01.02.2019

Statistical Methods in AI (CSE/ECE 471)

Lecture-8:

- Unsupervised Learning
- Clustering: k-means, GMM, Hierarchical methods

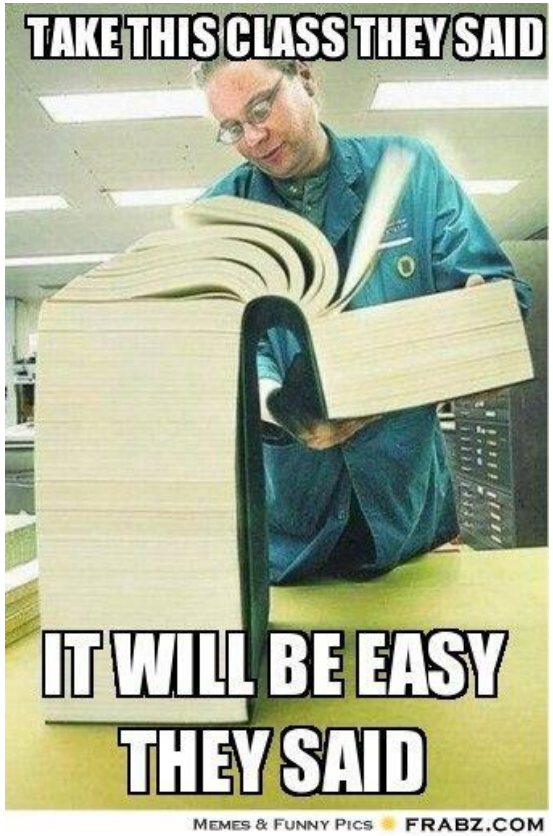
Ravi Kiran

Center for Visual Information Technology (CVIT), IIIT Hyderabad

Announcements

- A2: Questions on Logistic Regression → A3
- SMAI Mid-1 will be on Feb 7 (Thursday)
  - Syllabus: Lec 1 – Lec 8 (up to today's lecture)
- Tutorial tomorrow: Logistic Regression (parameter learning), Linear Algebra, Probability
- Assignment Schedule will be up on Moodle

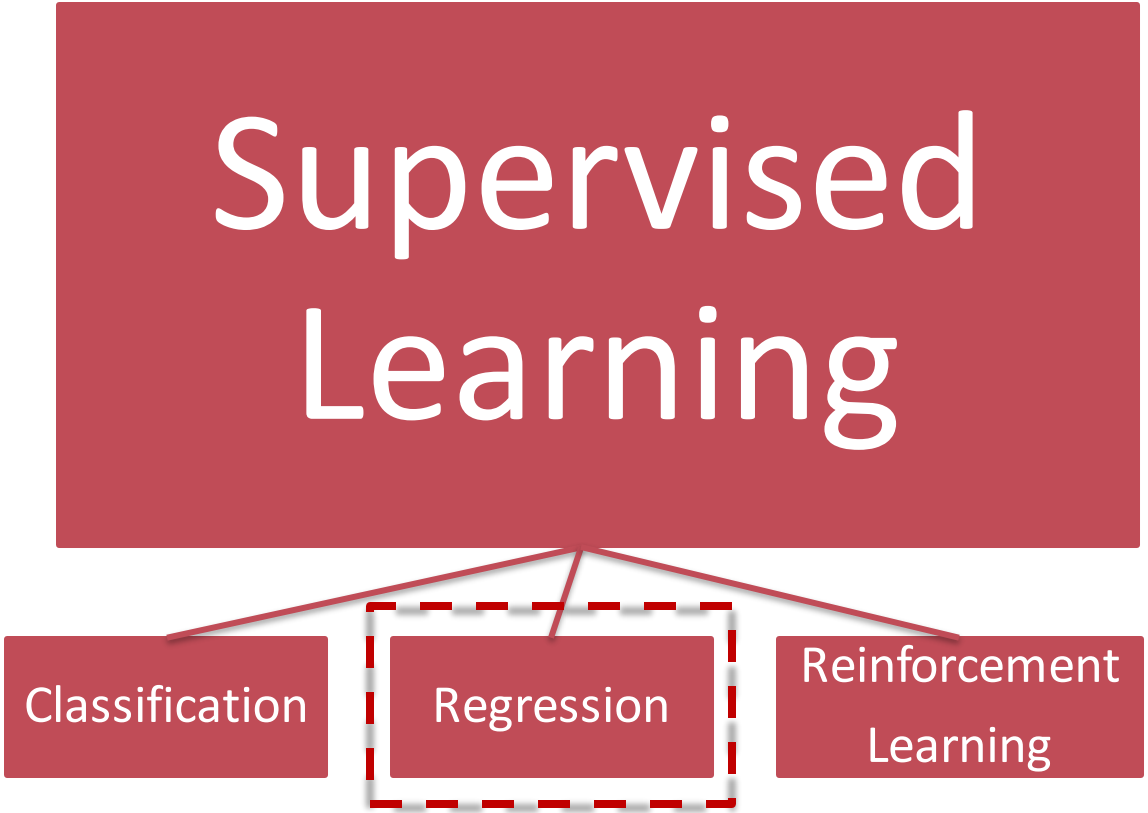
About the course – Grading Policy



(Revised) Grading Scheme

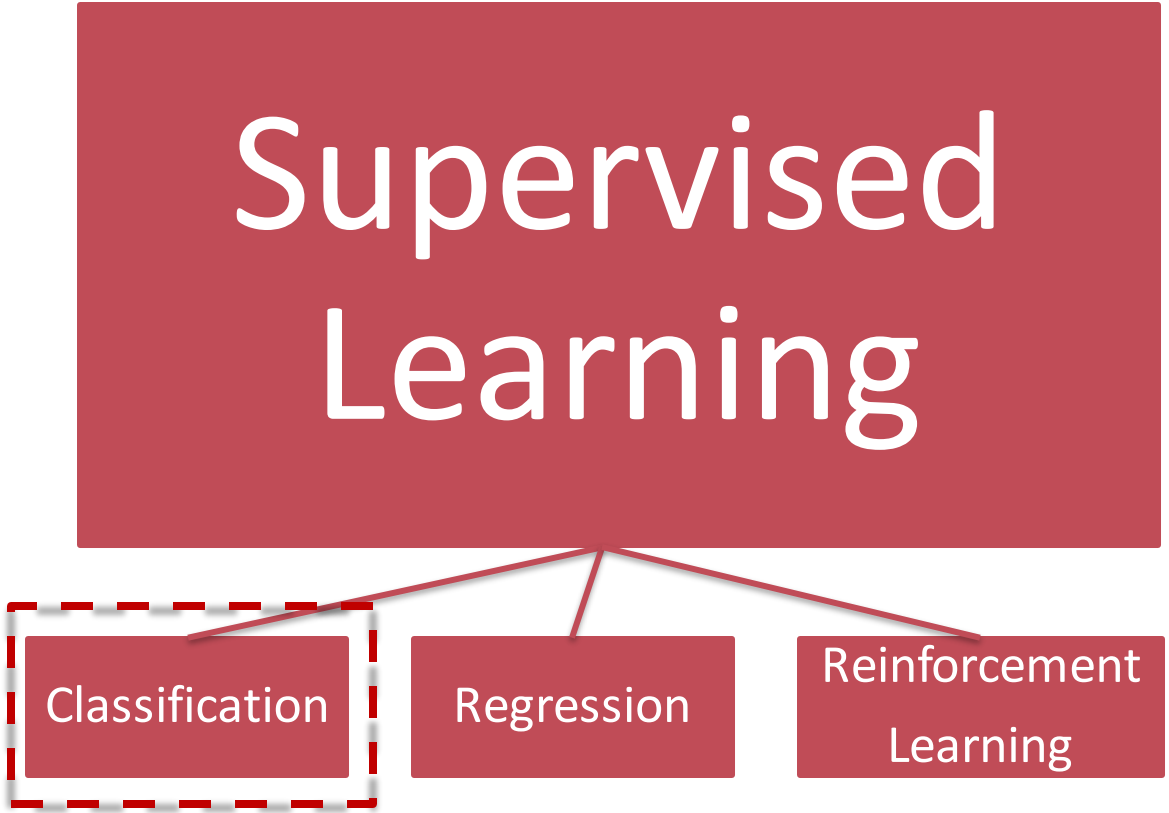
Revised

Assignments	30%	10 * 3	26%	13 * 2
Mid Terms	25%	12.5 * 2	30%	15 * 2
Project	20%	20 * 1	14%	14 * 1
Finals	25%	25 * 1	30%	30 * 1


Supervised Learning
Classification
Regression
Reinforcement Learning

Regression (revisiting k-NN)

- k-NN regression


Supervised Learning
Classification
Regression
Reinforcement Learning

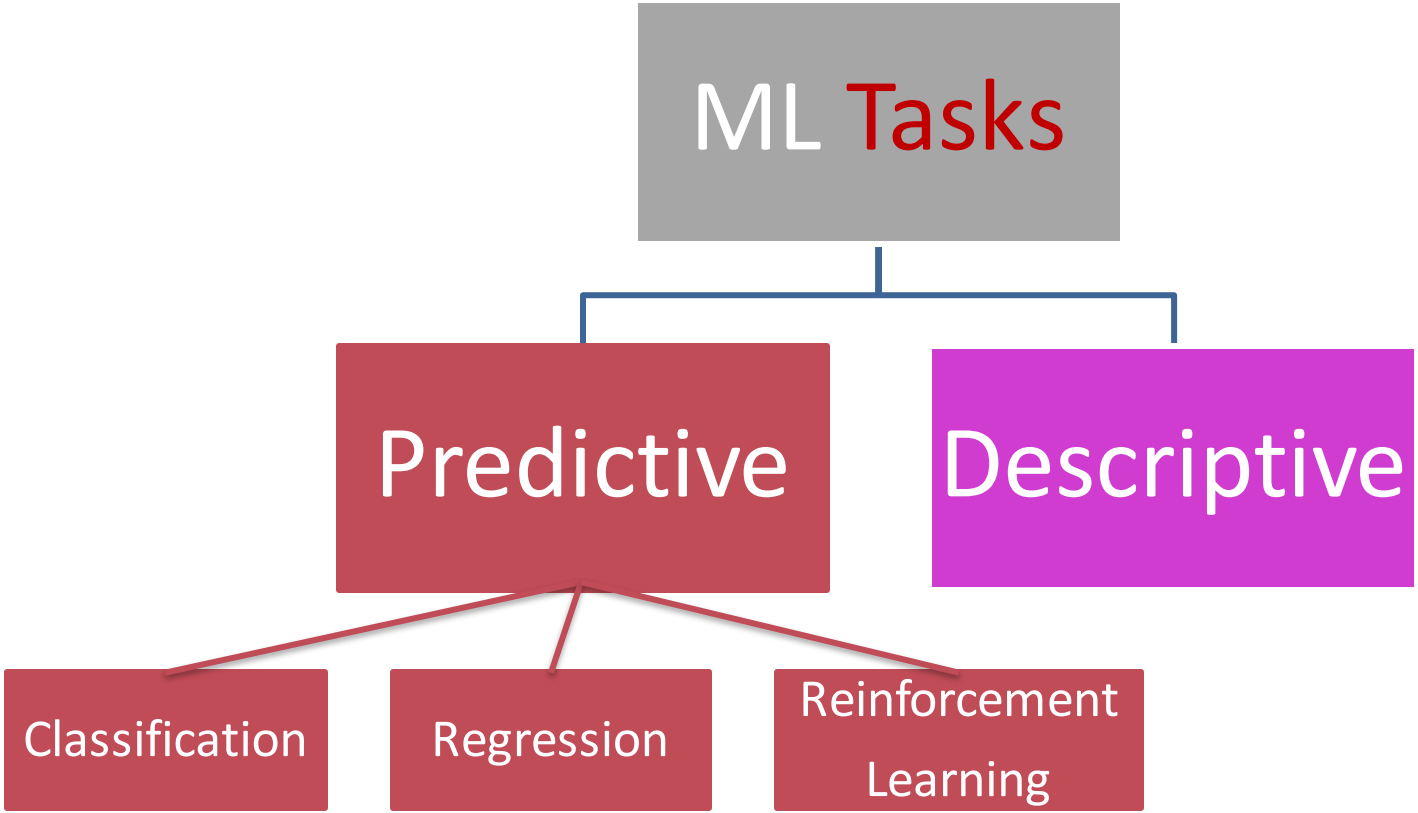
Taxonomy of classifiers : a modelling perspective

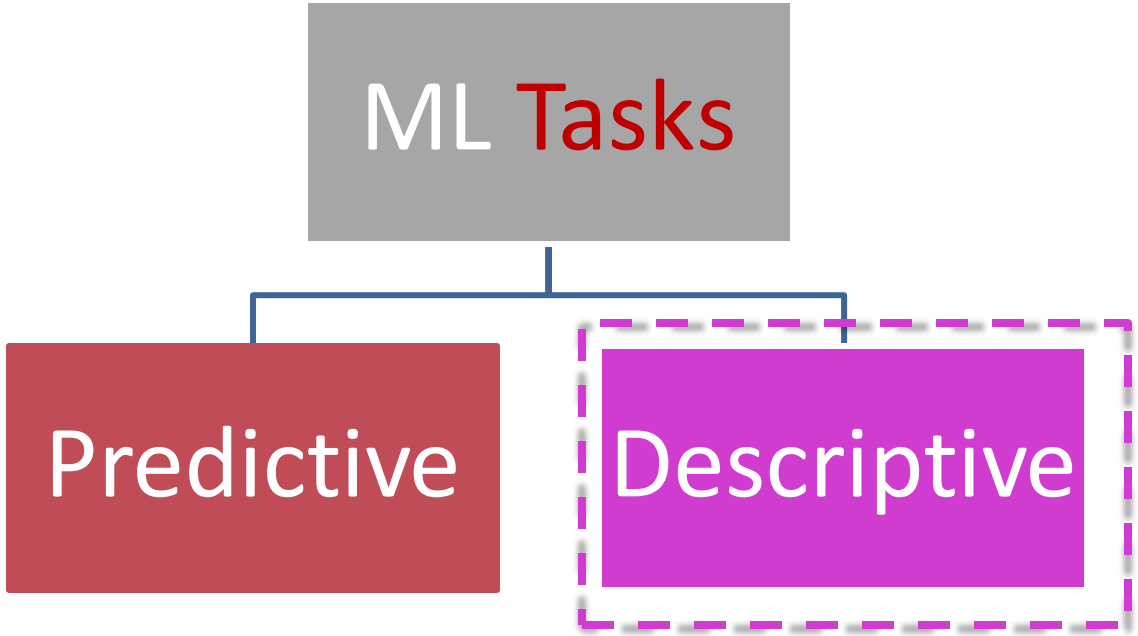
	Probabilistic	Non-Probabilistic
Discriminative	• Logistic Regression • Probabilistic neural nets • ……..	• K-nn • Linear classifier • SVM • Neural networks • ……
Generative	• Naïve Bayes • Model-based (e.g., GMM) • ……	

Model $p(class = C|x)$

Model the classification rule directly

Model $p(x\,|class = C)$


ML Tasks
Predictive
Descriptive
Classification
Regression
Reinforcement Learning


ML Tasks
Predictive
Descriptive

ML::Tasks → Descriptive

- Study/Exploit the ‘structure’ of data
  - Density Estimation
  - Clustering
  - Dimensionality Reduction
- Also studied as ‘Unsupervised Learning’
  - ‘Input’ data without paired ‘Output’

Unsupervised Learning → Density Estimation

Aka "learning without a teacher"

Feature Space $\mathcal{X}$

Word distribution
(Probability of a word)

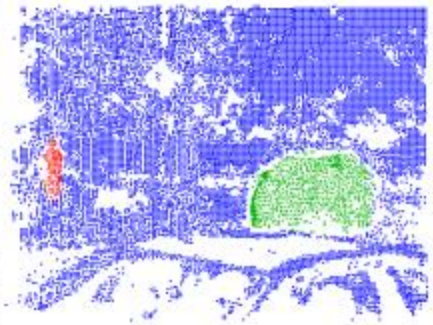
Task: Given $X \in \mathcal{X}$, learn $f(X)$.

Unsupervised Learning → Clustering

Group similar things e.g. images

[Goldberger et al.]

- Determine groups of people in image above
  - based on clothing styles
  - gender, age, etc

- Determine moving objects in videos

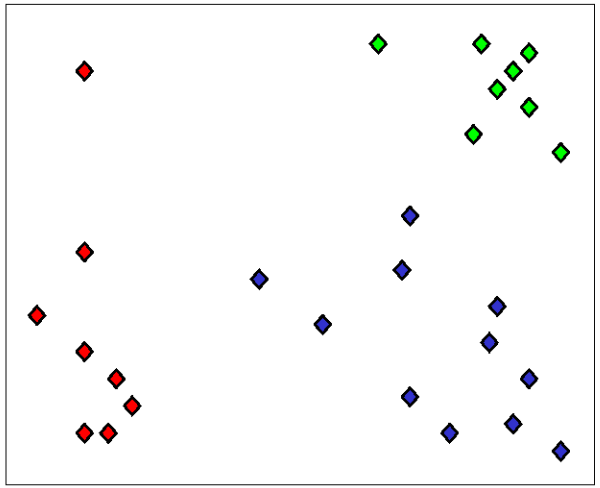
What is Clustering?

• Organizing data into *clusters* such that there is

  • high intra-cluster similarity

  • low inter-cluster similarity

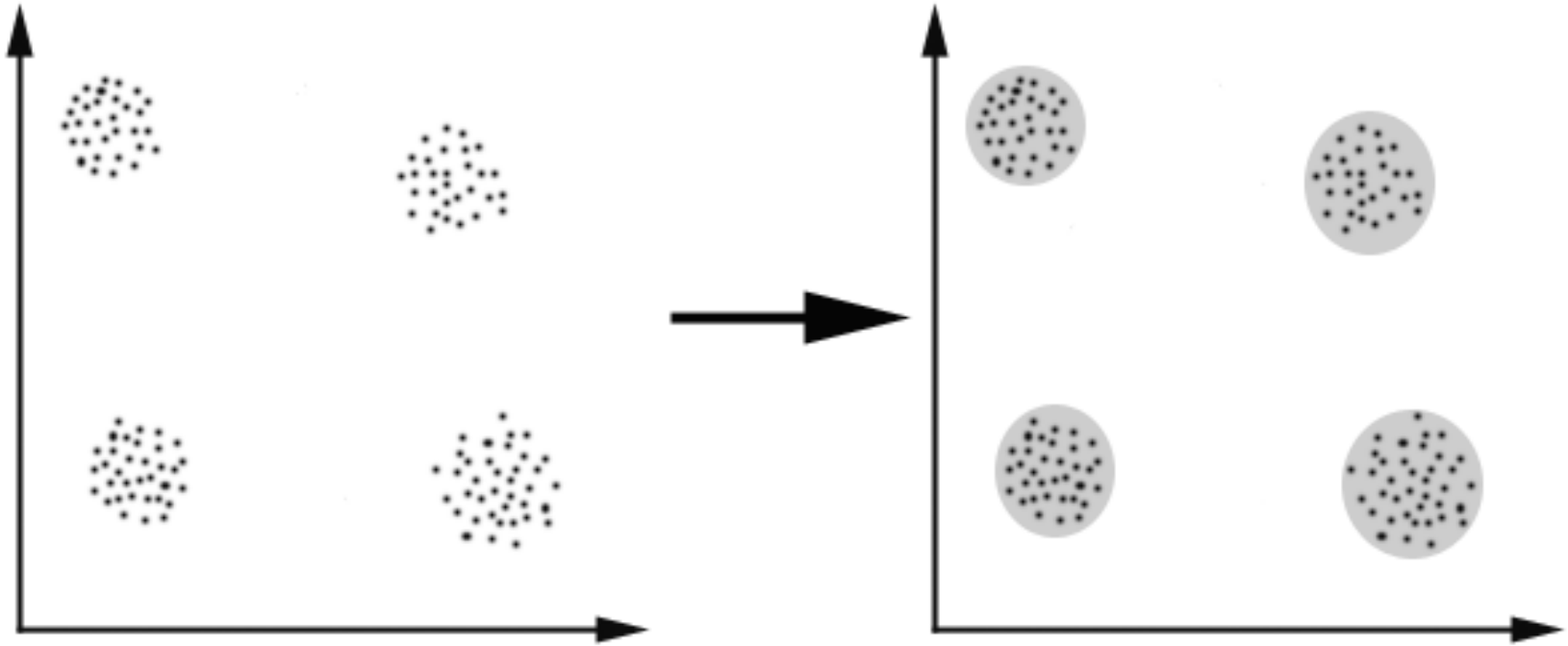
•Informally, finding natural groupings among objects.



Question: When and why would we want to do this?

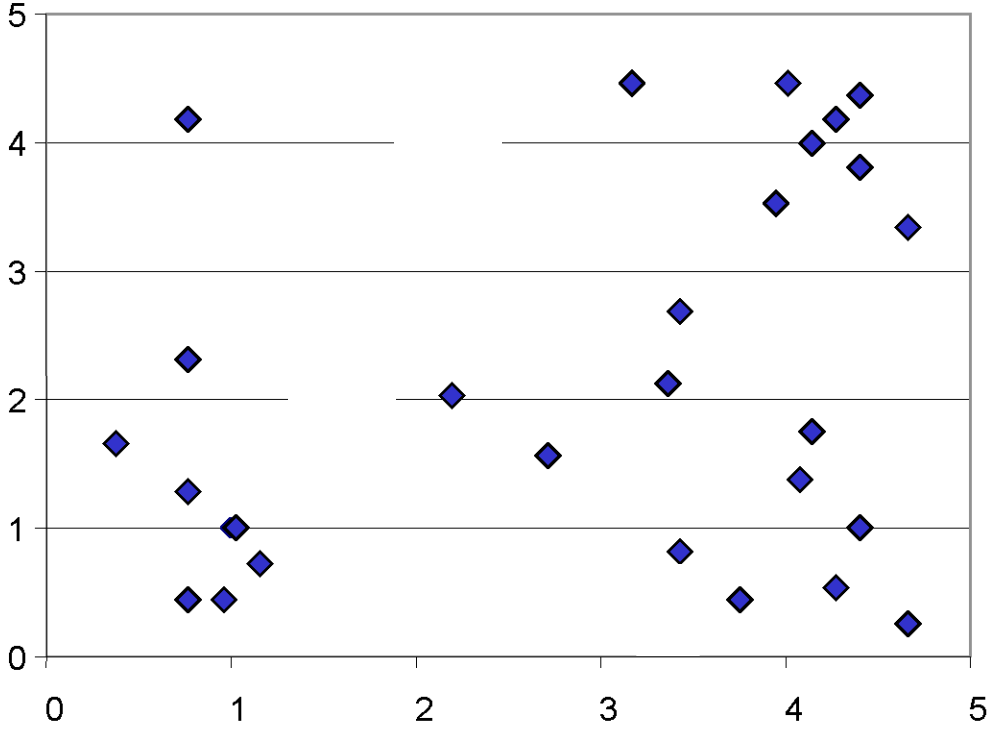
Useful for:

- Automatically organizing data.
- Understanding hidden structure in data.



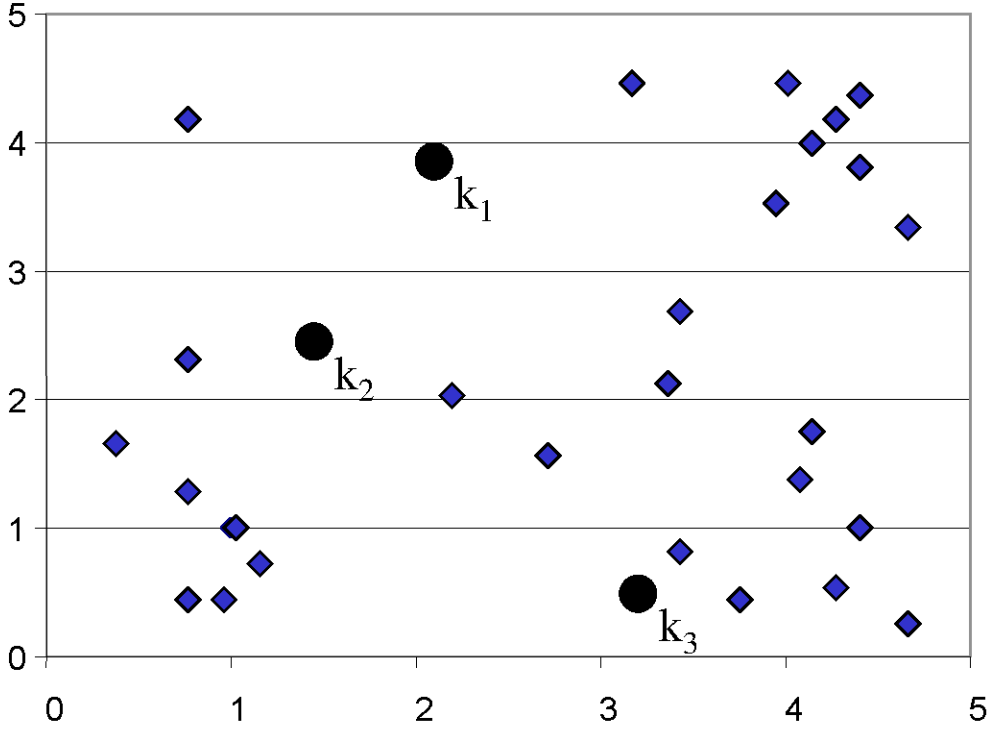
K-means Clustering: Initialization

Decide K, and initialize K centers (randomly)



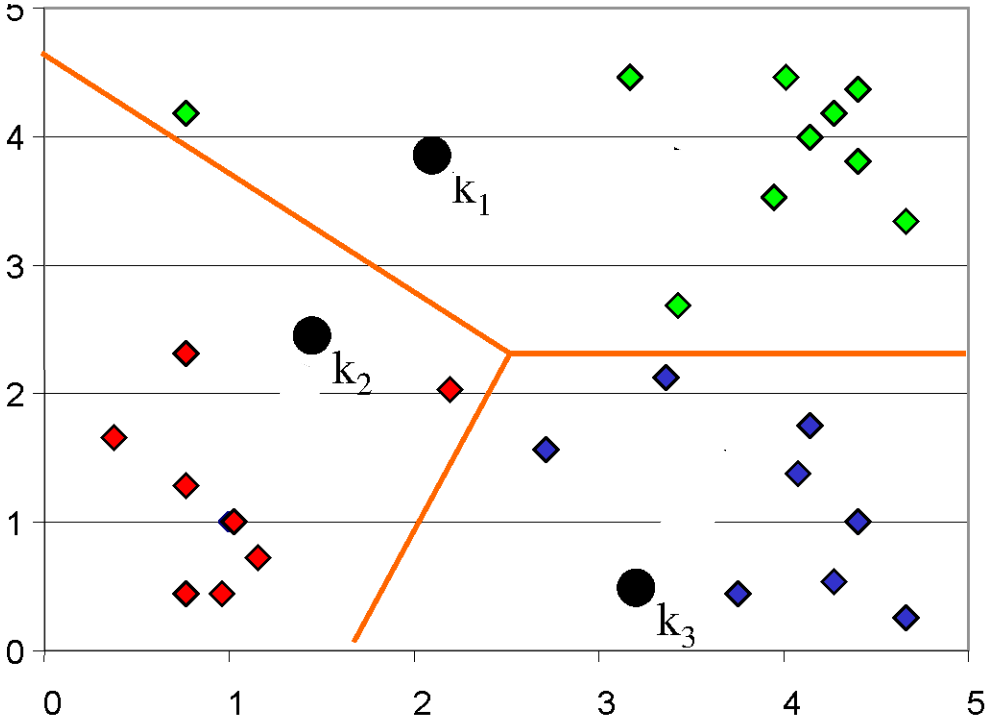
K-means Clustering: Initialization

Decide K, and initialize K centers (randomly)



K-means Clustering: Iteration 1

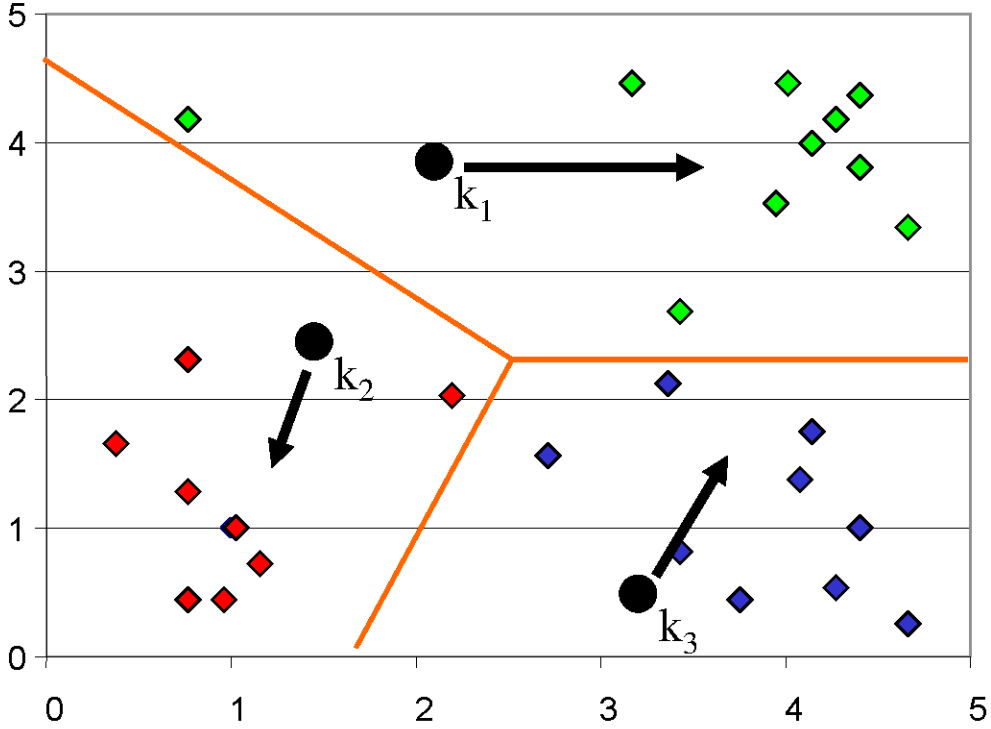
Assign all objects to the nearest center.



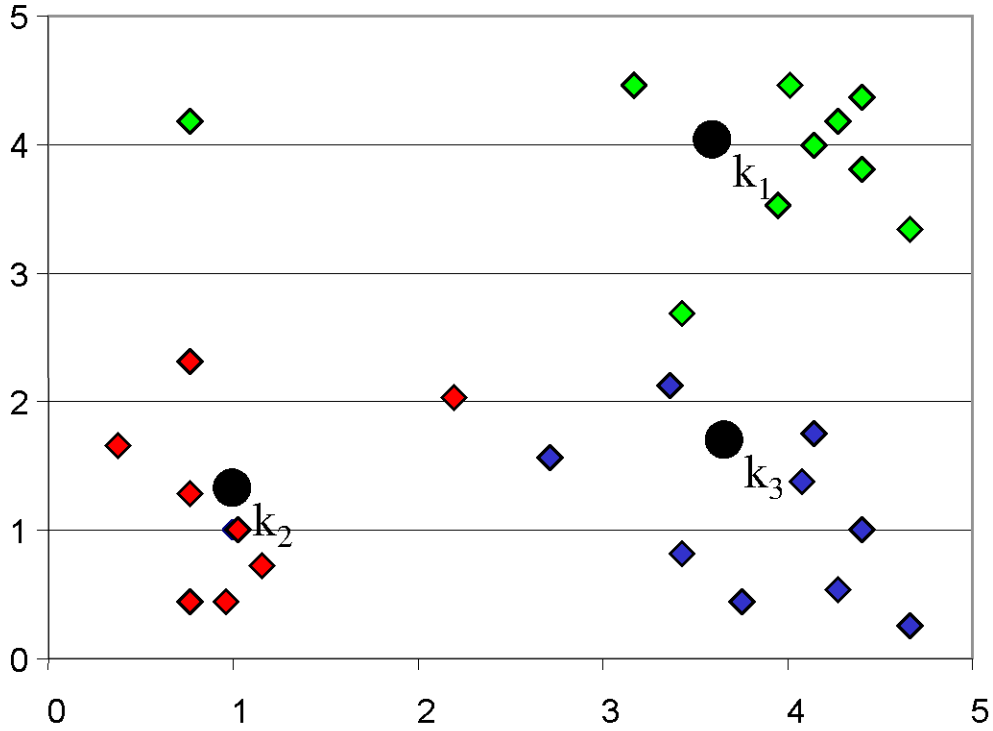
K-means Clustering: Iteration 1

Assign all objects to the nearest center.

Move a center to the mean of its members.

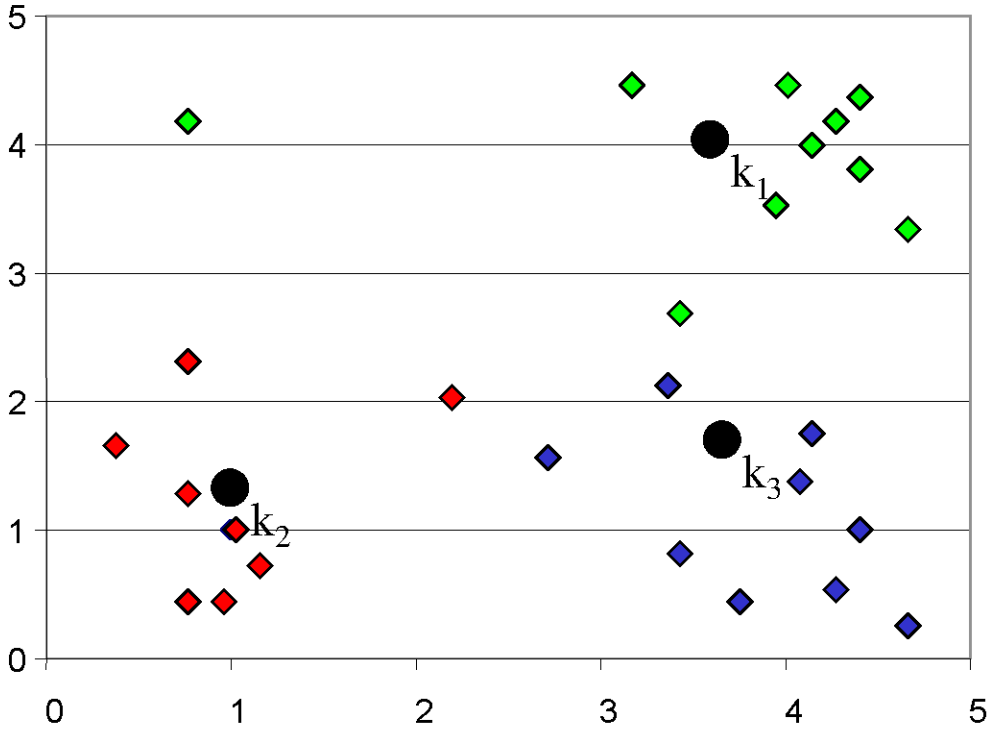


K-means Clustering: Iteration 2



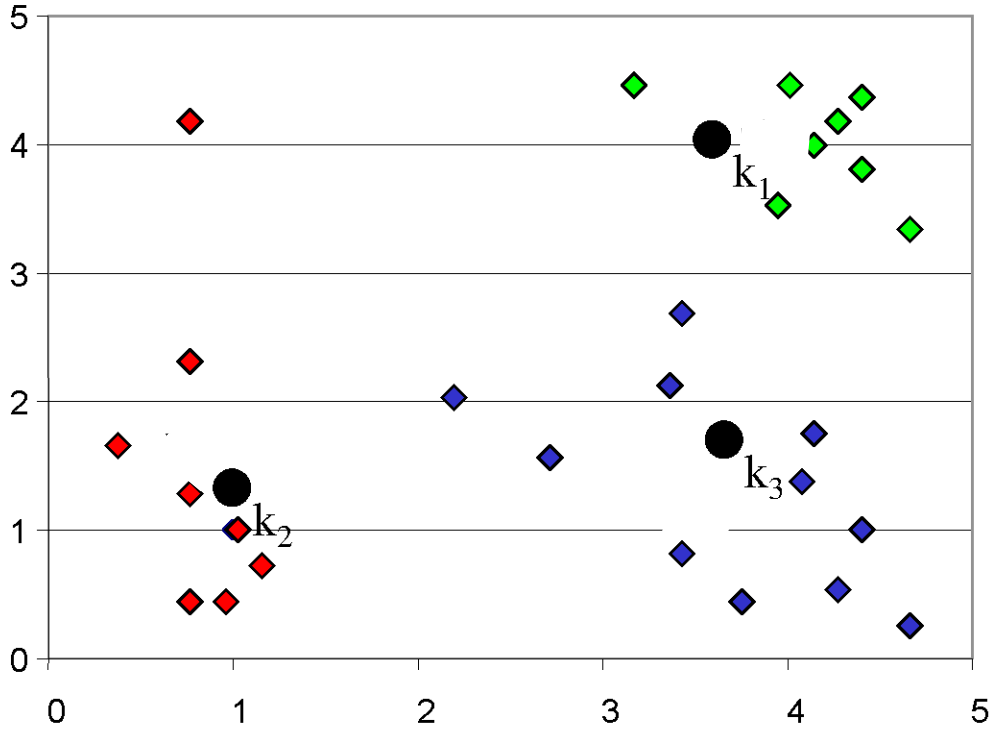
K-means Clustering: Iteration 2

After moving centers, re-assign the objects…



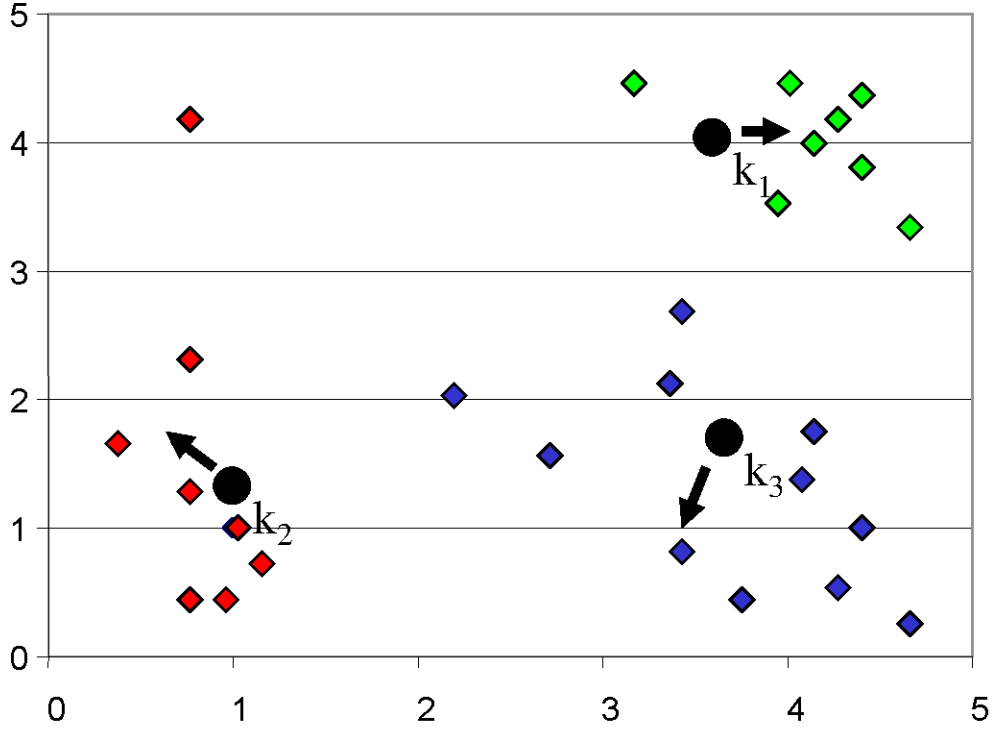
K-means Clustering: Iteration 2

After moving centers, re-assign the objects to nearest centers.



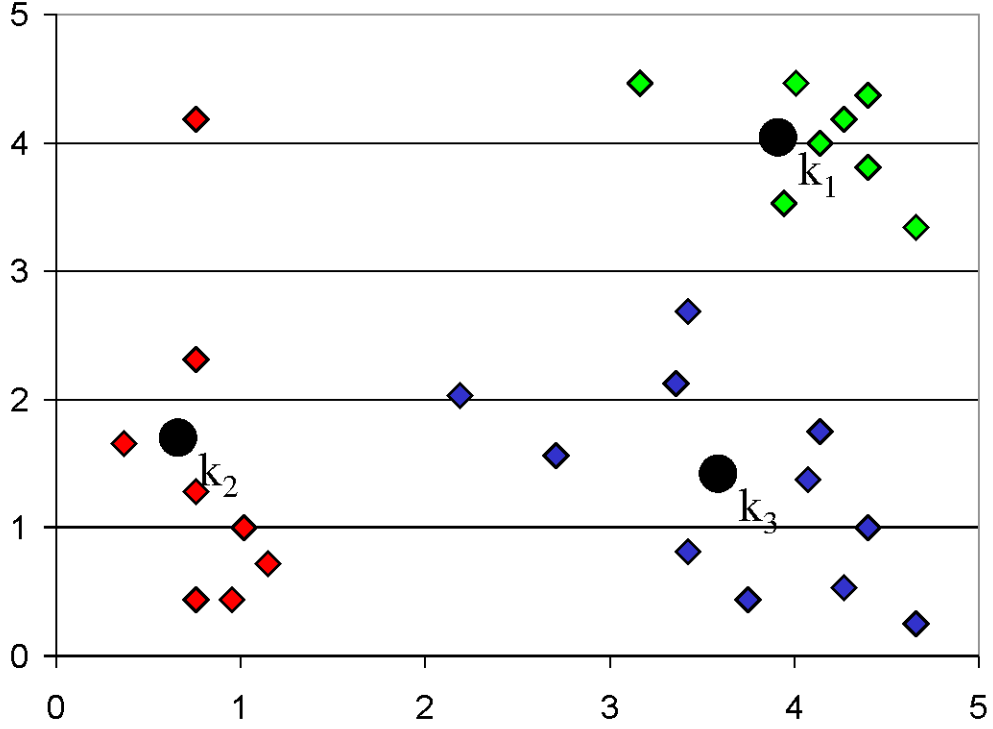
K-means Clustering: Iteration 2

After moving centers, re-assign the objects to nearest centers.
Move a center to the mean of its new members.



K-means Clustering: Finished!

Re-assign and move centers, until …
no objects changed membership.



$\{x^{(1)}, \ldots, x^{(m)}\}$ $\quad x^{(i)} \in \mathbb{R}^n$

The k-means clustering algorithm is as follows:

1. Initialize **cluster centroids** $\mu_1, \mu_2, \ldots, \mu_k \in \mathbb{R}^n$ randomly.

2. Repeat until convergence: {

For every i, set

$$c^{(i)} := \arg\min_j ||x^{(i)} - \mu_j||^2.$$

Assignment step: Assign each data point to the closest cluster

For each j, set

$$\mu_j := \frac{\sum_{i=1}^m 1\{c^{(i)} = j\} x^{(i)}}{\sum_{i=1}^m 1\{c^{(i)} = j\}}.$$

Refitting step: Move each cluster center to the center of the data assigned to it

}

EXPECTATION MAXIMIZATION

$\{x^{(1)}, \ldots, x^{(m)}\}$ $\quad x^{(i)} \in \mathbb{R}^n$

The k-means clustering algorithm is as follows:

1. Initialize **cluster centroids** $\mu_1, \mu_2, \ldots, \mu_k \in \mathbb{R}^n$ randomly.

2. Repeat until convergence: {

   For every i, set

$$c^{(i)} := \arg\min_j ||x^{(i)} - \mu_j||^2.$$

   For each j, set

$$\mu_j := \frac{\sum_{i=1}^m 1\{c^{(i)} = j\} x^{(i)}}{\sum_{i=1}^m 1\{c^{(i)} = j\}}.$$

   }

E

Assignment step: Assign each data point to the closest cluster

M

Refitting step: Move each cluster center to the center of the data assigned to it

$\{x^{(1)}, \ldots, x^{(m)}\}$ $\quad x^{(i)} \in \mathbb{R}^n$

The k-means clustering algorithm is as follows:

1. Initialize **cluster centroids** $\mu_1, \mu_2, \ldots, \mu_k \in \mathbb{R}^n$ randomly.
2. Repeat until convergence: {

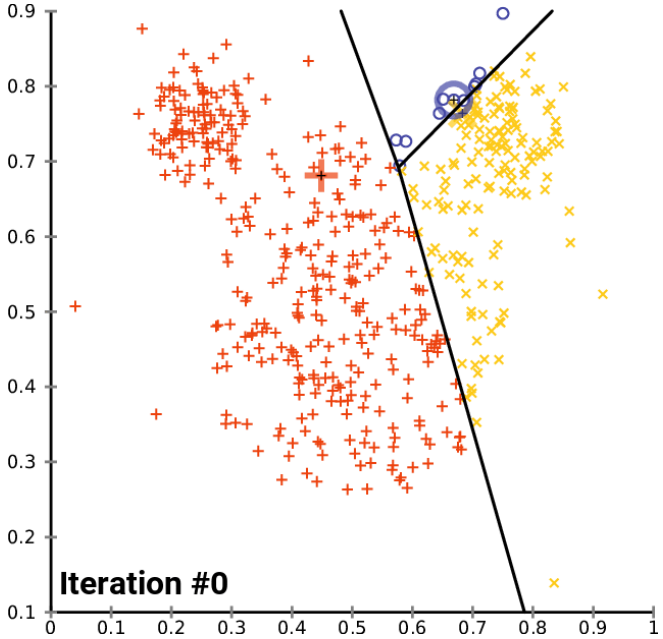
   For every i, set

   $$c^{(i)} := \arg\min_j ||x^{(i)} - \mu_j||^2.$$

   For each j, set

   $$\mu_j := \frac{\sum_{i=1}^m 1\{c^{(i)} = j\} x^{(i)}}{\sum_{i=1}^m 1\{c^{(i)} = j\}}.$$

   }



Algorithm *k-means*

1. Decide on a value for K, the number of clusters.

2. Initialize the K cluster centers (randomly, if necessary).

3. Decide the class memberships of the N objects by assigning them to the nearest cluster center.

4. Re-estimate the K cluster centers, by assuming the memberships found above are correct.

5. Repeat 3 and 4 until none of the N objects changed membership in the last iteration.

Algorithm *k-means*

1. Decide on a value for *K*, th

2. Initialize the *K* cluster cente
necessary).

3. Decide the class memberships of the *N* objects by assigning them to the nearest cluster center.

4. Re-estimate the *K* cluster centers, by assuming the memberships found above are correct.

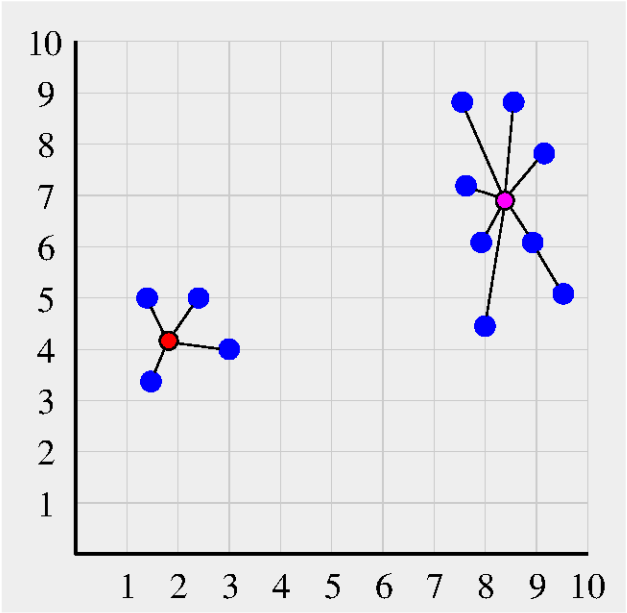
5. Repeat 3 and 4 until none of the *N* objects changed membership in the last iteration

Use one of the distance / similarity functions we discussed earlier

Average / median of class members

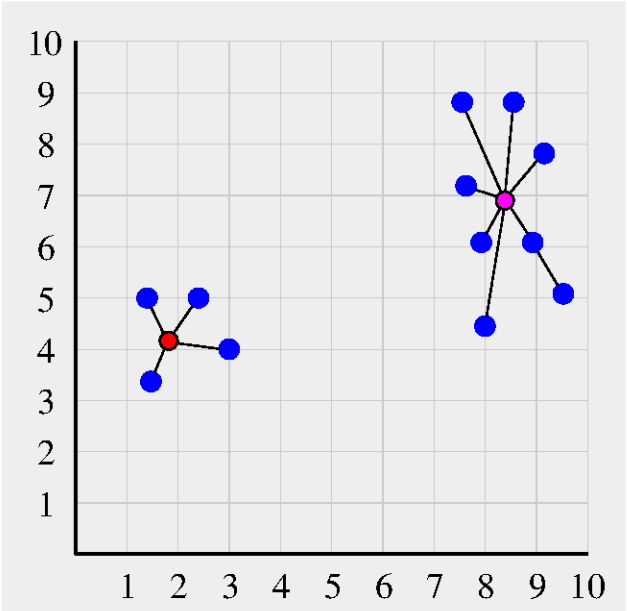
Why K-means Works

- What is a good partition?
- High intra-cluster similarity



Why K-means Works

- What is a good partition?
- High intra-cluster similarity
- K-means optimizes



$$se = \sum_{k=1}^{K} \sum_{i=1}^{n_k} \left\| x_{ki} - \mu_k \right\|^2$$

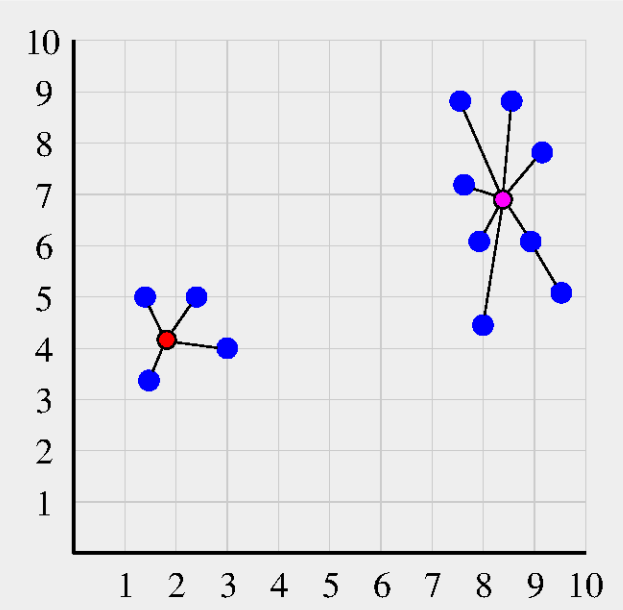
Why K-means Works

- What is a good partition?
- High intra-cluster similarity
- K-means optimizes
  - the average distance to members of the same cluster

$$\sum_{k=1}^{K} \frac{1}{n_k} \sum_{i=1}^{n_k} \sum_{j=1}^{n_k} \left\| x_{ki} - x_{kj} \right\|^2$$

  - which is twice the total distance to centers, also called squared error

$$se = \sum_{k=1}^{K} \sum_{i=1}^{n_k} \left\| x_{ki} - \mu_k \right\|^2$$



Repeat until convergence: {

For every i, set

$$c^{(i)} := \arg\min_j ||x^{(i)} - \mu_j||^2.$$

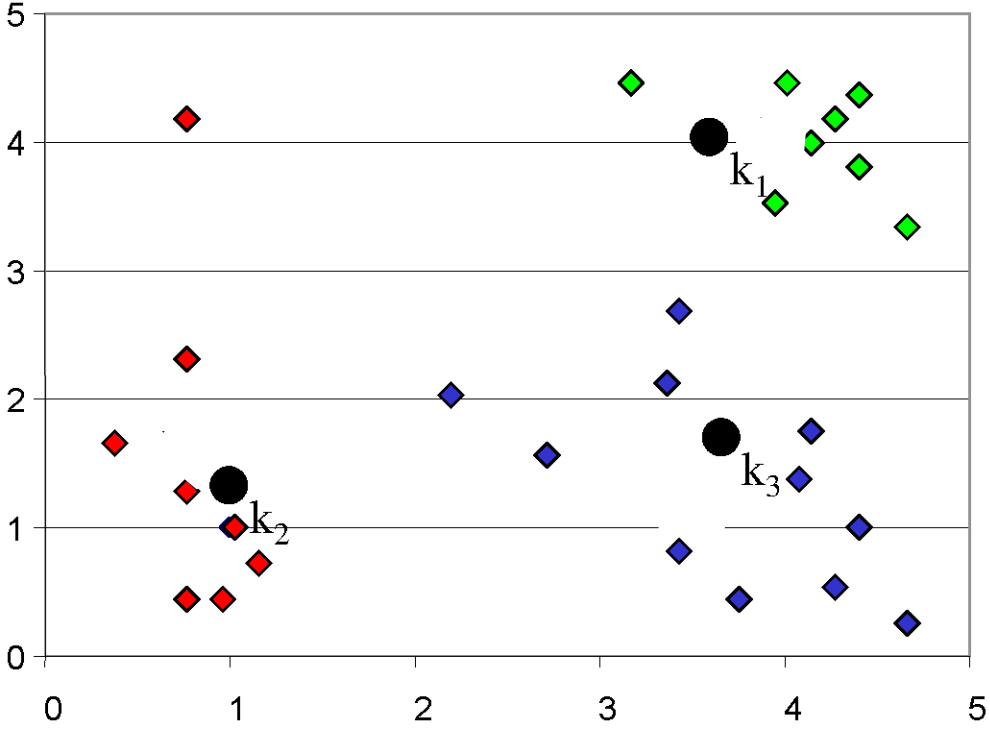
For each j, set

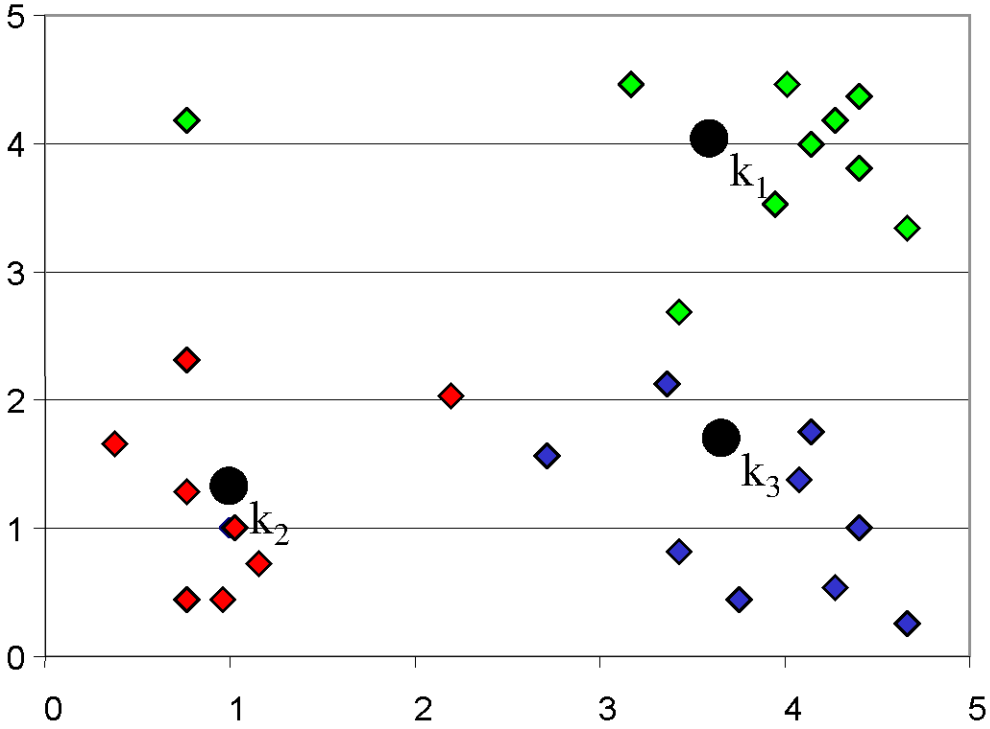
$$\mu_j := \frac{\sum_{i=1}^{m} 1\{c^{(i)} = j\} x^{(i)}}{\sum_{i=1}^{m} 1\{c^{(i)} = j\}}.$$

}

$$\sum_{k=1}^{K} \sum_{i=1}^{n_k} \left\| x_{ki} - \mu_k \right\|^2$$

- Whenever an assignment is changed, the sum squared distances J of data points from their assigned cluster centers is reduced.


5
4
3
2
1
0
0
1
2
3
4
5
k1
k2
k3


5
4
3
2
1
0
0
1
2
3
4
5
k1
k2
k3

Repeat until convergence: {

For every i, set

$$c^{(i)} := \arg\min_j ||x^{(i)} - \mu_j||^2.$$

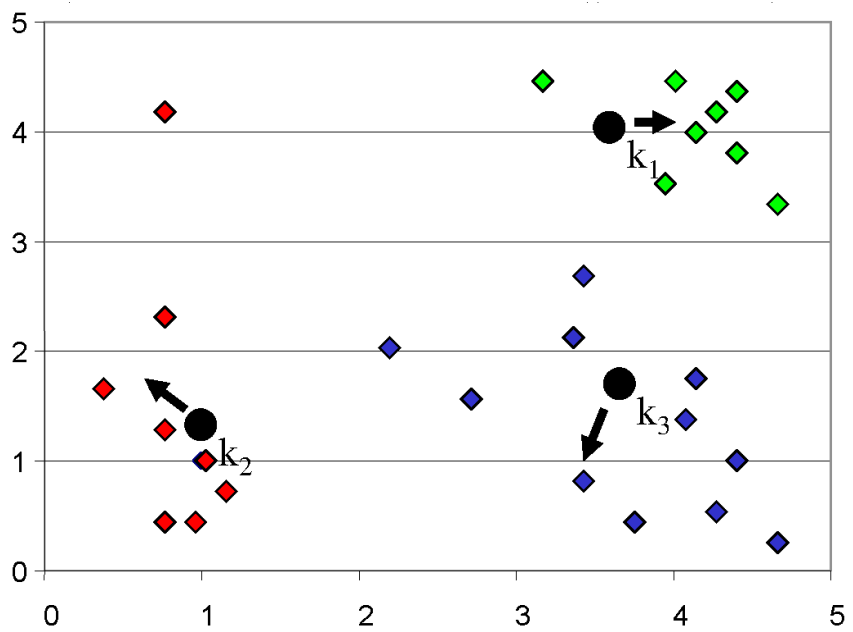
For each j, set

$$\mu_j := \frac{\sum_{i=1}^{m} 1\{c^{(i)} = j\} x^{(i)}}{\sum_{i=1}^{m} 1\{c^{(i)} = j\}}.$$

}

$$\sum_{k=1}^{K} \sum_{i=1}^{n_k} \left\| x_{ki} - \mu_k \right\|^2$$

- Whenever an assignment is changed, the sum squared distances J of data points from their assigned cluster centers is reduced.
- Whenever a cluster center is moved, J is reduced.



Repeat until convergence: {

For every i, set

$$c^{(i)} := \arg\min_j ||x^{(i)} - \mu_j||^2.$$

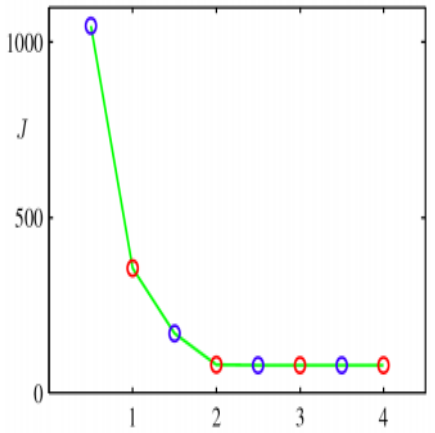
For each j, set

$$\mu_j := \frac{\sum_{i=1}^{m} 1\{c^{(i)} = j\} x^{(i)}}{\sum_{i=1}^{m} 1\{c^{(i)} = j\}}.$$

}

$$\sum_{k=1}^{K} \sum_{i=1}^{n_k} \left\| x_{ki} - \mu_k \right\|^2$$

- Whenever an assignment is changed, the sum squared distances J of data points from their assigned cluster centers is reduced.
- Whenever a cluster center is moved, J is reduced.
- Test for convergence: If the assignments do not change in the assignment step, we have converged (to at least a local minimum).



- K-means cost function after each E step (blue) and M step (red). The algorithm has converged after the third M step

Block/Alternate Coordinate Descent

$$f(x) = \tfrac{1}{2} x^T \begin{pmatrix} 1 & 0.5 \\ 0.5 & 1 \end{pmatrix} x - \begin{pmatrix} 1.5 & 1.5 \end{pmatrix} x, \quad x_0 = \begin{pmatrix} 0 & 0 \end{pmatrix}^T$$

1. Initialize **cluster centroids** $\mu_1, \mu_2, \ldots, \mu_k \in \mathbb{R}^n$ randomly.
2. Repeat until convergence: {

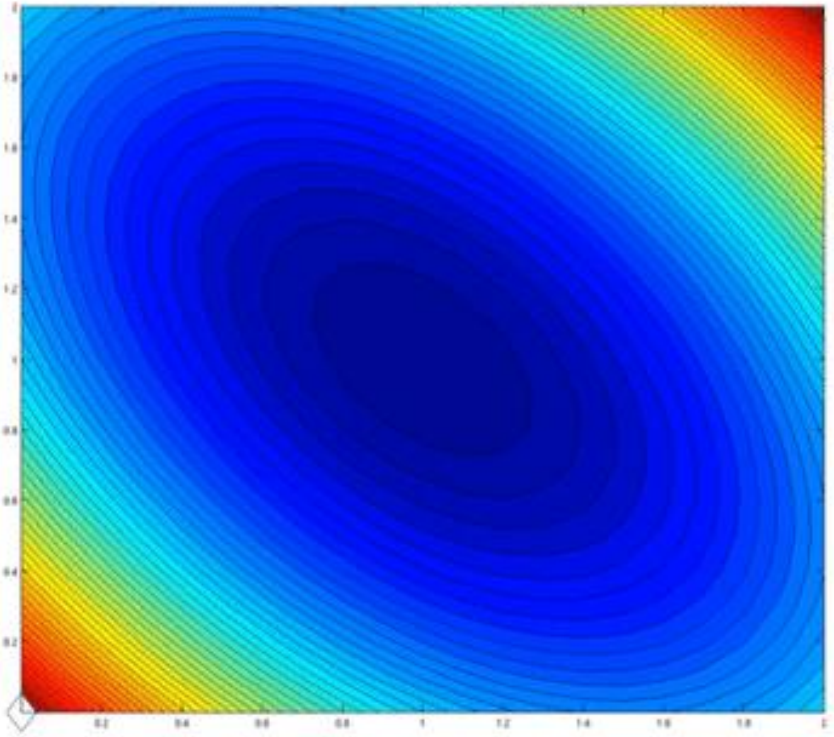
   For every i, set

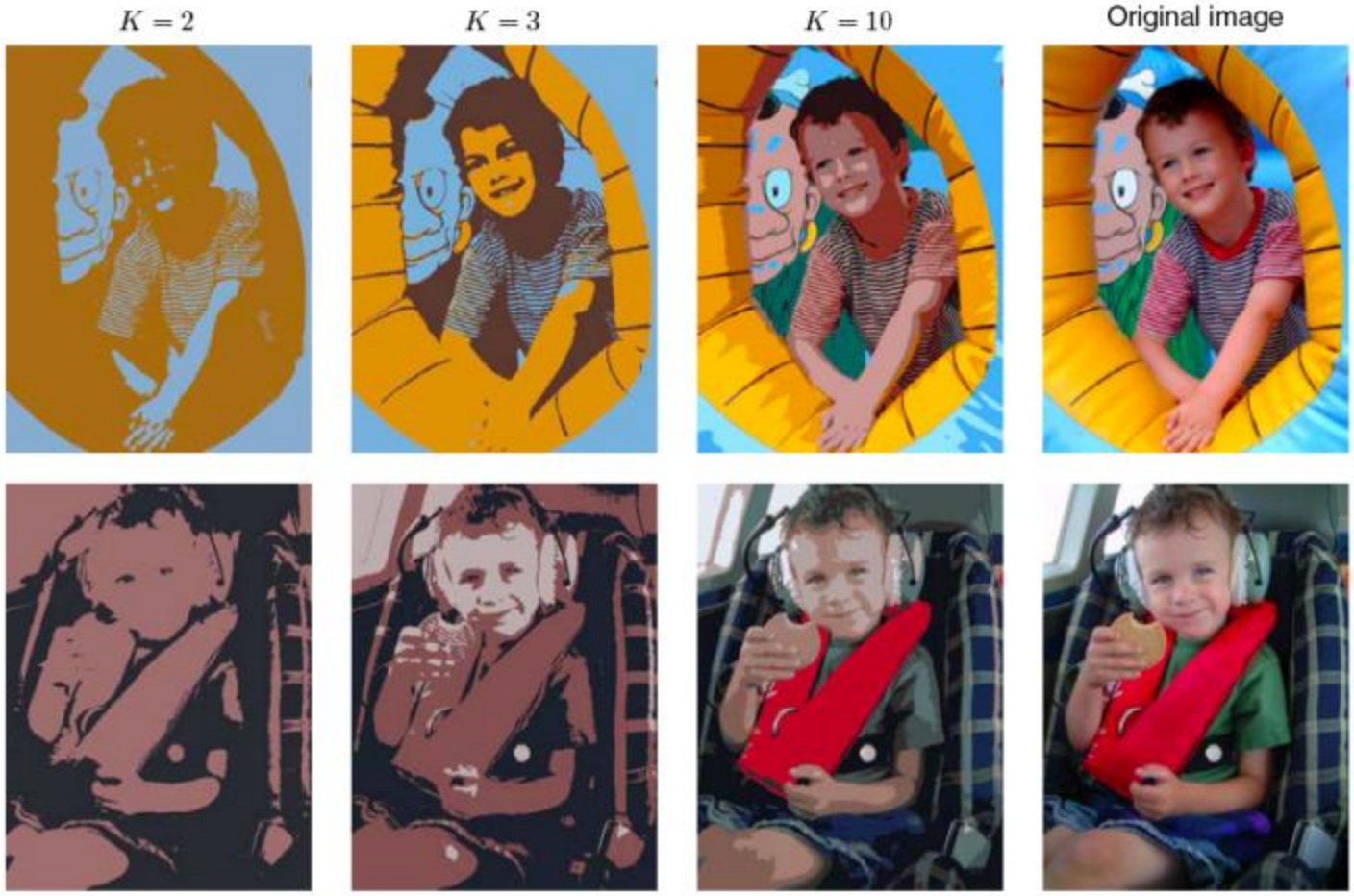
   $$c^{(i)} := \arg\min_j ||x^{(i)} - \mu_j||^2.$$

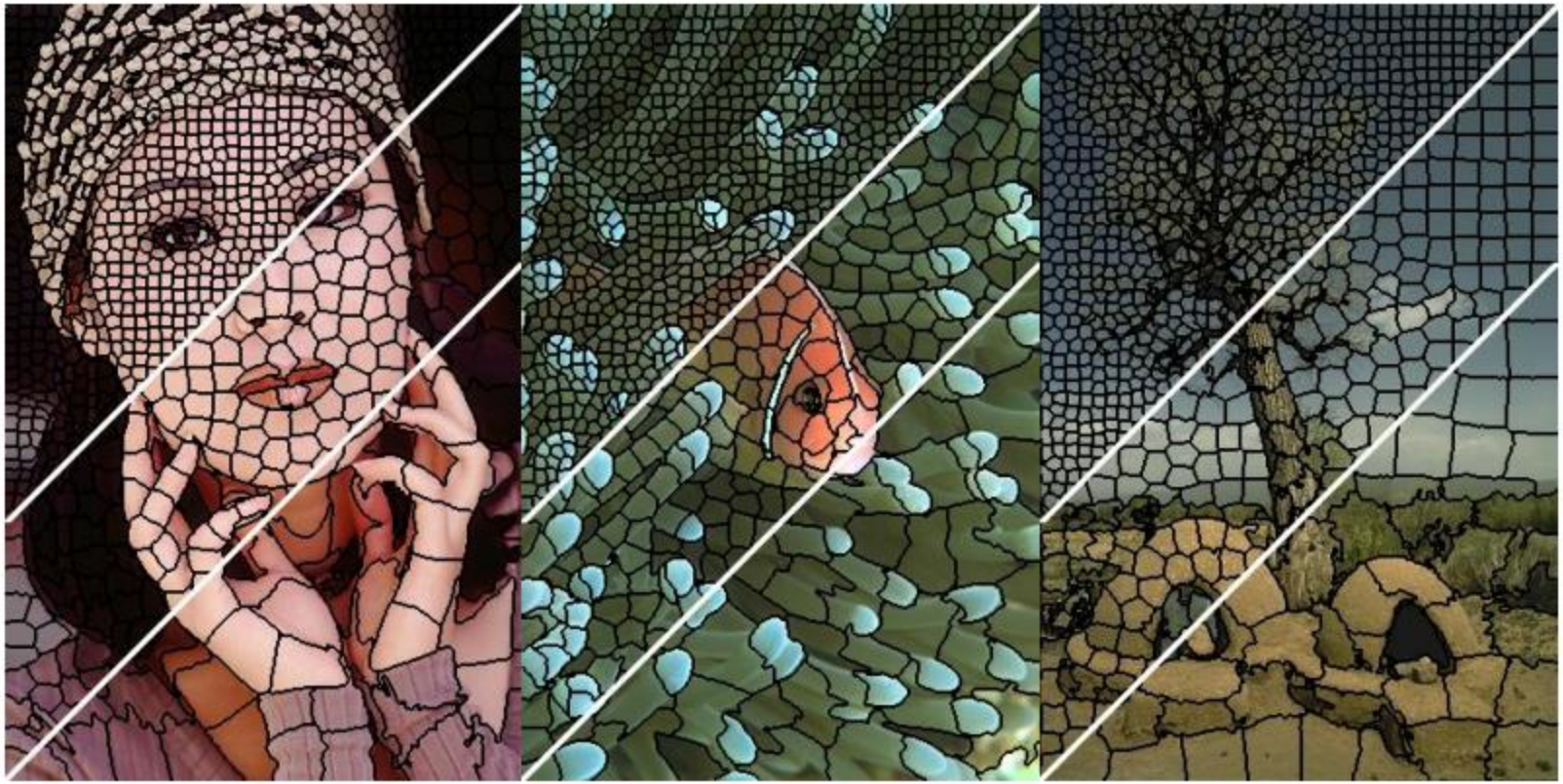
   For each j, set

   $$\mu_j := \frac{\sum_{i=1}^m 1\{c^{(i)} = j\} x^{(i)}}{\sum_{i=1}^m 1\{c^{(i)} = j\}}.$$

   }

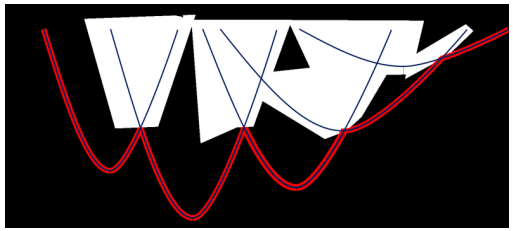



$K = 2$
$K = 3$
$K = 10$
Original image



- How would you modify k-means to get super pixels?

- The objective J is non-convex (so coordinate descent on J is not guaranteed to converge to the global minimum)



1. Initialize **cluster centroids** $\mu_1, \mu_2, \ldots, \mu_k \in \mathbb{R}^n$ randomly.
2. Repeat until convergence: {

   For every i, set

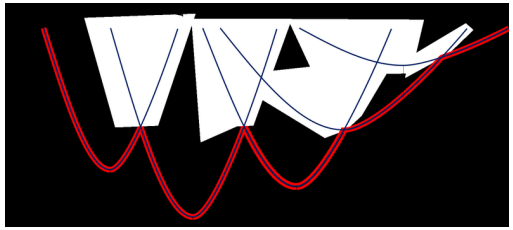
   $$c^{(i)} := \arg\min_j ||x^{(i)} - \mu_j||^2.$$

   For each j, set

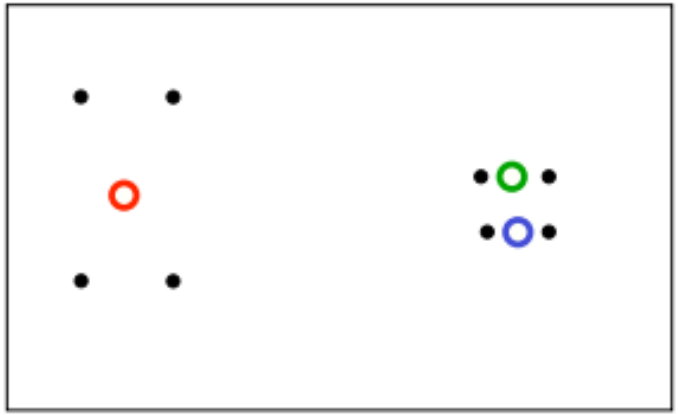
   $$\mu_j := \frac{\sum_{i=1}^m 1\{c^{(i)} = j\} x^{(i)}}{\sum_{i=1}^m 1\{c^{(i)} = j\}}.$$

   }

- The objective J is non-convex (so coordinate descent on J is not guaranteed to converge to the global minimum)
- There is nothing to prevent k-means getting stuck at local minima.



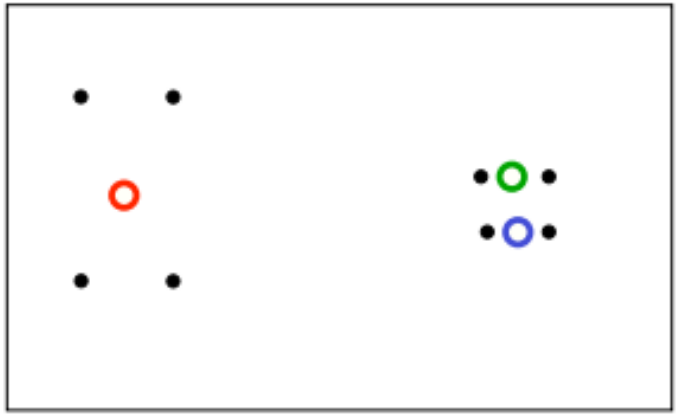
A bad local optimum



- The objective J is non-convex (so coordinate descent on J is not guaranteed to converge to the global minimum)
- There is nothing to prevent k-means getting stuck at local minima.
- We could try many random starting points

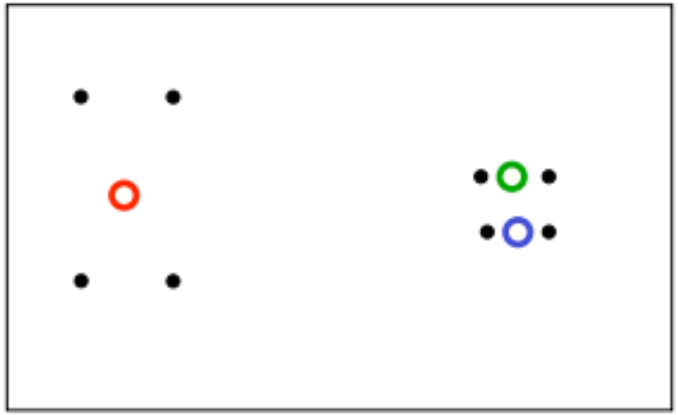
$$\sum_{k=1}^{K} \sum_{i=1}^{n_k} \left\| x_{ki} - \mu_k \right\|^2$$

A bad local optimum



- The objective J is non-convex (so coordinate descent on J is not guaranteed to converge to the global minimum)
- There is nothing to prevent k-means getting stuck at local minima.
- We could try many random starting points
- We could try non-local split-and-merge moves:
  - Simultaneously merge two nearby clusters
  - and split a big cluster into two

A bad local optimum



$$\sum_{k=1}^{K}\sum_{i=1}^{n_k}\left\|x_{ki}-\mu_k\right\|^2$$

K-means++: Improving K-means initialization

- Common way to improve k-means - smart initialization!
- General idea - try to get good coverage of the data.
- k-means++ algorithm:
  1. Pick the first center randomly
  2. For all points $\mathbf{x}^{(n)}$ set $d^{(n)}$ to be the distance to closest center.
  3. Pick the new center to be at $\mathbf{x}^{(n)}$ with probability proportional to $d^{(n)2}$
  4. Repeat steps 2+3 until you have k centers

K-means: Additional issues

- ‘Hard’ assignments
- Euclidean ➔ Favours ‘Spherical’ clusters
  - Cluster-distribution-adaptive ?

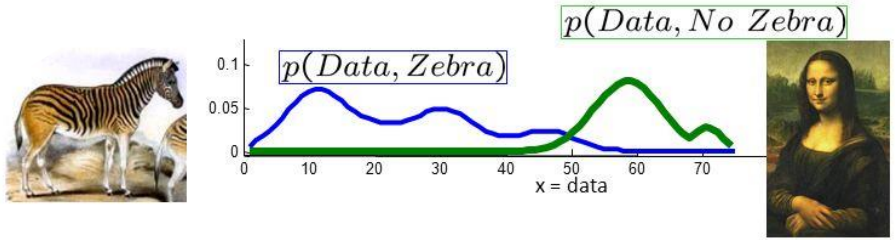
Clustering – A Generative approach

- Imagine that the data was produced by a **probabilistic** generative model
  - Generate cluster centers
  - Generate points ‘conditioned on cluster centers’
- Strategy: Adjust the model parameters to maximize the observed ‘data’ probability

Mixture Models

- We model the joint distribution as,

$$p(\mathbf{x}, z) = p(\mathbf{x}|z)p(z)$$



Mixture Models

- We model the joint distribution as,

$$p(\mathbf{x}, z) = p(\mathbf{x}|z)p(z)$$

- But in unsupervised clustering we do not have the class labels z.
- What can we do instead?

Mixture Models

- We model the joint distribution as,

$$p(\mathbf{x}, z) = p(\mathbf{x}|z)p(z)$$

- But in unsupervised clustering we do not have the class labels z.
- What can we do instead?

$$p(\mathbf{x}) = \sum_z p(\mathbf{x}, z) = \sum_z p(\mathbf{x}|z)p(z)$$

- This is a mixture model

Mixture Models

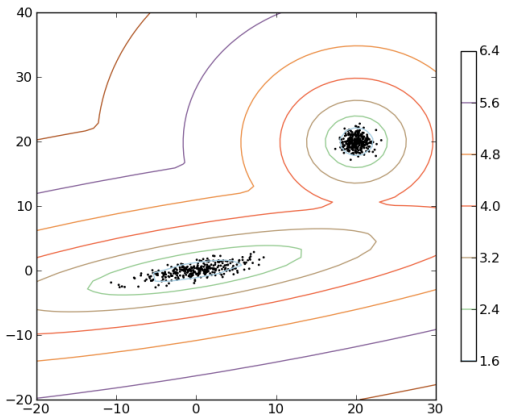
Most common mixture model: Gaussian mixture model (GMM)

- A GMM represents a **distribution** as

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\mu_k, \Sigma_k)$$

with π_k the mixing coefficients, where:

$$\sum_{k=1}^{K} \pi_k = 1 \quad \text{and} \quad \pi_k \geq 0 \quad \forall k$$



http://scikit-learn.sourceforge.net/0.5/auto_examples/gmm/plot_gmm_pdf.html

Parameter Estimation in GMMs

Most common mixture model: Gaussian mixture model (GMM)

- A GMM represents a **distribution** as

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\mu_k, \Sigma_k)$$

with π_k the mixing coefficients, where:

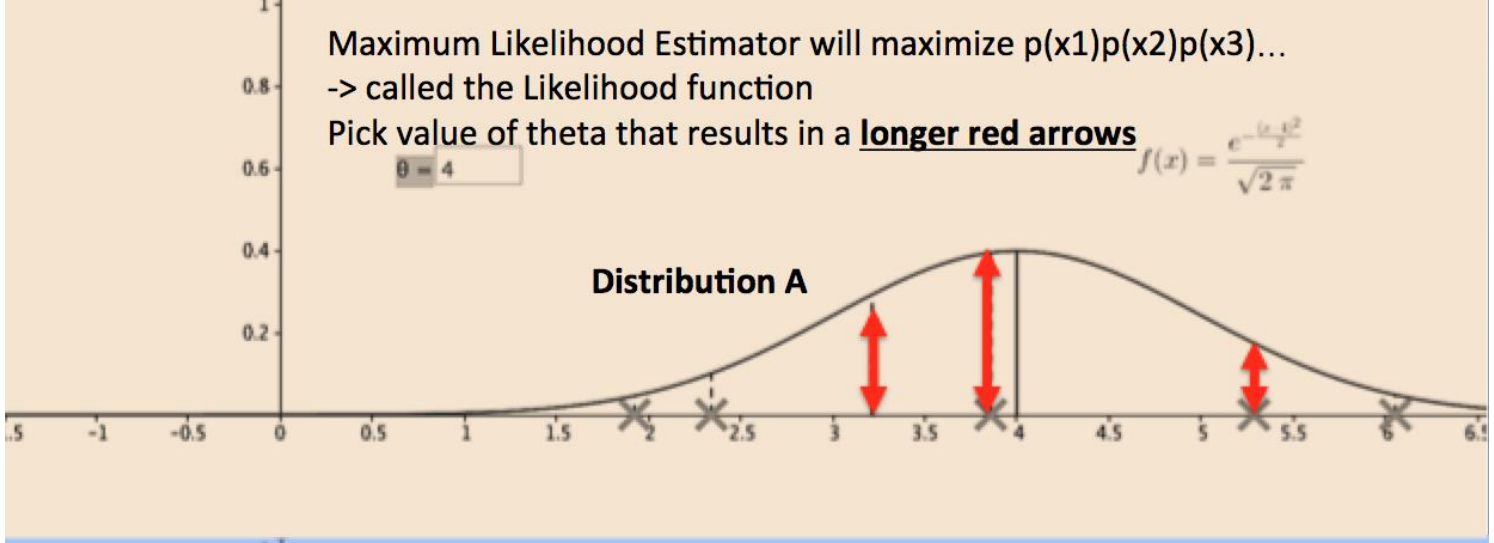
$$\sum_{k=1}^{K} \pi_k = 1 \quad \text{and} \quad \pi_k \geq 0 \quad \forall k$$

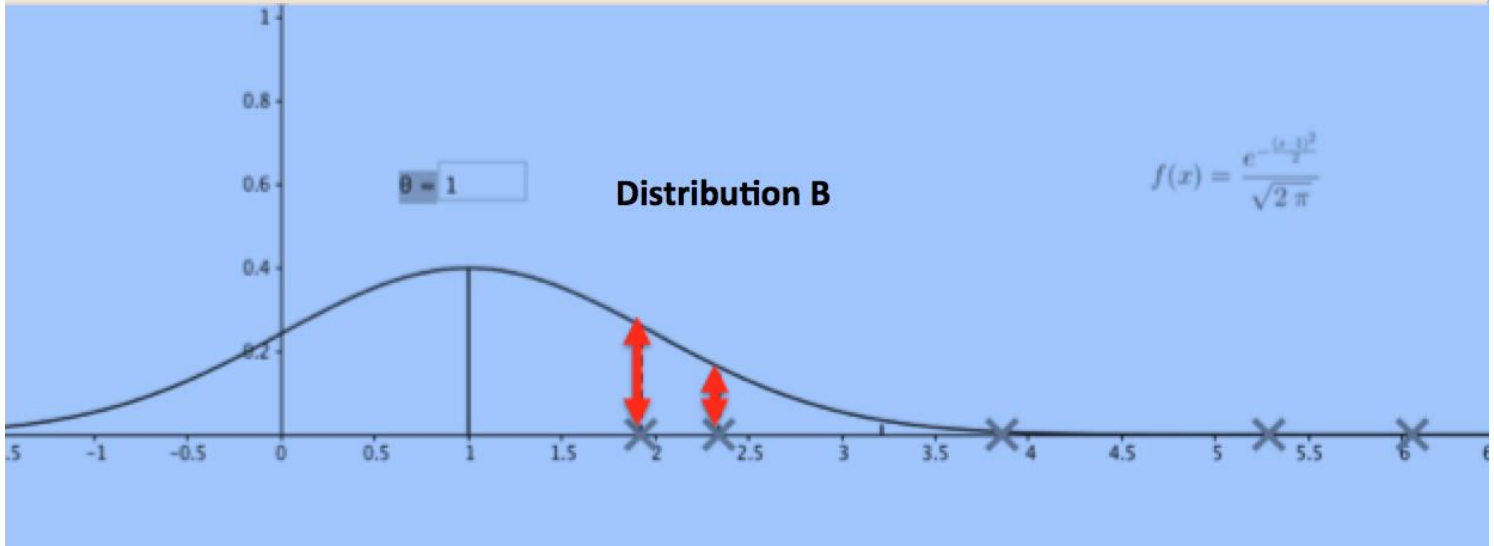
- Maximum likelihood maximizes

$$\ln p(\mathbf{X}|\pi, \mu, \Sigma) = \sum_{n=1}^{N} \ln \left(\sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}^{(n)}|\mu_k, \Sigma_k) \right)$$

w.r.t $\Theta = \{\pi_k, \mu_k, \Sigma_k\}$

Maximum Likelihood Estimator will maximize p(x1)p(x2)p(x3)...
-> called the Likelihood function
Pick value of theta that results in a **longer red arrows**





- Maximum likelihood maximizes

$$\ln p(\mathbf{X}|\pi, \mu, \Sigma) = \sum_{n=1}^{N} \ln \left(\sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}^{(n)}|\mu_k, \Sigma_k) \right)$$

w.r.t $\Theta = \{\pi_k, \mu_k, \Sigma_k\}$

NOTE: We are estimating per-sample memberships & GMM parameters

$$z \sim \text{Categorical}(\boldsymbol{\pi}) \quad (\text{where } \pi_k \geq 0, \quad \textstyle\sum_k \pi_k = 1)$$

- Joint distribution: $p(\mathbf{x}, \mathbf{z}) = p(\mathbf{z})p(\mathbf{x}|\mathbf{z})$

- Maximum likelihood maximizes

$$\ln p(\mathbf{X}|\pi,\mu,\Sigma) = \sum_{n=1}^{N} \ln \left(\sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}^{(n)}|\mu_k,\Sigma_k) \right)$$

w.r.t $\Theta = \{\pi_k, \mu_k, \Sigma_k\}$

NOTE: We are estimating per-sample memberships & GMM parameters

$$z \sim \text{Categorical}(\boldsymbol{\pi}) \quad (\text{where } \pi_k \geq 0, \quad \textstyle\sum_k \pi_k = 1)$$

- Joint distribution: $p(\mathbf{x},\mathbf{z}) = p(\mathbf{z})p(\mathbf{x}|\mathbf{z})$
- Log-likelihood:

$$\begin{aligned}\ell(\boldsymbol{\pi},\mu,\Sigma) = \ln p(\mathbf{X}|\pi,\mu,\Sigma) &= \sum_{n=1}^{N} \ln p(\mathbf{x}^{(n)}|\pi,\mu,\Sigma) \\ &= \sum_{n=1}^{N} \ln \sum_{z^{(n)}=1}^{K} p(\mathbf{x}^{(n)}|\, z^{(n)};\mu,\Sigma) p(z^{(n)}|\,\boldsymbol{\pi})\end{aligned}$$

- Note: We have a hidden variable $z^{(n)}$ for every observation

- Maximum likelihood maximizes

$$\ln p(\mathbf{X}|\pi,\mu,\Sigma) = \sum_{n=1}^{N} \ln \left(\sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}^{(n)}|\mu_k,\Sigma_k) \right)$$

w.r.t $\Theta = \{\pi_k, \mu_k, \Sigma_k\}$

NOTE: We are estimating per-sample memberships & GMM parameters

$$z \sim \text{Categorical}(\boldsymbol{\pi}) \quad (\text{where } \pi_k \geq 0, \quad \textstyle\sum_k \pi_k = 1)$$

- Joint distribution: $p(\mathbf{x},\mathbf{z}) = p(\mathbf{z})p(\mathbf{x}|\mathbf{z})$
- Log-likelihood:

$$\ell(\boldsymbol{\pi},\mu,\Sigma) = \ln p(\mathbf{X}|\pi,\mu,\Sigma) = \sum_{n=1}^{N} \ln p(\mathbf{x}^{(n)}|\pi,\mu,\Sigma)$$

$$= \sum_{n=1}^{N} \ln \sum_{z^{(n)}=1}^{K} p(\mathbf{x}^{(n)}|\, z^{(n)};\mu,\Sigma) p(z^{(n)}|\,\boldsymbol{\pi})$$

- Note: We have a hidden variable $z^{(n)}$ for every observation
- General problem: sum inside the log
- How can we optimize this?

- Log-likelihood:

$$\ell(\boldsymbol{\pi}, \mu, \Sigma) = \ln p(\mathbf{X}|\pi, \mu, \Sigma) = \sum_{n=1}^{N} \ln p(\mathbf{x}^{(n)}|\pi, \mu, \Sigma)$$

$$= \sum_{n=1}^{N} \ln \sum_{z^{(n)}=1}^{K} p(\mathbf{x}^{(n)}|\, z^{(n)}; \mu, \Sigma) p(z^{(n)}|\, \boldsymbol{\pi})$$

- Log-likelihood:

$$\ell(\boldsymbol{\pi}, \mu, \Sigma) = \ln p(\mathbf{X}|\pi, \mu, \Sigma) = \sum_{n=1}^{N} \ln p(\mathbf{x}^{(n)}|\pi, \mu, \Sigma)$$
$$= \sum_{n=1}^{N} \ln \sum_{z^{(n)}=1}^{K} p(\mathbf{x}^{(n)}|\, z^{(n)}; \mu, \Sigma) p(z^{(n)}|\, \boldsymbol{\pi})$$

- **If we knew** $z^{(n)}$ for every $x^{(n)}$, the maximum likelihood problem is easy:

$$\ell(\boldsymbol{\pi}, \mu, \Sigma) = \sum_{n=1}^{N} \ln p(x^{(n)}, z^{(n)}|\pi, \mu, \Sigma) = \sum_{n=1}^{N} \ln p(\mathbf{x}^{(n)}|\, z^{(n)}; \mu, \Sigma) + \ln p(z^{(n)}|\, \boldsymbol{\pi})$$

- Log-likelihood:

$$\ell(\boldsymbol{\pi}, \mu, \Sigma) = \ln p(\mathbf{X}|\pi, \mu, \Sigma) = \sum_{n=1}^{N} \ln p(\mathbf{x}^{(n)}|\pi, \mu, \Sigma)$$
$$= \sum_{n=1}^{N} \ln \sum_{z^{(n)}=1}^{K} p(\mathbf{x}^{(n)}|\, z^{(n)}; \mu, \Sigma) p(z^{(n)}|\, \boldsymbol{\pi})$$

- **If we knew** $z^{(n)}$ for every $x^{(n)}$, the maximum likelihood problem is easy:

$$\ell(\boldsymbol{\pi}, \mu, \Sigma) = \sum_{n=1}^{N} \ln p(x^{(n)}, z^{(n)}|\pi, \mu, \Sigma) = \sum_{n=1}^{N} \ln p(\mathbf{x}^{(n)}|\, z^{(n)}; \mu, \Sigma) + \ln p(z^{(n)}|\, \boldsymbol{\pi})$$

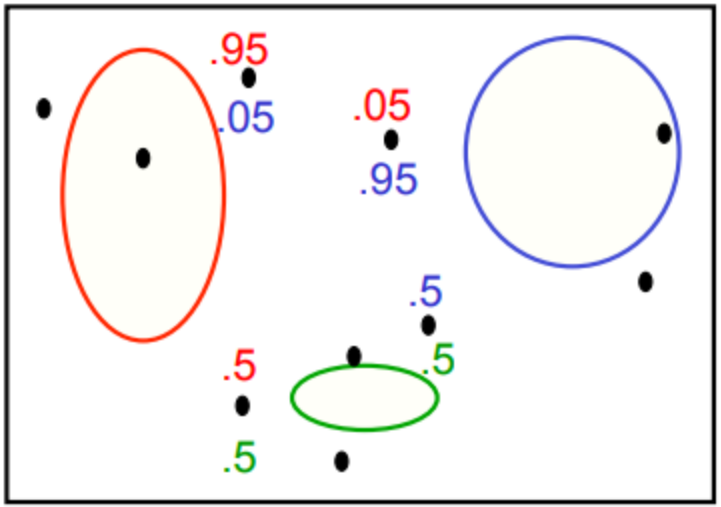
$$\mu_k = \frac{\sum_{n=1}^{N} 1_{[z^{(n)}=k]}\, \mathbf{x}^{(n)}}{\sum_{n=1}^{N} 1_{[z^{(n)}=k]}}$$

$$\Sigma_k = \frac{\sum_{n=1}^{N} 1_{[z^{(n)}=k]}\, (\mathbf{x}^{(n)} - \mu_k)(\mathbf{x}^{(n)} - \mu_k)^T}{\sum_{n=1}^{N} 1_{[z^{(n)}=k]}}$$

$$\pi_k = \frac{1}{N} \sum_{n=1}^{N} 1_{[z^{(n)}=k]}$$

- Optimization uses the Expectation Maximization algorithm, which alternates between two steps:
  1. E-step: Compute the posterior probability over z given our current model - i.e. how much do we think each Gaussian generates each datapoint.

$$\ell(\pi, \mu, \Sigma) = \ln p(\mathbf{X}|\pi, \mu, \Sigma) = \sum_{n=1}^{N} \ln p(\mathbf{x}^{(n)}|\pi, \mu, \Sigma)$$
$$= \sum_{n=1}^{N} \ln \sum_{z^{(n)}=1}^{K} p(\mathbf{x}^{(n)}|\, z^{(n)}; \mu, \Sigma) p(z^{(n)}|\, \boldsymbol{\pi})$$

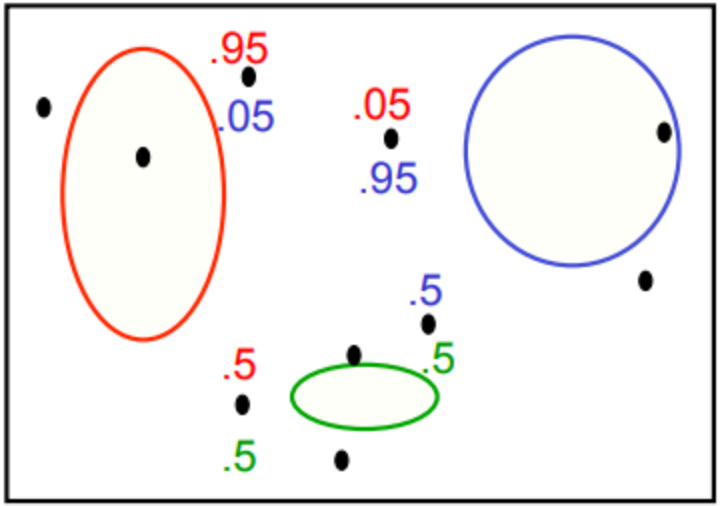


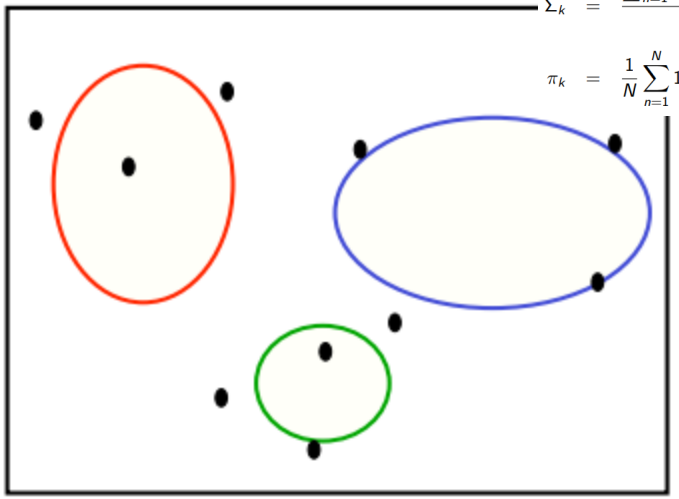
- Optimization uses the Expectation Maximization algorithm, which alternates between two steps:
  1. E-step: Compute the posterior probability over z given our current model - i.e. how much do we think each Gaussian generates each datapoint.
  2. M-step: Assuming that the data really was generated this way, change the parameters of each Gaussian to maximize the probability that it would generate the data it is currently responsible for.

$$\mu_k = \frac{\sum_{n=1}^{N} 1_{[z^{(n)}=k]} \mathbf{x}^{(n)}}{\sum_{n=1}^{N} 1_{[z^{(n)}=k]}}$$

$$\Sigma_k = \frac{\sum_{n=1}^{N} 1_{[z^{(n)}=k]} (\mathbf{x}^{(n)} - \mu_k)(\mathbf{x}^{(n)} - \mu_k)^T}{\sum_{n=1}^{N} 1_{[z^{(n)}=k]}}$$

$$\pi_k = \frac{1}{N} \sum_{n=1}^{N} 1_{[z^{(n)}=k]}$$





GMM - Algorithm

- Initialize the means μ_k, covariances Σ_k and mixing coefficients π_k
- Iterate until convergence:
  - E-step: Evaluate the responsibilities given current parameters

$$\gamma_k^{(n)} = p(z^{(n)}|\mathbf{x}) = \frac{\pi_k \mathcal{N}(\mathbf{x}^{(n)}|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}^{(n)}|\mu_j, \Sigma_j)}$$

  - M-step: Re-estimate the parameters given current responsibilities

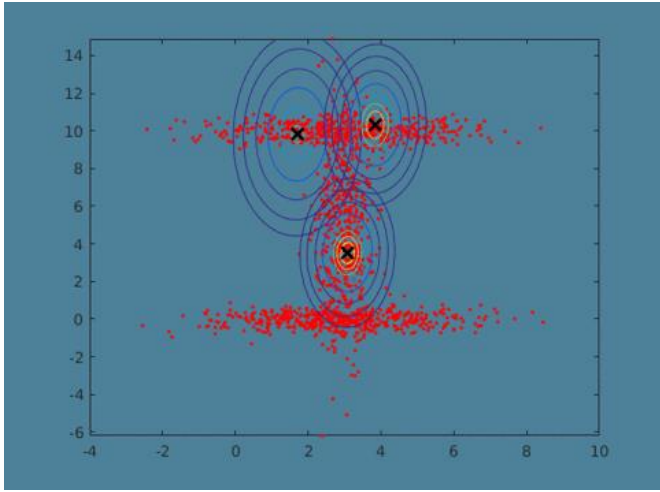
$$\mu_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma_k^{(n)} \mathbf{x}^{(n)}$$

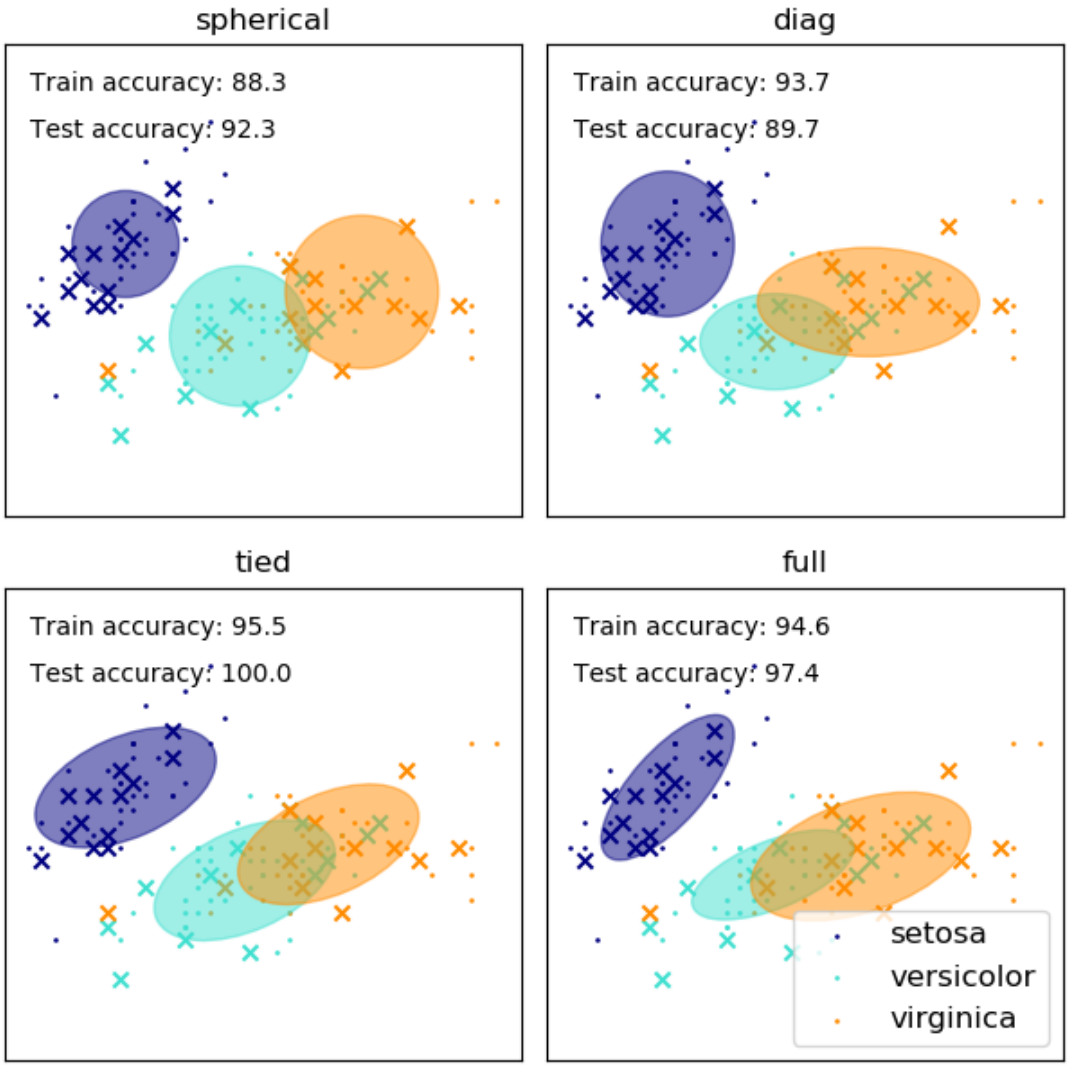
$$\Sigma_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma_k^{(n)} (\mathbf{x}^{(n)} - \mu_k)(\mathbf{x}^{(n)} - \mu_k)^T$$

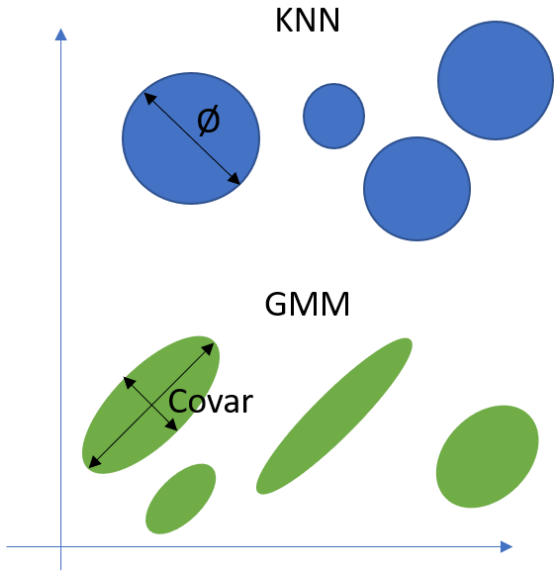
$$\pi_k = \frac{N_k}{N} \quad \text{with} \quad N_k = \sum_{n=1}^{N} \gamma_k^{(n)}$$

  - Evaluate log likelihood and check for convergence

$$\ln p(\mathbf{X}|\pi, \mu, \Sigma) = \sum_{n=1}^{N} \ln \left(\sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}^{(n)}|\mu_k, \Sigma_k) \right)$$




spherical
Train accuracy: 88.3
Test accuracy: 92.3
diag
Train accuracy: 93.7
Test accuracy: 89.7
tied
Train accuracy: 95.5
Test accuracy: 100.0
full
Train accuracy: 94.6
Test accuracy: 97.4
setosa
versicolor
virginica


KNN
Ø
GMM
Covar

How to choose k ?

- Simple: Pick a ‘k’ which generates maximum likelihood for a ‘hold out’ set
- Log-likelihood:

$$\ell(\boldsymbol{\pi}, \mu, \Sigma) = \ln p(\mathbf{X}|\pi, \mu, \Sigma) = \sum_{n=1}^{N} \ln p(\mathbf{x}^{(n)}|\pi, \mu, \Sigma)$$

$$= \sum_{n=1}^{N} \ln \sum_{z^{(n)}=1}^{K} p(\mathbf{x}^{(n)}|\, z^{(n)}; \mu, \Sigma) p(z^{(n)}|\, \boldsymbol{\pi})$$

Better criteria exist → Cross-validation, Information-Theoretic (AIC, BIC)

- The K-Means Algorithm:
  1. Assignment step: Assign each data point to the closest cluster
  2. Refitting step: Move each cluster center to the center of gravity of the data assigned to it
- The EM Algorithm:
  1. E-step: Compute the posterior probability over z given our current model
  2. M-step: Maximize the probability that it would generate the data it is currently responsible for.

Hierarchical Clustering

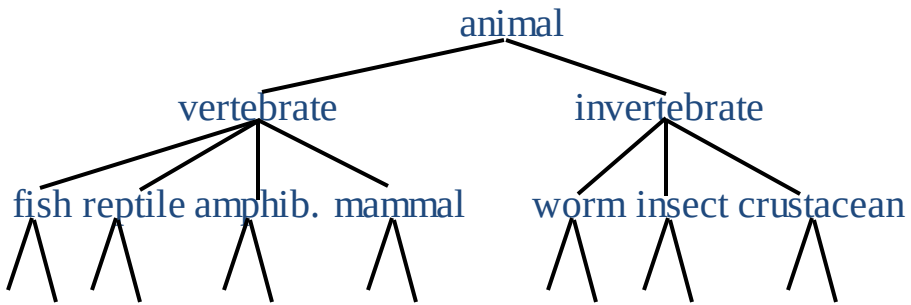
Adapted from Slides by Prabhakar Raghavan, Christopher Manning, Ray Mooney and Soumen Chakrabarti

“The Curse of Dimensionality”

- Why clustering is difficult
  - While clustering looks intuitive in 2 dimensions, many of our applications involve 10,000 or more dimensions…
  - High-dimensional spaces look different
    - The probability of random points being close drops quickly as the dimensionality grows.
    - Furthermore, random pair of vectors are all almost perpendicular.

Hierarchical Clustering

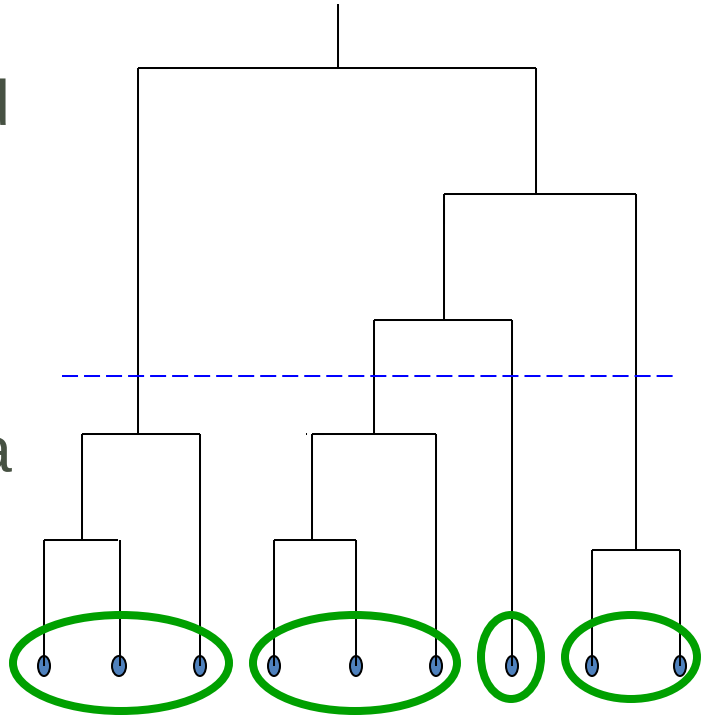
- Build a tree-based hierarchical taxonomy (*dendrogram*) from a set of documents.



- *One approach*: recursive application of a partitional clustering algorithm.

Dendogram: Hierarchical Clustering

- Clustering obtained by cutting the dendrogram at a desired level: each connected component forms a cluster.



Hierarchical Clustering algorithms

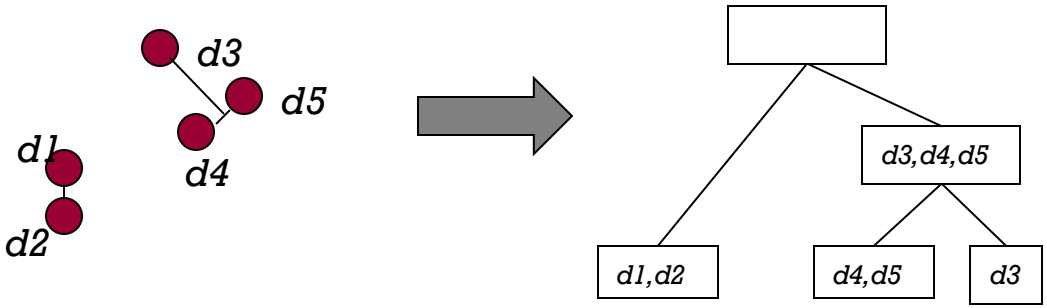
- **Agglomerative (bottom-up):**
  - Start with each document being a single cluster.
  - Eventually all documents belong to the same cluster.
- **Divisive (top-down):**
  - Start with all documents belong to the same cluster.
  - Eventually each node forms a cluster on its own.
- Does not require the number of clusters ***k*** in advance
- Needs a termination/readout condition
  - The final mode in both Agglomerative and Divisive is of no use.

Hierarchical Agglomerative Clustering (HAC) Algorithm

Start with all instances in their own cluster.
Until there is only one cluster:
    Among the current clusters, determine the two clusters, c_i and c_j, that are most similar.
    Replace c_i and c_j with a single cluster $c_i \cup c_j$

Dendrogram: Document Example

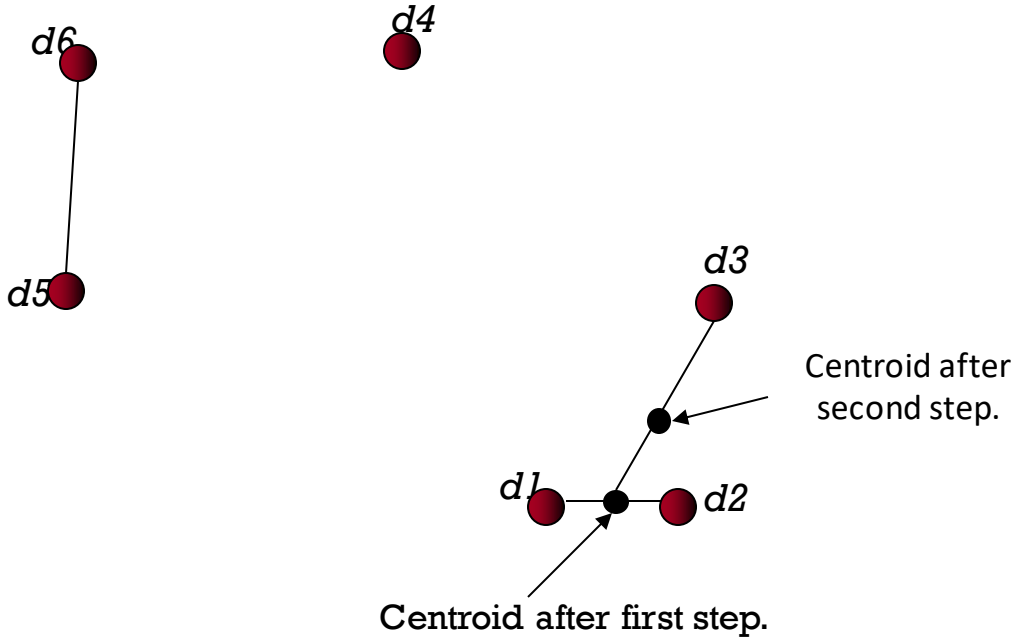
- As clusters *agglomerate*, docs likely to fall into a hierarchy of "topics" or concepts.



Key notion: *cluster representative*

- We want a notion of a representative point in a cluster, to represent the location of each cluster
- Representative should be some sort of "typical" or central point in the cluster, e.g.,
  - point inducing smallest radii to docs in cluster
  - smallest squared distances, etc.
  - point that is the "average" of all docs in the cluster
    - Centroid or center of gravity
    - Measure intercluster distances by distances of centroids.

Example: *n=6, k=3,* closest pair of centroids



Closest pair of clusters

- Many variants to defining closest pair of clusters
- **Single-link**
  - Similarity of the *most* cosine-similar (single-link)
- **Complete-link**
  - Similarity of the "furthest" points, the *least* cosine-similar
- **Centroid**
  - Clusters whose centroids (centers of gravity) are the most cosine-similar
- **Average-link**
  - Average cosine between pairs of elements

Single Link Agglomerative Clustering

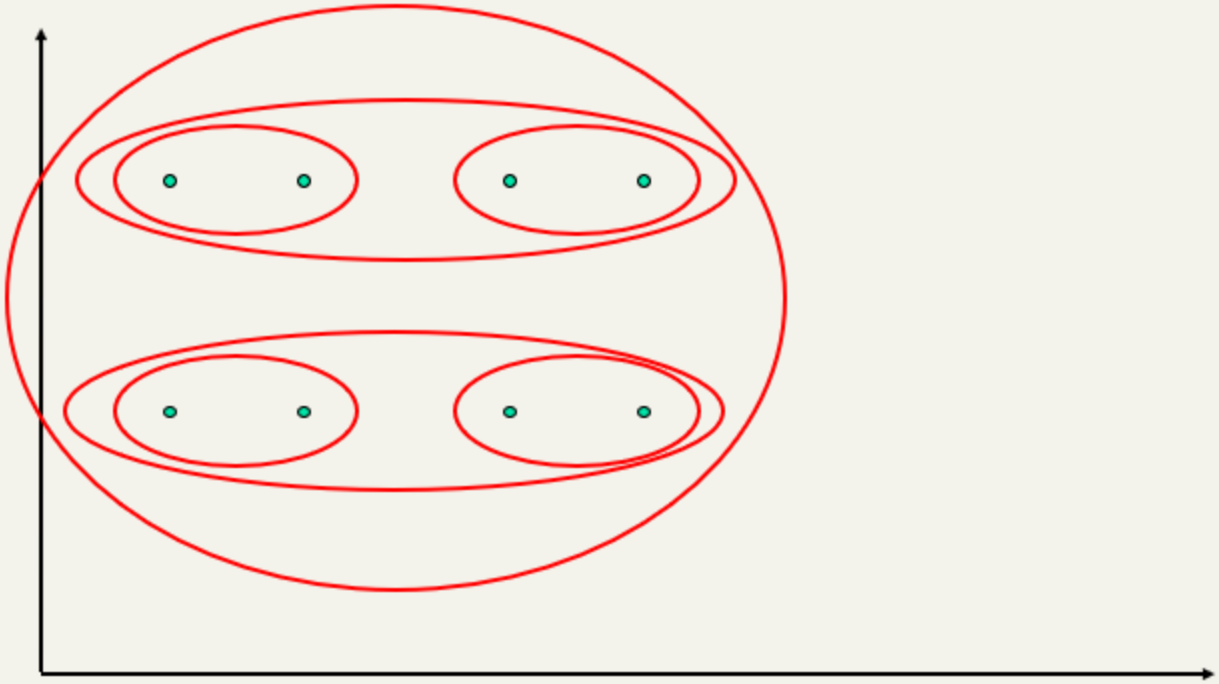
- Use maximum similarity of pairs:

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

- Can result in "straggly" (long and thin) clusters due to chaining effect.
- After merging c_i and c_j, the similarity of the resulting cluster to another cluster, c_k, is:

$$sim((c_i \cup c_j), c_k) = \max(sim(c_i, c_k), sim(c_j, c_k))$$

Single Link Example



Complete Link Agglomerative Clustering

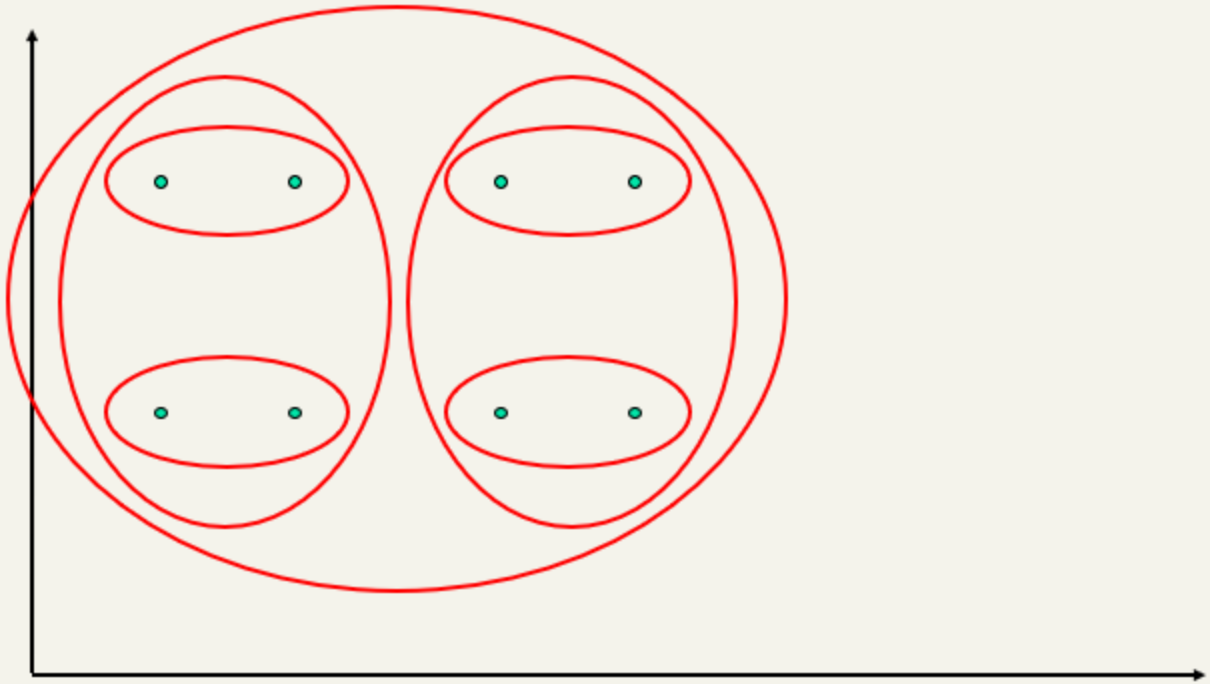
- Use minimum similarity of pairs:

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

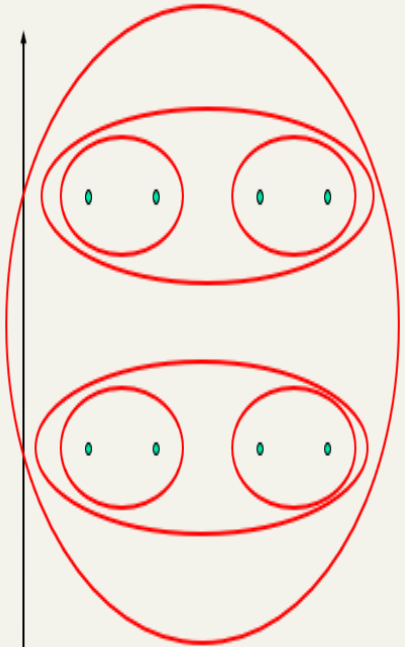
- Makes "tighter," spherical clusters that are typically preferable.
- After merging c_i and c_j, the similarity of the resulting cluster to another cluster, c_k, is:

$$sim((c_i \cup c_j), c_k) = \min(sim(c_i, c_k), sim(c_j, c_k))$$

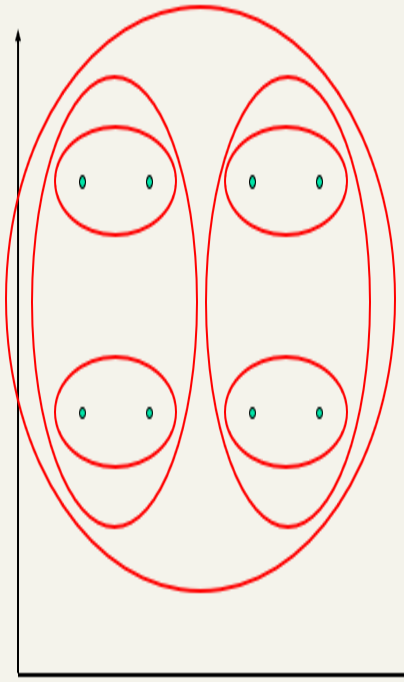
Complete Link Example



Single Link Example



Complete Link Example



Computational Complexity

- In the first iteration, all HAC methods need to compute similarity of all pairs of n individual instances which is $O(n^2)$.
- In each of the subsequent $n-2$ merging iterations, compute the distance between the most recently created cluster and all other existing clusters.
- In order to maintain an overall $O(n^2)$ performance, computing similarity to each cluster must be done in constant time.
  - Else $O(n^2 \log n)$ or $O(n^3)$ if done naively

Group Average Agglomerative Clustering

- Use average similarity across all pairs within the merged cluster to measure the similarity of two clusters.

$$sim(c_i, c_j) = \frac{1}{\left|c_i \cup c_j\right|\left(\left|c_i \cup c_j\right| - 1\right)} \sum_{\vec{x} \in (c_i \cup c_j)} \sum_{\vec{y} \in (c_i \cup c_j): \vec{y} \neq \vec{x}} sim(\vec{x}, \vec{y})$$

- Compromise between single and complete link.
- Two options:
  - Averaged across all ordered pairs in the merged cluster
  - Averaged over all pairs *between* the two original clusters
- Some previous work has used one of these options; some the other. No clear difference in efficacy

Major issue - labeling

- After clustering algorithm finds clusters - how can they be useful to the end user?
- Need pithy label for each cluster
  - In search results, say “Animal” or “Car” in the ***jaguar*** example.
  - In topic trees, need navigational cues.
    - Often done by hand, a posteriori.

How to Label Clusters

- Show titles of typical documents
  - Titles are easy to scan
  - Authors create them for quick scanning!
  - But you can only show a few titles which may not fully represent cluster
- Show words/phrases prominent in cluster
  - More likely to fully represent cluster
  - Use distinguishing words/phrases
    - Differential labeling

What is a Good Clustering?

- *Internal criterion*: A good clustering will produce high quality clusters in which:
  - the intra-class (that is, intra-cluster) similarity is high
  - the inter-class similarity is low
  - The measured quality of a clustering depends on both the document representation and the similarity measure used

External criteria for clustering quality

- Quality measured by its ability to discover some or all of the hidden patterns or latent classes in gold standard data
- Assesses a clustering with respect to ground truth
- Assume documents with C gold standard classes, while our clustering algorithms produce K clusters, $\omega_1, \omega_2, \ldots, \omega_K$ with n_i members.

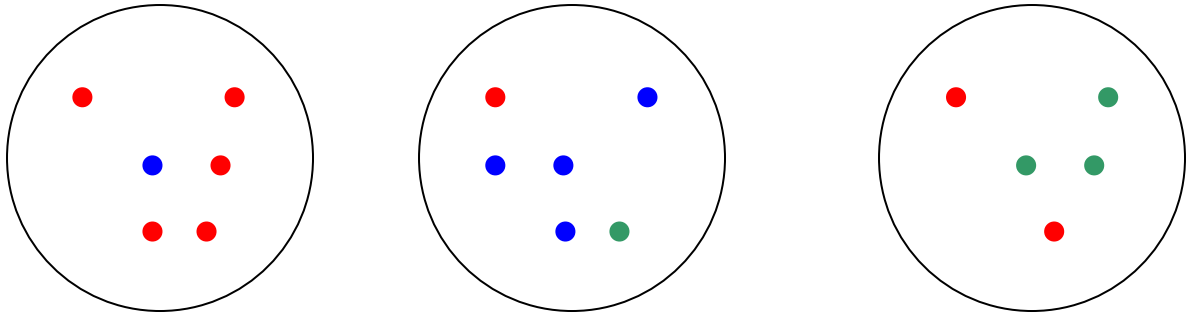
External Evaluation of Cluster Quality

- *Simple measure*: purity, the ratio between the dominant class in the cluster π_i and the size of cluster ω_i

$$Purity(\omega_i) = \frac{1}{n_i} \max_j (n_{ij}) \quad j \in C$$

- Others are entropy of classes in clusters (or mutual information between classes and clusters)

Purity example



Cluster I

Cluster II

Cluster III

Cluster I: Purity = 1/6 (max(5, 1, 0)) = 5/6

Cluster II: Purity = 1/6 (max(1, 4, 1)) = 4/6

Cluster III: Purity = 1/5 (max(2, 0, 3)) = 3/5

Evaluation of clustering

- Perhaps the most substantive issue in data mining in general:
  - how do you measure goodness?
- Most measures focus on computational efficiency
  - Time and space
- For application of clustering to search:
  - Measure retrieval effectiveness

References

- PRML (Bishop) – Chapter 9: 9.1,9.2,9.3.2
- Pattern Classification (Duda, Hart, Stork)
  - 10.4.3,10.6.1,10.7.1,10.7.2, 10.8, 10.10
  - 10.9 (Hierarchical Clustering)