KMeans Clustering & Principal Component Analysis (PCA)

Department of Computing and Mathematical Sciences
University of Greenwich, London

Mar 19 & 26, 2021

Outline

1. KMeans Clustering
   - Clustering – Workflow
   - KMeans Clustering – Algorithm
   - KMeans Clustering - Demonstration
   - KMeans Clustering – Some Insights
   - KMeans Clustering - Image Segmentation
   - KMeans Clustering – Conclusion
2. Principal Component Analysis
   - PCA - Problem Statement and Goal
   - PCA – Algorithm
   - PCA - Dimensionality Reduction Technique
   - PCA – Some Insights
   - PCA – Conclusion
3. References

KMeans Clustering

- Unsupervised ML algorithm – could be the first step to understand data (unlabeled) - **Grouping**.
- Wide range of applications -
    - Customer Segmentation [e.g., for targeted advertising]
    - Grouping Search Results [e.g., topic-wise]
    - Image Segmentation [e.g., identify different parts of the image]
    - Many more...

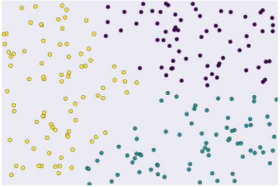


Figure: 3 Clusters/Groups

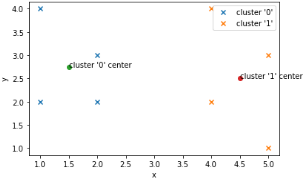


Figure: 2 Clusters/Groups with Centroids

Clustering - Workflow

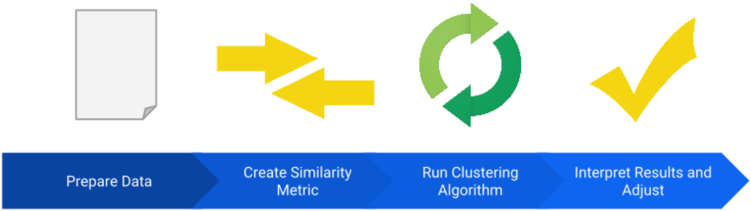


Figure: Clustering Workflow – image taken from here

Clustering – Workflow

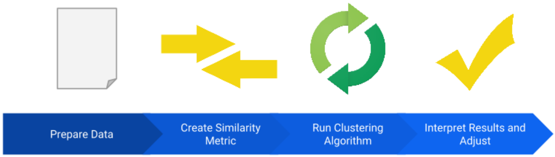


Figure: Clustering Workflow – image taken from here

- Prepare Data – follow similar process as any other ML workflow. Scale, Transform, fixing missing values, etc. – ensure the next step can be applied between the features.
- Create Similarity Metric – Pearson's coefficient, Distance measures (e.g., Euclidean), Cosine Similarity, etc. – many we have seen already.
- Run Clustering Algorithm – follow the algorithm steps of the selected one. We will show KMeans Clustering here.
- Analyse the results (remember, unlabelled data), and adjust. Cluster cardinality, magnitude, similar/dissimilar identification.

KMeans Clustering – Algorithm

Assumption: prepared dataset.

- Step 0: Select the number of clusters [K of KMeans].
- Step 1: Choose K randomly within the dataset's feature space [basic version]. These are the K initial 'means' of the desired clusters.
- Step 2: Calculate distance (e.g., Euclidean) from each data to each cluster's means.
- Step 3: Assign each data to the 'closest' cluster.
- Step 4: Compute new 'means' of the K clusters after these assignments.
- Repeat Step 2, 3, 4 until
  - No change in clusters, i.e., between two iterations, the 'means' of each cluster do not change.
  - Maximum number of iterations reached [you can specify it if required]

KMeans Clustering - Demonstration

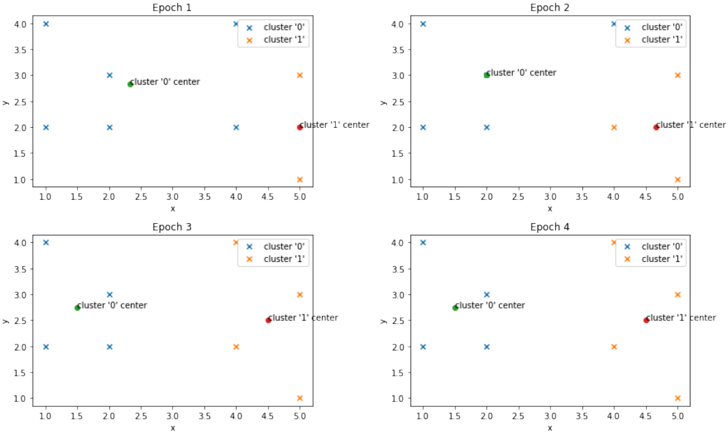


Figure: Clustering Workflow – image taken from the notebook here

At this point, we will demonstrate this notebook to understand the KMeans clustering steps - here.

KMeans Clustering – Some Insights

- Advantages
  - Easy to understand and implement
  - Scale. Ensure Convergence.
  - Can be made adaptable to different shape and size clusters
- Disadvantages
  - Selecting K. One approach *Elbow* is shown in our notebook.
  - Initial choice of K means can result in different clusters [even not optimised]
  - Different shape and sizes clusters [even though generalisation can be possible], outliers, large dimensions of dataset all have impact on the clustering.

KMeans Clustering - Image Segmentation

Figure: Original Image – collected from here

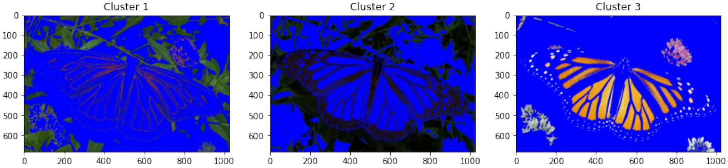


Figure: Image Segmentation (3 clusters) – from the notebook here

At this point, we will demonstrate this *image segmentation* notebook to show its application - here.

KMeans Clustering – Conclusion

We discussed KMeans clustering by identifying its strength and weakness, and demonstrated some use-cases. Different Clustering approaches (images taken from here):

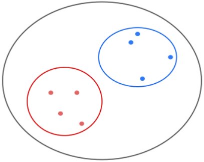


Figure: Hierarchical

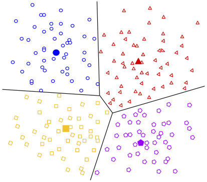


Figure: Centroid-based [KMeans]

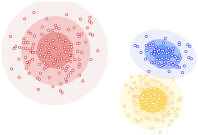


Figure: Distribution-based

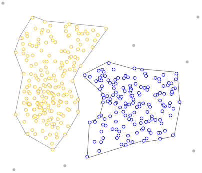


Figure: Density-based

Principal Component Analysis (PCA)

- Previously, we showed how a complex real-valued data point can be grouped (i.e., clustering).
- There can be many situations where you might be working with data in a large dimensional space (i.e., image), or they can be noisy. PCA transforms such data in a lower dimensional space *retaining* most information.
- Formal definition: given dataset in d dimensions – convert it to k dimensions ($k < d$) with minimal loss of information.

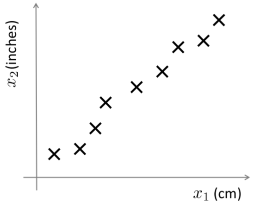


Figure: 2D Data

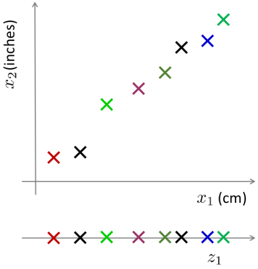


Figure: 1D Data Transformation

Principal Component Analysis (PCA)

- PCA is a linear transformation which transforms the data into a new coordinate system
- Greatest variance by any projection of the data lie on the first axis (1^{st} principal component), second greatest variance on the second axis(2^{nd} principal component), and so on.
- PCA is sometimes referred to as a dimensional reduction technique which is achieved by eliminating the later principal components.

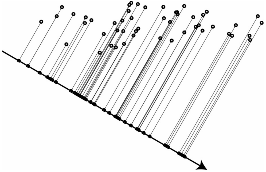


Figure: Find projection that maximises the variance

PCA - Variance and Covariance

- Both measure the *spread* of a set of points around their centre of mass (mean). In 1D, deviation from the mean for all data is variance. Cavariance measures variations from the mean in each dimension with respect to each other.

- If there are n data point sample in 2D (x, y) - this can be either figures on the right.
  - variance, $\sigma_x^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu_x)^2$, &

    $$\sigma_y^2 = \frac{1}{n}\sum_{i=1}^{n}(y_i - \mu_y)^2.$$
  - covariance,

    $$cov(x, y) = \sigma_{xy} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu_x)(y_i - \mu_y)$$

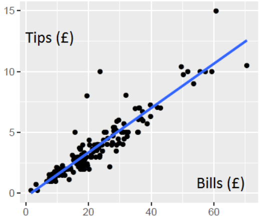


Figure: Positive Covariance

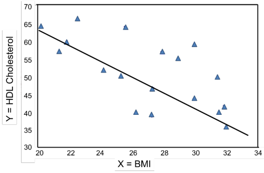


Figure: Negative Covariance

PCA - Problem Statement and Goal

- **Problem Statement:** How to find the principal components so that the feature, $x_1, x_2, \ldots, x_n$ is reduced from n-dimension to k-dimension, i.e., find $z_1, z_2, \ldots, z_k$ vectors to project the data, so as to minimize the projection error. Subsequently, a transformation matrix, $W \in \mathbb{R}^{d \times k}$ of the form $xW = z$, is needed for the conversion.

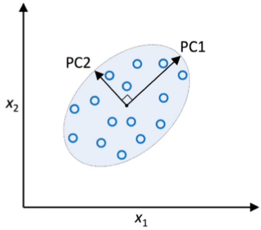
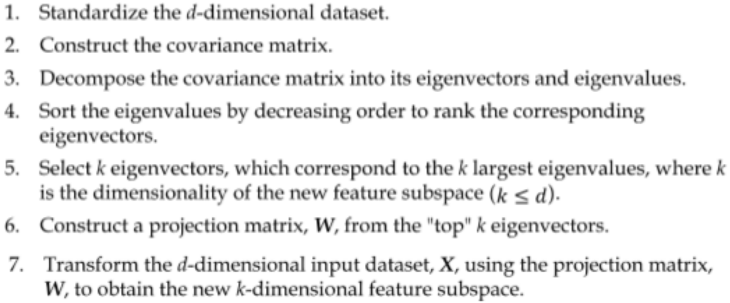


Figure: x_1 and x_2 original feature axes, PC_1 and PC_2 are the principal components

PCA – Algorithm



1. Standardize the d-dimensional dataset.
2. Construct the covariance matrix.
3. Decompose the covariance matrix into its eigenvectors and eigenvalues.
4. Sort the eigenvalues by decreasing order to rank the corresponding eigenvectors.
5. Select k eigenvectors, which correspond to the k largest eigenvalues, where k is the dimensionality of the new feature subspace ($k \leq d$).
6. Construct a projection matrix, $\mathbf{W}$, from the "top" k eigenvectors.
7. Transform the d-dimensional input dataset, $\mathbf{X}$, using the projection matrix, $\mathbf{W}$, to obtain the new k-dimensional feature subspace.

Figure: PCA Algorithm Steps [RM19]

Eigenvectors and eigenvalues: all vectors (e.g., x) change direction when multiplied by a matrix, say Σ (covariance matrix). Eigenvectors (v) are special vectors which do not, $\Sigma v = \lambda v$. λ is the eigenvalue = stretching or compressing and/or flipping factor.

PCA - Dimensionality Reduction Technique

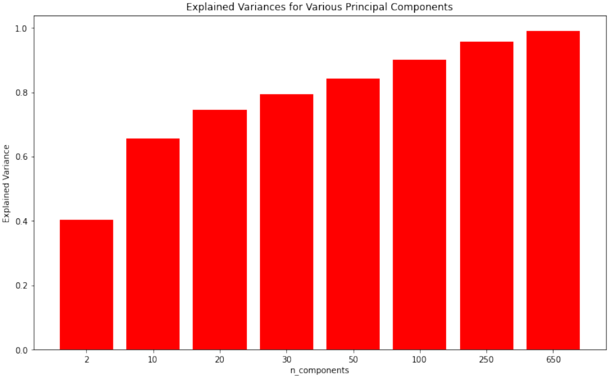


Figure: PCA on CIFAR dataset – from the notebook here. Original dataset dimension: $32 \times 32 \times 3 = 3072$. Almost 96% variance could be explained by the 250 principal component axes.

At this point, we will demonstrate *PCA as dimensionality reduction technique* in this notebook here.

PCA – Some Insights

- Advantages
  - Sometimes useful for data visualisation after converting it to lower dimension (e.g., 2 or 3).
  - Widely used as dimension reduction technique, e.g., image, signal processing, etc.
  - Can also be helpful if noisy data on original feature space (de-noise).
- Disadvantages
  - Trade-off between information loss and dimensionality reduction.
  - It is not straightforward to determine which features are more important than others since feature space is transformed along principal components.

PCA – Conclusion

We discussed PCA by identifying its strength and weakness, and demonstrated some use-cases. We will also discuss if PCA could be incorporated inside the coursework - notebook here):

References

Sebastian Raschka and Vahid Mirjalili.
Python Machine Learning, 3rd Ed.
Packt Publishing, Birmingham, UK, 2019.