

Foundation of Artificial Intelligence

Assignement 3

Davide Vitagliano

Academic Year 2023/2024

Sommario

[1. Requirements 1](#_Toc157442012)

[2. Introduction 1](#_Toc157442013)

[3. Designing the solution 2](#_Toc157442014)

[3.1. Gaussian Mixture 3](#_Toc157442015)

[3.1.1. Formalization 3](#_Toc157442016)

[3.1.2. Code explanation 4](#_Toc157442017)

[3.1.3. Conclusion 4](#_Toc157442018)

[3.2. Normalized Cut 4](#_Toc157442019)

[3.2.1. Formalization 4](#_Toc157442020)

[3.2.2. Code explanation 4](#_Toc157442021)

[3.2.3. Conclusion 5](#_Toc157442022)

[3.3. Mean Shift 5](#_Toc157442023)

[3.3.1. Formalization 5](#_Toc157442024)

[3.3.2. Code Explanation 6](#_Toc157442025)

[3.3.3. Conclusion 6](#_Toc157442026)

[4. Comparison 6](#_Toc157442027)

# Requirements

Perform clustering of the MNIST dataset using:

* Gaussian Mixture with diagonal covariance;
* Spectral Clustering with Normalized Cut;
* Mean Shift.

The clustering is performed at varying levels of dimensionality reduction through PCA from 2 to 72 with a step length of 10. PCA is used to assess the effect of the dimensionality in learning time and accuracy.

We also vary the extracted clusters as the number of clusters ‘k’ varies from 10 to 15, for the Gaussian Mixture and Normalized Cut, while for Mean Shift we vary the kernel width.

For this assignment we used GPU acceleration to increase the performance of the training process using NVIDIA CUDA Toolkit.

# Introduction

The first thing to do during the training process is applying feature selection for dimensionality reduction. This is a process of selecting a subset of the original features that are most relevant and informative for the classification. The main benefits of feature selection are:

* Reducing computational cost: fewer features means less time and space required for training and testing models;
* Improving model performance: removing irrelevant or redundant features can reduce noise and overfitting, and increase accuracy and generalization.

In our case we use Principal Components Analysis to reduce the dimensionality of the dataset by maximizing the variance of each dimension. Then we choose which model to train to classify the dataset.

Mixture of Gaussians clustering is a probabilistic model that is used to group data points into clusters based on their similarity. Each cluster is modeled as a Gaussian distribution with its own mean and covariance matrix. The algorithm estimates the parameters of these Gaussians by iteratively computing the probability of each data point belonging to each cluster and updating the parameters of the Gaussians based on these probabilities. This algorithm is useful when the data points are not clearly separable and belong to multiple clusters with different means and variances. It's commonly used in image segmentation, where the goal is to partition an image into regions that correspond to different objects or backgrounds.

Normalized Cut clustering is a technique used in multivariate statistics to perform dimensionality reduction before clustering in fewer dimensions. It’s a type of spectral clustering that makes use of the spectrum, eigenvalues, of the similarity matrix of the data to perform clustering in fewer dimensions. The similarity matrix is provided as an input and consists of a quantitative assessment of the relative similarity of each pair of points in the dataset. It’s commonly used for image segmentation and partitions points into two sets based on the eigenvector corresponding to the second-smallest eigenvalue of the symmetric normalized Laplacian matrix.

Mean Shift clustering is a clustering algorithm that can identify clusters in a dataset with arbitrary shapes and sizes. It works by updating candidates for centroids to be the mean of the points within a given region. These candidates are then filtered in a post-processing stage to eliminate near-duplicates to form the final set of centroids. The algorithm aims to discover “blobs” in a smooth density of samples. It’s a centroid-based algorithm that uses a flat kernel. Seeding, which is choosing the initial cenroids, is performed using a binning technique for scalability. Mean Shift clustering is widely used in computer vision, image processing, and bioinformatics.

In opposition of supervised learning, where we use accuracy as an indicator of performance of the classifier, in clustering we use the rand index to evaluate such performance.

For each value of ‘k’ we provide the value of the Rand index as:

Where

* *n:* it’s the number of samples in the dataset;
* *a:* it’s the number of pairs of samples that represent the same digit and that are clustered together;
* *b:* it’s the number of pairs of samples that represent different digits and are placed in different clusters.

# Designing the solution

The first portion of code is the same for every classifier, we define the parent class *Classifier*:

* *\_\_init\_\_* method:
  + It loads the MNIST dataset using the *fetch\_openml* function, which fetches datasets from the OpenML repository. The data is then normalized by dividing each feature by 255;
  + An *ATOMClassifier* object is created with the loaded data. This object acts as a wrapper to divide the dataset in training and test set.
* *\_fit\_predict\_* method:
  + The training data is transformed using PCA and then used to fit the model;
  + The test data is also transformed using PCA and then used to make predictions.
* *\_compute\_rand\_index* method:
  + Pairwise equality is calculated for the test labels and the predicted labels;
  + The Rand index is calculated using these pairwise equalities;
  + If the calculated Rand index is greater than the current best Rand index, the best parameters are updated with the current number of clusters, Rand index, and fitting and prediction times.

## Gaussian Mixture

### Formalization

Gaussian Mixture Clustering is a method of clustering that assumes all the data points are generated from a mixture of a finite number of Gaussian distributions with unknown parameters. Here’s a detailed explanation:

* **Gaussian Mixture Model**: Suppose there are k clusters, and the mean and variance are estimated for each k. But since there are k such clusters and the probability density is defined as a linear function of densities of all these k distributions, the parameters cannot be estimated in closed form;
* **Expectation-Maximization algorithm**: The EM algorithm is an iterative way to find maximum-likelihood estimates for model parameters when the data is incomplete or has some missing data points or has some hidden variables. The algorithm chooses some random values for the missing data points and estimates a new set of data. These new values are then recursively used to estimate a better first date, by filling up missing points, until the values get fixed;
* **Estimation Step and Maximization Step**: These are the two most important steps that are iteratively performed to update the model parameters until the model convergence;
* **Clustering**: Once the parameters of the Gaussian distributions are learned, we can then cluster the samples and our error will be nearly as accurate as the Bayes optimal classifier.

### Code explanation

The class *Mixture* is a subclass of *Classifier* and implements a new method:

* *train* method:
  + it iterates over different numbers of PCA components;
  + for each PCA execution it iterates over different number of clusters;
  + for each ‘k’ clusters it calls the *\_fit\_predict* method of *Classifier* with a Gaussian Mixture with diagonal covariance and ‘k’ clusters as the model parameter.

### Conclusion

Algorithm task: multiclass classification.

Parallel processing with 12 cores.

Parallelization backend: loky

Dataset stats ==================== >>

Shape: (70000, 785)

Train set size: 60000

Test set size: 10000

-------------------------------------

Memory: 439.60 MB

Scaled: False

Outlier values: 415414 (0.9%)

| **Components** | **Clusters** | **Fit Time** | **Predict Time** | **Rand Index** |
| --- | --- | --- | --- | --- |
| 2 | 15 | 2.1631 | 0.0450 | 0.8701 |
| 12 | 14 | 3.0613 | 0.0394 | 0.8946 |
| 22 | 15 | 7.2375 | 0.0337 | 0.8960 |
| 32 | 15 | 4.0653 | 0.0411 | 0.8933 |
| 42 | 14 | 5.0723 | 0.0290 | 0.8936 |
| 52 | 12 | 4.7651 | 0.0312 | 0.8860 |
| 62 | 14 | 8.6649 | 0.0442 | 0.8749 |
| 72 | 15 | 10.4653 | 0.0294 | 0.8556 |
| 82 | 14 | 9.8508 | 0.0445 | 0.8647 |
| 92 | 15 | 7.3651 | 0.0383 | 0.8531 |
| 102 | 13 | 6.4071 | 0.0352 | 0.8504 |
| 112 | 15 | 14.0154 | 0.0388 | 0.8504 |
| 122 | 13 | 11.8794 | 0.0399 | 0.8526 |
| 132 | 13 | 9.6332 | 0.0393 | 0.8481 |
| 142 | 14 | 10.0557 | 0.0461 | 0.8577 |
| 152 | 11 | 10.6206 | 0.0374 | 0.8452 |
| 162 | 11 | 8.2323 | 0.0377 | 0.8385 |
| 172 | 14 | 16.3227 | 0.0419 | 0.8425 |
| 182 | 13 | 11.3045 | 0.1135 | 0.8491 |
| 192 | 14 | 12.7288 | 0.0477 | 0.8233 |
| 202 | 15 | 12.6883 | 0.0434 | 0.8334 |

From the table we notice that greater values of the rand index are obtained by higher number of clusters with lower value of PCA components.

## Normalized Cut

### Formalization

Normalized Cut clustering is a graph-based approach for unsupervised data analysis. Here’s a detailed explanation:

* **Graph Representation**: In this method, data points are represented as nodes of a graph. The edges between nodes represent the similarity between data points;
* **Graph Partitioning**: The goal is to partition the graph into disjoint clusters such that the similarity within a group is high and the similarity across groups is low;
* **Normalized Cut**: The normalized cut criterion measures both the total dissimilarity between the different groups as well as the total similarity within the groups. It is defined as the total weight of the edges that need to be removed to achieve the partition, normalized by the sum of the weights of the edges touching each group;
* **Optimization Problem**: The problem of finding the partition that minimizes the normalized cut is a difficult optimization problem. It is often solved within an eigenvector based framework. However, in some cases, the calculation of eigenvectors can be computationally expensive, especially when dealing with high dimensional data;
* **Relaxation and Approximation**: To make the problem tractable, it is often relaxed to a continuous domain, where it can be solved using techniques from linear algebra. The continuous solution is then discretized to obtain the final clustering5.

### Code explanation

The class *Cut* is a subclass of *Classifier* and implements two new method:

* *\_predict* method:
  + Compute the similarity between the test and training data using the Radial basis function (RBF) kernel.
  + For each test sample, find the indices of the k most similar training samples.
  + Get the labels of these k nearest neighbors.
  + For each test sample, predict the label as the most common label among its k nearest neighbors.
* *train* method:
  + it iterates over different numbers of PCA components;
  + for each PCA execution it iterates over different number of clusters;
  + for each ‘k’ clusters it fits the model to the PCA transformed training data and it calls the *\_predict* method.

### Conclusion

Algorithm task: multiclass classification.

Parallel processing with 12 cores.

Parallelization backend: loky

Dataset stats ==================== >>

Shape: (10000, 785)

Train set size: 8000

Test set size: 2000

-------------------------------------

Memory: 62.80 MB

Scaled: False

Outlier values: 56114 (0.9%)

| **Components** | **Clusters** | **Fit Time** | **Predict Time** | **Rand Index** |
| --- | --- | --- | --- | --- |
| 2 | 11 | 1.3914 | 1.7524 | 0.3879 |
| 12 | 12 | 5.6159 | 1.8726 | 0.6344 |
| 22 | 15 | 4.8037 | 1.9169 | 0.6897 |
| 32 | 15 | 5.3216 | 1.8027 | 0.6118 |
| 42 | 15 | 5.8770 | 1.7356 | 0.5628 |
| 52 | 15 | 5.2564 | 1.8247 | 0.5672 |
| 62 | 15 | 6.5250 | 1.8239 | 0.5705 |
| 72 | 15 | 5.6794 | 1.7589 | 0.5602 |
| 82 | 15 | 6.0430 | 1.7914 | 0.5707 |
| 92 | 15 | 5.4131 | 1.7634 | 0.5798 |
| 102 | 12 | 6.3463 | 1.7732 | 0.4843 |
| 112 | 12 | 5.0850 | 1.7569 | 0.4741 |
| 122 | 15 | 6.7058 | 1.7635 | 0.4722 |
| 132 | 11 | 5.8398 | 1.7642 | 0.4900 |
| 142 | 10 | 5.6116 | 1.8445 | 0.4949 |
| 152 | 10 | 5.7662 | 1.7304 | 0.4956 |
| 162 | 10 | 6.2055 | 1.8561 | 0.4982 |
| 172 | 15 | 7.1337 | 1.8465 | 0.5199 |
| 182 | 15 | 6.1475 | 1.8072 | 0.5081 |
| 192 | 15 | 7.3519 | 1.8579 | 0.5776 |
| 202 | 15 | 6.8346 | 1.8458 | 0.5743 |

From the table we notice that the values of the rand index are more or less the same for every value of PCA components, but higher number of clusters.

## Mean Shift

### Formalization

Mean Shift clustering is a centroid based algorithm that aims to discover “blobs” in a smooth density of samples. Here’s a detailed explanation:

* **Initialization**: Each data point is initialized as a cluster centroid;
* **Shifting**: The algorithm works by updating candidates for centroids to be the mean of the points within a given region, also called bandwidth. Each of the sliding windows is shifted towards higher density regions by shifting their centroid to the data-points’ mean within the sliding window. This step will be repeated until no shift yields a higher density;
* **Selection of Sliding Windows**: When multiple sliding windows overlap, the window containing the most points is preserved, and the others are deleted;
* **Assigning Data Points**: The data points are assigned to the sliding window in which they reside;
* **Kernel Density Estimation**: Mean Shift is shifting the windows to a higher density region by shifting their centroid to the mean of the data-points inside the sliding window. Higher density regions correspond to regions with more samples, and lower density regions correspond to regions with fewer points;
* **Choosing the Right Bandwidth**: Depending on the bandwidth, the resulting clusters can look quite different. Therefore instead of selecting the bandwidth by hand, we can estimate it using our data.

Mean Shift is a simple cluster method that works very well on spherical-shaped data. Furthermore, it automatically selects the number of clusters contrary to other clustering algorithms. Also, the output of Mean Shift is not dependent on the initialization since, at the start, each point is a cluster. The downside to Mean Shift is that it is computationally expensive O(n²). It can be sensitive to the choice of kernel and the radius of the kernel.

### Code Explanation

The class *Shift* is a subclass of *Classifier* and implements a new method:

* *train* method:
  + it iterates over different numbers of PCA components;
  + for each PCA execution it iterates over different number of kernel bandwidth;
  + for each ‘k’ kernel bandwidth it calls the *\_fit\_predict* method of *Classifier* with a Mean Shift with ‘k’ bandwidth as the model parameter.

### Conclusion

Algorithm task: multiclass classification.

Parallel processing with 12 cores.

Parallelization backend: loky

Dataset stats ==================== >>

Shape: (70000, 785)

Train set size: 60000

Test set size: 10000

-------------------------------------

Memory: 439.60 MB

Scaled: False

Outlier values: 415414 (0.9%)

| **Components** | **Clusters** | **Fit Time** | **Predict Time** | **Rand Index** |
| --- | --- | --- | --- | --- |
| 2 | 5 | 3.3333 | 0.3640 | 0.1002 |
| 12 | 5 | 9.0991 | 0.0247 | 0.1002 |
| 22 | 5 | 10.0237 | 0.0290 | 0.4375 |
| 32 | 5 | 8.1232 | 0.0339 | 0.4706 |
| 42 | 5 | 6.2824 | 0.0345 | 0.6504 |
| 52 | 5 | 7.4788 | 0.0374 | 0.6495 |
| 62 | 5 | 7.8912 | 0.0499 | 0.6514 |
| 72 | 5 | 8.6858 | 0.0419 | 0.7451 |
| 82 | 5 | 8.0007 | 0.0506 | 0.7782 |
| 92 | 5 | 9.2022 | 0.0427 | 0.8196 |
| 102 | 5 | 9.6179 | 0.0412 | 0.7981 |
| 112 | 5 | 10.7555 | 0.0494 | 0.8325 |
| 122 | 5 | 10.9896 | 0.0526 | 0.8305 |
| 132 | 5 | 11.9510 | 0.0428 | 0.8311 |
| 142 | 5 | 13.2086 | 0.0572 | 0.8306 |
| 152 | 5 | 14.7382 | 0.0393 | 0.8321 |
| 162 | 5 | 13.3471 | 0.0346 | 0.8317 |
| 172 | 5 | 16.1631 | 0.0478 | 0.8312 |
| 182 | 5 | 16.0770 | 0.0385 | 0.8304 |
| 192 | 5 | 15.5176 | 0.0385 | 0.8315 |
| 202 | 5 | 18.0257 | 0.0363 | 0.8465 |

From the table we notice that greater values of the rand index are obtained by lower number of clusters with higher value of PCA components.

# Comparison