Unsupervised Learning

Administrivia

- As 2
  - Due Thursday
  - For Q3: in the soft margin case, **\alpha <= C, m = # data points**
  - This was a challenging assignment, As 3 will be lighter

Administrivia

- Project report
  - Due Dec 8
  - Format very similar to the project proposal, but with methods/results/conclusion added
  - Fine to re-use material from your proposal. Self-plagiarism ok in this instance

- Project presentations
  - Dec 1, 6. I will post sign up sheet next week (once I know # projects remain)
  - Graduate students: **there will be questions on your final exam about these presentations**
  - Ugrad students: you will be able to earn bonus points by answering these questions, so I strongly suggest you attend the presentations

Administrivia

- Small change to lecture schedule
  - Two language model lectures
  - Nov 22, 24
  - Guest lecturer Dr. Alex Murphy

Administrivia

- Demystifying grad school event
  - Tomorrow (Wednesday) Nov 16, 5-7pm
  - Register: https://forms.gle/1K62BmkHtN6fHKRF7
  - https://ualberta-ca.zoom.us/j/95723954564?pwd=SDNrNStPYUZLMmVXTXJpTSt6bFpldz09
  - Meeting ID: 957 2395 4564
  - Passcode: 384328

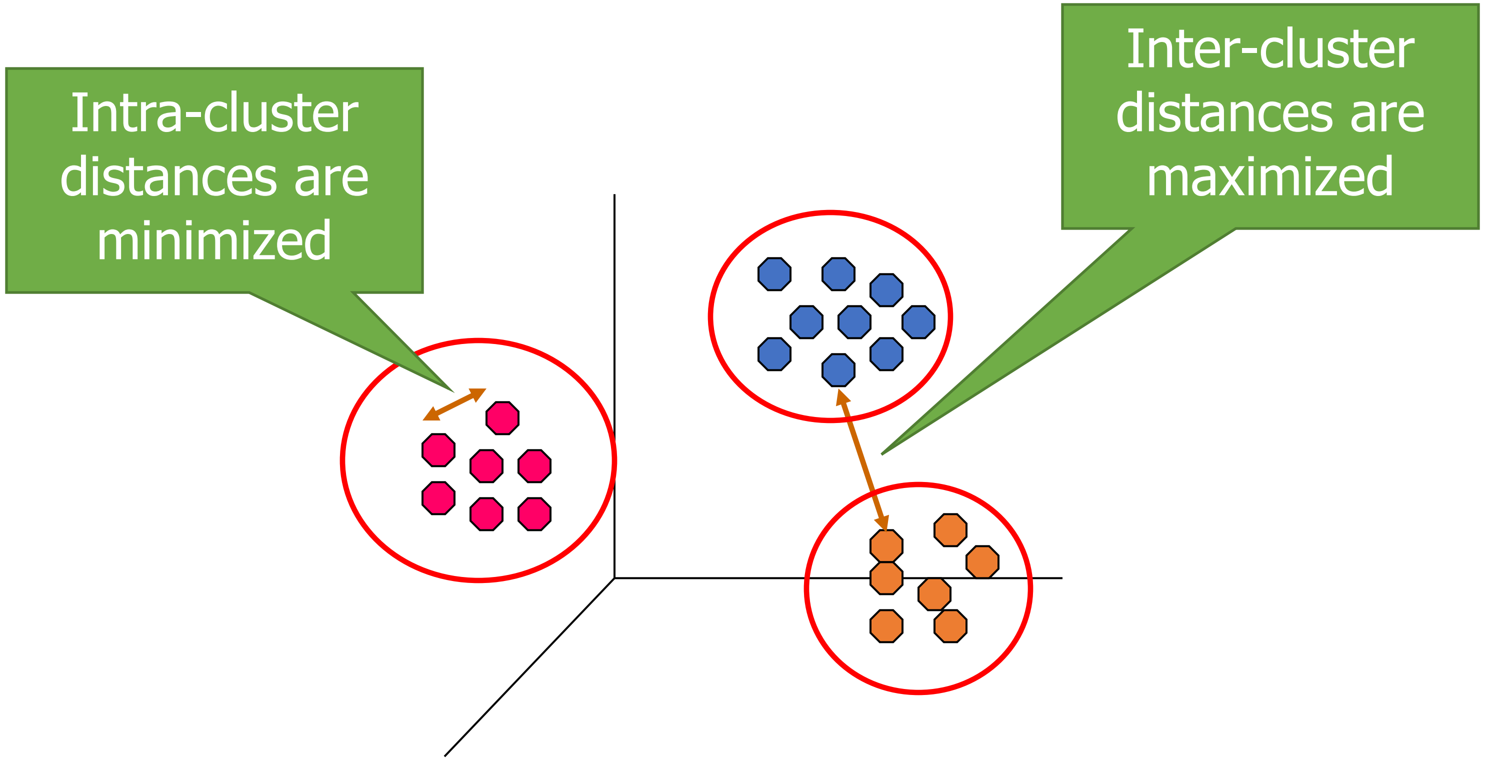
Unsupervised learning

- Clustering: assigning groupings to objects, without knowing the ground truth
  - These assigned groups/classes are unnamed
  - Don’t have meaning *a priori*
  - May not be meaningful, but can give you a place to start, and/or be the basis for further analysis.

Clustering

What is Clustering?

- Finding groups of objects
  - such that the objects in a group will be **similar** to one another and
  - **dissimilar** from the objects in other groups



Applications

- Numerous!!
  - https://en.wikipedia.org/wiki/Cluster_analysis#Applications
- Any problem where you would like to find groups of items/objects

- Cluster movies to create genres
- Cluster brain images to find groups of people
- Cluster credit card users to find anomalies
- etc…

Solution Format

- Finding an optimal clustering clustering is a combinatorial problem (i.e. really expensive to ensure a correct answer), but we can use heuristics

- A solution would be an assignment for each data point to a cluster. Say we have 10 data points and 3 clusters:

  0 0 0 1 1 1 2 2 2 0

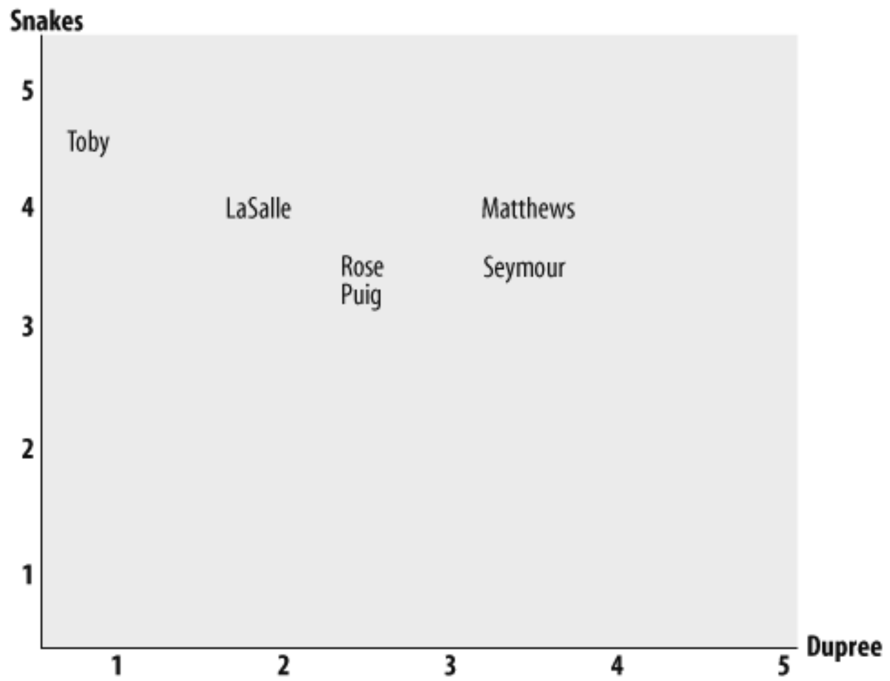
- How can we measure the goodness of a solution?
  - To keep it simple, let's just consider intra-cluster distance
  - Each cluster i has a mean point, called the centroid μ_i

Tools

- In order to cluster, we need to define intra- and inter-cluster distances
  - We will need either similarity or dissimilarity measures
    - -1*similarity can be used as dissimilarity

Finding Similar Datapoints (ex. Netflix users)

- Simple way to calculate a similarity score is to use Euclidean distance, which considers the items that people have ranked in common.



People in preference space (assuming two movies)

Euclidean Distance

- Suppose we have two vectors of ratings:

$$\mathbf{x} = \left[x_1, \ldots, x_m\right]$$
$$\mathbf{y} = \left[y_1, \ldots, y_m\right]$$

- Then we can measure their similarity by the Euclidean distance:

$$sim_{\mathbf{x},\mathbf{y}} = \sqrt{\sum_{i=1}^{m}\left(x_i - y_i\right)^2}$$

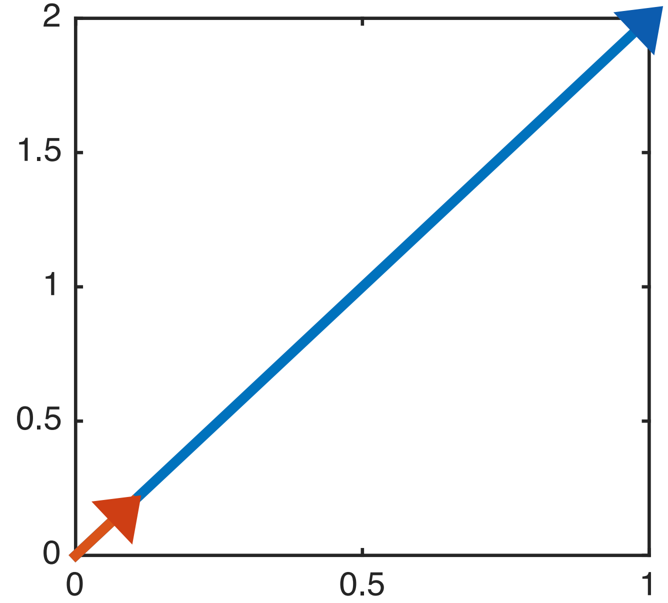
Problem with Euclidean Distance

E.g.,

- suppose a critic rated five movies by 1, 2, 3, 5, 8.
- and another critic rated those movies by .01, .02, .03, .05, .08.

- Obviously, they are quite similar in the relative tastes, yet their Euclidean distance is big!

Plot of the first 2 dims



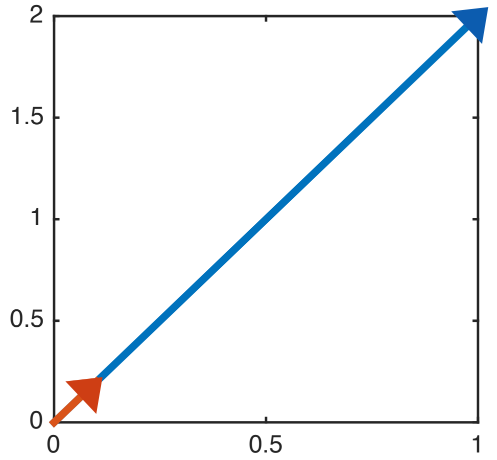
Fix

E.g.,

- suppose a user rated five movies by 1, 2, 3, 5, 8.
- and another user rated those movies by .01, .02, .03, .05, .08.

- We can consider vectors
  x=[1, 2, 3, 5, 8] and **y**=[.01, .02, .03, .05, .08] and see that
  the **angle** (theta) between them is 0 degrees,
  i.e. they point in the same direction.

- So, we can employ the **cosine** of theta:
  - the greater the cosine, the closer to 0 degrees theta is,
  - i.e. the more similar the two rating vectors are,
  - i.e. the more similar the users are.



$$sim_{\mathbf{x},\mathbf{y}} = \cos\theta = \frac{\mathbf{x}\cdot\mathbf{y}}{\|\mathbf{x}\|\|\mathbf{y}\|} = \frac{1.03}{\sqrt{103}\sqrt{0.0103}} = 1$$

The greatest value **cos** can have, which happens for theta=0.

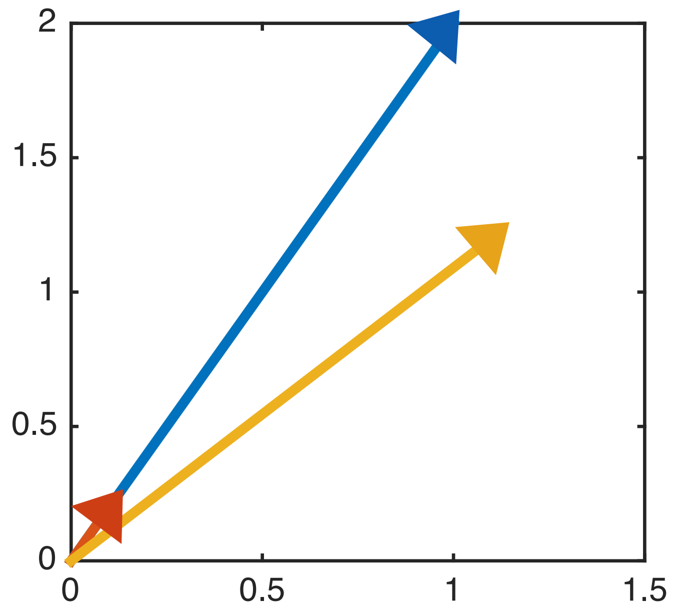
However…

E.g.,

- suppose a user rated five movies by 1, 2, 3, 5, 8.
- and another user rated those movies by .01, .02, .03, .05, .08.

- Now suppose the second user, seeing he has been too harsh, increases by 0.1 all his ratings (now the yellow line). So, the vectors are now

  x=[1, 2, 3, 5, 8] and **y**=[.11, .12, .13, .15, .18]



*This image is the first two ratings only, but gives the general idea

However…

- They are still very similar, but cosine similarity will get confused:

$$sim_{\mathbf{x},\mathbf{y}} = \cos\theta = \frac{\mathbf{x}\cdot\mathbf{y}}{\|\mathbf{x}\|\|\mathbf{y}\|} = \frac{2.93}{\sqrt{103}\sqrt{0.0983}} = 0.92$$

- Their similarity is reduced, while it should have been (intuitively) invariant.

Better Fix: Pearson Correlation

- Recall the vectors are:
  x=[1, 2, 3, 5, 8] and **y**=[.11, .12, .13, .15, .18]

- First **centralize** by subtracting their mean, then compute cosine similarity.
- $m_{\mathbf{x}}$ = (1 + 2 + 3 + 5 + 8)/5 = 3.8
- $m_{\mathbf{y}}$ = (.11 + .12 + .13 + .15 + .18)/5 = .138

x’=[1-3.8, 2-3.8, 3-3.8, 5-3.8, 8-3.8] =
[−2.8, −1.8, −0.8, 1.2, 4.2]
y’=[.11-.138, .12-.138, .13-.138, .15-.138, .18-.138] =
[−0.028, −0.018, −0.008, 0.012, 0.042]

$$sim_{\mathbf{x},\mathbf{y}} = \cos\theta = \frac{\mathbf{x}'\cdot\mathbf{y}'}{\|\mathbf{x}'\|\|\mathbf{y}'\|} = \frac{0.308}{\sqrt{30.8}\sqrt{0.00308}} = 1$$

as we intuitively expect.

This is the Pearson Correlation Coefficient.

Pearson Correlation Formula

$$\mathbf{x} = \left[x_1, \ldots, x_m\right]$$
$$\mathbf{y} = \left[y_1, \ldots, y_m\right]$$

- Formula:

$$sim_{\mathbf{x},\mathbf{y}} = \frac{\sum_{i=1}^{m}\left(x_i - \bar{x}\right)\cdot\left(y_i - \bar{y}\right)}{\sqrt{\sum_{i=1}^{m}\left(x_i - \bar{x}\right)^2}\cdot\sqrt{\sum_{i=1}^{m}\left(y_i - \bar{y}\right)^2}}$$

Similarity and Dissimilarity

- **Similarity**
  - Numerical measure of how alike two data objects are.
  - Higher when objects are more alike.

- **Dissimilarity (Distance)**
  - Numerical measure of how different are two data objects
  - Lower when objects are more alike

Similarity/Dissimilarity for Simple Attributes

p and *q* are the attribute values for two data objects.

- **Nominal**
  - E.g. province attribute of an address with values:
    {BC, AB, ON, QC, …}
    Order not important.
  - **Dissimilarity**
    d=0 if p=q
    d=1 if $p \neq q$

- **Can also convert to one-hot vectors**
- **How will one-hot vectors work with Euclidean, cosine, correlation?**

Similarity Between Binary Vectors

- Common situation is that objects, **p** and **q**, have only binary attributes
- Compute similarities using the following quantities
  - M_{01} = the number of attributes where **p** was 0 and **q** was 1
  - M_{10} = the number of attributes where **p** was 1 and **q** was 0
  - M_{00} = the number of attributes where **p** was 0 and **q** was 0
  - M_{11} = the number of attributes where **p** was 1 and **q** was 1

- Simple Matching and Jaccard Coefficients
- SMC = number of matches / number of attributes
- = $(M_{11} + M_{00}) / (M_{01} + M_{10} + M_{11} + M_{00})$

- J = number of M_{11} matches / number of not-both-zero attributes values
- = $(M_{11}) / (M_{01} + M_{10} + M_{11})$
- = intersection / union

SMC versus Jaccard: Example

p = 1 0 0 0 0 0 0 0 0 0

q = 0 0 0 0 0 0 1 0 0 1

$M_{01} = 2$ (the number of attributes where p was 0 and q was 1)

$M_{10} = 1$ (the number of attributes where p was 1 and q was 0)

$M_{00} = 7$ (the number of attributes where p was 0 and q was 0)

$M_{11} = 0$ (the number of attributes where p was 1 and q was 1)

$\text{SMC} = (M_{11} + M_{00})/(M_{01} + M_{10} + M_{11} + M_{00}) = (0+7) / (2+1+0+7) = 0.7$

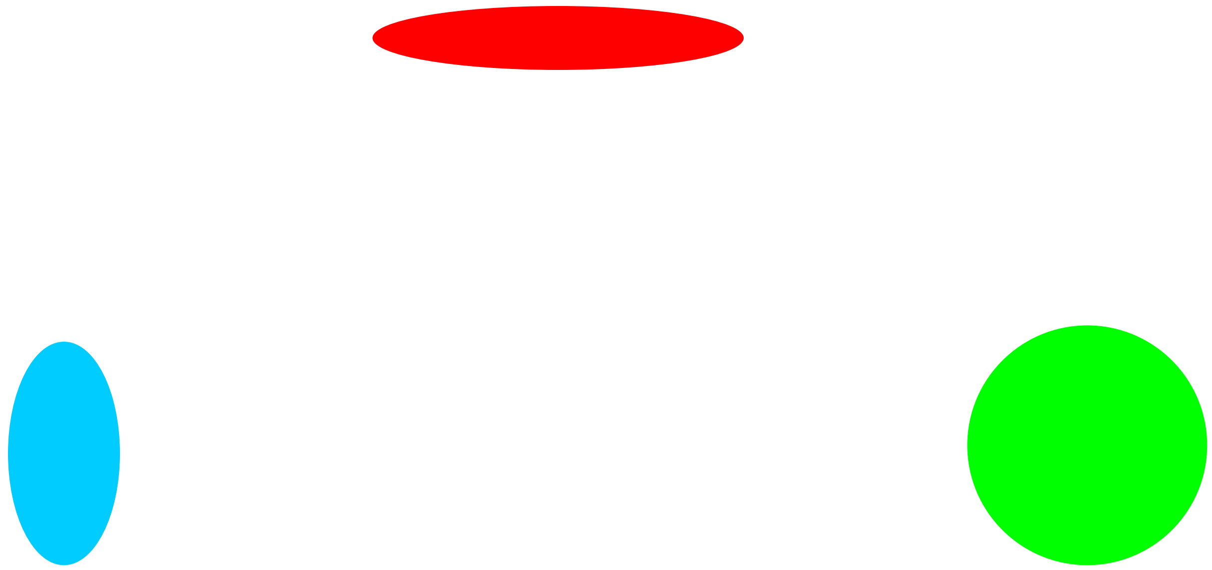
SMC can be inflated when the number of expected 0s is large… like in one-hot vectors!

$J = (M_{11}) / (M_{01} + M_{10} + M_{11}) = 0 / (2 + 1 + 0) = 0$

Types of Clusters in Practice

Types of Clusters: Well-Separated

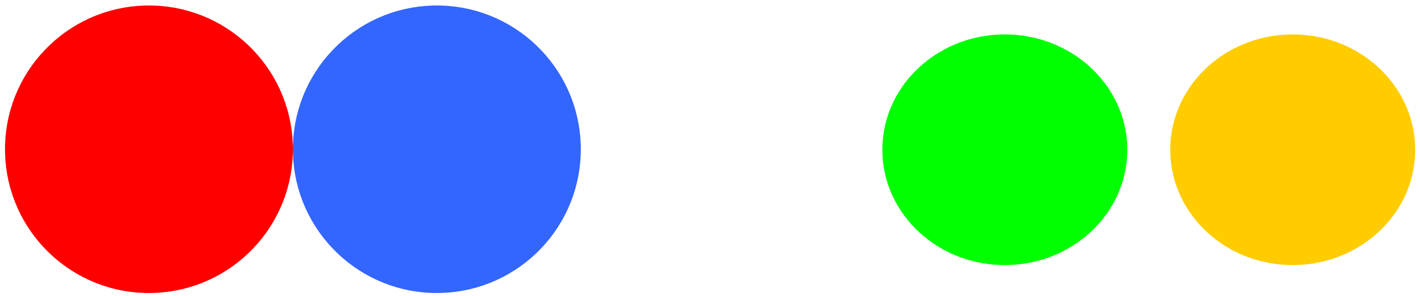
- Well-Separated Clusters:
  - Any point in a cluster is closer (or more similar) **to every other point** in the cluster than to any point not in the cluster.



3 well-separated clusters

Types of Clusters: Center-Based

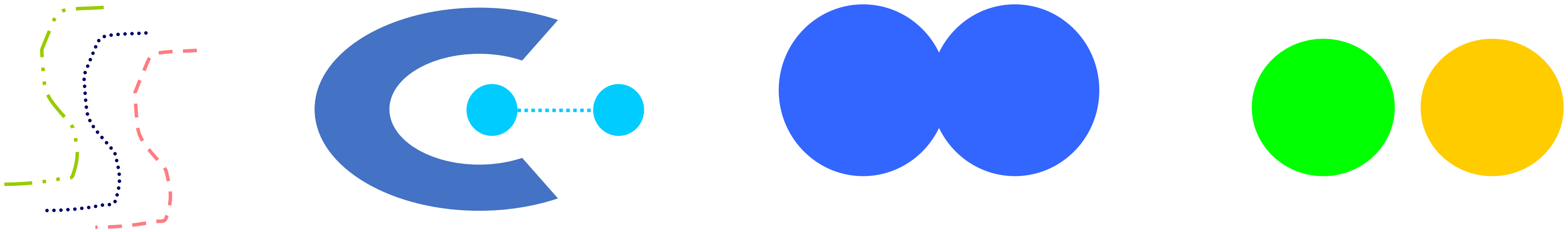
- Center-based
  - An object in a cluster is **closer** (more similar) **to the “center”** of a cluster, **than to the center of any other cluster**
    - The center of a cluster is often a centroid, the average of all the points in the cluster, or a medoid, the most “representative” point of a cluster



4 center-based clusters

Types of Clusters: Contiguity-Based

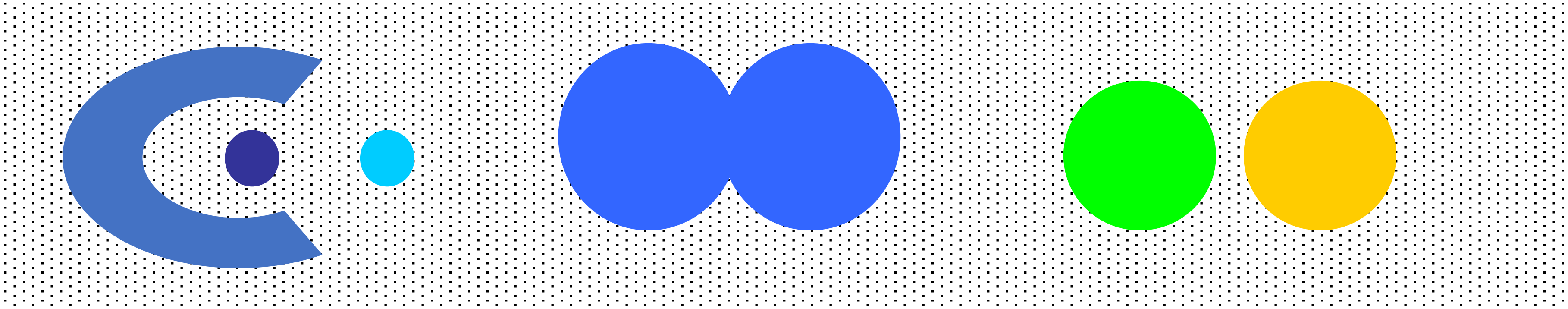
- Contiguous Cluster (Nearest neighbor or Transitive)
  - A point in a cluster is **closer** (or more similar) **to one** or more other points in the cluster **than to any point not in the cluster**.



8 contiguous clusters

Types of Clusters: Density-Based

- Density-based
  - A cluster is a **dense** region of points, which is **separated by low-density** regions, from other regions of high density.
  - Density: the number of points per space unit



6 density-based clusters

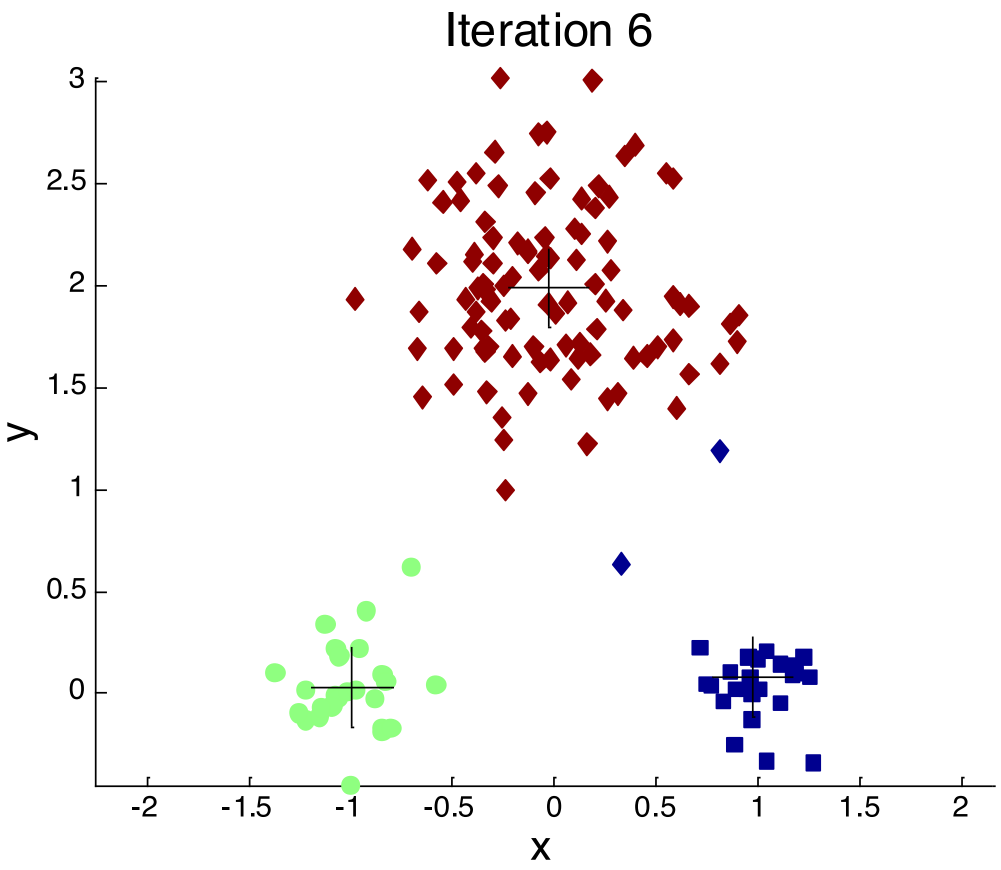
Algorithms

K-means Clustering

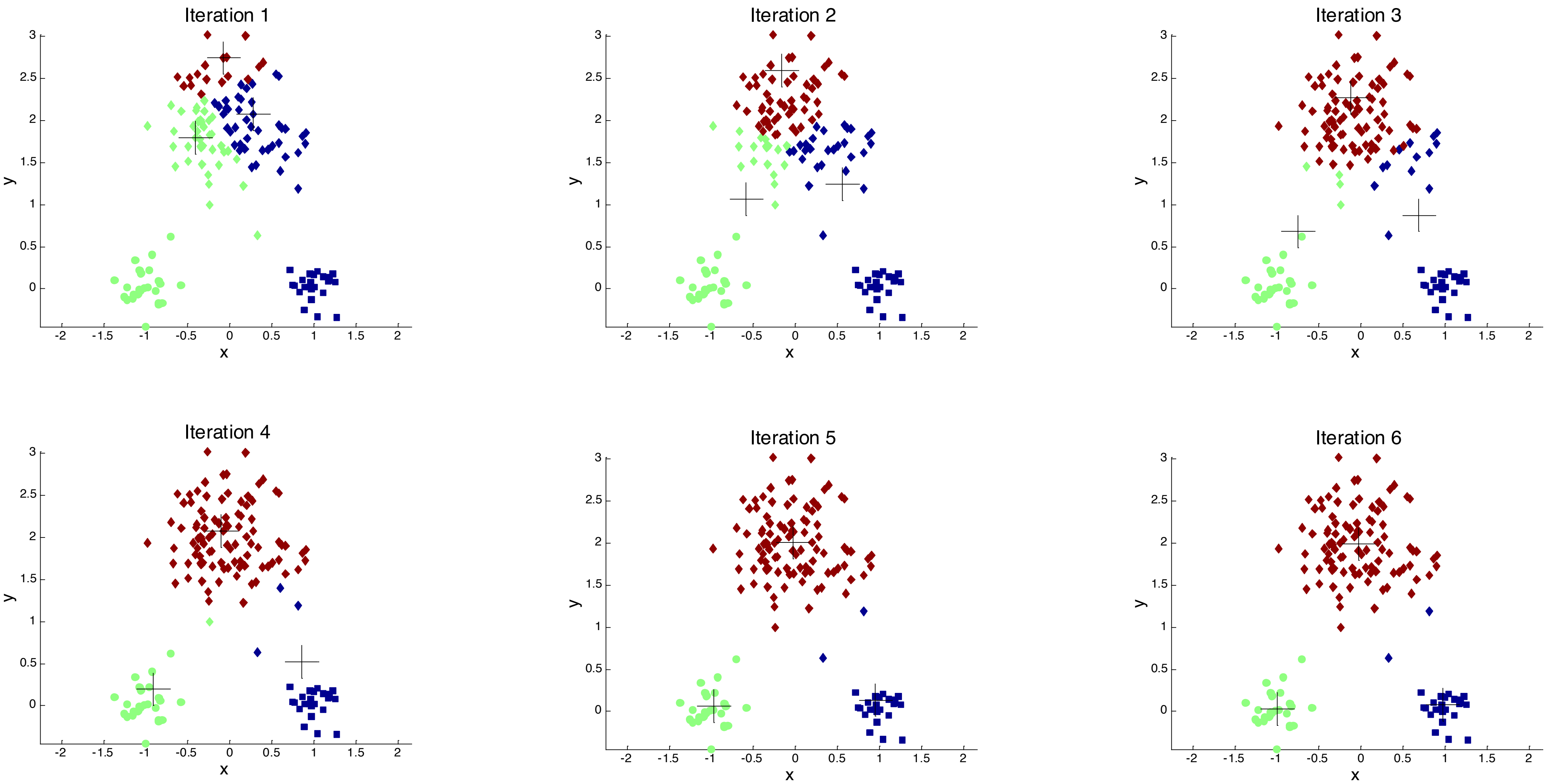
- Each cluster is associated with a centroid (center point)
- Each point is assigned to the cluster with the closest centroid
- Number of clusters, *K*, must be specified
- Basic algorithm is very simple

Select K points as the initial centroids.
repeat
    Form K clusters by assigning all points to the closest centroid.
    Recompute the centroid of each cluster.
until The centroids don't change

Example



Example (without animations)



K-means Clustering – Details

- Initial centroids may be chosen randomly.
    - Clusters produced vary from one run to another.
    - Rerun several times and pick the clustering with the smallest SSE (see next slide).
- The centroid is (typically) the mean of the points in the cluster.
- ‘Closeness’ is measured by Euclidean distance, cosine similarity, etc.
- Most of the convergence happens in the first few iterations.
    - Often the stopping condition is changed to ‘Until relatively few points change clusters’

Evaluating K-means Clusters

- Most common measure is Sum of Squared Error (SSE)
  - For each point, the error is the distance to the nearest centroid
  - To get SSE, we square these errors and sum them up.

$$SSE = \sum_{i=1}^{K} \sum_{x \in C_i} [dist(\mu_i, x)]^2$$

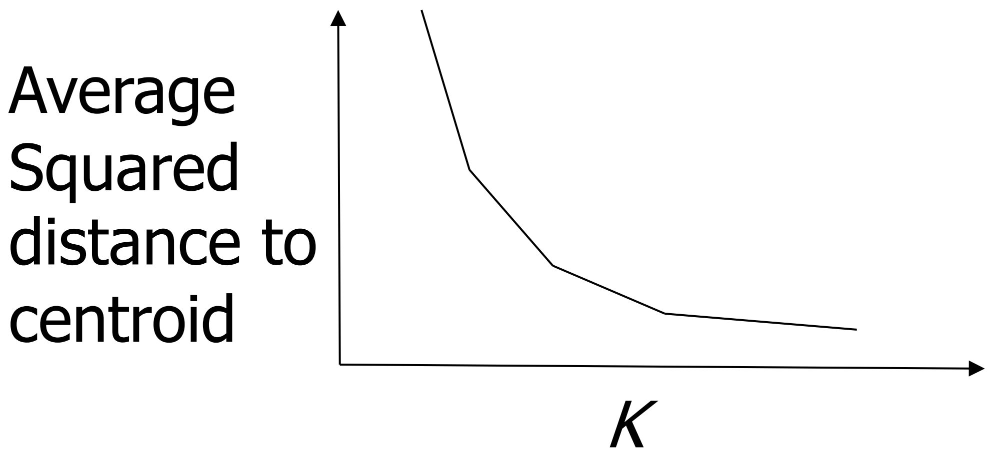
x is a data point in cluster C_i and

μ_i is the centroid for cluster C_i

How to choose K?

$$SSE = \sum_{i=1}^{K} \sum_{x \in C_i} [dist(\mu_i, x)]^2$$

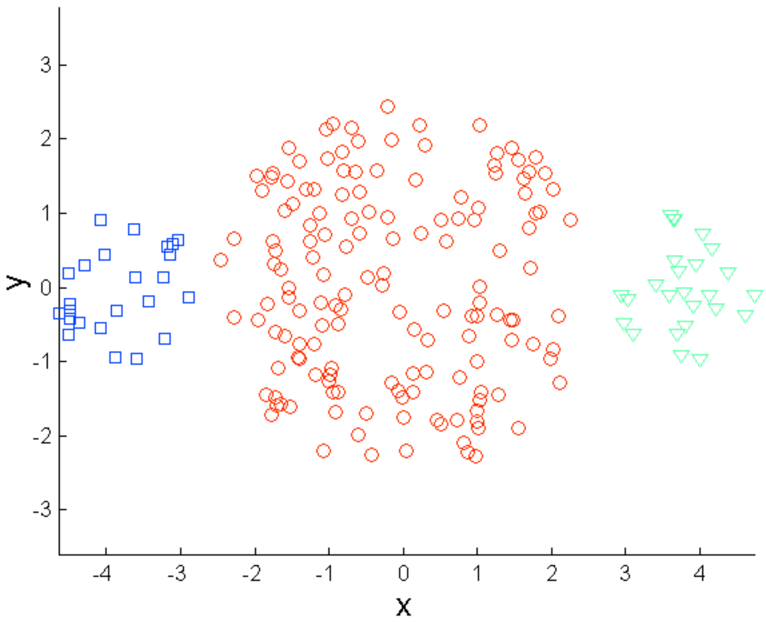
- As K grows, SSE will fall.
  - How low can SSE be?
  - For what setting of K will you have that SSE?
- So how to choose K?
- Try different *K*, looking at the change in the average distance to centroid, as *K* increases.
- Average falls rapidly until “right” *K*, then changes little.
  - looks like an “elbow” in the graph



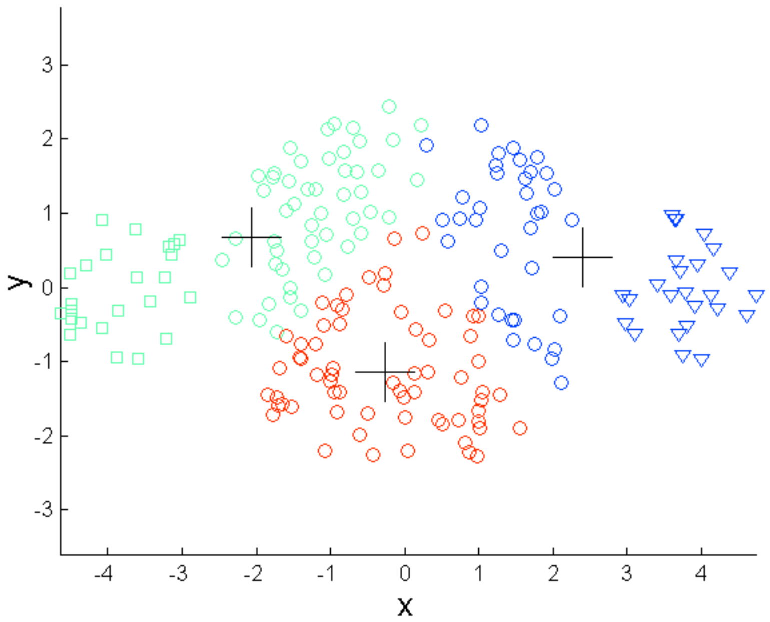
Limitations of K-means

- K-means has problems when (the real) clusters are of
  - Differing Sizes
  - Differing Densities
  - Non-globular shapes

Limitations of K-means: Differing Sizes

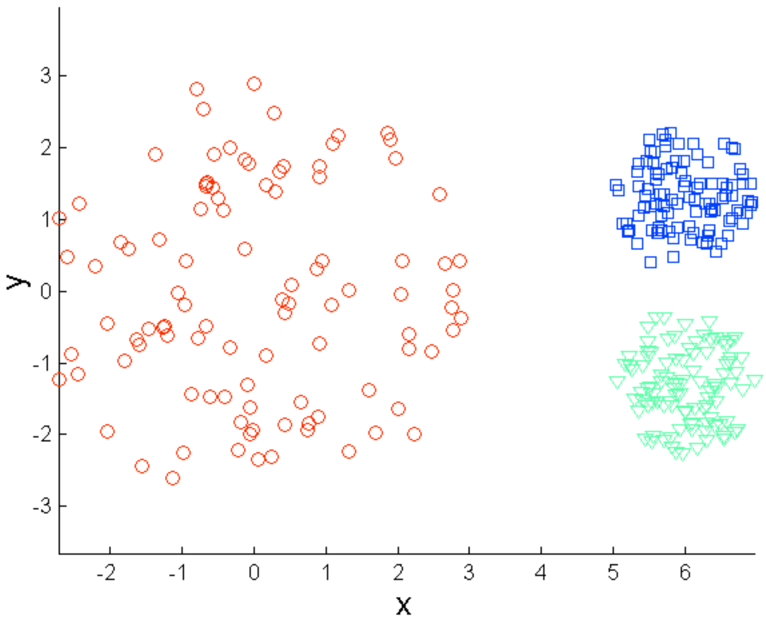


Original Points

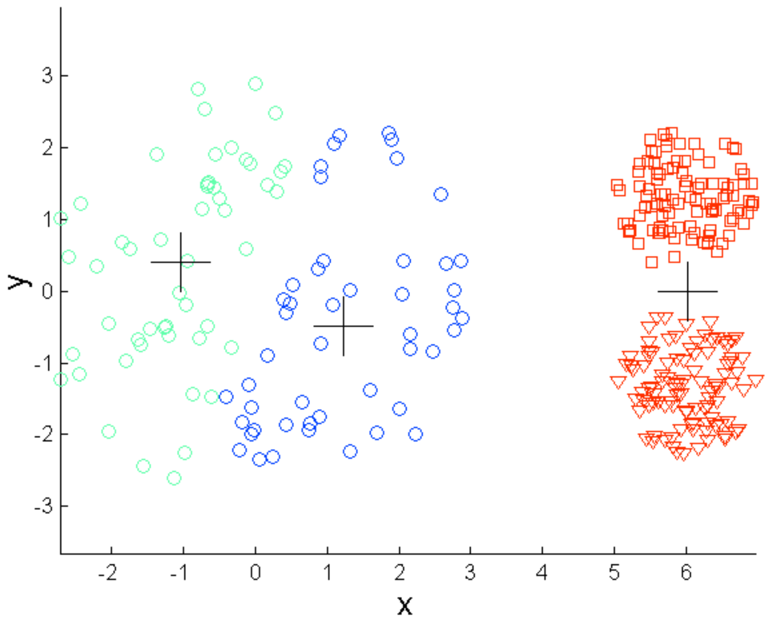


K-means (3 Clusters)

Limitations of K-means: Differing Density

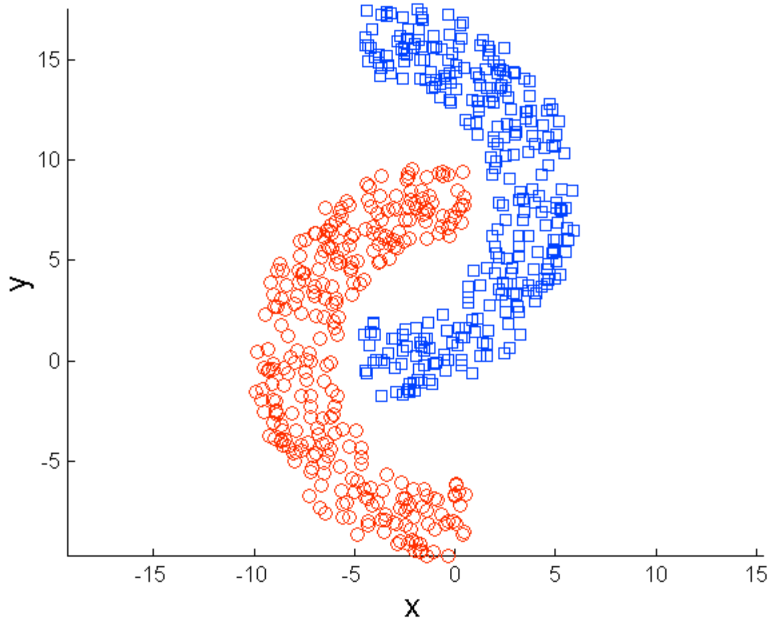


Original Points

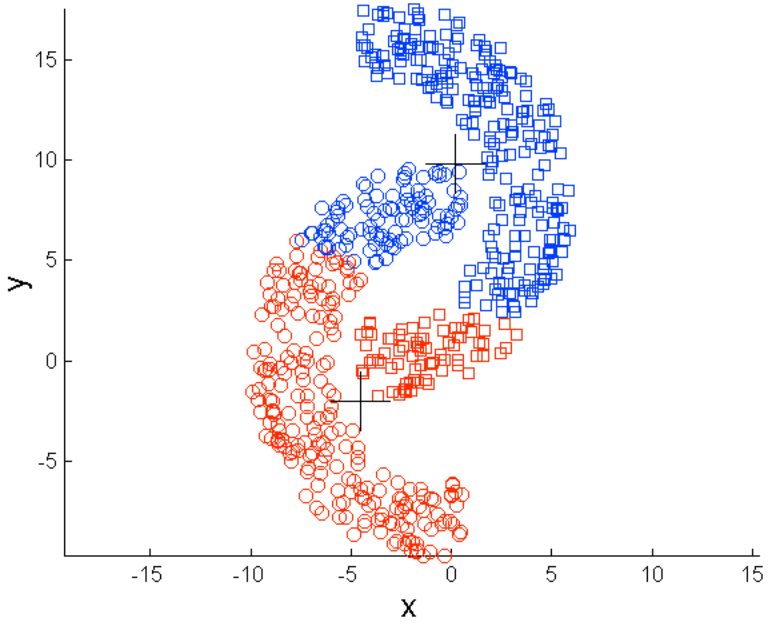


K-means (3 Clusters)

Limitations of K-means: Non-globular Shapes

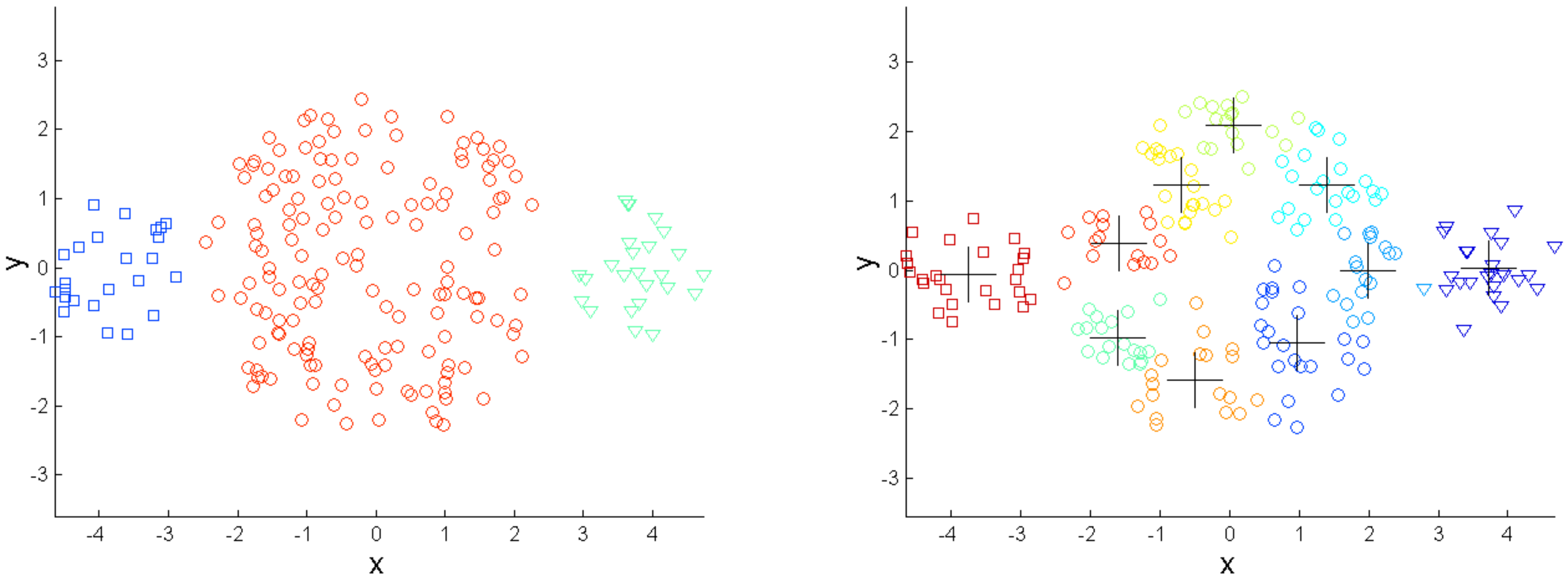


Original Points



K-means (2 Clusters)

Overcoming K-means Limitations



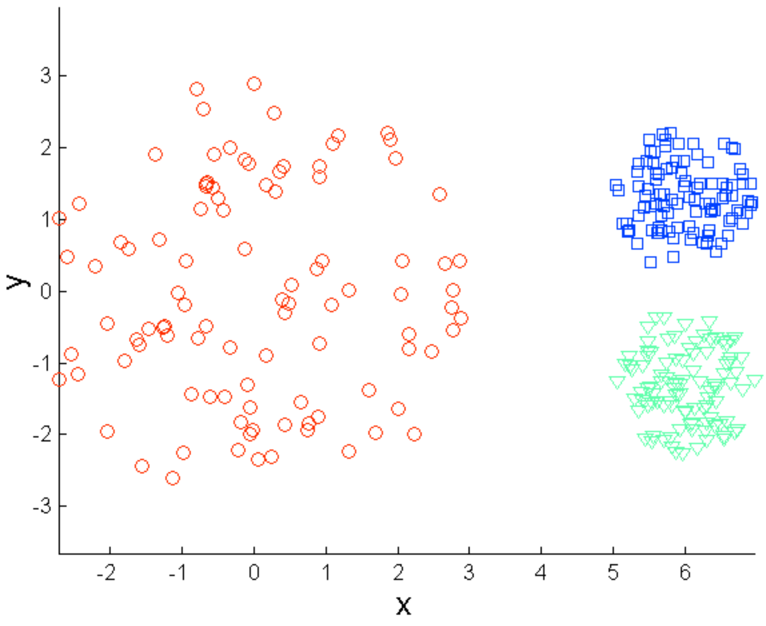
Original Points **K-means Clusters**

One solution is to use many clusters.

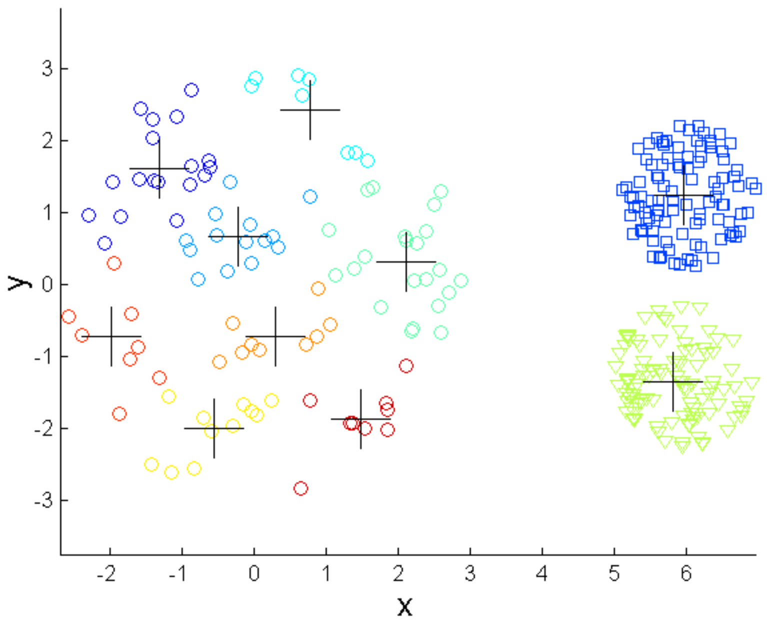
Find parts of clusters.

Apply **merge** strategy (merge clusters that would cause the least increase in SSE)

Overcoming K-means Limitations

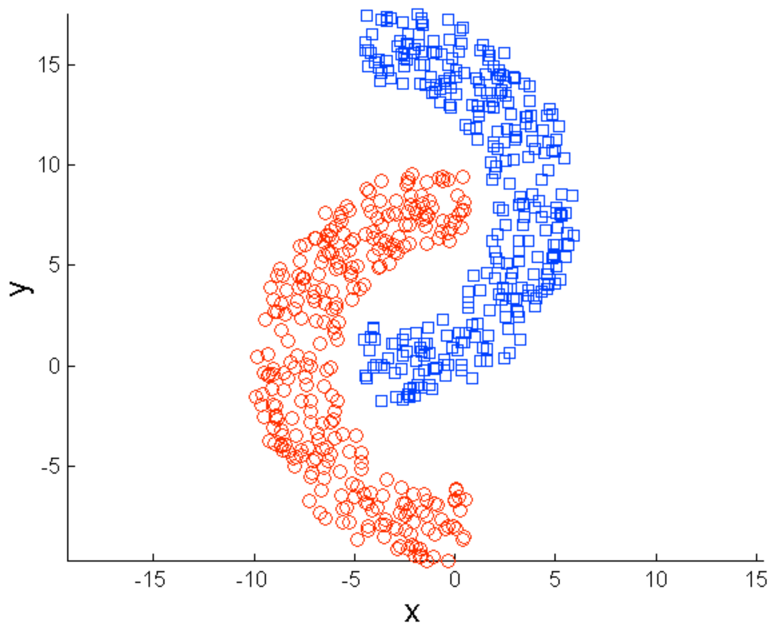


Original Points

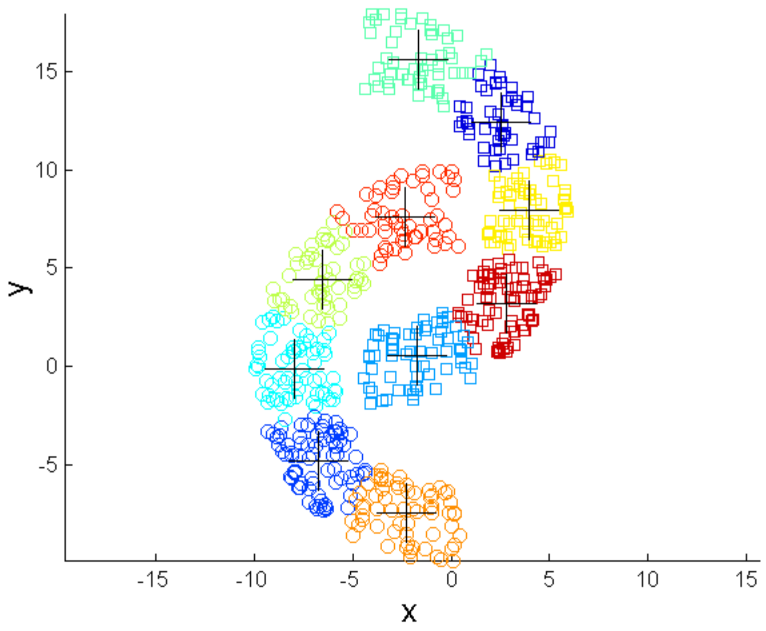


K-means Clusters

Overcoming K-means Limitations

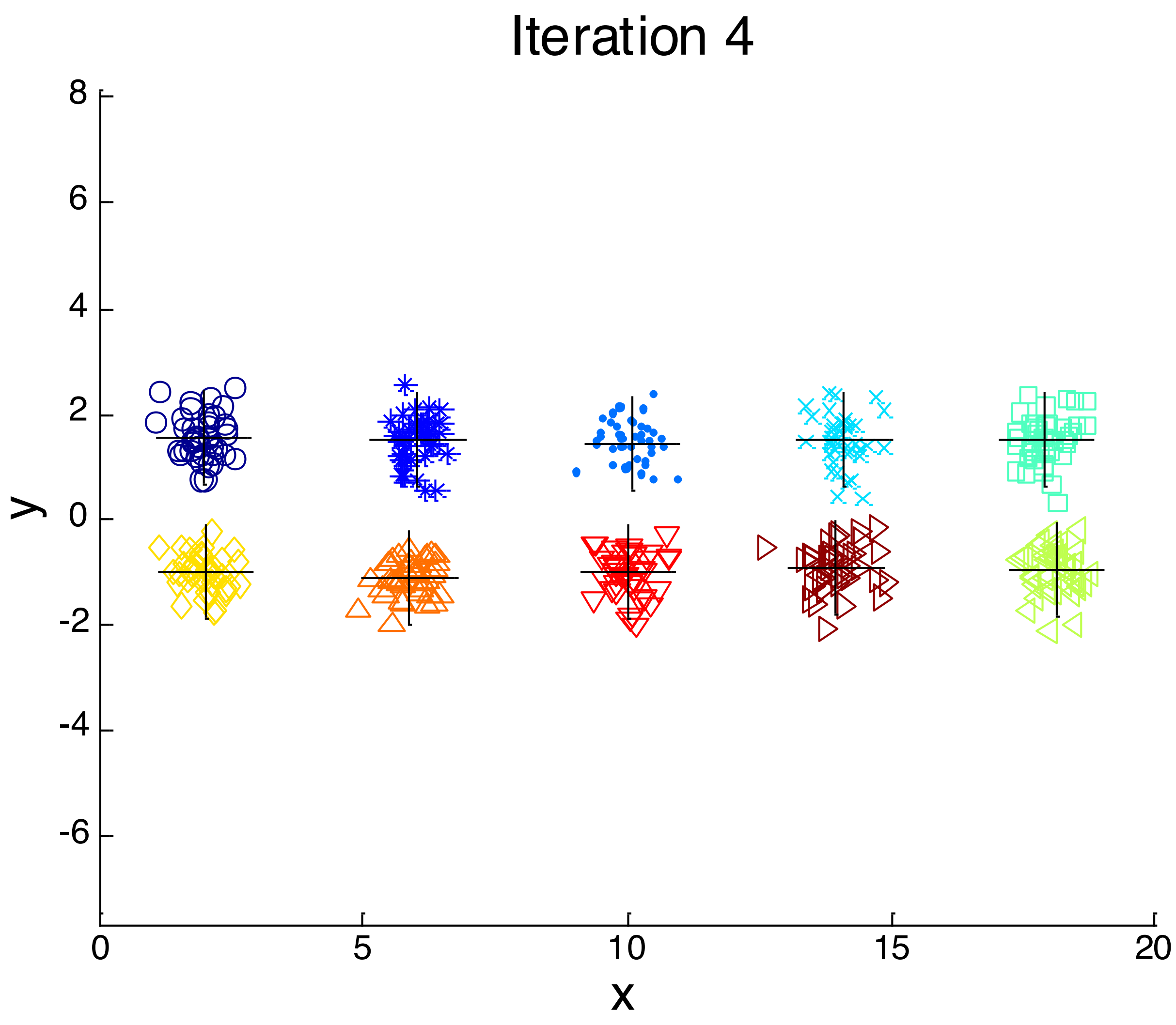


Original Points



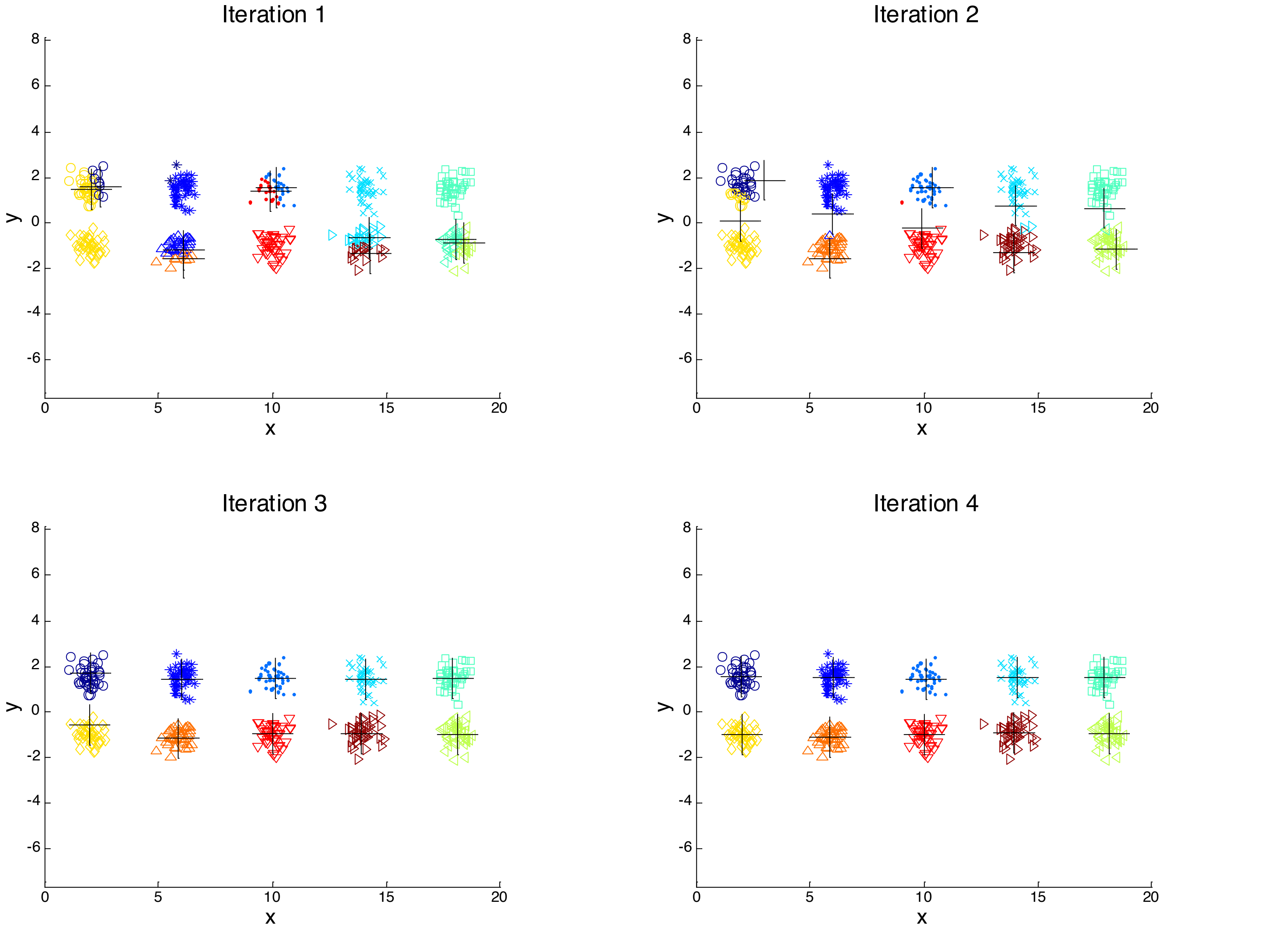
K-means Clusters

Importance of Choosing Initial Centroids



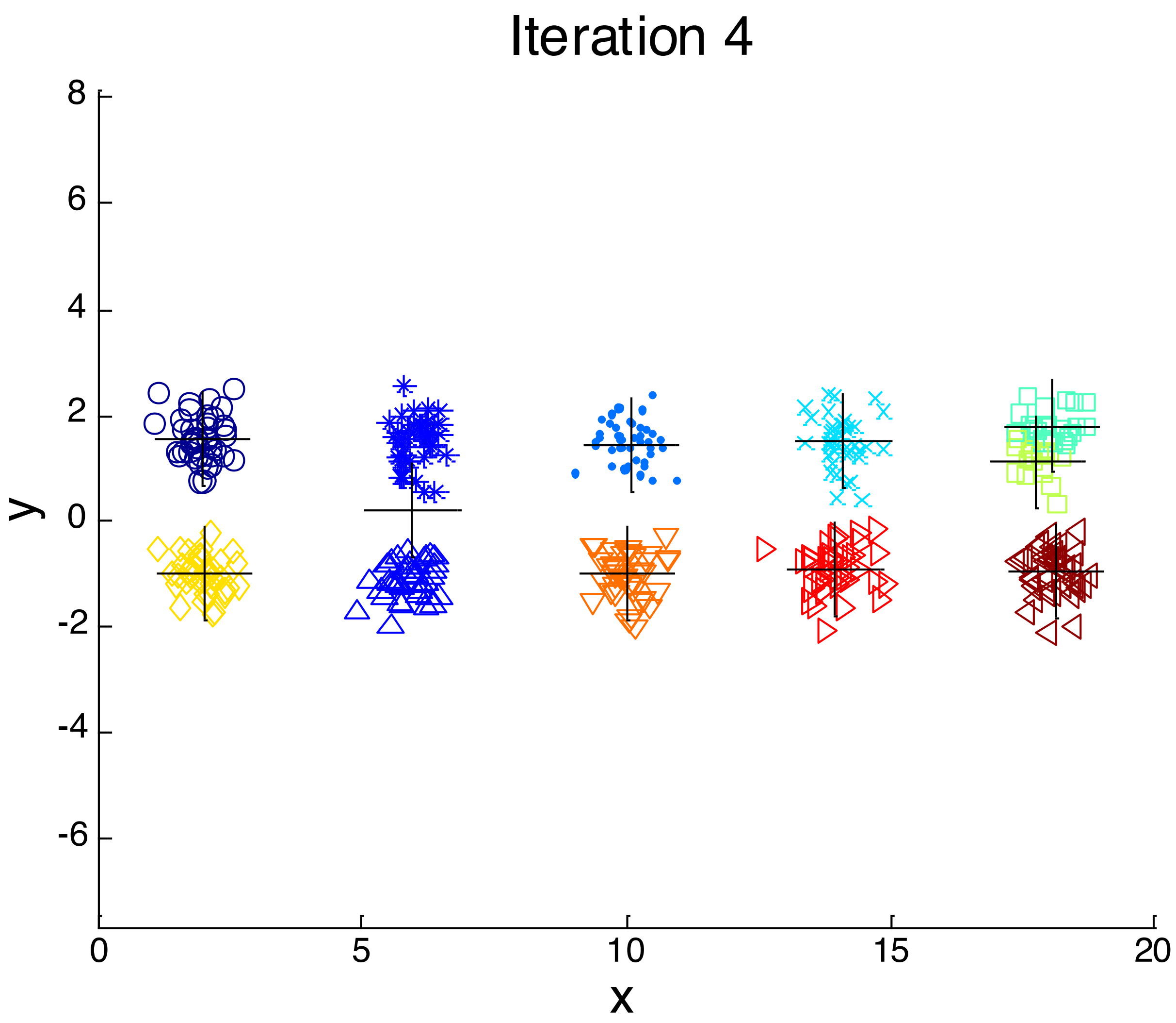
Starting with two initial centroids in one cluster of each pair of clusters

(no animation)



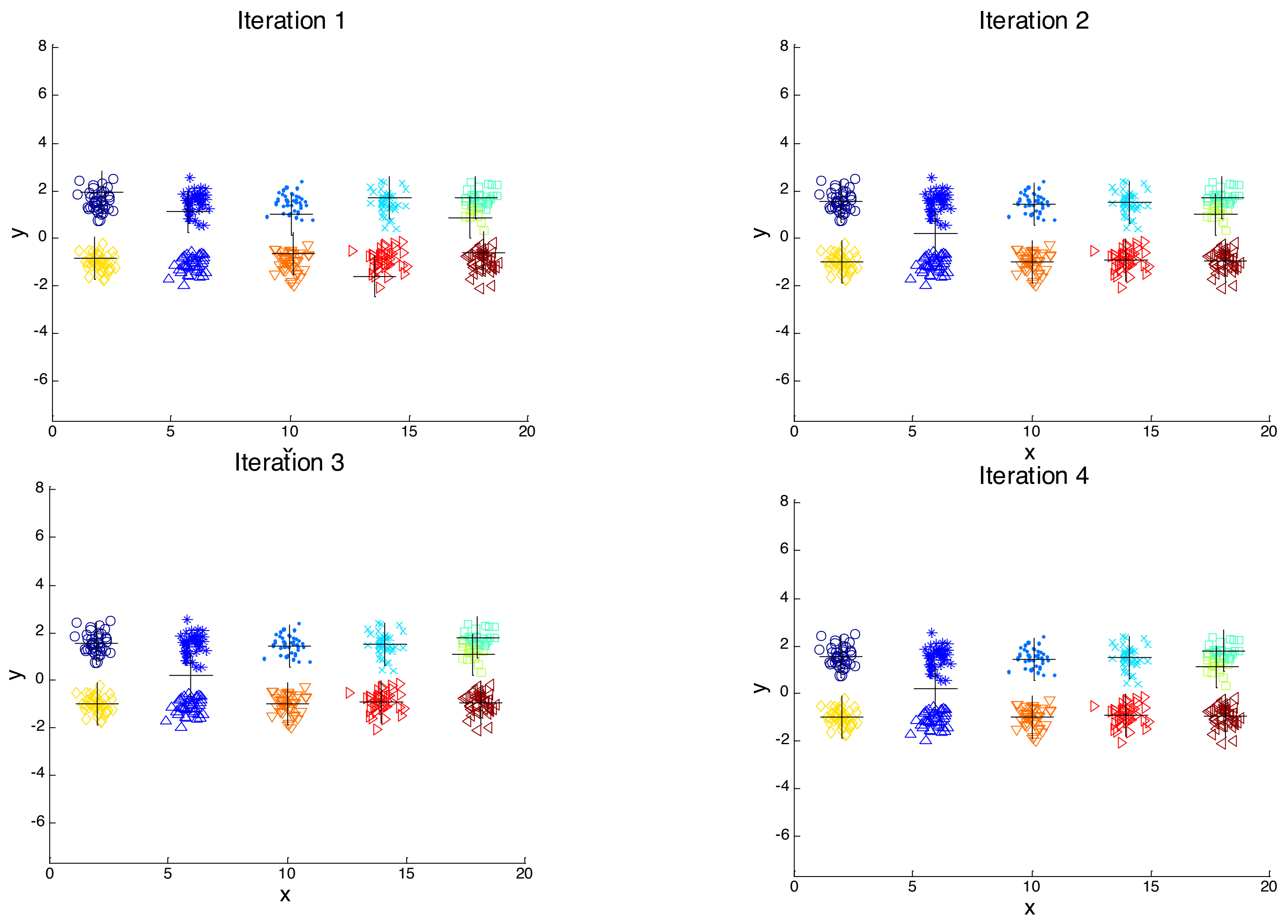
Starting with two initial centroids in one cluster of each pair of clusters

Importance of Choosing Initial Centroids



Starting with some pairs of clusters having three initial centroids, while other have only one.

Importance of Choosing Initial Centroids



Starting with some pairs of clusters having three initial centroids, while other have only only one.

Problems with Selecting Initial Points

- The ideal would be to choose initial centroids, one from each true cluster. However, this is very difficult.
  - indeed if we could do this we would know the clusters already!
- If there are *K* 'real' clusters, then the chance of selecting one centroid from each cluster is small.
  - Chance is relatively small when *K* is large
    - E.g. If clusters are the same size, *n*, then

$$P = \frac{\text{number of ways to select one centroid from each cluster}}{\text{number of ways to select } K \text{ centroids}} = \frac{K!n^K}{(Kn)^K} = \frac{K!}{K^K}$$

- For example, if $K = 10$, then *probability* $= 10!/10^{10} = 0.00036$ 😬

Solutions to Initial Centroids Problem

- Multiple runs
  - Helps, but probability is not on your side

- Bisecting K-means
  - Not as susceptible to initialization issues
  - Straightforward extension of the basic Kmeans algorithm. Simple idea:

    First, split the set of points into two clusters, select
    one of these clusters to split into 2, and so on, until K clusters

    Each of the splits is itself a round of kmeans (k=2)

Bisecting Kmeans

Algorithm (Parameters: K, num_trials)

Initialize -> one cluster consisting of all points. k = 1

while k < K

Choose and remove a cluster from the list of clusters.

// (Can be the biggest cluster or the cluster with the worst quality)

//Perform several “trial” bisections of the chosen cluster.

for *i* = 1 **to** num_trials **do**

Bisect the selected cluster using basic *K*means (i.e. 2-means).

Record SSE for this bisection

end for

Select the two-clustering trial with the lowest total SSE.

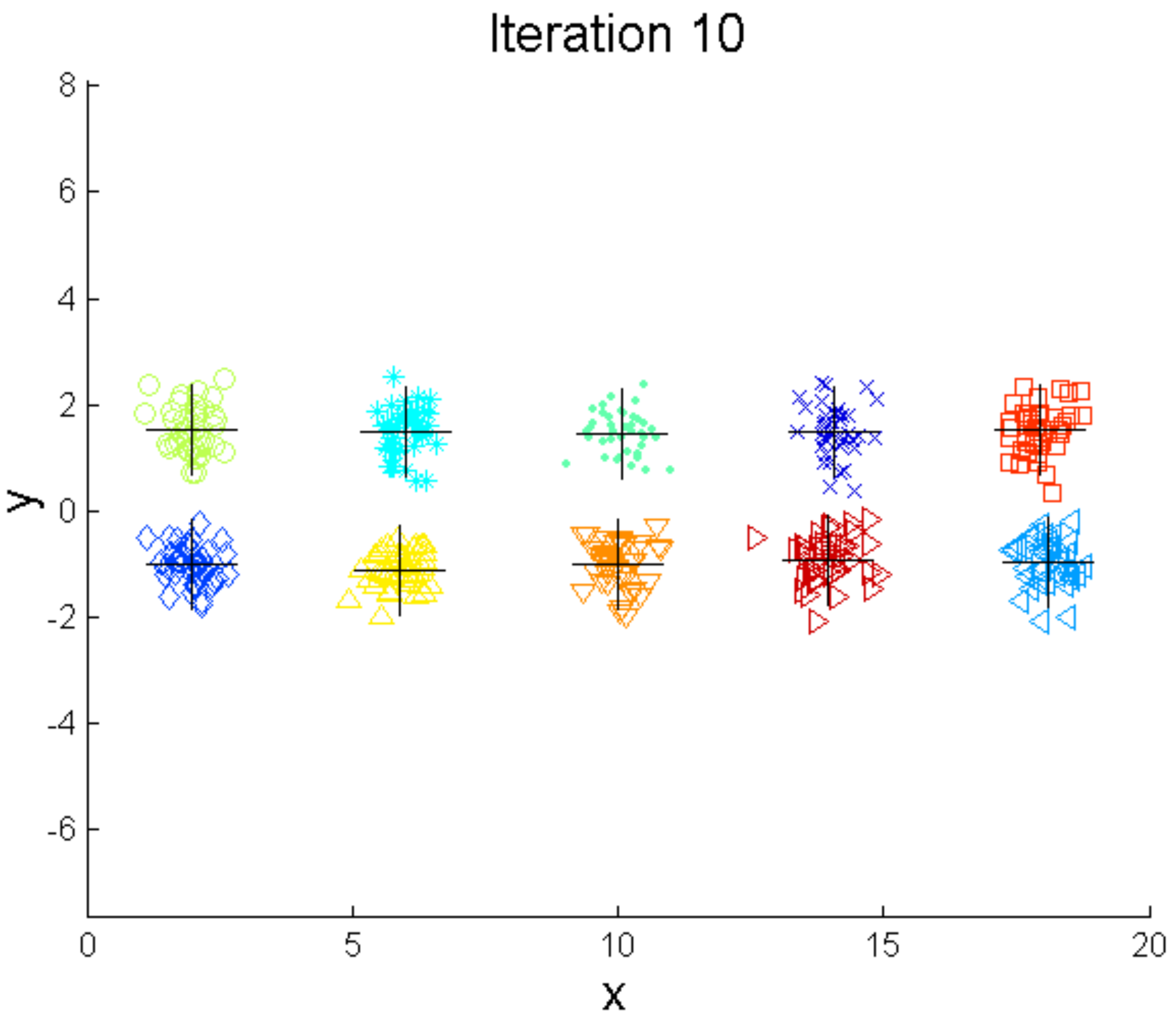
Add these two clusters to the list of clusters.

k = k+1

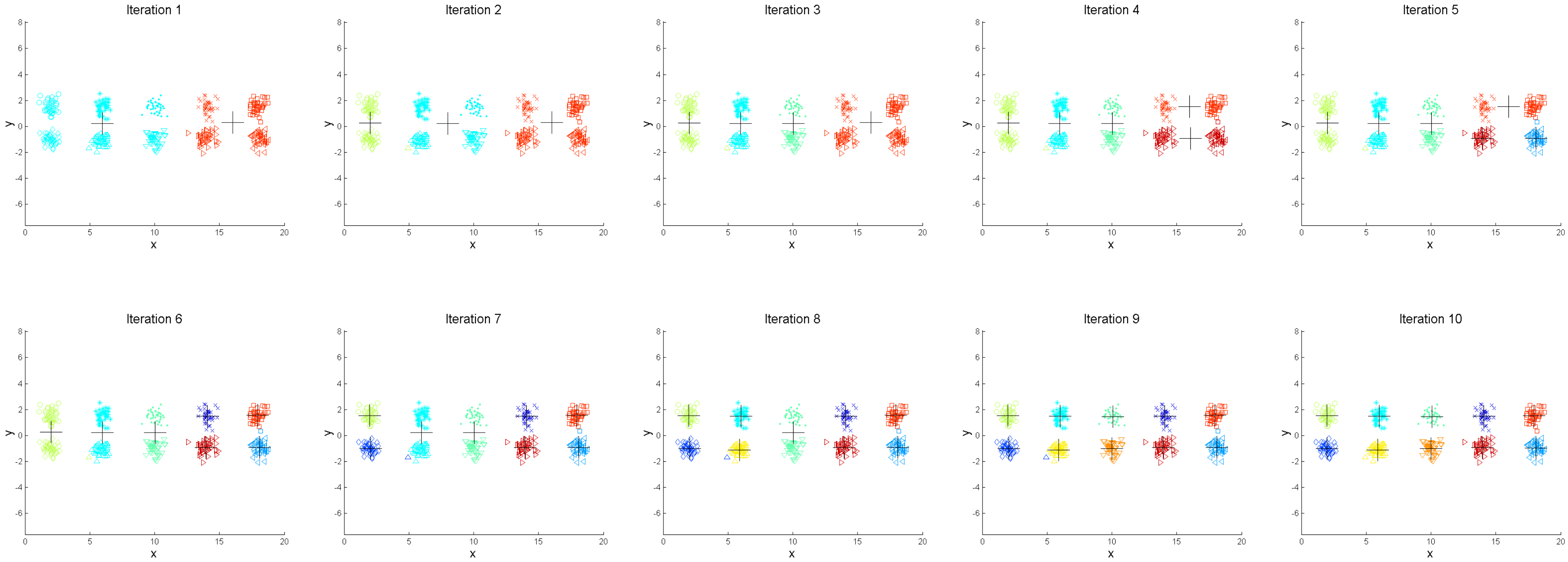
end for

Do one last round of assign-to-cluster and computer centroid

Bisecting K-means Example



B. K-means Example (without animation)

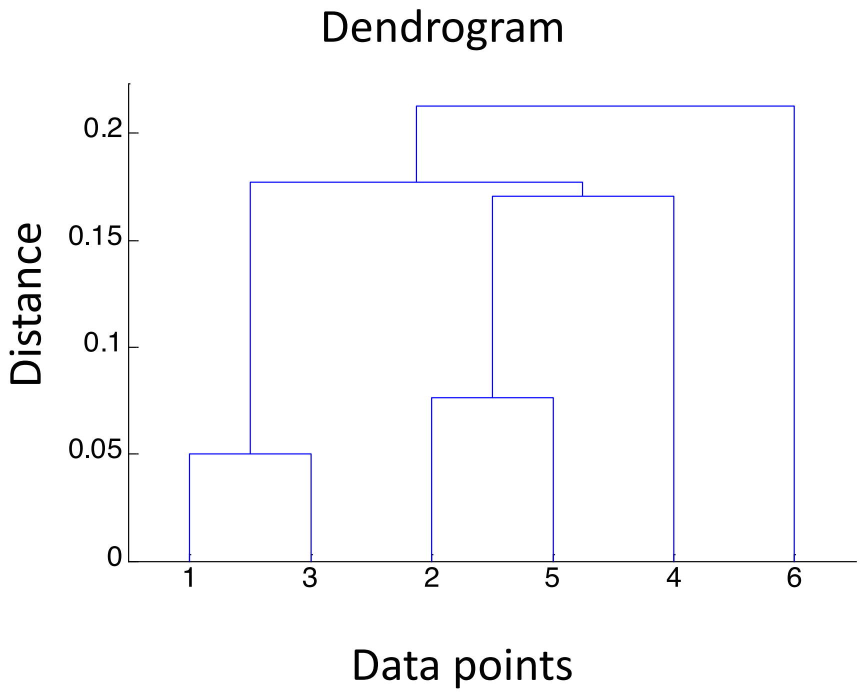


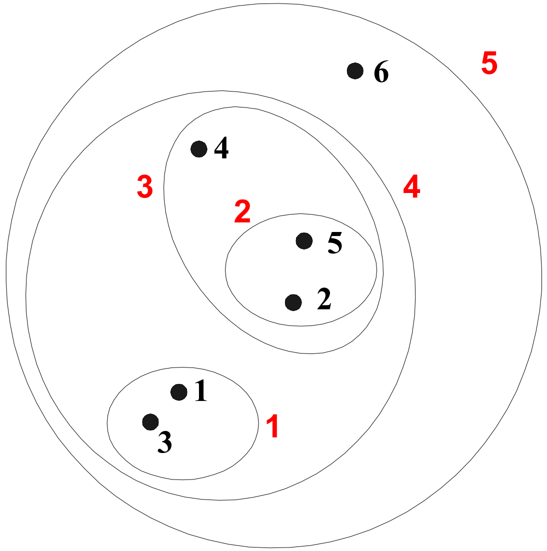
K-means vs. Hierarchical Clustering

- K-means uses the **intra** cluster distance to find clusters
- Hierarchical clustering uses the **inter** cluster distances.

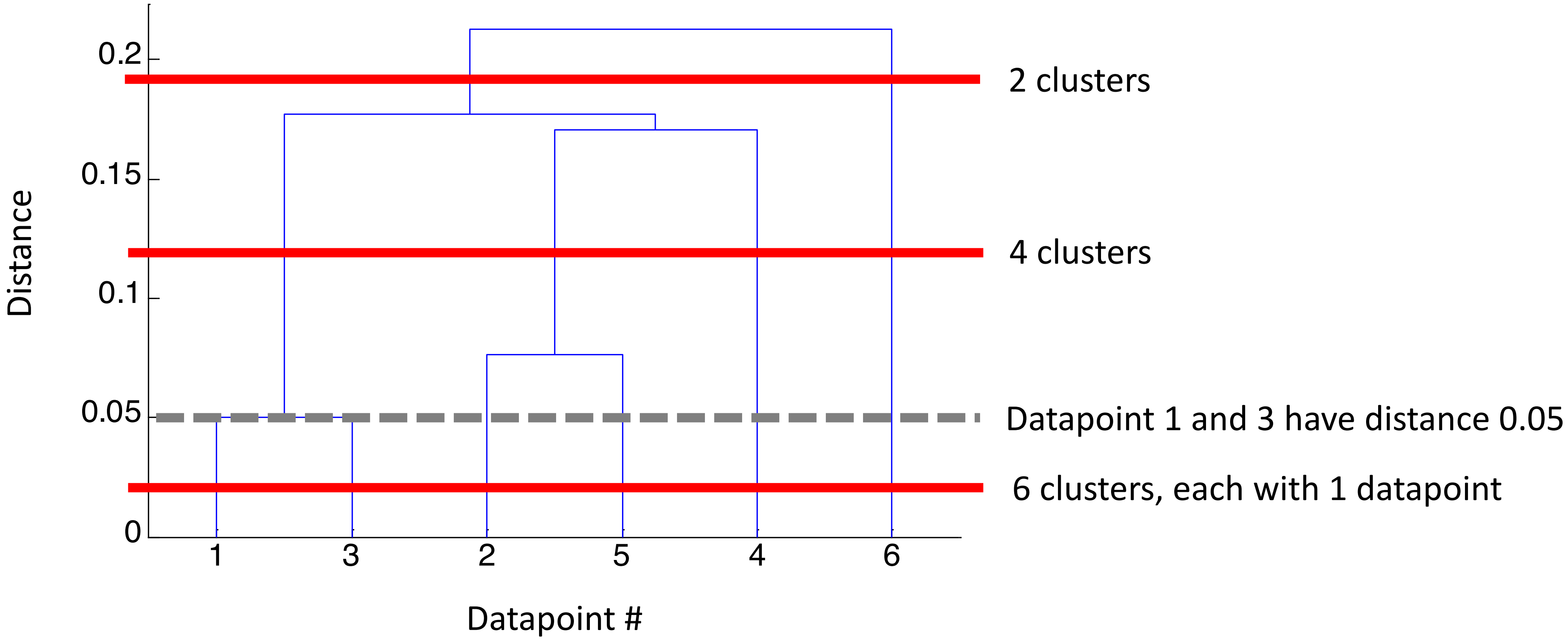
Hierarchical Clustering

- Produces a set of nested clusters organized as a hierarchical tree





How to read a dendrogram



Strengths of Hierarchical Clustering

- Do not have to assume any particular number of clusters
  - ‘cut’ the dendogram at the proper level to have a certain number of clusters
- They may correspond to meaningful taxonomies
  - Example in biological sciences e.g.,
    - animal kingdom,
    - phylogeny reconstruction,
    - …

Hierarchical Clustering

Algorithm

Let each data point be a cluster

Compute the proximity matrix

Repeat

  Merge **the two closest** clusters

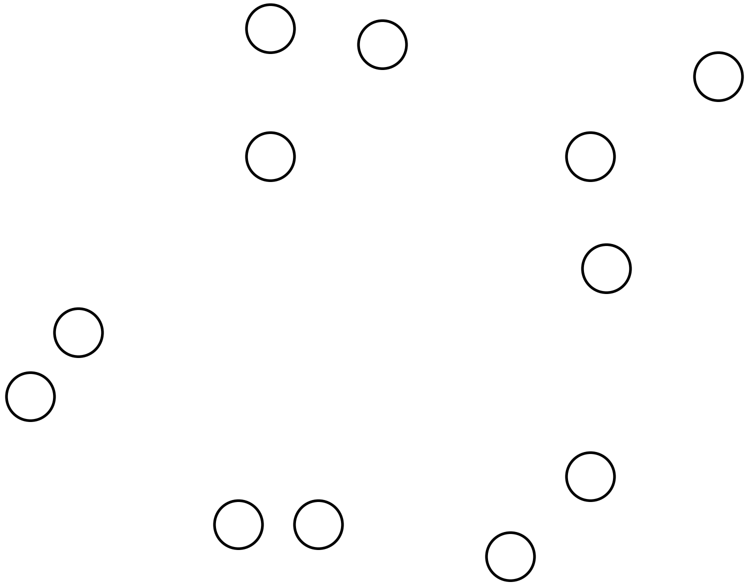
  Update the proximity matrix

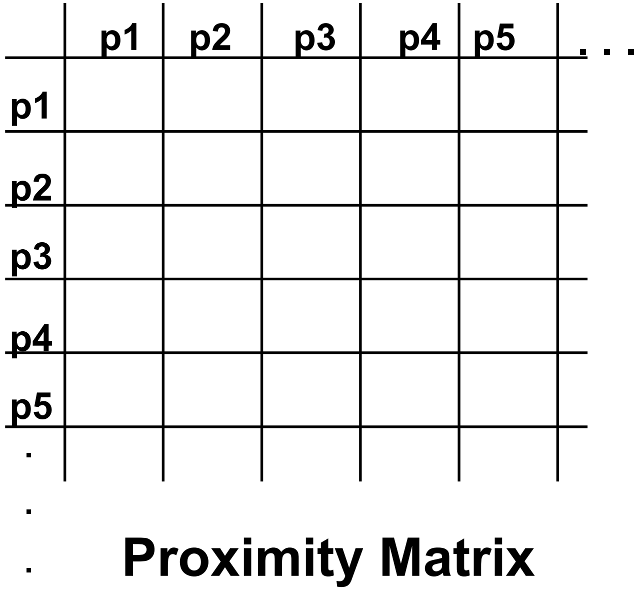
Until only a single cluster remains

- Key operation is the computation of *cluster closeness*

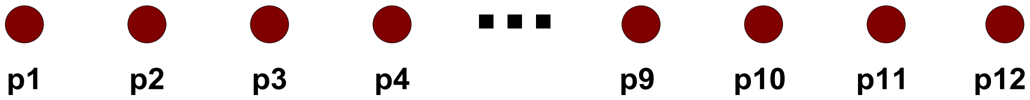
Starting Situation

- Start with clusters of individual points and a proximity matrix



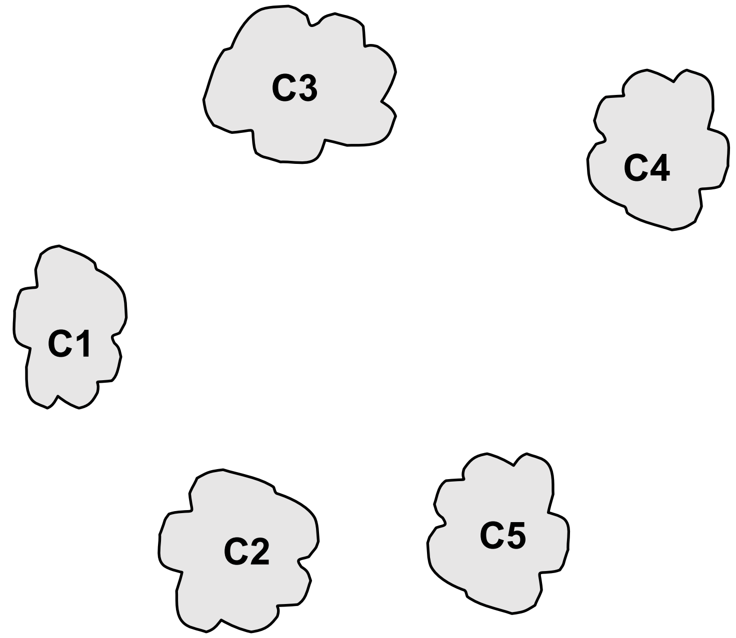


Proximity Matrix



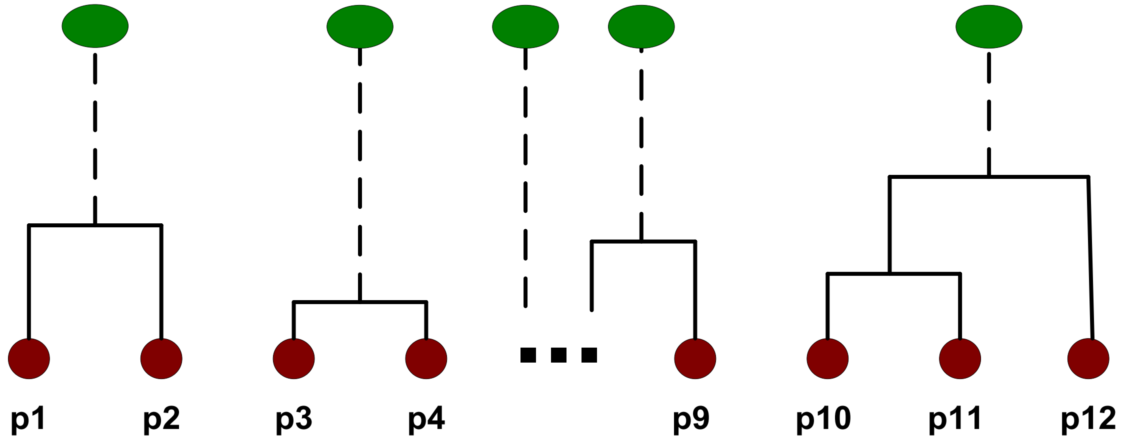
Intermediate Situation

- After some merging steps, we have some clusters



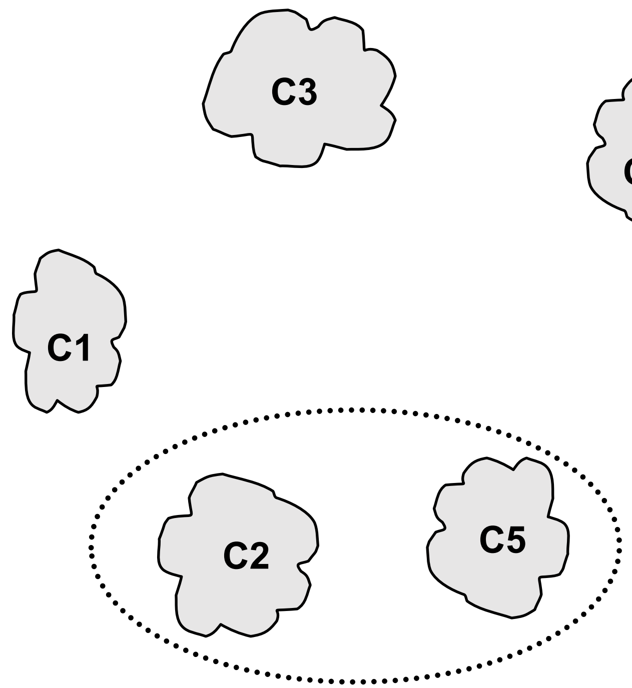
	C1	C2	C3	C4	C5
C1					
C2					
C3					
C4					
C5					

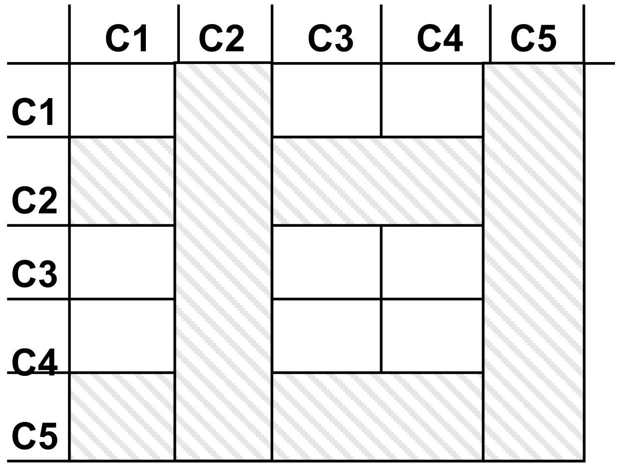
Proximity Matrix



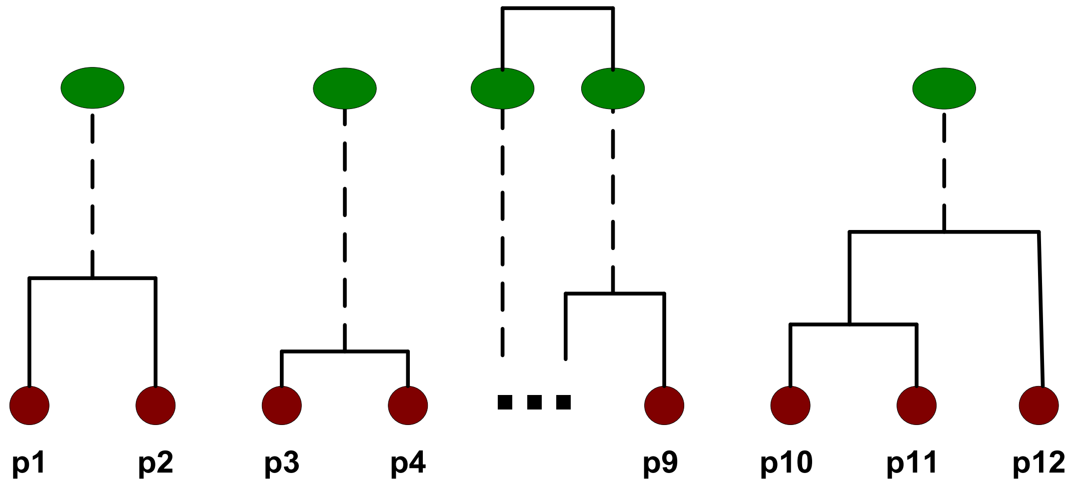
Intermediate Situation

- We want to merge the two closest clusters (say C2 and C5) and update the proximity matrix.



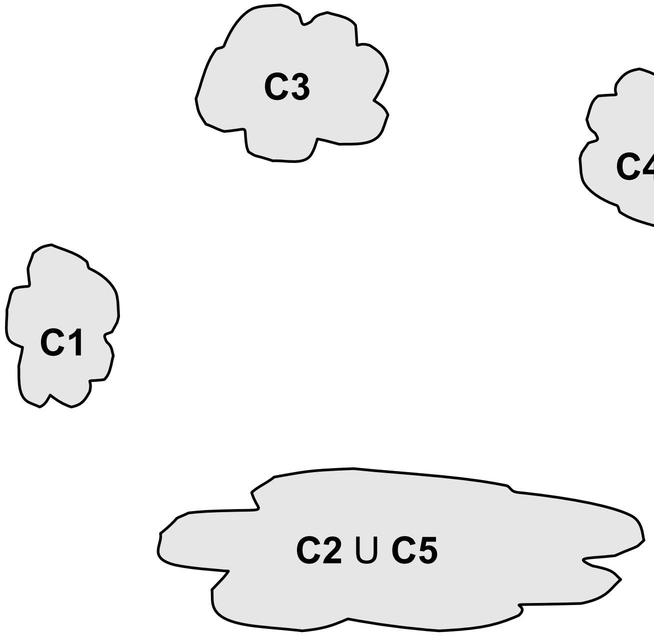


Proximity Matrix



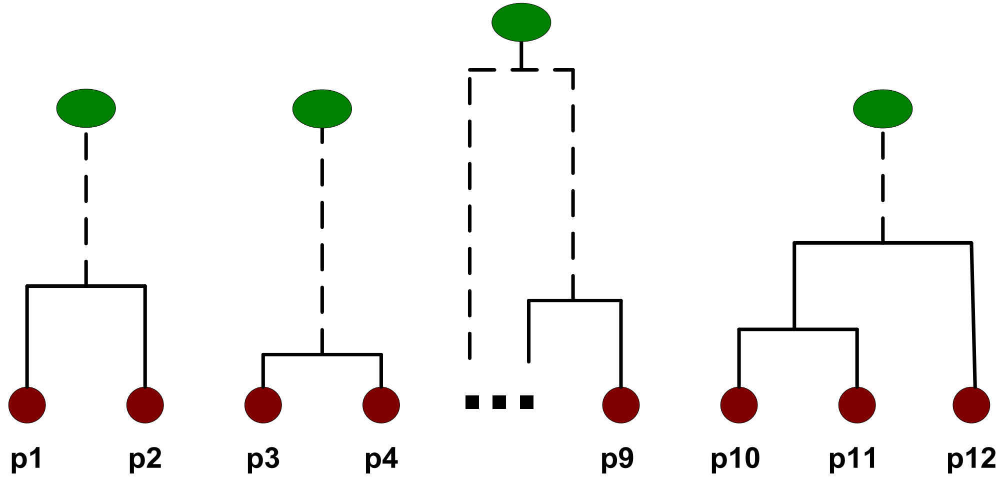
After Merging
(Symbol U means union [merge])

- The question is “How do we update the proximity matrix?”

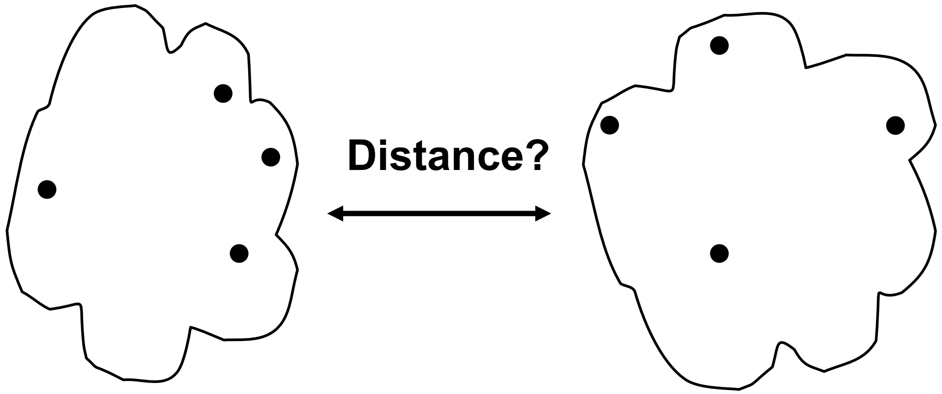


	C1	C2 U C5	C3	C4
C1		?		
C2 U C5	?	?	?	?
C3		?		
C4		?		

Proximity Matrix



First Define Inter-Cluster Similarity

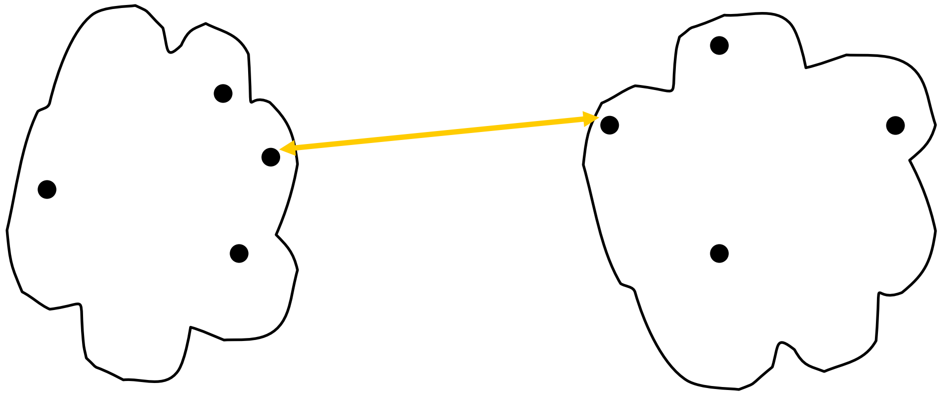


- MIN
- MAX
- Group Average

	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix

How to Define Inter-Cluster Similarity



	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						

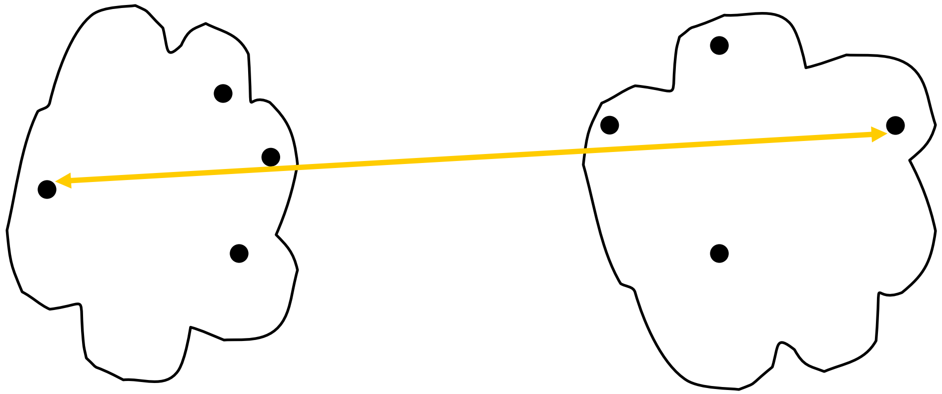
.

.

Proximity Matrix

- MIN
- MAX
- Group Average

How to Define Inter-Cluster Similarity

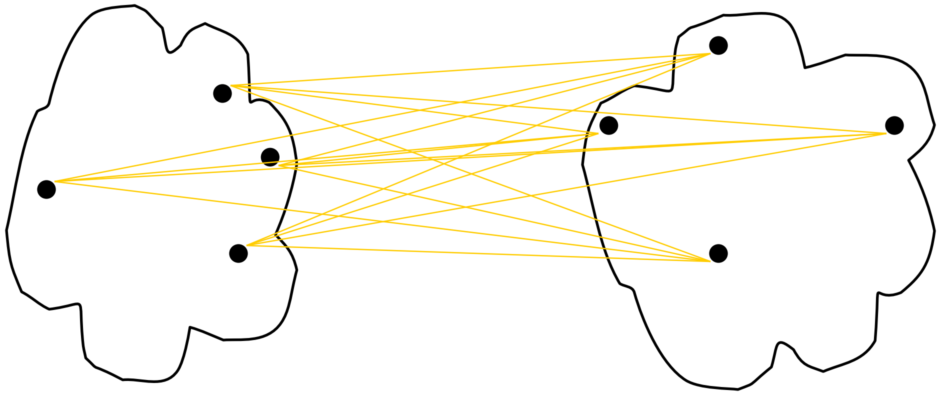


- MIN
- MAX
- Group Average

	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix

How to Define Inter-Cluster Similarity



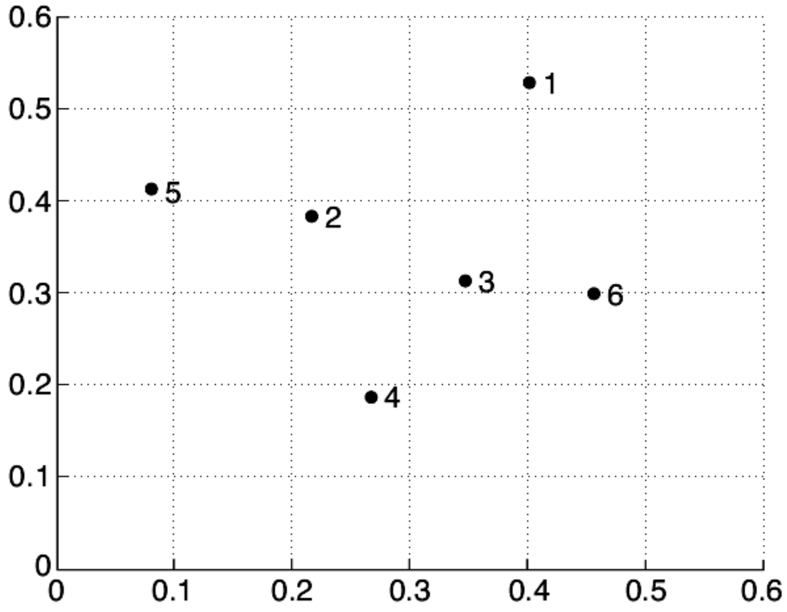
- MIN
- MAX
- Group Average

	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix

Cluster Similarity: MIN

- Similarity of two clusters is based on the two most similar (closest) points in the different clusters
  - Determined by one pair of points

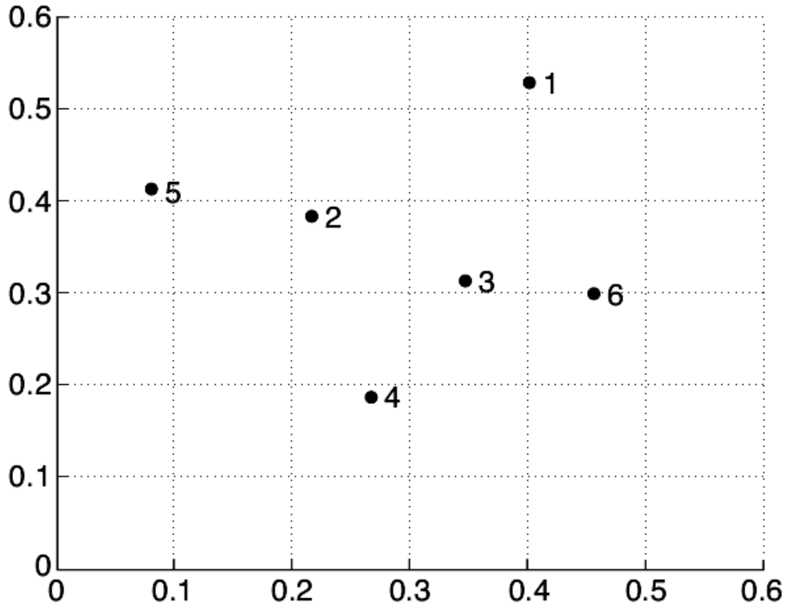



```
        1      2      3      4      5      6
[1[ 0.  ,  0.24,  0.22,  0.37,  0.34,  0.23],
 2[ 0.24,  0.  ,  0.15,  0.2 ,  0.14,  0.25],
 3[ 0.22,  0.15,  0.  ,  0.16,  0.28,  0.11],
 4[ 0.37,  0.2 ,  0.16,  0.  ,  0.29,  0.22],
 5[ 0.34,  0.14,  0.28,  0.29,  0.  ,  0.39],
 6[ 0.23,  0.25,  0.11,  0.22,  0.39,  0.  ]]
```

```
          1      2      3&6    4      5
  [1[ 0.  ,  0.24,  0.22,  0.37,  0.34],
   2[ 0.24,  0.  ,  0.15,  0.2 ,  0.14],
 3&6[ 0.22,  0.15,  0.  ,  0.16,  0.28],
   4[ 0.37,  0.2 ,  0.16,  0.  ,  0.29],
   5[ 0.34,  0.14,  0.28,  0.29,  0.  ],]
```

Cluster Similarity: MIN

- Similarity of two clusters is based on the two most similar (closest) points in the different clusters
  - Determined by one pair of points

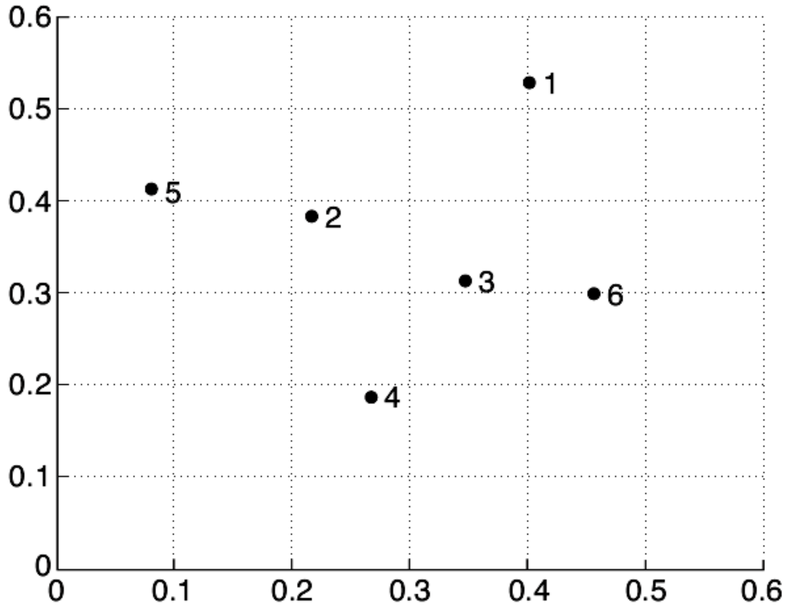



```
         1       2      3&6     4       5
   [1[ 0.   ,  0.24,  0.22,  0.37,  0.34],
    2[ 0.24,  0.   ,  0.15,  0.2 ,  0.14],
  3&6[ 0.22,  0.15,  0.   ,  0.16,  0.28],
    4[ 0.37,  0.2 ,  0.16,  0.   ,  0.29],
    5[ 0.34,  0.14,  0.28,  0.29,  0.   ],]
```

```
          1      2&5     3&6      4
    [1[ 0.   ,  0.24,  0.22,  0.37],
  2&5[ 0.24,  0.   ,  0.15,  0.2 ],
  3&6[ 0.22,  0.15,  0.   ,  0.16],
    4[ 0.37,  0.2 ,  0.16,  0.   ],]
```

Cluster Similarity: MIN

- Similarity of two clusters is based on the two most similar (closest) points in the different clusters

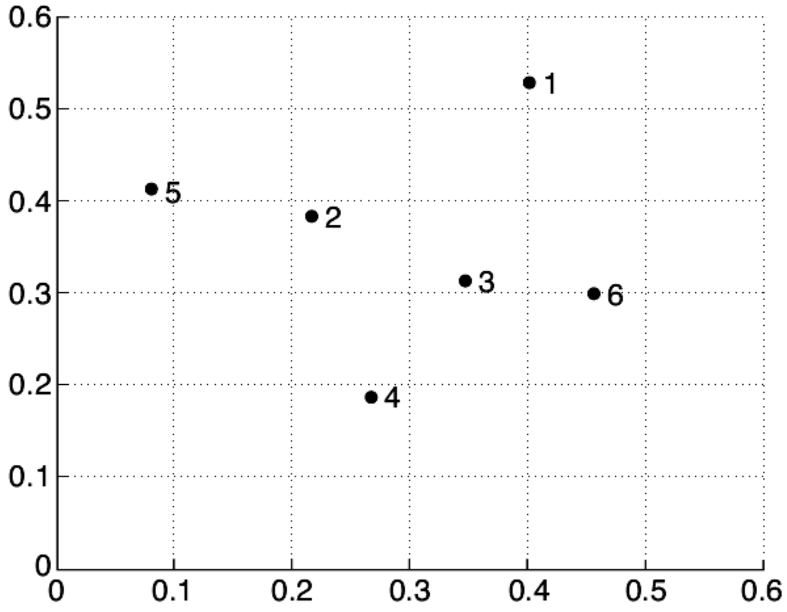


	1	2&5	3&6	4
1	0.	0.24	0.22	0.37
2&5	0.24	0.	0.15	0.2
3&6	0.22	0.15	0.	0.16
4	0.37	0.2	0.16	0.

	1	2&3&5&6	4
1	0.	0.22	0.37
2&3&5&6	0.22	0.	0.16
4	0.37	0.16	0.

Cluster Similarity: MIN

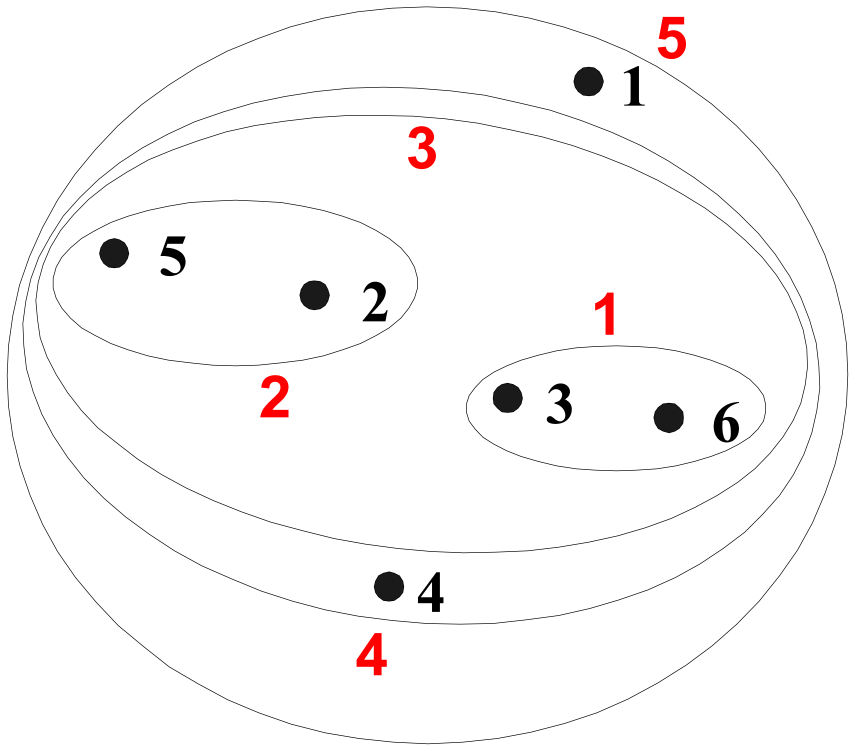
- Similarity of two clusters is based on the two most similar (closest) points in the different clusters
  - Determined by one pair of points



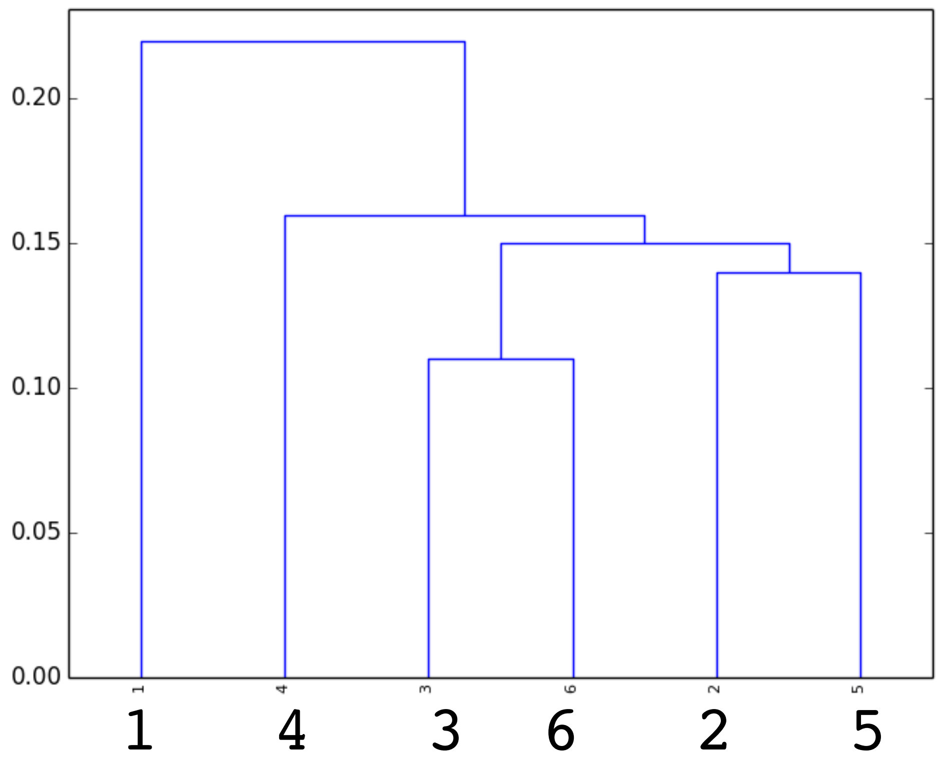

```
              1    2&3&5&6    4
         [1[ 0.  ,  0.22,  0.37],
  2&3&5&6[ 0.22,  0.   ,  0.16 ],
         4[ 0.37,  0.16 ,  0.   ],]
```

```
              1    2&3&4&5&6
         [1[ 0.  ,  0.22],
  2&3&5&6[ 0.22,  0.   ]]
```

Hierarchical Clustering: MIN

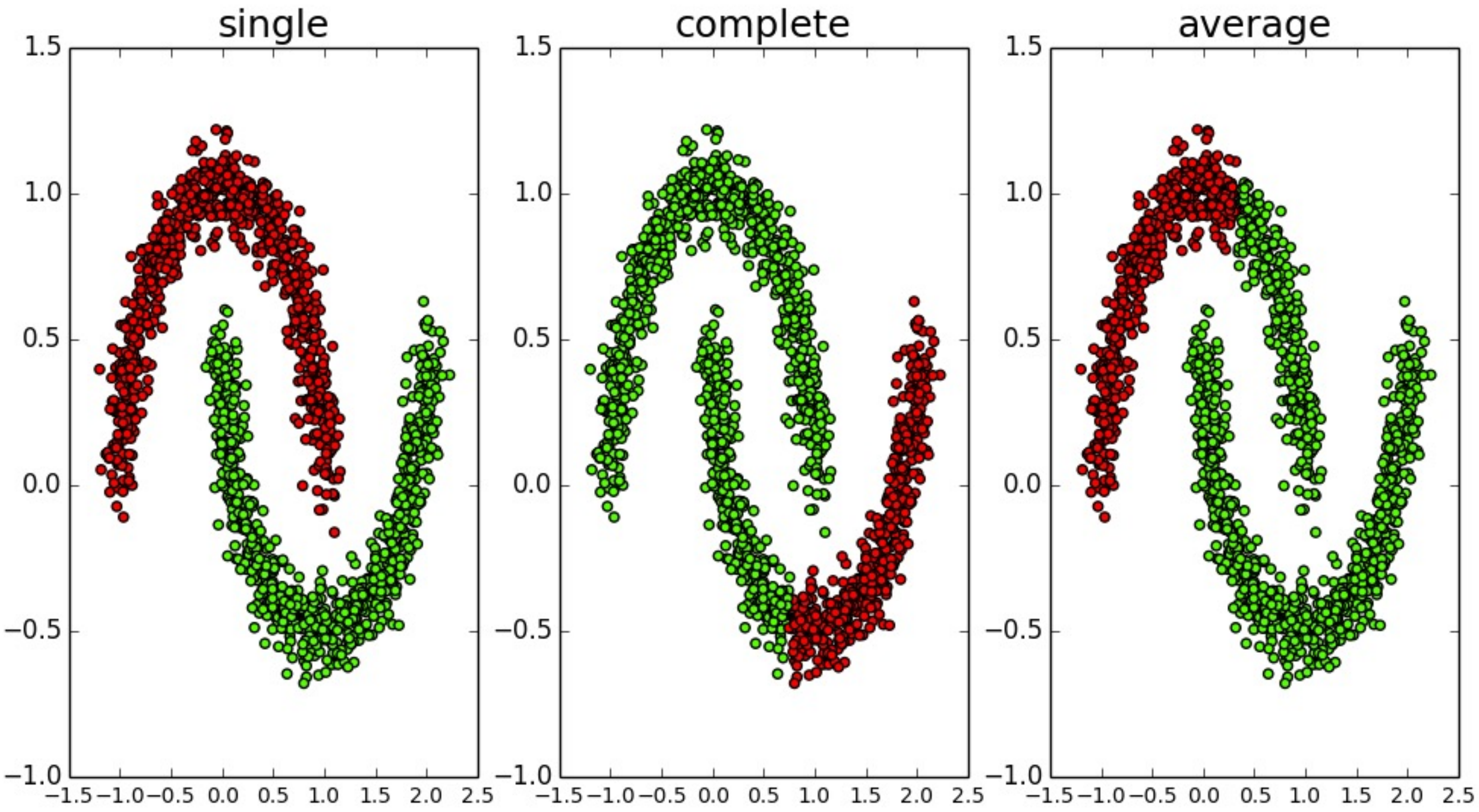


Nested Clusters



Dendrogram

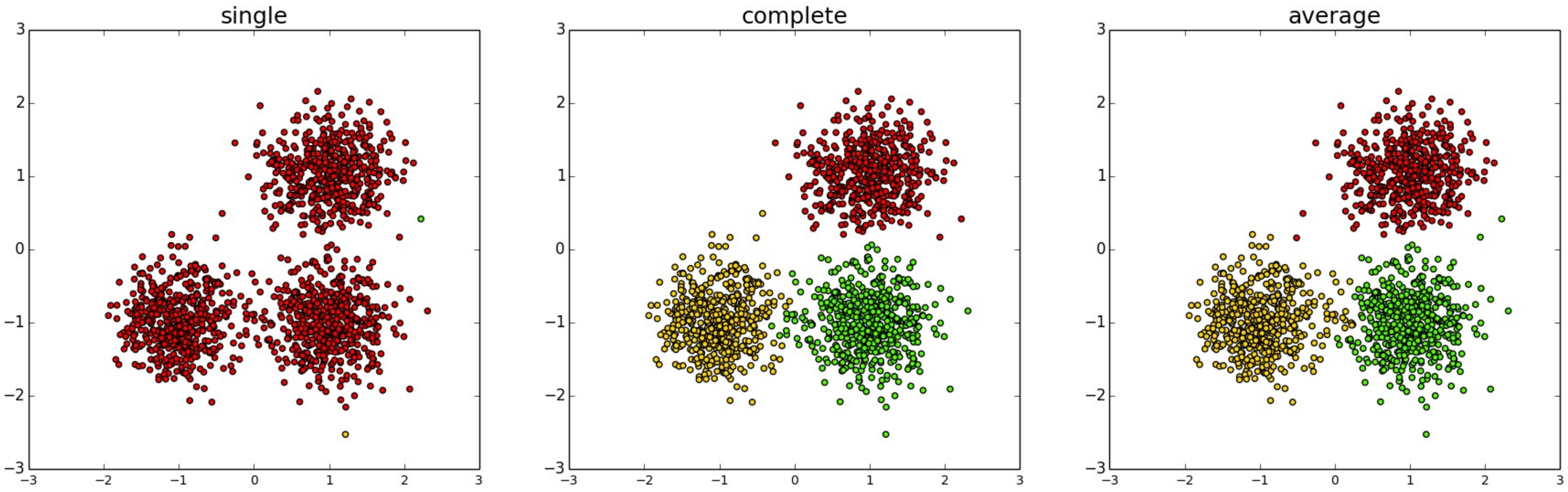
Strength of MIN (single)



Can handle non-globular shapes

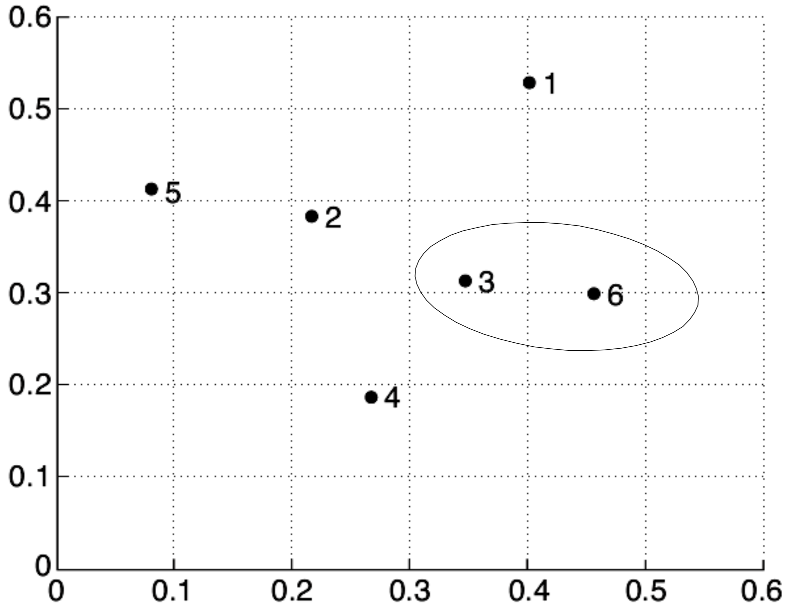
Limitations of MIN (single)

Sensitive to noise and outliers



Cluster Similarity: MAX

- Similarity of two clusters is based on the two least similar (most distant) points in the different clusters
  - Determined by all pairs of points in the two clusters

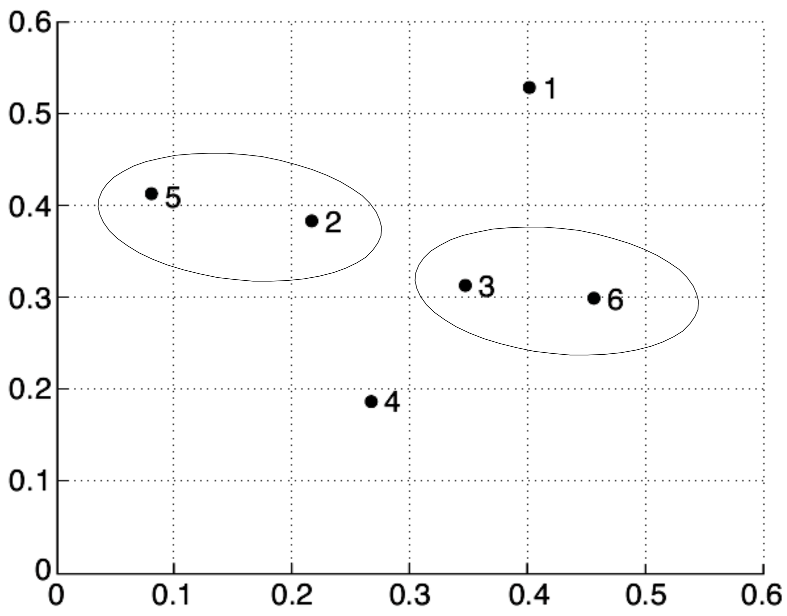



```
[[ 0.  ,  0.24,  0.22,  0.37,  0.34,  0.23],
 [ 0.24,  0.  ,  0.15,  0.2 ,  0.14,  0.25],
 [ 0.22,  0.15,  0.  ,  0.16,  0.28,  0.11],
 [ 0.37,  0.2 ,  0.16,  0.  ,  0.29,  0.22],
 [ 0.34,  0.14,  0.28,  0.29,  0.  ,  0.39],
 [ 0.23,  0.25,  0.11,  0.22,  0.39,  0.  ]]
```

```
           1      2     3&6      4      5
[   1[ 0.  ,  0.24,  0.23,  0.37,  0.34],
    2[ 0.24,  0.  ,  0.25,  0.2 ,  0.14],
  3&6[ 0.23,  0.25,  0.  ,  0.22,  0.39],
    4[ 0.37,  0.2 ,  0.22,  0.  ,  0.29],
    5[ 0.34,  0.14,  0.39,  0.29,  0.  ]]
```

Cluster Similarity: MAX

- Similarity of two clusters is based on the two least similar (most distant) points in the different clusters

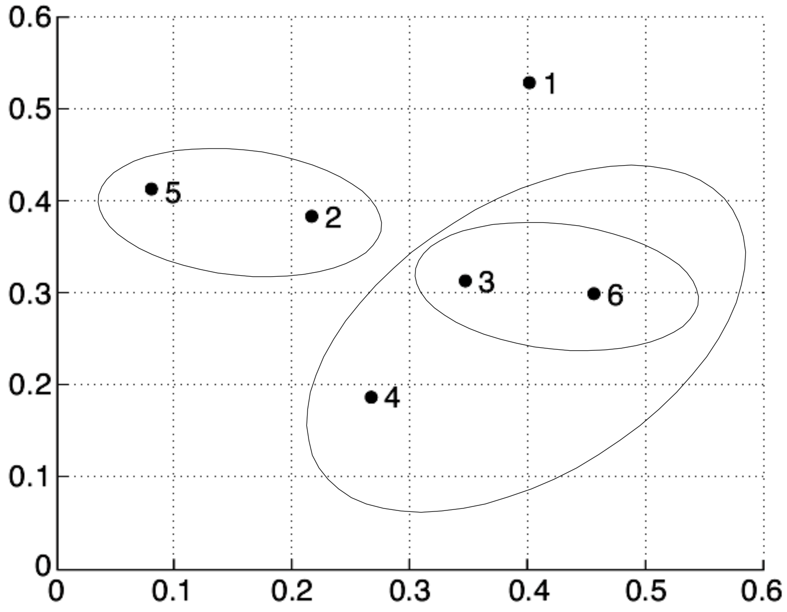


	1	2	3&6	4	5
1	0.	0.24	0.23	0.37	0.34
2	0.24	0.	0.25	0.2	0.14
3&6	0.23	0.25	0.	0.22	0.39
4	0.37	0.2	0.22	0.	0.29
5	0.34	0.14	0.39	0.29	0.

	1	2&5	3&6	4
1	0.	0.34	0.23	0.37
2&5	0.34	0.	0.39	0.29
3&6	0.23	0.39	0.	0.22
4	0.37	0.29	0.22	0.

Cluster Similarity: MAX

- Similarity of two clusters is based on the two least similar (most distant) points in the different clusters
  - Determined by all pairs of points in the two clusters

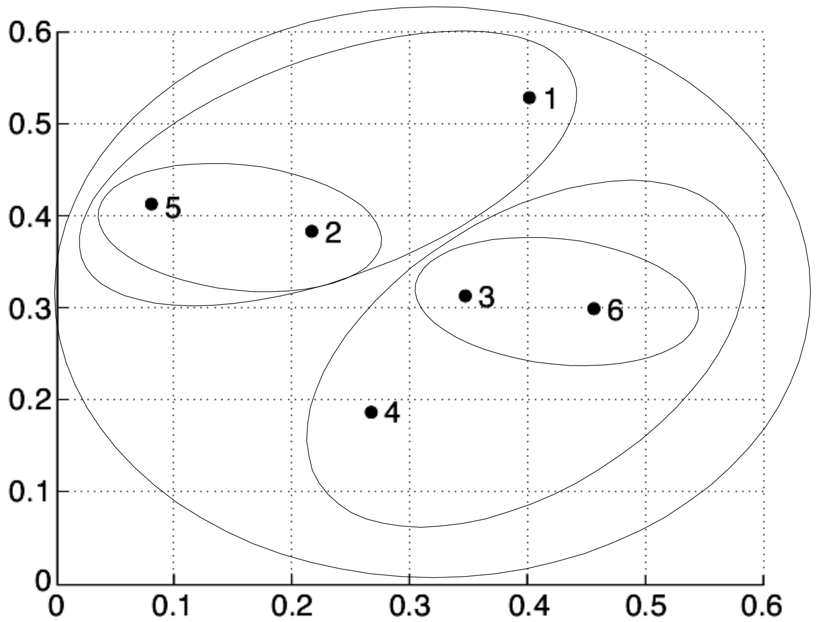



```
          1      2&5     3&6     4
[   1[ 0.   ,  0.34,  0.23,  0.37],
  2&5[ 0.34,  0.   ,  0.39,  0.29],
  3&6[ 0.23,  0.39,  0.   ,  0.22],
    4[ 0.37,  0.29,  0.22,  0.  ]]
```

```
          1      2&5    3&4&6
[   1[ 0.   ,  0.34,  0.37],
  2&5[ 0.34,  0.   ,  0.39],
  3&6[ 0.37,  0.39,  0.  ]]
```

Cluster Similarity: MAX

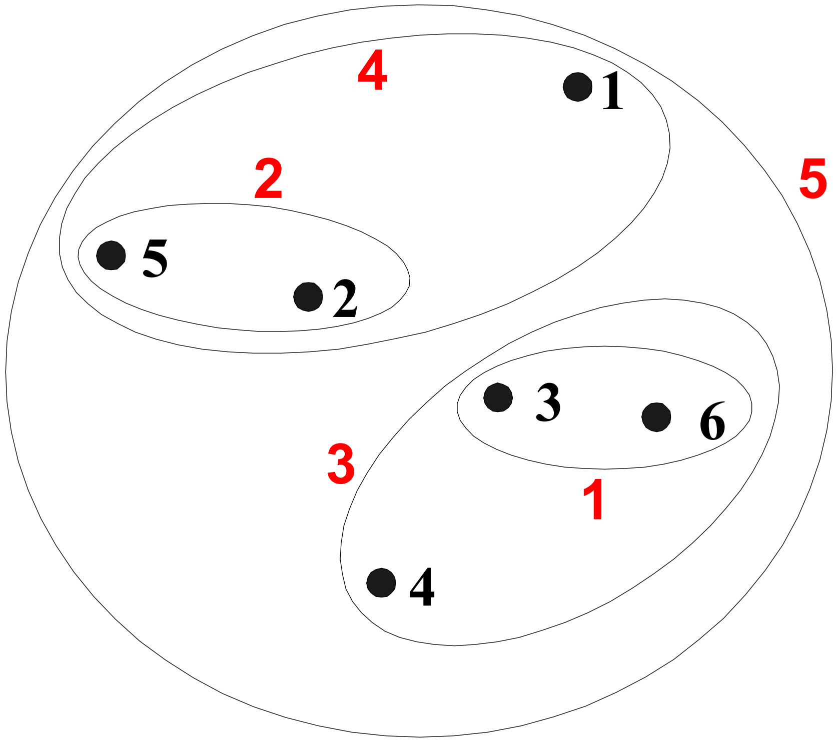
- Similarity of two clusters is based on the two least similar (most distant) points in the different clusters
  - Determined by all pairs of points in the two clusters
-



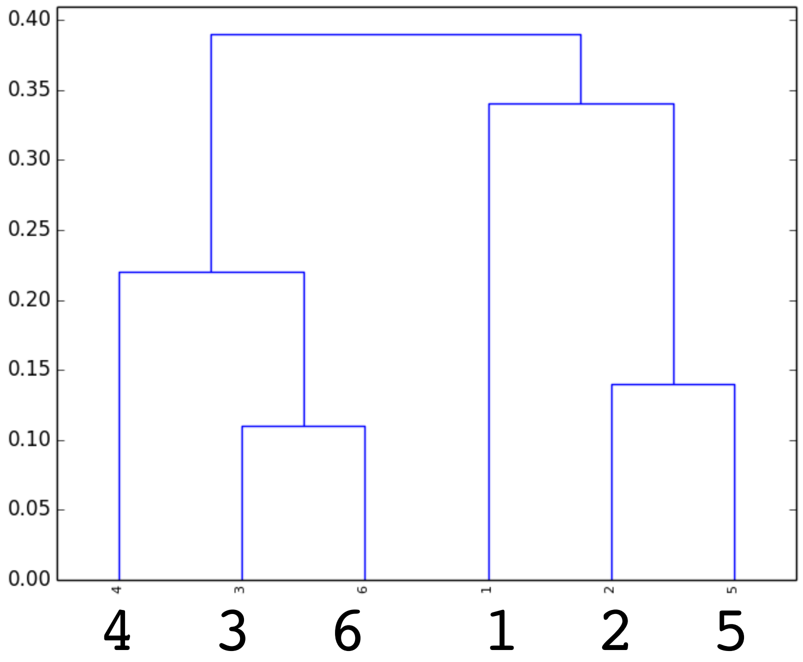

```
            1      2&5     3&4&6
[    1[ 0.   ,  0.34,  0.37],
   2&5[ 0.34,  0.   ,  0.39],
 3&4&6[ 0.37,  0.39,  0.   ]]
```

```
        1&2&5    3&4&6
[1&2&5[ 0.   ,  0.39],
   3&6[ 0.39,  0.   ]]
```

Hierarchical Clustering: MAX

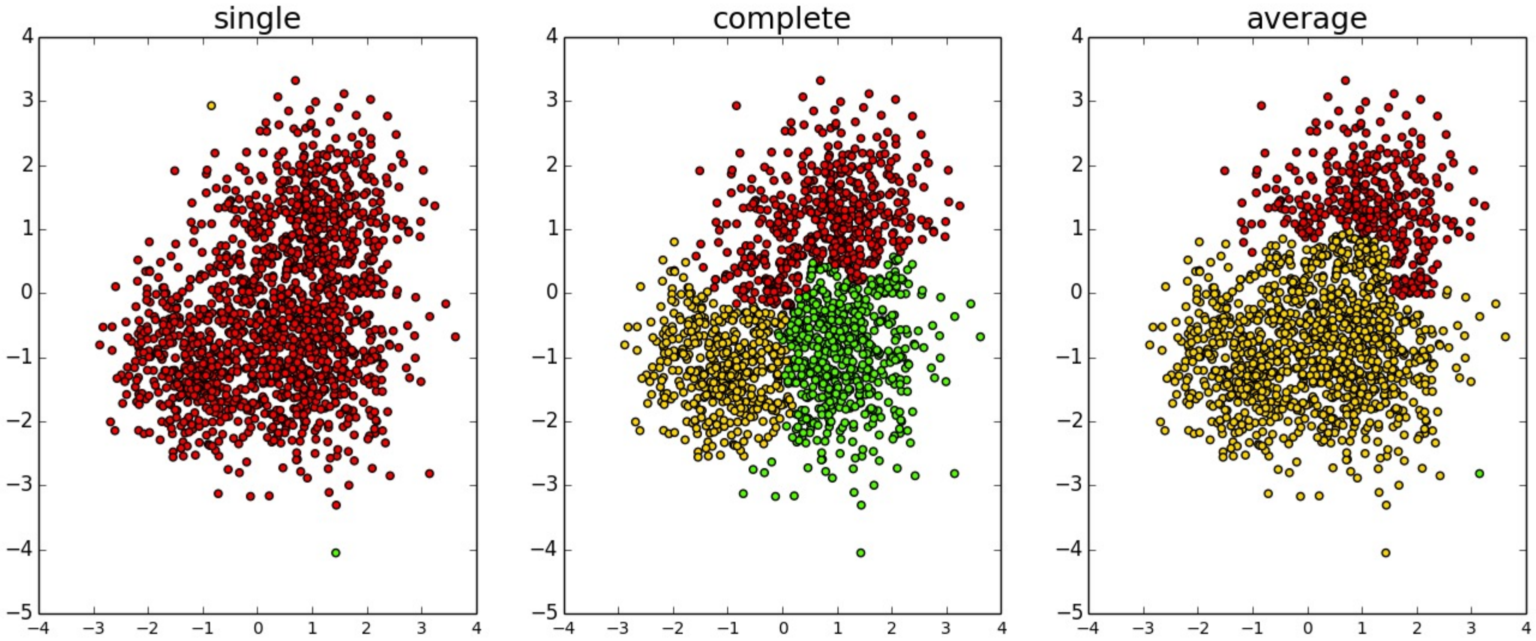


Nested Clusters



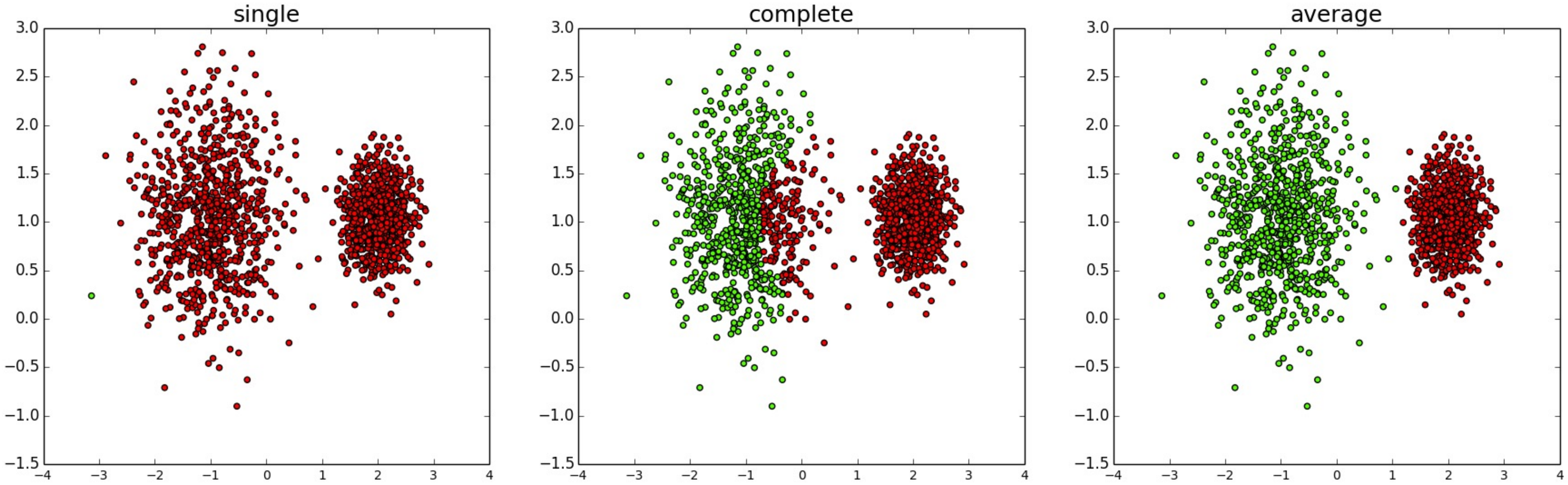
Dendrogram

Strengths of MAX (complete)



Less susceptible with respect to noise and outliers

Limitations of MAX (complete)



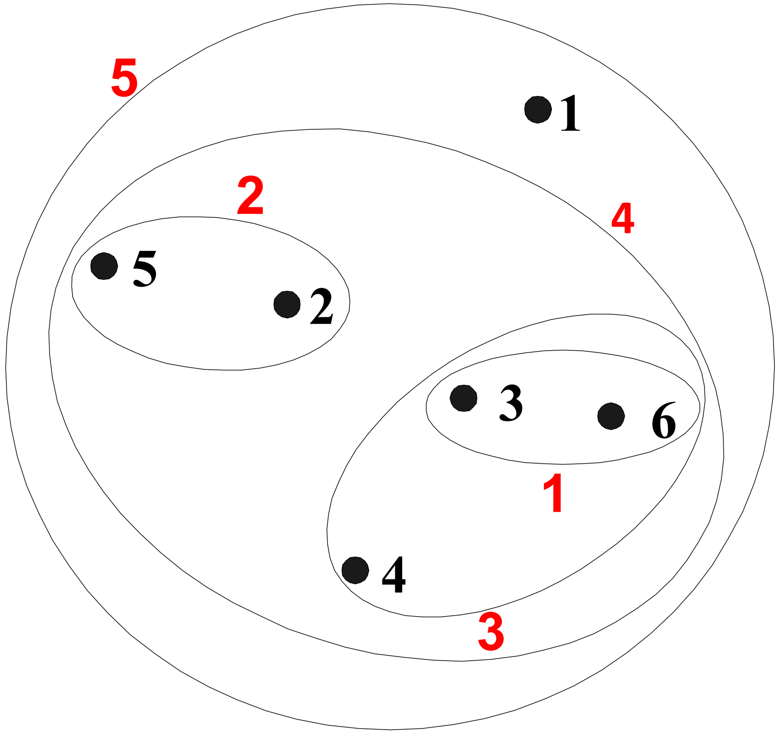
Tends to break large clusters

Cluster Similarity: Group Average

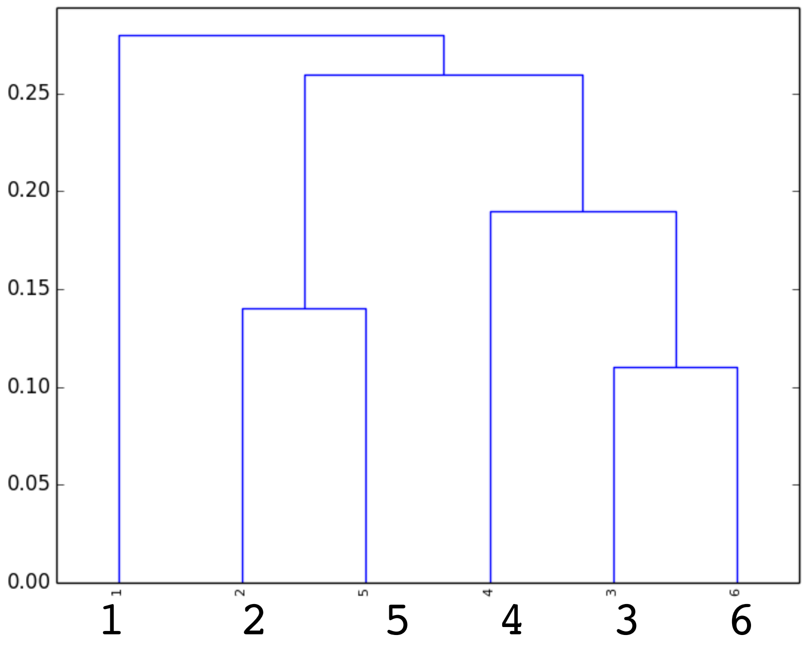
- Proximity of two clusters is the average of pairwise proximity between points in the two clusters.

$$\mathbf{proximity}(\mathbf{Cluster_i}, \mathbf{Cluster_j}) = \frac{\displaystyle\sum_{\substack{\mathbf{p_i} \in \mathbf{Cluster_i} \\ \mathbf{p_j} \in \mathbf{Cluster_j}}} \mathbf{proximity}(\mathbf{p_i}, \mathbf{p_j})}{|\mathbf{Cluster_i}| * |\mathbf{Cluster_j}|}$$

Hierarchical Clustering: Group Average

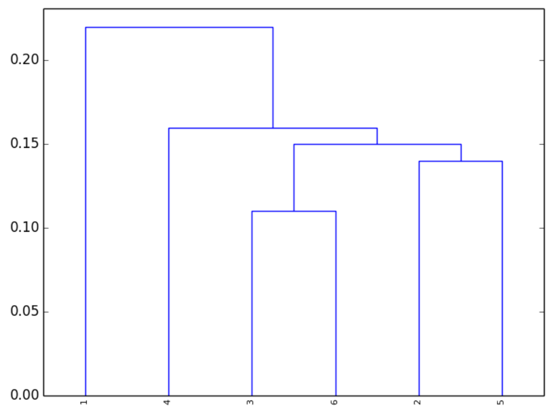


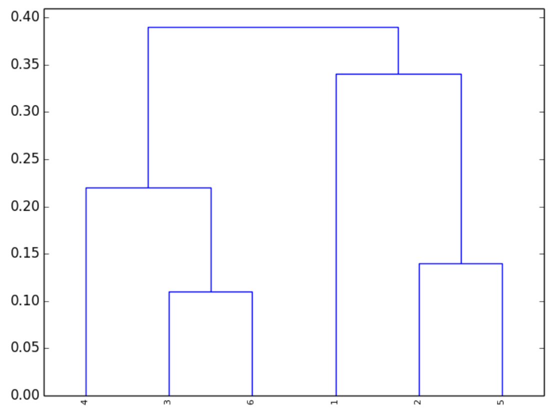
Nested Clusters

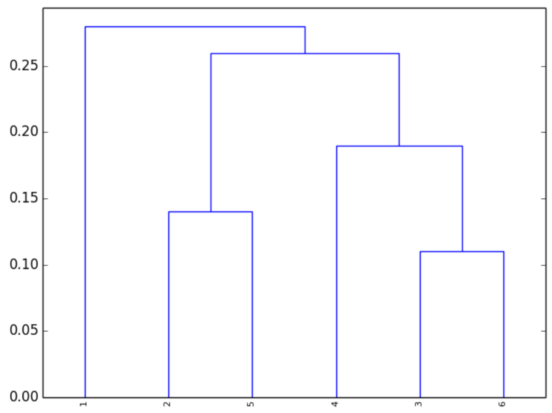


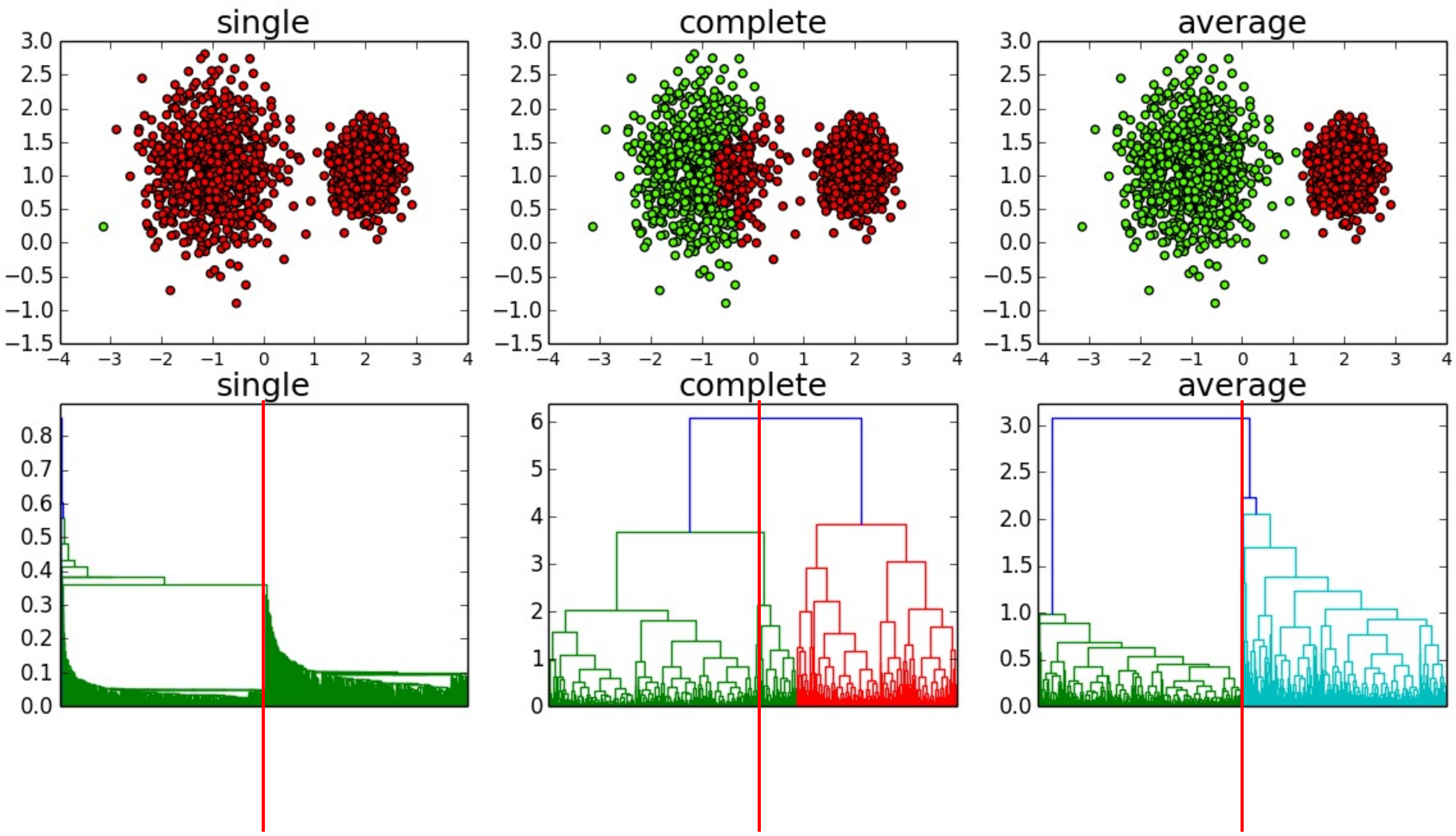
Dendrogram

Different Dendrograms









single
complete
average
single
complete
average

Unsupervised learning and compression

- Characteristics that can negatively impact clustering
  - Correlated features
  - Irrelevant features

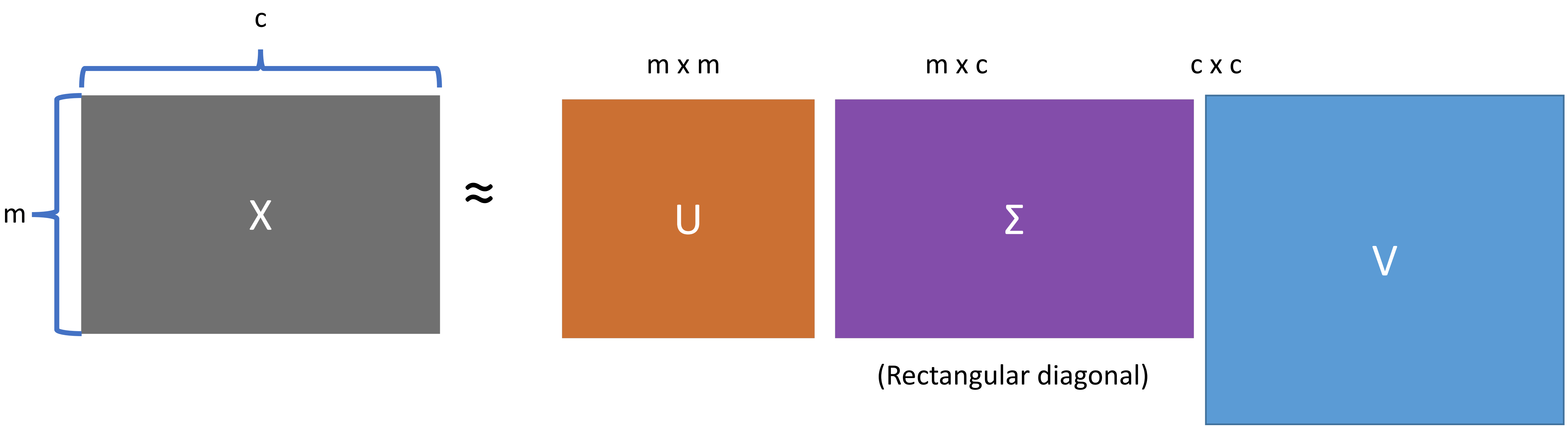
- How to solve this?

Unsupervised learning and compression

- Compress data first

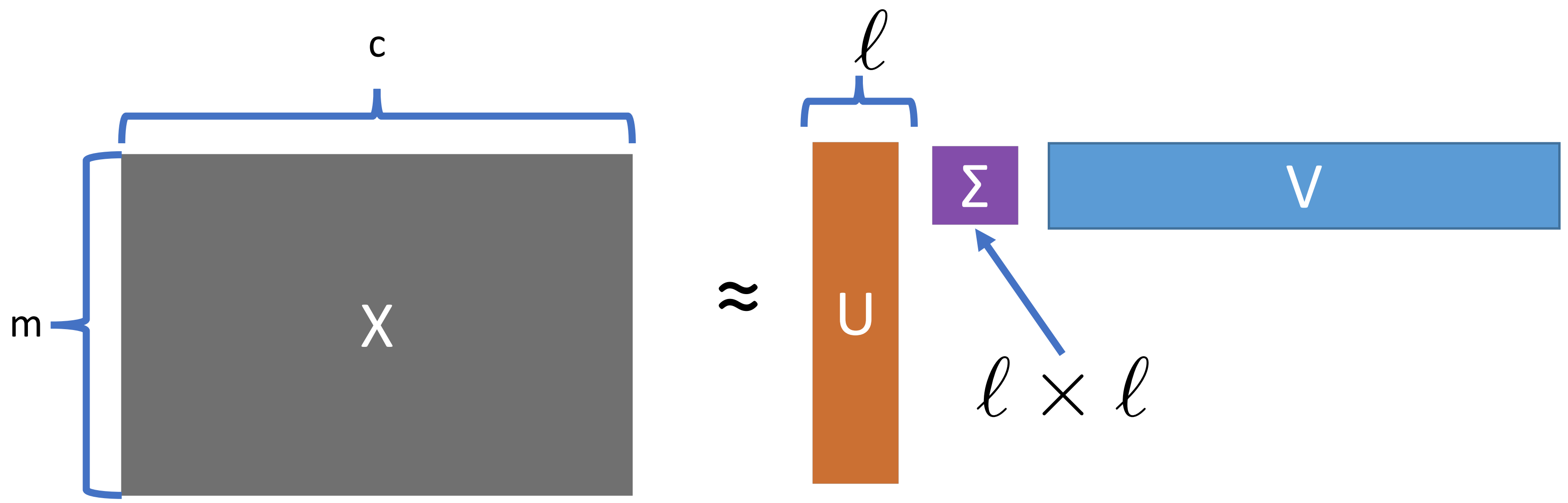
Singular Value Decomposition

$$X = U\Sigma V^T$$



Singular Value Decomposition

$$X = U\Sigma V^T$$

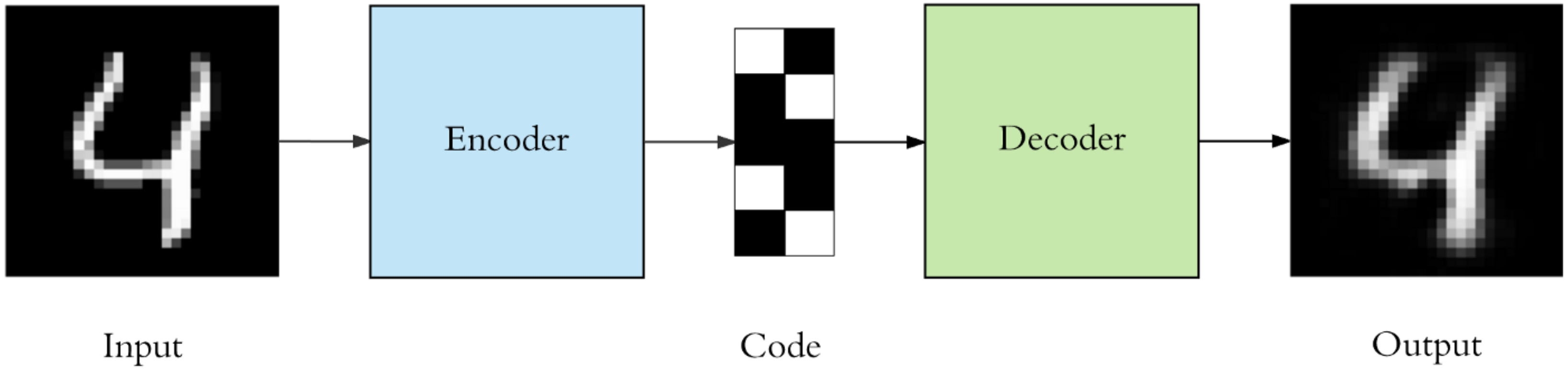


$$\ell << c$$

Autoencoders

- How can we set up compression as a learning problem?
  - Can set it up as a supervised learning problem, solve via least squares

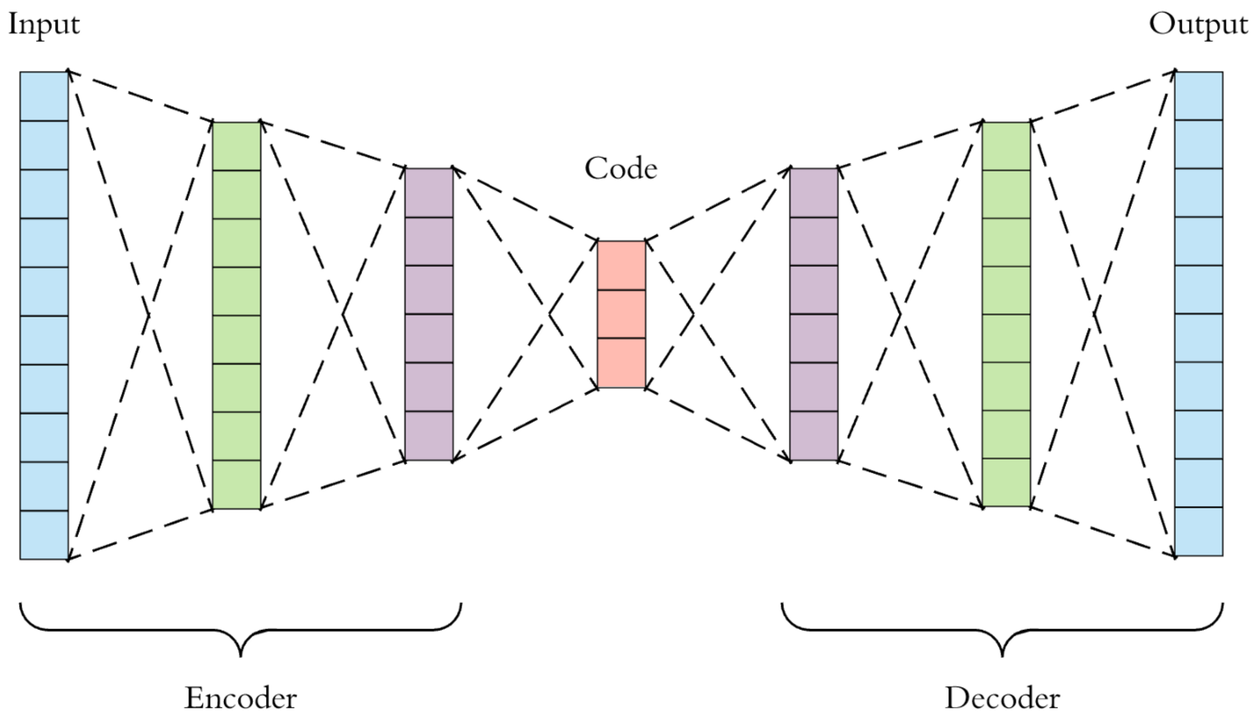
- Use a neural network



https://towardsdatascience.com/applied-deep-learning-part-3-autoencoders-1c083af4d798

Autoencoders

- A neural network that is trained to reproduce the input as output
- Some constraint to avoid learning the identity matrix
  - Non-linearity
  - compression



Autoencoders

- Supervised neural networks wish to learn
  - F(x) = y
  - Network applied to input x should predict label y

- Unsupervised neural networks wish to learn
  - F(x) = x
  - Network applied to input x should reproduce x
  - The data is the label!