Large-Scale Data Mining: Models and Algorithms **ECE 219 Winter 2023**
Prof. Vwani Roychowdhury
UCLA, Department of ECE

Project 2: Data Representations and Clustering

Due Feb 12, 2023 by 11:59 pm

Introduction

Machine learning algorithms are applied to a wide variety of data, including text and images. Before applying these algorithms, one needs to convert the raw data into feature representations that are suitable for downstream algorithms. In project 1, we studied feature extraction from text data, and the downstream task of classification. We also learned that reducing the dimension of the extracted features often helps with a downstream task.

In this project, we explore the concepts of feature extraction and clustering together. In an ideal world, all we need are data points – encoded using certain features– and AI should be able to find what is important to learn, or more specifically, determine what are the underlying modes or categories in the dataset. This is the ultimate goal of General AI: the machine is able to bootstrap a knowledge base, acting as its own teacher and interacting with the outside world to explore to be able to operate autonomously in an environment.

We first explore this field of unsupervised learning using textual data, which is a continuation of concepts learned in Project 1. We ask if a combination of feature engineering and clustering techniques can automatically separate a document set into groups that match known labels.

Next we focus on a new type of data, i.e. images. Specifically, we first explore how to use "deep learning" or "deep neural networks (DNNs)" to obtain image features. Large neural networks have been trained on huge labeled image datasets to recognize objects of different types from images. For example, networks trained on the Imagenet dataset can classify more than one thousand different categories of objects. Such networks can be viewed as comprising two parts: the first part maps a given RGB image into a feature vector using convolutional filters, and the second part then classifies this feature vector into an appropriate category, using a fully-connected multi-layered neural network (we will study such NNs in a later lecture). Such pre-trained networks could be considered as experienced agents that have learned to discover features that are salient for image understanding. Can one use the experience of such pre-trained agents in understanding new images that the machine has never seen before? It is akin to asking a human expert on forensics to explore a new crime scene. One would expect such an expert to be able to *transfer* their domain knowledge into a new scenario. In a similar vein, can a pre-trained network for image understanding be used for **transfer learning**? One could use the output of the network in the last few layers as expert features. Then, given a multi-modal dataset –consisting of images from categories that the DNN was not trained for– one can use feature engineering (such as dimensionality reduction) and clustering algorithms to automatically extract unlabeled categories from such expert features.

For both the text and image data, one can use a common set of multiple evaluation metrics to compare the groups extracted by the unsupervised learning algorithms to the corresponding ground truth human labels.

Clustering Methods

Clustering is the task of grouping a dataset in such a way that data points in the same group (called a cluster) are more similar (in some sense) to each other than to those in other groups (clusters). Thus, there is an inherent notion of a metric that is used to compute similarity among data points, and different clustering algorithms differ in the type of similarity measure they use, e.g., Euclidean vs Riemannian geometry. Clustering algorithms are considered "unsupervised learning", i.e. they do not require labels during training. In principle, if two categories of objects or concepts are distinct from some perspective (e.g. visual or functional), then data points from these two categories – when properly coded in a feature space and augmented with an associated distance metric – should form distinct clusters. Thus, if one can perform perfect clustering then one can discover and obtain computational characterizations of categories without any labeling. In practice, however, finding such optimal choices of features and metrics has proven to be a computationally intractable task, and any clustering result needs to be validated against tasks for which one can measure performance. Thus, we use labeled datasets in this project, which allows us to evaluate the learned clusters by comparing them with ground truth labels.

Below, we summarize several clustering algorithms:

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, and a hyperparameter K denoting the number of clusters, the algorithm finds the K cluster centers such that each data point belongs to exactly one cluster. This cluster membership is found by minimizing the sum of the squares of the distances between each data point and the center of the cluster it belongs to. 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 $J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \left\| \mathbf{x}_n - \boldsymbol{\mu}_k \right\|^2$. 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 builds 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 yields a 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, the algorithm 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 an empirical technique to extract a flat clustering based on the stability of clusters (similar to the elbow method in k-Means). The resulting algorithm gets rid of the hyperparameter "epsilon", which is necessary in DBSCAN (see here for more on that).

Common Clustering Evaluation Metrics

In order to evaluate a clustering pipeline, one can use the ground-truth class labels and compare them with the cluster labels. This analysis determines the quality of the clustering algorithm in recovering the ground-truth underlying labels. It also indicates if the adopted feature extraction and dimensionality reduction methods retain enough information about the ground-truth classes. Below we provide several evaluation metrics available in `sklearn.metrics`. *Note that for the clustering sub-tasks, you do not need to separate your data to training and test sets.* Question 25 requires you to split the data.

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.

Completeness indicates how much of the data points of a class are assigned to the same cluster.

V-measure is the harmonic average of homogeneity score and completeness score.

Adjusted Rand Index is similar to accuracy, 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.

Adjusted mutual information score measures the mutual information between the cluster label distribution and the ground truth label distributions.

Dimensionality Reduction Methods

In project 1, we studied SVD/PCA and NMF as linear dimensionality reduction techniques. Here, we consider some additional non-linear methods.

Uniform Manifold Approximation and Projection (UMAP) The UMAP algorithm constructs a graph-based representation of the high-dimensional data manifold, and learns a low-dimensional representation space based on the relative inter-point distances. UMAP allows more choices of distance metrics besides Euclidean distance. In particular, we are interested in "cosine distance" for text data, because as we shall see it bypasses the magnitude of the vectors, meaning that the length of the documents does not affect the distance metric.

Autoencoders An autoencoder[1] is a special type of neural network that is trained to copy its input to its output. For example, given an image of a handwritten digit, an autoencoder first encodes the image into a lower dimensional latent representation, then decodes the latent representation back to an image. An autoencoder learns to compress the data while minimizing the reconstruction error. Further details can be found in chapter 14 of [4].

[1] also known as "auto-associative networks" in older jargon

Part 1 - Clustering on Text Data

In this part of the project, we will work with "20 Newsgroups" dataset, which is a collection of approximately 20, 000 documents, partitioned (nearly) evenly across 20 different newsgroups (newsgroups are discussion groups like forums, which originated during the early age of the Internet), each corresponding to a different topic. Use the `fetch_20newsgroups` function from `scikit-learn` to load the dataset. Detailed usage of the dataset and sample code can be found at this link.

To get started with a simple clustering task, we work with a well-separable subset of samples from the larger dataset. Specifically, we 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`

Clustering with Sparse Text Representations

1. **Generate sparse TF-IDF representations:** 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. No need to do any additional data cleaning.

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

2. **Clustering:** 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[2]. 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 $\mathbf{A}$ is the matrix whose entries $\mathbf{A}_{ij}$ is the number of data points that belong to the i'th class and the j'th cluster.

   QUESTION 2: Report the contingency table of your clustering result. You may use the provided `plotmat.py` to visualize the matrix. Does the contingency matrix have to be square-shaped?

   QUESTION 3: Report the 5 clustering measures explained in the introduction for K-means clustering.

Clustering with Dense Text Representations

As you may have observed, high-dimensional sparse TF-IDF vectors do not yield a good clustering result, especially when K-Means clustering is used. 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.*

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

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 performance of the downstream task of K-means clustering. Towards finding a better representation, we can reduce the dimension of our data with different methods before clustering.

1. **Generate dense representations for better K-Means Clustering**

   (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 percentage of variance that the top r principle components 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 $r = \mathbf{1-10, 20, 50, 100, 300}$, 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?

   QUESTION 7: Are these measures on average better than those computed in Question 3?

2. **Visualize the clusters**

   We can visualize the clustering results by projecting the dimension-reduced data points onto a 2-D plane by once again using SVD, and coloring the points according to the:

   - Ground truth class label;
   - Clustering label

   respectively.

QUESTION 8: Visualize the clustering results for:

- SVD with your optimal choice of r for K-Means clustering;
- NMF with your choice of r for K-Means clustering.

To recap, you can accomplish this by first creating the dense representations and then once again projecting these representations into a 2-D plane for visualization.

QUESTION 9: What do you observe in the visualization? How are the data points of the two classes distributed? Is distribution of the data ideal for K-Means clustering?

3. Clustering of the Entire 20 Classes

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 10: Load documents with the same configuration as in Question 1, but for ALL 20 categories. Construct the TF-IDF matrix, reduce its dimensionality using BOTH NMF and SVD (specify settings you choose and why), and perform K-Means clustering with `k=20`. **Visualize the contingency matrix and report the five clustering metrics (DO BOTH NMF AND SVD).**

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:

```python
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))
```

4. UMAP

The clustering performance is poor for the 20 categories data. To see if we can improve this performance, we consider UMAP for dimensionality reduction. UMAP uses cosine distances to compare representations. Consider two documents that 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 though the orientation of their TF-IDF vectors might be very close. In this case, the cosine distance adopted by UMAP will correctly identify the similarity, whereas Euclidean distance might fail.

QUESTION 11: Reduce the dimension of your dataset with UMAP. Consider the following settings: `n_components` $= [5, 20, 200]$, `metric` $=$ "cosine" vs. "euclidean". If "cosine" metric fails, please look at the FAQ at the end of this spec.

Report the permuted contingency matrix and the five clustering evaluation metrics for the different combinations (6 combinations).

QUESTION 12: Analyze the contingency matrices. Which setting works best and why? What about for each metric choice?

QUESTION 13: So far, we have attempted K-Means clustering with 4 different representation learning techniques (sparse TF-IDF representation, PCA-reduced, NMF-reduced, UMAP-reduced). Compare and contrast the clustering results across the 4 choices, and suggest an approach that is *best* for the K-Means clustering task on the 20-class text data. Choose any choice of clustering metrics for your comparison.

Clustering Algorithms that do not explicitly rely on the Gaussian distribution per cluster

While we have successfully shown in the previous section that some representation learning techniques perform better than others for the task of K-Means clustering on this text dataset, this sweep only covers a half of the end-to-end solution for representation learning. What if we changed the clustering method? In this section we introduce 2 additional 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 14: 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. **HDBSCAN**

   QUESTION 15: Apply HDBSCAN on UMAP-transformed 20-category data.

   Use `min_cluster_size=100`.

   Vary the min_cluster_size among $20, 100, 200$ **and report your findings in terms of the five clustering evaluation metrics - you will plot the best contingency matrix in the next question. Feel free to try modifying other parameters in HDBSCAN to get better performance.**

   QUESTION 16: Contingency matrix

   Plot the contingency matrix for the best clustering model from Question 15.

   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.

QUESTION 17: Based on your experiments, which dimensionality reduction technique and clustering methods worked best together for 20-class text data and why? Follow the table below. If UMAP takes too long to converge, consider running it once and saving the intermediate results in a pickle file.
Hint: DBSCAN and HDBSCAN do not accept the number of clusters as an input parameter. So pay close attention to how the different clustering metrics are being computed for these methods.

Module	Alternatives	Hyperparameters
Dimensionality Reduction	None	N/A
	SVD	r = [5,20,200]
	NMF	r = [5,20,200]
	UMAP	n_components = [5,20,200]
Clustering	K-Means	k = [10,20,50]
	Agglomerative Clustering	n_clusters = [20]
	HDBSCAN	min_cluster_size = [100,200]

QUESTION 18: Extra credit: If you can find creative ways to further enhance the clustering performance, report your method and the results you obtain.

Part 2 - Deep Learning and Clustering of Image Data

In this part, we aim to cluster the images of the `tf_flowers` dataset. This dataset consists of images of five types of flowers. Explore this link to see actual samples of the data.

Extracting meaningful features from images has a long history in computer vision. Instead of considering the raw pixel values as features, researchers have explored various hand-engineered feature extraction methods, e.g. [5]. With the recent rise of "deep learning", these methods are replaced with using appropriate neural networks. Particularly, one can adopt a neural network already trained to classify another large dataset of images[3]. These pre-trained networks have been trained to morph the highly non-smooth scatter of images in the higher dimension, into smooth lower-dimensional manifolds.

In this project, we use a VGG network [6] pre-trained on the ImageNet dataset [7]. We provide **a helper codebase** (check Week 4 in BruinLearn), which guides you through the necessary steps for loading the VGG network and for using it for feature extraction.

QUESTION 19: In a brief paragraph discuss: If the VGG network is trained on a dataset with perhaps totally different classes as targets, why would one expect the features derived from such a network to have discriminative power for a custom dataset?

Use the helper code to load the flowers dataset, and extract their features. To perform computations on deep neural networks fast enough, GPU resources are often required. GPU resources can be freely accessed through "Google Colab".

QUESTION 20: In a brief paragraph explain how the helper code base is performing feature extraction.

QUESTION 21: How many pixels are there in the original images? How many features does the VGG network extract per image; i.e what is the dimension of each feature vector for an image sample?

[3] Such an approach, in which the knowledge gained to solve one problem, is applied to a different but related problem, is often referred to as "transfer learning". We visited another instance of transfer learning when we used GLoVe vectors for text classification in project 1.

QUESTION 22: Are the extracted features dense or sparse? (Compare with sparse TF-IDF features in text.)

QUESTION 23: In order to inspect the high-dimensional features, t-SNE is a popular off-the-shelf choice for visualizing Vision features. Map the features you have extracted onto 2 dimensions with t-SNE. Then plot the mapped feature vectors along x and y axes. Color-code the data points with ground-truth labels. Describe your observation.

While PCA is a powerful method for dimensionality reduction, it is limited to "linear" transformations. This might not be particularly good if a dataset is distributed non-linearly. An alternative approach is use of an "autoencoder" or UMAP. The helper has implemented an autoencoder which is ready to use.

QUESTION 24: Report the best result (in terms of rand score) within the table below.
For HDBSCAN, introduce a conservative parameter grid over `min_cluster_size` and `min_samples`.

Module	Alternatives	Hyperparameters
Dimensionality Reduction	None	N/A
	SVD	r = 50
	UMAP	n_components = 50
	Autoencoder	num_features = 50
Clustering	K-Means	k = 5
	Agglomerative Clustering	n_clusters = 5
	HDBSCAN	min_cluster_size & min_samples

Lastly, we can conduct an experiment to ensure that VGG features are rich enough in information about the data classes. In particular, we can train a fully-connected neural network classifier to predict the labels of data. For this task, you may use the `MLP`[4] module provided in the helper code base.

QUESTION 25: Report the test accuracy of the MLP classifier on the original VGG features. Report the same when using the reduced-dimension features (you have freedom in choosing the dimensionality reduction algorithm and its parameters). Does the performance of the model suffer with the reduced-dimension representations? Is it significant? Does the success in classification make sense in the context of the clustering results obtained for the same features in Question 24.

FAQ

- **Help! My UMAP function works with the Euclidean distance but not cosine?**: Please follow the following library versions to get UMAP to work.

```
annoy==1.17.0
cython==0.29.21
fuzzywuzzy==0.18.0
hdbscan==0.8.26
joblib==1.0.0
kiwisolver==1.3.1
llvmlite==0.35.0
matplotlib==3.3.2
```

[4] Multi-Layer Perceptron

```
numba==0.52.0
numpy==1.20.0
pandas==1.1.2
pillow==8.1.0
pyarrow==1.0.1
python-levenshtein==0.12.1
pytz==2021.1
scikit-learn==0.24.1
scipy==1.6.0
six==1.15.0
threadpoolctl==2.1.0
tqdm==4.50.0
umap-learn==0.5.0
```

Submission

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

In addition, please submit a zip file containing your **report**, and your **codes** with a **readme file** on how to run your code to BruinLearn. The zip file should be named as "Project2_UID1_UID2_..._UIDn.zip" where UIDx's are student ID numbers of the team members. Only one submission per team is required. If you have 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] Heaton, J. (2018). Ian goodfellow, yoshua bengio, and aaron courville: Deep learning.

[5] Rybski, P. E., Huber, D., Morris, D. D., & Hoffman, R. (2010, June). Visual classification of coarse vehicle orientation using histogram of oriented gradients features. In 2010 IEEE Intelligent vehicles symposium (pp. 921-928). IEEE.

[6] Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

[7] Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., & Fei-Fei, L. (2009, June). Imagenet: A large-scale hierarchical image database. In 2009 IEEE conference on computer vision and pattern recognition (pp. 248-255). Ieee.