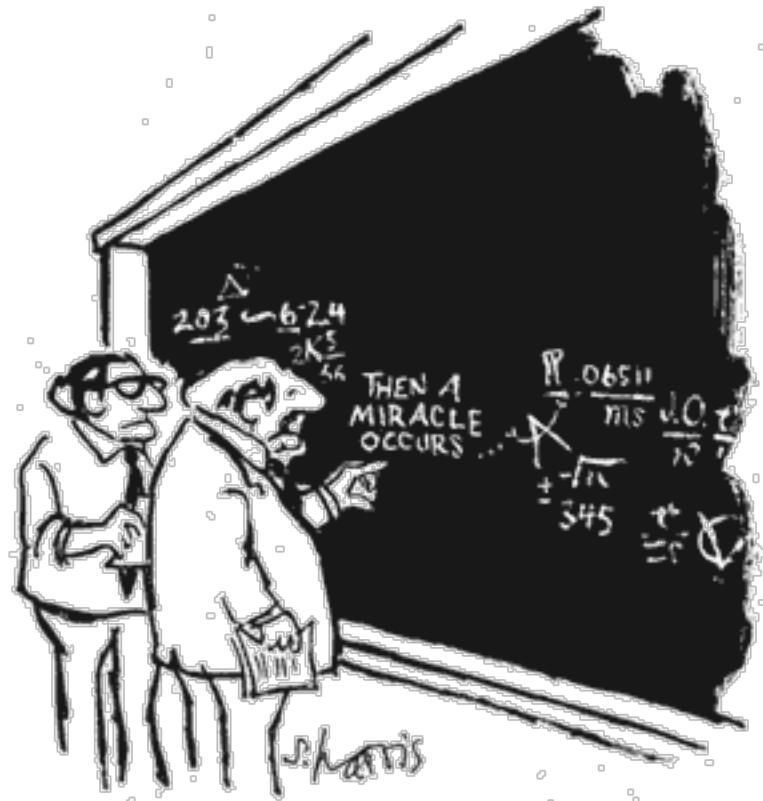
Expectation Maximization

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:**  $\pi, \mu_{1:K}, \sigma_{1:K}^2$

**Iterate:**

E-step: compute expectation of z variables

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

M-step: compute new model parameters

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

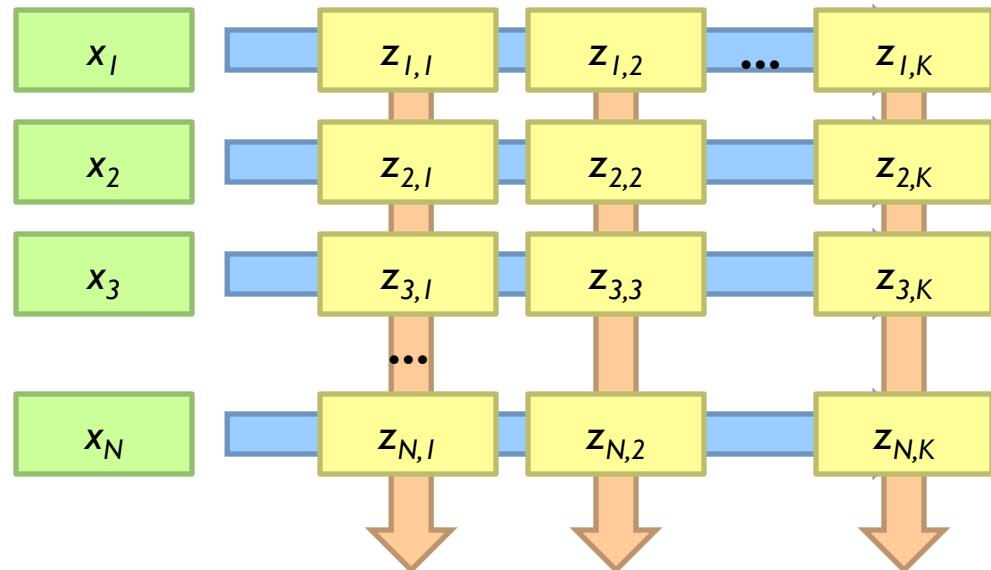
$$\mu_k = \frac{1}{\sum_n z_{n,k}} \sum_n z_{n,k} \cdot x_n$$

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

# 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

Map

K-Means

GMM

Reduce

Compute distance of  
points to centroids

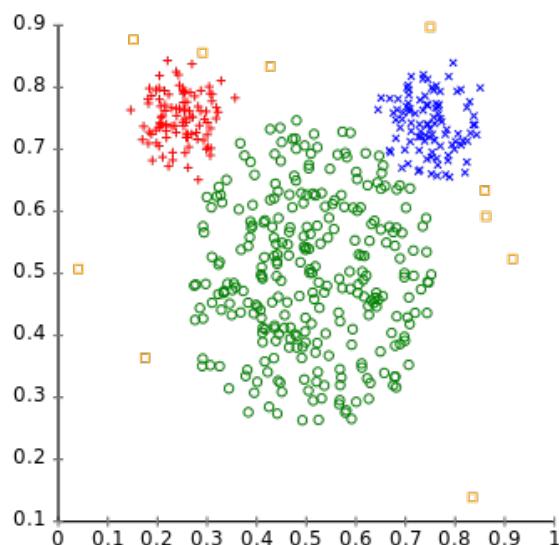
E-step: compute expectation  
of z indicator variables

Recompute new  
centroids

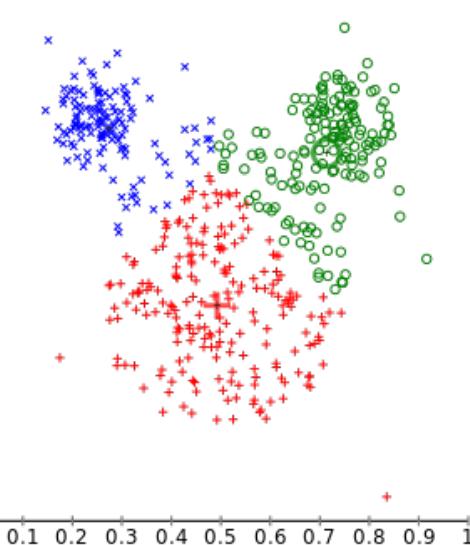
M-step: update values of  
model parameters

## Different cluster analysis results on "mouse" data set:

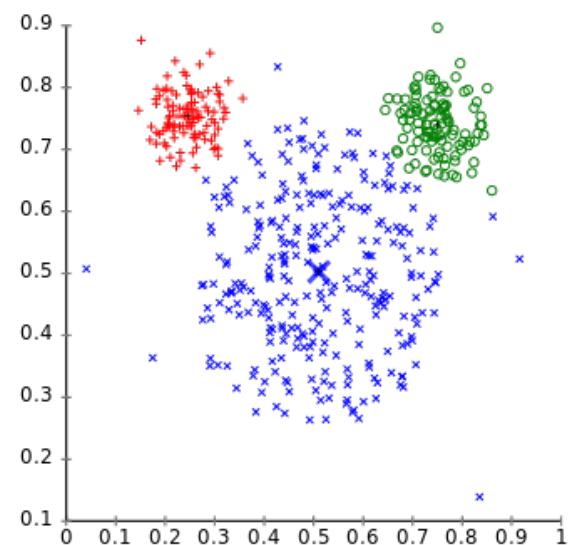
Original Data



k-Means Clustering



EM Clustering



A photograph of a traditional Japanese rock garden. In the foreground, a gravel path is raked into fine, parallel lines. Several large, dark, irregular stones are scattered across the garden. A small, shallow pond is visible in the middle ground, surrounded by more stones and some low-lying green plants. In the background, there are several traditional wooden buildings with dark, tiled roofs. The overall atmosphere is serene and minimalist.

# Questions?