Project 2: Clustering

Due Feb 3rd, 2021 by 11:59 pm

Introduction

Clustering algorithms are unsupervised methods for finding groups of data points that have similar representations in a feature space. Clustering differs from classification in that no *a priori* labeling (grouping) of the data points is available.

K-means K-means clustering is a simple and popular clustering algorithm. Given a set of data points $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ in multidimensional space, it tries to find K clusters such that each data point belongs to exactly one cluster, and that the sum of the squares of the distances between each data point and the center of the cluster it belongs to is minimized. If we define $\boldsymbol{\mu}_k$ to be the "center" of the kth cluster, and

$$r_{nk} = \begin{cases} 1, & \text{if } \mathbf{x}_n \text{ is assigned to cluster } k \\ 0, & \text{otherwise} \end{cases}, \quad n = 1, \ldots, N \quad k = 1, \ldots, K$$

Then our goal is to find r_{nk}'s and $\boldsymbol{\mu}_k$'s that minimize $J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2$. The approach of K-means algorithm is to repeatedly perform the following two steps until convergence:

1. (Re)assign each data point to the cluster whose center is nearest to the data point.
2. (Re)calculate the position of the centers of the clusters: setting the center of the cluster to the mean of the data points that are currently within the cluster.

The center positions may be initialized randomly.

Hierarchical Clustering Hierarchical clustering is a general family of clustering algorithms that build nested clusters by merging or splitting them successively. This hierarchy of clusters is represented as a tree (or dendrogram). A flat clustering result is obtained by cutting the dendrogram at a level that gives desired number of clusters.

DBSCAN DBSCAN or Density-Based Spatial Clustering of Applications with Noise finds core samples of high density and expands clusters from them. It is a density-based clustering non-parametric algorithm: given a set of points in some space, it groups together points that are closely packed together (points with many nearby neighbors), marking as outliers points that lie alone in low-density regions (whose nearest neighbors are too far away).

HDBSCAN HDBSCAN extends DBSCAN by converting it into a hierarchical clustering algorithm, and then using a technique to extract a flat clustering based in the stability of clusters. The resulted algorithm gets rid of the hyperparameter "epsilon", which is necessary to DBSCAN (see here for more analysis on that).

In this project, the goal includes:

1. To find proper representations of the data, s.t. the clustering is efficient and gives out reasonable results.
2. To perform K-means, HDBscan and DBscan clustering on the dataset, and evaluate the result of the clustering.
3. To try different preprocessing methods which may increase the performance of the clustering.

Dataset

We work with "20 Newsgroups" dataset that we already explored in **Project 1**. It is a collection of approximately 20,000 documents, partitioned (nearly) evenly across 20 different newsgroups, each corresponding to a different category (topic). Each topic can be viewed as a "class".

In order to define the clustering task, we pretend as if the class labels are not available and aim to find groupings of the documents, where documents in each group are more similar to each other than to those in other groups. We then use class labels as the ground truth to evaluate the performance of the clustering task.

To get started with a simple clustering task, we work with a well-separable portion of the data set that we used in Project 1, and see if we can retrieve the known classes. Specifically, let us define two classes comprising of the following categories.

Table 1: Two well-separated classes

Class 1	`comp.graphics`	`comp.os.ms-windows.misc`	`comp.sys.ibm.pc.hardware`	`comp.sys.mac.hardware`
Class 2	`rec.autos`	`rec.motorcycles`	`rec.sport.baseball`	`rec.sport.hockey`

We would like to evaluate how purely the *a priori* known classes can be reconstructed through clustering. That is, we take all the documents belonging to these two classes and perform unsupervised clustering into two clusters. Then we determine how pure each cluster is when we look at the labels of the documents belonging to each cluster.

Part 1 - Clustering of Text Data

Basic Workflow

1. Building the TF-IDF matrix.

   Following the steps in Project 1, **transform the documents into TF-IDF vectors**.

   Use `min_df = 3`, exclude the stopwords (no need to do stemming or lemmatization), and **remove** the headers and footers.

   QUESTION 1: Report the dimensions of the TF-IDF matrix you get.

2. Apply K-means clustering with $k = 2$ using the TF-IDF data. Note that the `KMeans` class in `sklearn` has parameters named `random_state`, `max_iter` and `n_init`. Please use `random_state=0`, `max_iter` ≥ 1000 and `n_init` ≥ 30[1]. Compare the clustering results

[1] If you have enough computation power, the larger the better

with the known class labels. (you can refer to sklearn - Clustering text documents using k-means for a basic work flow)

(a) Given the clustering result and ground truth labels, contingency table **A** is the matrix whose entries A_{ij} is the number of data points that belong to both the class C_i the cluster K_j.

QUESTION 2: Report the contingency table of your clustering result. You may use the provided `plotmat.py` to visualize the matrix.

(b) In order to evaluate clustering results, there are various measures for a given partition of the data points with respect to the ground truth. We will use the measures **homogeneity score**, **completeness score**, **V-measure**, **adjusted Rand score** and **adjusted mutual info score**, all of which can be calculated by the corresponding functions provided in `sklearn.metrics`.

- **Homogeneity** is a measure of how "pure" the clusters are. If each cluster contains only data points from a single class, the homogeneity is satisfied.
- On the other hand, a clustering result satisfies **completeness** if all data points of a class are assigned to the same cluster. Both of these scores span between 0 and 1; where 1 stands for perfect clustering.
- The **V-measure** is then defined to be the harmonic average of homogeneity score and completeness score.
- The **adjusted Rand Index** is similar to accuracy measure, which computes similarity between the clustering labels and ground truth labels. This method counts all pairs of points that both fall either in the same cluster and the same class or in different clusters and different classes.
- Finally, the **adjusted mutual information score** measures the mutual information between the cluster label distribution and the ground truth label distributions.

QUESTION 3: Report the 5 measures above for the K-means clustering results you get.

Dimensionality Reduction and Data Transformation

As you may have observed, high dimensional sparse TF-IDF vectors do not yield a good clustering result. One of the reasons is that in a high-dimensional space, the Euclidean distance is not a good metric anymore, in the sense that the distances between data points tends to be almost the same (see [1]).

K-means clustering has other limitations. Since its objective is to minimize the sum of within-cluster l_2 distances, it implicitly assumes that the clusters are isotropically shaped, *i.e.* round-shaped. When the clusters are not round-shaped, K-means may fail to identify the clusters properly. Even when the clusters are round, K-means algorithm may also fail when the clusters have unequal variances. A direct visualization for these problems can be found at sklearn - Demonstration of k-means assumptions.

In this part we try to find a "better" representation tailored to the way that K-means clustering algorithm works, by reducing the dimension of our data before clustering and applying other transformations.

1. Dimensionality Reduction

(a) First we want to find the effective dimension of the data through inspection of the top singular values of the TF-IDF matrix and see how many of them are significant in reconstructing the matrix with the truncated SVD representation. A guideline is to see what ratio of the variance of the original data is retained after the dimensionality reduction.

QUESTION 4: Report the plot of the percent of variance the top r principle components can retain v.s. r, for $r = 1$ to 1000.

Hint: `explained_variance_ratio_` of `TruncatedSVD` objects. See sklearn document.

(b) Now, use the following two methods to reduce the dimension of the data. Sweep over the dimension parameters for each method, and choose one that yields better results in terms of clustering purity metrics.

- Truncated SVD / PCA

  Note that you don't need to perform SVD multiple times: performing SVD with $r = 1000$ gives you the data projected on all the top 1000 principle components, so for smaller r's, you just need to exclude the least important features.
- NMF

QUESTION 5:
Let r be the dimension that we want to reduce the data to (*i.e.* `n_components`).
Try $r = \mathbf{1}, \mathbf{2}, \mathbf{3}, \mathbf{5}, \mathbf{10}, \mathbf{20}, \mathbf{50}, \mathbf{100}, \mathbf{300}$, and plot the 5 measure scores v.s. r for both SVD and NMF.
Report a good choice of r for SVD and NMF respectively.

Note: In the choice of r, there is a trade-off between the information preservation, and better performance of k-means in lower dimensions.

QUESTION 6: How do you explain the non-monotonic behavior of the measures as r increases?

2. Visualization. We can visualize the clustering results by projecting the dim-reduced data points onto 2-D plane with SVD, and coloring the points according to

   - Ground truth class label
   - Clustering label

   respectively.

   QUESTION 7: Visualize the clustering results for:

   - SVD with your choice of r
   - NMF with your choice of r

   QUESTION 8: What do you observe in the visualization? How are the data points of the two classes distributed? Is the data distribution ideal for K-Means clustering?

3. More challenging clustering. We have been dealing with a relatively simple clustering task with only two well-separated classes. Now let's face a more challenging one: clustering for the entire 20 categories in the 20newsgroups dataset.

QUESTION 9: Load documents with the same configuration as in Question 1, but for ALL 20 categories. Construct the TF-IDF matrix, reduce its dimensionality properly using either NMF or SVD, and perform K-Means clustering with `n_components=20`. **Visualize the contingency matrix and report the five clustering metrics.**

There is a mismatch between cluster labels and class labels. For example, the cluster #3 may correspond to the class #8. As a result, the high-value entries of the 20×20 contingency matrix can be scattered around, making it messy to inspect, even if the clustering result is not bad.

One can use `scipy.optimize.linear_sum_assignment` to identify the best-matching cluster-class pairs, and permute the columns of the contingency matrix accordingly. See below for an example:

```
import numpy as np
from plotmat import plot_mat # using the provided plotmat.py
from scipy.optimize import linear_sum_assignment
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(labels, clustering_labels)

rows, cols = linear_sum_assignment(cm, maximize=True)

plot_mat(cm[rows[:, np.newaxis], cols], xticklabels=cols,
 ↪  yticklabels=rows, size=(15,15))
```

QUESTION 10: Kullback-Leibler Divergence for NMF

By default `sklearn.decomposition.NMF` uses Frobenius norm as its cost function. Another choice is Kullback-Leibler divergence. It is shown that with this cost function, NMF is equivalent to Probabilistic Latent Semantic Analysis (PLSA)[4]. Try Kullback-Leibler divergence for NMF and see whether it helps with the clustering of our text data. **Report the five evaluation metrics.**

4. Yet another dimensionality reduction tool: UMAP.

   The clustering result is probably not very satisfactory for the 20 categories data. Therefore we introduce a more advanced dimensionality reduction method: Uniform Manifold Approximation and Projection (UMAP). It constructs a graphical representation of the high-dimensional data manifold, and learns a low-dimensional embedding based on the representation.

   UMAP allows more choices of distance metric besides Euclidean distance. In particular, we are interested in "cosine distance", because it ignores the magnitude of the vectors, meaning that the length of the documents does not affect the distance metric.[2]

   QUESTION 11: Use UMAP to reduce the dimensionality of the 20 categories TF-IDF matrix, and apply K-Means clustering with `n_components=20`.

   Find a good `n_components` choice for UMAP, and compare the performance of two metrics by setting `metric="euclidean"` and `metric="cosine"` respectively.

[2] For example, two documents are about the same topic and are similar, but one is very long while the other is short. The magnitude of the TF-IDF vector will be high for the long document and low for the short one, even thought the orientation of their TF-IDF vectors might be very close. In this case, cosine distance will correctly identify the similarity but Euclidean distance might fail to.

Report the permuted contingency matrix and the five clustering evaluation metrics for `"euclidean"` and `"cosine"`.

QUESTION 12: Analyze the contingency matrix.

From the contingency matrices, identify the categories that are prone to be confused with each other. Does the result make sense?

More Clustering Algorithms

1. Agglomerative clustering.

   The `AgglomerativeClustering` object performs a hierarchical clustering using a bottom up approach: each observation starts in its own cluster, and clusters are successively merged together. There are 4 linkage criteria that determines the merge strategy.

   QUESTION 13: Use UMAP to reduce the dimensionality properly, and perform Agglomerative clustering with `n_clusters=20`. Compare the performance of "ward" and "single" linkage criteria.

   Report the five clustering evaluation metrics for each case.

2. DBSCAN and HDBSCAN

   QUESTION 14: Apply DBSCAN and HDBSCAN on UMAP-transformed 20-category data. Use `min_cluster_size=100`.

   Experiment on the hyperparameters and report your findings in terms of the five clustering evaluation metrics.

   QUESTION 15: Contingency matrix

   Plot the permuted contingency matrix for the best clustering model from Question 14.

   How many clusters are given by the model? What does "-1" mean for the clustering labels?

   Interpret the contingency matrix considering the answer to these questions.

Part 2 - BBCNews Dataset

Using the dataset provided on Kaggle: `https://www.kaggle.com/c/learn-ai-bbc`

QUESTION 16: Report your process:

- data acquiring,
- feature engineering (doesn't need to be the same as those in part 1),
- clustering,
- performance evaluation

Submission

Please submit a PDF report to Gradescope via CCLE. You can find a link to Gradescope in the left panel on CCLE.

In addition, please submit a zip file containing your **report**, and your **codes** with a **readme file** on how to run your code to CCLE. The zip file should be named as "Project2_UID1_UID2_..._UIDn.zi where UIDx's are student ID numbers of the team members. Only one submission per team is required. If you had any questions, please ask on Piazza or through email.

References

[1] Why is Euclidean distance not a good metric in high dimensions? [online]. (https://stats.stackexchange.com/questions/99171/why-is-euclidean-distance-not-a-good-metric-in-high-dimensions).

[2] https://en.wikipedia.org/wiki/DBSCAN

[3] https://hdbscan.readthedocs.io/en/latest/how_hdbscan_works.html

[4] Gaussier, E., & Goutte, C. (2005). Relation between PLSA and NMF and implications. Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 601–602. https://doi.org/10.1145/1076034.1076148