Clustering

CSE 4309 – Machine Learning

Vassilis Athitsos
Computer Science and Engineering Department
University of Texas at Arlington

Supervised vs. Unsupervised Learning

- In **supervised learning**, the training data is a set of pairs (x_n, y_n), where x_n is an example input, and y_n is the target output for that input.
  - The goal is to learn a general function $\mathrm{H}(x)$ that maps inputs to outputs.
  - The majority of the methods we have covered this semester fall under the category of supervised learning.
- In unsupervised learning, the training data is a set of values x_n.
  - There is **no associated target value** for any x_n.
  - Because of that, data values x_n are called **unlabeled data**.

Supervised vs. Unsupervised Learning

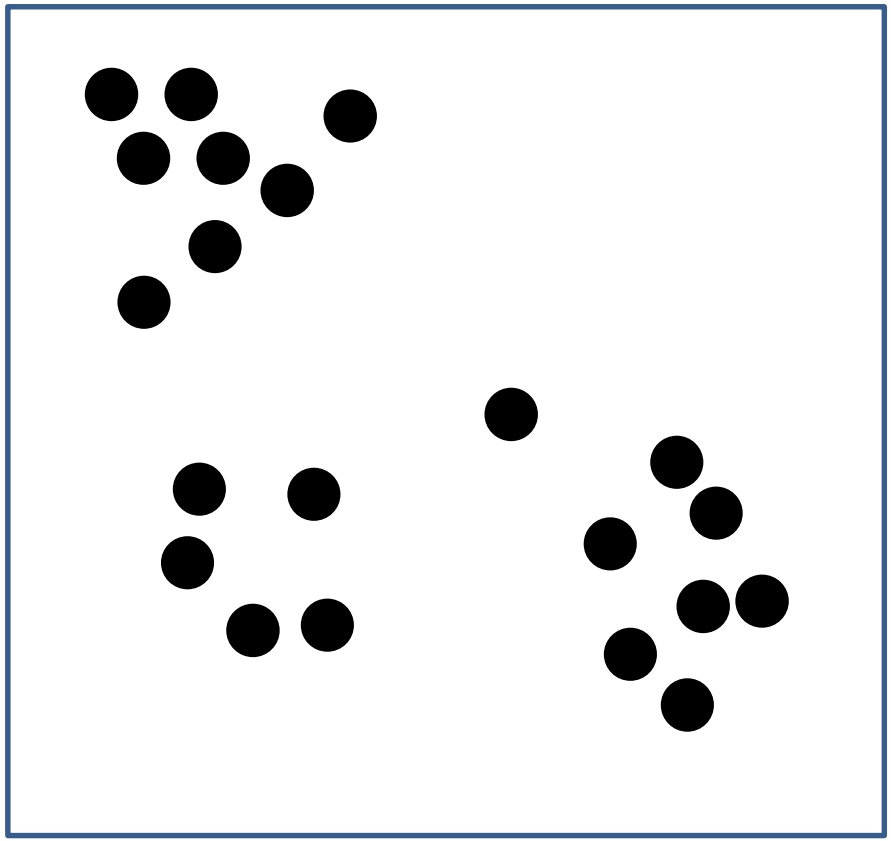
- The goal in supervised learning is **regression** or **classification**.
- The goal in unsupervised learning is to discover **hidden structure** in the data.

Supervised vs. Unsupervised Learning

- You may have heard of some methods that do different types of unsupervised learning.
  - PCA (principal component analysis): it learns how to represent high-dimensional vectors using low-dimensional vectors.
  - SVD (singular value decomposition): it learns how to represent matrix data as dot products of low-dimensional vectors. An example of such matrix data is movie ratings by users (one row per user, one column per movie, most values left unspecified), where we can use SVD to build a model that predicts how much a specific user will like a specific movie.
- We will cover some of these methods towards the end of the semester, as optional, non-graded material.

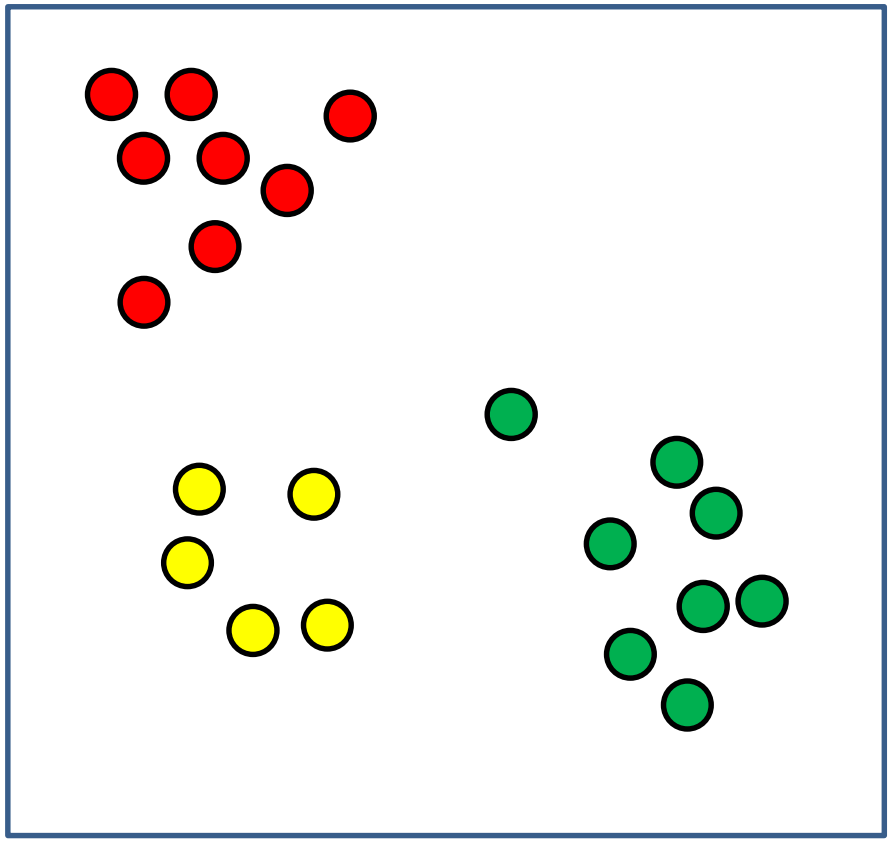
Clustering

- The goal of clustering is to separate the data into coherent subgroups.
  - Data within a cluster should be more similar to each other than to data in other clusters.
- For example:
  - Can you identify clusters in this dataset?
  - How many? Which ones?



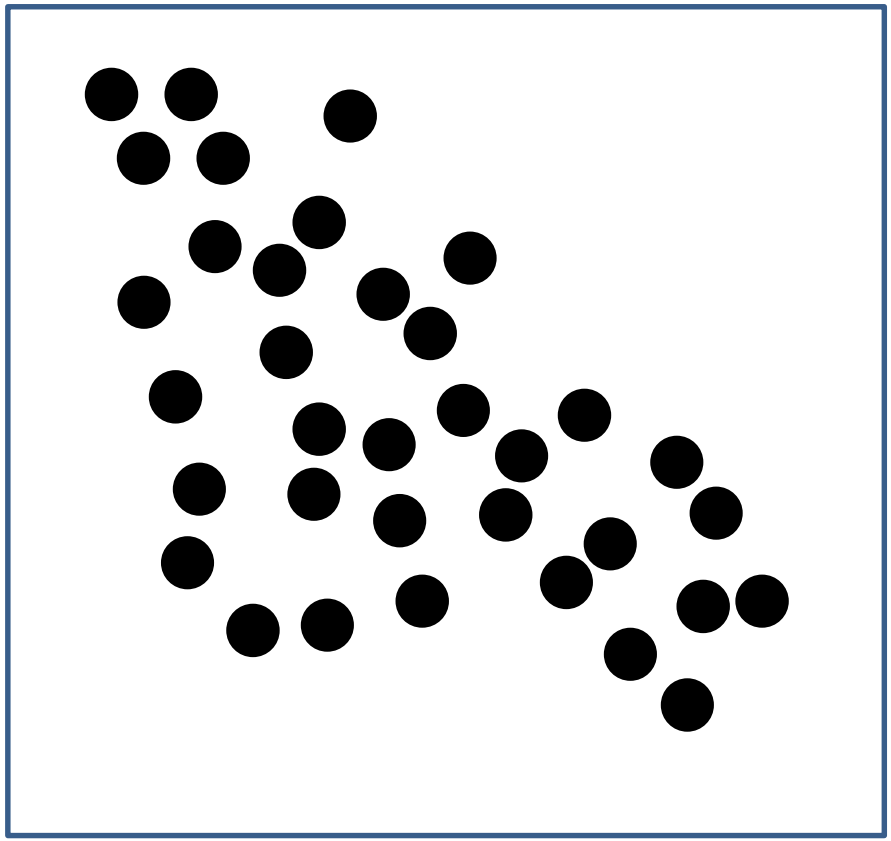
Clustering

- The goal of clustering is to separate the data into coherent subgroups.
  - Data within a cluster should be more similar to each other than to data in other clusters.
- For example:
  - Can you identify clusters in this dataset?
  - Many people would identify three clusters.



Clustering

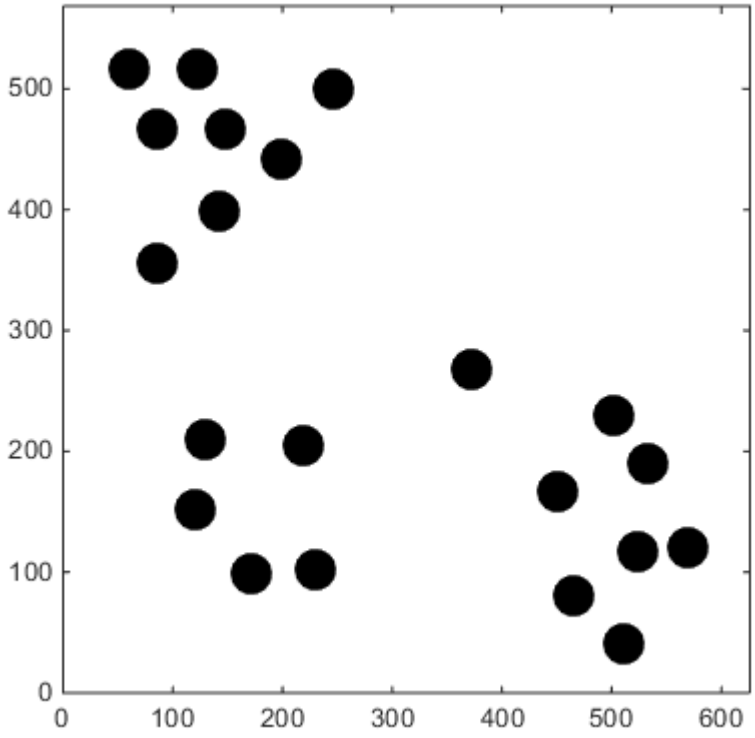
- Some times the clusters may not be as obvious.
- Identifying clusters can be even harder with high-dimensional data.



Applications of Clustering

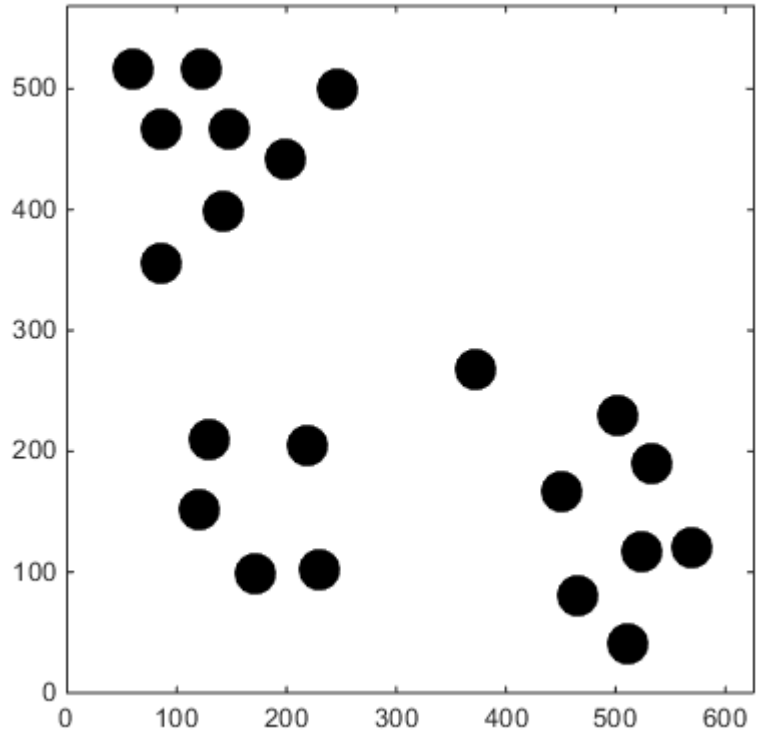
- Finding subgroups of similar items is useful in many fields:
  - In biology, clustering is used to identify relationships between organisms, and to uncover possible evolutionary links.
  - In marketing, clustering is used to identify segments of the population that would be specific targets for specific products.
  - For anomaly detection, anomalous data can be identified as data that cannot be assigned to any of the "normal" clusters.
  - In search engines and recommender systems, clustering can be used to group similar items together.

K-Means Clustering



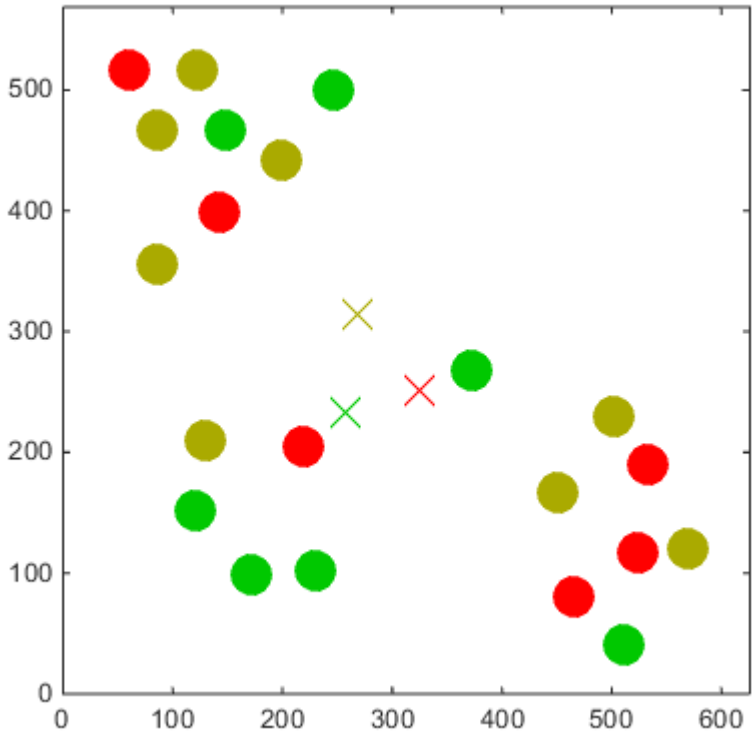
- **<u>K-means clustering</u>** is a simple and widely used clustering method.
- First, clusters are initialized by assigning each object randomly to a cluster.
- Then, the algorithm alternates between:
  - Re-assigning objects to clusters based on distances from each object to the mean of each current cluster.
  - Re-computing the means of the clusters.
- The number of clusters must be chosen manually.

K-Means Clustering: Initialization



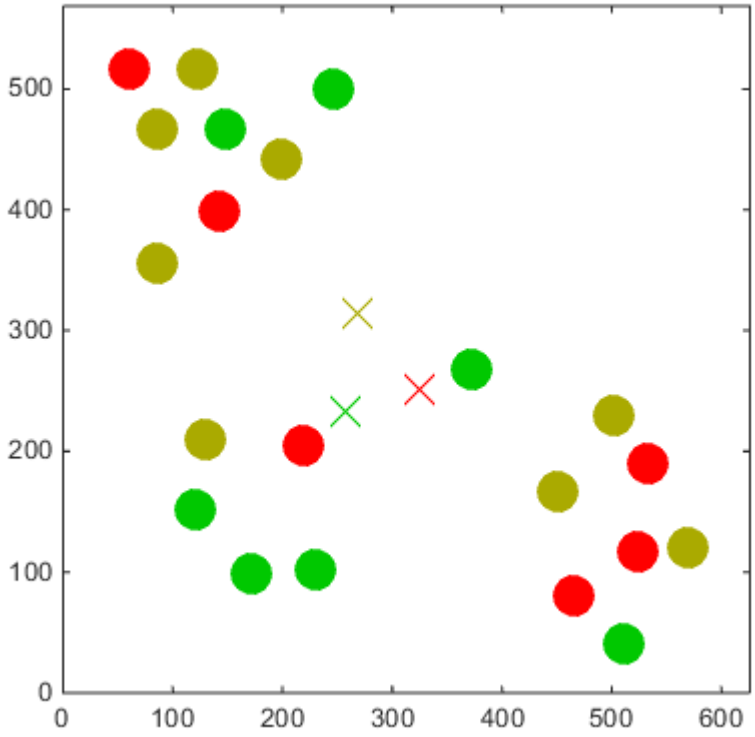
- To start the iterative process, we first need to provide some initial values.
- We manually pick the number of clusters.
  - In the example shown, we pick 3 as the number of clusters.
- We randomly assign objects to clusters.

K-Means Clustering: Initialization



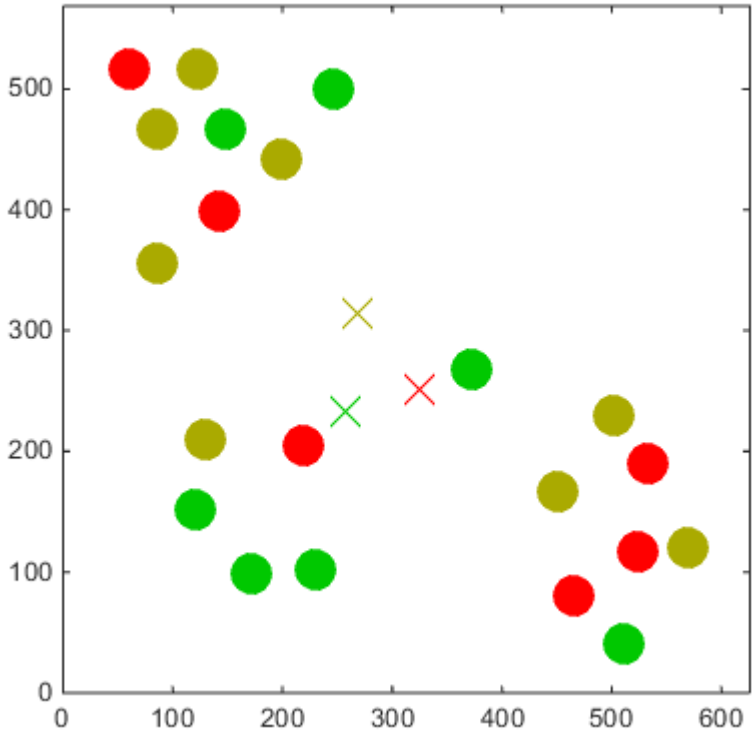
- To start the iterative process, we first need to provide some initial values.
- We manually pick the number of clusters.
  - In the example shown, we pick 3 as the number of clusters.
- We randomly assign objects to clusters.
- In the example figure, cluster membership is indicated with color.
  - There is a red cluster, a green cluster, and a brown cluster.

K-Means Clustering: Initialization



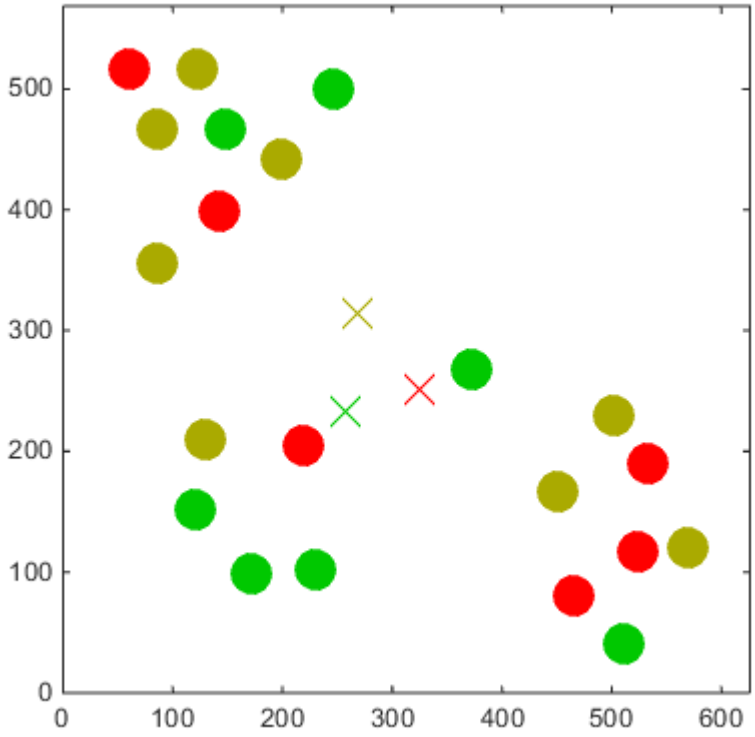
- To start the iterative process, we first need to provide some initial values.
- We manually pick the number of clusters.
  - In the example shown, we pick 3 as the number of clusters.
- We randomly assign objects to clusters.
- The means of these random clusters are shown with × marks.

K-Means Clustering: Main Loop



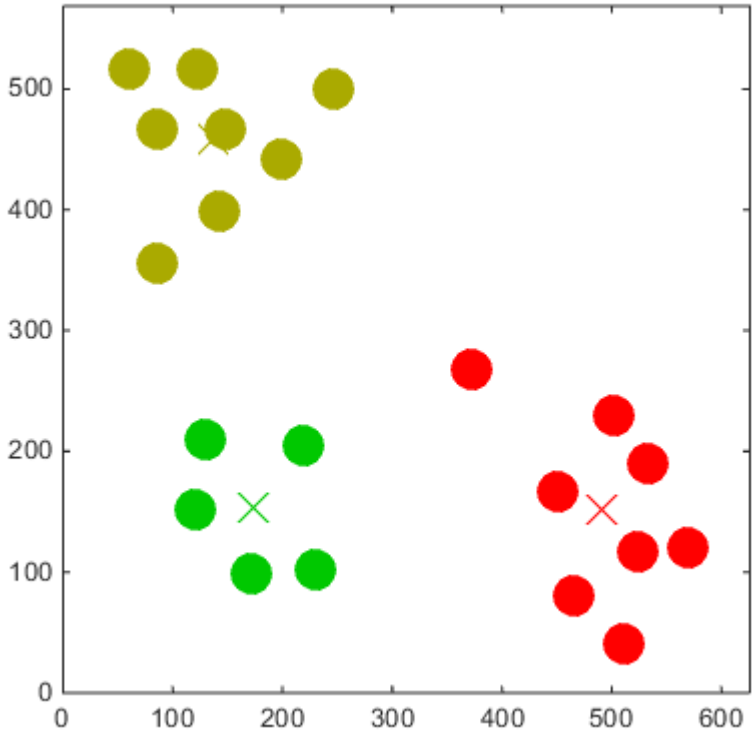
- The main loop alternates between:
  - Computing new assignments of objects to clusters, based on distances from each object to each cluster mean.
  - Computing new means for the clusters, using the current cluster assignments.
- At this point, we have provided some random initial assignments of points to clusters.
- What is the next step?

K-Means Clustering: Main Loop



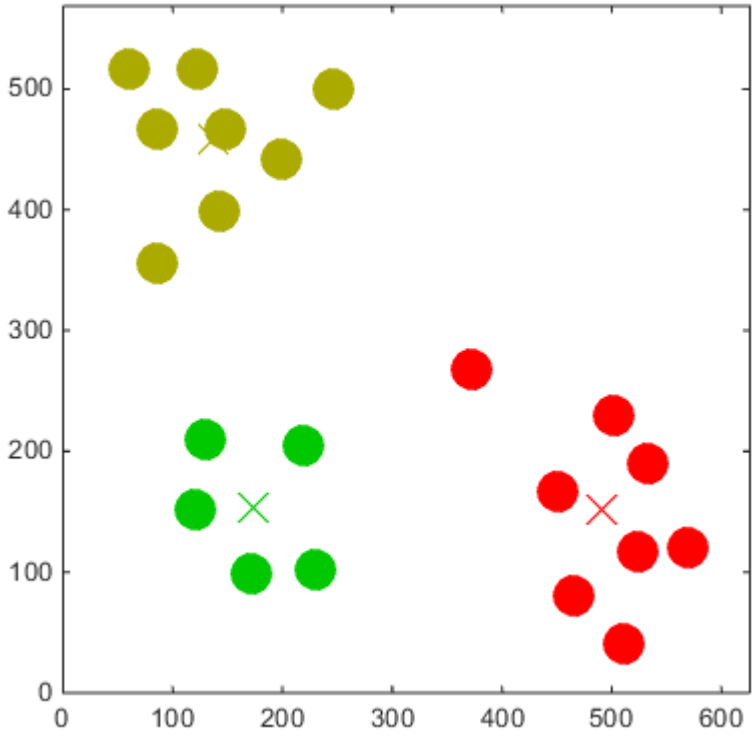
- The main loop alternates between:
  - **Computing new assignments of objects to clusters, based on distances from each object to each cluster mean.**
  - Computing new means for the clusters, using the current cluster assignments.
- The next step is to compute new assignments of objects to clusters.

K-Means Clustering: Main Loop



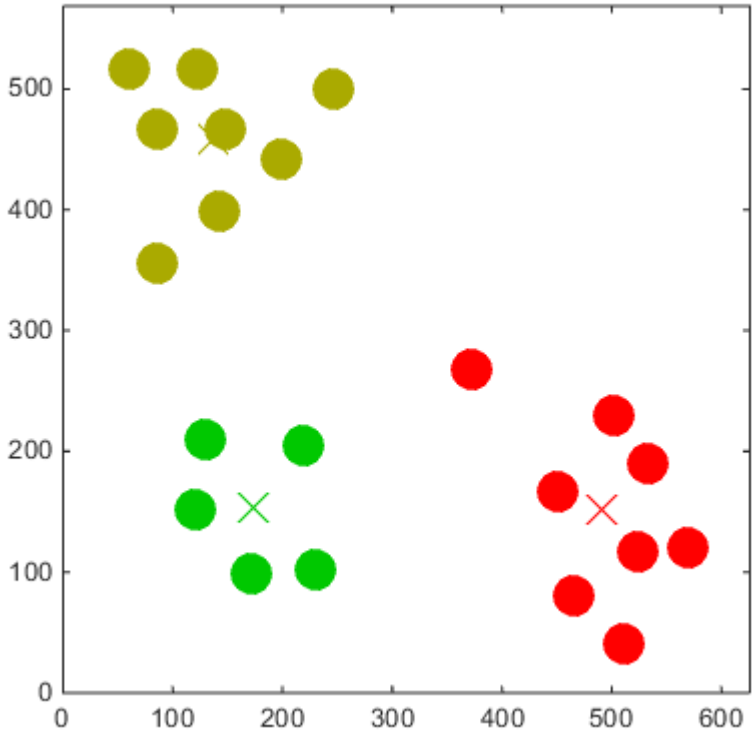
- The main loop alternates between:
  - **Computing new assignments of objects to clusters, based on distances from each object to each cluster mean.**
  - Computing new means for the clusters, using the current cluster assignments.
- The next step is to compute new assignments of objects to clusters.
  - The figure shows the new assignments.

K-Means Clustering: Main Loop



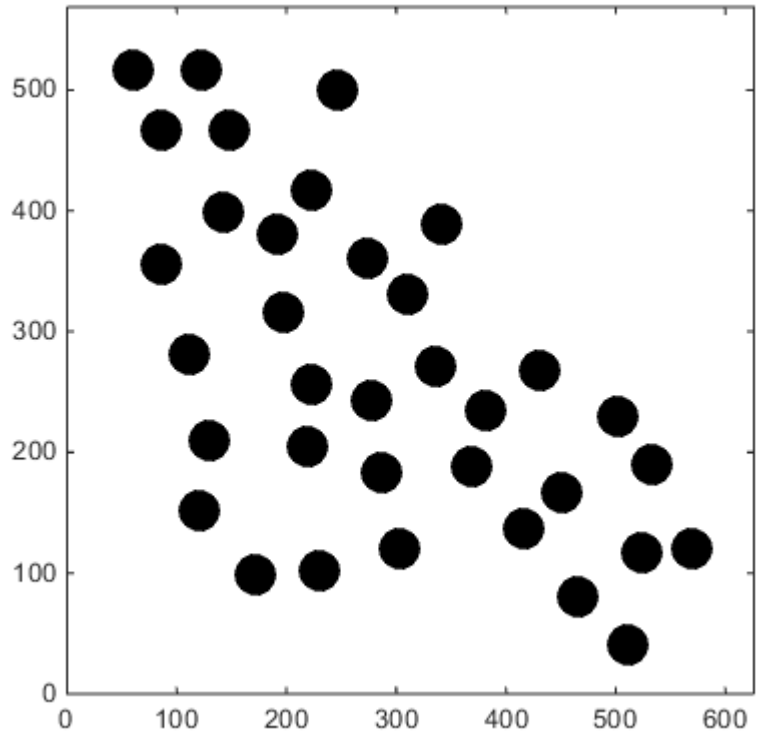
- The main loop alternates between:
  - Computing new assignments of objects to clusters, based on distances from each object to each cluster mean.
  - **Computing new means for the clusters, using the current cluster assignments.**
- Next, we move on to compute the new means.
  - Again, the new means are shown with × marks.

K-Means Clustering: Main Loop



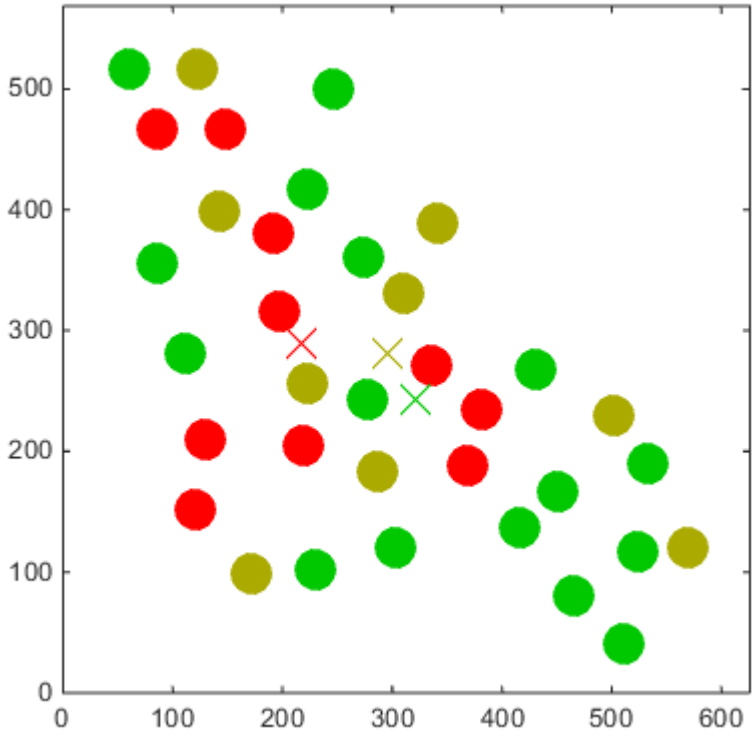
- The main loop alternates between:
  - **Computing new assignments of objects to clusters, based on distances from each object to each cluster mean.**
  - Computing new means for the clusters, using the current cluster assignments.
- Next, we compute again new assignments.
  - We end up with the same assignments as before.
  - When this happens, we can terminate the algorithm.

K-Means Clustering: Another Example



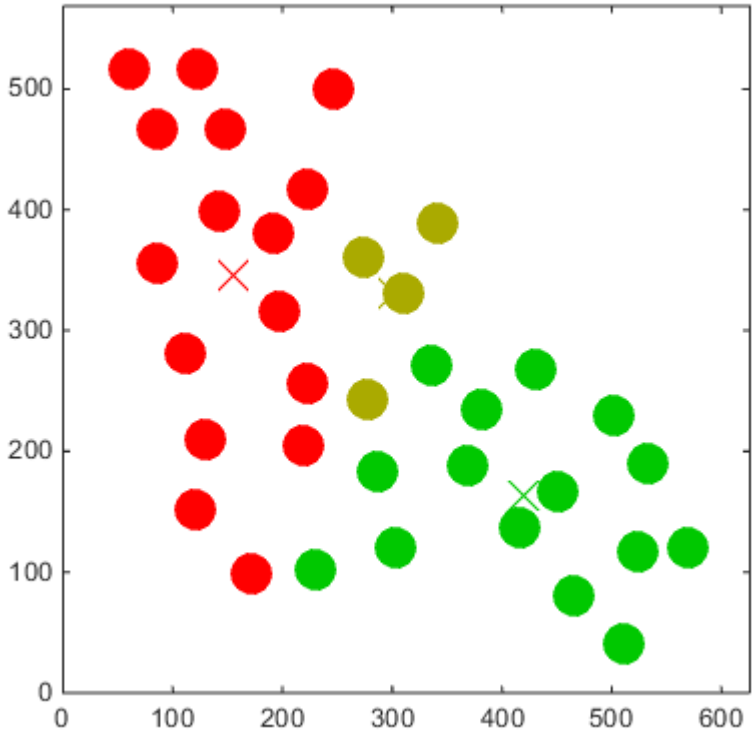
- Here is a more difficult example, where there are no obvious clusters.
- Again, we specify manually that we want to find 3 clusters.
- First step: ???

K-Means Clustering: Another Example



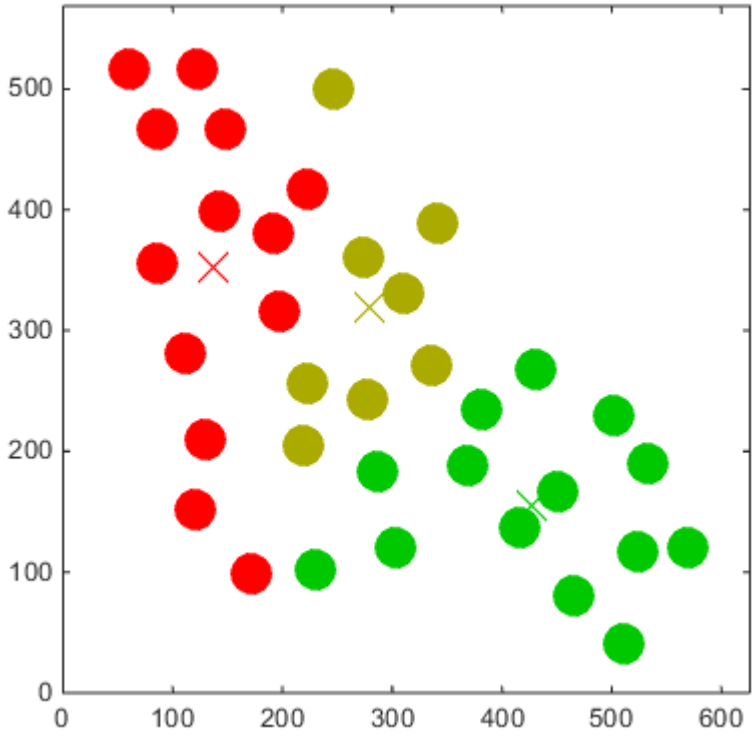
- Here is a more difficult example, where there are no obvious clusters.
- Again, we specify manually that we want to find 3 clusters.
- First step: randomly assign points to the three clusters.
  - We see the random assignments and the resulting means for the three clusters.

K-Means Clustering: Another Example



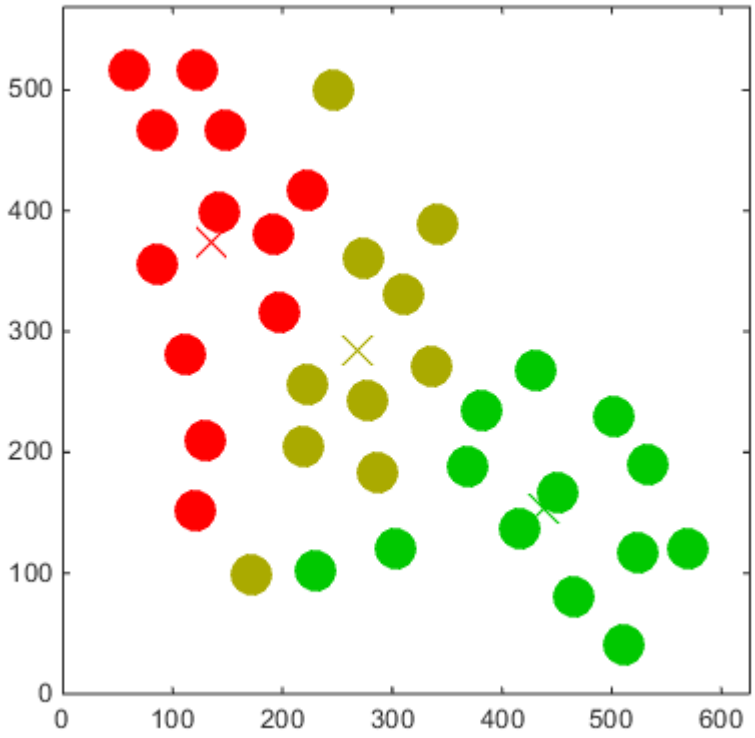
- Here is a more difficult example, where there are no obvious clusters.
- Again, we specify manually that we want to find 3 clusters.
- Main loop, iteration 1: recompute cluster assignments.

K-Means Clustering: Another Example



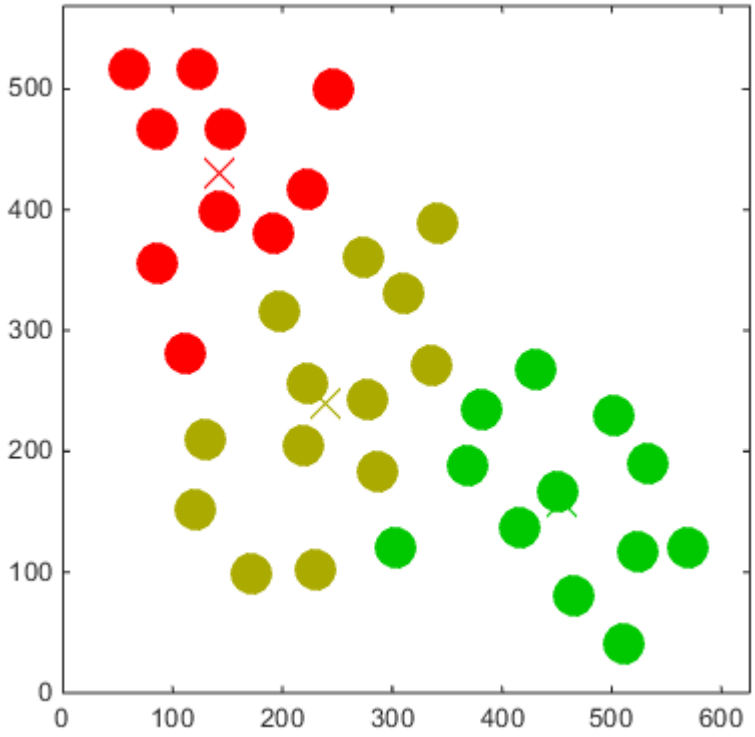
- Here is a more difficult example, where there are no obvious clusters.
- Again, we specify manually that we want to find 3 clusters.
- Main loop, iteration 2: recompute cluster assignments.

K-Means Clustering: Another Example



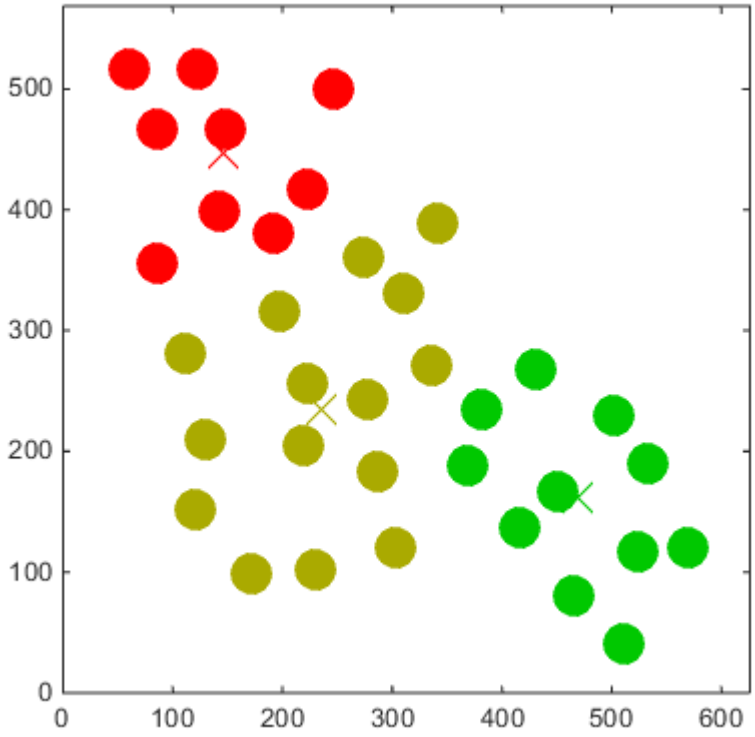
- Here is a more difficult example, where there are no obvious clusters.
- Again, we specify manually that we want to find 3 clusters.
- Main loop, iteration 3: recompute cluster assignments.

K-Means Clustering: Another Example



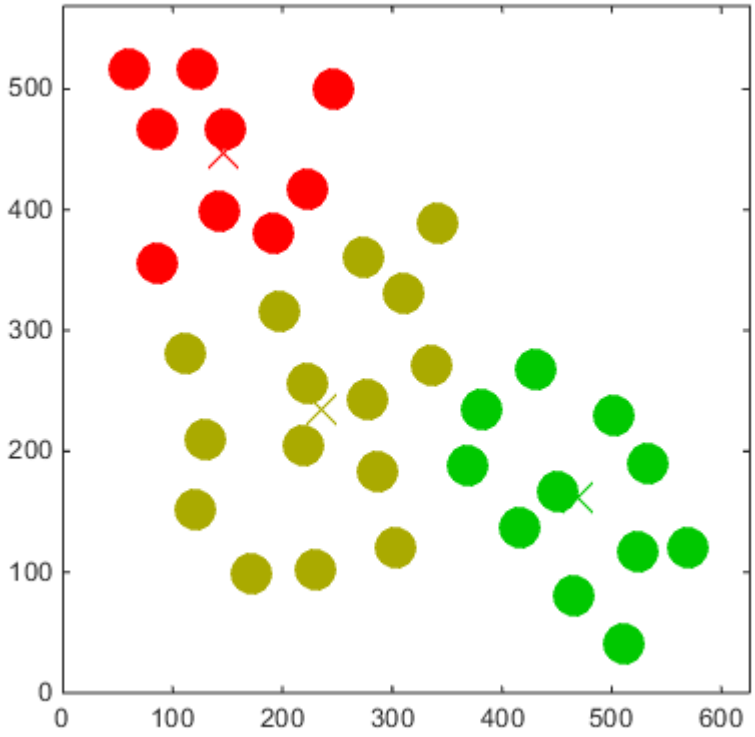
- Here is a more difficult example, where there are no obvious clusters.
- Again, we specify manually that we want to find 3 clusters.
- Main loop, iteration 4: recompute cluster assignments.

K-Means Clustering: Another Example



- Here is a more difficult example, where there are no obvious clusters.
- Again, we specify manually that we want to find 3 clusters.
- Main loop, iteration 5: recompute cluster assignments.

K-Means Clustering: Another Example



- Here is a more difficult example, where there are no obvious clusters.
- Again, we specify manually that we want to find 3 clusters.
- Main loop, iteration 6: recompute cluster assignments.
  - The cluster assignments do not change, compared to iteration 5, so we can stop.

K-Means Clustering - Theory

- Let $\boldsymbol{X} = \{x_1, x_2, \ldots, x_N\}$ be our set of points.
  - We assume that $\boldsymbol{X}$ is a subset of D-dimensional vector space $\mathbb{R}^D$.
  - Thus, each x_n is a D-dimensional vector.
- Let $\boldsymbol{S}_1, \boldsymbol{S}_2, \ldots, \boldsymbol{S}_K$ be some set of clusters for $\boldsymbol{X}$.
  - The union of all clusters should be $\boldsymbol{X}$.

$$\bigcup_{k=1}^{K} \boldsymbol{S}_k = \boldsymbol{X}$$

  - The intersection of any two clusters should be empty.

$$\forall j, k: \boldsymbol{S}_k \cap \boldsymbol{S}_k = \emptyset$$

K-Means Clustering - Theory

- Let $\boldsymbol{X} = \{x_1, x_2, \ldots, x_N\}$ be our set of points.
- Let $\boldsymbol{S}_1, \boldsymbol{S}_2, \ldots, \boldsymbol{S}_K$ be some set of clusters for $\boldsymbol{X}$.
- Let μ_k be the mean of all elements of $\boldsymbol{S}_k$.
- The error measure for such a set of clusters is defined as:

Euclidean distance between x_n and μ_k

$$E(\boldsymbol{S}_1, \boldsymbol{S}_2, \ldots, \boldsymbol{S}_K) = \sum_{k=1}^{K} \sum_{x_n \in \boldsymbol{S}_k} \|x_n - \mu_k\|$$

- The optimal set of clusters is the one that minimizes this error measure.

K-Means Clustering - Theory

- The error measure for a set of clusters $\boldsymbol{S}_1$, $\boldsymbol{S}_2, \ldots, \boldsymbol{S}_K$ is defined as:

$$E(\boldsymbol{S}_1, \boldsymbol{S}_2, \ldots, \boldsymbol{S}_K) = \sum_{k=1}^{K} \sum_{x_n \in \boldsymbol{S}_k} \|x_n - \mu_k\|$$

- The optimal set of clusters is the one that minimizes this error measure.
- Finding the optimal set of clusters is NP-hard.
  - Solving this problem takes time $O(N^{KD+1})$, where:
    - N is the number of objects.
    - D is the number of dimensions.
    - K is the number of clusters.

K-Means Clustering - Theory

- The error measure for a set of clusters $\boldsymbol{S}_1$, $\boldsymbol{S}_2, \ldots, \boldsymbol{S}_K$ is defined as:

$$E(\boldsymbol{S}_1, \boldsymbol{S}_2, \ldots, \boldsymbol{S}_K) = \sum_{k=1}^{K} \sum_{x_n \in \boldsymbol{S}_k} \|x_n - \mu_k\|$$

- The iterative algorithm we described earlier decreases the error E at each iteration.
- Thus, this iterative algorithm converges.
  - However, it only converges to a local optimum.

K-Medoid Clustering

- The main loop in k-means clustering is:
  - Computing new assignments of objects to clusters, based on distances from each object to each cluster **mean**.
  - Computing new **means** for the clusters, using the current cluster assignments.
- **K-medoid clustering** is a variation of k-means, where we use **medoids** instead of means.
- The main loop in k-medoid clustering is:
  - Computing new assignments of objects to clusters, based on distances from each object to each cluster **medoid**.
  - Computing new **medoids** for the clusters, using the current cluster assignments.

K-Medoid Clustering

- To complete the definition of k-medoid clustering, we need to define what a medoid of a set is.
- Let $\boldsymbol{S}$ be a set, and let F be a distance measure, that evaluates distances between any two elements of $\boldsymbol{S}$.
- Then, the medoid $m_{\boldsymbol{S}}$ of $\boldsymbol{S}$ is defined as:

$$m_{\boldsymbol{S}} = \underset{s \in \boldsymbol{S}}{\operatorname{argmin}} \sum_{x \in \boldsymbol{S}} F(s, x)$$

- In words, $m_{\boldsymbol{S}}$ is the object in $\boldsymbol{S}$ with the smallest sum of distances to all other objects in $\boldsymbol{S}$.

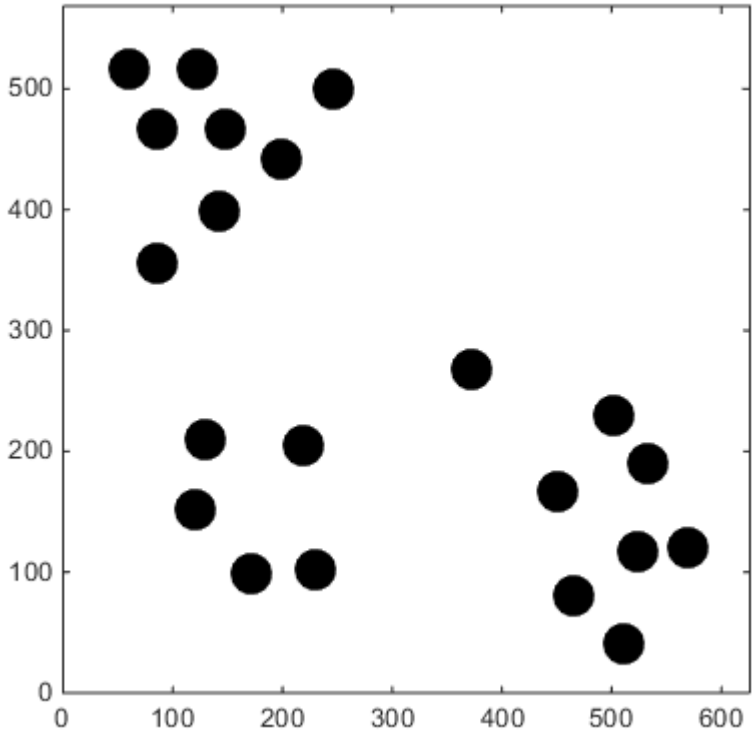
K-Medoid vs. K-means

- K-medoid clustering can be applied on non-vector data with non-Euclidean distance measures.
- For example:
  - K-medoid can be used to cluster a set of time series objects, using DTW as the distance measure.
  - K-medoid can be used to cluster a set of strings, using the edit distance as the distance measure.
- K-means cannot be used in such cases.
  - Means may not make sense for non-vector data.
  - For example, it does not make sense to talk about the mean of a set of strings. However, we can define (and find) the medoid of a set of strings, under the edit distance.

K-Medoid vs. K-means

- K-medoid clustering can be applied on non-vector data with non-Euclidean distance measures.
- K-medoid clustering is more robust to outliers.
  - A single outlier can dominate the mean of a cluster, but it typically has only small influence on the medoid.
- The K-means algorithm can be proven to converge to a local optimum.
  - The k-medoid algorithm may not converge.

EM for Clustering



- Another clustering method is based on an algorithm called Expectation-Maximization (EM).
- The EM algorithm models the data as being generated by a **mixture of Gaussians**.
  - I.e., by multiple Gaussians.
- The EM algorithm estimates the parameters (mean and covariance matrix) of each Gaussian.
- Each Gaussian defines a cluster.
- Key difference from K-means: here, membership to a cluster is typically partial.
  - The algorithm assigns a "membership" weight between each data point and each Gaussian.

Review of Gaussians

- Review: a 1D normal distribution is defined as:

$$N(x) = \frac{1}{\sigma\sqrt{2\pi}}\, e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

- To define a Gaussian, we need to specify just two parameters:
  - μ, which is the mean (average) of the distribution.
  - σ, which is the standard deviation of the distribution.
  - Note: σ^2 is called the **variance** of the distribution.

Estimating a Gaussian

- In one dimension, a Gaussian is defined like this:

$$N(x) = \frac{1}{\sigma\sqrt{2\pi}}\, e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

- Given a set of n real numbers x_1, …, x_n, we can easily find the best-fitting Gaussian for that data.
- The mean μ is simply the average of those numbers:

$$\mu = \frac{1}{n}\sum_{1}^{n} x_i$$

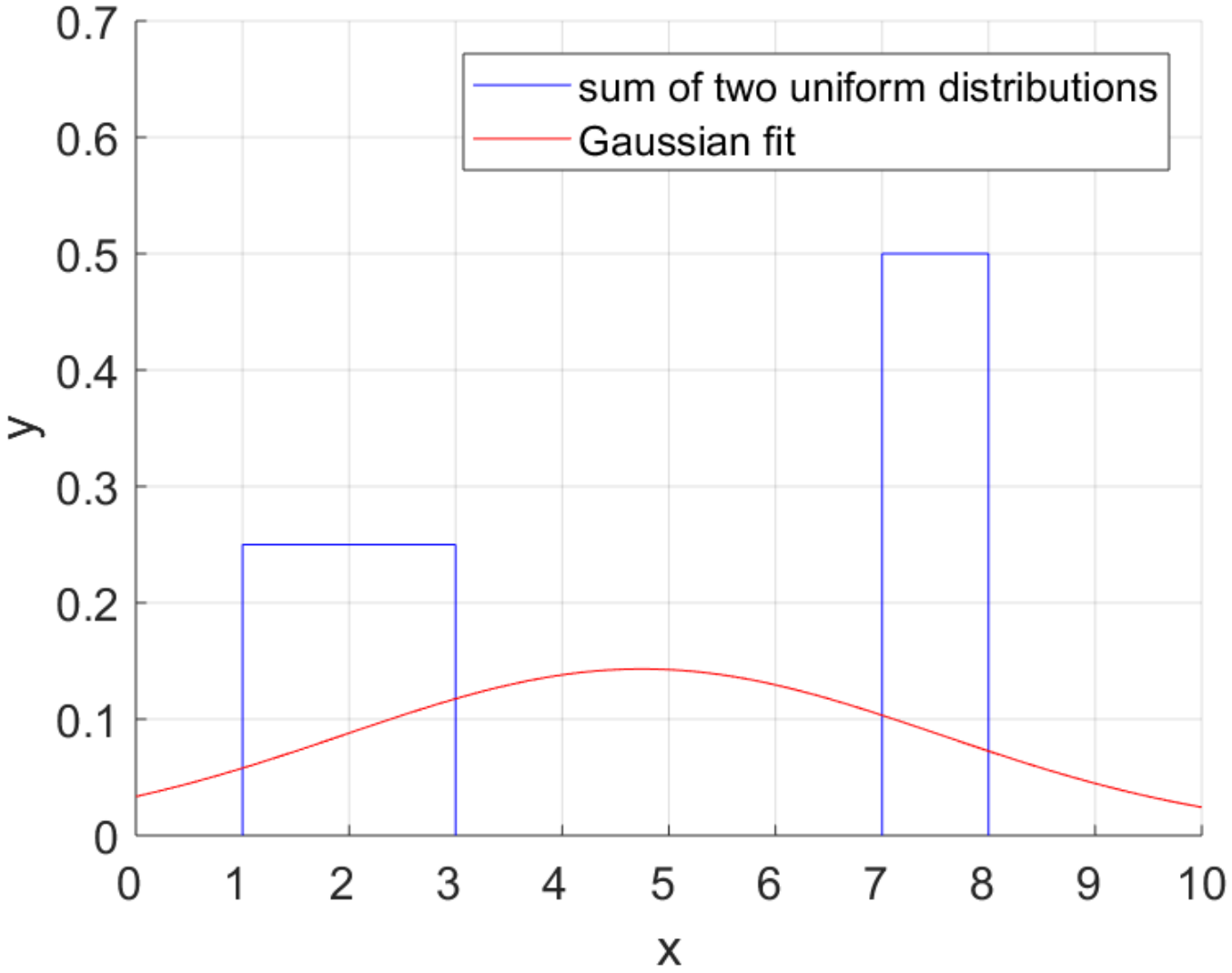
- The standard deviation σ is computed as:

$$\sigma = \sqrt{\frac{1}{n-1}\sum_{1}^{n}(x_i - \mu)^2}$$

Estimating a Gaussian

- Fitting a Gaussian to data does not guarantee that the resulting Gaussian will be an accurate distribution for the data.
- The data may have a distribution that is very different from a Gaussian.

Example of Fitting a Gaussian

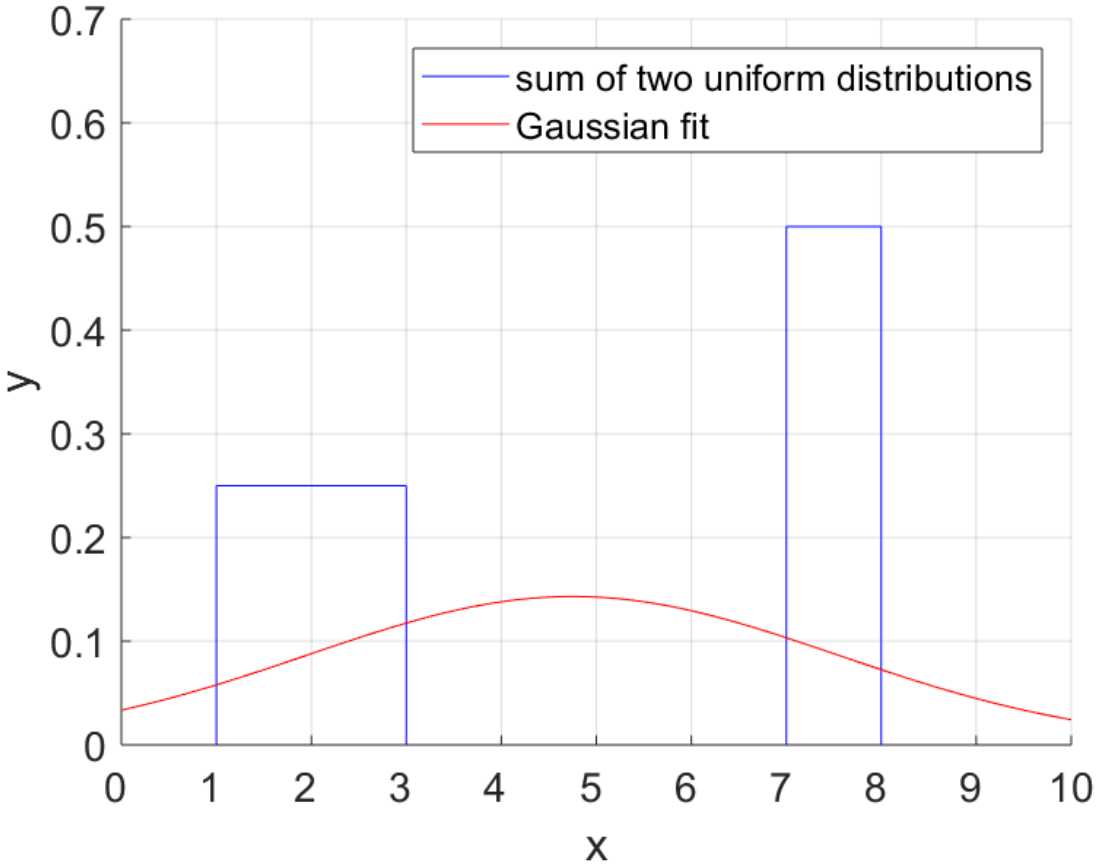


The blue curve is a density function F such that:

- $F(x) = 0.25$ for $1 \leq x \leq 3$.
- $F(x) = 0.5$ for $7 \leq x \leq 8$.

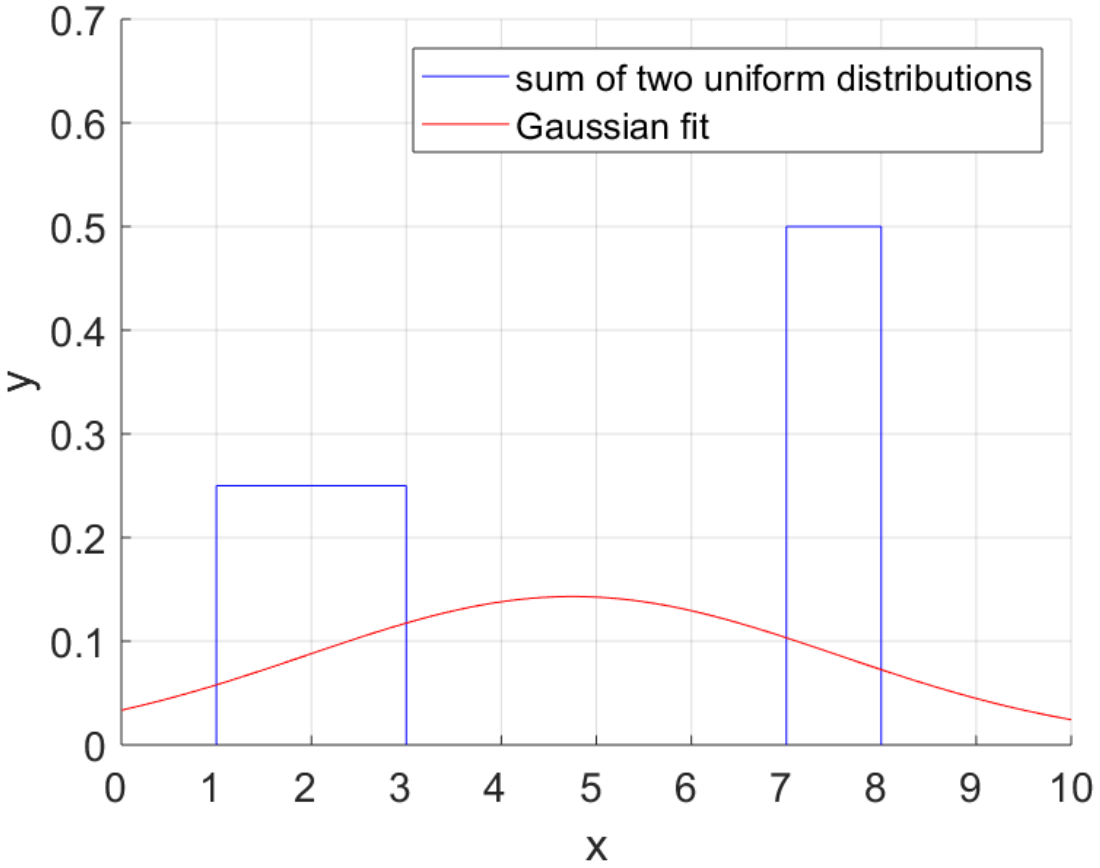
The red curve is the Gaussian fit G to data generated using F.

Mixtures of Gaussians



- This figure shows our previous example, where we fitted a Gaussian into some data, and the fit was poor.
- Overall, Gaussians have attractive properties:
  - They require learning only two numbers (μ and σ), and thus require few training data to estimate those numbers.
- However, for some data, Gaussians are just not good fits.

Mixtures of Gaussians



- **<u>Mixtures of Gaussians</u>** are oftentimes a better solution.
  - They are defined in the next slide.
- They still require relatively few parameters to estimate, and thus can be learned from relatively small amounts of data.
- They can fit pretty well actual distributions of data.

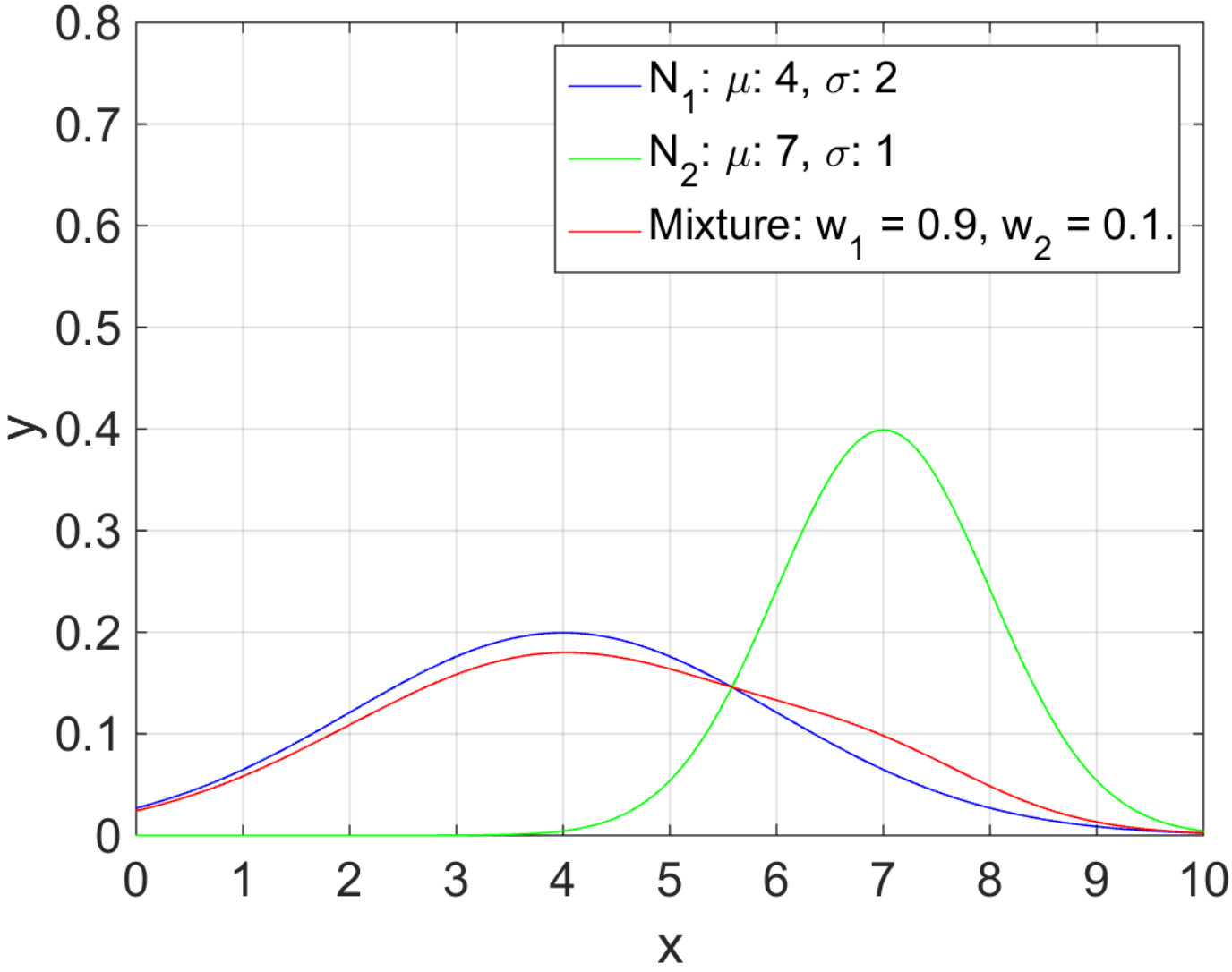
Mixtures of Gaussians

- Suppose we have k Gaussian distributions N_i.
- Each N_i has its own mean μ_i and std σ_i.
- Using these k Gaussians, we can define a **Gaussian mixture** M as follows:

$$M(x) = \sum_{i=1}^{k} w_i N_i(x)$$

- Each w_i is a weight, specifying the relative importance of Gaussian N_i in the mixture.
  - Weights w_i are real numbers between 0 and 1.
  - Weights w_i must sum up to 1, so that the integral of M is 1.

Mixtures of Gaussians – Example



The blue and green curves show two Gaussians.

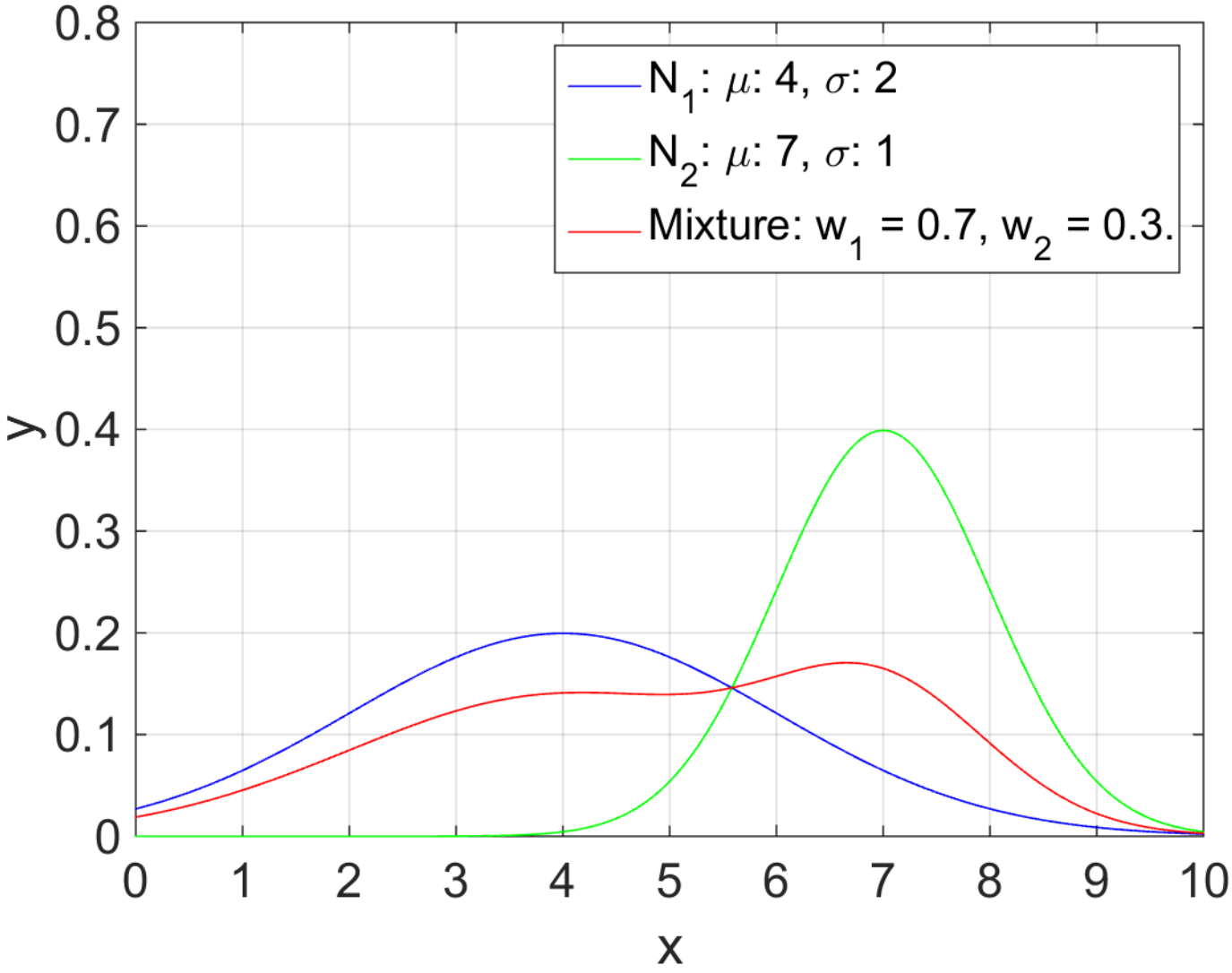
The red curve shows a mixture of those Gaussians.

$w_1 = 0.9$.

$w_2 = 0.1$.

The mixture looks a lot like N_1, but is influenced a little by N_2 as well.

Mixtures of Gaussians – Example



The blue and green curves show two Gaussians.

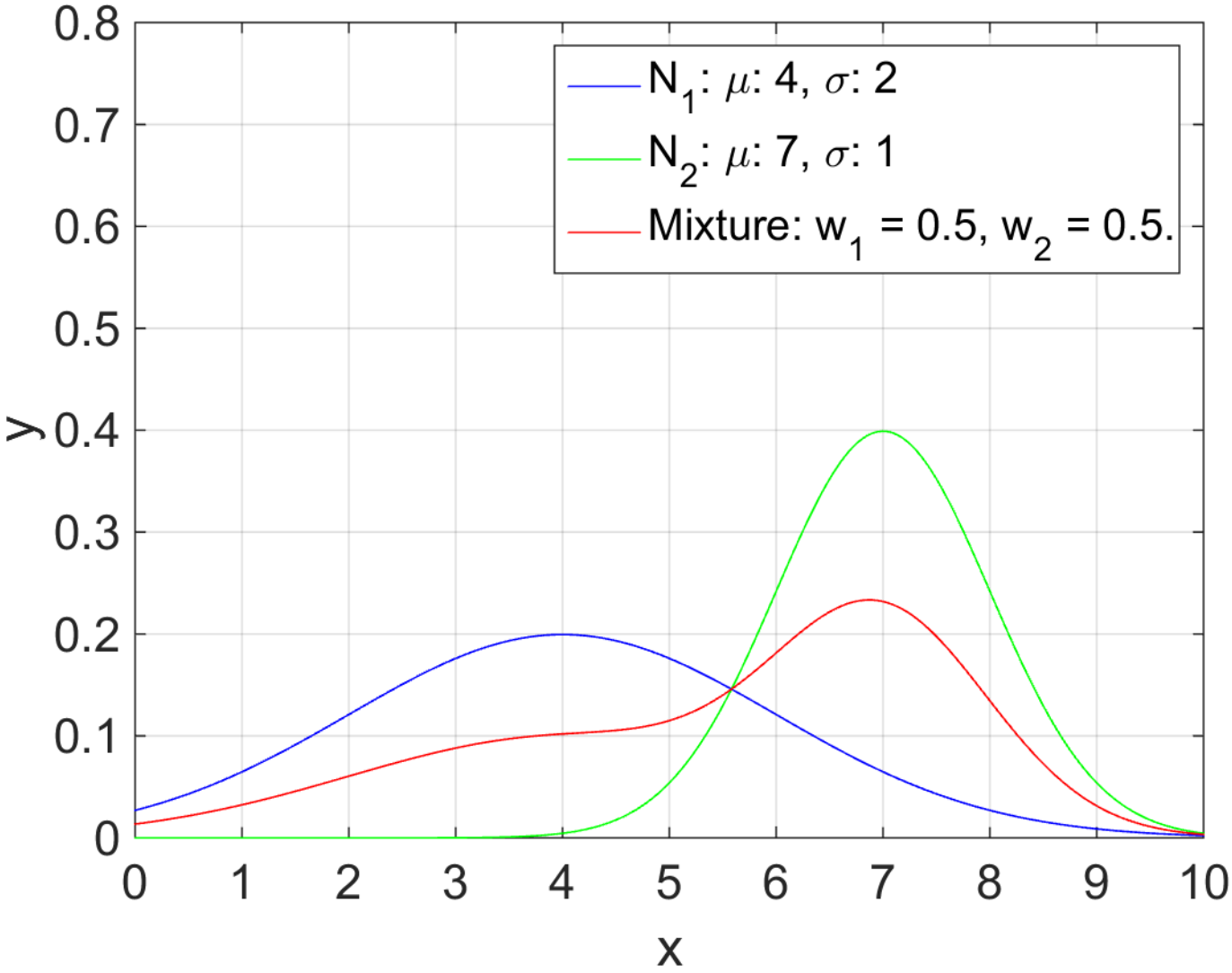
The red curve shows a mixture of those Gaussians.

$w_1 = 0.7$.

$w_2 = 0.3$.

The mixture looks less like N_1 compared to the previous example, and is influenced more by N_2.

Mixtures of Gaussians – Example



The blue and green curves show two Gaussians.

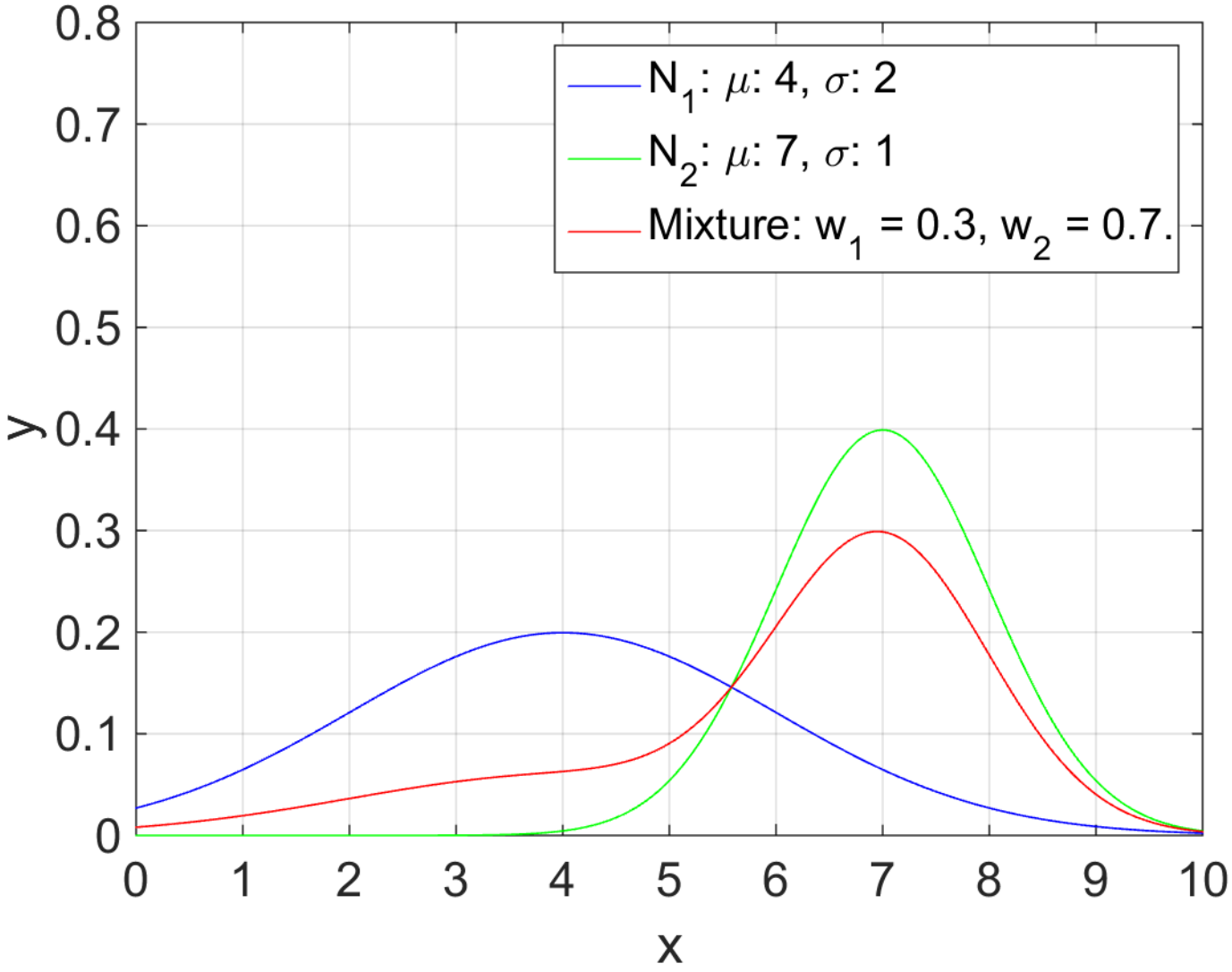
The red curve shows a mixture of those Gaussians.

$w_1 = 0.5$.

$w_2 = 0.5$.

At each point x, the value of the mixture is the average of $N_1(x)$ and $N_2(x)$.

Mixtures of Gaussians – Example



The blue and green curves show two Gaussians.

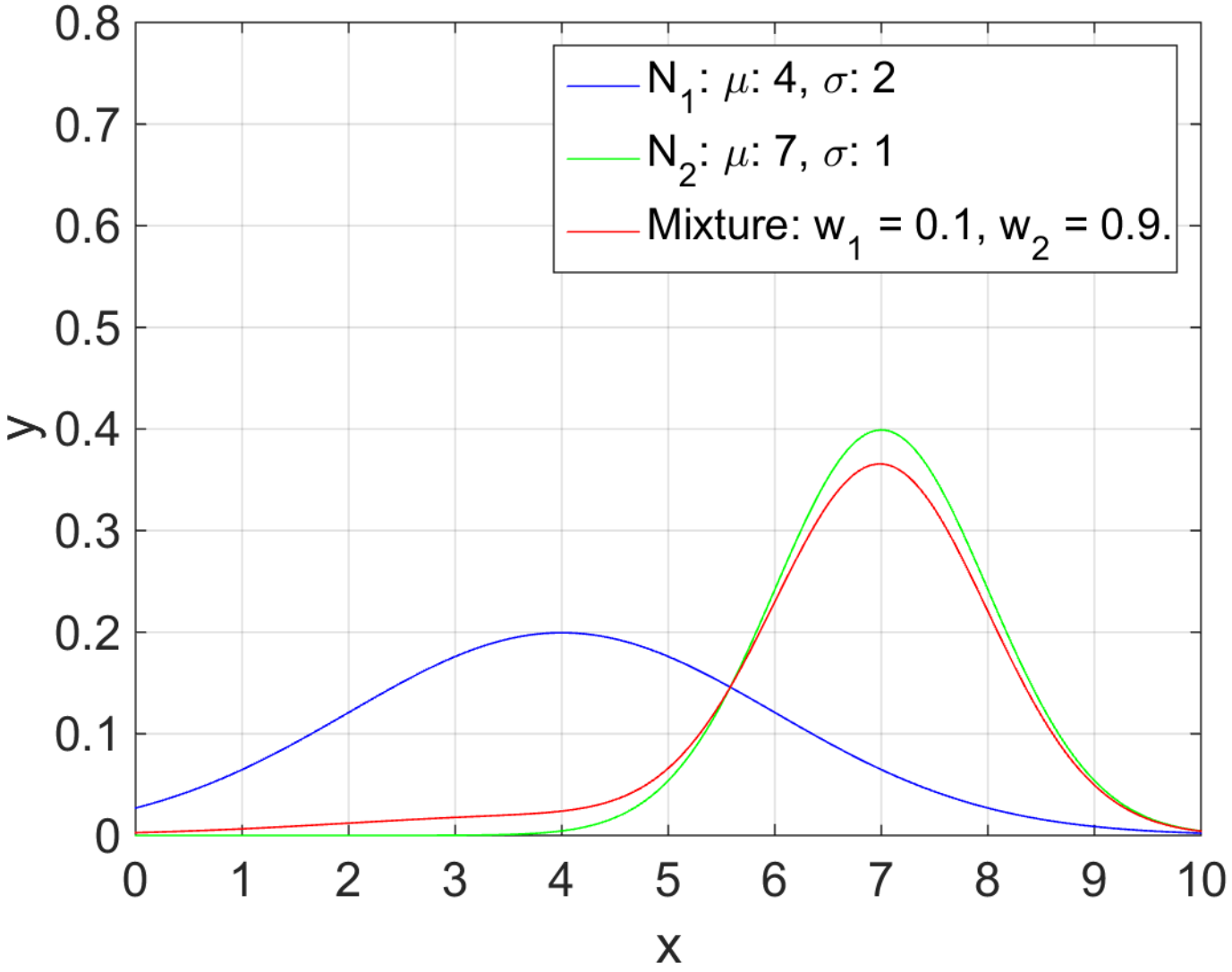
The red curve shows a mixture of those Gaussians.

$w_1 = 0.3$.

$w_2 = 0.7$.

The mixture now resembles N_2 more than N_1.

Mixtures of Gaussians – Example



The blue and green curves show two Gaussians.

The red curve shows a mixture of those Gaussians.

$w_1 = 0.1$.

$w_2 = 0.9$.

The mixture now is almost identical to $N_2(x)$.

Learning a Mixture of Gaussians

- Suppose we are given training data $x_1, x_2, \ldots, x_n$.
- How can we fit a mixture of Gaussians to this data?
- This will be the topic of the next few slides.
- We will learn a very popular machine learning algorithm, called **the EM algorithm.**
  - EM stands for **Expectation-Maximization**.
- Step 0 of the EM algorithm: pick k manually.
  - Decide how many Gaussians the mixture should have.
  - Any approach for choosing k automatically is beyond the scope of this class.

Learning a Mixture of Gaussians

- Suppose we are given training data x_1, x_2, …, x_n.
- We want to model P(x) as a mixture of Gaussians.
- Given k, how many parameters do we need to estimate in order to fully define the mixture?
- Remember, a mixture M of k Gaussians is defined as:

$$M(x) = \sum_{i=1}^{k} w_i N_i(x) = \sum_{i=1}^{k} \left[w_i \frac{1}{\sigma_i \sqrt{2\pi}} e^{-\frac{(x-\mu_i)^2}{2\sigma_i^2}} \right]$$

- For each N_i, we need to estimate three numbers:
  - w_i, μ_i, σ_i.
- So, in total, we need to estimate 3*k numbers.

Learning a Mixture of Gaussians

- Suppose we are given training data x_1, x_2, …, x_n.
- A mixture M of k Gaussians is defined as:

$$M(x) = \sum_{i=1}^{k} w_i N_i(x) = \sum_{i=1}^{k} \left[w_i \frac{1}{\sigma_i \sqrt{2\pi}} e^{-\frac{(x-\mu_i)^2}{2\sigma_i^2}} \right]$$

- For each N_i, we need to estimate w_i, μ_i, σ_i.

- Suppose that we knew for each x_j, that it belongs to one and only one of the k Gaussians.
- Then, learning the mixture would be a piece of cake:
- For each Gaussian N_i:
  - Estimate μ_i, σ_i based on the examples that belong to it.
  - Set w_i equal to the fraction of examples that belong to N_i.

Learning a Mixture of Gaussians

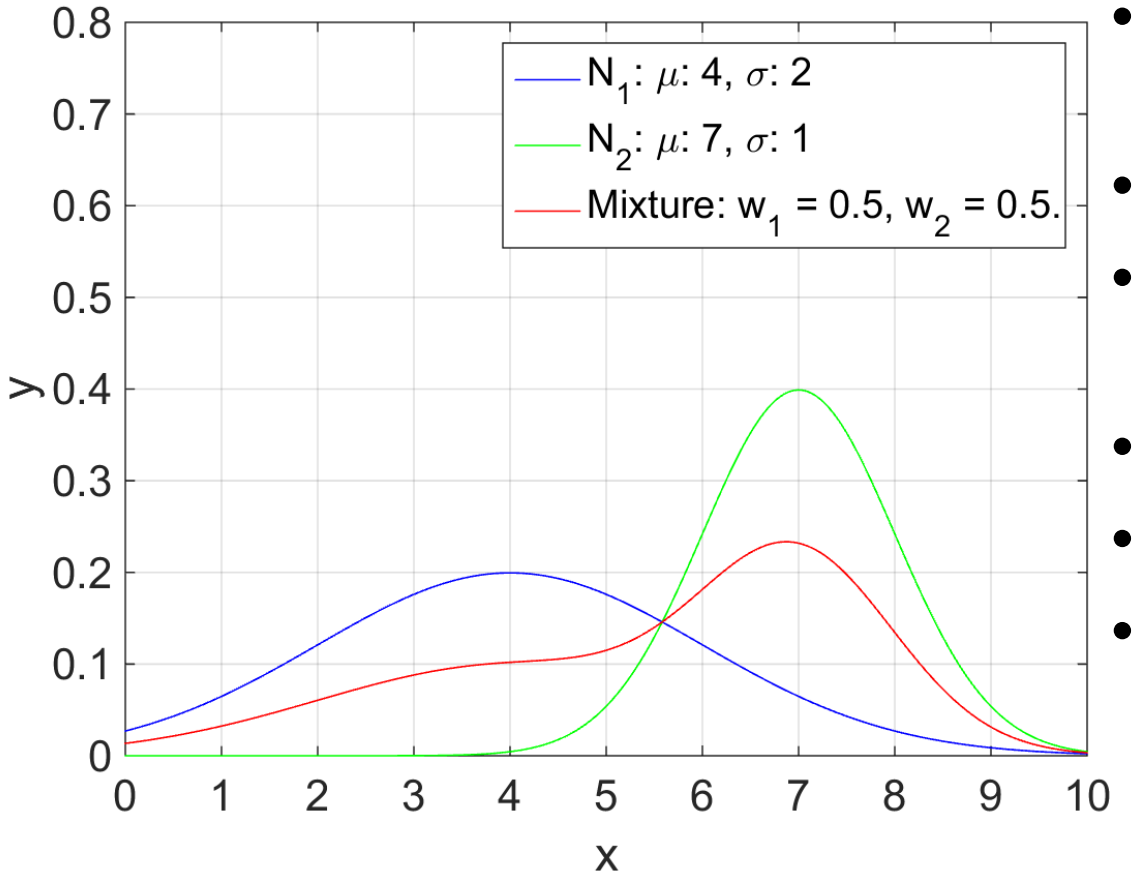
- Suppose we are given training data x_1, x_2, …, x_n.
- A mixture M of k Gaussians is defined as:

$$M(x) = \sum_{i=1}^{k} w_i N_i(x) = \sum_{i=1}^{k} \left[w_i \frac{1}{\sigma_i \sqrt{2\pi}} e^{-\frac{(x-\mu_i)^2}{2\sigma_i^2}} \right]$$

- For each N_i, we need to estimate w_i, μ_i, σ_i.

- However, we have no idea which mixture each x_j belongs to.
- If we knew μ_i and σ_i for each N_i, we could **probabilistically** assign each x_j to a component.
  - "Probabilistically" means that we would not make a hard assignment, but we would partially assign x_j to different components, with each assignment weighted proportionally to the density value $N_i(x_j)$.

Example of Partial Assignments



- Using our previous example of a mixture:
- Suppose $x_{j=}$ 6.5.
- How do we assign 6.5 to the two Gaussians?
- $N_1(6.5) = 0.0913$.
- $N_2(6.5) = 0.3521$.
- So:
  - 6.5 belongs to N_1 by $\frac{0.0913}{0.0913+0.3521} = 20.6\%$.
  - 6.5 belongs to N_2 by $\frac{0.3521}{0.0913+0.3521} = 79.4\%$.

The Chicken-and-Egg Problem

- To recap, fitting a mixture of Gaussians to data involves estimating, for each N_i, values w_i, μ_i, σ_i.
- If we could assign each x_j to one of the Gaussians, we could compute easily w_i, μ_i, σ_i.
  - Even if we probabilistically assign x_j to multiple Gaussians, we can still easily w_i, μ_i, σ_i, by adapting our previous formulas. We will see the adapted formulas in a few slides.
- If we knew μ_i, σ_i and w_i, we could assign (at least probabilistically) x_j's to Gaussians.
- So, this is a chicken-and-egg problem.
  - If we knew one piece, we could compute the other.
  - But, we know neither. So, what do we do?

On Chicken-and-Egg Problems

- Such chicken-and-egg problems occur frequently in AI.
- Surprisingly (at least to people new in AI), we can easily solve such chicken-and-egg problems.
- Overall, chicken and egg problems in AI look like this:
  - We need to know A to estimate B.
  - We need to know B to compute A.
- There is a fairly standard recipe for solving these problems.
- Any guesses?

On Chicken-and-Egg Problems

- Such chicken-and-egg problems occur frequently in AI.
- Surprisingly (at least to people new in AI), we can easily solve such chicken-and-egg problems.
- Overall, chicken and egg problems in AI look like this:
  - We need to know A to estimate B.
  - We need to know B to compute A.
- There is a fairly standard recipe for solving these problems.
- Start by giving to A values chosen randomly (or perhaps non-randomly, but still in an uninformed way, since we do not know the correct values).
- Repeat this loop:
  - Given our current values for A, estimate B.
  - Given our current values of B, estimate A.
  - If the new values of A and B are very close to the old values, break.

The EM Algorithm - Overview

- We use this approach to fit mixtures of Gaussians to data.
- This algorithm, that fits mixtures of Gaussians to data, is called the EM algorithm (**Expectation-Maximization** algorithm).
- Remember, we choose k (the number of Gaussians in the mixture) manually, so we don't have to estimate that.
- To initialize the EM algorithm, we initialize each μ_i, σ_i, and w_i. Values w_i are set to 1/k. We can initialize μ_i, σ_i in different ways:
  - Giving random values to each μ_i.
  - Uniformly spacing the values given to each μ_i.
  - Giving random values to each σ_i.
  - Setting each σ_i to 1 initially.
- Then, we iteratively perform two steps.
  - The **E-step**.
  - The **M-step**.

The E-Step

- E-step. Given our **current estimates** for μ_i, σ_i, and w_i:
  - We compute, for each i and j, the probability $p_{ij} = P(N_i \mid x_j)$: the probability that x_j was generated by Gaussian N_i.
  - How? Using Bayes rule.

$$p_{ij} = \mathrm{P}(N_i|x_j) = \frac{P(x_j \mid N_i) * P(N_i)}{P(x_j)} = \frac{N_i(x_j) * w_i}{P(x_j)}$$

$$N_i(x_j) = \frac{1}{\sigma_i\sqrt{2\pi}} e^{-\frac{(x-\mu_i)^2}{2\sigma_i^2}}$$

The M-Step: Updating μ_i and σ_i

- M-step. Given our current estimates of p_{ij}, for each i, j:
  - We compute μ_i and σ_i for each N_i, as follows:

$$\mu_i = \frac{\sum_{j=1}^{n}[p_{ij}x_j]}{\sum_{j=1}^{n} p_{ij}}$$

$$\sigma_i = \sqrt{\frac{\sum_{j=1}^{n}[p_{ij}(x_j - \mu_i)^2]}{\sum_{j=1}^{n} p_{ij}}}$$

  - To understand these formulas, it helps to compare them to the standard formulas for fitting a Gaussian to data:

$$\mu = \frac{1}{n}\sum_{1}^{n} x_j$$

$$\sigma = \sqrt{\frac{1}{n-1}\sum_{j=1}^{n}(x_j - \mu)^2}$$

The M-Step: Updating μ_i and σ_i

$$\mu_i = \frac{\sum_{j=1}^{n} [p_{ij} x_j]}{\sum_{j=1}^{n} p_{ij}}$$

$$\sigma_i = \sqrt{\frac{\sum_{j=1}^{n} [p_{ij} (x_j - \mu_i)^2]}{\sum_{j=1}^{n} p_{ij}}}$$

- To understand these formulas, it helps to compare them to the standard formulas for fitting a Gaussian to data:

$$\mu = \frac{1}{n} \sum_{1}^{n} x_j$$

$$\sigma = \sqrt{\frac{1}{n-1} \sum_{j=1}^{n} (x_j - \mu)^2}$$

- Why do we take **<u>weighted</u>** averages at the M-step?
- Because each x_j is probabilistically assigned to multiple Gaussians.
- We use $p_{ij} = P(N_i | x_j)$ as weight of the assignment of x_j to N_i.

The M-Step: Updating w_i

$$w_i = \frac{\sum_{j=1}^{n} p_{ij}}{\sum_{i=1}^{k} \left[\sum_{j=1}^{n} p_{ij} \right]}$$

- At the M-step, in addition to updating μ_i and σ_i, we also need to update w_i, which is the weight of the i-th Gaussian in the mixture.
- The formula shown above is used for the update of w_i.
  - We sum up the weights of all objects for the i-th Gaussian.
  - We divide that sum by the sum of weights of all objects for all Gaussians.
  - The division ensures that $\sum_{i=1}^{k} w_i = 1$.

The EM Steps: Summary

- E-step: Given current estimates for each μ_i, σ_i, and w_i, update p_{ij} :

$$p_{ij} = \frac{N_i\left(x_j\right) * w_i}{P(x_j)}$$

- M-step: Given our current estimates for each p_{ij}, update μ_i, σ_i, and w_i:

$$\mu_i = \frac{\sum_{j=1}^{n}[p_{ij}\, x_j]}{\sum_{j=1}^{n} p_{ij}}$$

$$\sigma_i = \sqrt{\frac{\sum_{j=1}^{n}[p_{ij}\,(x_j - \mu_i)^2]}{\sum_{j=1}^{n} p_{ij}}}$$

$$w_i = \frac{\sum_{j=1}^{n} p_{ij}}{\sum_{i=1}^{k}\left[\sum_{j=1}^{n} p_{ij}\right]}$$

The EM Algorithm - Termination

- The log likelihood of the training data is defined as:

$$L(x_1, \ldots, x_n) = \sum_{j=1}^{n} \ln\big(M(x_j)\big)$$

- As a reminder, M is the Gaussian mixture, defined as:

$$M(x) = \sum_{i=1}^{k} w_i N_i(x) = \sum_{i=1}^{k} \left[w_i \frac{1}{\sigma_i \sqrt{2\pi}} e^{-\frac{(x-\mu_i)^2}{2{\sigma_i}^2}} \right]$$

- One can prove that, after each iteration of the E-step and the M-step, this log likelihood increases or stays the same.
- We check how much the log likelihood changes at each iteration.
- When the change is below some threshold, we stop.

The EM Algorithm: Summary

- Initialization:
  - Initialize each μ_i, σ_i, w_i, using your favorite approach (e.g., set each μ_i to a random value, and set each σ_i to 1, set each w_i equal to 1/k).
  - last_log_likelihood = -infinity.
- Main loop:
  - E-step:
    - Given our current estimates for each μ_i, σ_i, and w_i, update each p_{ij}.
  - M-step:
    - Given our current estimates for each p_{ij}, update each μ_i, σ_i, and w_i.
  - log_likelihood = $L(x_1, \ldots, x_n)$.
  - if (log_likelihood – last_log_likelihood) < threshold, **break**.
  - last_log_likelihood = log_likelihood

The EM Algorithm: Limitations

- When we fit a **Gaussian** to data, we always get the same result.
- We can also prove that the result that we get is the best possible result.
  - There is no other Gaussian giving a higher log likelihood to the data, than the one that we compute as described in these slides.
- When we fit a **mixture of Gaussians** to the same data, do we always end up with the same result?

The EM Algorithm: Limitations

- When we fit a **Gaussian** to data, we always get the same result.
- We can also prove that the result that we get is the best possible result.
  - There is no other Gaussian giving a higher log likelihood to the data, than the one that we compute as described in these slides.
- When we fit a **mixture of Gaussians** to the same data, we (sadly) do not always get the same result.
- The EM algorithm is a greedy algorithm.
- The result depends on the initialization values.
- We may have bad luck with the initial values, and end up with a bad fit.
- There is no good way to know if our result is good or bad, or if better results are possible.

Multidimensional Gaussians

- Instead of assuming that each dimension is independent, we can instead model the distribution using a multi-dimensional Gaussian:

$$N(v) = \frac{1}{\sqrt{(2\pi)^d |\Sigma|}} \exp\left(-\frac{1}{2}(x - \mu)^{\mathrm{T}} \Sigma^{-1} (x - \mu)\right)$$

- To specify this Gaussian, we need to estimate the mean μ and the covariance matrix Σ.
- In the above formula, $|\Sigma|$ denotes the **determinant** of covariance matrix Σ.

Multidimensional Gaussians - Mean

- Let x_1, x_2, …, x_n be d-dimensional vectors.
- $x_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,d})$, where each $x_{i,j}$ is a real number.
- Then, the mean $\mu = (\mu_1, \ldots, \mu_d)$ is computed as:

$$\mu = \frac{1}{n} \sum_{1}^{n} x_i$$

- Therefore, $\mu_j = \frac{1}{n} \sum_{i=1}^{n} x_{i,j}$

Multidimensional Gaussians – Covariance Matrix

- Let $x_1, x_2, \ldots, x_n$ be d-dimensional vectors.
- $x_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,d})$, where each $x_{i,j}$ is a real number.
- Let Σ be the covariance matrix. Its size is dxd.
- Let $\sigma_{r,c}$ be the value of Σ at row r, column c.

$$\sigma_{r,c} = \frac{1}{n-1} \sum_{j=1}^{n} (x_{j,r} - \mu_r)(x_{j,c} - \mu_c)$$

EM With Multidimensional Gaussians

- We follow the same steps as in the one-dimensional case.
- However, now the means are vectors, and instead of variances we need to estimate covariance matrices.
- Notation:
  - $\boldsymbol{\mu}_i$: mean of the i-th Gaussian in the mixture. It is a vector of the same dimensionality as the data points.
  - $\mu_{i,c}$: value at dimension c of vector $\boldsymbol{\mu}_i$.
  - $\sigma_{i,r,c}$: value at row r and column c of the covariance matrix for the i-th Gaussian in the mixture.

EM With Multidimensional Gaussians

- E-step: Given current estimates for each μ_i, $\sigma_{i,r,c}$, and w_i, update p_{ij}:

$$P(x_j) = \sum_{i=1}^{k} N_i(x_j) * w_i$$

$P(x_j)$ is the probability of x_j, and is computed using the sum rule.

$$p_{ij} = \frac{N_i(x_j) * w_i}{P(x_j)}$$

p_{ij} is the probability that x_j belongs to cluster i (i.e., to Gaussian i) and it is computed using Bayes rule.

EM With Multidimensional Gaussians

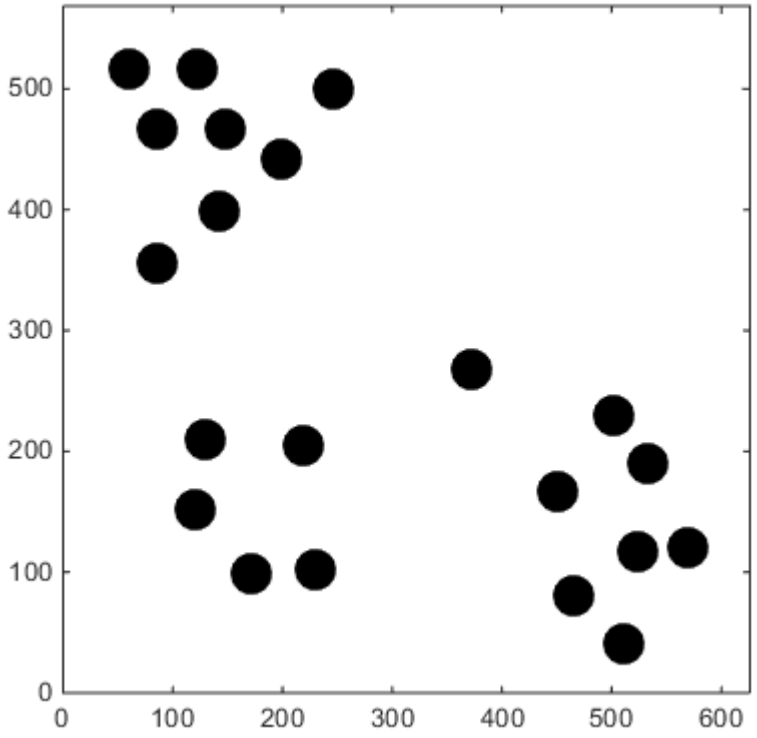
- M-step: Given our current estimates for each p_{ij}, update μ_i, $\sigma_{i,r,c}$, and w_i:

$$w_i = \frac{\sum_{j=1}^{n} p_{ij}}{\sum_{i=1}^{k}\left[\sum_{j=1}^{n} p_{ij}\right]}$$

$$\mu_i = \frac{\sum_{j=1}^{n}[p_{ij}\, x_j]}{\sum_{j=1}^{n} p_{ij}}$$

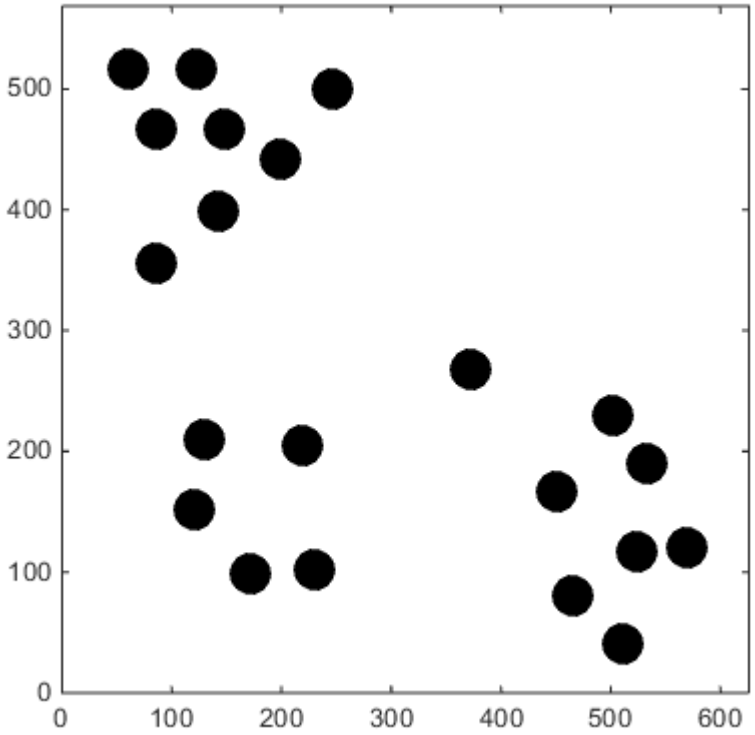
$$\sigma_{i,r,c} = \frac{\sum_{j=1}^{n}[p_{ij}(x_{j,r} - \mu_{i,r})(x_{j,c} - \mu_{i,c})]}{\sum_{j=1}^{n} p_{ij}}$$

EM for Clustering



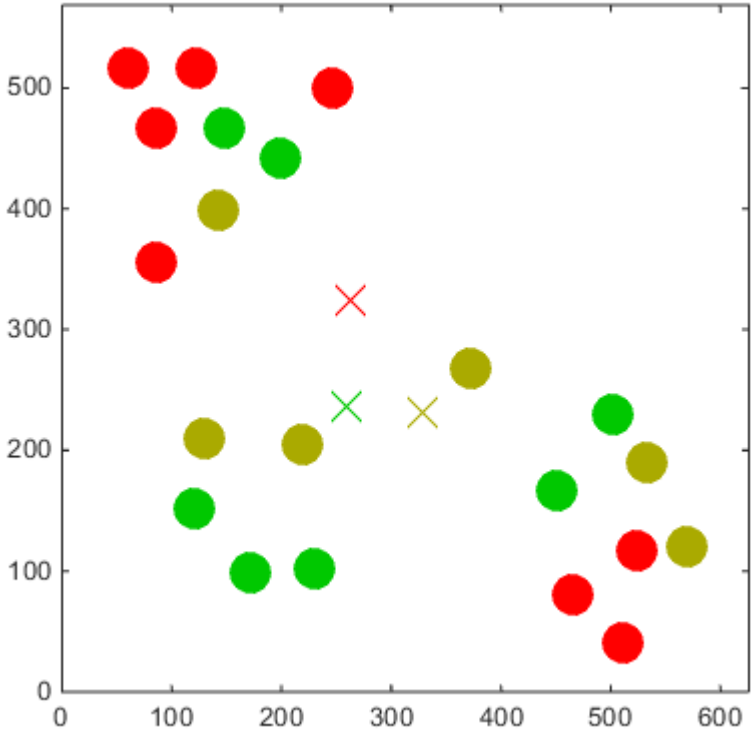
- This EM algorithm can be viewed as a clustering algorithm.
  - Each Gaussian defines a cluster.
  - Weights p_{ij} define the degree to which object $\boldsymbol{x}_j$ is a member of the cluster defined by the i-th Gaussian.
  - These are "soft" assignments of objects to clusters.
  - They can be converted to "hard" assignments by finding, for each object $\boldsymbol{x}_j$, the highest value among weights p_{ij}.

EM Clustering: Initialization



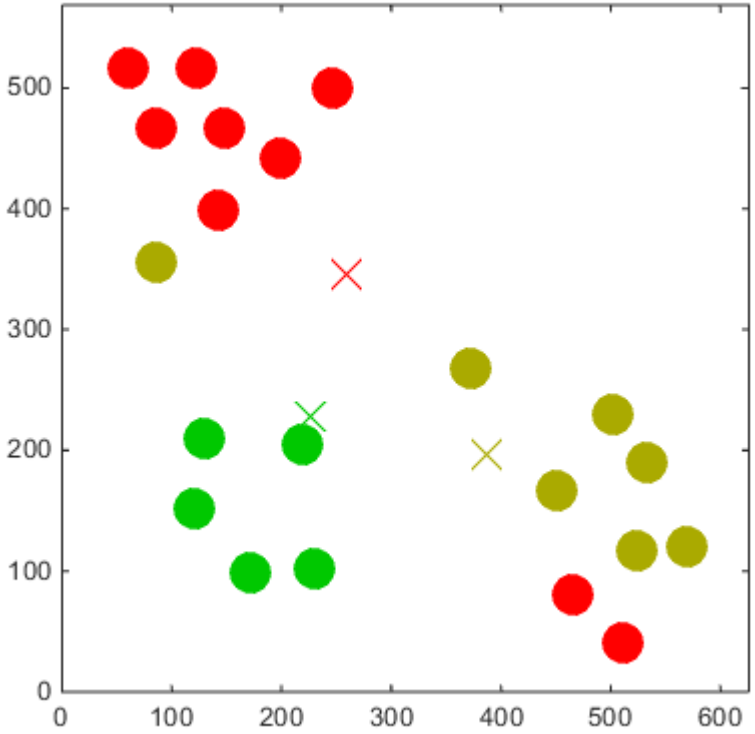
- Let's assume that each x_n is a D-dimensional column vector.
- We want to learn the parameters of K Gaussians N_1, N_2,…, N_K.
- Gaussian N_k is defined by these parameters:
  - Mean μ_k, which is a D-dimensional column vector.
  - Covariance matrix Σ_k, which is a $D \times D$ matrix.
- We initialize each μ_k and each Σ_k to random values.

EM Clustering: Initialization



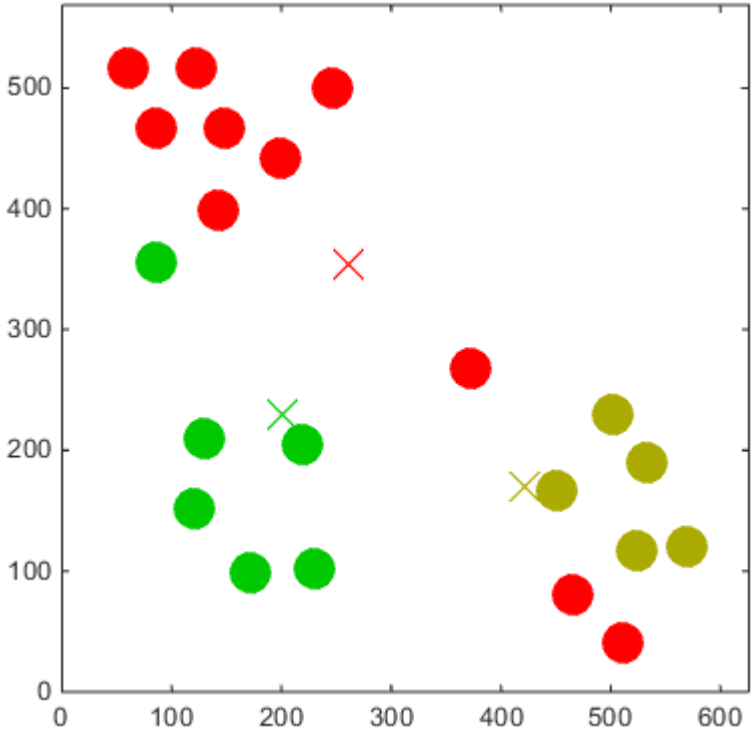
- We initialize each μ_k and each Σ_k to random values.
- The figure shows those initial assignments.
  - For every object x_j, we give it the color of the cluster k for which p_{kn} is the highest.

EM Clustering: Main Loop



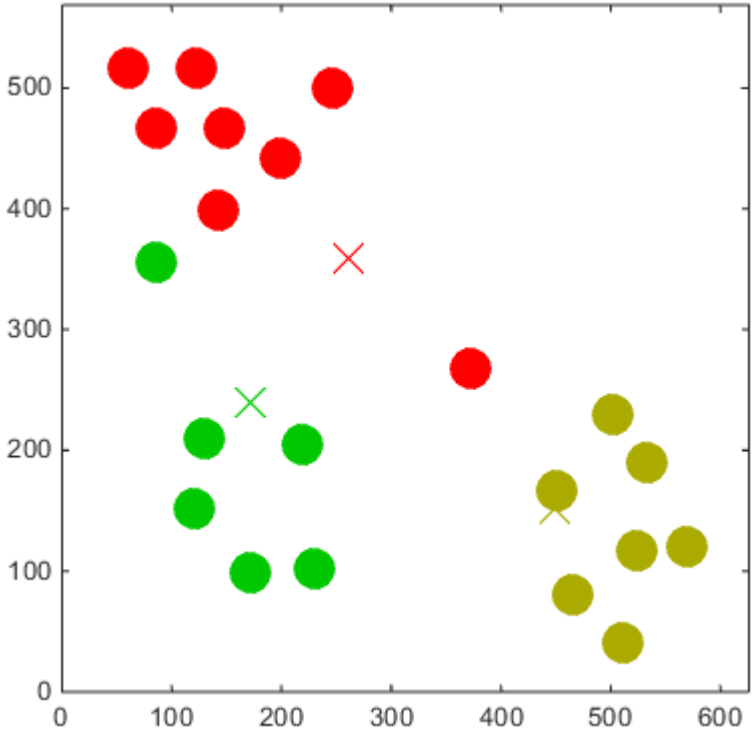
- The main loop alternates between:
  - Computing new assignment probabilities p_{kn} that object x_n belongs to Gaussian N_k.
  - Computing, for each Gaussian N_k, a new mean μ_k, a new covarriance matrix Σ_k , and a new weight w_k, using the current assignment probabilities p_{kn}.
- Here is the result after one iteration of the main loop:

EM Clustering: Main Loop



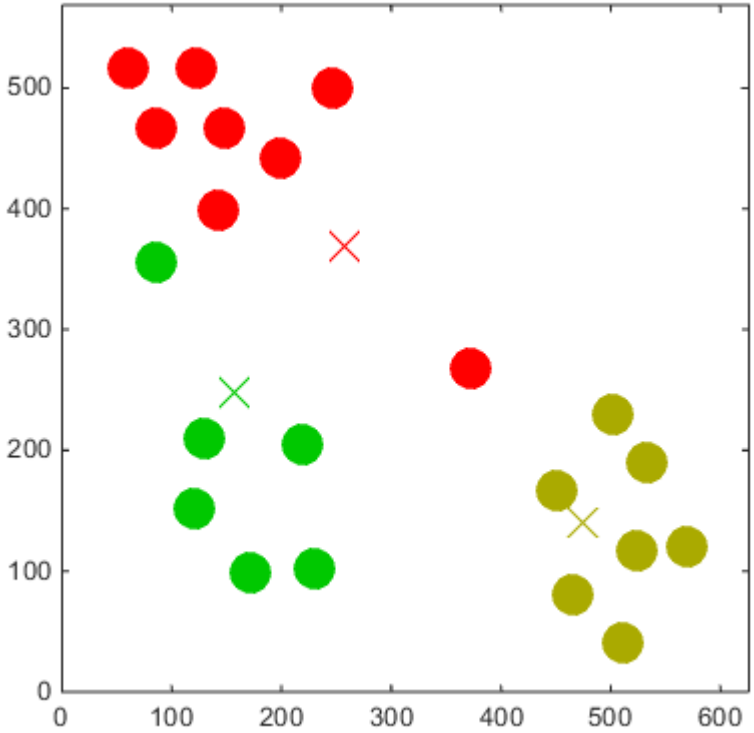
- The main loop alternates between:
  - Computing new assignment probabilities p_{kn} that object x_n belongs to Gaussian N_k.
  - Computing, for each Gaussian N_k, a new mean μ_k, a new covarriance matrix Σ_k , and a new weight w_k, using the current assignment probabilities p_{kn}.
- Here is the result after two iterations of the main loop:

EM Clustering: Main Loop



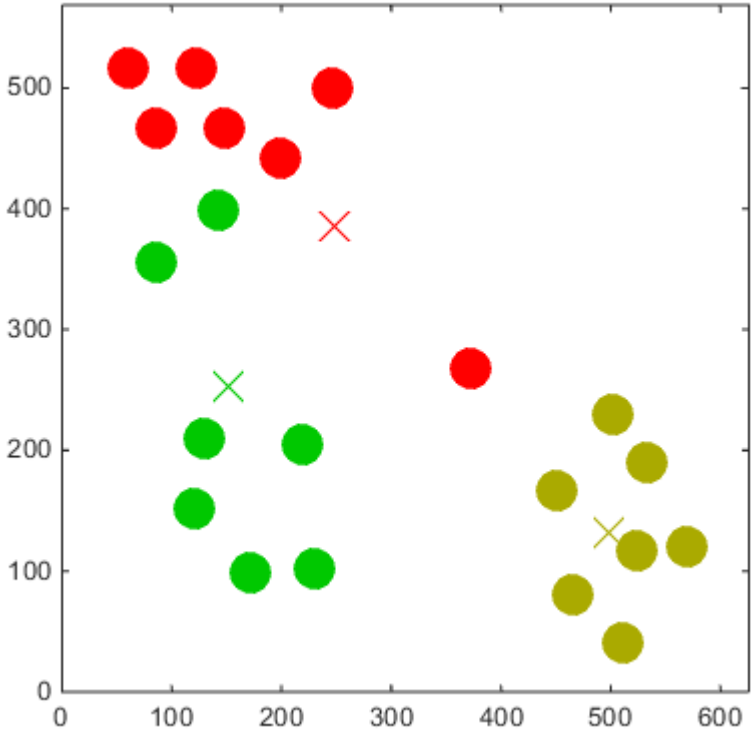
- The main loop alternates between:
  - Computing new assignment probabilities p_{kn} that object x_n belongs to Gaussian N_k.
  - Computing, for each Gaussian N_k, a new mean μ_k, a new covarriance matrix Σ_k , and a new weight w_k, using the current assignment probabilities p_{kn}.
- Here is the result after three iterations of the main loop:

EM Clustering: Main Loop



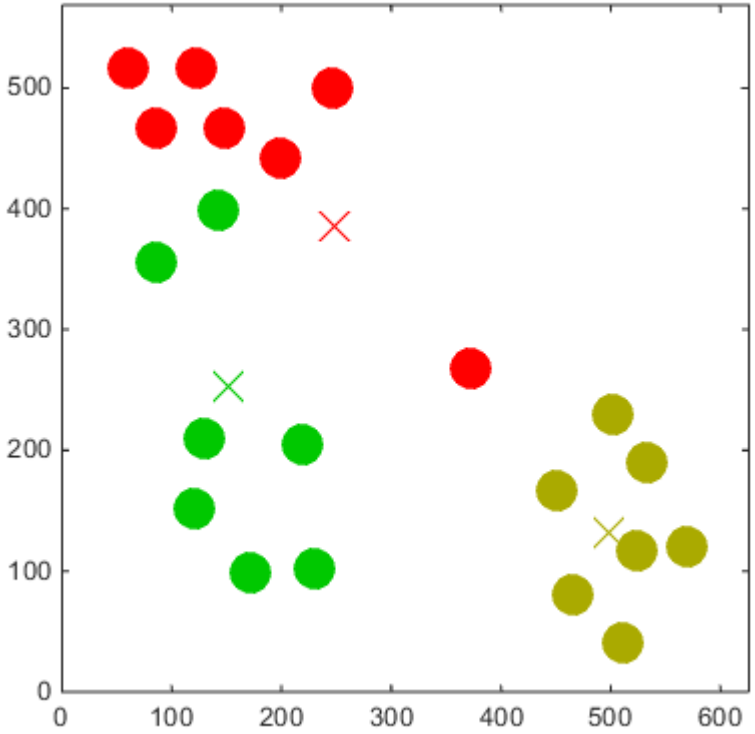
- The main loop alternates between:
  - Computing new assignment probabilities p_{kn} that object x_n belongs to Gaussian N_k.
  - Computing, for each Gaussian N_k, a new mean μ_k, a new covarriance matrix Σ_k , and a new weight w_k, using the current assignment probabilities p_{kn}.
- Here is the result after four iterations of the main loop:

EM Clustering: Main Loop



- The main loop alternates between:
  - Computing new assignment probabilities p_{kn} that object x_n belongs to Gaussian N_k.
  - Computing, for each Gaussian N_k, a new mean μ_k, a new covarriance matrix Σ_k , and a new weight w_k, using the current assignment probabilities p_{kn}.
- Here is the result after five iterations of the main loop.

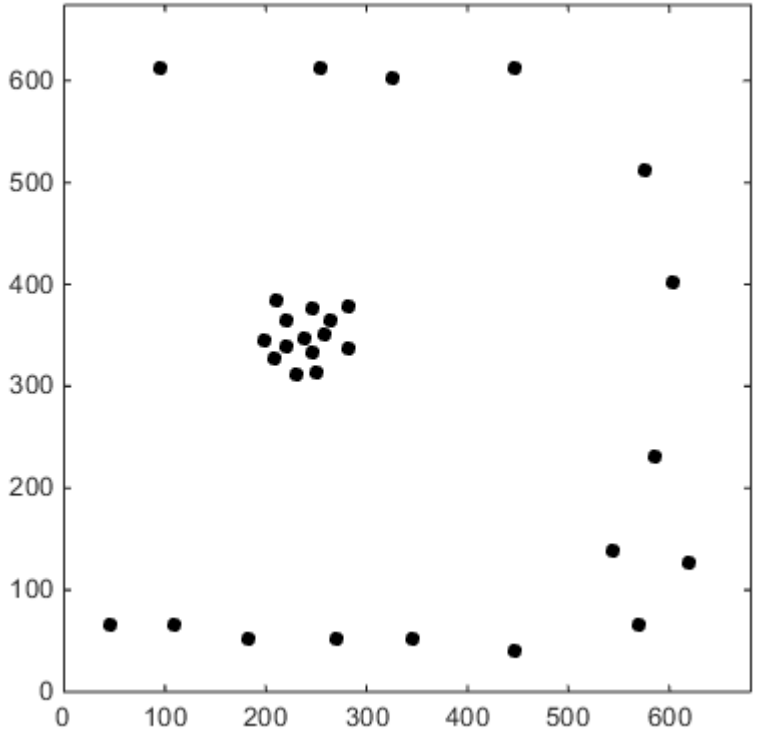
EM Clustering: Main Loop



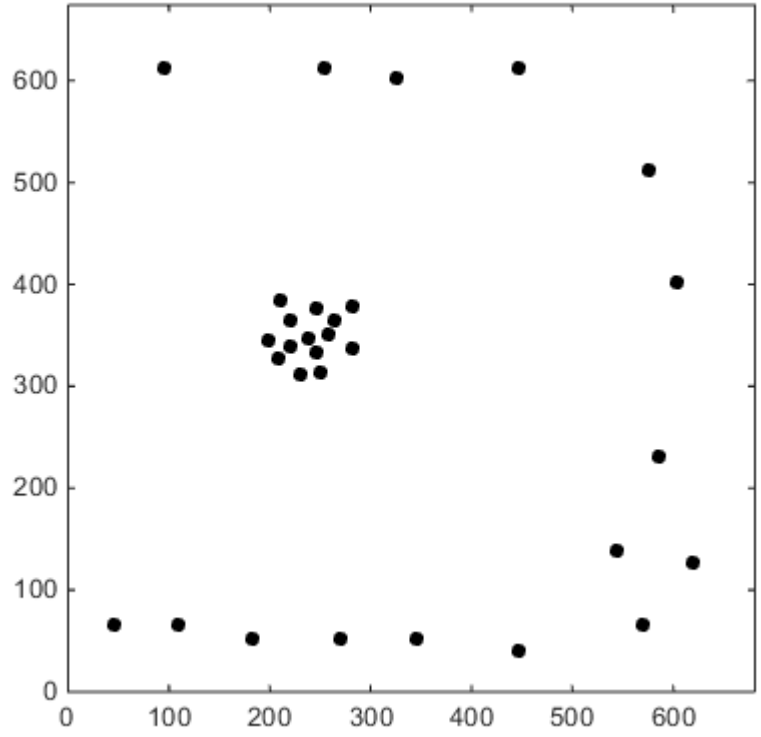
- Here is the result after six iterations of the main loop.
- The results have not changed, so we stop.
- Note that, for this example, EM has not found the same clusters that k-means produced.
  - In general, k-means and EM may perform better or worse, depending on the nature of the data we want to cluster, and our criteria for what defines a good clustering result.

EM Clustering: Another Example

- Here is another example.
- What clusters would you identify here, if we were looking for two clusters?



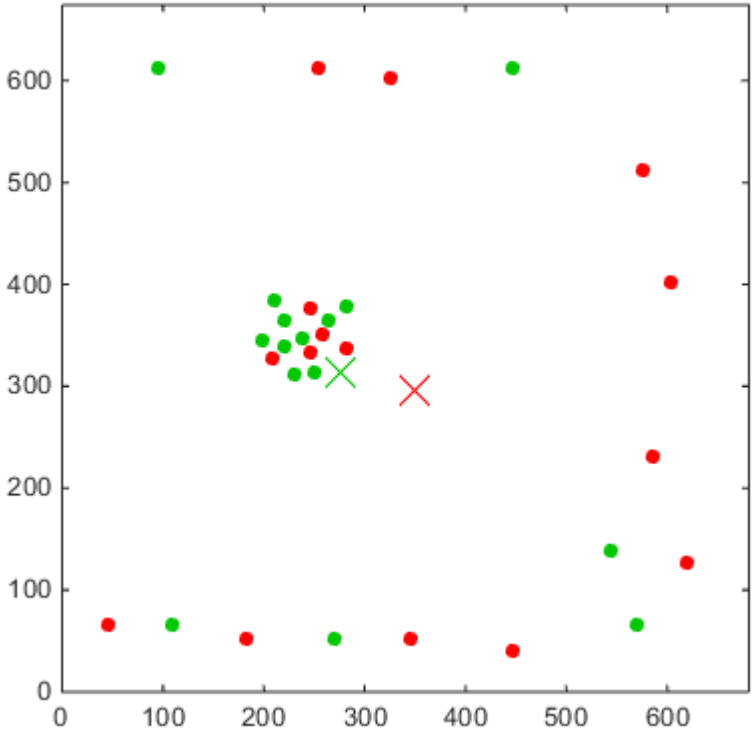
EM Clustering: Another Example



- Here is another example.
- What clusters would you identify here, if we were looking for two clusters?
- In general, there are no absolute criteria for what defines a "good" clustering result, and different people may give different answers.
- However, for many people, the two clusters are:
  - A central cluster of points densely crowded together.
  - A peripheral cluster of points spread out far from the center.

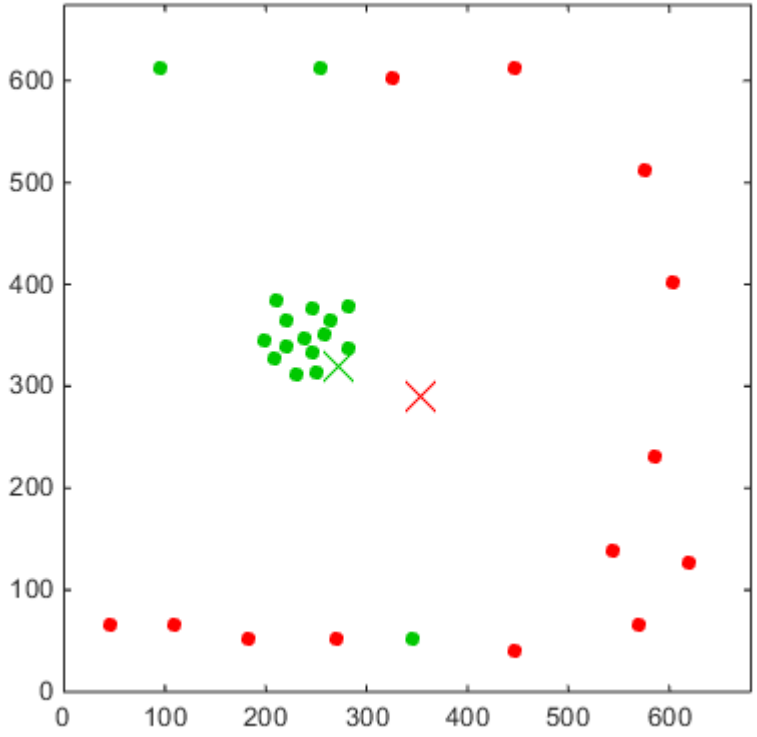
EM Clustering: Another Example

- Let's see how EM does on this dataset.
- Here we see the initialization.



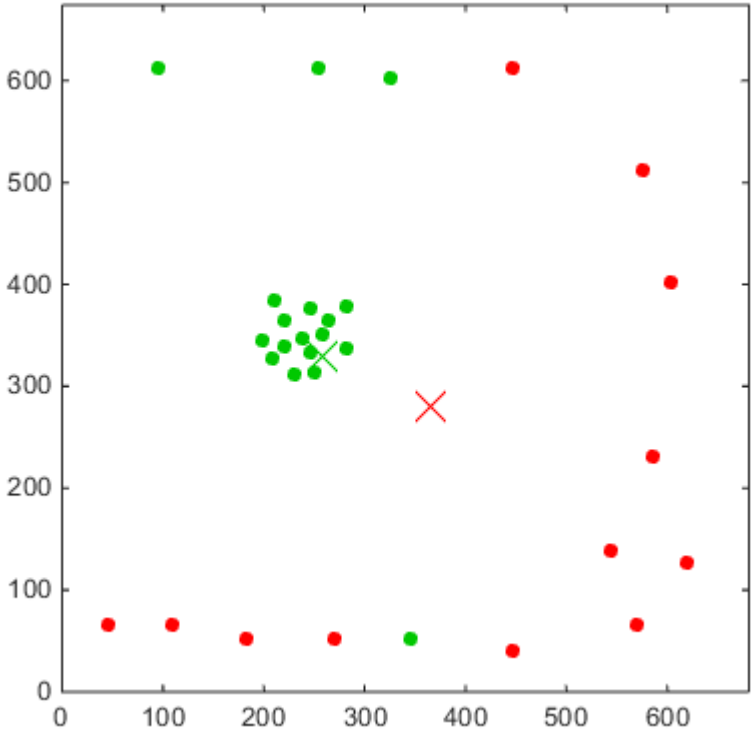
EM Clustering: Another Example

- Let's see how EM does on this dataset.
- Here we see the result after one iteration of the main loop.



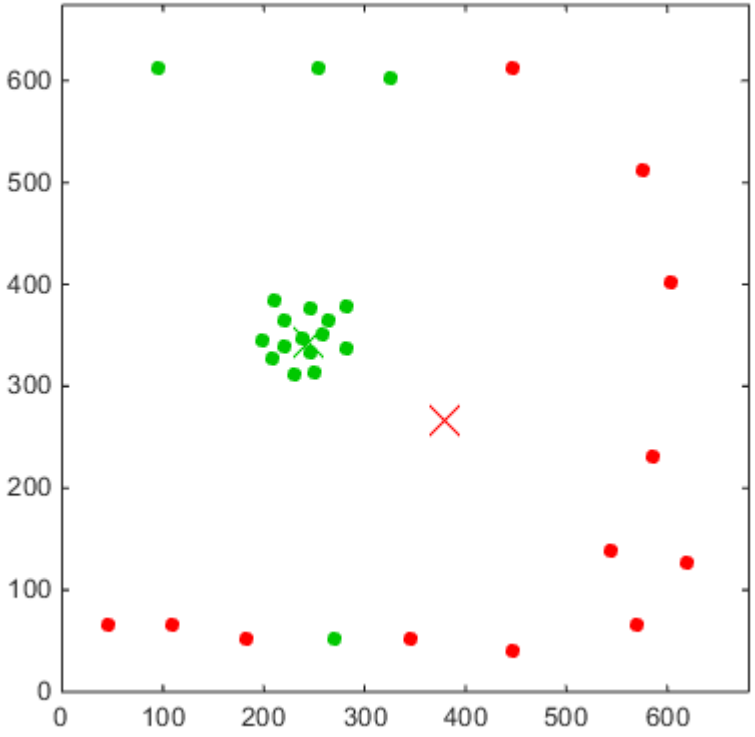
EM Clustering: Another Example

- Let's see how EM does on this dataset.
- Here we see the result after two iterations of the main loop.



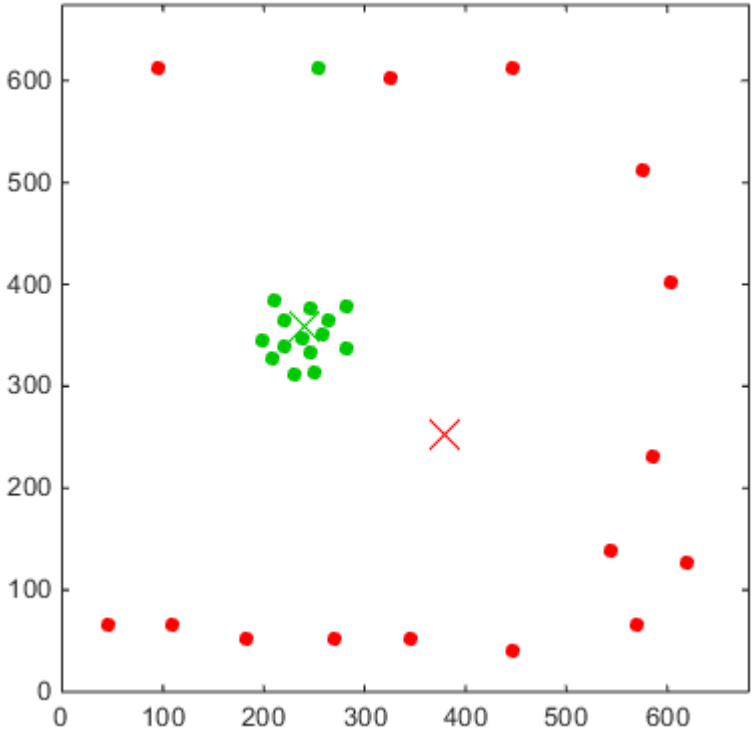
EM Clustering: Another Example

- Let's see how EM does on this dataset.
- Here we see the result after three iterations of the main loop.



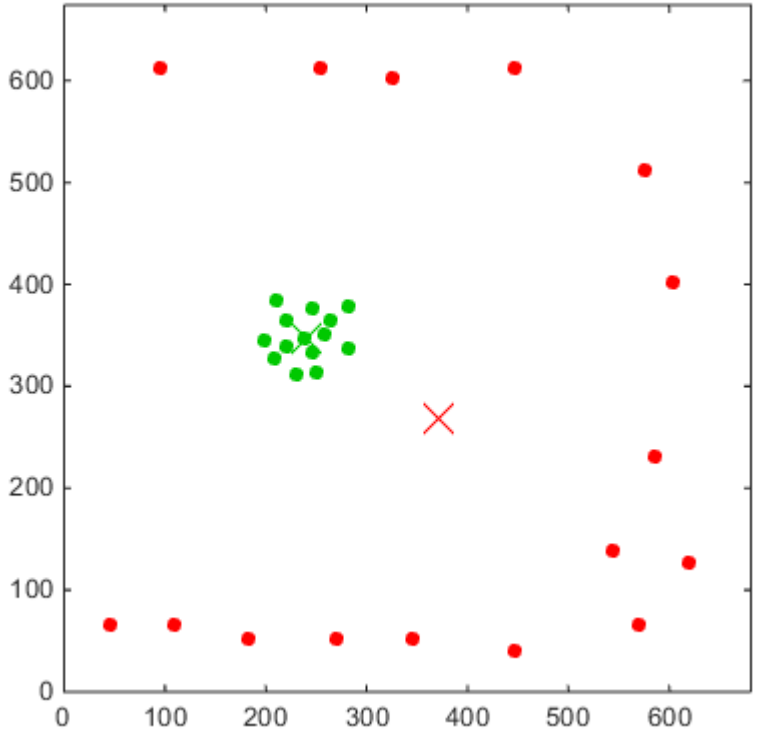
EM Clustering: Another Example

- Let's see how EM does on this dataset.
- Here we see the result after six iterations of the main loop.

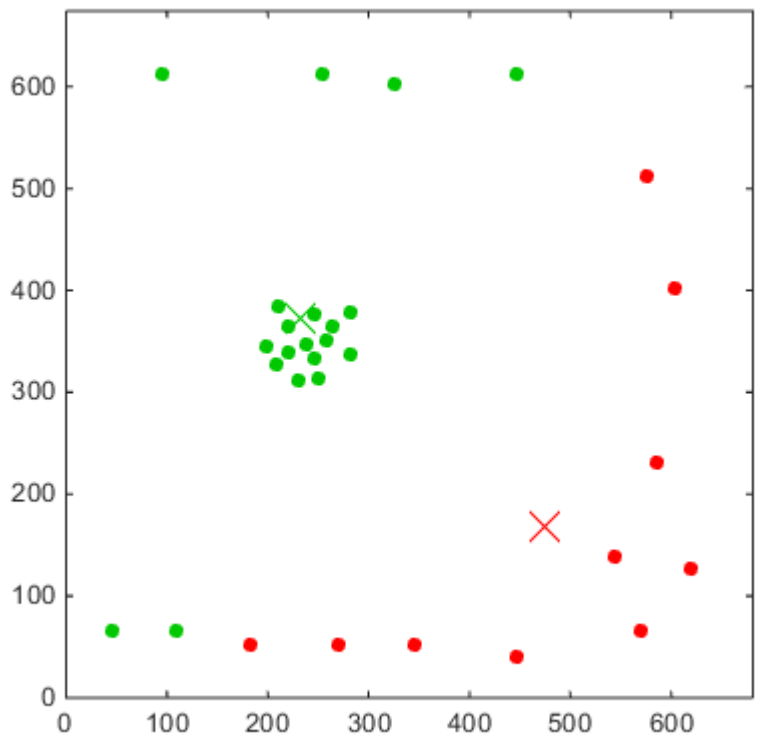


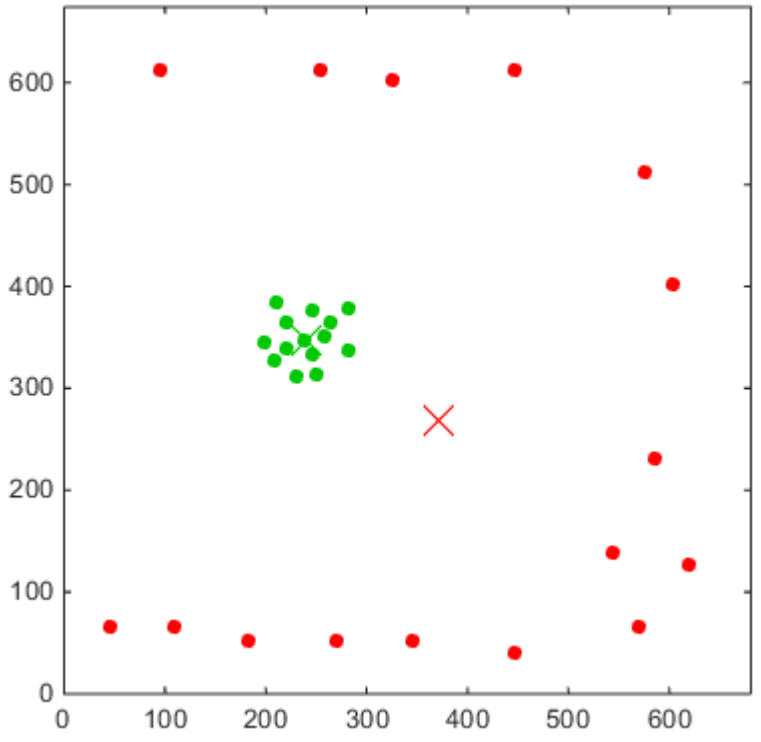
EM Clustering: Another Example

- Let's see how EM does on this dataset.
- Here we see the result after nine iterations of the main loop.
- This is the final result, subsequent iterations do not lead to any changes.



EM vs. K-Means



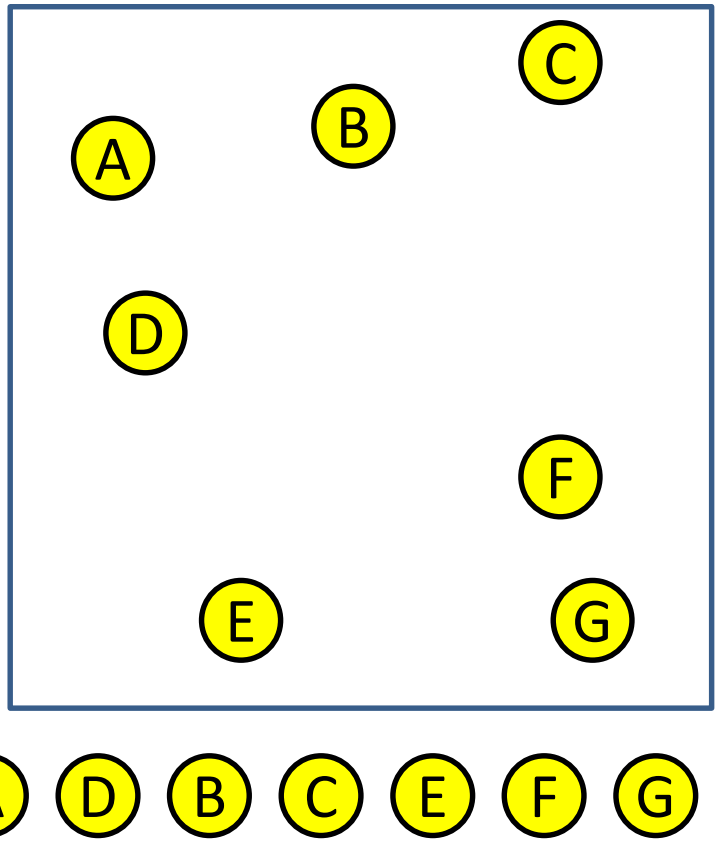


- The k-means result (on the left) is different than the EM result (on the right).
- EM can assign an object to cluster A even if the object is closer to the mean of cluster B.

EM vs. K-Means

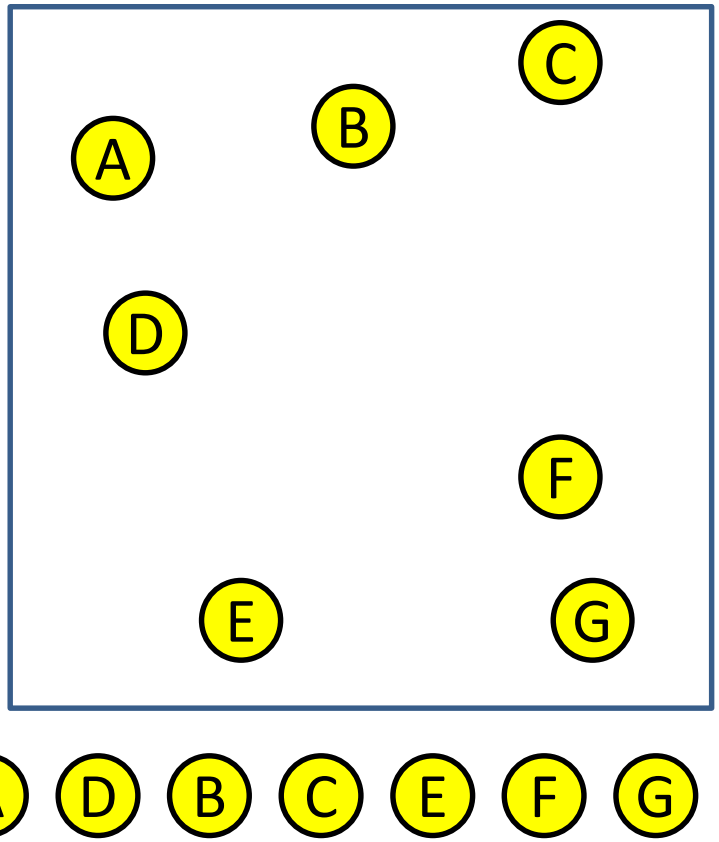
- Another important difference between K-means clustering and EM clustering is:
  - K-means makes hard assignments: an object belongs to one and only one cluster.
  - EM makes soft assignments: weights p_{ij} specify how much each object x_j belongs to cluster i.

Agglomerative Clustering



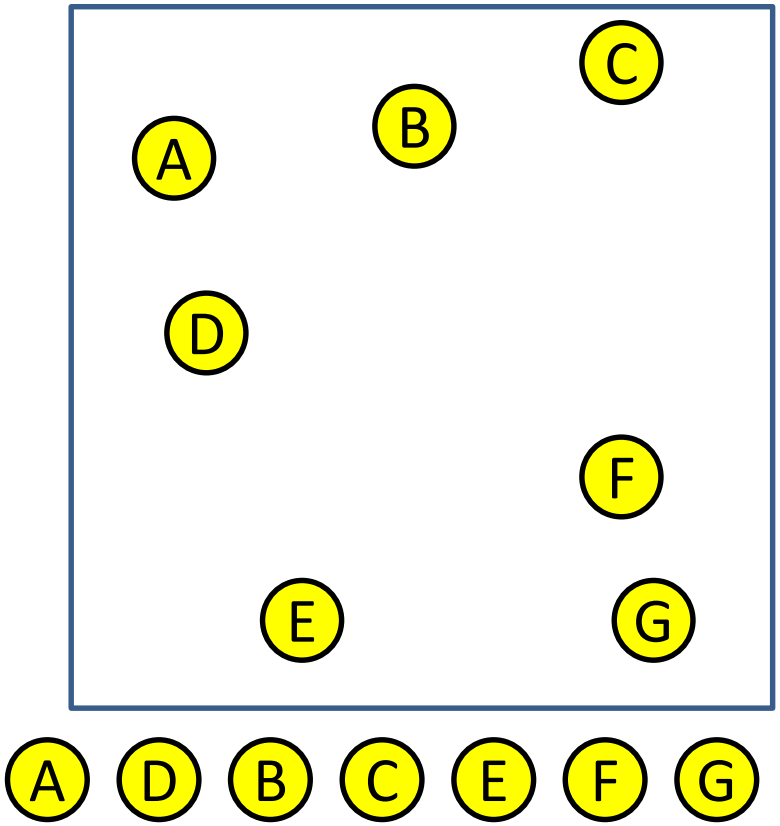
- In agglomerative clustering, there are different levels of clustering.
  - At the top level, every object is its own cluster.
  - Under the top level, each level is obtained by merging the two most similar (less distant) clusters from the previous level.
  - The bottom level has just one cluster, covering the entire dataset.

Agglomerative Clustering



- Under the top level, each level is obtained by merging the **two most similar, or less distant** clusters from the previous level.
- There are different variants of agglomerative clustering.
  - Each variant is specified by the measure for measuring the similarity or distance between two clusters.

Agglomerative Clustering

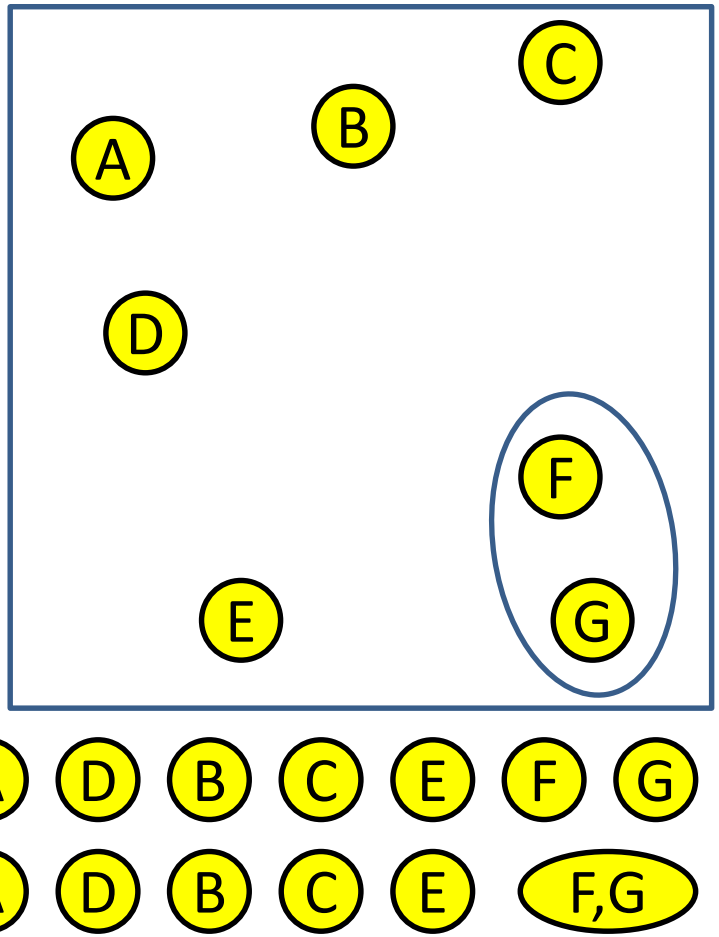


- Suppose that we define the distance of two clusters to be the minimum distance between objects from the two clusters.
- Let d be a distance measure between objects in our data.
- The minimum distance $d_{\min}(X, Y)$ between two sets X and Y is defined as:

$$d_{\min}(X, Y) = \min_{x \in X, y \in Y} d(x, y)$$

- Using $d_{\min}$, what do we merge next?

Agglomerative Clustering

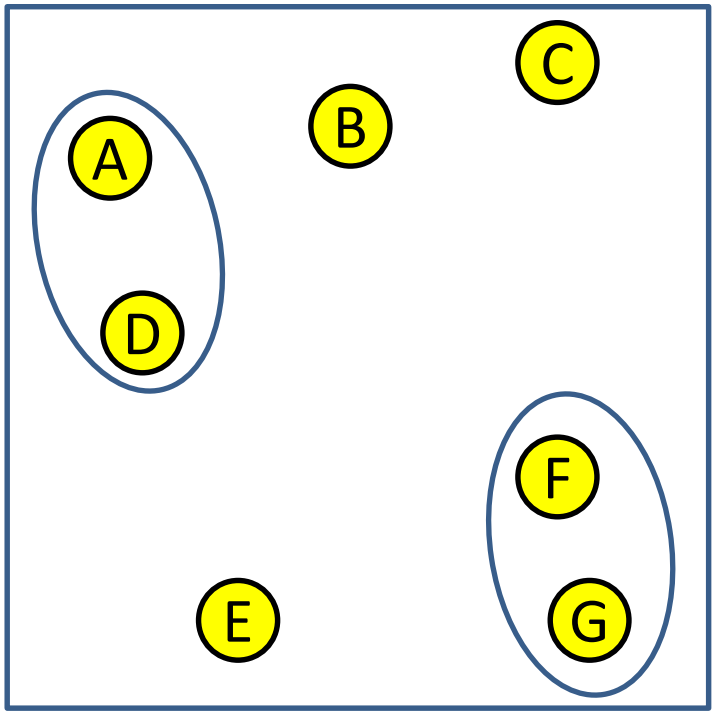


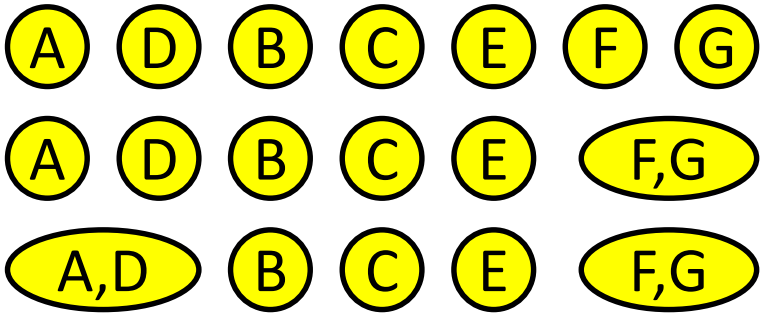
- Suppose that we define the distance of two clusters to be the minimum distance between objects from the two clusters.
- Let d be a distance measure between objects in our data.
- The minimum distance $d_{\min}(X, Y)$ between two sets X and Y is defined as:

$$d_{\min}(X, Y) = \min_{x \in X, y \in Y} d(x, y)$$

- Clusters $\{F\}$ and $\{G\}$ are the closest to each other. Next?

Agglomerative Clustering



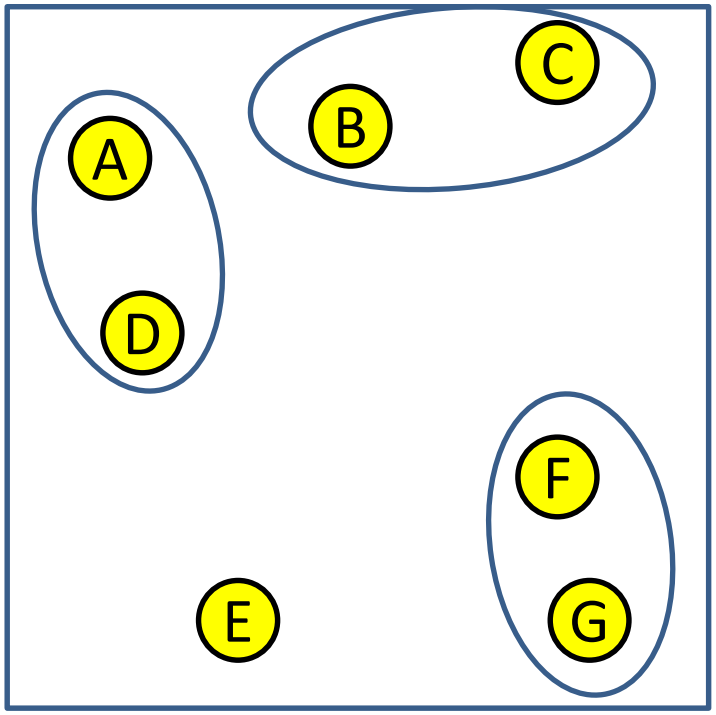


- Suppose that we define the distance of two clusters to be the minimum distance between objects from the two clusters.
- Let d be a distance measure between objects in our data.
- The minimum distance $d_{\min}(X, Y)$ between two sets X and Y is defined as:

$$d_{\min}(X, Y) = \min_{x \in X, y \in Y} d(x, y)$$

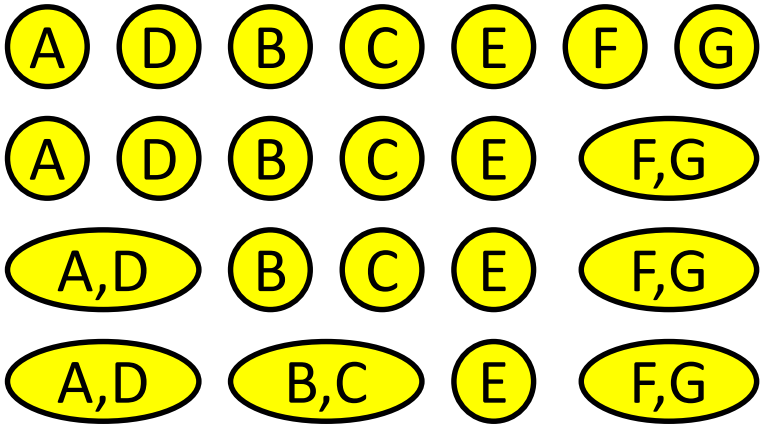
- Clusters $\{A\}$ and $\{D\}$ are the closest to each other. Next?

Agglomerative Clustering



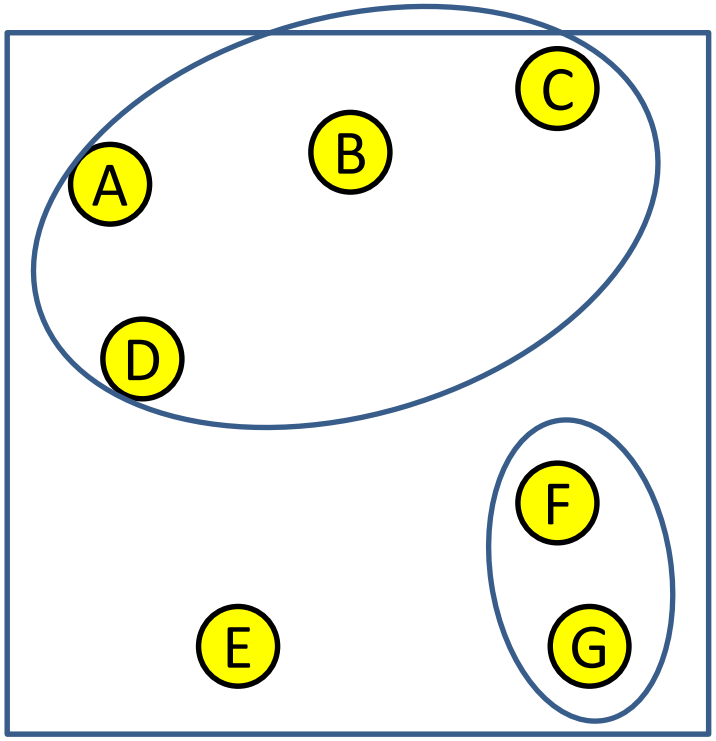
- Suppose that we define the distance of two clusters to be the minimum distance between objects from the two clusters.
- Let d be a distance measure between objects in our data.
- The minimum distance $d_{\min}(X, Y)$ between two sets X and Y is defined as:

$$d_{\min}(X, Y) = \min_{x \in X, y \in Y} d(x, y)$$



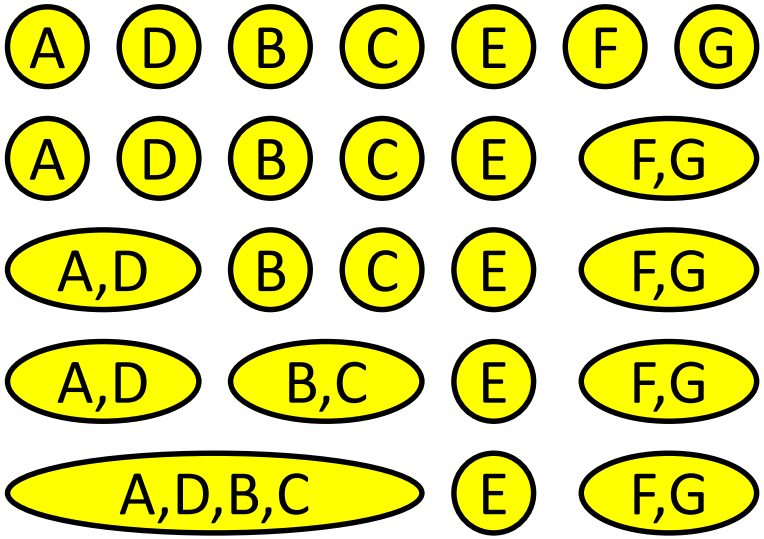
- Clusters $\{B\}$ and $\{C\}$ are the closest to each other. Next?

Agglomerative Clustering



- Suppose that we define the distance of two clusters to be the minimum distance between objects from the two clusters.
- Let d be a distance measure between objects in our data.
- The minimum distance $d_{\min}(X, Y)$ between two sets X and Y is defined as:

$$d_{\min}(X, Y) = \min_{x \in X, y \in Y} d(x, y)$$



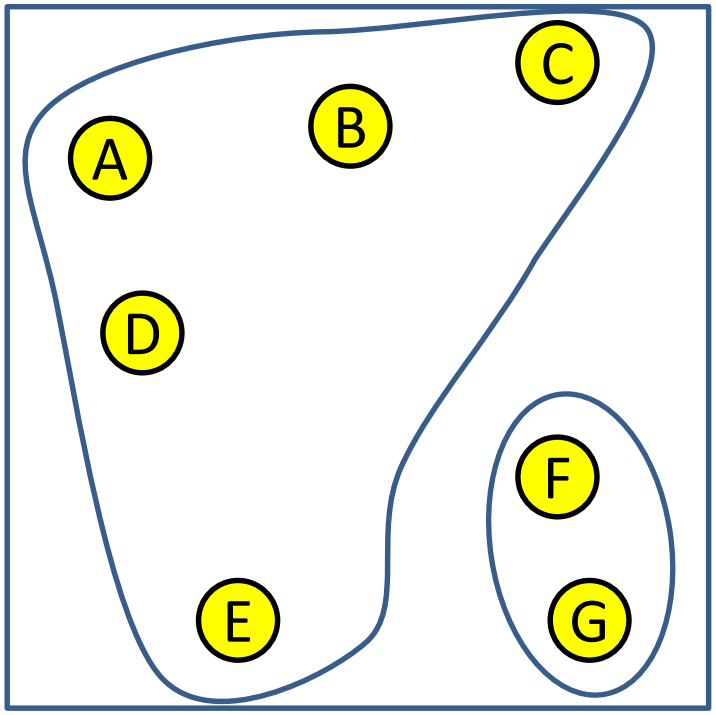
- Clusters $\{A, D\}$ and $\{B, C\}$ are the closest to each other. Next?

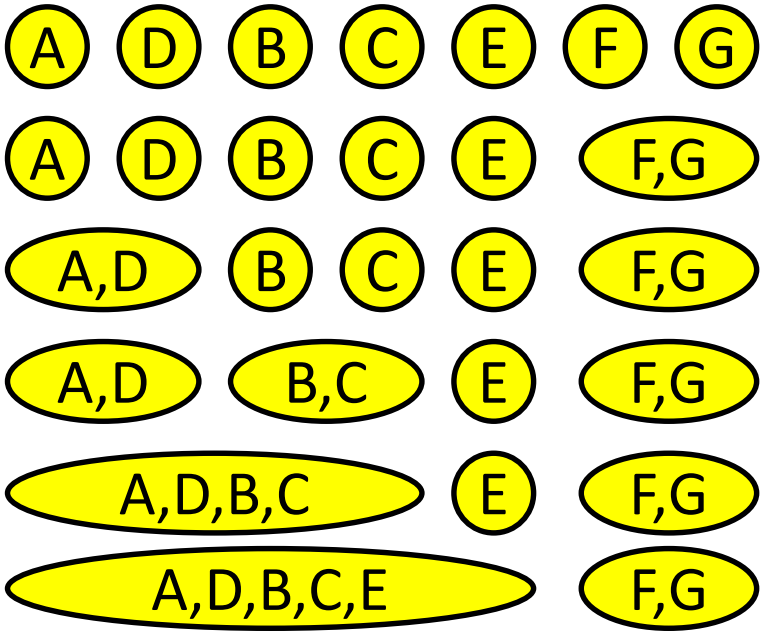
Agglomerative Clustering

- Suppose that we define the distance of two clusters to be the minimum distance between objects from the two clusters.
- Let d be a distance measure between objects in our data.
- The minimum distance $d_{\min}(X, Y)$ between two sets X and Y is defined as:

$$d_{\min}(X, Y) = \min_{x \in X, y \in Y} d(x, y)$$

- Merging $\{A, D, B, C\}$ and $\{E\}$. Next?



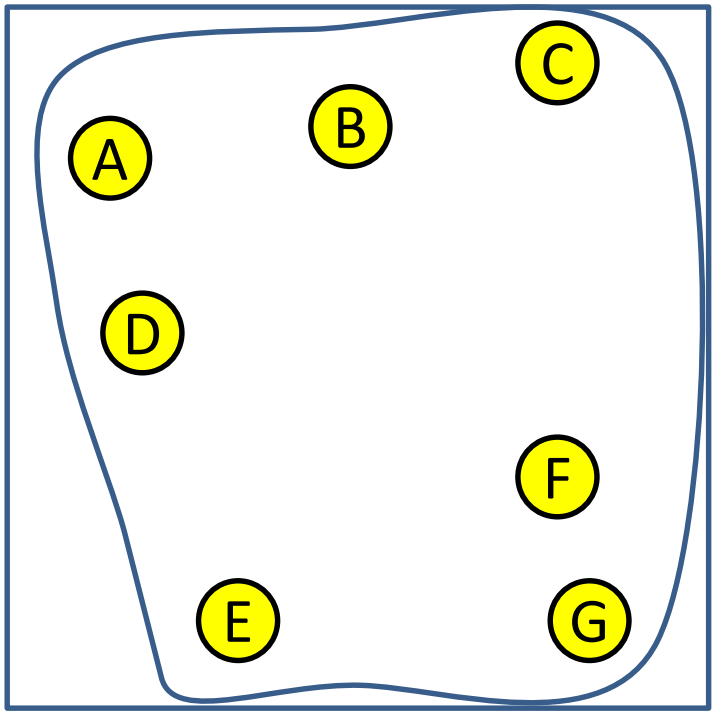


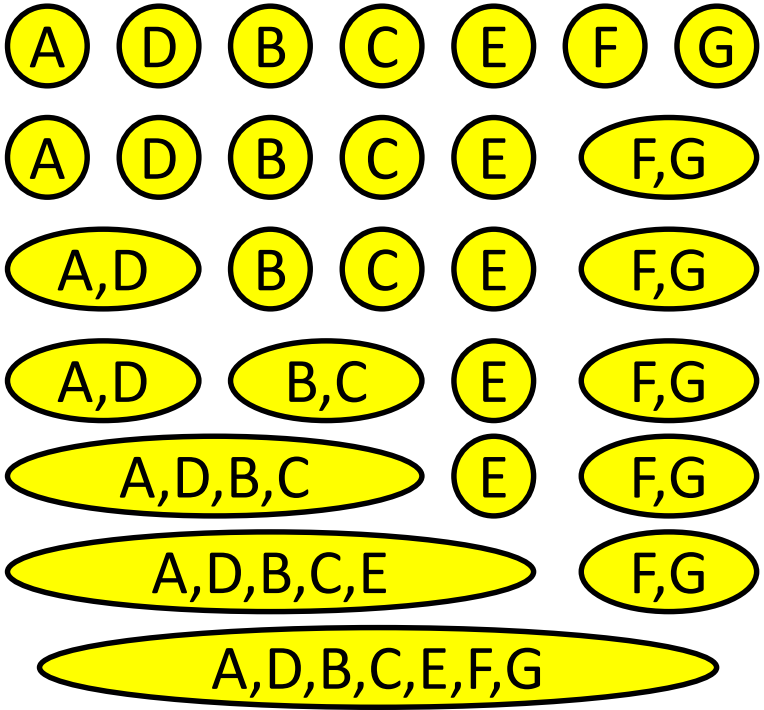
Agglomerative Clustering

- Suppose that we define the distance of two clusters to be the minimum distance between objects from the two clusters.
- Let d be a distance measure between objects in our data.
- The minimum distance $d_{\min}(X, Y)$ between two sets X and Y is defined as:

$$d_{\min}(X, Y) = \min_{x \in X, y \in Y} d(x, y)$$

- Merging $\{A, D, B, C, E\}$ and $\{F, G\}$.





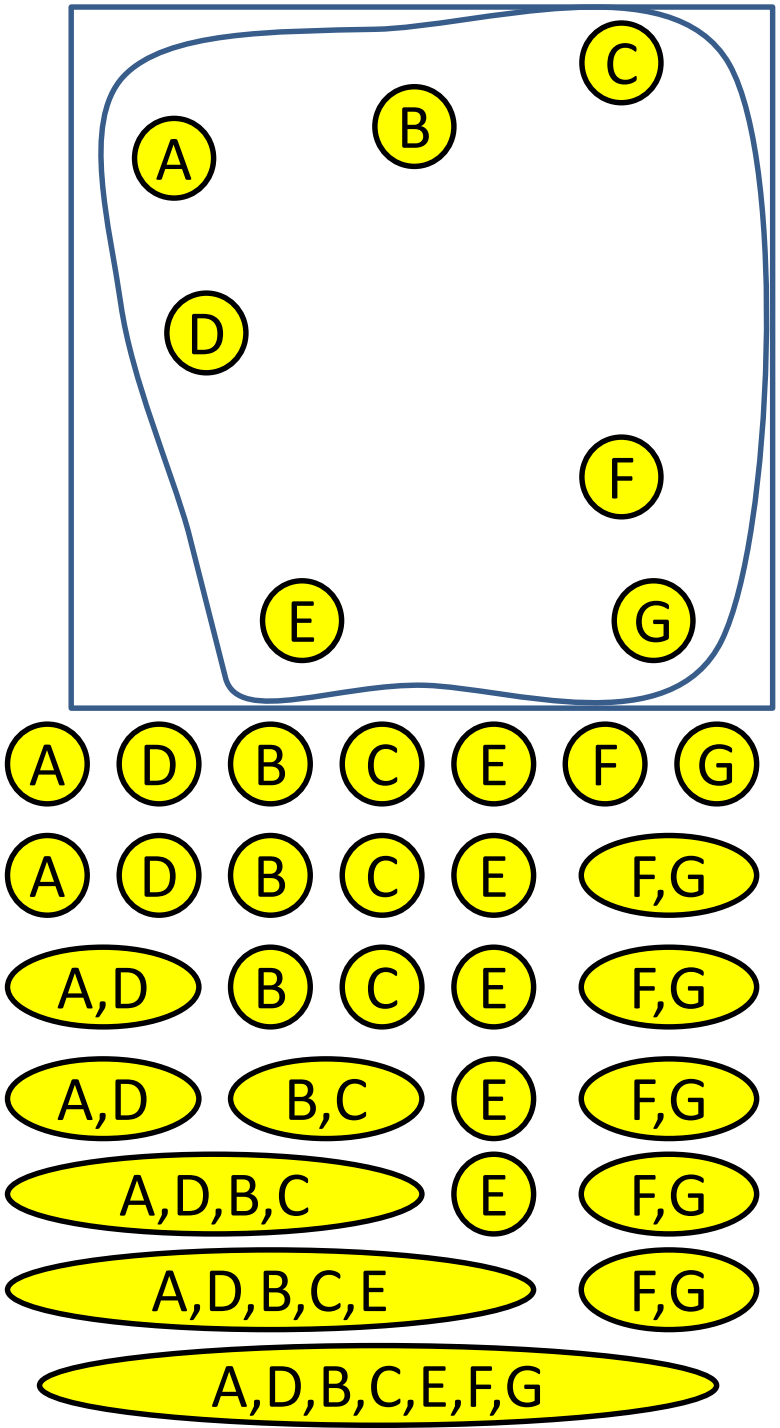
Agglomerative Clustering

- Instead of $d_{\min}$, we could use other measures of distance. For example:

$$d_{\max}(X, Y) = \max_{x \in X, y \in Y} d(x, y)$$

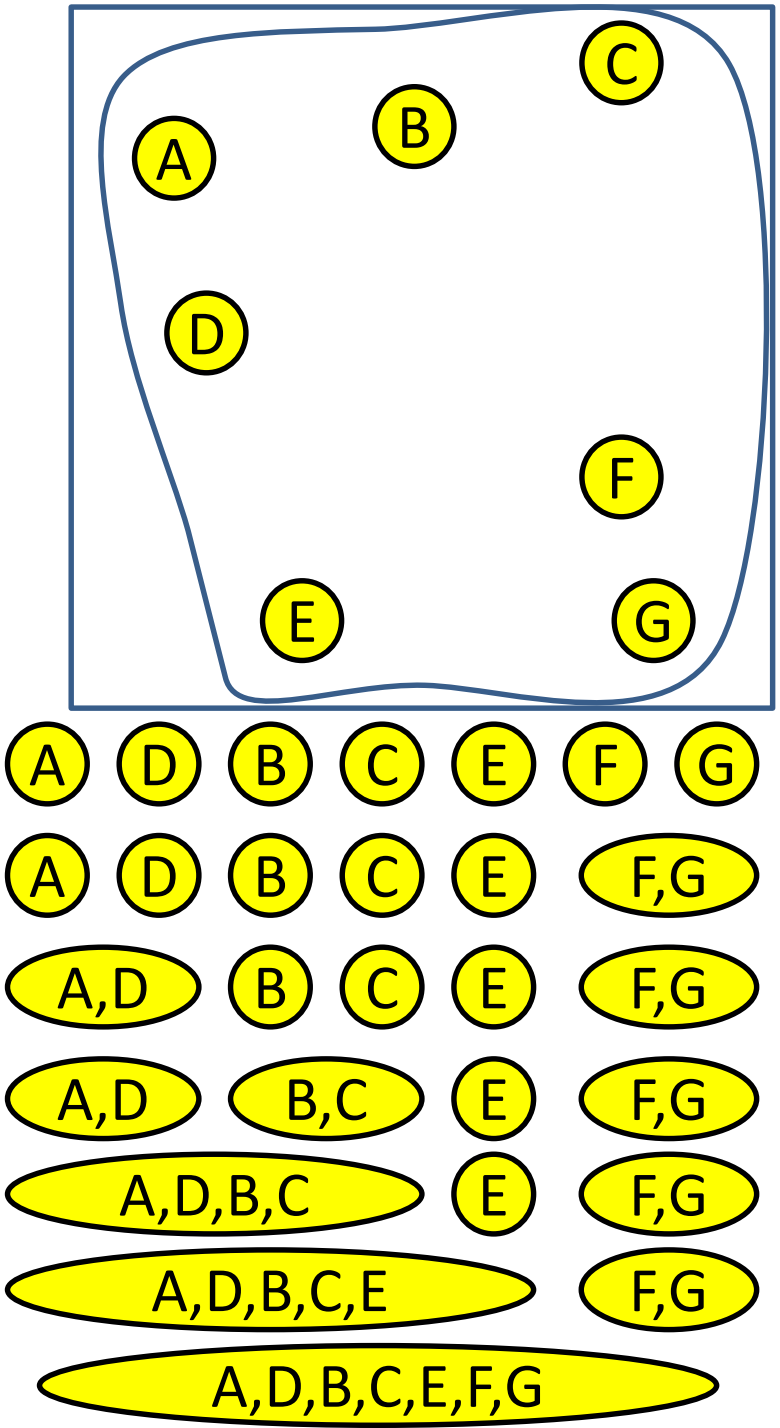
$$d_{\text{mean}}(X, Y) = \text{mean}_{x \in X, y \in Y} d(x, y)$$

- The distance measure (d_{min}, d_{mean}, or $d_{\max}$) is a black box.
  - Agglomerative clustering merges, at each step, the two clusters that are closest to each other according to the chosen distance measure.
  - Different distance measures can lead to different results.



Hierarchical Clustering

- Agglomerative clustering is an example of what we call **hierarchical clustering**.
- In hierarchical clustering, there are different levels of clustering.
  - Each level is obtained by merging, or splitting, clusters from the previous level.
- If we merge clusters from the previous level, we get agglomerative clustering.
- If we split clusters from the previous level, it is called **divisive clustering**.



Clustering - Recap

- The goal in clustering is to split a set of objects into groups of similar objects.
- There is no single criterion for measuring the quality of a clustering result.
- The number of clusters typically needs to be specified in advance.
  - Methods (that we have not seen) do exist for trying to find the number of clusters automatically.
  - In hierarchical clustering (e.g., agglomerative clustering), we do not need to pick a number of clusters.
- A large variety of clustering methods exist.
- We saw a few examples of such methods:
  - K-means, K-medoid, EM, agglomerative clustering.