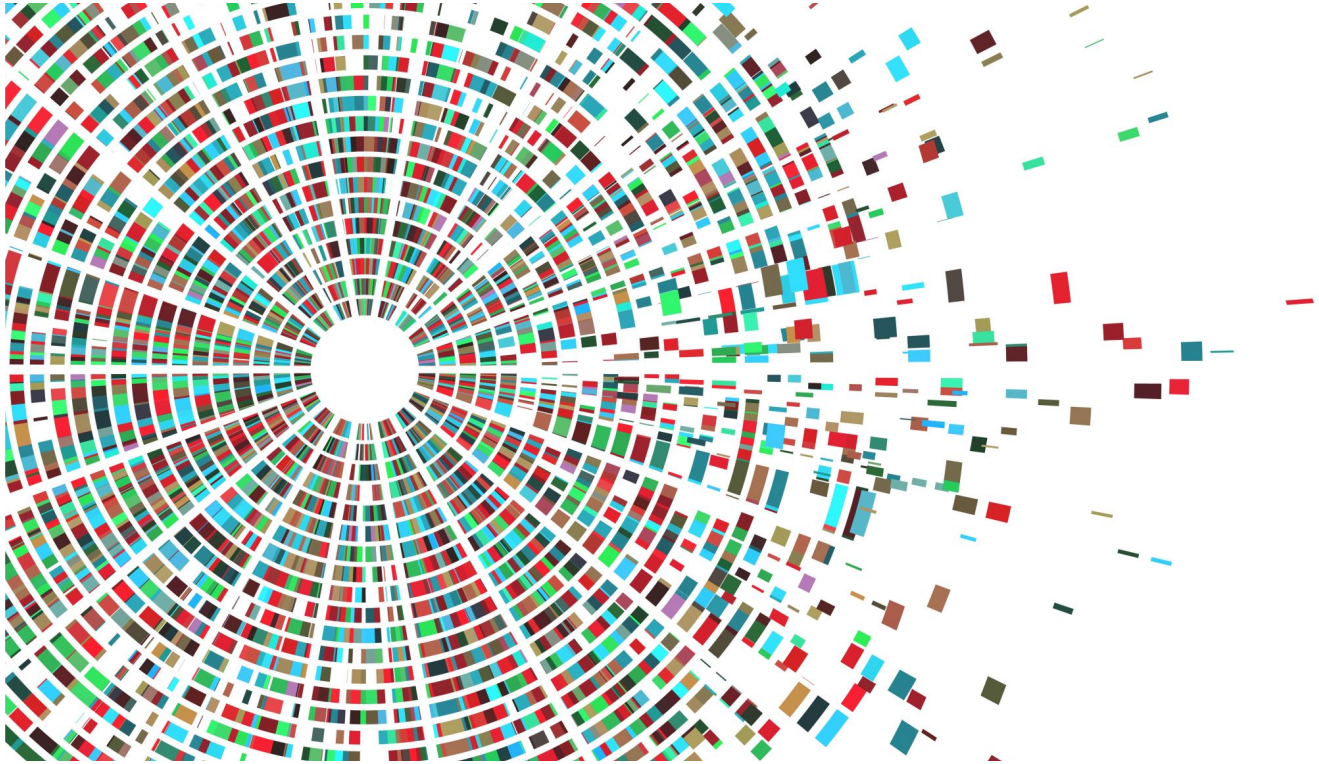
CHULA ENGINEERING
Foundation toward Innovation

Unsupervised learning

2110498: AI for Engineers



Duangdao Wichadakul, Ph.D.

Department of Computer Engineering, Faculty of Engineering, Chulalongkorn University

duangdao.w@chula.ac.th

Unsupervised learning

- Find patterns in data
- E.g., clustering customers by their purchases, clustering gene expression in biological science, cluster movements of people according to their mobility / density, image segmentation
- Reduce complexity of data dimensions

Source image.

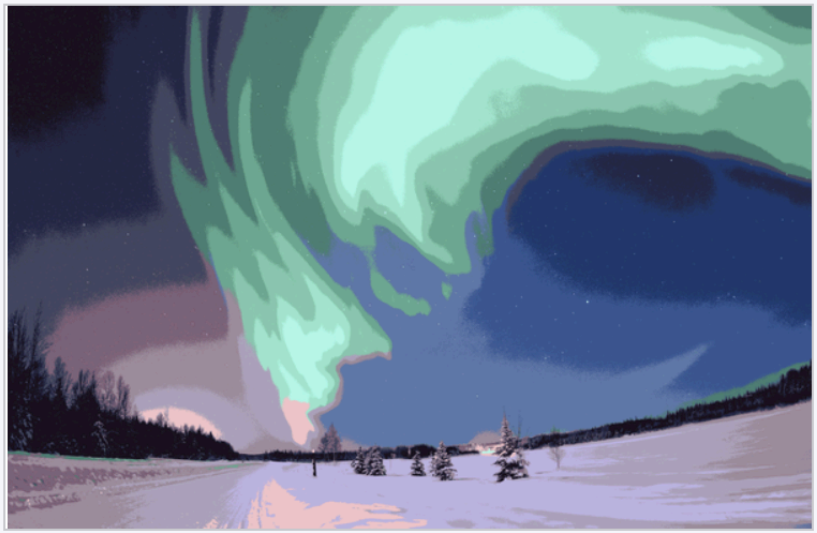


Image after running *k*-means with *k = 16*. Note that a common technique to improve performance for large images is to downsample the image, compute the clusters, and then reassign the values to the larger image if necessary.

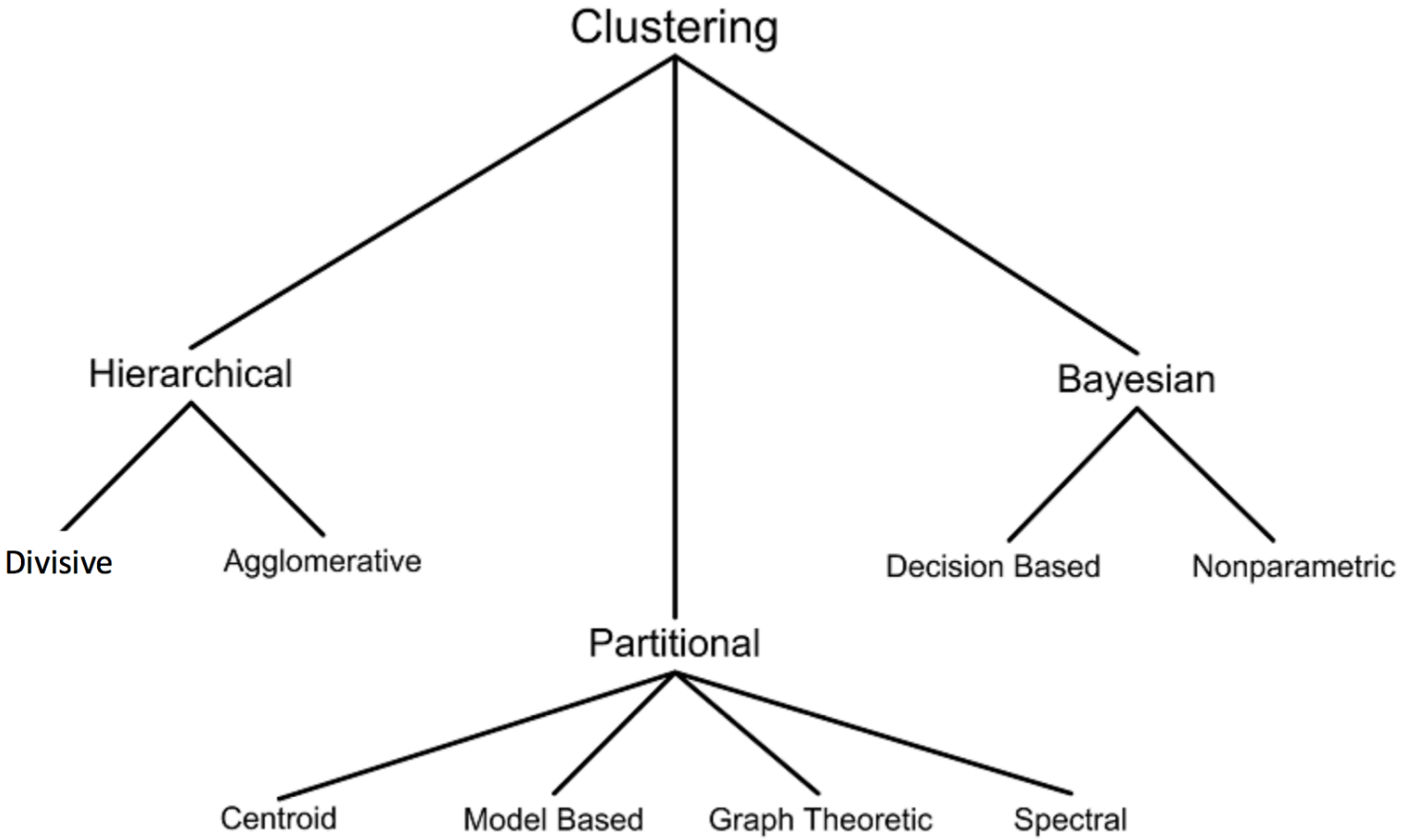
Image segmentation with K-means clustering with k = 16

https://en.wikipedia.org/wiki/Image_segmentation

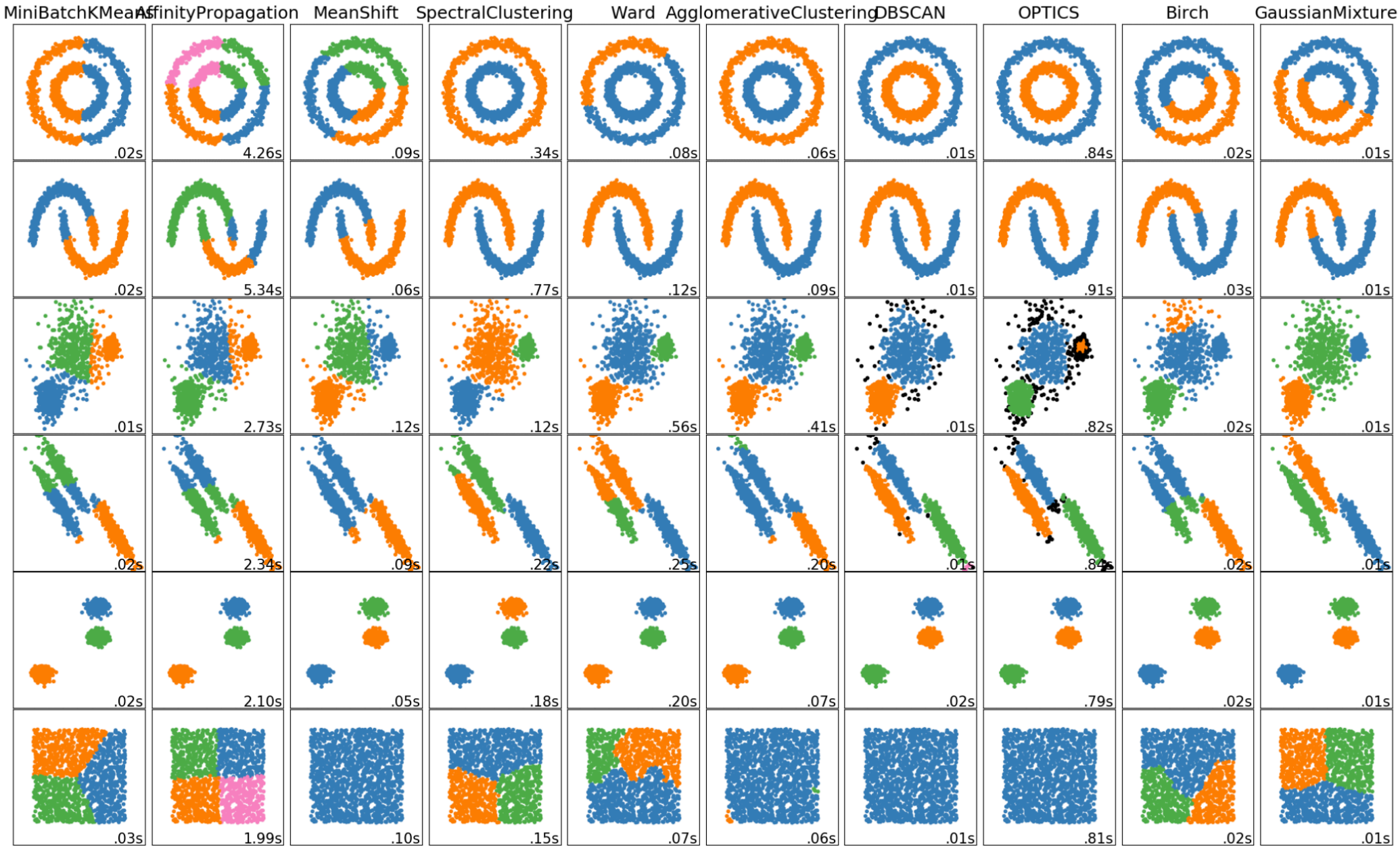
What is clustering?

- The assigning of unlabeled data into similarity groups called clusters
- A cluster contains data items which are similar between members and dissimilar from from data items in other groups

Clustering algorithms in overall



http://www.mit.edu/~9.54/fall14/slides/Class13.pdf



A comparison of the clustering algorithms in scikit-learn

https://scikit-learn.org/stable/modules/clustering.html

Outlines

- K-means clustering
- Hierarchical clustering
- PCA
- t-SNE

K-means algorithm

1. Initialize cluster for k centroids
2. Assign each data point to a cluster with the closest centroid How to measure the closest?
3. Update cluster centroids
4. Repeat steps 2-3 until the stopping condition is met.

What is the stopping condition?

How to measure the closest?

Euclidian distance or L2 distance

$$d(\mathbf{p}, \mathbf{q}) = d(\mathbf{q}, \mathbf{p}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2}$$

$$= \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}.$$

https://en.wikipedia.org/wiki/Euclidean_distance

How to measure the closest?

Euclidian distance or L2 distance

A : 7, 2, 0

c1 : 1, 5, 6

$$dist(c1,A) = (7-1)^2 + (2-5)^2 + (0-6)^2$$

$$dist(c1,A) = 36 + 9 + 36$$

c2 : 8, 1, 3

$$dist(c2,B) = (7-8)^2 + (2-1)^2 + (0-3)^2$$

$$dist(c1,A) = 1 + 1 + 9$$

c3 : 6, 7, 5

$$dist(c2,B) = (7-6)^2 + (2-7)^2 + (0-5)^2$$

$$dist(c1,A) = 1 + 25 + 25$$

How to update cluster centroids?

c2 : 8, 1, 3

Data points in c2

A : 7, 2, 0

B : 3, 3, 1

C : 4, 5, 6

D : 6, 1, 5

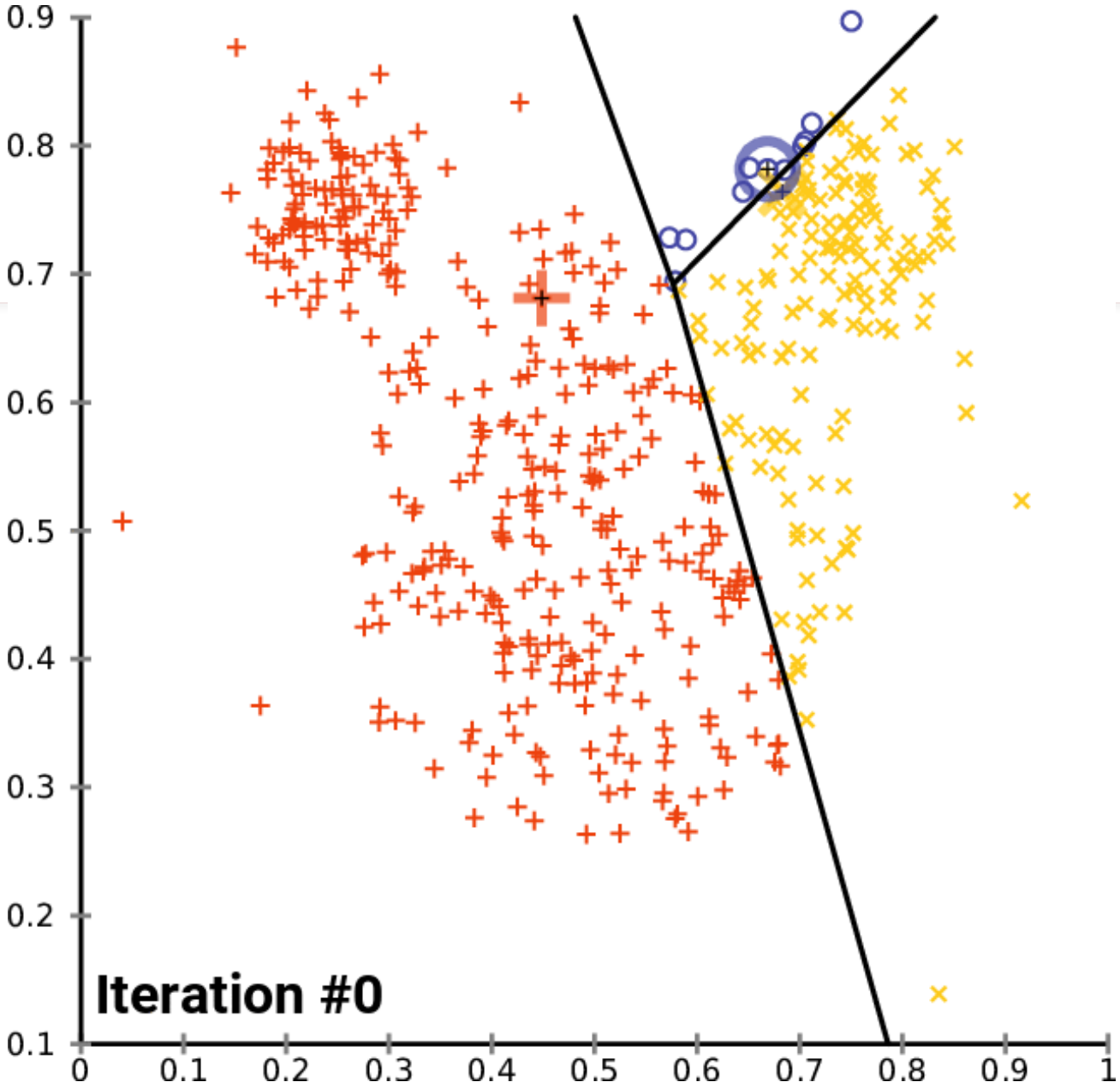
New centroid = Average of data points feature wise

$$\frac{(7+3+4+6)}{4} = 5 \qquad \frac{(2+3+5+1)}{4} = 2.75 \qquad \frac{(0+1+6+5)}{4} = 3$$

What is the stopping condition?

- The stopping criteria tells the algorithm to get out of the loop.
- The stopping criteria might not return the BEST results!!!
- Examples of stopping criteria:
  - The datapoints assigned to a specific cluster remain the same
  - Centroids remain the same
  - The distance of datapoints from their centroid in minimum (what we just set)
  - Fixed number of iterations have reached (insufficient iterations -> poor results)

Convergence of K-means clustering



https://upload.wikimedia.org/wikipedia/commons/e/ea/K-means_convergence.gif

Expectation Maximization

K-means is an EM (Expectation Maximization) algorithm applied to a particular naïve bayes model.

E-Steps: for each datapoint, choose the centroid that it is closest to by Euclidian distance.

M-Steps: take the mean of all the datapoints that were labelled with the same cluster c.

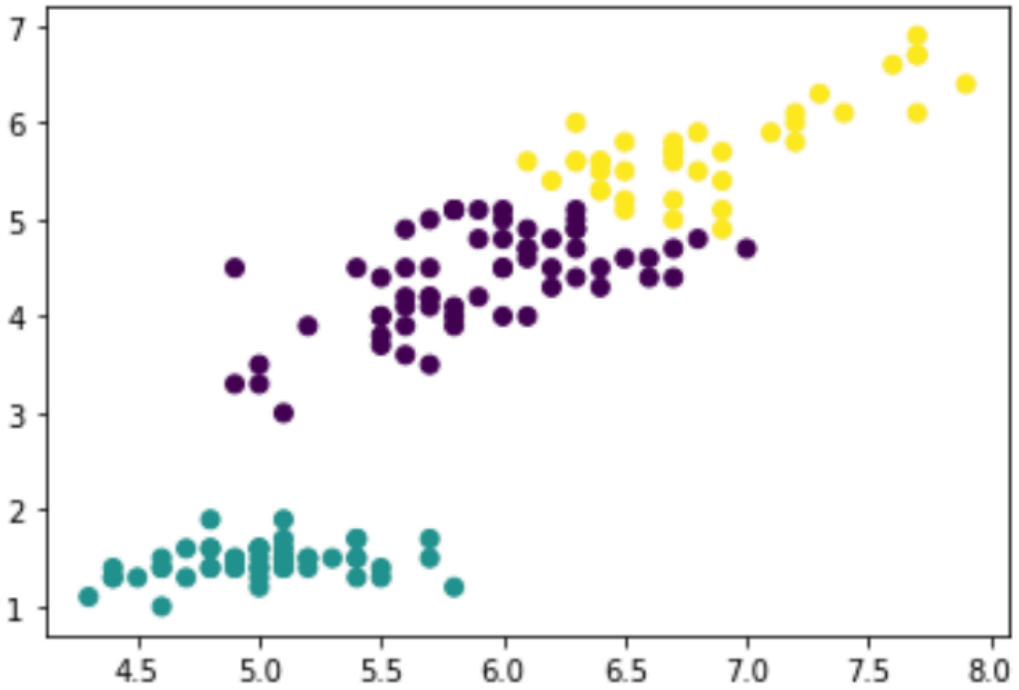
Evaluation the cluster quality

- Two methods to measure the cluster quality

1. Inertia: tells how far the datapoints within a cluster are. The lower the better (starting from 0 to grows up)
2. Silhouette score: tells how far away the datapoints in one cluster are from the datapoints in another cluster. The score is ranged from -1 to 1. Close to 1 in better.

What is the good K?

1. Use the scatter plot to visualize the distribution of your datapoints

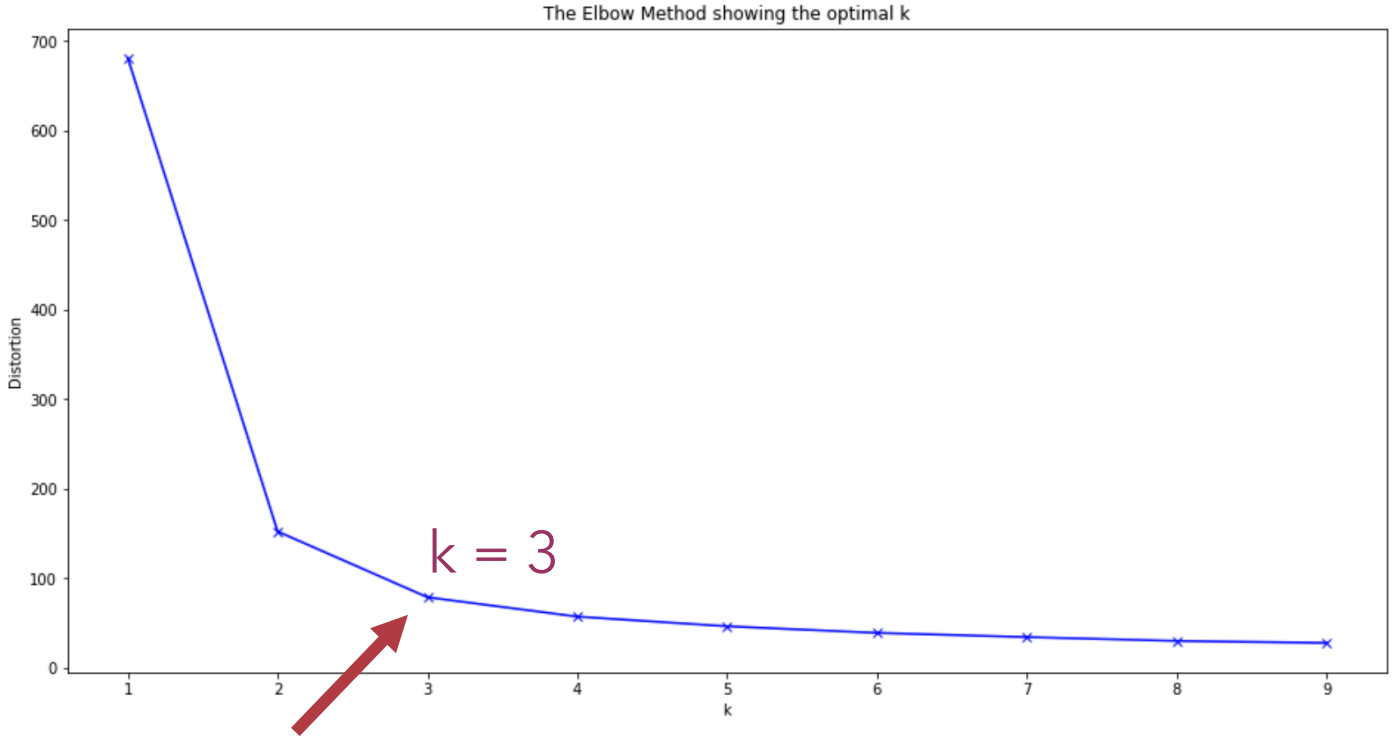


This code shows how to generate the scatter plot on the previous

```
[ ]  # Try Scatter plot
     import matplotlib.pyplot as plt
     xs = samples.data[:,0]
     print(xs)
     ys = samples.data[:,2]
     print(ys)
     print(labels)
     plt.scatter(xs, ys, c=labels)
     plt.show()
```

What is the good k?

2. Use the value of inertia. The lower inertia, the better clustering (with the small number of clusters)



This code shows how to generate the inertia graph

```
[ ]  # Run K-means for a set of k
     distortions = []
     K = range(1,10)
     for k in K:
         kmeanModel = KMeans(n_clusters=k)
         kmeanModel.fit(df)
         distortions.append(kmeanModel.inertia_)
```

```
[ ]  #Plotting the distortions of K-Means
     plt.figure(figsize=(16,8))
     plt.plot(K, distortions, 'bx-')
     plt.xlabel('k')
     plt.ylabel('Distortion')
     plt.title('The Elbow Method showing the optimal k')
     plt.show()
```

Why use K-means?
Strengths

- Simple: easy to understand and implement
- Efficient: Time complexity is O(tkn), where
  - n is the number of datapoints
  - k is the number of clusters
  - t is the number of iterations
- K-means in the most popular clustering algorithm
- It terminates at local optimum with if SSE is used.

Why use K-means?
Weakness

- We need k
- The algorithm is sensitive to outliers

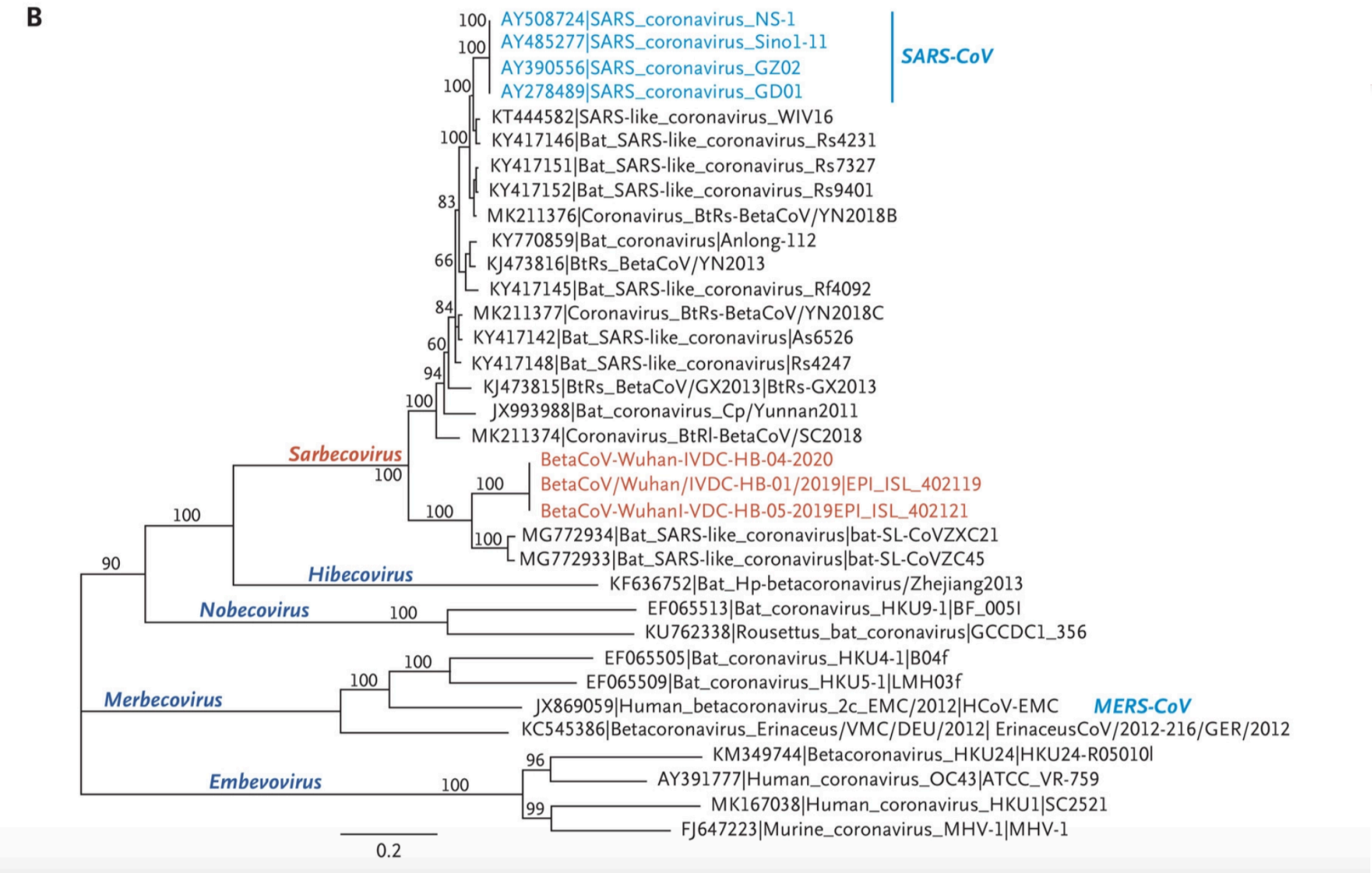
References

- https://towardsdatascience.com/k-means-clustering-from-a-to-z-f6242a314e9a
- https://stanford.edu/~cpiech/cs221/handouts/kmeans.html
- http://www.mit.edu/~9.54/fall14/slides/Class13.pdf

Agglomerative hierarchical clustering algorithm

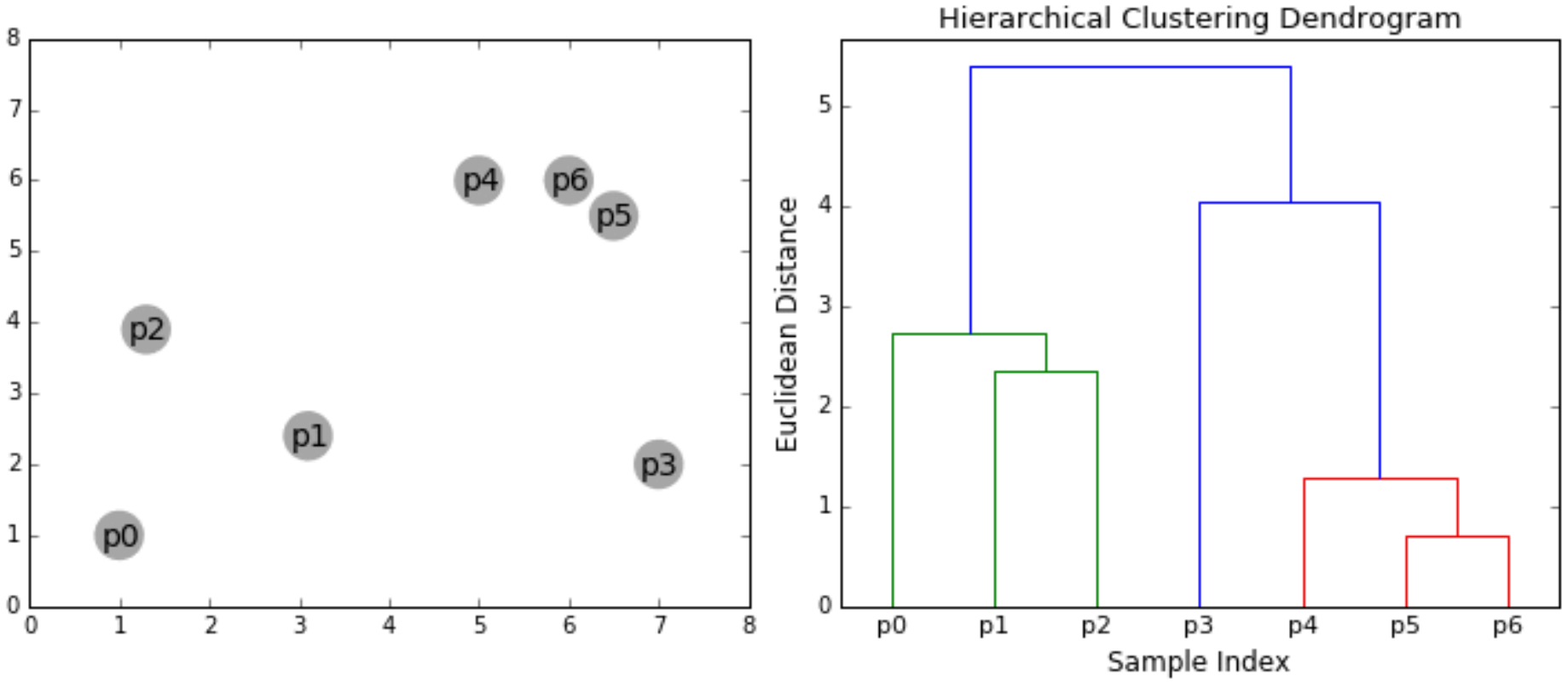
- Initialize every datapoint as its own cluster.
- Find the two closest points and combine them as a cluster.
- Find the next closet points and combine them as a cluster.
- Repeat until all datapoints form a single big cluster.

Example of hierarchical clustering



https://www.nejm.org/doi/full/10.1056/NEJMoa2001017

How the Agglomerative hierarchical clustering works



https://www.digitalvidya.com/blog/the-top-5-clustering-algorithms-data-scientists-should-know/

How to find the closest cluster?

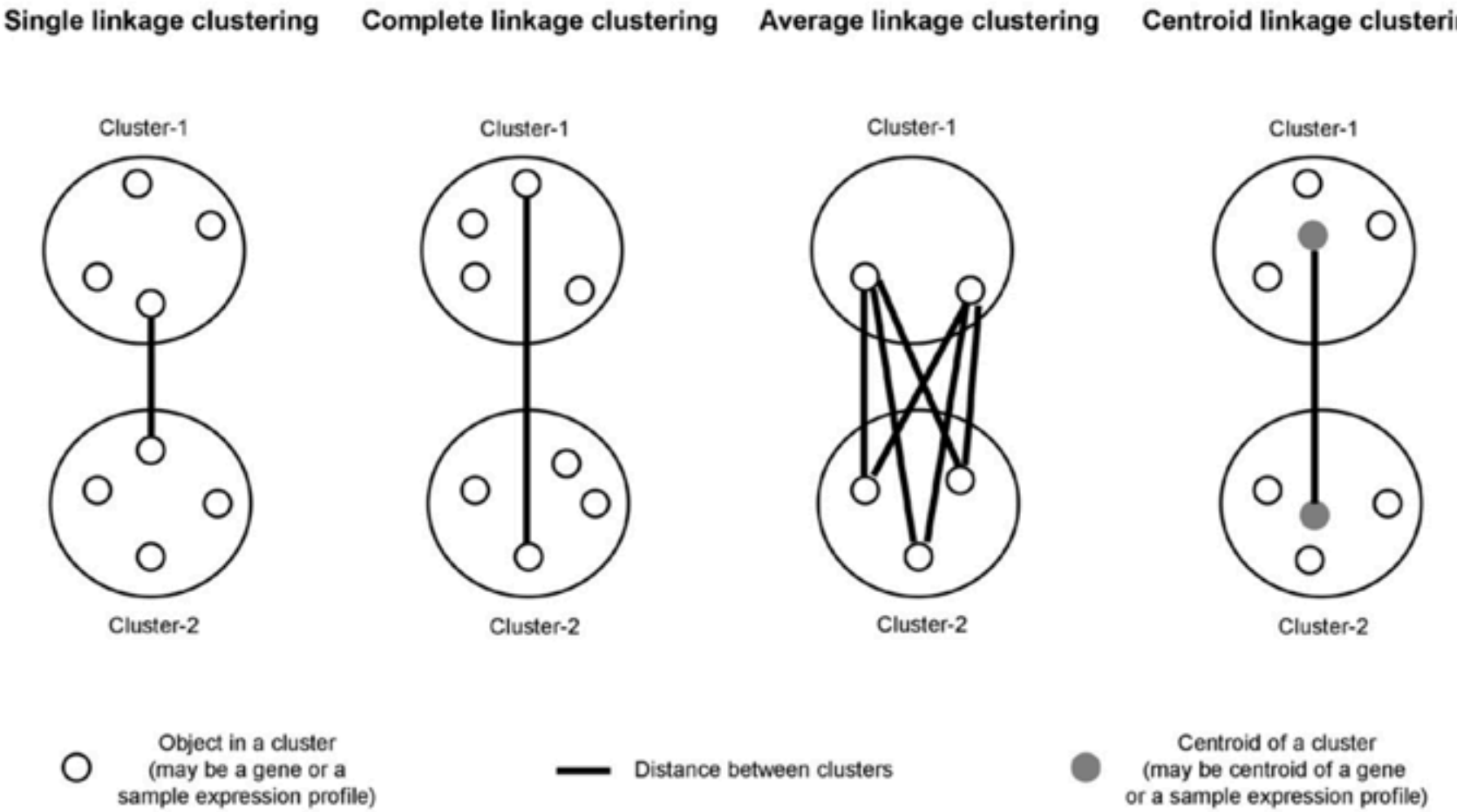


Figure 7. Different algorithms to find distance between two clusters.

Figure 7: http://www.mrc-lmb.cam.ac.uk/genomes/madanm/microarray/chapter-final.pdf

How does these method setting related to distance between clusters?

```
[ ]  # Import normalize
     from sklearn.preprocessing import normalize
     # Normalize the movements: normalized_movements
     normalized_movements = normalize(data)

     # Calculate the linkage: mergings
     mergings = linkage(normalized_movements, method='complete')
     # Plot the dendrogram
     dendrogram(mergings,
                labels=companies,
                leaf_rotation=90,
                leaf_font_size=6)
     plt.show()
```

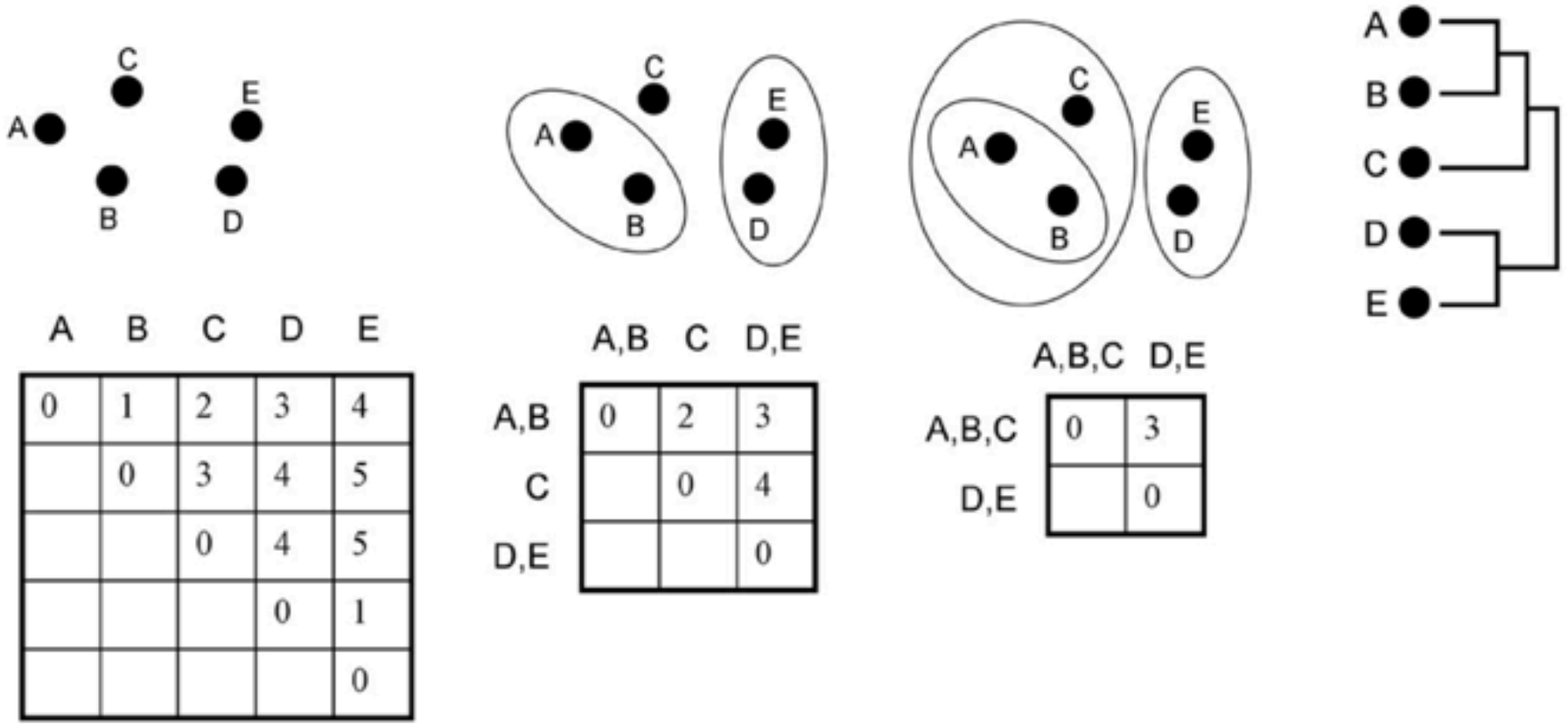
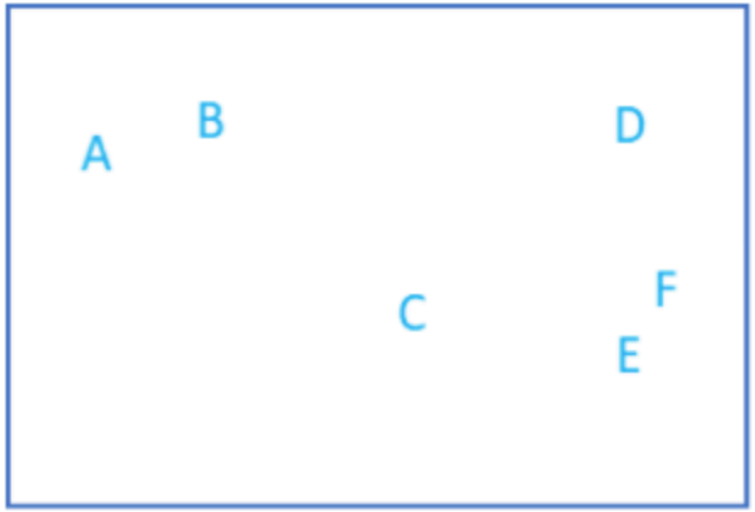


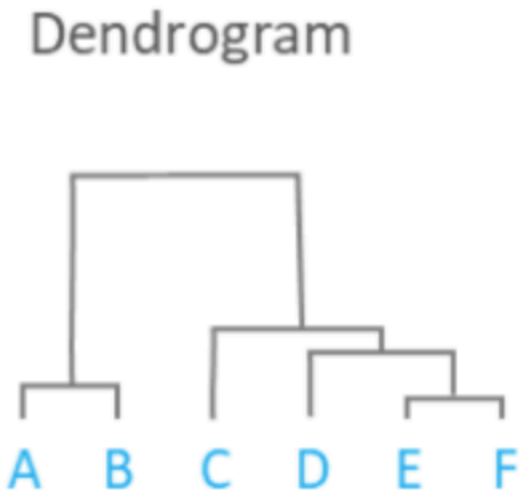
Figure 8. An example of a hierarchical clustering using single linkage algorithm. Consider five genes and the distances between them as shown in the table. In the first step, genes that are close to each other are grouped together and the distances are re-calculated using the single linkage algorithm. This procedure is repeated until all genes are grouped into one cluster. This information can be represented as a tree (shown to the right), where the distance from the branch point reflects the distance between genes or clusters. This image was adapted from Causton *et al*. (2003).

Figure 8: http://www.mrc-lmb.cam.ac.uk/genomes/madanm/microarray/chapter-final.pdf

Dendrogram



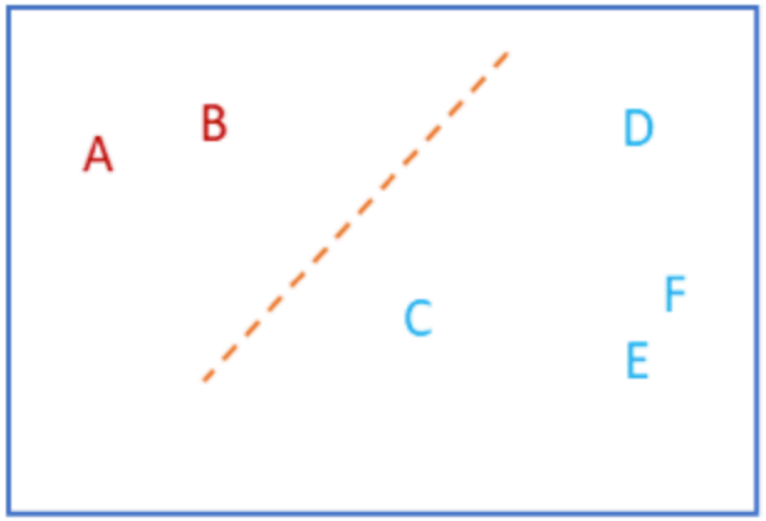
- It is a summary of distance matrix.
- As occurs with most summaries, information is lost.
- B is not that far from C in the real data but not in the dendrogram.
- So, it is most accurate at the leaf nodes.

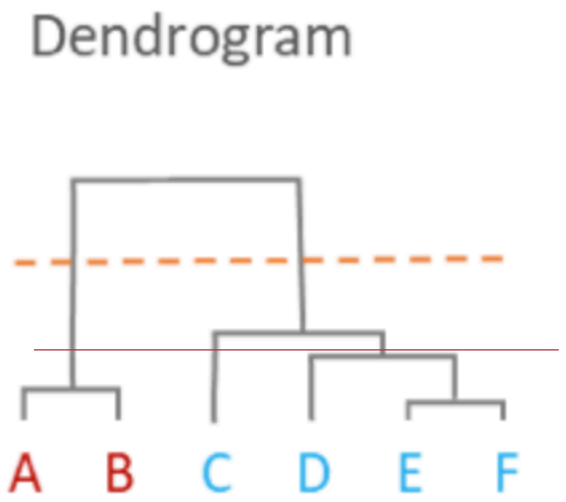


https://www.displayr.com/what-is-dendrogram/

Dendrogram (more)

- Observations are allocated to clusters by drawing a horizontal line through the dendrogram.
- Dendrogram cannot tell how many clusters we should have.





PCA

- PCA = “Principal Component Analysis”
- Fundamental dimension reduction technique

PCA aligns data with axes

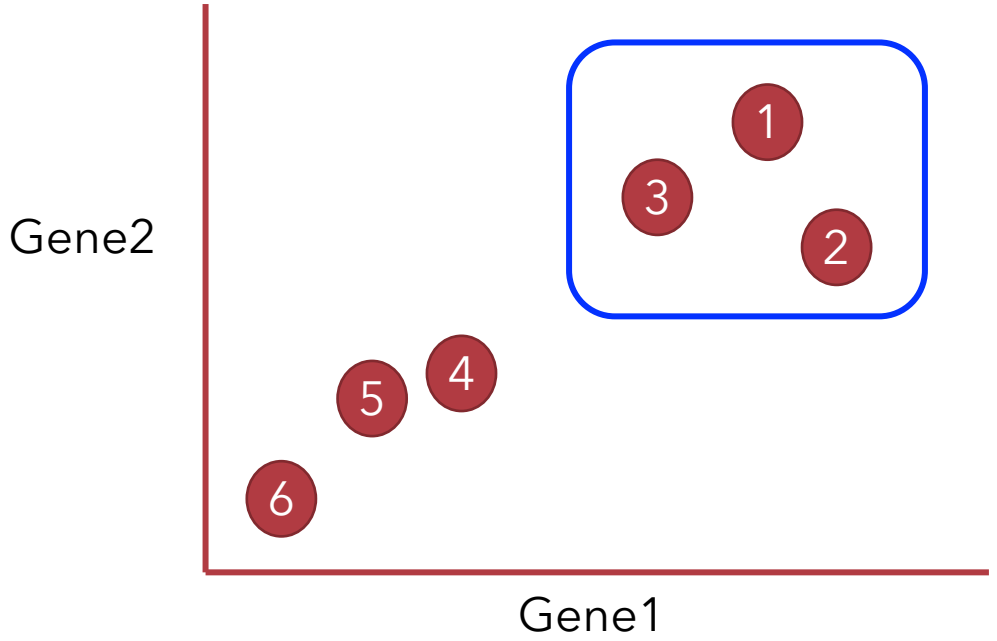
	Business 1	Business 2	Business 3	…
Market cap	9.5 M	88 M	93 M	…
#Employees	960	79	9800	…

	Mouse 1	Mouse 2	Mouse 3	Mouse 4	Mouse 5	Mouse 6
Gene1	10	11	8	3	2	1

PCA aligns data with axes

If we measure two genes (2 variables)

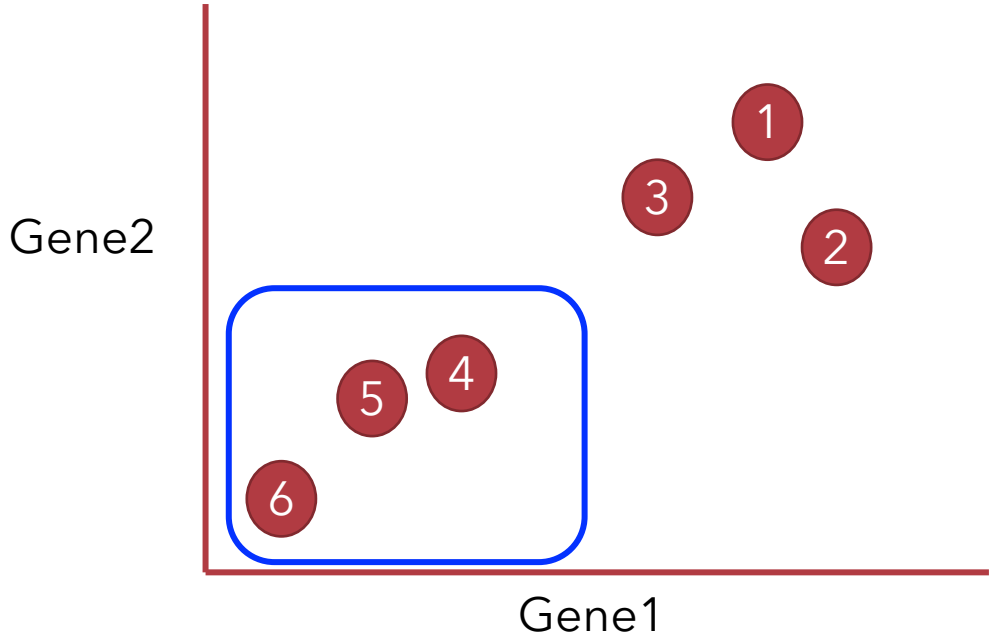
	Mouse 1	Mouse 2	Mouse 3	Mouse 4	Mouse 5	Mouse 6
Gene1	10	11	8	3	2	1
Gene2	6	4	5	3	2.8	1



PCA aligns data with axes

If we measure two genes (2 variables)

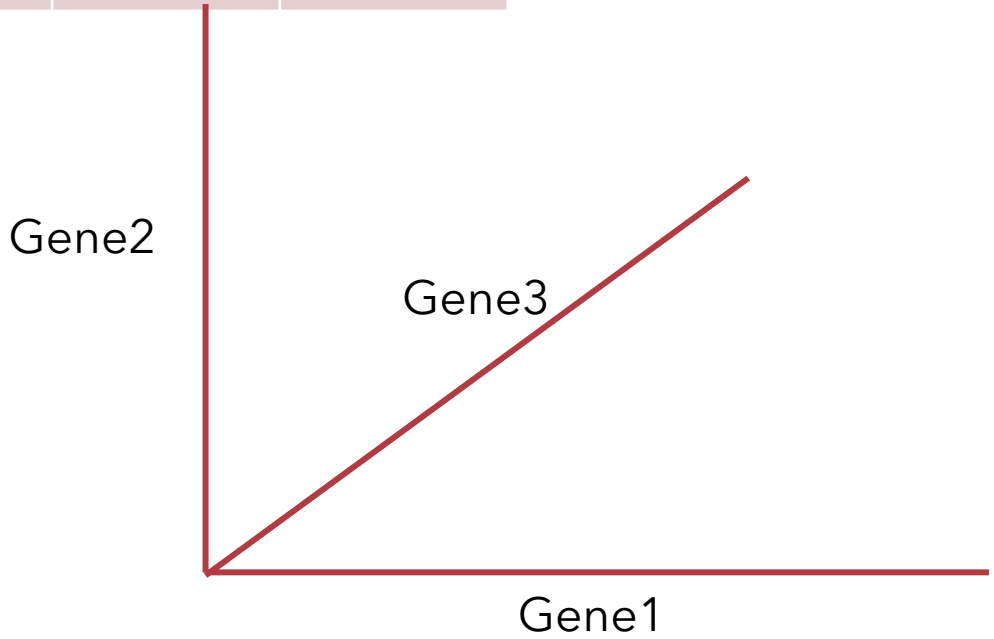
	Mouse 1	Mouse 2	Mouse 3	Mouse 4	Mouse 5	Mouse 6
Gene1	10	11	8	3	2	1
Gene2	6	4	5	3	2.8	1



PCA aligns data with axes

If we measure two genes (2 variables)

	Mouse 1	Mouse 2	Mouse 3	Mouse 4	Mouse 5	Mouse 6
Gene1	10	11	8	3	2	1
Gene2	6	4	5	3	2.8	1
Gene3	12	9	10	2.5	1.3	2



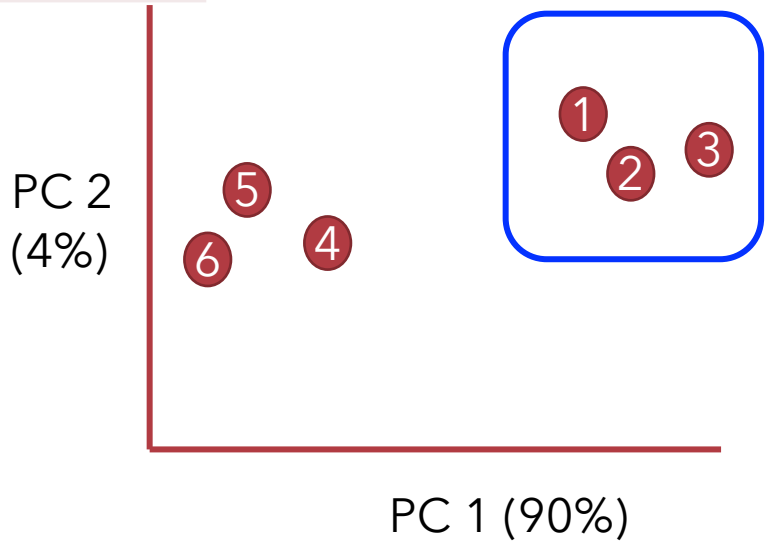
PCA aligns data with axes

If we measure two genes (2 variables)

	Mouse 1	Mouse 2	Mouse 3	Mouse 4	Mouse 5	Mouse 6
Gene1	10	11	8	3	2	1
Gene2	6	4	5	3	2.8	1
Gene3	12	9	10	2.5	1.3	2
Gene4	5	7	6	2	4	7

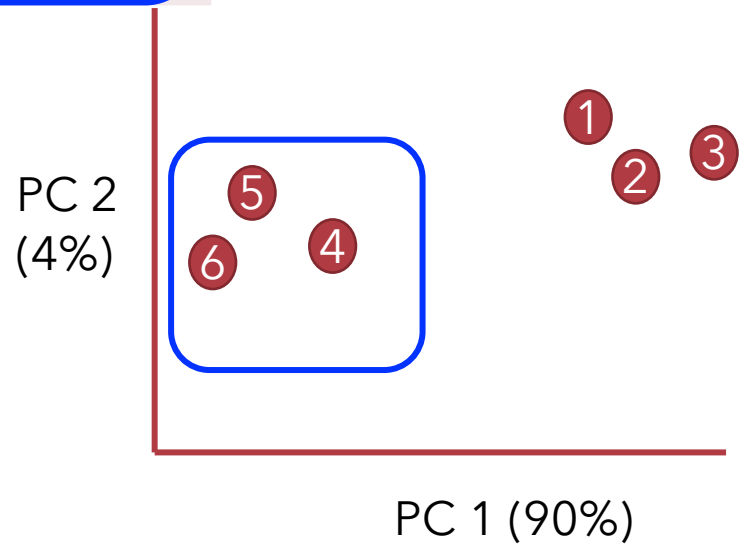
How PCA can take 4 or more feature measurements and make a 2D PCA plot

	Mouse 1	Mouse 2	Mouse 3	Mouse 4	Mouse 5	Mouse 6
Gene1	10	11	8	3	2	1
Gene2	6	4	5	3	2.8	1
Gene3	12	9	10	2.5	1.3	2
Gene4	5	7	6	2	4	7

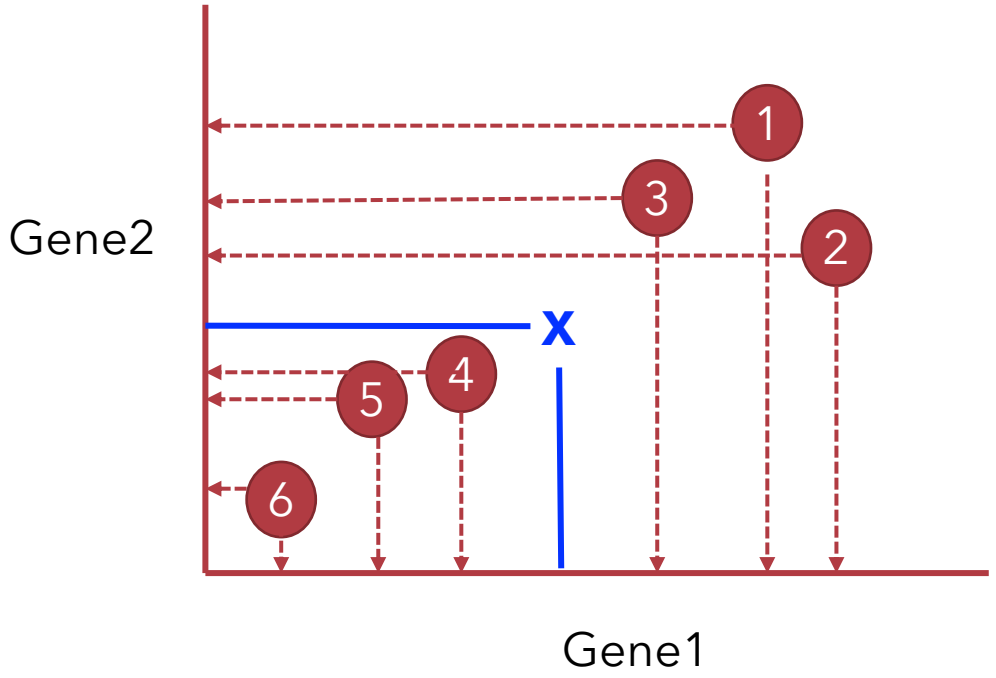


How PCA can take 4 or more feature measurements and make a 2D PCA plot

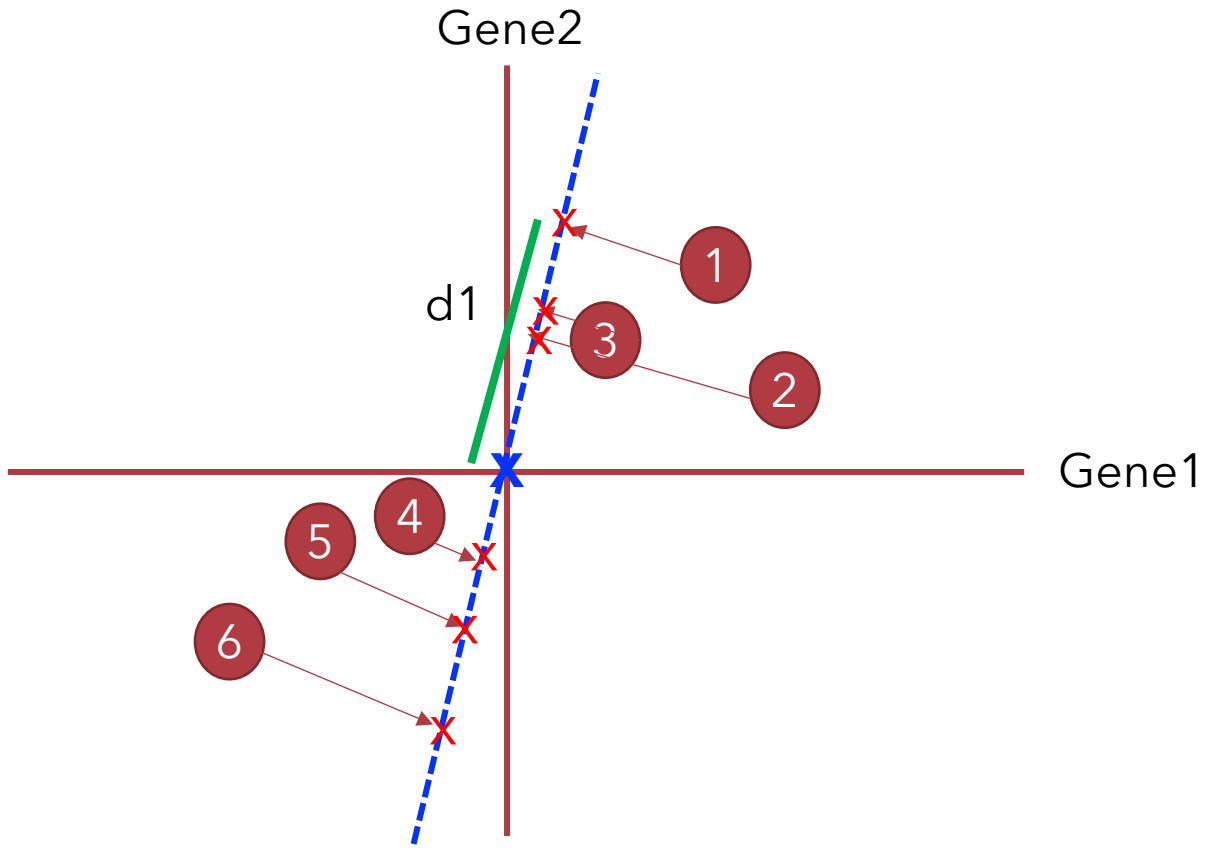
	Mouse 1	Mouse 2	Mouse 3	Mouse 4	Mouse 5	Mouse 6
Gene1	10	11	8	3	2	1
Gene2	6	4	5	3	2.8	1
Gene3	12	9	10	2.5	1.3	2
Gene4	5	7	6	2	4	7



PCA aligns data with axes
If we measure two genes (2 variables)

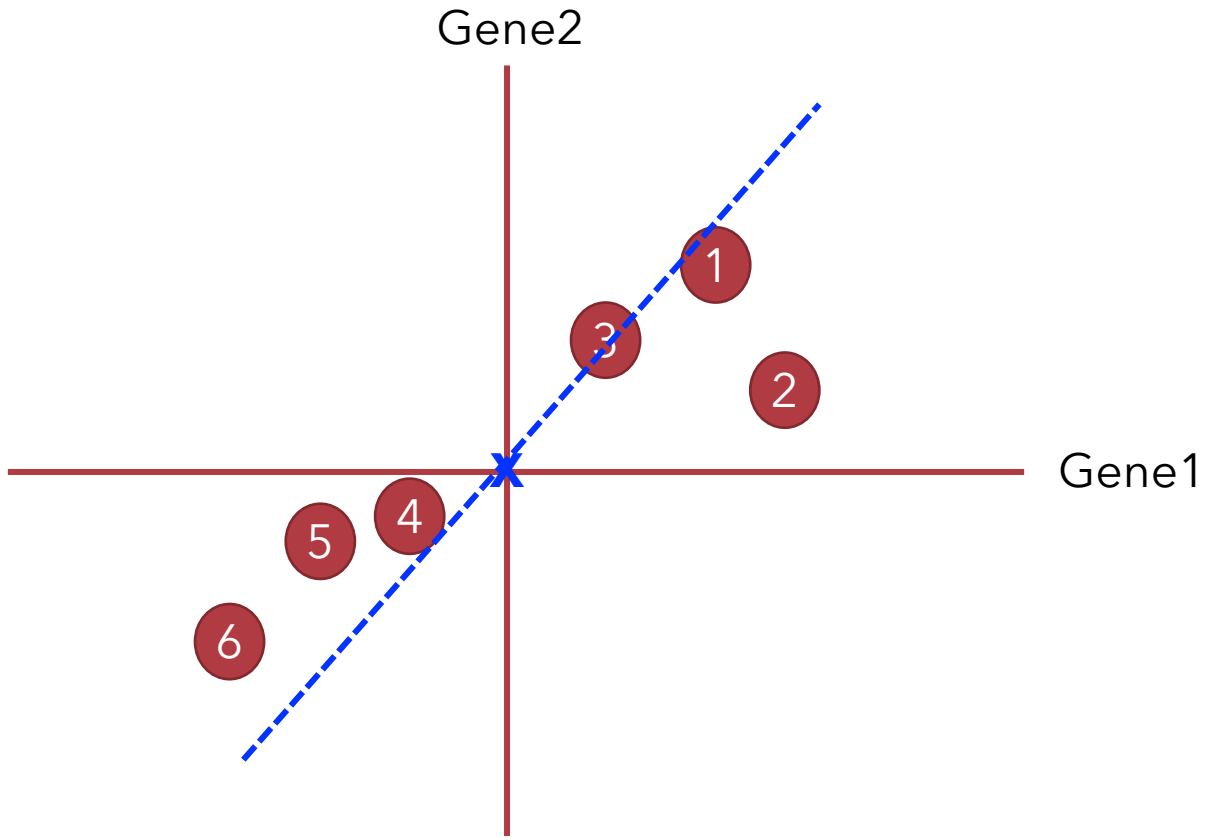


PCA aligns data with axes
If we measure two genes (2 variables)

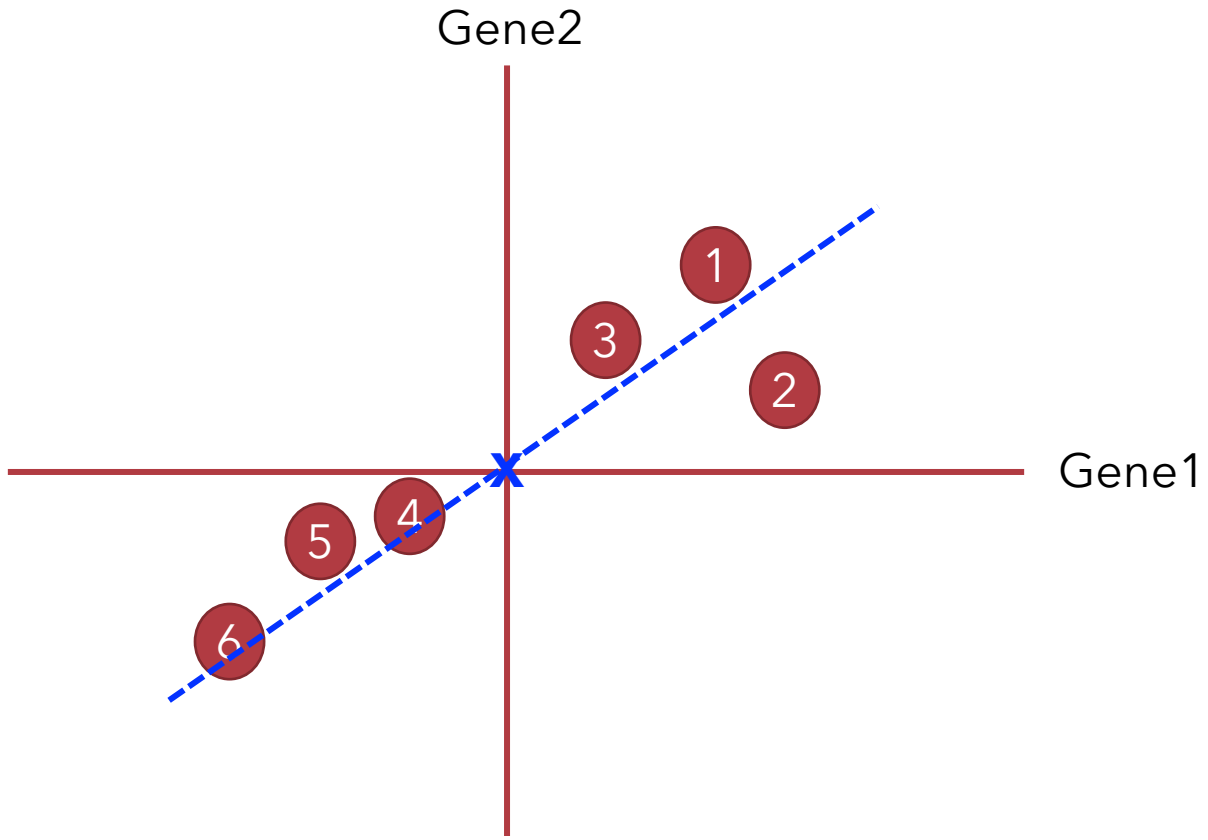


$d1^2 + d2^2 + d3^2 + d4^2 + d5^2 + d6^2$ = sum of squared distance = SS(distances)

PCA aligns data with axes
If we measure two genes (2 variables)

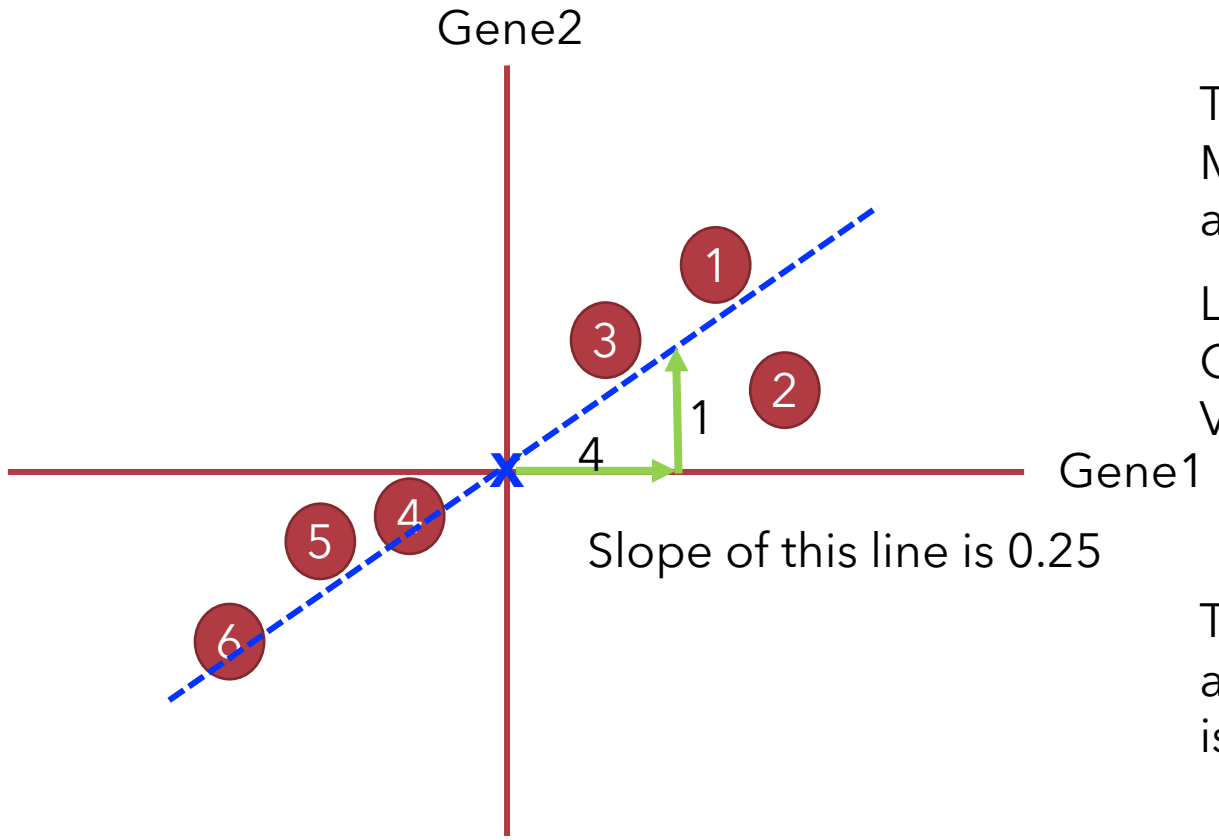


PCA aligns data with axes
If we measure two genes (2 variables)



Largest sum of squared distance = SS(distances) => Principal Component 1 (PC1)

PCA aligns data with axes
If we measure two genes (2 variables)



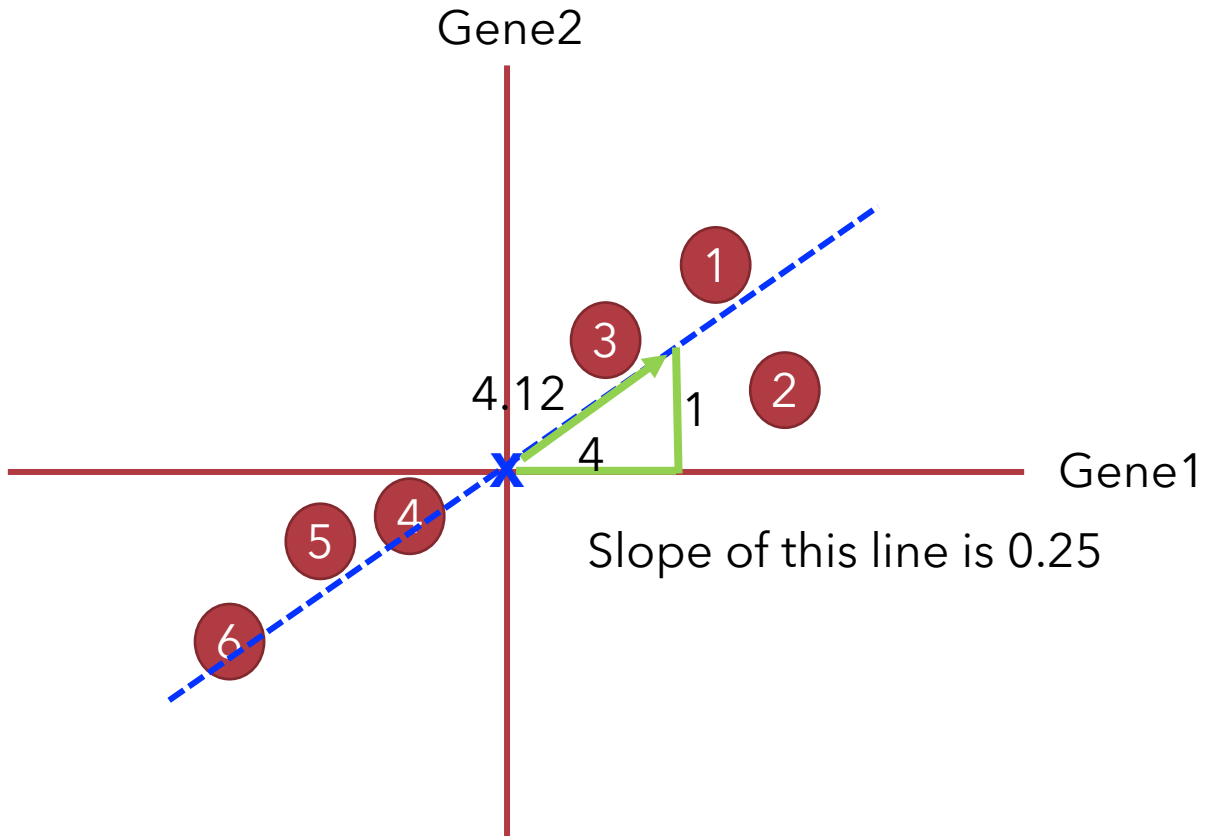
To make PC1
Mix 4 parts of Gene1
and 1 part of Gene 2

Linear combination between
Gene1 and Gene2
Variable and

The ratio of Gene1
and Gene2 indicates that Gene1
is more important than Gene2

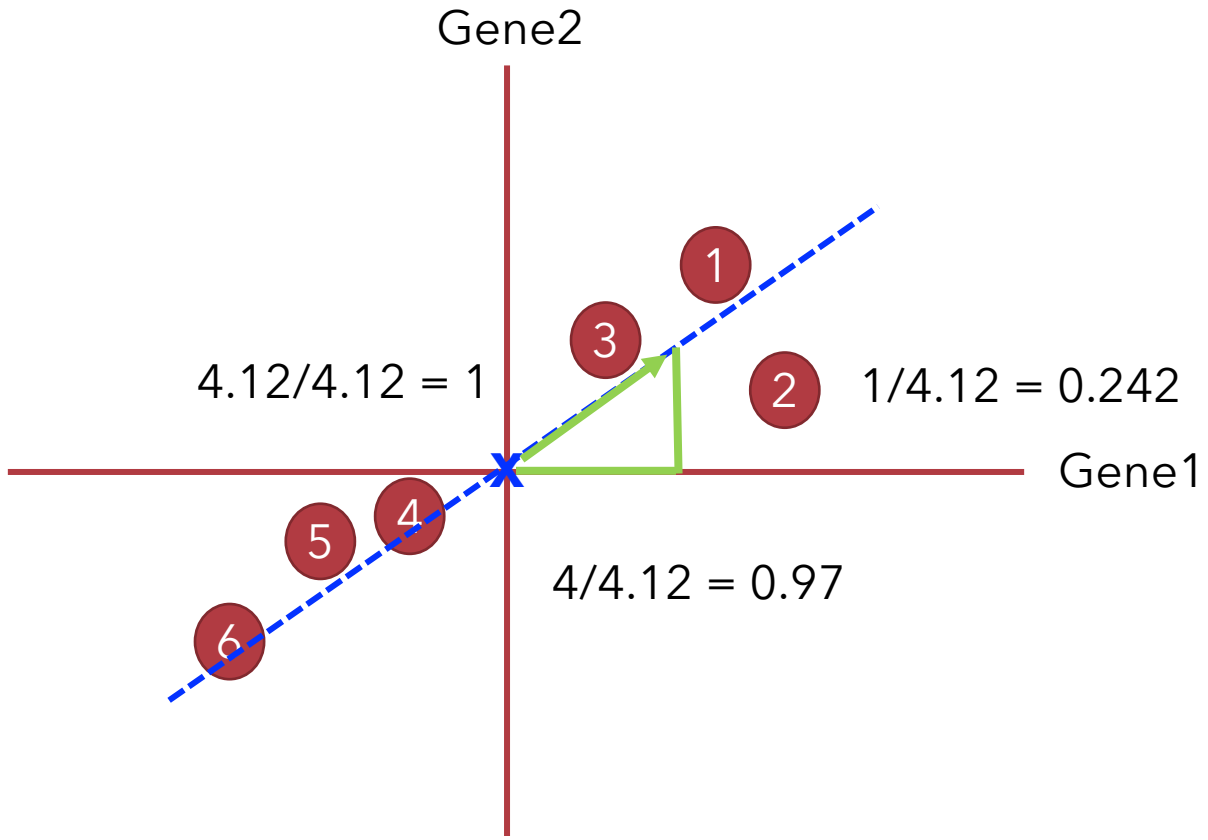
Largest sum of squared distance = SS(distances) => Principal Component 1 (PC1)

PCA aligns data with axes
If we measure two genes (2 variables)



Largest sum of squared distance = SS(distances) => Principal Component 1 (PC1)

PCA aligns data with axes
If we measure two genes (2 variables)



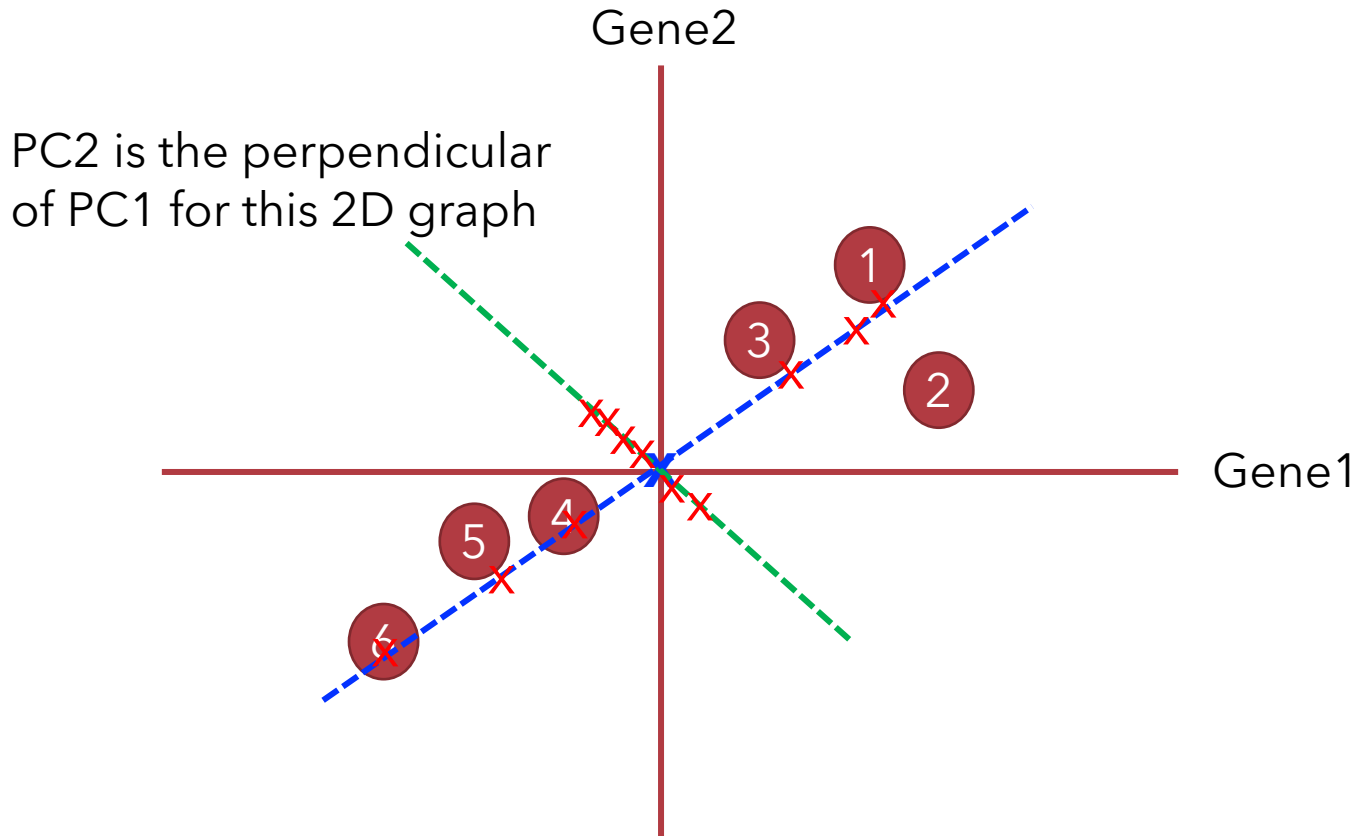
To make PC1
Mix 0.97 of Gene1
and 0.242 of Gene 2

1 unit long vector of
0.97 parts Gene1 and
0.242 parts Gene2 is called
"Singular Vector" or the
"Eigenvector" for PC1

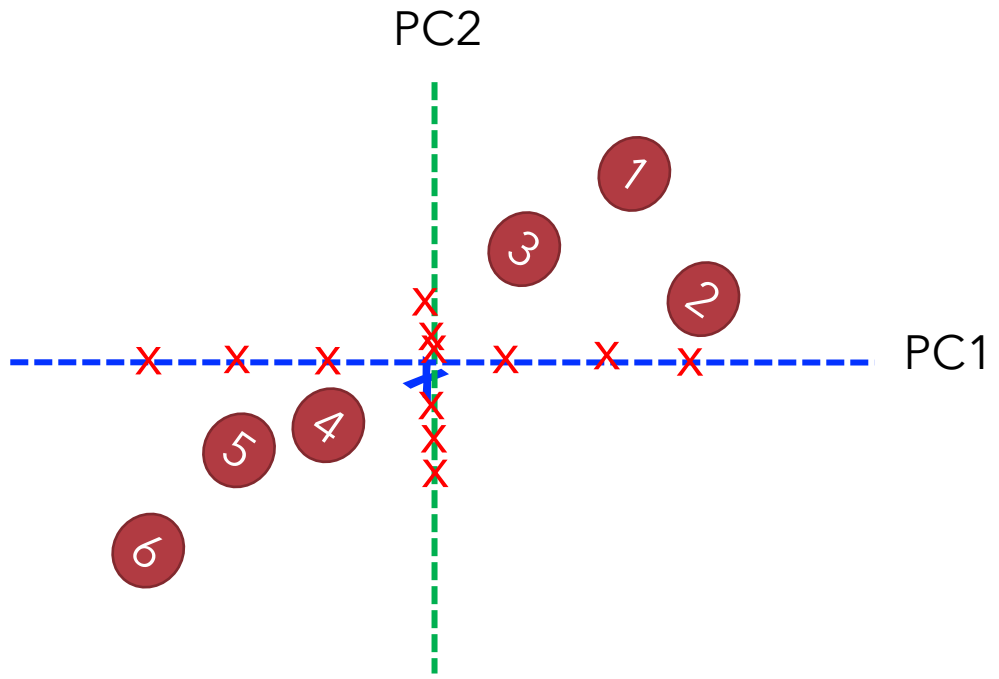
0.97 and 0.242 are
called loading scores

Largest sum of squared distance = SS(distances) => Principal Component 1 (PC1) = Eigenvalue for PC1

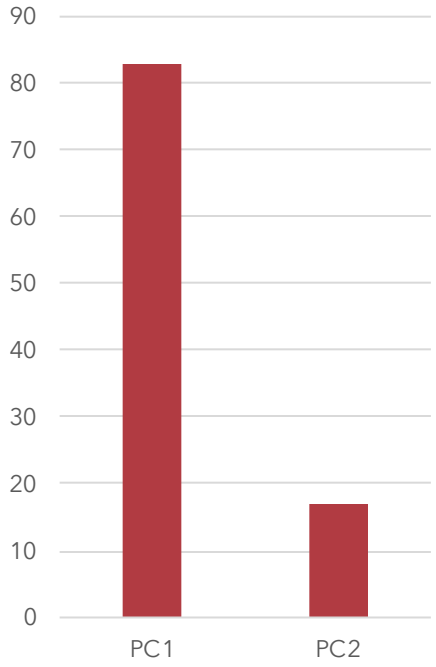
PCA aligns data with axes
If we measure two genes (2 variables)



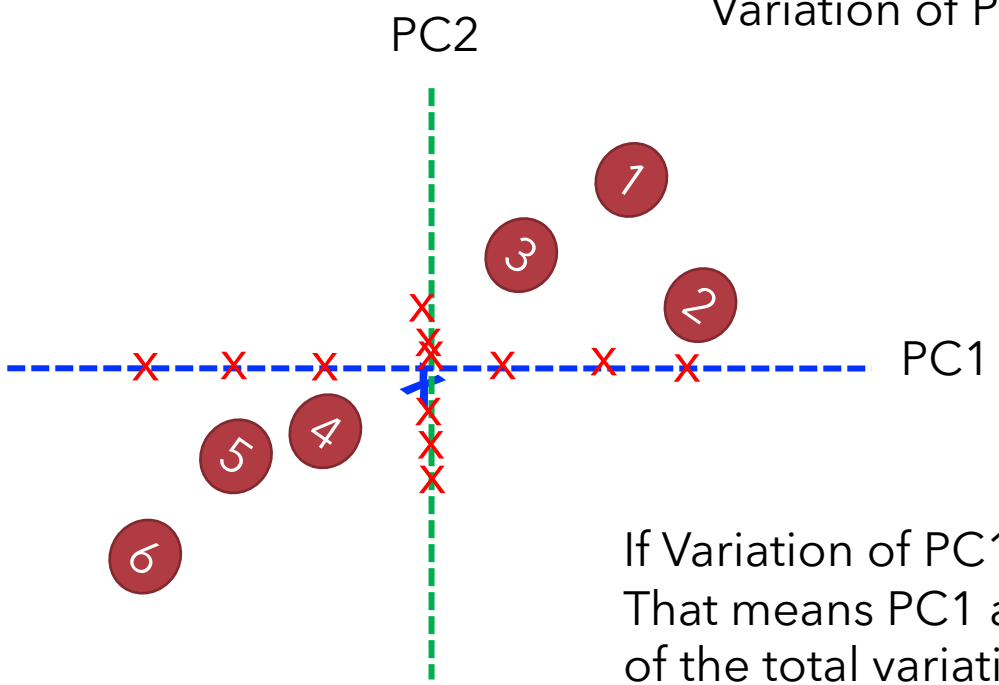
PCA aligns data with axes
If we measure two genes (2 variables)



Variation for the PCs



Scree plot



Variation of PC1 = SS(distance for PC1) / (n-1)
Variation of PC2 = SS(distance for PC2) / (n-1)

If Variation of PC1 = 15 and Variation of PC2 = 3
That means PC1 accounts for 15/18 = 0.83 or 83%
of the total variation around the PCs.
PC2 accounts for 3/18 = 0.17 or 17% in this example.

```
[ ] # Perform the necessary imports
    from sklearn.decomposition import PCA
    from sklearn.preprocessing import StandardScaler
    from sklearn.pipeline import make_pipeline
    import matplotlib.pyplot as plt

    # Create scaler: scaler
    scaler = StandardScaler()

    # Create a PCA instance: pca
    pca = PCA()

    # Create pipeline: pipeline
    pipeline = make_pipeline(scaler, pca)

    # Fit the pipeline to 'samples'
    pipeline.fit(wine_data)

    # Plot the explained variances
    features = range(pca.n_components_)
    print(features)
    plt.bar(features, pca.explained_variance_)
    plt.xlabel('PCA feature')
    plt.ylabel('variance')
    plt.xticks(features)
    plt.show()
```

How do these attributes related to the PCA described in slides?

References

- StatQuest: Principal Component Analysis (PCA), Step-by-Step (https://www.youtube.com/watch?v=FgakZw6K1QQ)
- https://towardsdatascience.com/an-approach-to-choosing-the-number-of-components-in-a-principal-component-analysis-pca-3b9f3d6e73fe

t-SNE

- Takes a high dimensional dataset and reduces it to a low dimensional graph and still preserves a lot of original information.

Watch the StatQuest: t-SNE clearly explained

- https://www.youtube.com/watch?v=NEaUSP4YerM

t-SNE algorithm

For each point ***a***

1. Measure the distance between point *a* to another point ***b***.
2. Plot that distance on a t-distribution curve with point ***a*** centered.
3. Draw a line from a point ***b*** to the curve. The length of this line called "unscaled similarity."
4. Sum of all ***bs'*** "unscaled similarity" and scaled to the total of 1 for this ***a***.

Why do we need to scaled them to 1?

Is it better than PCA or not? and Why?

Additional exploration

- https://www.datacamp.com/community/tutorials/introduction-t-sne