10. UNSUPERVISED LEARNING

LEARNING OBJECTIVES

- K-
  Means
- PCA

REVIEW OF LESSON 9

LAST LESSON REVIEW

- Why not use a simple linear regression to predict classification?
- Binomial Logistic Regression
- Transformations in Scikit
- Recap on encoding categorical values

LAST SESSION

ANY QUESTIONS?

- What's the sigmoid function?
- Odds ratio
- ...

TODAY

UNSUPERVISED LEARNING

UNSUPERVISED LEARNING

Not interested in predictions

We only have observations

Can we discover patterns or subgroups, visualize the observations ?

PCA: Principal Compenent analysis for data visualization or data preprocessing

Clustering: for discovering sub groups in the data

CHALLENGES

- Often performed part of exploratory analysis
- Subjective, no way to check
- Difficult to assess the result
- No labels

CLUSTERING

Looks for homogeneous subgroups in the data

Market segmentation

2 most common methods

- K-Means clustering: find a predefined number of clusters / groups of equal variance
- Hierarchical clustering: build nested clusters by merging or splitting them successively. This hierarchy of clusters is represented as a tree (or dendrogram).
- Other clustering algorithms

K-MEANS CLUSTERING

The number of clusters K is abitrary.

Let C_k for $k \in [1, \ldots, K]$ be the list of K clusters.

We make 2 assumptions:

- All observations belong to at least one cluster C_k
- No observations belong to 2 clusters

And define good clustering as clustering for which the *within-cluster* variation is as small as possible.

We want each clusters to have minimal variation between the observations that compose it.

K-MEANS CLUSTERING

A more formal definition is

The KMeans algorithm clusters data by trying to separate samples in ***n groups of equal variance****, minimizing a criterion known as the* ***inertia***

$$Inertia = \sum_{i=0}^{n} \min_{\mu_j \in C_K} (||x_i - \mu_j||^2)$$

K-MEANS CLUSTERING

- Choose K arbitrarily
- [*Initialize*] Randomly assign a number 1 to K to each one of the samples

Then Iterate until convergence

1. For all clusters, calculate the center of all the points in the cluster. This creates K points called centroids.
2. Then reassign all the observations to the cluster that has the closest centroid
3. Until convergence

Using the Euclidian distance between points.

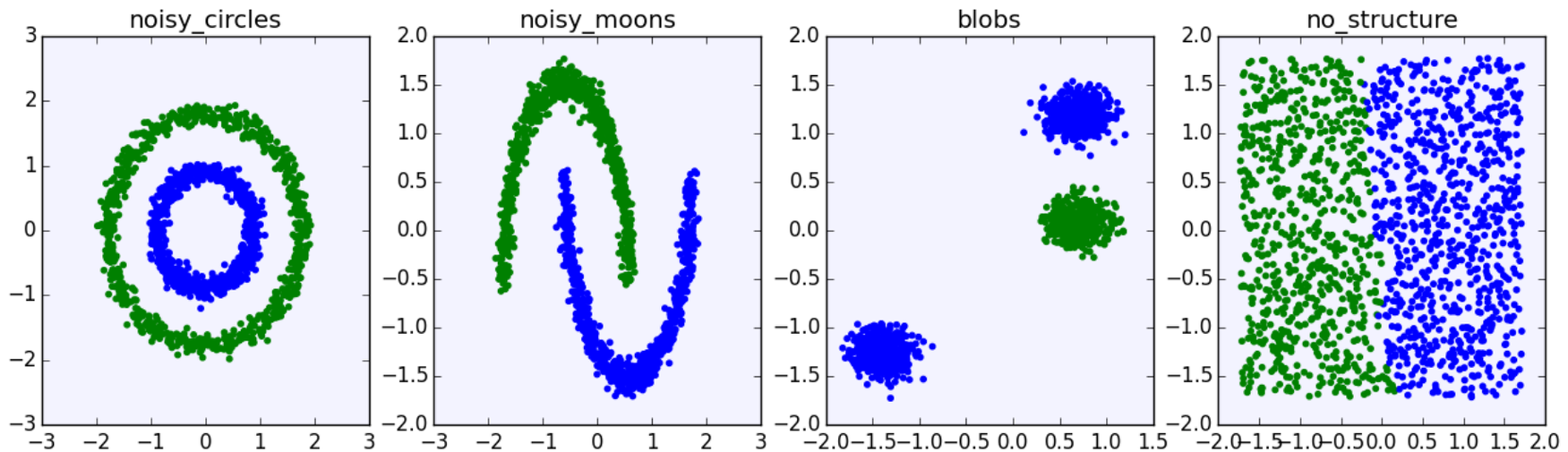
Until convergence: no more changes or very little difference between iterations

K-MEANS CLUSTERING: CHALLENGES

We have a metric: **Inertia**! We can surely select the *Best* clustering! LOL

- Dependent on the initialization => must run the clustering with several initializations and average
- Choosing the number of clusters K is not an easy problem
- Should you normalize your data?
- In the end are you clustering the noise?
- Will outliers skew your clusters?
- Sensitive to perturbations!

LAB



L10_N1_unsupervised-k-means.py

LAB

Make these 4 [datasets]:

```
n_samples = 1500
noisy_circles   = datasets.make_circles(n_samples=n_samples, factor=.5, noise=.05)
noisy_moons     = datasets.make_moons(n_samples=n_samples, noise=.05)
blobs           = datasets.make_blobs(n_samples=n_samples, random_state=8)
no_structure    = np.random.rand(n_samples, 2), None
```

and check how these 3 algorithms perform on each

```
kmeans = cluster.KMeans(n_clusters=2)
dbscan = cluster.DBSCAN(eps=.2)
spectral = cluster.SpectralClustering(n_clusters=2, eigen_solver='arpack', affinity="nearest_neigh
```

Explore the datasets objects you've created, and the k-means scikit model

K-MEANS FINDING K

Make classification with N clusters

Plot the number of clusters against the average variance

MORE ON CLUSTERING

Demonstration of k-means assumption

Vector Quantization

Very thorough article: Cluster Analysis and Segmentation

PART II PRINCIPAL COMPONENT ANALYSIS

Principal component analysis (PCA): Linear dimensionality reduction using **Singular Value Decomposition** of the data and keeping only the most significant singular vectors to project the data to a lower dimensional space.

PCA

DECOMPOSITION OF THE COVARIANCE MATRIX

We have $X|$ data (n x p) n samples, p features

- Calculate the **Covariance Matrix**: $X.X^T|$ => n * n matrix => square => can be decomposed
- There exists unique matrices W and Λ such that $X.X^T = W.\Lambda.W^T|$ where
  - $\Lambda|$ is diagonal (composed of eigenvalues)
  - W is triangular composed of the eigenvectors

=> all the vectors in W are decorrelated!!!

Recall: $\lambda|$ is an eigenvalue and $v|$ an eigenvector of a matrix A $\iff A.v = \lambda.v|$

PCA GEOMETRIC INTERPRETATION

- 1st component (vector) gets the max of the variance in the data
- 2nd gets the rest
- 3rd gets the rest ....

and all these vectors are orthogonal

- The data is projected first on the 1st vector => 1st element of the decomposition
- Then on the 2nd direction => 2nd element

PCA FOR DIMENSIONALITY REDUCTION

Not all lambdas are equal ... in fact the tend to decrease rather fast

DEMO

Find how many eigenvalues do you need to keep to retain 80% of the variation in the data?

Caravan dataset

```
df = pd.read_csv('../../datasets/Caravan.csv', index_col = False)
X = df.values
scale = StandardScaler()
X = scale.fit_transform(X)

# Covariance matrix
covmat = X.T.dot(X)

# Eigenvalues
from scipy import linalg
evs, evmat = linalg.eig(covmat)
evs = evs.astype(float)
```

LAB 2

And now with scikit PCA and the USArrests.csv dataset

```
df = pd.read_csv('../../datasets/USArrests.csv', index_col=0)
```

Decompose with 4 dimensions

Plot the states along the 2 first eigenvectors

LINKS

- PCA by Sebastian Raschka