College Dublin
Computing • IT • Business

Machine Learning for Data Analysis
MSc in Data Analytics
CCT College Dublin

Unsupervised Learning (PCA and k-Means)
Week 5

Lecturer: Dr. Muhammad Iqbal*

Email: miqbal@cct.ie

Agenda

- Introduction to Unsupervised Learning
- Principle Component Analysis (PCA)
- Reducing Dimensionality of Data
- Clustering
- Clustering Criterion: Elbow Method and The Silhouette Method
- Types of Clustering
- Feature Extraction Example
- Comparison of PCA and Kmeans Clustering

Introduction to
Unsupervised Learning

- Unsupervised learning means learning by observation, not by example or labeled data. This type of learning works with unlabelled data. **Dimensionality reduction** and **clustering** are examples of such learning.

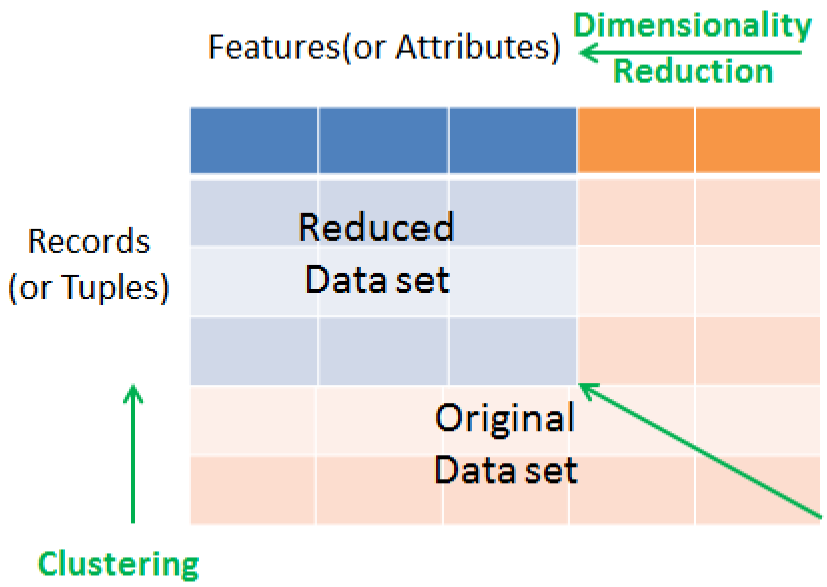
- **Dimensionality reduction** is used to reduce a large number of attributes to a few that can produce the same results.

- There are several methods that are available for reducing the dimensionality of data, such as **principal component analysis (PCA)**, **t-SNE**, **wavelet transformation**, and attribute subset selection.

- The term cluster means a group of similar items that are closely related to each other.

- We can say that a cluster is a set of data points that are similar to others in its cluster and dissimilar to data points of other clusters.

- **Clustering** has numerous applications, such as in searching documents, business intelligence, information security, and recommender systems.



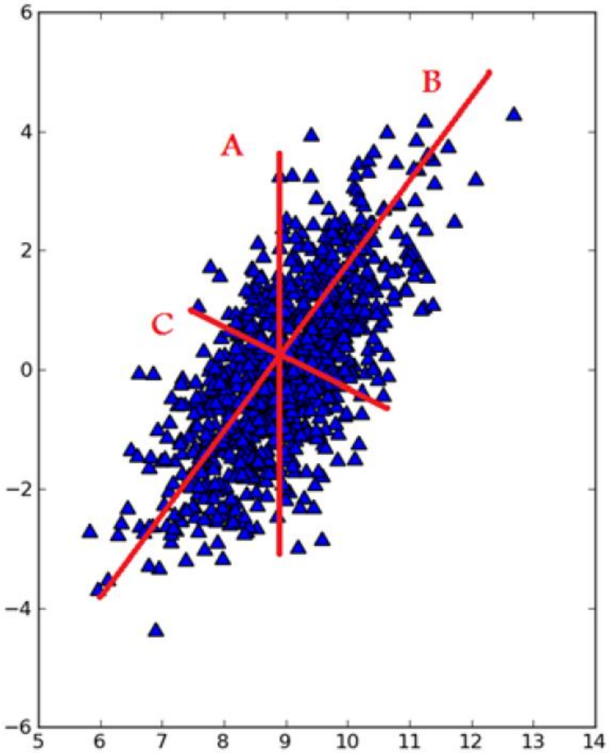
- we can see how clustering puts data records or observations into a few groups, and dimensionality reduction reduces the number of features or attributes.

Introduction to
Unsupervised Learning

- Suppose you are watching a sports match involving a ball on a flat screen TV. The TV probably contains a million pixels, and the ball is represented by, say, a thousand pixels.
- In most sports, we are concerned with the position of the ball at a given time. For your brain to follow what’s going on, you need to follow the position of the ball on the playing field. You do this naturally, without even thinking about it.
- Behind the scene, you are converting the million pixels on the monitor into a three-dimensional image showing the ball’s position on the playing field, in real time.
- You have reduced the data from one million dimensions to three. In this sports match example, you are presented with millions of pixels, but it’s the ball’s three-dimensional position that’s important. **This is known as dimensionality reduction**.
- You are reducing data from more than one million values to the three relevant values. It is much easier to work with data in fewer dimensions.
- We have to identify relevant features before we can begin to apply machine learning algorithms.

Principle Component Analysis (PCA)

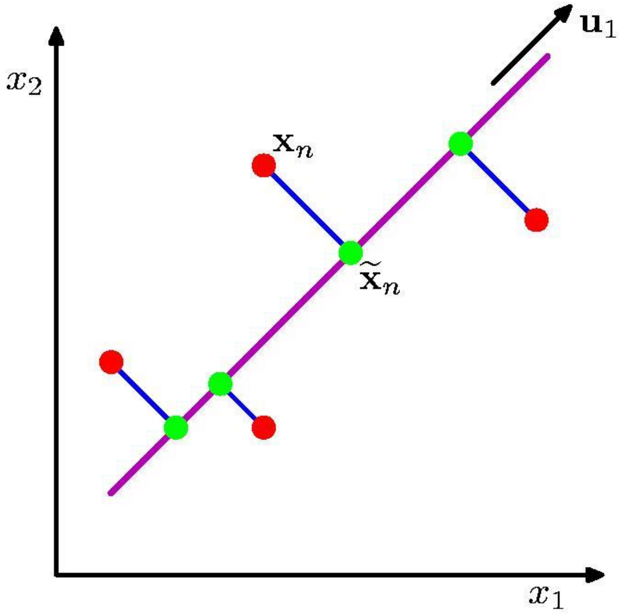
- The first method for **dimensionality reduction** is called principal component analysis (PCA).
- In PCA, the dataset is transformed from its original coordinate system to a new coordinate system.
- The new coordinate system is chosen by the data itself. The first new axis is chosen in the direction of the most variance in the data.
- The second axis is orthogonal to the first axis and in the direction of an orthogonal axis with the largest variance.
- This procedure is repeated for as many features as we had in the original data.
- We find that the majority of the variance is contained in the first few axes. Therefore, we can ignore the rest of the axes, and we reduce the dimensionality of our data.



- Three choices for lines that span the entire dataset. Line B is the longest and accounts for the most variability in the dataset.

Reducing Dimensionality of Data

- **Dimensionality Reduction** entails scaling down a large number of attributes or columns (features) into a smaller number of attributes.
- The main objective of this technique is to get the best number of features for classification, regression, and other unsupervised approaches.
- Techniques for **linear transformations** include **PCA**, Linear Discriminant analysis (**LDA**), and Factor Analysis.
- **Non-linear transformations** include techniques such as t-SNE, Hessian eigenmaps and isometric feature mapping. Dimensionality reduction offers the following benefits
  - It filters redundant and less important features.
  - It reduces model complexity with less dimensional data.
  - It reduces memory and computation costs for model generation.
  - It visualizes high-dimensional data.



- Maximizes the variance of the projected line (purple).
- Minimizes the MSE between the data points and projections (blue).

Principle Component Analysis (PCA)

- In machine learning, it is considered that having a large amount of data means having a good-quality model for prediction, but a large dataset also poses the challenge of higher dimensionality.

- It causes an increase in complexity for prediction models due to the large number of attributes.

- For example, if we have 200 attributes or columns in our data, it becomes very difficult for us to proceed, what with such a huge number of attributes. In such cases, we need to reduce that number to 10 or 20 variables.

- Another objective of **PCA** is to reduce the dimensionality without affecting the significant information. For p-dimensional data, the **PCA** equation can be written as follows

$$PC_j = w_{1j}X_1 + w_{2j}X_2 + \ldots\ldots + w_{pj}X_p$$

Principal Components are a weighted sum of all the attributes. Here, $X_1, X_2, X_3 \ldots.X_p$ are the attributes in the original dataset and $w_{1j},\ w_{2j},\ w_{3j}, \ldots.w_{pj}$ are the weights of the attributes.

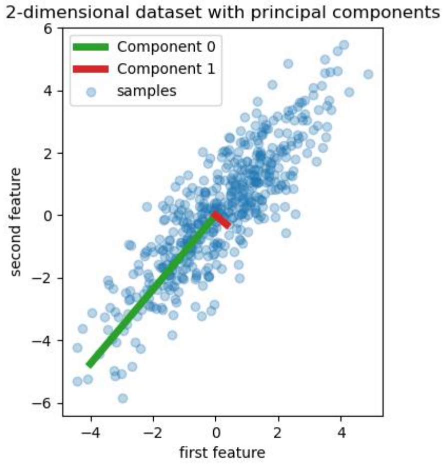
Principle Component Analysis

Example

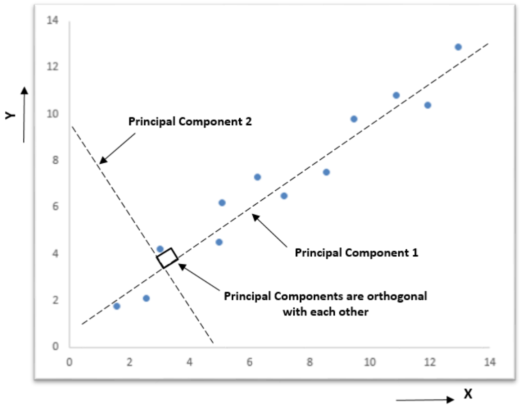
- Suppose we consider the streets in a given city as attributes, and let's say you want to visit this city. Now the question is, how many streets you will visit? Obviously, you would like to visit the popular or main streets of the city, which let's say is 10 out of the 50 available streets. These 10 streets will give you the best understanding of that city. These streets are principal components, as they explain enough of the variance in the data (the city's streets).

- **Performing PCA**

  1. Compute the **correlation** or **covariance** matrix of a given dataset.
  2. Find the **eigenvalues** and **eigenvectors** of the correlation or covariance matrix.
  3. Multiply the eigenvector matrix by the original dataset and you will get the principal component matrix.



PCA illustration of 2-Dimensional Data with 2 Principal Components



$$\sigma_{ij} = \frac{1}{n-1}\sum_{k=1}^{n}(X_{ki}-\bar{X}_i)(X_{kj}-\bar{X}_j)$$

Clustering

- Finding groups of objects such that the objects in a group will be similar (or related) to one another and different from (or unrelated to) the objects in other groups.

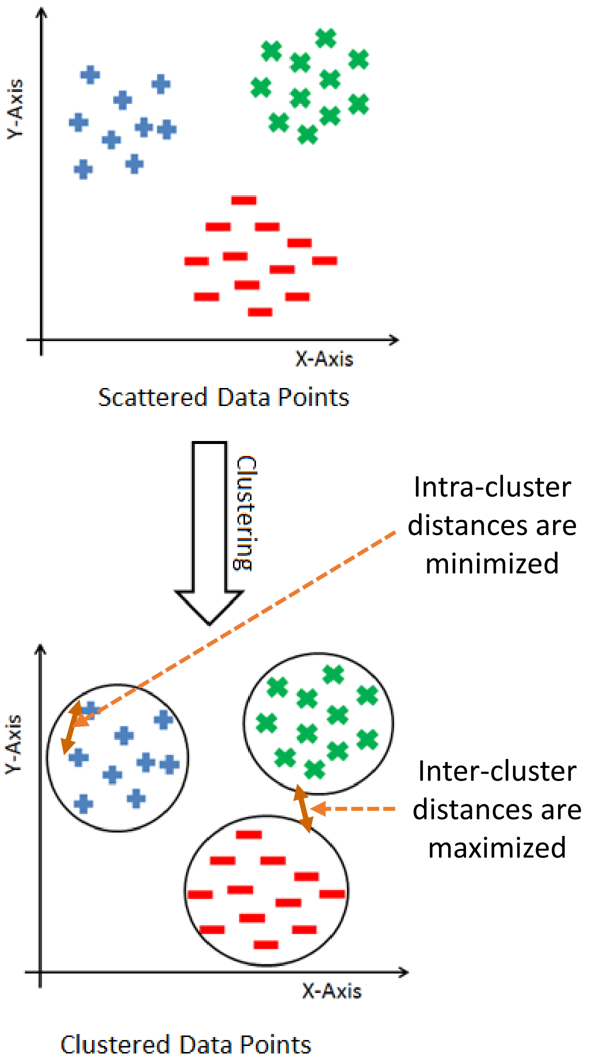
- Grouping similar products, grouping similar articles or documents, and grouping similar customers for market segmentation are all examples of clustering.

- The core principle of clustering is ***minimizing the intra-cluster distance*** and ***maximizing the inter-cluster distance***.

- The ***intra-cluster distance*** is the distance between data items within a group, and the ***inter-cluster distance*** is the distance between different groups.

- Since the data points are not labeled, so clustering is a kind of unsupervised problem.

- There are various methods for clustering and each method uses a different way to group the data points.



Clustering Criterion

- When we are combining similar data points, the question that arises is how to find the similarity between two data points, so we can group similar data objects into the same cluster.

- In order to measure the similarity or dissimilarity between data points, we can use distance measures, such as **Euclidean, Manhattan, and Minkowski** distance.

- Where the distance formula calculates the distance between 2 k-dimensional vectors, $\boldsymbol{x_i}$ and $\boldsymbol{y_i}$.

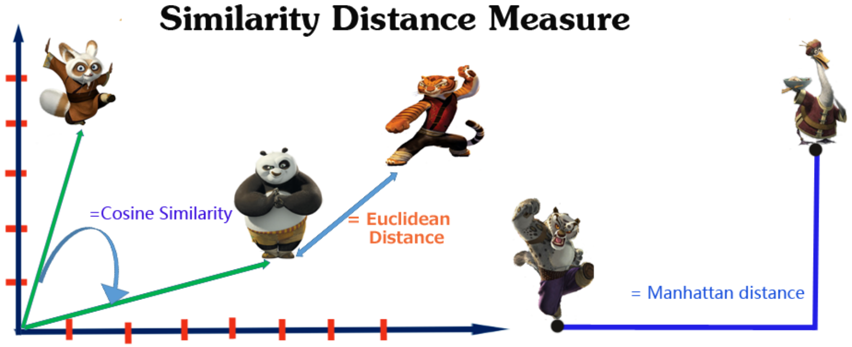
- We know what clustering is, but the most important question is, ***how many numbers of clusters should be considered when grouping the data?***

- This is the biggest challenge for most clustering algorithms.

$$Euclidean\ dist. = \sqrt{\sum_{i=1}^{k} (x_i - y_i)^2}$$

$$Manhattan\ dist. = \sum_{i=1}^{k} |x_i - y_i|$$

$$Minkowski\ dist. = \left(\sum_{i=1}^{k} (|x_i - y_i|)^q\right)^{\left(\frac{1}{q}\right)}$$



https://dataaspirant.com/five-most-popular-similarity-measures-implementation-in-python

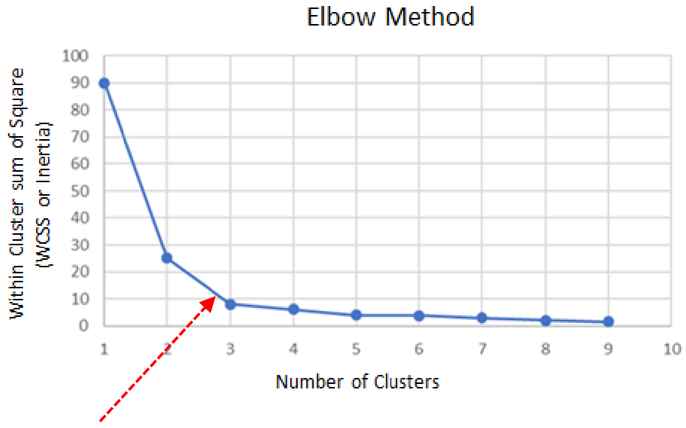
Finding Number of Clusters

- We focus on the most fundamental issue of clustering algorithms, which is discovering the number of clusters in a dataset. In **k-means clustering**, we need to define the number of clusters. Selecting the right value for the number of clusters is tricky. We can use two methods to decide the number of clusters

- ## The elbow method

- The elbow method is a well-known method for finding out the best number of clusters.

- In this method, we focus on the percentage of variance for the different numbers of clusters. The core concept of this method is to select the number of clusters that appending another cluster should not cause a huge change in the variance. **The sum of squares is also known as the Within-Cluster Sum of Squares (WCSS) or inertia.**



- We can plot a graph for the sum of squares within a cluster using the number of clusters to find the optimal value.

- As we can see, at k = 3, the graph begins to flatten significantly, so we would choose 3 as the number of clusters.

$$WCSS = \sum_{j=1}^{k} \sum_{i}^{n} distance(x_i, C_j)^2$$

- Where C_j is the cluster centroid and x_i is the data points in each cluster.

The Silhouette Method

- The **silhouette method** assesses and validates cluster data. It finds how well each data point is classified. The plot of the silhouette score helps us to visualize and interpret how well data points are tightly grouped within their own clusters and separated from others. It helps us to evaluate the number of clusters. **Its score ranges from -1 to +1.**
- A **positive value** indicates a well-separated cluster and a **negative value** indicates incorrectly assigned data points.
- The more positive the value, the further data points are from the nearest clusters; a value of zero indicates data points that are at the separation line between two clusters. Let's see the formula for the **silhouette score**

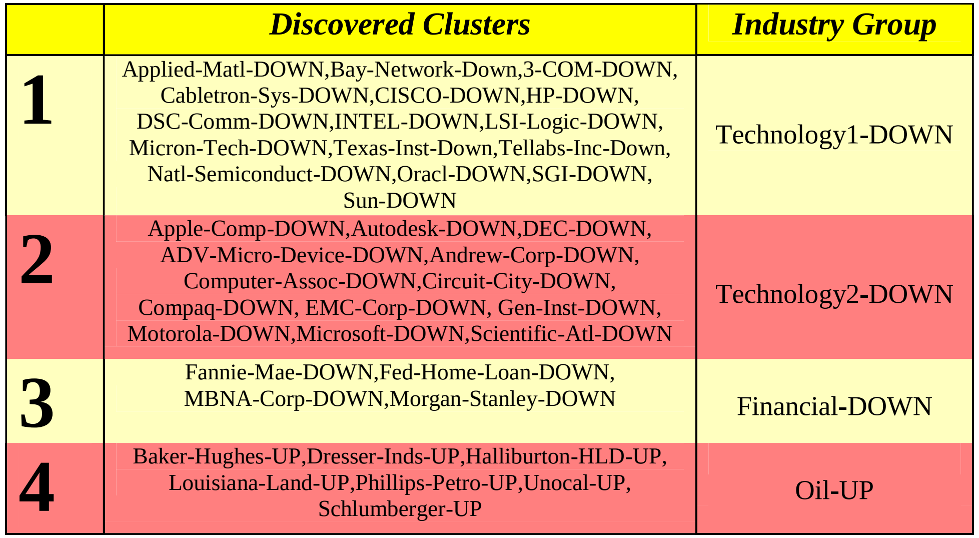
$$S(i) = \frac{b_i - a_i}{max(b_i\,, a_i\,)}$$

- $\boldsymbol{a}_\mathrm{i}$ is the average distance of the **i**$^{\mathbf{th}}$ data point from other points within the cluster.
- $\boldsymbol{b}_\mathrm{i}$ is the average distance of the **i**$^{\mathbf{th}}$ data point from other cluster points.
- This means we can easily say that **S(i)** would be between **[-1, 1].** So, for **S(i)** to be near to **1**, $\boldsymbol{a}_\mathrm{i}$ must be very small compared to $\boldsymbol{b}_\mathrm{i}$, that is, $\boldsymbol{a}_\mathrm{i}$ **<<** $\boldsymbol{b}_\mathrm{i}$.

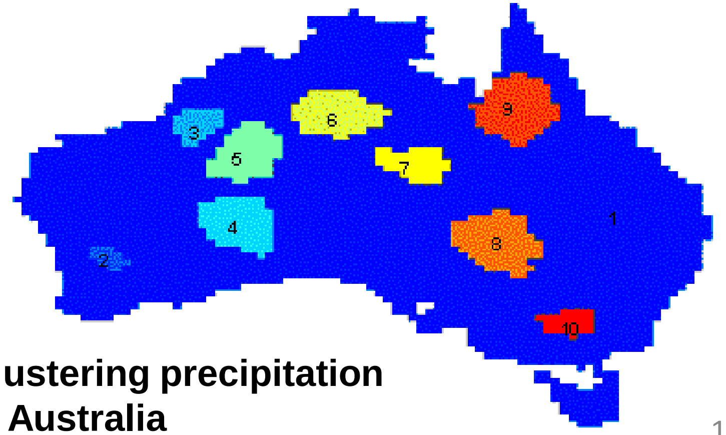
What is not Cluster Analysis?

- **Simple segmentation**
  - Dividing students into different registration groups alphabetically, by last name
- **Results of a query**
  - Groupings are a result of an external specification
  - **Clustering is a grouping of objects based on the data using some algorithm**
- **Supervised classification**
  - Have **class label** information
- **Association Analysis**
  - Local vs. global connections

Applications of Cluster Analysis



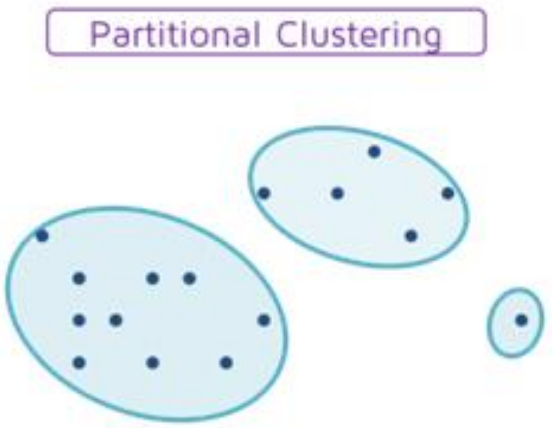
	Discovered Clusters	*Industry Group*
1	Applied-Matl-DOWN,Bay-Network-Down,3-COM-DOWN, Cabletron-Sys-DOWN,CISCO-DOWN,HP-DOWN, DSC-Comm-DOWN,INTEL-DOWN,LSI-Logic-DOWN, Micron-Tech-DOWN,Texas-Inst-Down,Tellabs-Inc-Down, Natl-Semiconduct-DOWN,Oracl-DOWN,SGI-DOWN, Sun-DOWN	Technology1-DOWN
2	Apple-Comp-DOWN,Autodesk-DOWN,DEC-DOWN, ADV-Micro-Device-DOWN,Andrew-Corp-DOWN, Computer-Assoc-DOWN,Circuit-City-DOWN, Compaq-DOWN, EMC-Corp-DOWN, Gen-Inst-DOWN, Motorola-DOWN,Microsoft-DOWN,Scientific-Atl-DOWN	Technology2-DOWN
3	Fannie-Mae-DOWN,Fed-Home-Loan-DOWN, MBNA-Corp-DOWN,Morgan-Stanley-DOWN	Financial-DOWN
4	Baker-Hughes-UP,Dresser-Inds-UP,Halliburton-HLD-UP, Louisiana-Land-UP,Phillips-Petro-UP,Unocal-UP, Schlumberger-UP	Oil-UP

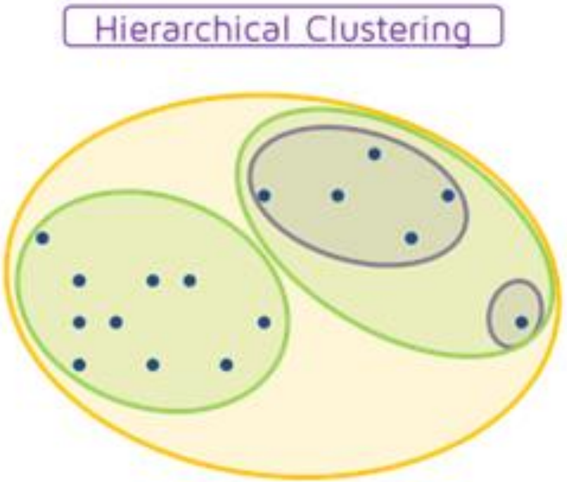


Clustering precipitation in Australia

Types of Clustering

- A **clustering** is a set of clusters.
- Important distinction between **hierarchical** and **partitional** sets of clusters.
- **Partitional Clustering**
  - A division of data objects into non-overlapping subsets (clusters), such that each data object is in exactly one subset.
- **Hierarchical Clustering**
  - A set of nested clusters organized as a hierarchical tree.

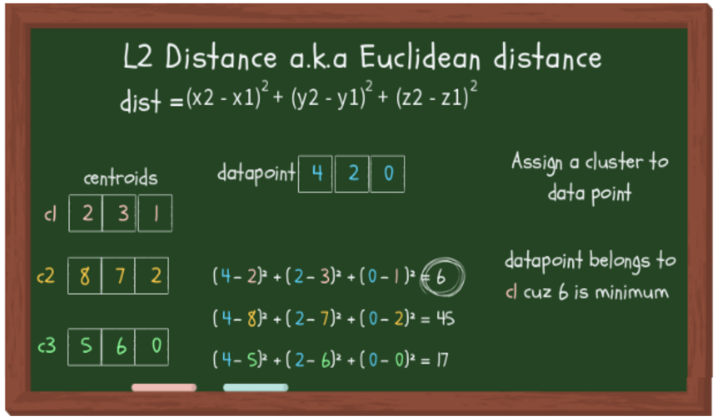


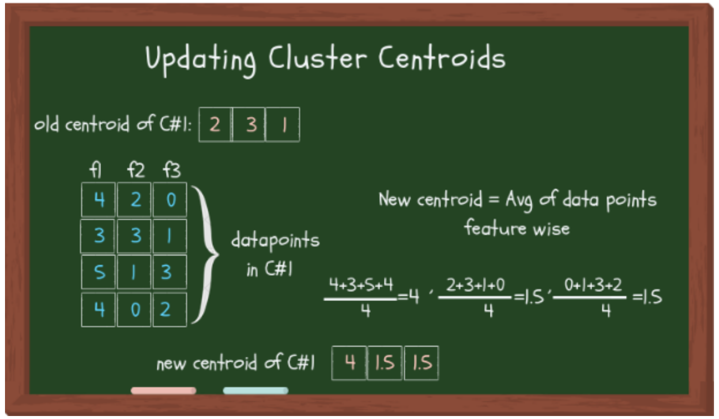


KMeans Clustering

- **Partitional clustering approach**
- Number of clusters, K, must be specified
- Each cluster is associated with a **centroid** (center point)
- Each point is assigned to the cluster with the closest centroid
- The 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
```



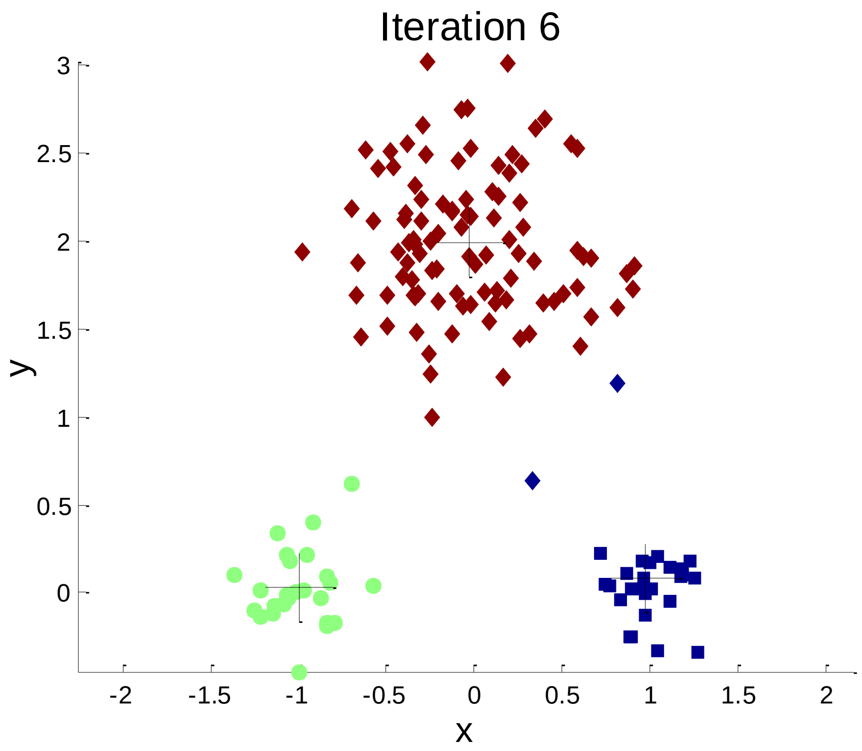


Updating Cluster Centroids
old centroid of C#1: 2 3 1
f1 f2 f3
4 2 0
3 3 1
5 1 3
4 0 2
datapoints in C#1
New centroid = Avg of data points feature wise
(4+3+5+4)/4 =4 , (2+3+1+0)/4 =1.5 , (0+1+3+2)/4 =1.5
new centroid of C#1 4 1.5 1.5

K-means Clustering

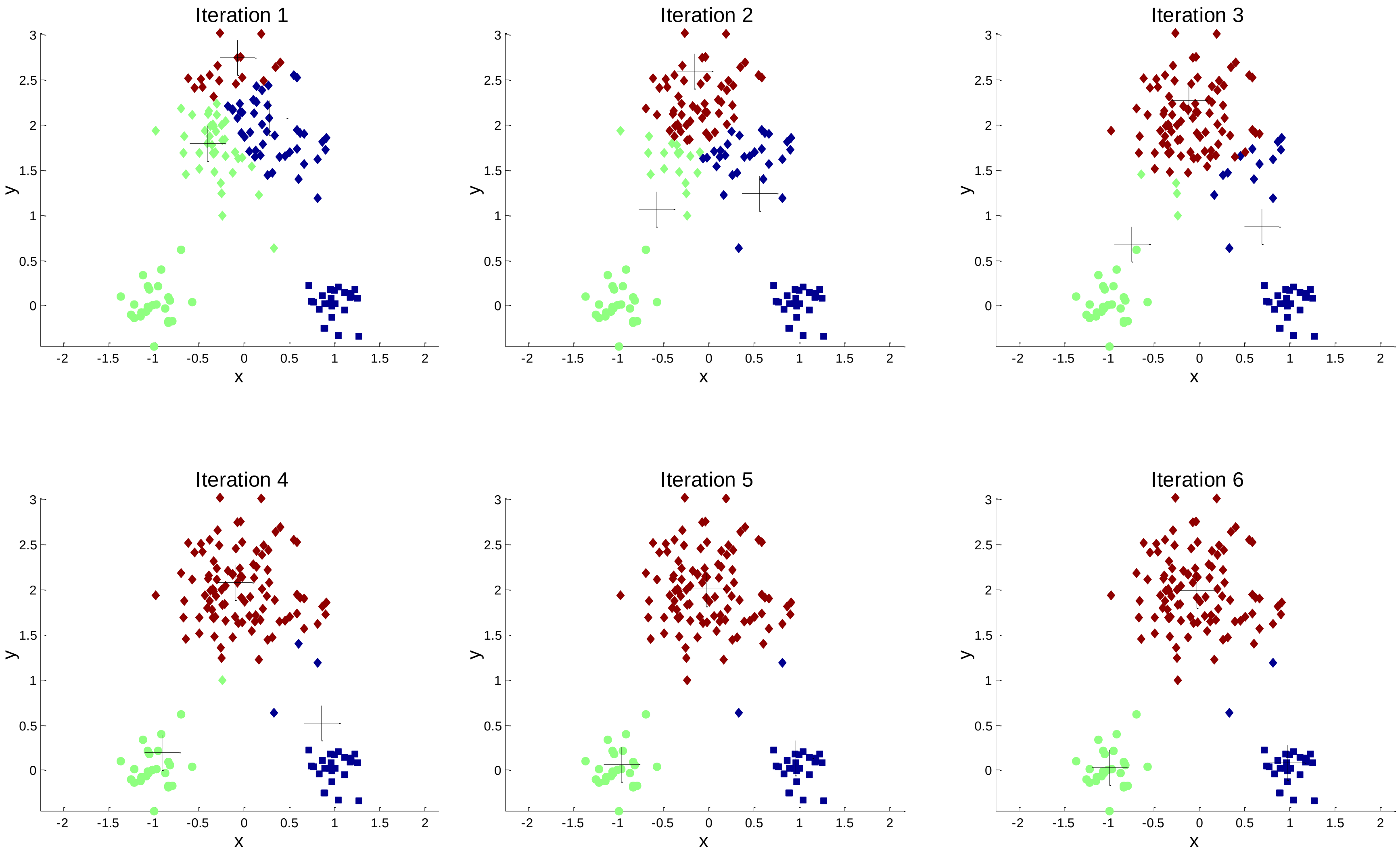
Example

- Initial centroids are chosen randomly.
  - Clusters produced vary from one run to another.
- The centroid is (typically) the mean of the points in the cluster.
- ‘**Closeness**’ is measured by **Euclidean distance**, **cosine similarity, correlation, etc.**
- **K-means** will converge for common similarity measures as mentioned previously.
- Most of the convergence happens in the first few iterations.
  - Often the stopping condition is changed to ‘Until relatively few points change clusters’
- **Complexity is O(n * K * I * d)**
  - n = number of points, K = number of clusters,
    I = number of iterations, d = number of attributes



K-means Clustering

Example



Evaluating K-means Clusters

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

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

  - ***x*** is a data point in cluster ***C*$_{\mathbf{i}}$** and ***m$_i$*** is the representative point for cluster ***C*$_{\mathbf{i}}$**
    - can show that ***m$_i$*** corresponds to the center (mean) of the cluster
  - Given two sets of clusters, we prefer the one with the smallest error
  - One easy way to reduce **SSE** is to increase **K**, the number of clusters
    - A good clustering with smaller **K** can have a lower **SSE** than a poor clustering with higher **K**

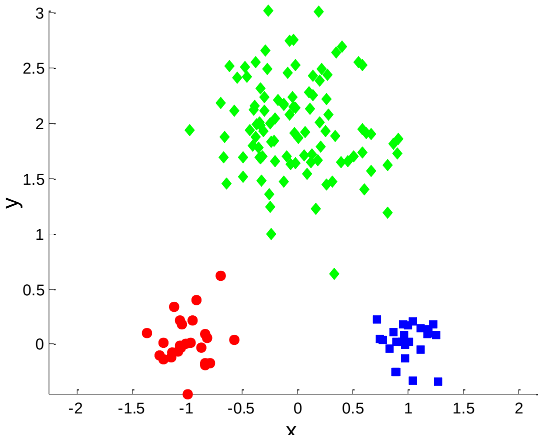
Two different K-means Clustering

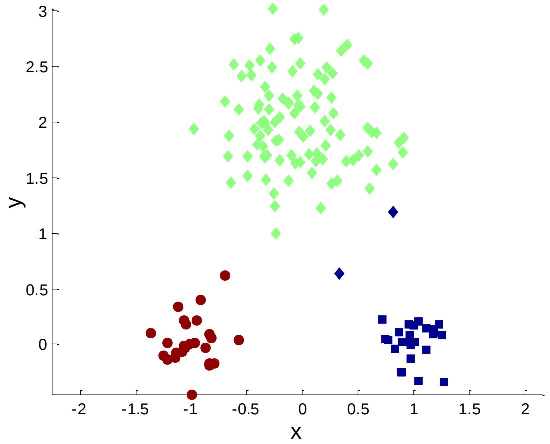
Limitations of K-means

- **K-means** has problems when clusters are of differing
  - Sizes
  - Densities
  - Non-globular shapes

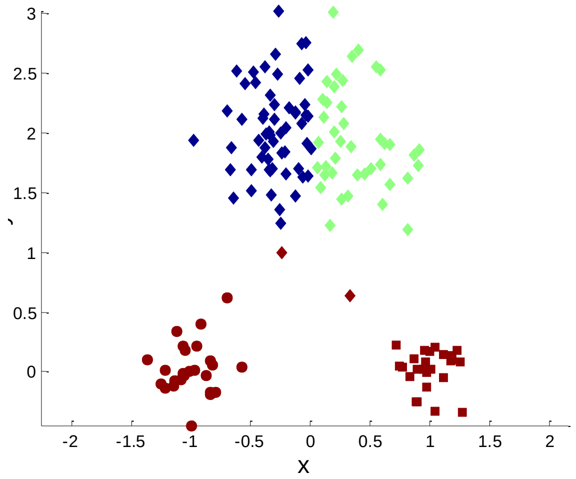
- **K-means** has problems when the data contains outliers.

Original Points





Optimal Clustering



Sub-optimal Clustering

Solutions to
Initial Centroids Problem

- Multiple runs
  - Helps, but probability is not on your side
- Sample and use hierarchical clustering to determine initial centroids
- Select more than **K** initial centroids and then select among these initial centroids
  - Select most widely separated
- Post-processing
- Generate a larger number of clusters and then perform a hierarchical clustering
- Bisecting **K-means**
  - Not as susceptible to initialization issues

K-means++

- This approach can be slower than random initialization, but very consistently produces better results in terms of **SSE**
  - The **k-means++** algorithm guarantees an approximation ratio **O(log k)** in expectation, where **k** is the number of centers
- To select a set of initial centroids, ***C***, perform the following

1. Select an initial point at random to be the first centroid
2. For **k – 1** steps
3. For each of the **N** points, $\mathbf{x}_i$, $1 \le i \le N$, find the minimum squared distance to the currently selected centroids, $C_1, \ldots, C_j,\ 1 \le j < k$, i.e., $\min_j d^2(C_j, x_i)$
4. Randomly select a new centroid by choosing a point with probability proportional to

$$\frac{\min_j d^2(C_j, x_i)}{\sum_i \min_j d^2(C_j, x_i)}$$ is

5. End For

Updating Centers Incrementally

- In the basic **K-means algorithm**, centroids are updated after all points are assigned to a centroid.
- An alternative is to update the centroids after each assignment (incremental approach)
  - Each assignment updates zero or two centroids
  - More expensive
  - Introduces an order dependency
  - Never get an empty cluster
  - Can use “weights” to change the impact

Feature Extraction

Application

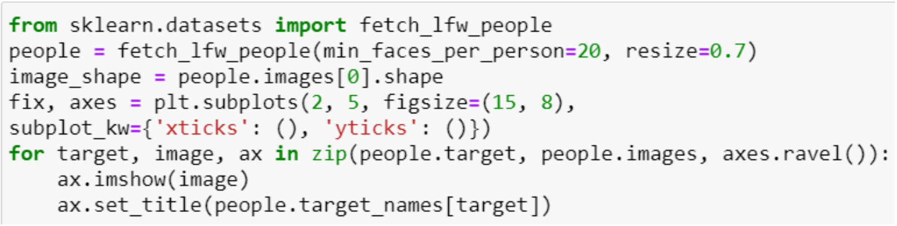
- A great example of an application where **feature extraction** is helpful with images. Images are made up of pixels, usually stored as red, green, and blue (RGB) intensities.

- Objects in images are made up of thousands of pixels, and only together are they meaningful.

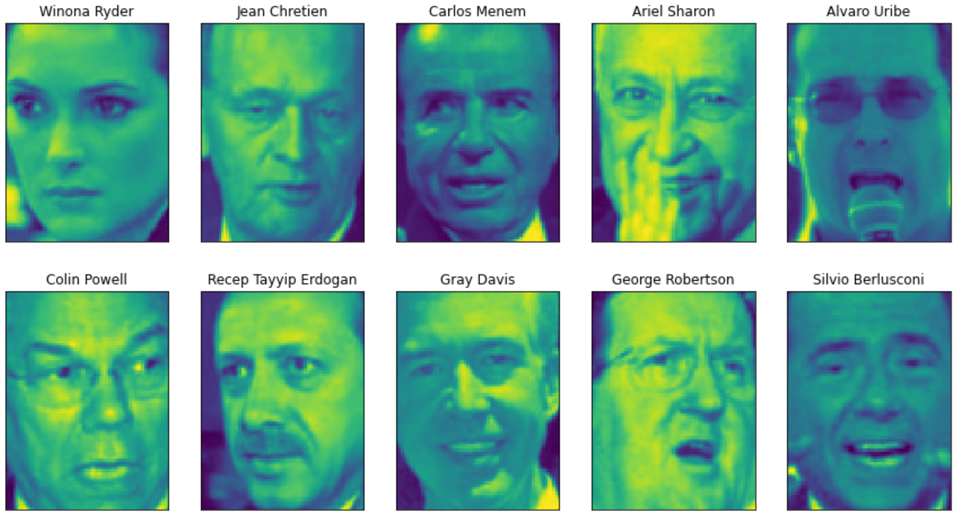
- We give a very simple application of feature extraction on images using **PCA**, by working with face images from the Labeled Faces in the Wild dataset.

- This dataset contains face images of celebrities downloaded from the Internet, and it includes faces of politicians, singers, actors, and athletes from the early 2000s.

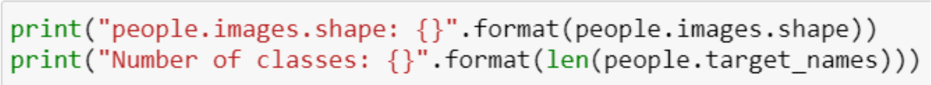
- We use grayscale versions of these images, and scale them down for faster processing. We can see some of the images in Figure.



```python
from sklearn.datasets import fetch_lfw_people
people = fetch_lfw_people(min_faces_per_person=20, resize=0.7)
image_shape = people.images[0].shape
fix, axes = plt.subplots(2, 5, figsize=(15, 8),
subplot_kw={'xticks': (), 'yticks': ()})
for target, image, ax in zip(people.target, people.images, axes.ravel()):
    ax.imshow(image)
    ax.set_title(people.target_names[target])
```



There are 3,023 images, each 87×65 pixels large, belonging to 62 different people



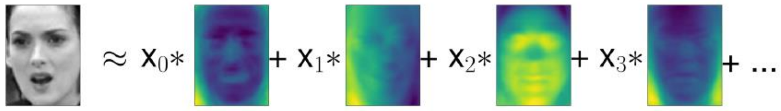
```python
print("people.images.shape: {}".format(people.images.shape))
print("Number of classes: {}".format(len(people.target_names)))
```

https://scikit-learn.org/stable/modules/generated/sklearn.datasets.fetch_lfw_people.html

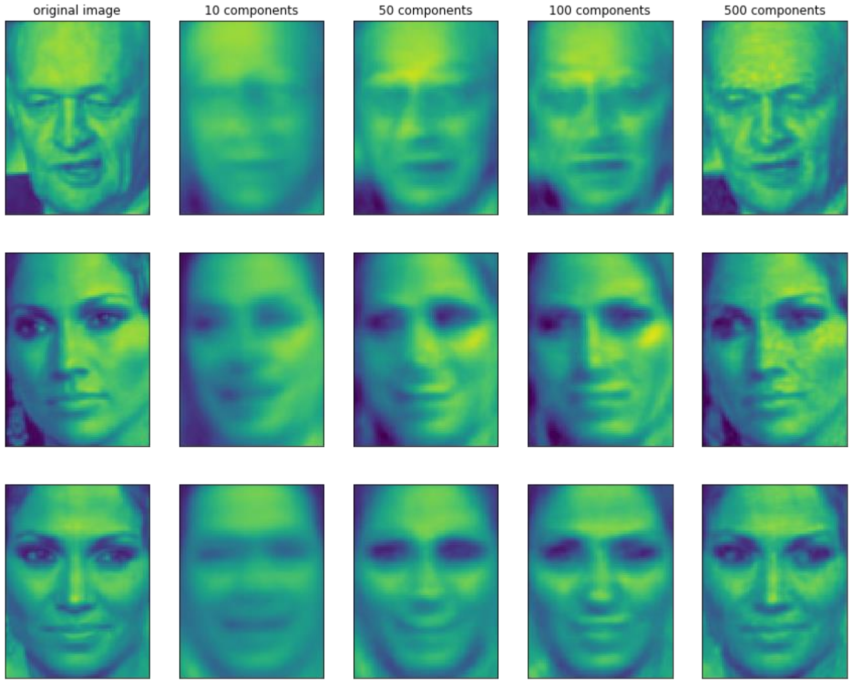
Feature Extraction

Application

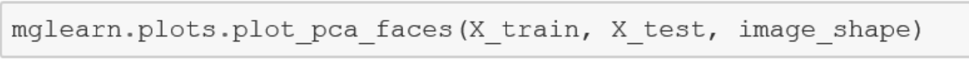
- We can see that when we use only the first 10 principal components, only the essence of the picture, like the face orientation and lighting, is captured.
- By using more and more principal components, more and more details in the image are preserved.
- This corresponds to extending the sum in Figure to include more and more terms.
- Using as many components as there are pixels would mean that we would not discard any information after the rotation, and we would reconstruct the image perfectly.



Schematic view of PCA as decomposing an image into a weighted sum of components



Reconstructing three face images using increasing numbers of principal components



```
mglearn.plots.plot_pca_faces(X_train, X_test, image_shape)
```

Comparison of PCA and K-means

- Although **k-means** is a clustering algorithm, there are interesting parallels between **k-means** and the decomposition methods like **PCA**.

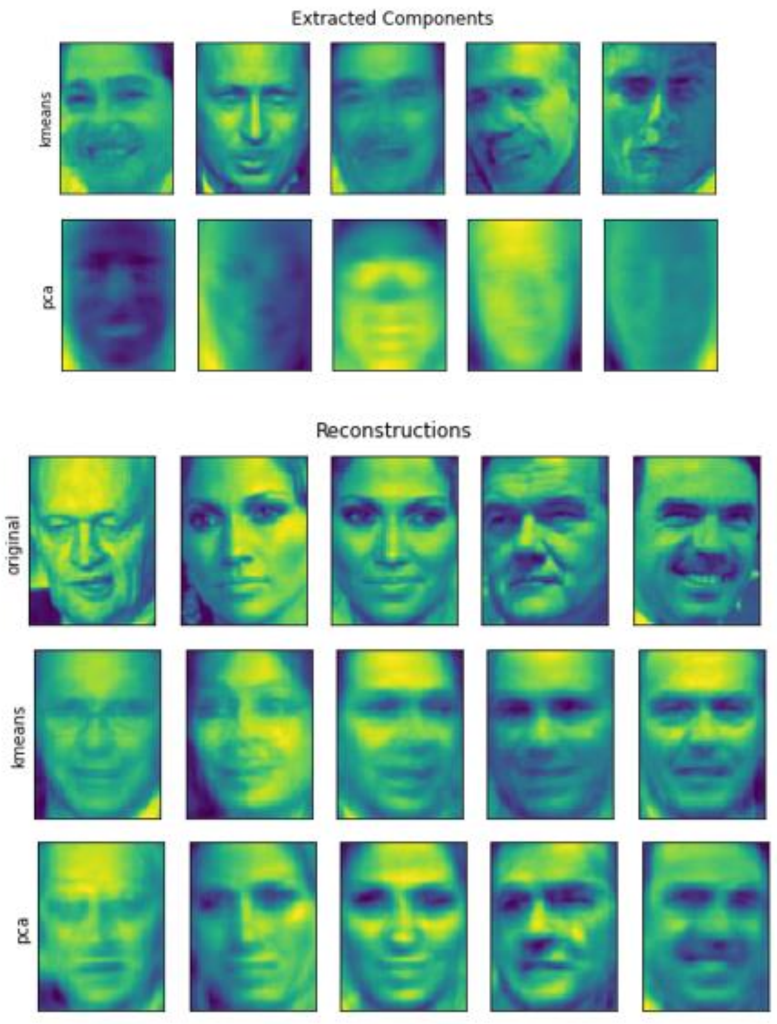
- **PCA** tries to find directions of **maximum variance** in the data, while **k-means** tries to represent each data point using a cluster center.

- We can think of that as each point being represented using a single component, which is given by the cluster center.

- This view of **k-means** as a decomposition method, where each point is represented using a single component, is called **vector quantization**.

- Lets compare **PCA** and **k-means** outcomes, showing the components extracted, as well as reconstructions of faces from the test set using 100 components.

- An interesting aspect of vector quantization using **k-means** is that we can use many more clusters than input dimensions to encode our data.



Comparing image reconstructions using k-means and PCA with 100 components (or cluster centers)—k-means uses only a single cluster center per image

Resources/ References

- Introduction to Data Mining, 2nd Edition, Pang-Ning Tan, Michael Steinbach, Anuj Karpatne, Vipin Kumar, 2019, Pearson.
- Introduction to Machine Learning with Python A Guide for Data Scientists, Andreas C. Müller and Sarah Guido, Copyright © 2017, O'Reilly.
- Machine Learning with Python Cookbook: Practical Solutions from Preprocessing to Deep Learning Paperback – 23 Mar. 2018. by. Chris Albon
- https://pub.towardsai.net/principal-component-analysis-pca-with-python-examples-tutorial-67a917bae9aa#2000
- Numerical Computing with Python, Pratap Dangeti, Allen Yu, Claire Chung, Aldrin Yim, Packt Publishing, 2018.
- https://towardsdatascience.com/k-means-clustering-from-a-to-z-f6242a314e9a