Machine Learning & its Application

Unsupervised Learning

06/10/2023

Done by: Issa Hammoud

Outline

- Introduction
- Principal Component Analysis
- Clustering
  - KMeans
  - DBScan
- Deep Representation
- Exercise

Introduction

- We presented supervised algorithms where we have data to learn from.
- Unsupervised problems aim to find similarities in an existing data.
- Some examples of unsupervised problems are clustering and recommendation.
- In general, we compress the data of interest to make the task easier.
- An example of a dimensionality reduction algorithm is PCA.

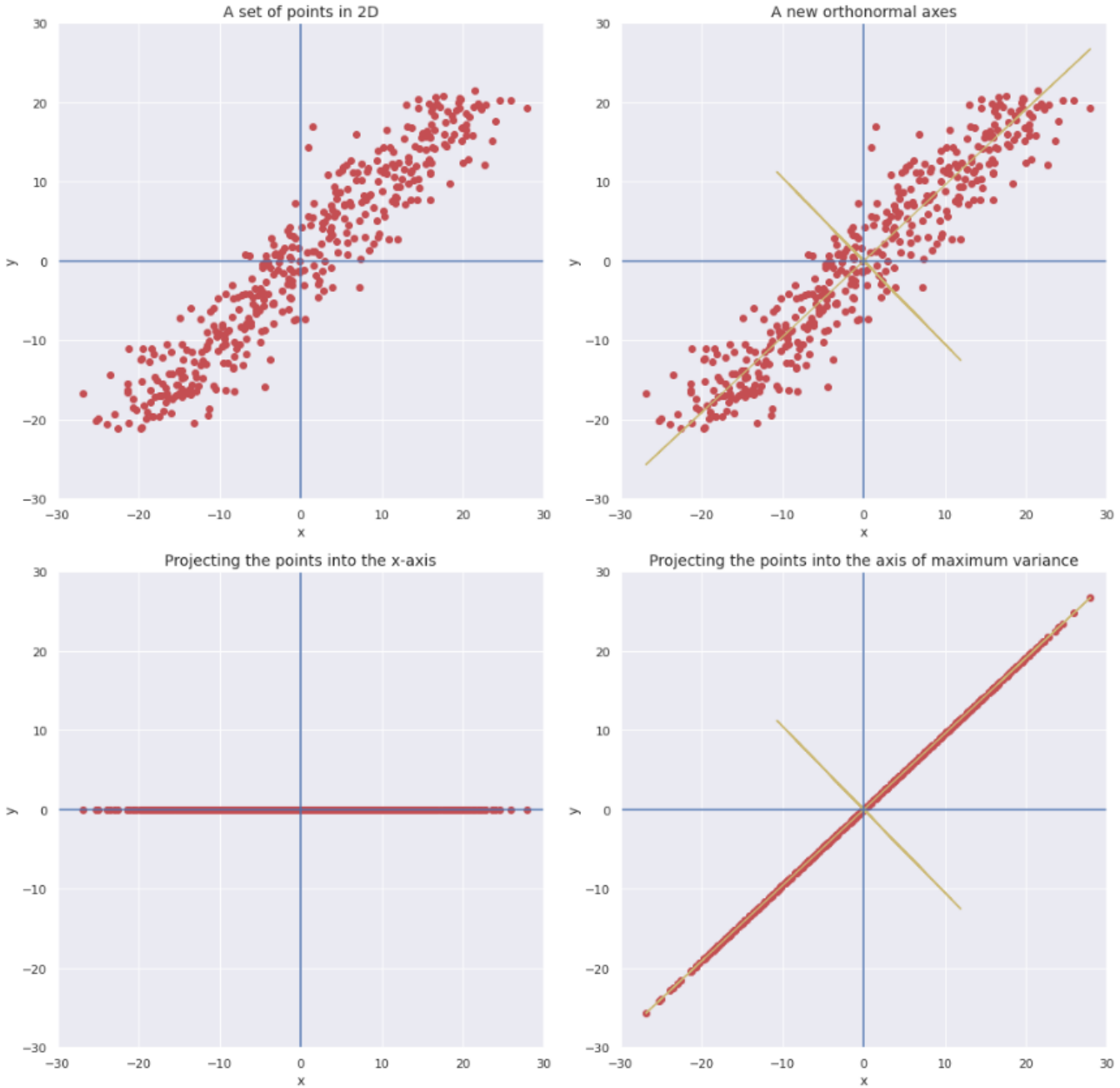
Outline

- Introduction
- Principal Component Analysis
- Clustering
  - KMeans
  - DBScan
- Deep Representation
- Exercise

PCA

- PCA is a compression algorithm that aims to reduce the input dimensionality.
- It does so by finding the direction of maximum variance.
- It will find a basis where the data variance is maximized along its axis.
- PCA is a linear model. It supposes there is a linear relationship in the data.

PCA

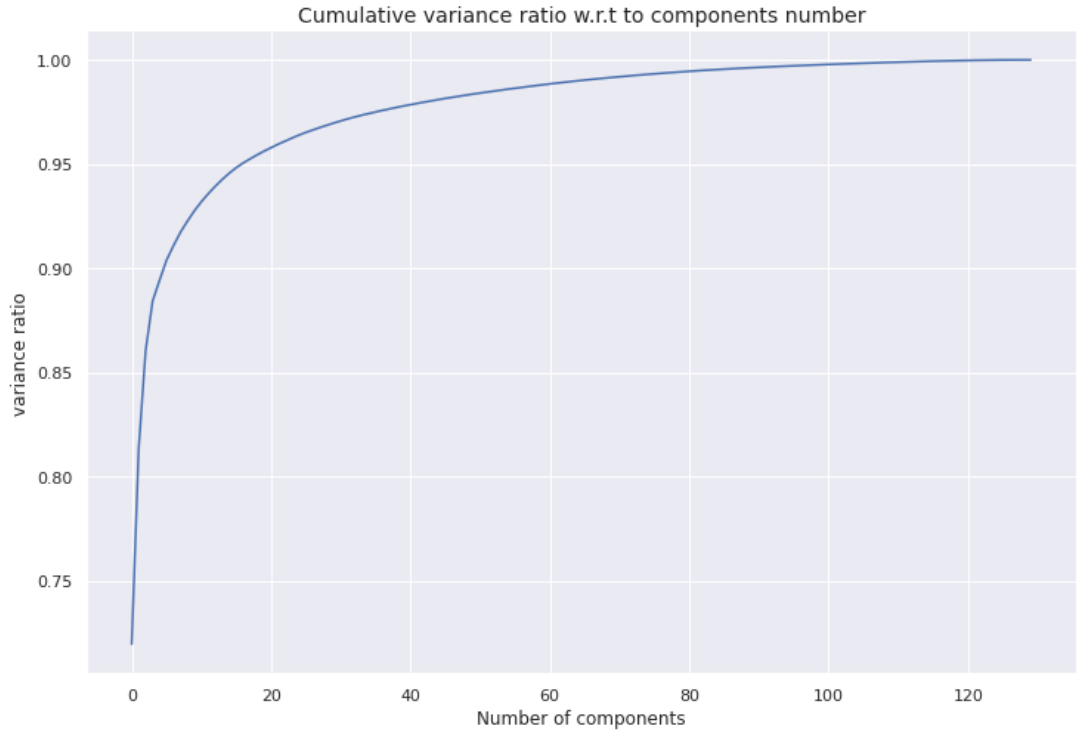

A set of points in 2D
A new orthonormal axes
Projecting the points into the x-axis
Projecting the points into the axis of maximum variance
x
y

PCA

- We can prove that:
  - The eigen vectors of X^TX are the direction of maximum variance.
  - The eigen values of X^TX represent the importance of each eigen vector.
- PCA will order the principal components in decreasing order of their contribution.
- To compress an input, we can choose a number of components that explains enough the data.
- Note that the data should be centered before applying PCA.

PCA

- We can plot the explained variance as a function of the number of components.
- For instance, if the following example, 20 components explains 96% of the variance.



PCA

- PCA is implemented in sklearn as well.

```
def plot_cumulative_variance(frames):
  """
  frames: 3D np.array, type float, shape (number_of_samples, frame_height, frame_width)
          array of gray scale images
  """
  pca = sklearn.decomposition.PCA()
  # vectorize each image. The input of PCA should always be (number_of_samples, number_of_features)
  data = frames.reshape((frames.shape[0], -1))
  pca.fit(data)
  # compute the cumulative sum of each components contribution
  cumsum = np.cumsum(pca.explained_variance_ratio_)
  plt.figure(figsize=(12,8))
  plt.title("Cumulative variance ratio w.r.t to components number", fontsize=14)
  plt.xlabel("Number of components")
  plt.ylabel("variance ratio")
  plt.plot(np.arange(data.shape[0]), cumsum)
  plt.show()
```

PCA

- To apply PCA into a set of images, we should apply the following steps:
    1. Flatten all the images.
    2. Stack them in a (number_of_samples, number_of_features) array.
    3. Substract the mean over the batch dimension.
    4. Apply PCA (mainly compute the eigen vectors and value, or equivalently the singular one).
- Thus, PCA should be applied on all images in order to reduce their dimensionality.
- This makes it very resources intensive when applied to a lot of data.
- However, some iterative technique exist to overcome this.

PCA

- PCA was widely used before, but it becomes less and less popular after deep learning.
- One of the first face recognition algorithm used PCA, known as eigen face.
- The idea was to apply PCA on a set of faces.
- Then, for a new face, find the most similar person to it.
- PCA goal was to reduce the images dimensionality and make the algorithm robust to small changes.
- However, the performance is uncomparable with what we have today.

Outline

- Introduction
- Principal Component Analysis
- Clustering
  - KMeans
  - DBScan
- Deep Representation
- Exercise

Clustering

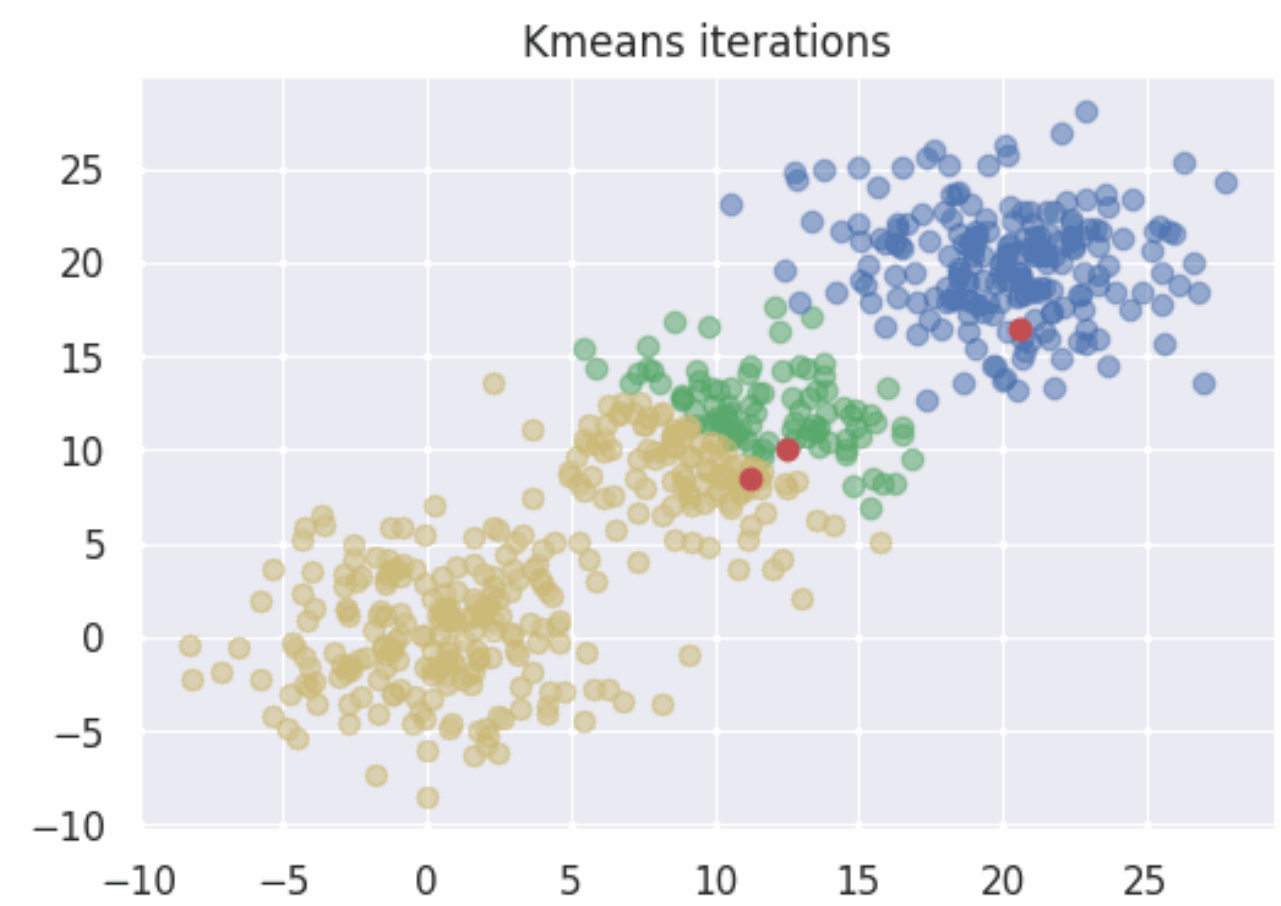
- Clustering is the task of grouping data into homogeneous groups.
- It is very important in practice and can be applied in multiple applications.
- However, it is a hard problem to solve.
- Many clustering algorithms exist. Some needs to define the number of clusters, some don't.
- We will present an example of each algorithm.

KMeans

- KMeans is an unsupervised clustering algorithm that separates the data into k clusters.
- It works as follows:
  - Generate k random values to be the initial centroid of your clusters.
  - Assign each point into the nearest cluster (nearest centroid).
  - Update the centroids, to be the average of each cluster.
  - Repeat until the centroids don't move anymore.
- A more intelligent approach is to use k random points instead of random values for initialization, known as Kmeans++.

KMeans

- Kmeans has some drawbacks:
  - We should know beforehand K.
  - It may not converge based on the initial points.
  - It can give different result at each run.



KMeans

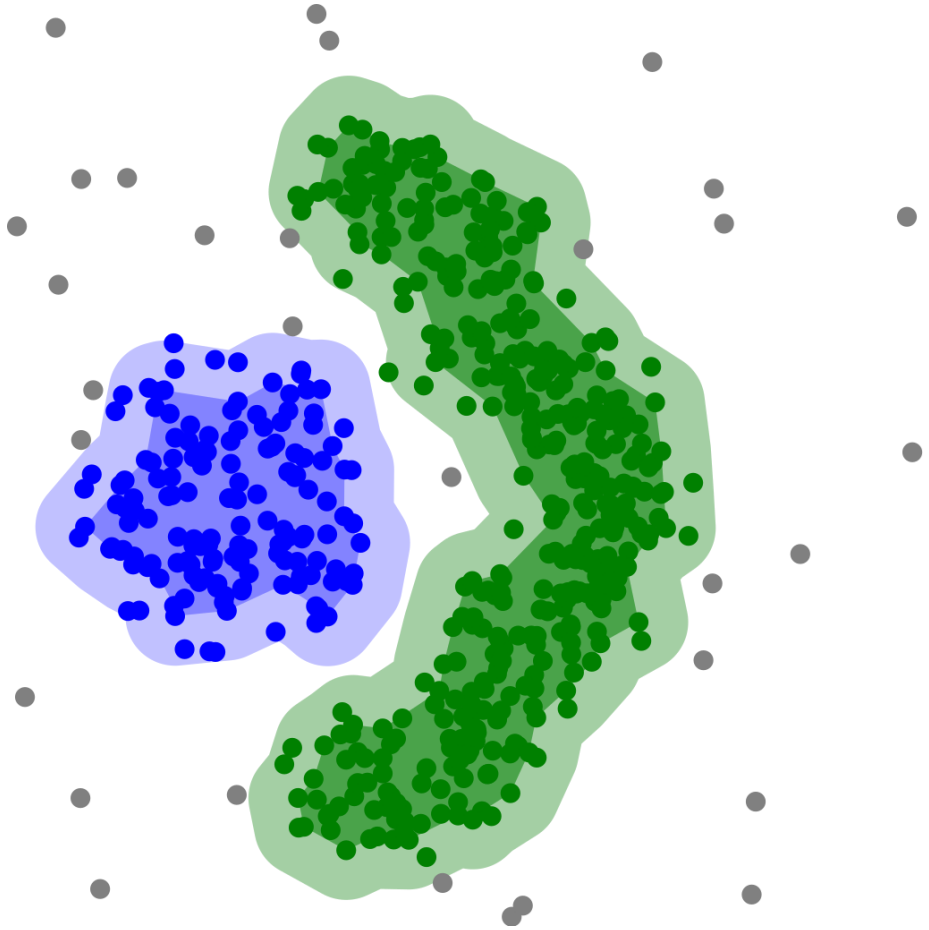
```
def kmeans(points, clusters_n, iteration_n)

  centroids = points[np.random.choice(np.arange(points.shape[0]), clusters_n, replace=False)]

  points_expanded = np.expand_dims(points, axis=0)
  centroids_expanded = np.expand_dims(centroids, axis=1)

  for step in range(iteration_n):

    means = []
    distances = np.sum((points_expanded - centroids_expanded)**2, 2)
    assignments = np.argmin(distances, 0)

    for c in range(clusters_n):
        indices = np.reshape(np.where(assignments == c), [1,-1])
        means.append(np.mean(points[indices], axis=1))

    centroids_expanded = np.expand_dims(np.concatenate(means, 0), axis=1)

  distances = np.sum((points_expanded - centroids_expanded)**2, 2)
  assignments = np.argmin(distances, 0)
  return assignments
```

DBScan

- DBScan is an algorithm that clusters data without requirement a predefined number of clusters.
- It will also tell you if there are outliers in your data.
- The idea is to cluster points is the same neighborhood as part of the same cluster.
- We need to define what is the maximal distance that can separate neighbors.
- In addition, we need to define the minimum number of points in a neighborhood.
- Otherwise, they will be clustered as noise.

DBScan

- We can use the Euclidian distance or any other distances.
- This kind of points cannot be adequetely clustered using kmeans.
- DBScan refers to Density-Based Spatial clustering of applications with noise.
- DBScan may fail if the if the data points have large difference in density.



Deep Representation

- In order to cluster a set of data points, we have 2 steps:
  - Reduce the dimensionality of the data to make clustering easier.
  - Apply a clustering algorithm to the compressed data.
- We may choose different clustering algorithm based on our knowledge of the data.
- If we want something robust to noise, DBScan is our choice.
- If we know the number of clusters, Kmeans if the way to go.
- We also explored PCA as a dimensionality reduction algorithm.

Deep Representation

- PCA can be useful if there is a linear relationship between the data points.
- However, this is not always the case.
- There are some variations, like Kernel PCA, to treat non linear data.
- But as we saw in previous lessons, we prefer to use Deep Learning in such cases.
- But how to use deep learning models in such unsupervised cases?

Deep Representation

- We talked about transfer learning and that models learn generic features.
- We also mentioned that in a deep learning algorithm we have 2 components:
  - The backbone: responsible for feature extraction.
  - The classifier or regressor on top: specialized in the task in hand.
- Thus, to compress images we can proceed as follows:
  - Apply a convolutional backbone trained on a big and diverse dataset.
  - Apply this backbone on the image to output the extracted features.
- Those features encodes the useful information of an image into a vector.

Deep Representation

- Many architectures are trained on ImageNet and are ready to use.

Available models

Model	Size (MB)	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth	Time (ms) per inference step (CPU)	Time (ms) per inference step (GPU)
Xception	88	79.0%	94.5%	22.9M	81	109.4	8.1
VGG16	528	71.3%	90.1%	138.4M	16	69.5	4.2
VGG19	549	71.3%	90.0%	143.7M	19	84.8	4.4
ResNet50	98	74.9%	92.1%	25.6M	107	58.2	4.6
ResNet50V2	98	76.0%	93.0%	25.6M	103	45.6	4.4
ResNet101	171	76.4%	92.8%	44.7M	209	89.6	5.2
ResNet101V2	171	77.2%	93.8%	44.7M	205	72.7	5.4
ResNet152	232	76.6%	93.1%	60.4M	311	127.4	6.5
ResNet152V2	232	78.0%	94.2%	60.4M	307	107.5	6.6
InceptionV3	92	77.9%	93.7%	23.9M	189	42.2	6.9

Outline

- Introduction
- Principal Component Analysis
- Clustering
  - KMeans
  - DBScan
- Deep Representation
- Exercise

Exercise

- The goal is to detect outliers in a randomly shuffled video.
- We will compare 2 dimensionality reduction algorithms:
  - PCA.
  - Pretrained deep convolutional networks.
- We will apply DBScan to spot the outliers on the compressed data.