Hands-on clustering #2:

Ensemble clustering

Joris Guérin

March 2022

Abstract

The objective of this hands-on exercise is to present the concept of ensemble learning for unsupervised data. Through generated toy examples, the student should understand the underlying motivations behind ensemble clustering and get a strong intuition regarding why and how it works. The learned concepts are then applied to a practical example of unsupervised image classification.

1 Objectives

Ensemble learning rely on the idea that several weak learners can learn better predictive models than one strong expert model. In this exercise, we aim at illustrating this concept and show how it can be implemented for clustering (unsupervised classification).

2 Understanding ensemble methods

The underlying principle behind ensemble learning is called *wisdom of the crowd*. It says that:

"if you aggregate many different opinions from a diverse group of people, you are much more likely to arrive at the best opinion than if you just listen to one specialist".

– Simon Kuper.

Let's illustrate this concept through an example. Consider a binary classification problem. A good expert model M^* can reach a classification accuracy of 90%. On the other hand, we build 1001 weak models $\{M_i,\ i \in \{1, ...1001\}\}$, each with an accuracy of 51%. Predictions using the ensemble consist in having all weak models make a prediction and set the final prediction to the one predicted by the majority of the weak models.

1. When we get a new data point to predict, which strategy (expert or ensemble) is more likely to be correct? *Hint: For a Bernoulli distribution with probability of success p, the corresponding Binomial distribution with n trials is represented by* $P(X = k) = \binom{n}{k} p^k (1-p)^{n-k}$, *where* $P(X = k)$ *is the probability of having k successes.*
2. With 1001 models, what minimal accuracy is needed for the weak models to outperform the expert?
3. What are the three conditions required on the weak models for the ensemble setting to work properly. When possible, illustrate them with counter examples.

3 Ensemble methods for clustering

Ensemble clustering approaches are composed of two steps:

- Generation of clustering partitions
- Consensus function to come up with final partition

3.1 Generating clustering partitions

The first step for any ensemble clustering approach is to generate several different clustering partitions of the data. In other words, starting from a dataset $D = \{x_1, ...x_n\}$, containing n data points in $\mathcal{R}^d$, we want to generate N partitions $\{\pi_1, ...\pi_N\}$ such that each $\pi_i \in \mathbb{N}^n$ represent a clustering of D (a set of integer indices representing the clusters associated with a each data point). Of course, as discussed in the previous section, these partition should be good, diversified and sufficiently numerous.

- Before reading further, try to think and make a list of different ways to generate such clustering partitions.

In this section, we will implement the **voting Kmeans** algorithm. It consists in generating many partitions using Kmeans, each containing many small clusters. The main idea behind voting Kmeans is to use a regular Kmeans algorithm with a high number of desired clusters. Then, different partitions are generated using different initialization. We now propose to implement this first partition generation step and apply it to the concentric circle toy data (see the starter notebook).

1. Run regular Kmeans (K=2) on the concentric circle data. How well does it perform? Compute the corresponding Normalized Mutual Information (NMI).
2. Implement the voting Kmeans generation mechanism (100 partitions with 100 clusters each). Visualize some of these partitions. Do they seem to verify the ensemble learning hypothesis?

3.2 Consensus function

To obtain the final clustering results, we need to combine these results.

- Can we use the same grouping strategy as in Section 2? Why?
- Before reading further, try to think of possible ways to combine clustering results into a unique partition with the final desired number of clusters.

Here we propose to implement the consensus function based on the Co-Association Matrix. A Co-Association Matrix is a $n \times n$ symmetric matrix, with zeros on the diagonal and where entry (i, j), $i \neq j$, represents the number of times x_i and x_j where classified together in the different partitions. This matrix can be used as a similarity matrix between data points. To obtain the final partition, we can then run a clustering algorithm that works with precomputed distances (graph-based or density-based clustering algorithms).

1. Build a function to compute the Co-Association Matrix for a given set of partitions.
2. What clustering algorithm can be used to generate a final clustering partition using the Co-Association Matrix to represent distance between points. Using this, generate the final partition of voting Kmeans. (*Hint: Does the Co-Association Matrix represent the similarity or dissimilarity between points?*)
3. Did it work as expected? If not, look at the default initialization parameters of Kmeans (init, n_init), why did it fail? Try again.
4. Keeping the number of partition to 100. Find experimentally the good number of clusters in each partitions? (We can compute the NMI for different values of the number of clusters, between 10 and 25 with a step of 2 is sufficient). *As a rule of thumb, $K \sim \sqrt{n}$ is generally a good choice.*

In practice, good results on the concentric data could be obtained using simply an Agglomerative Clustering algorithm. To illustrate the power of Ensemble Clustering approaches, repeat the above process on the elongated blobs data, where Agglomerative Clustering is actually not successful.

4 Practical implementation of Ensemble Clustering

We now propose to solve an unsupervised image classification problem using ensemble clustering. The objective is to recover groups of similar images, without ground truth labels and preliminary training. This setting can be used in practice to suggest similar image contents for web search, or to organize picture folders automatically, for example.

For this study, you will need two data folders:

- FEI/representations
- FEI/images

FEI is an image dataset representing faces of different people under different camera angles and lighting conditions. The "images" folder contains the original dataset that we want to cluster.

- Look at the images and try to understand well the task to solve.

The "representations" folder represents feature vectors extracted from the images using different deep Convolutional Neural Networks (CNNs). CNNs network are large Machine Learning models able to extract features representing high-level characteristics about images. None of the CNNs used here were actually trained to classify human faces. The goal here is to use these different views of the same data, to generate different clustering partitions using Kmeans. Then, we will use the Co-Association Matrix consensus to generate a single final ensemble partition. We provide the code to read the ".h5" partition files:

```
import h5py

filename = "path_to_file.h5"
f = h5py.File(filename, "r")
data = np.array(f["data"])
labels = np.array(f["labels"])
```

The labels correspond to the ground truth classification of the face images in FEI. In practice, they are not used for clustering, but we will use them to compare the results obtained with different approaches.

1. Load a single representation and run Kmeans on it. Compute the NMI to get an idea of how good the generated clustering partition is.
2. Repeat this process for all the different representation and build an ensemble partition using all these different representations. Are the results satisfying? Discuss if the three ensemble learning conditions are verified in this example.

5 Going further

1. Use common Machine Learning datasets available in scikit-learn (iris, wine, breast_cancer) to test different ideas you had earlier for generating weak clustering partitions (Section 3.1).