

CLUSTERING

JELENA JOVANOVIĆ

Email: jeljov@gmail.com

Web: <http://jelenajovanovic.net>

OUTLINE

- What is clustering?
- Application domains
- K-Means clustering
 - Understanding it through an example
 - The K-Means algorithm
 - Some challenging issues
 - An example in WEKA

WHAT IS CLUSTERING?

Clustering is an unsupervised learning task

- its input is a set of instances (described with a set of attributes) to be grouped based on their similarity
- there is no data about the desired/correct group for any of the input instances

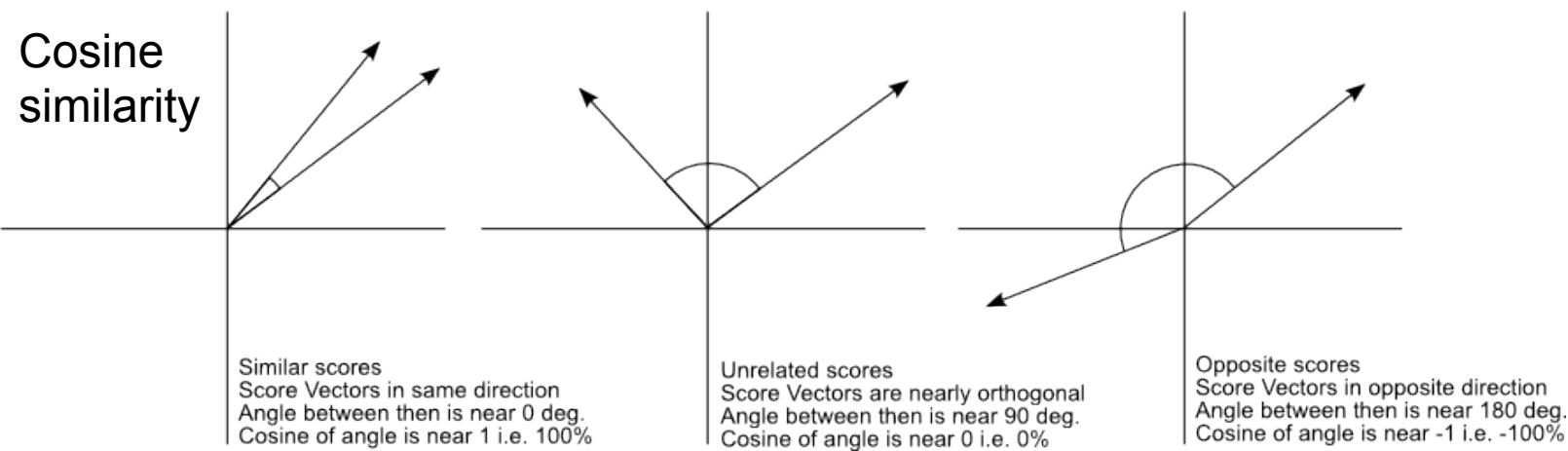
WHAT IS CLUSTERING?

It is about grouping instances in such a manner that for each instance the following is true:

- the instance is more *similar* to the instances from its group (cluster), than to instances from other groups (clusters)

Similarity between instances is computed using certain

- similarity measure (e.g., Cosine similarity), or
- distance measure (e.g., Euclidian distance)

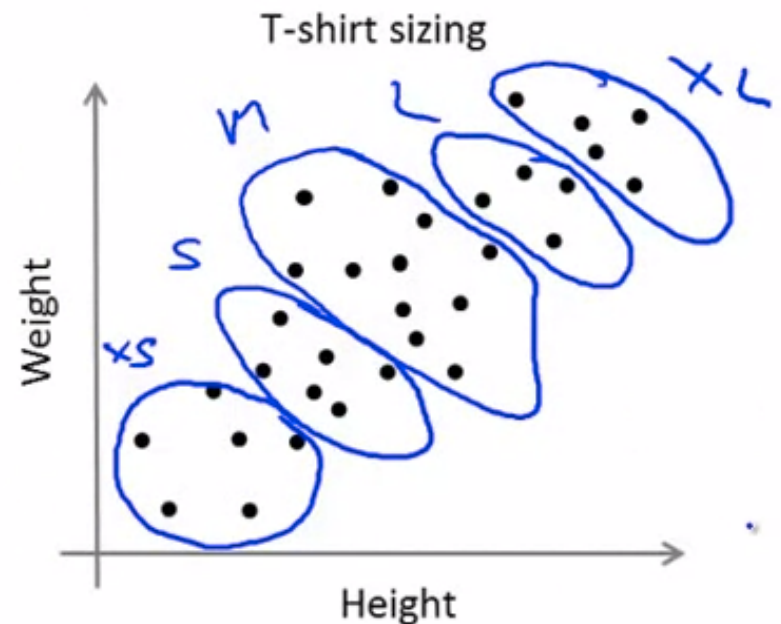
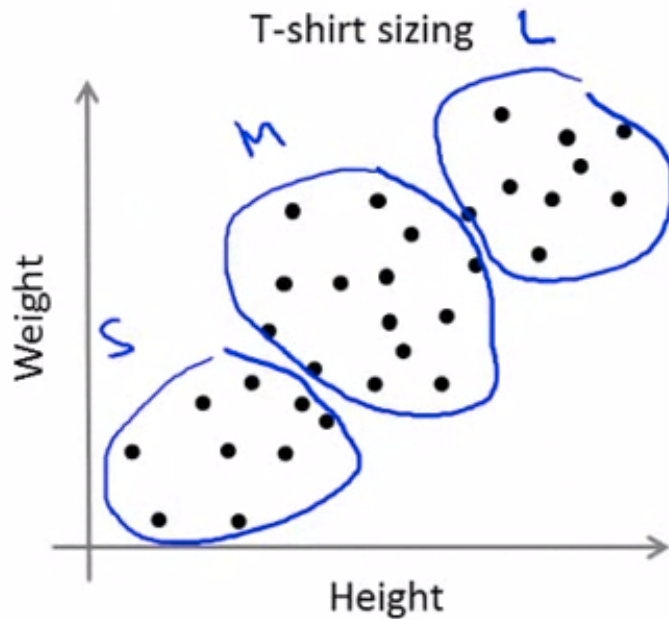
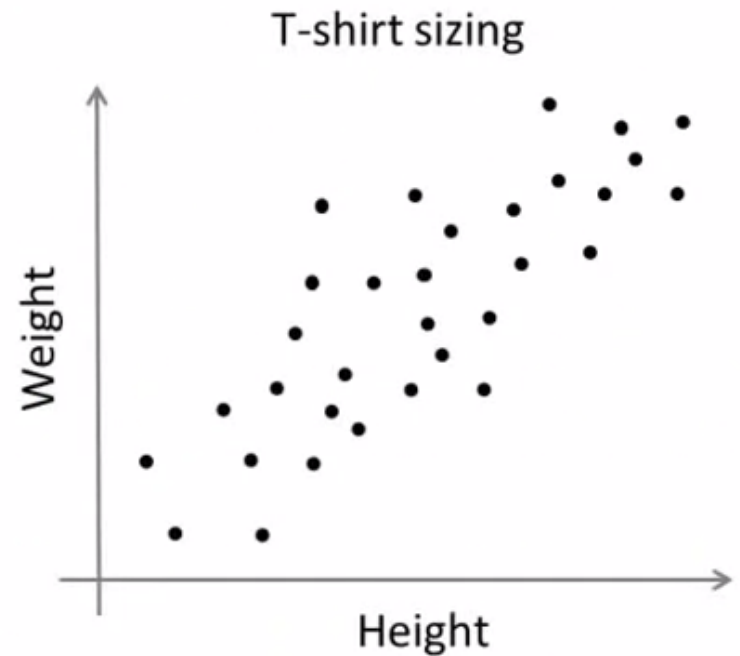


WHAT IS CLUSTERING?

Unlike the classification task, for this task, there is no unique “correct” or best solution

- how good/suitable a solution is, that depends upon the specific domain and the application case
 - the same solution might be differently evaluated in different application cases
- if it is to be done properly, domain experts need to evaluate the solution(s) produced by the model

An example illustrating different valid solutions for the same input dataset



APPLICATION DOMAINS

- Market segmentation
- Detection of groups/communities in social networks
- Identifying patterns in user tracking data -> allow for learning about the ways people use an application
- Grouping of objects (e.g., images or documents) based on their common characteristics
- ...

K-MEANS ALGORITHM

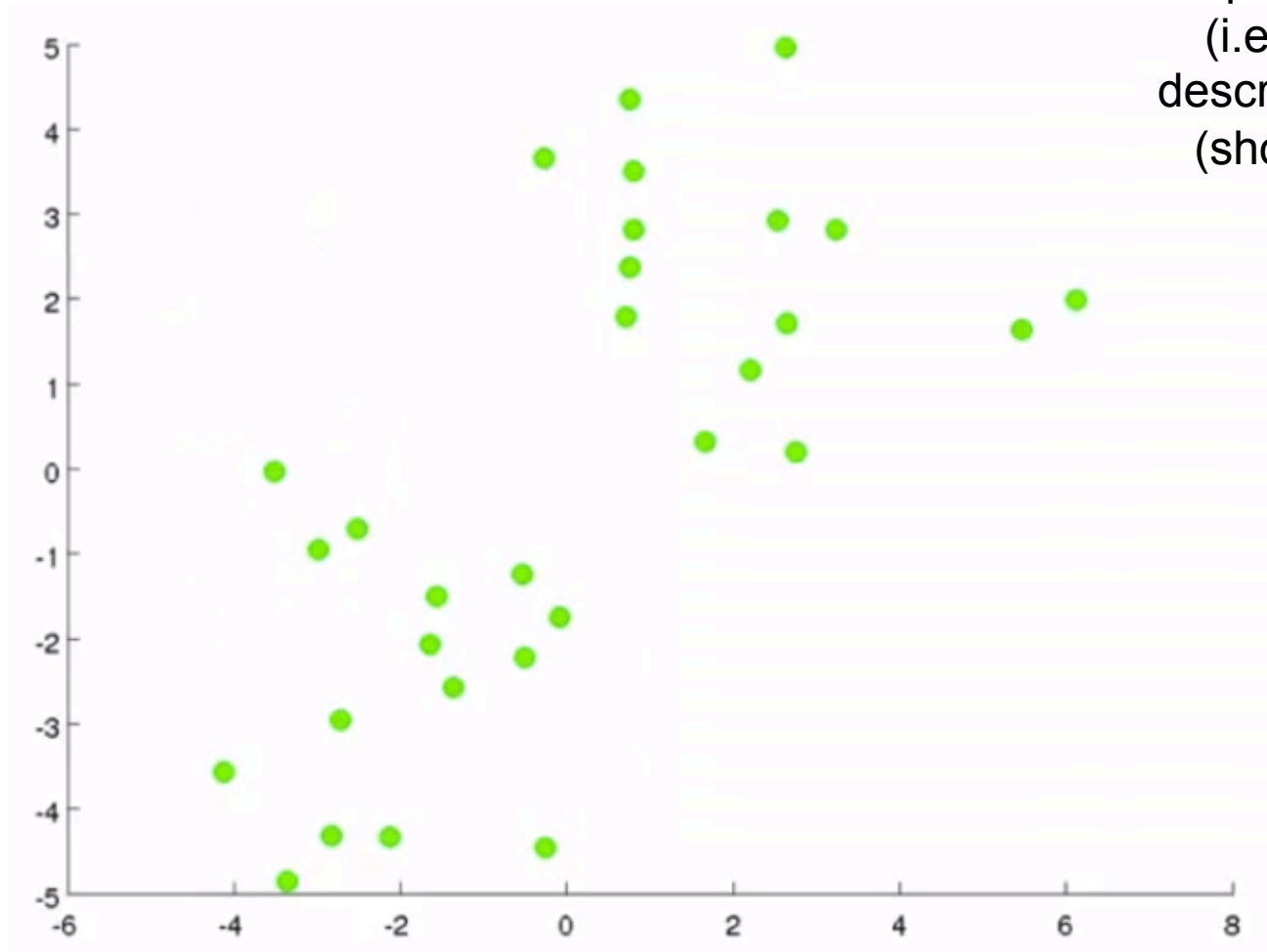
K-MEANS

One of the simplest and the most widely known and used clustering algorithm

It can be best understood through examples, so we will first have a look at an example

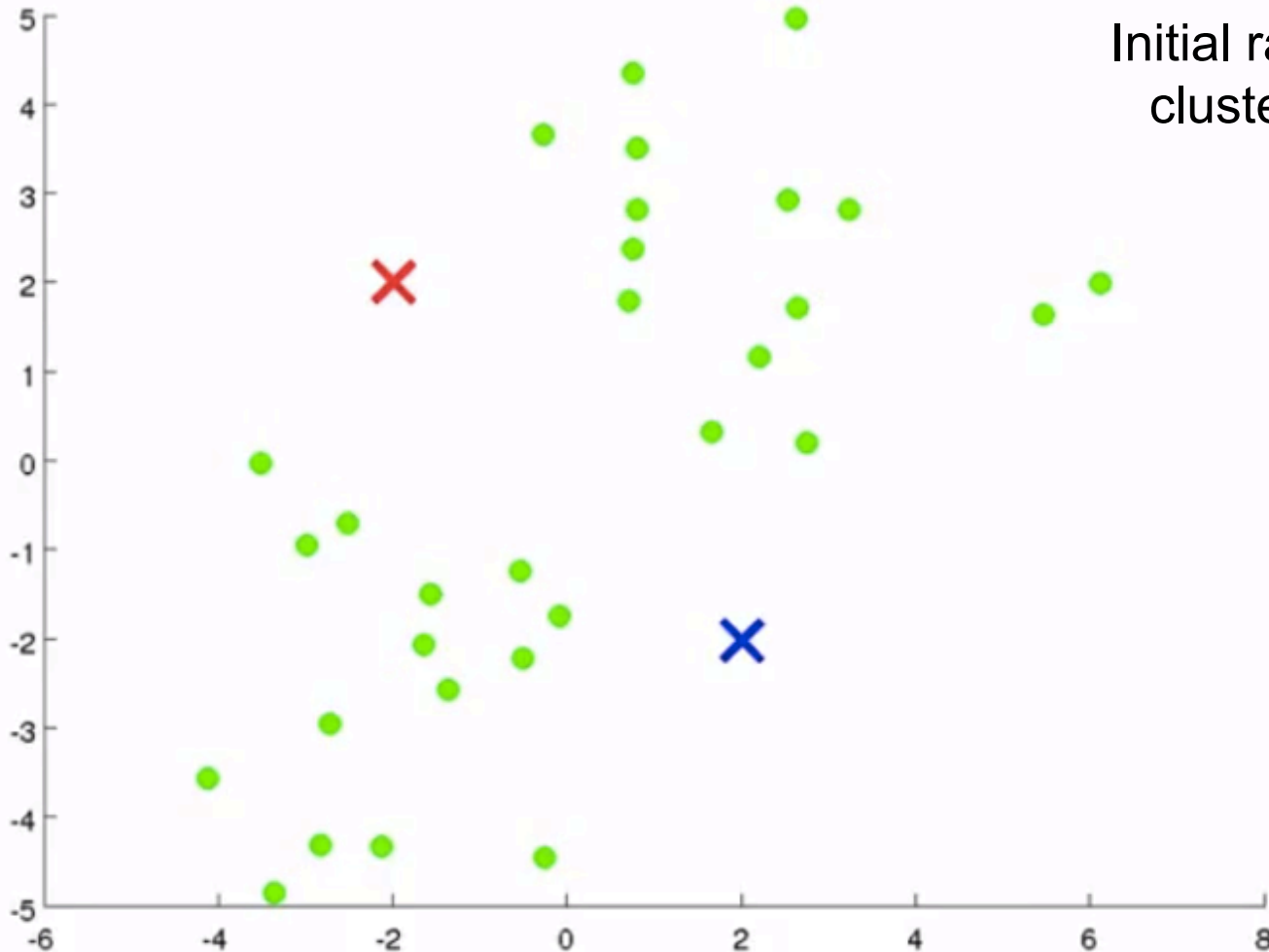
K-MEANS: AN EXAMPLE

Let's suppose the diagram presents the input data (i.e., a set of instances), described with 2 attributes (shown on x and y axes)

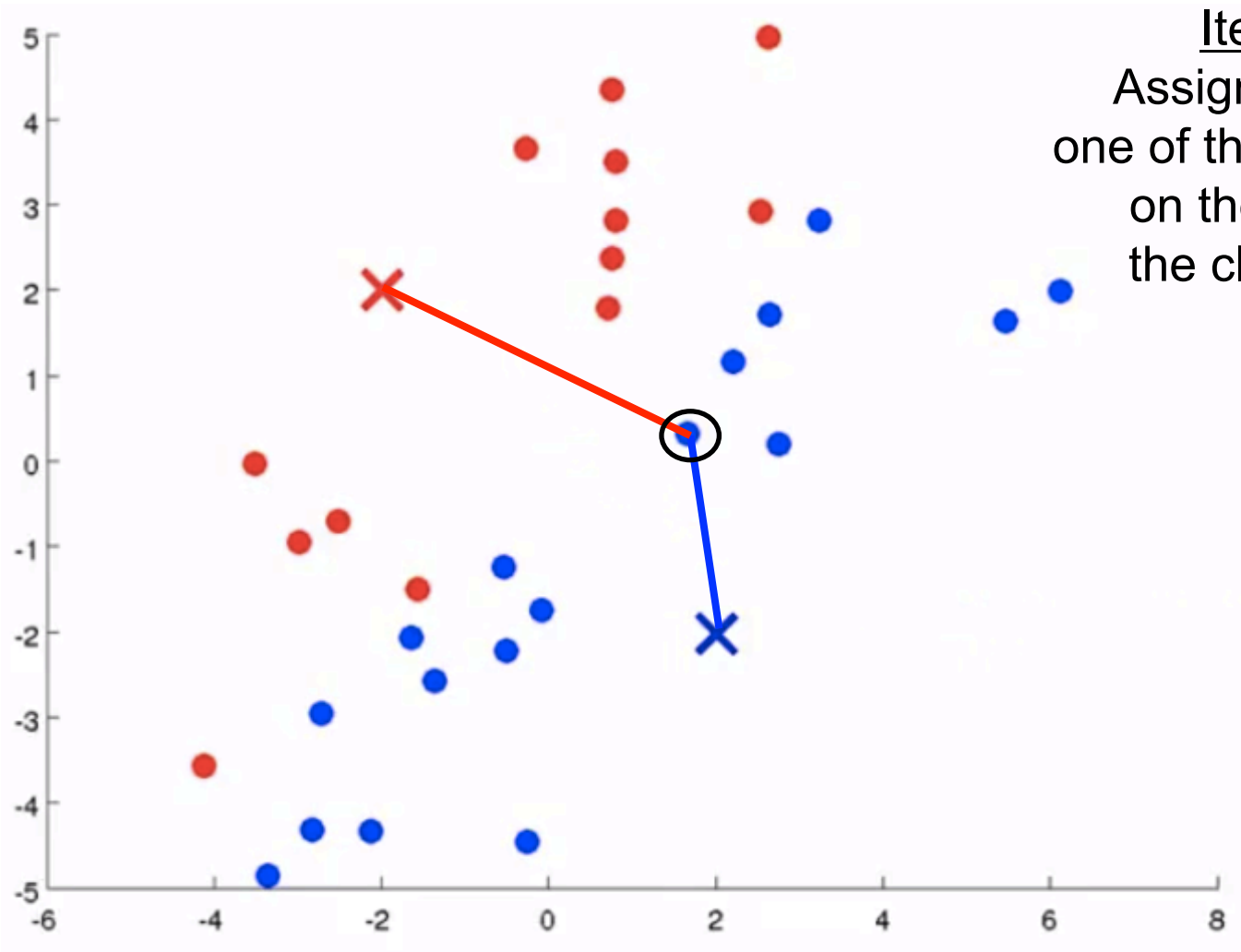


K-MEANS: AN EXAMPLE

Initialization:
Initial random selection of
cluster centroids ($K = 2$)



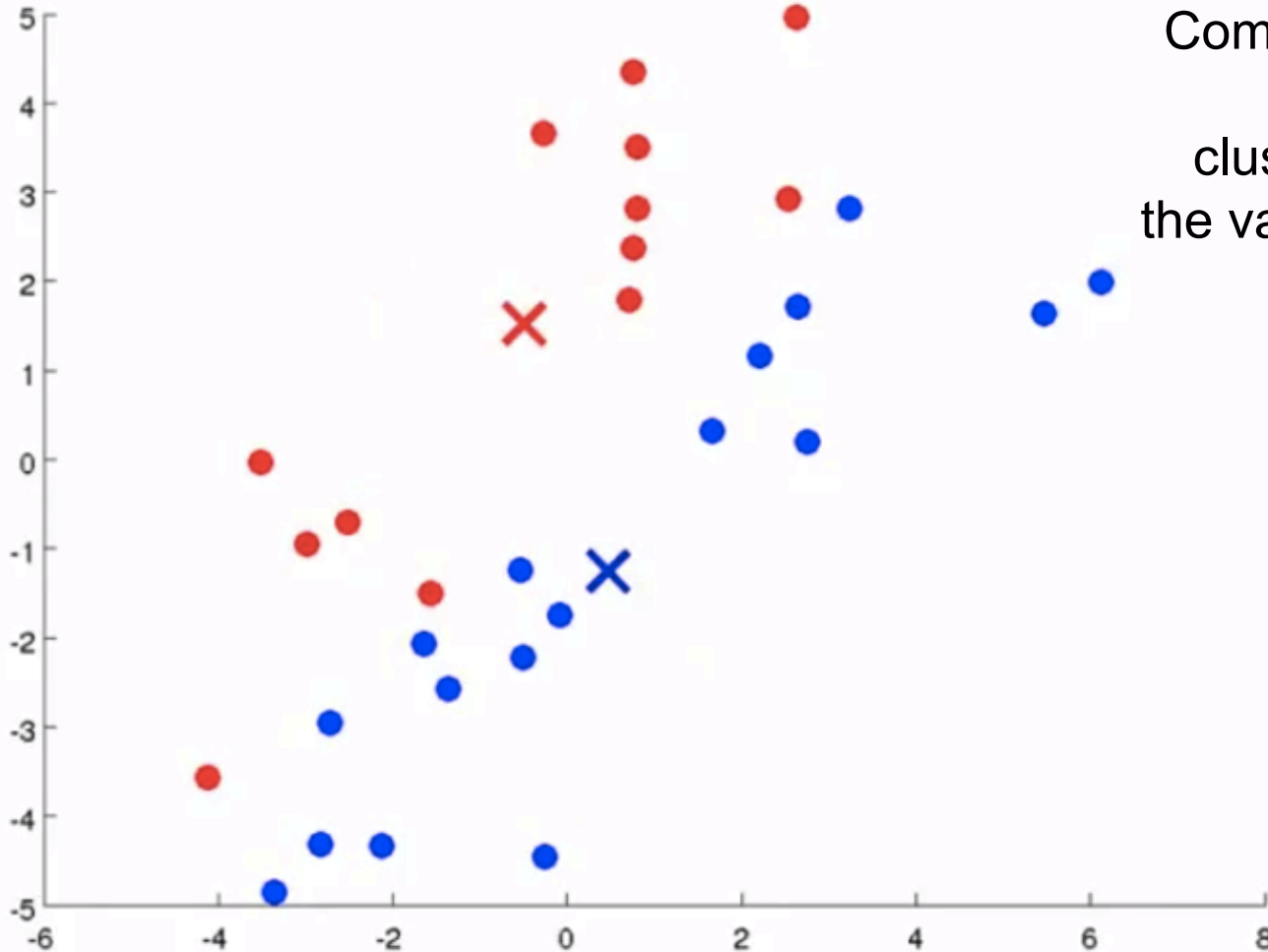
K-MEANS: AN EXAMPLE



Iteration 1, Step 1:
Assigning instances to one of the clusters based on their distance from the clusters' centroids

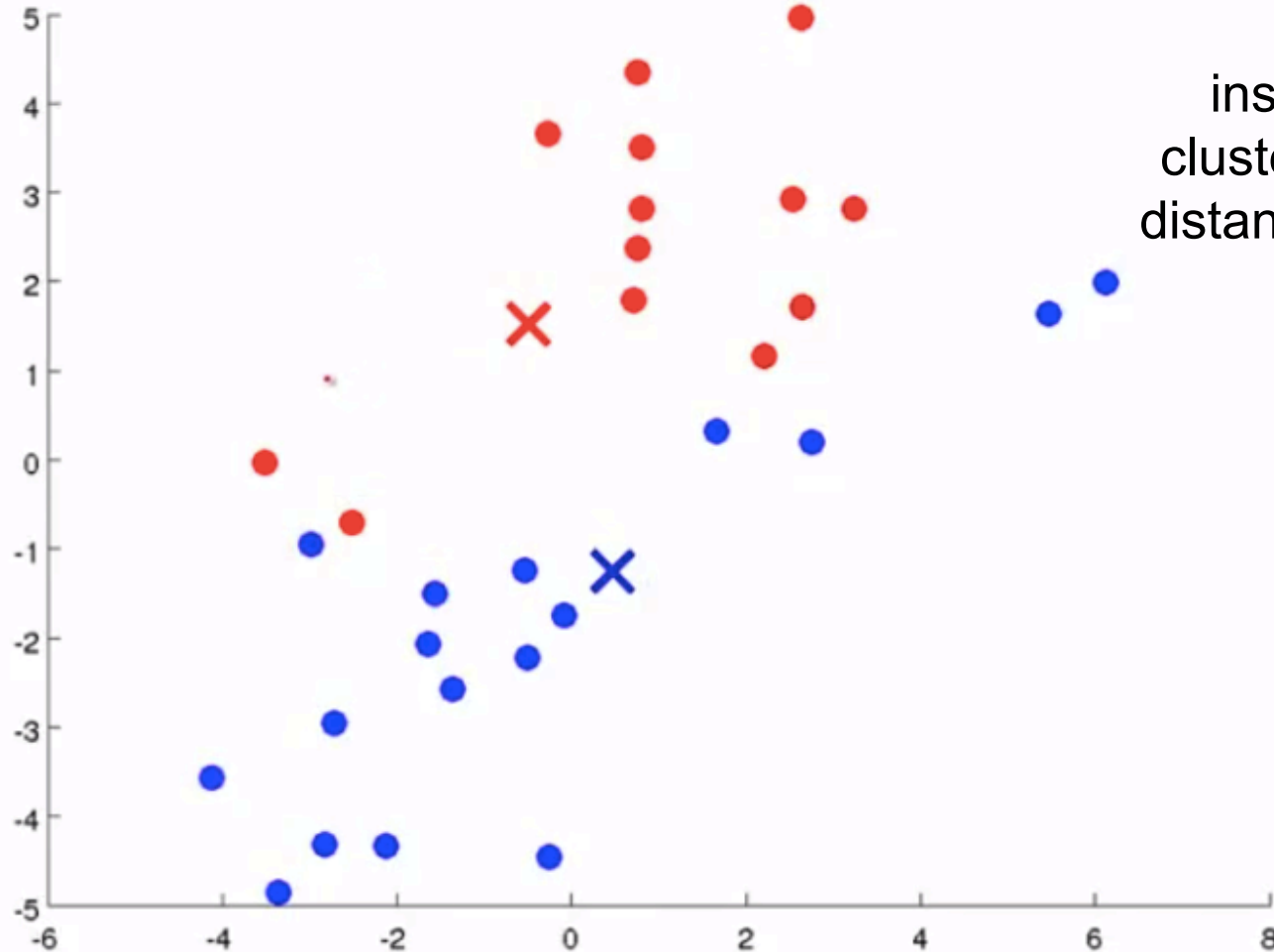
K-MEANS: AN EXAMPLE

Iteration 1, Step 2:
Computation of a new centroid for each cluster, by averaging the values of instances within the cluster

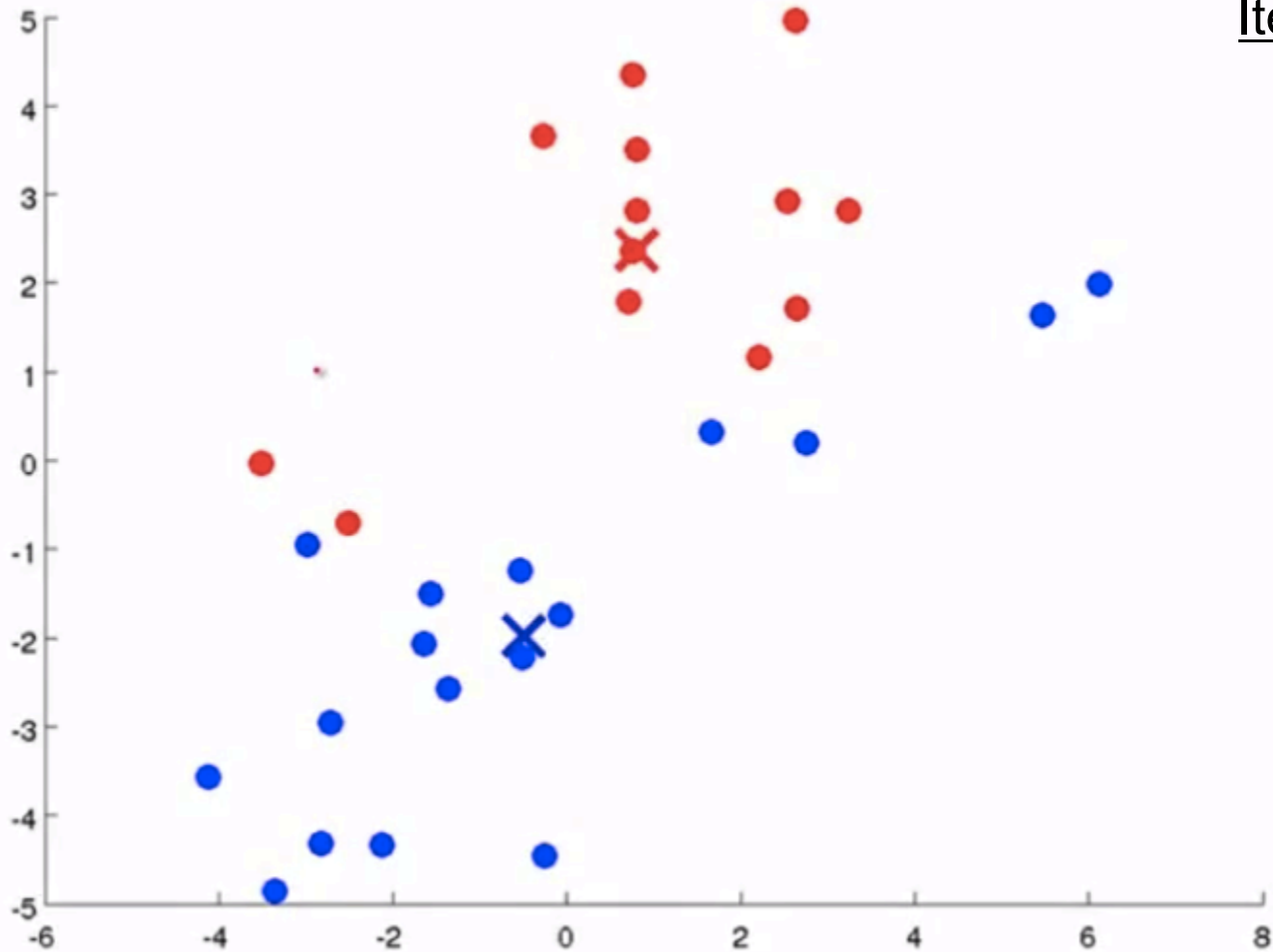


K-MEANS: AN EXAMPLE

Iteration 2, Step 1:
Re-assignment of instances across the clusters based on their distance from the (new) cluster centroids

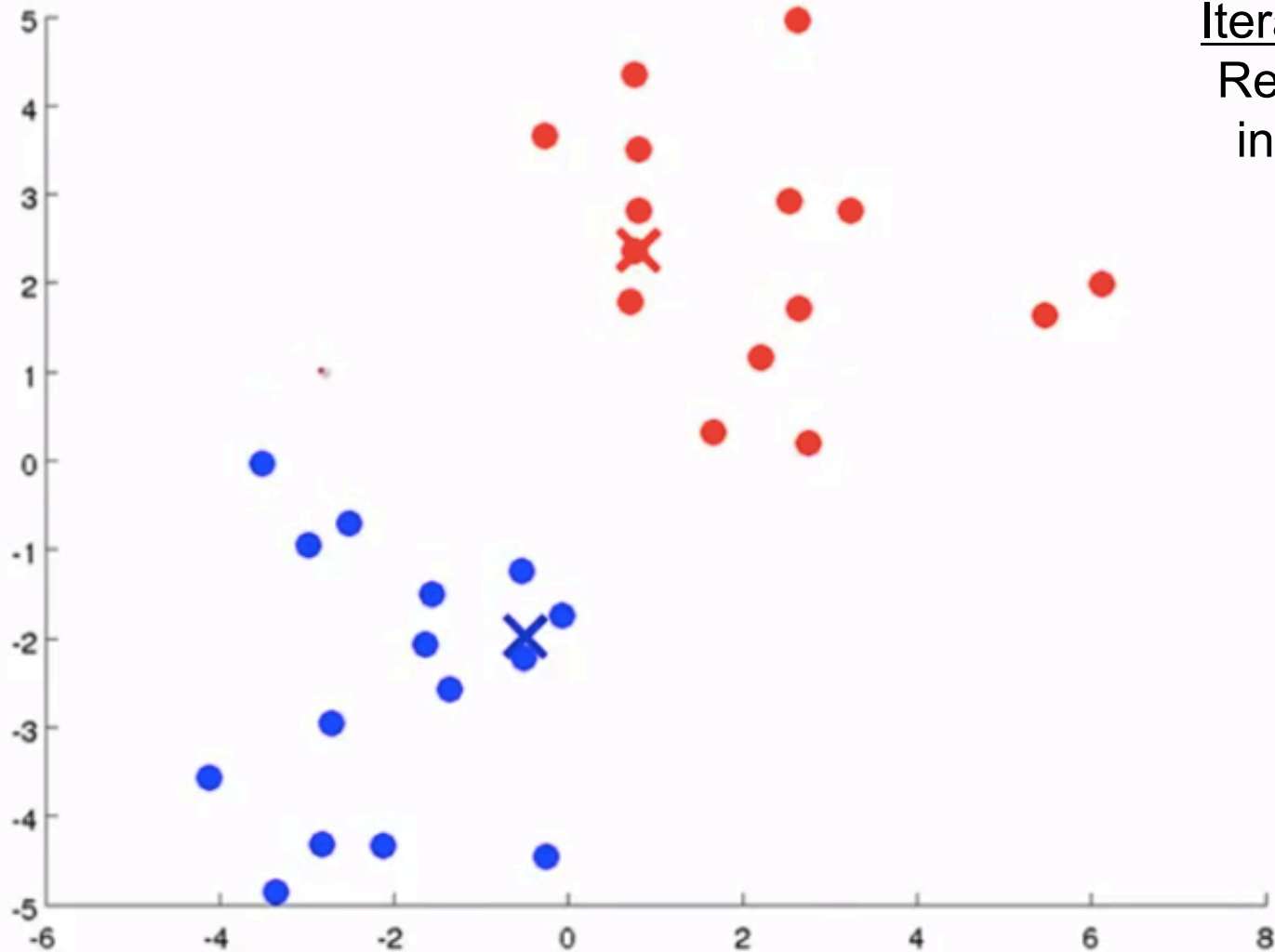


K-MEANS: AN EXAMPLE



Iteration 2, Step 2:
Re-calculation of
cluster centroids

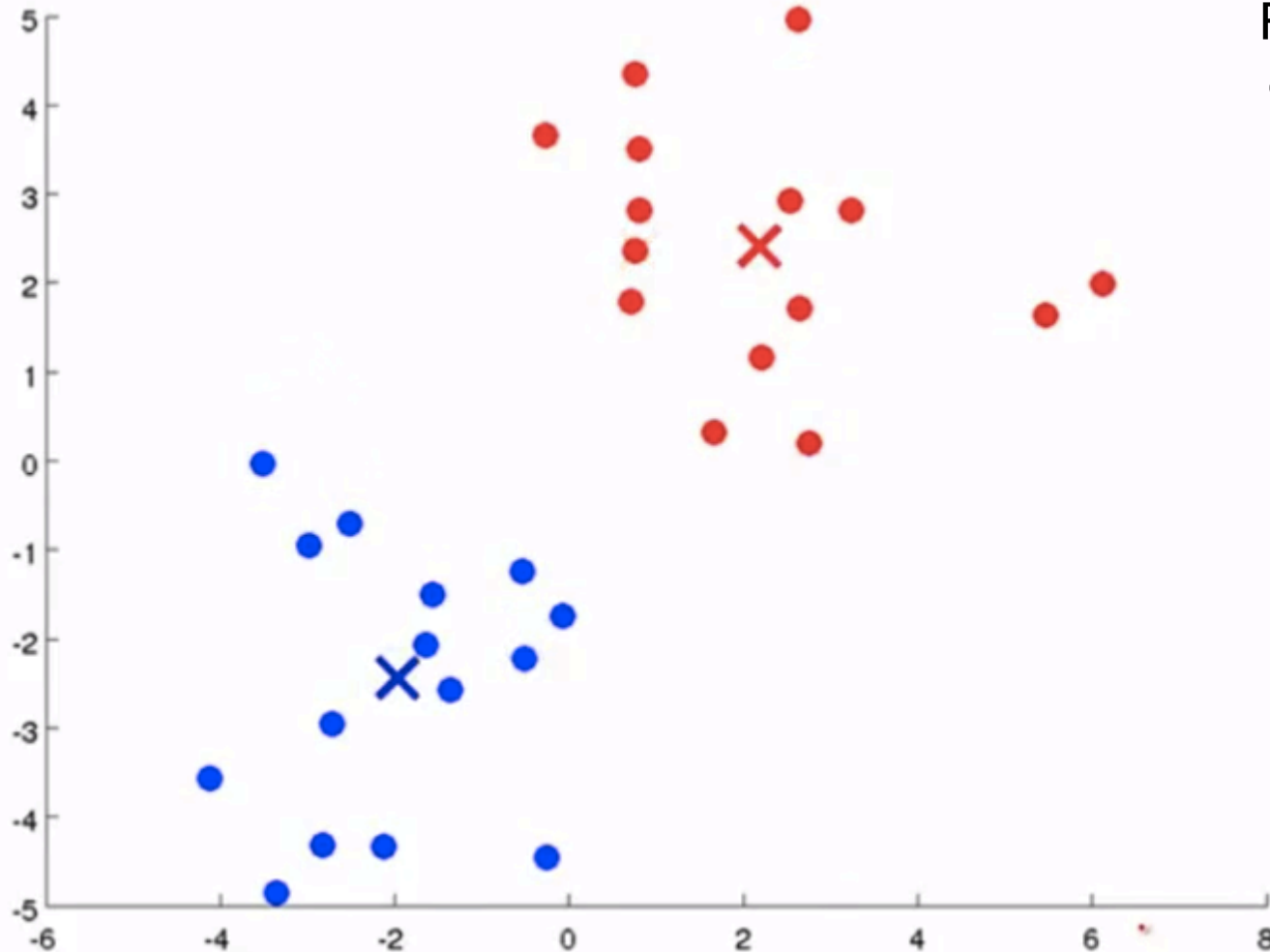
K-MEANS: AN EXAMPLE



Iteration 3, Step 1:
Re-assignment of
instances across
the clusters

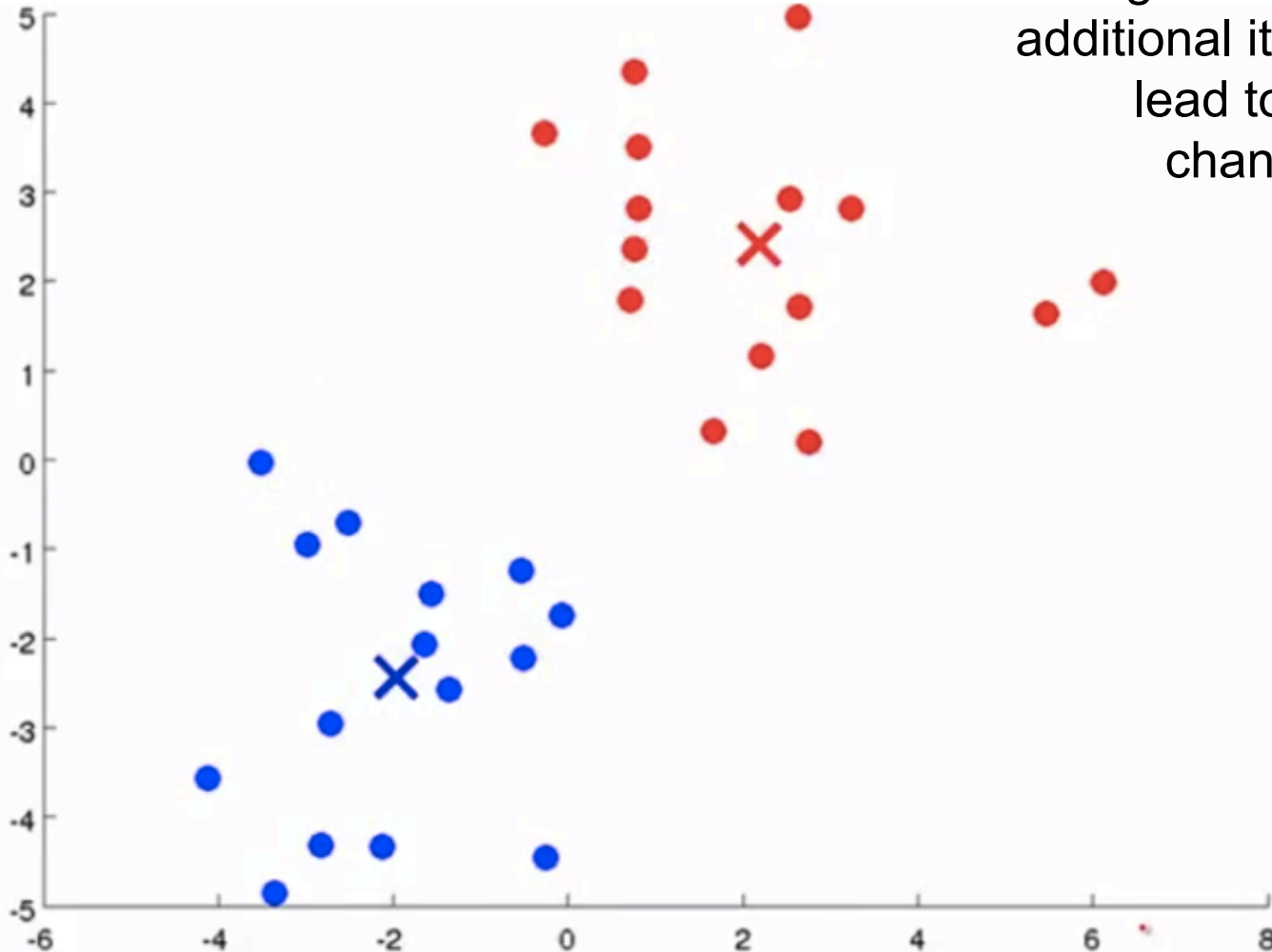
K-MEANS: AN EXAMPLE

Iteration 3, Step 2:
Re-calculation of
cluster centroids



K-MEANS: AN EXAMPLE

The algorithm is converging:
additional iterations will not
lead to any significant
change; the process
terminates



K-MEANS: THE ALGORITHM

Input:

- K – the number of clusters
- (unlabeled) training set with m instances; each instance in this set is described with a vector of n attributes (x_1, x_2, \dots, x_n)
- *max* - max number of iterations (optional parameter)

K-MEANS: THE ALGORITHM

Steps:

- 1) Initial, random selection of a centroid for each cluster
 - centroids are chosen from the training set, i.e., K instances are randomly taken from the training set and set as centroids
 - 2) Repeat:
 - 1) *Cluster assignment*: for each instance i from the training set, $i = 1, m$, identify the closest centroid and assign the instance to the corresponding cluster
 - 2) *Repositioning of centroids*: for each cluster, compute a new centroid by averaging the values of instances assigned to that cluster
- until the algorithm starts converging or the number of iterations reaches max

K-MEANS: THE COST FUNCTION

The objective of the K-means algorithm is to *minimize the cost function J* :

$$J(c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K) = \frac{1}{m} \sum_{i=1}^m \|x^{(i)} - \mu_{c^{(i)}}\|^2$$

$x^{(i)}$ – i -th instance in the training dataset, $i=1, m$

$c^{(i)}$ – index of the cluster to which the instance $x^{(i)}$ is currently assigned

μ_j – centroid of the cluster j , $j=1, K$

$\mu_{c^{(i)}}$ – centroid of the cluster to which the instance $x^{(i)}$ has been assigned

This function is also known as *distortion function*

K-MEANS: THE COST FUNCTION

$$\min_{\substack{c^{(1)}, \dots, c^{(m)}, \\ \mu_1, \dots, \mu_K}} J(c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K)$$

K-means algorithm minimizes the cost function \mathbf{J} in the following manner:

- the *Cluster assignment* phase minimizes \mathbf{J} with respect to $c^{(1)}, \dots, c^{(m)}$, holding μ_1, \dots, μ_K fixed
- the *Repositioning of centroids* phase minimizes \mathbf{J} with respect to μ_1, \dots, μ_K , holding $c^{(1)}, \dots, c^{(m)}$ fixed

K-MEANS: EVALUATION

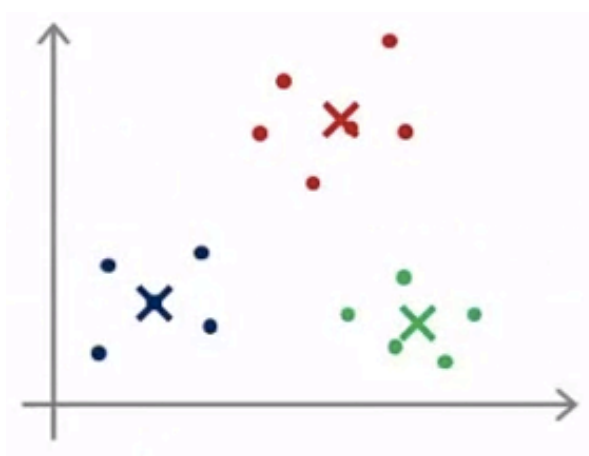
Criteria for evaluating the “quality” of the resulting clusters:

- Distance between the centroids
 - the more distant the centroids are, the lower is the overlap between the clusters, and thus their quality is higher
- St. deviation of instances from the centroid
 - the lower the st. deviation, the more tightly grouped are the instances, and thus, the clusters are considered better
- Within cluster sum of squared errors
 - a quantitative measure for estimating the quality of the clusters
 - we will consider it through an example (slide 23)

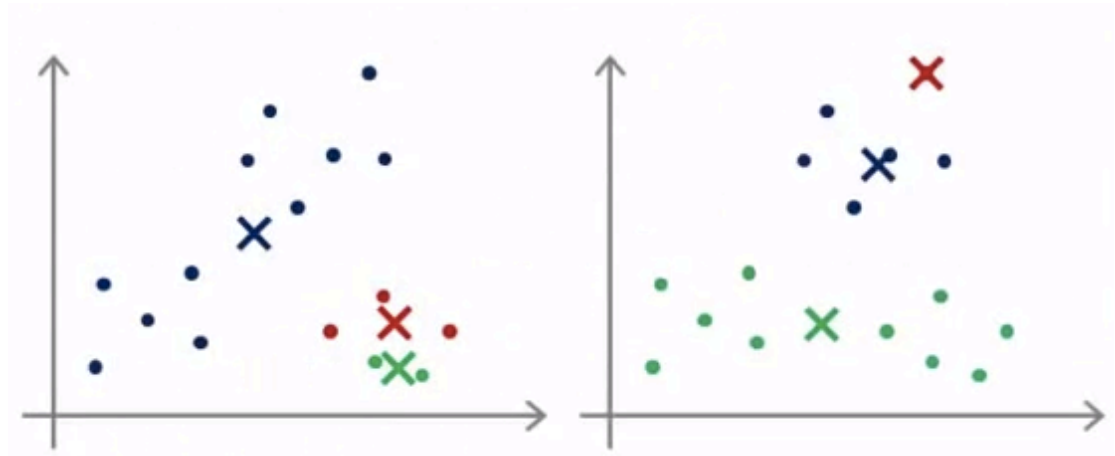
K-MEANS:

INITIAL SELECTION OF CENTROIDS

- Depending on how initial cluster centroids were chosen, the K-means algorithm would converge quicker or slower
- “Unlucky” selection of initial centroids may lead K-Means to get stuck in the so called *local optima* and produce poor results
 - this is a local minimum of the *cost function*



“Lucky” initialization



“Unlucky” initializations that lead to a local minimum

K-MEANS:

MULTIPLE RANDOM INITIALIZATIONS

It allows for avoiding situations that lead K-means in a local minimum

Consists of the following:

```
for i = 1 to n { //n is often in the range 50-1000
    Randomly select the initial set of centroids;
    Apply the K-Means algorithm;
    Compute the cost function
}
```

Choose the instance of the algorithm that produces the lowest value of the cost function

This approach gives good results if the number of clusters is relatively low (2 - 10); should not be used if the number of clusters is higher

Another option: [K-means++ algoritam](#)

K-MEANS: HOW TO CHOOSE K ?

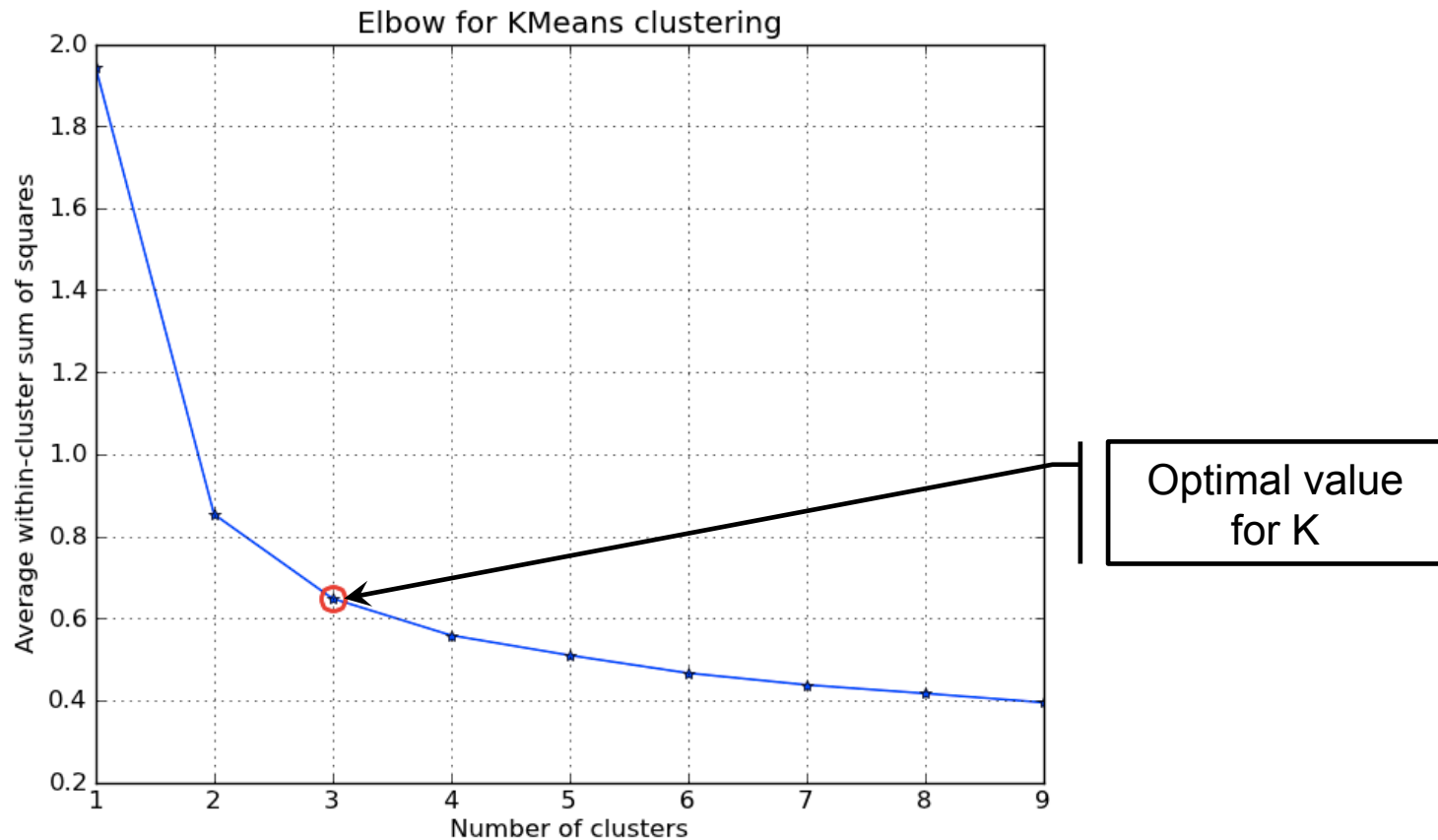
How to determine the number of clusters K?

- In case we have domain knowledge about the phenomenon described by the data
 - Make an assumption about the number of clusters (K) based on the domain knowledge
 - Test the model with K-1, K, K+1 clusters and compare the error*
- If we lack domain knowledge about the studied phenomenon
 - Start with a small number of clusters and in multiple iterations test the model by incrementally increasing the number of clusters
 - In each iteration, compare the error* of the current and the previous model, and when the error reduction becomes insignificant, terminate the process

*E.g., within cluster sum of squared errors can be used for the comparison

K-MEANS: HOW TO CHOOSE K ?

When we lack domain knowledge about the studied phenomenon



K-MEANS: AN EXAMPLE IN WEKA

The example we will see is taken from an article, published at the *IBM Developer Works* Web site:

<http://www.ibm.com/developerworks/library/os-weka2/>

EXPECTATION MAXIMIZATION (EM) ALGORITHM

PROBABILISTIC CLUSTERING

EM is used for probabilistic clustering

From a probabilistic perspective

- instances should not be placed categorically in one cluster or the other,
- instead, they have a certain probability of belonging to each cluster

The rationale: no finite amount of evidence is enough to make a completely firm decision on how to do the clustering

FINITE MIXTURES MODEL

The foundation for statistical clustering is a statistical model called *finite mixtures*

A *mixture* is a set of k probability distributions, representing k clusters

- each distribution governs the attribute values for members of the corresponding cluster, that is,...
- it gives the probability that a particular instance would have a certain set of attribute values if it was a member of that cluster

In addition, the clusters are not equally likely: there is a probability distribution that reflects their *prior probability*

THE SIMPLEST FINITE MIXTURE MODEL

The simplest finite-mixture model:

- instances are described with just one numeric attribute that has a Normal distribution in all clusters (K clusters)
- each cluster (C_i) has its specific mean (μ_i) and st. deviation (σ_i) – i.e., parameters of the Normal distribution
- p_i is the *prior probability* of the cluster C_i

FINITE MIXTURE PROBLEM

Suppose we've been given a set of instances that originate from a simplest finite mixture model with K clusters

What we do not know are

- specific clusters that each instance originates from
- parameters of the model $(\mu_i, \sigma_i, p_i, i=1, k)$.

The problem of determining the parameters of this model based on the given set of instances is known as the *finite mixture problem*.

EM ALGORITHM AS A SOLUTION OF THE FINITE MIXTURE PROBLEM

To resolve the finite mixture problem, we can apply the procedure we used for the K-means algorithm:

- 1) start with initial guesses for the model parameters ($\mu_i, \sigma_i, \rho_i, i=1, k$)
- 2) for the given parameter values, calculate the cluster probabilities for each instance
- 3) use the computed probabilities to re-estimate the parameters

Repeat steps 2) and 3) until the parameter values start to converge

This procedure is a simplified description of the EM algorithm

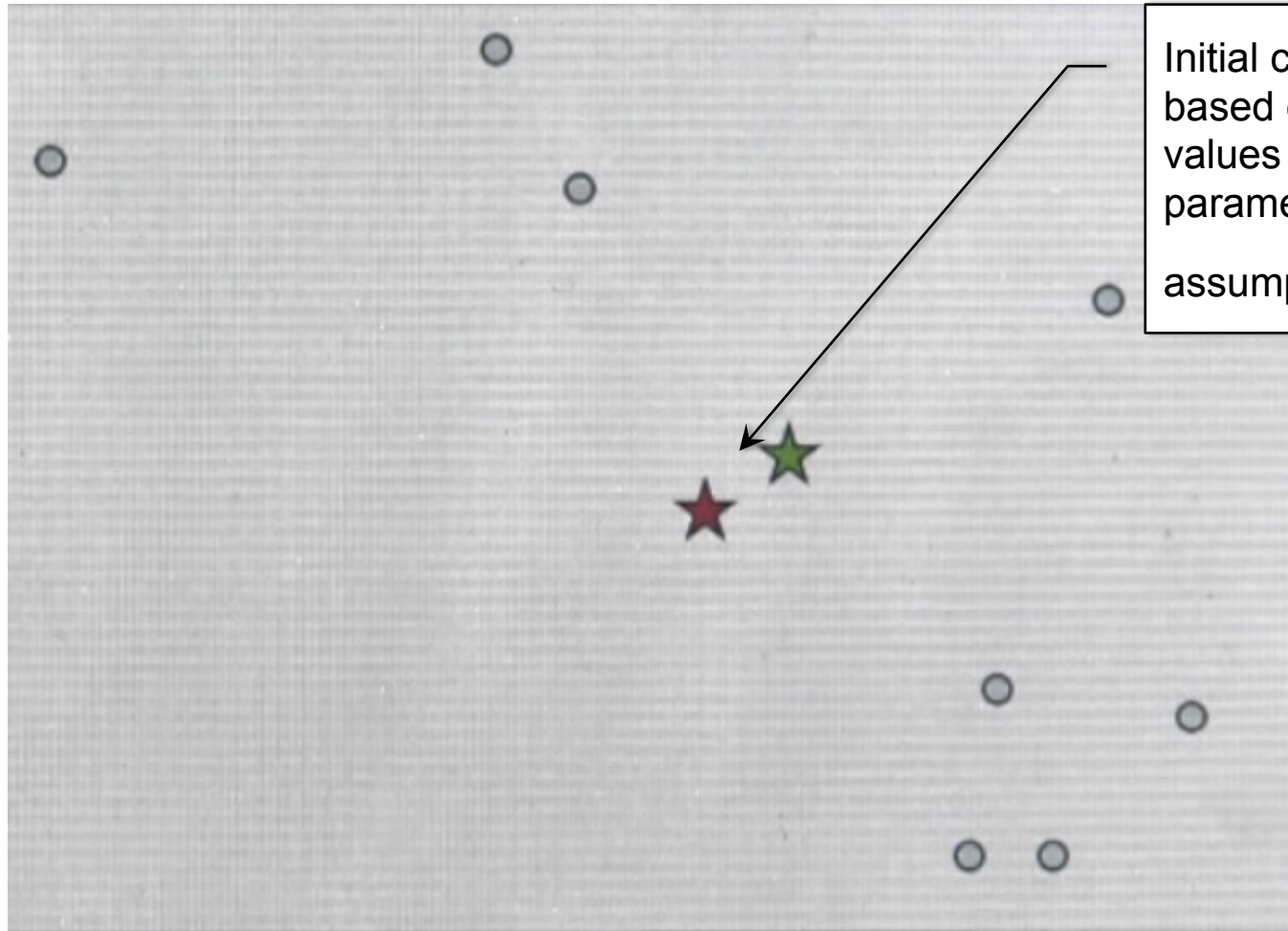
EM ALGORITHM

The EM algorithm consists of 2 key steps:

- **E (expectation) step** – calculation of the cluster probabilities; in this step we assume that we know the values of all the parameters;
- **M (maximization) step** – calculation of the model parameters; we aim to “maximize” the likelihood of the model given the data available.

These steps are repeated until the algorithm starts to converge

EM ALGORITHM: INITIALIZATION



The example is taken from the AI course: <https://www.udacity.com/course/cs271>

EM ALGORITHM: E STEP

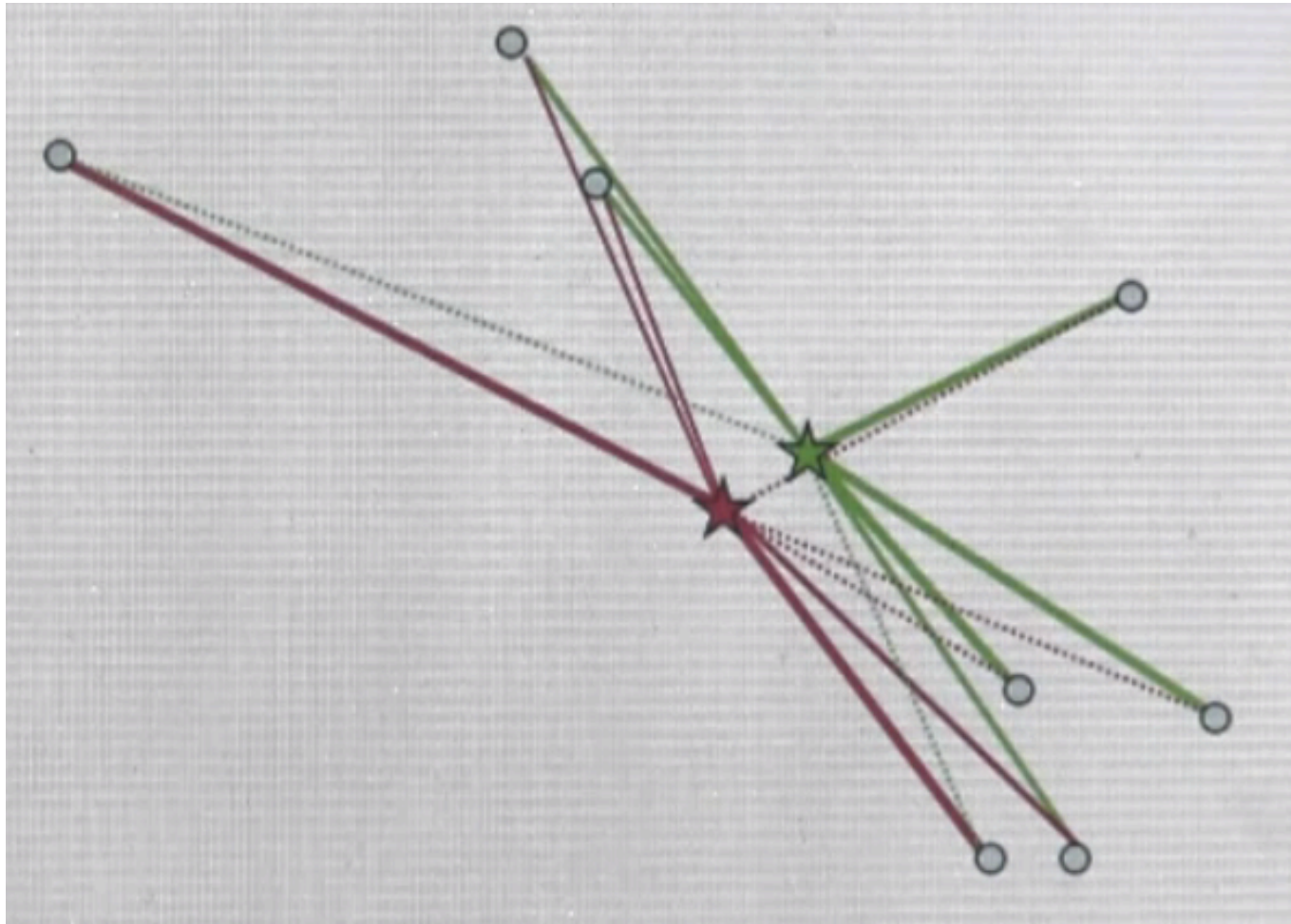
For each instance from the dataset x_j ($j=1,n$), we compute the probability that it belongs to the cluster C_i , $i=1,k$

$$e_{ij} = p_i * P(x_j | C_i)$$

$P(x_j|C_i)$ is computed using the formula of the Normal distribution $f(x; \mu, \sigma)$

Reminder: in this step we assume that the values of all the model parameters – $\mu_i, \sigma_i, p_i, i=1,k$ – are known

EM ALGORITHM: E STEP



The width of a line indicates the probability that an instance belongs to a certain cluster, i.e., it reflects the computed e_{ij} value

EM ALGORITHM: M STEP

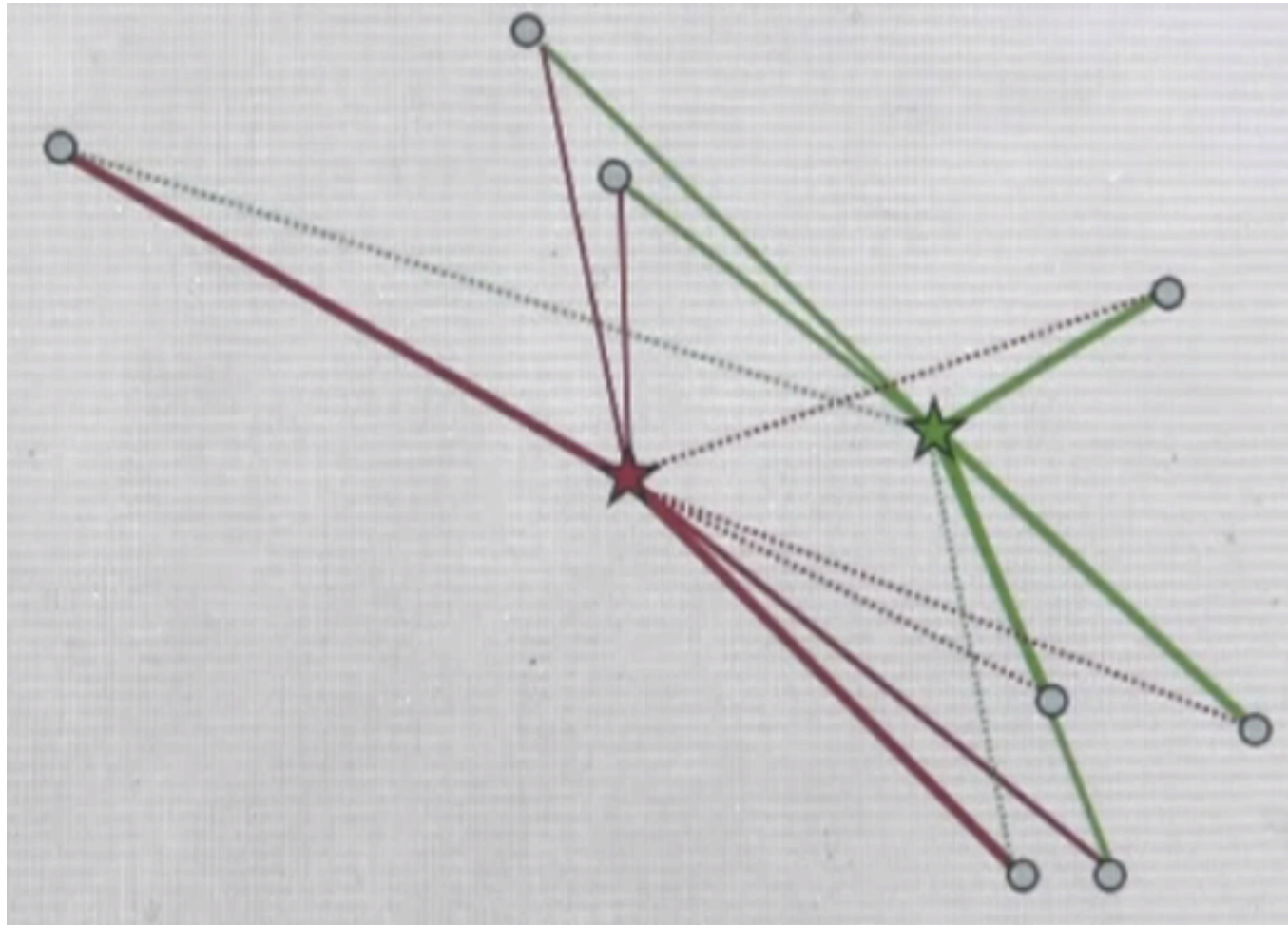
In this step, values of all the model parameters are re-computed

$$p_i = \sum_j e_{ij} / n$$

$$\mu_i = (\sum_j e_{ij} * x_j) / \sum_j e_{ij}$$

$$\sigma_i^2 = (\sum_j e_{ij} * (x_j - \mu_i)^2) / \sum_j e_{ij}$$

EM ALGORITHM: M STEP



Cluster centroids change their position based on the newly computed values of the model parameters

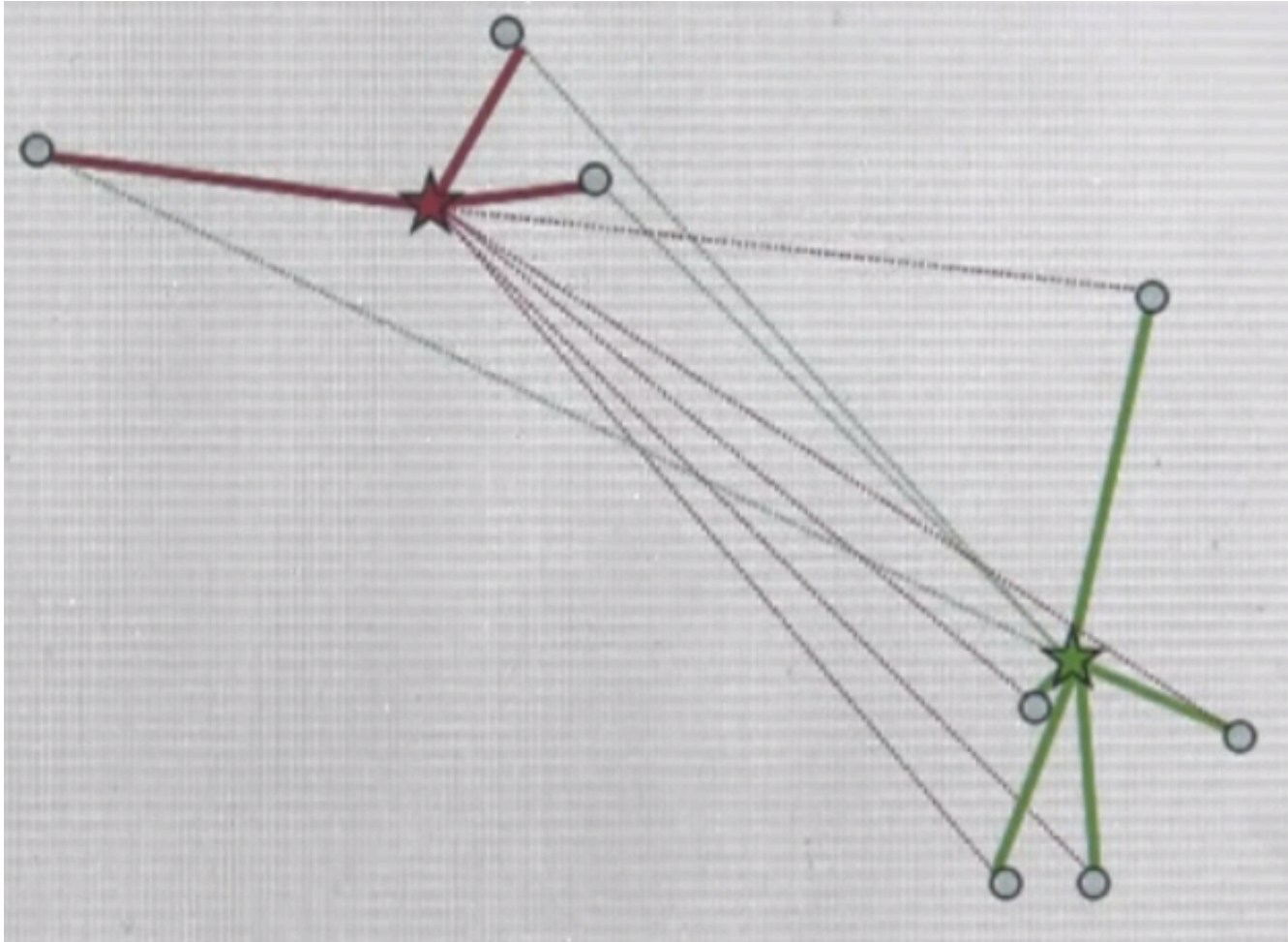
EM ALGORITHM: CONVERGENCE

The two steps of the EM algorithm should be repeated until the increase in the overall log-likelihood of the model becomes negligible:

$$\log P(x) = \log \sum_i (p_i * P(x|C_i))$$

Typically, the log-likelihood will increase very sharply over the first few iterations, and then converge rather quickly to a point that is virtually stationary

EM ALGORITHM: CONVERGENCE



The state of convergence of the model parameters

EM ALGORITHM

EM algorithm is guaranteed to converge to a maximum of the log-likelihood function

However, this is a local maximum that may not necessarily be the same as the global maximum

For a better chance of obtaining the global maximum, the whole procedure should be repeated several times, with different initial guesses for the parameter values

At the end, we should choose the configuration that produces the largest overall log-likelihood

EM ALGORITHM

We've considered the simplest *mixture* model; but EM can be equally well applied to more complex *mixture* models

- Instances can be described with more than one numeric attribute as long as independence between attributes is assumed
 - the probabilities for each attribute are multiplied together to obtain the joint probability for the instance, just as in the Naive Bayes method
- Attributes can be nominal, as well
 - In that case, Normal distribution has to be abandoned
 - nominal attribute with v possible values is characterized by v numbers representing the probability of each value

EM: AN EXAMPLE IN WEKA

ACKNOWLEDGEMENT AND RECOMMENDATION

Stanford Machine Learning

Andrew Ng

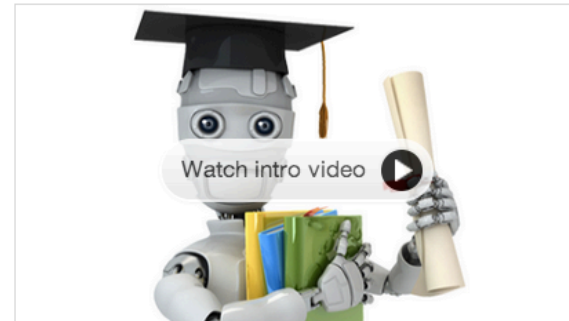
Learn about the most effective machine learning techniques, and gain practice implementing them and getting them to work for yourself.

Workload: 5-7 hours/week

Taught In: English

Subtitles Available In: English

Preview



Sessions:

Oct 14th 2013 (10 weeks long)

Sign Up

Apr 22nd 2013 (10 weeks long)

Sign Up

3,484

12k

13k

Tweet

+1

Like

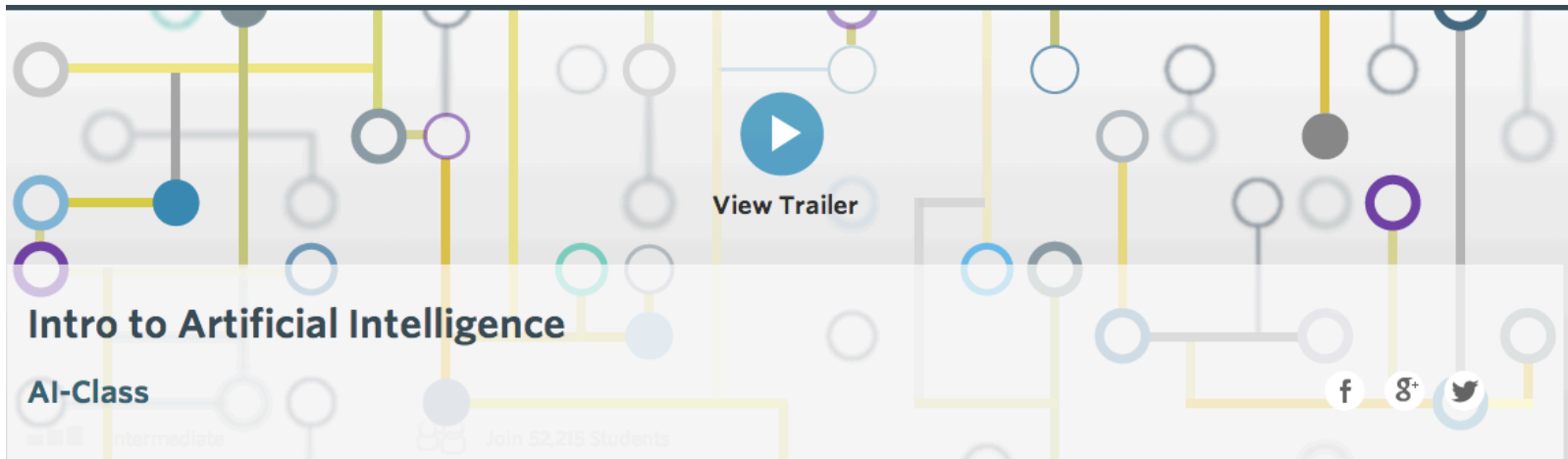
Coursera:

<https://www.coursera.org/course/ml>

Stanford YouTube channel:

http://www.youtube.com/view_play_list?p=A89DCFA6ADACE599

ACKNOWLEDGEMENT AND RECOMMENDATION



Intro to Artificial Intelligence

AI-Class

Intermediate

Join 52,215 Students

View Trailer

f g+ t

Intermediate

Join 52,215 Students

Course Instructors



Peter Norvig

INSTRUCTOR

Peter Norvig is Director of Research at Google Inc. He is also a Fellow of the American Association for Artificial Intelligence and the Association for Computing Machinery. Norvig is co-author of the popular textbook *Artificial Intelligence: A Modern Approach*. Prior to joining Google he was the head of the Computation Sciences Division at NASA Ames Research Center.



Sebastian Thrun

INSTRUCTOR

Sebastian Thrun is a Research Professor of Computer Science at Stanford University, a Google Fellow, a member of the National Academy of Engineering and the German Academy of Sciences. Thrun is best known for his research in robotics and machine learning, specifically his work with self-driving cars.

URL: <https://www.udacity.com/course/cs271>

(Anonymous) questionnaire for your
critiques, comments, suggestions:

<http://goo.gl/cqdp3l>