PROJECT 3 MULTI-CLASS CLASSIFICATION USING UNSUPERVISED LEARNING

Prakhathi Murugesan
University of Buffalo
prakhath@buffalo.edu

Abstract

Arthur Samuel, who coined the term 'Machine Learning' defines Machine Learning as, the field of study which provides computer the ability to learn without being programmed. The Computer can be able to perform a specific task by relying on patterns and interference instead of instructions. This involves developing a model which takes training data and makes predictions. Some of the application of machine learning are spam detection, search engines, photo tagging, medical diagnosis, etc. Machine Learning is classified into broad categories – Supervised Learning, Unsupervised Learning, Semi-supervised Learning, Reinforcement Learning. Unsupervised Learning is a paradigm which involves finding an unknown pattern in data set without pre-existing labels. The model developed using unsupervised learning, groups the data using similarities and patterns without any prior training of data. Clustering and Association are the two types of Unsupervised learning methods. The goal of this project to perform cluster analysis on fashion MNIST dataset using unsupervised learning. This report records the process of building various model and evaluates its performance with different parameters.

1.INTRODUCTION

When the training data consists of a set of feature without any corresponding target values, unsupervised learning is used. Unsupervised Learning may discover groups of similar data which is discovering hidden patterns in data or adapts feature learning. It is harder as compared to supervised learning as it predicts the output the prior knowledge of the outcome. But still used in places where annotating large dataset is very costly. One of the popular unsupervised learning methods is cluster analysis. Cluster analysis or clustering is the task of grouping a set of data in such a way that data in the same group/cluster are more similar to each other than to those in other group/cluster. Some applications where clustering involves are pattern recognition, image analysis, bioinformatics, data compression, etc. This project involves clustering the dataset using KMeans algorithm and building an auto-encoder based on K-Means clustering and Gaussian mixture model. The clustering accuracy is obtained for each model and its performance is evaluated.

2. DATASETS

For training and testing of our clustering models, we will use the Fashion-MNIST dataset. The Fashion-MNIST is a dataset of Zalando's article images, consisting of a training set of 60,000 examples and a test set of 10,000 examples. Each example is a 28x28 grayscale image. Each image is 28 pixels in height and 28 pixels in width, for a total of 784 pixels in total. Each pixel has a single pixel-value associated with it, indicating the lightness or darkness of that pixel, with higher numbers meaning darker. This pixel-value is an integer between 0 and 255. The training and test data sets have 784 columns. The Fashion MNIST

dataset is imported using keras library. Each training and test data are assigned to one of the labels ranges from 1 – 10. The labels are used only for evaluation but not for training.

3. ARCHITECTURE

The Initial step of the project is Pre-processing of the dataset. As a part of this step, the fashion mnist dataset is loaded using keras library. The next step of processing is normalization of the dataset which makes pixel values to be in the range of 0 to 1. Then the training and test dataset are reshaped according to the architecture and datatype conversion is done. The dataset should be in float32 format. Then the training dataset is partitioned into training and validation dataset. Once the pre-processing of dataset is done, the cluster analysis is done on the training dataset using clustering methods from Sklearns library.

3.1 K-Means:

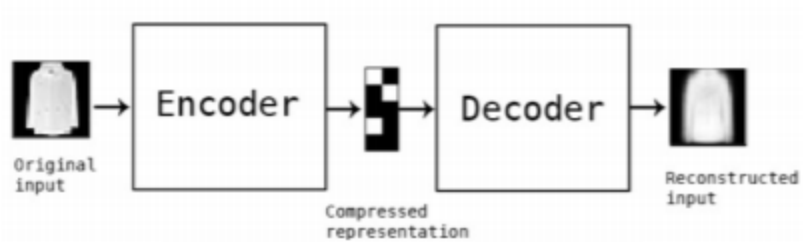
K-means clustering is one of the popular unsupervised machine learning algorithms. The *k*-means algorithm searches for a pre-determined number of clusters within an unlabeled dataset. It uses a simple optimal clustering in which, the "cluster center" is the arithmetic mean of all the points belonging to the cluster and each point is closer to its own cluster center than to other cluster center. K-means algorithm starts with group of randomly selected k number of centroids – center of the cluster. Then it assigns every data point to the nearest cluster and then it calculates the mean of the cluster and update the centroid. The process is repeated until there is no change in the value of the centroids or when the pre-defined number of iterations has been done. Clustering on dataset using K-Means can be easily with Sklearns library.

```
class sklearn.cluster. KMeans (n_clusters=8, init='k-means++', n_init=10, max_iter=300, tol=0.0001,
precompute_distances='auto', verbose=0, random_state=None, copy_x=True, n_jobs=None, algorithm='auto')
```

The above snippet is used to build a model and then we train it with training dataset using **fit** and predict the output using **predict** in-built function**.** The number of clusters is selected by using elbow criterion method. The K-Means clustering on dataset is iterated for a range of values of k (number of clusters) and for each value of k, sum of squared error (SSE) is calculated. SSE is calculated as sum of squared distance between each datapoint of the cluster and its centroid. When SSE is plotted against K, the graph looks like an arm ie. SSE tends to decrease as k increases. The optimal number of cluster is the point where SSE decreases abruptly. In Sklearns, Kmeans method returns inertia. Inertia can be recognized as a measure of how internally coherent clusters are, so it equals to SSE.

3.2 Auto Encoder:

Autoencoding is a data compression algorithm where the compression and decompression functions are data-specific, lossy, and learned automatically from examples rather than engineered by a human. Autoencoder uses compression and decompression functions which are implemented with neural networks. Autoencoder consists of two parts – Encoder and Decoder. Encoder compresses the input into a latent-space representation and decoder reconstructs the output from its representation.



Architecture of an Autoencoder

Few of the application of autoencoders are data denoising and dimensionality reduction for data visualization. Auto encoder extracts only the required features of an image and generates the output by removing any noise or unnecessary interruption. The reconstructed image is the same as our input but with reduced dimensions. It helps in providing the similar image with a reduced pixel value. It forces the model to learn from salient feature of the input.

Architecture used here for auto-encoder:

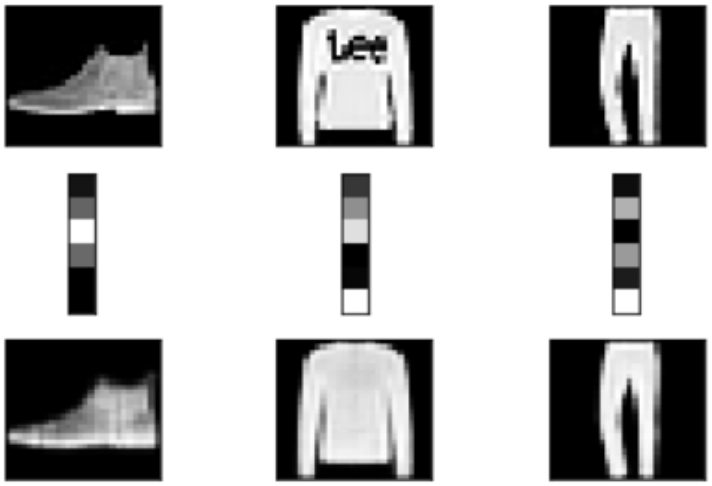
Input_layer → Dense(128, relu) → Dense(64, relu) → Dense(64, relu) → Dense(32, Sigmoid) → Dense(64, relu) → Dense(64, relu) → Dense(128, relu) → Output_Layer(Dense(784))

3.2.1 Auto-Encoder with K-Means Clustering:

K-Means algorithm uses Expectation-Maximization (EM) which consists of two steps:

i. Take random cluster centers.
ii. E-step: Assign datapoints to the nearest cluster center.
iii. M-step: Update the cluster center to the mean.
iv. Repeat step (ii) & (iii) until converged.

The K-means algorithm aims to choose centroids that minimize the inertia, or within-cluster sum-of squares criterion. The workflow of the autoencoder is that the auto-encoder is built using deep layer neural network and it is then trained with training data. The encoded image is visualized as below:



Then encoder is built using the existing auto-encoder and the encoded output is used to do cluster analysis using K-Means and Gaussian Mixture Clustering model.

3.2.2 Auto-Encoder with GMM Clustering:

The observation obtained from k-means is that the cluster models must be circular and when the datapoints are not distributed circular but linear or elliptical, circular cluster would be a poor fit and K-Means force-fit the data into circular cluster and results in mixing cluster assignment and clusters overlap. K-Means doesn't account for covariance (variance) which is in two dimensions, determines the shape of the distribution. K-Means doesn't give the probabilities that a data point belongs to each cluster which is probabilistic cluster assignment. A Gaussian mixture model is a probabilistic model that assumes all the data points are generated from a mixture of a finite number of Gaussian distributions with unknown parameters. The Gaussian Mixture model uses the expectation-maximization (EM) algorithm for fitting mixture-of-Gaussian models.

i. Choose starting guesses for the location and shape which are gaussian parameters.
ii. E-step: For each datapoint, find weights encoding the probability of membership in each cluster
iii. M-step: For each cluster, update its gaussian parameters - location, normalization and shape, based on all data points, making use of the weights.
iv. Repeat step (ii) & (iii) until converged.

The below snippet is used to perform gaussian mixture clustering using Sklearns.

class `sklearn.mixture.` **`GaussianMixture`** (*n_components=1, covariance_type='full', tol=0.001, reg_covar=1e-06, max_iter=100, n_init=1, init_params='kmeans', weights_init=None, means_init=None, precisions_init=None, random_state=None, warm_start=False, verbose=0, verbose_interval=10*) [source]

The number of components can be selected using Bayesian Information Criterion (BIC) which evaluate the likelihood. The optimal number of clusters is the value that minimizes BIC. GaussianMixture.fit method is provided that train a Gaussian Mixture Model with training data. GaussianMixture.predict method is used to predict the labels for each sample in given test data.

4. RESULT

Accuracy is often used to measure the quality of the classification. In unsupervised learning, the target labels are unknown, so accuracy of the clustering algorithm is not straight forward. Evaluating the performance of a clustering algorithm is done by various evaluation metrics. Algorithm labels each cluster arbitrary which may not match the ground truth label. Hence the target labels are provided, the confusion matrix can be obtained for the predicted output and target labels. But from the confusion matrix we can observe that the predicted output doesn't match with the true labels. There is no association provided by clustering between the class labels and the predicted cluster labels. For clustering, we have to find the best match between the class labels and the cluster labels, so accuracy is defined by:

$$accuracy(y, \hat{y}) = \max_{perm \in P} \frac{1}{n} \sum_{i=0}^{n-1} 1(perm(\hat{y}_i) = y_i)$$

Accuracy is also computed from confusion matrix and it is equal to the sum of the diagonal elements of the confusion matrix, divided by the number of data to get a value between 0 and 1. For clustering, the rows of the confusion matrix should be permuted to obtain the maximum value for the sum. From sklearns, linear_assignment is used to apply Hungarian algorithm and the reordered confusion matrix is obtained and then the accuracy is computed. Other Evaluation measure used here is normalized_mutual_info_score(NMI). NMI values close to one denotes high similarity between clusters and labels and closer to zero denotes dissimilarity.

I. Accuracy and confusion matrix obtained for K-Means clustering:

```
FOR TESTING DATASET:
Clustering Score with K-Means using NMI: 0.5136855183656902
Confusion Matrix based on K-Means clustering:
[[ 94 245   5   0 587   0   5  34   1  29]
 [ 22  29   0   0  50   0   0   9   0 890]
 [ 61 342   4   0  19   0   4 566   0   4]
 [ 95 110   3   0 277   0   2  10   0 503]
 [ 42 159   5   0 135   0   4 628   0  27]
 [650   6   0  69   0  44   0   0 231   0]
 [116 358   0   0 189   0  15 310   0  12]
 [ 62   0   0 142   0   2   0   0 794   0]
 [ 82  34 409   7   3   1 355  59  43   7]
 [ 29   4   2 533   0 406   0   0  26   0]]
The reordered confusion matrix
[[587  29   5   0  34  94 245   1   5   0]
 [ 50 890   0   0   9  22  29   0   0   0]
 [ 19   4   4   0 566  61 342   0   4   0]
 [277 503   2   0  10  95 110   0   3   0]
 [135  27   4   0 628  42 159   0   5   0]
 [  0   0   0  44   0 650   6 231   0  69]
 [189  12  15   0 310 116 358   0   0   0]
 [  0   0   0   2   0  62   0 794   0 142]
 [  3   7 355   1  59  82  34  43 409   7]
 [  0   0   0 406   0  29   4  26   2 533]]
Accuracy based on K-Means clustering: 48.53 %
```

II. Accuracy and confusion matrix obtained for Auto-Encoder based K-Means clustering:

```
K-Means Clutering Model:
Clustering Score using NMI: 0.5855856221826633
Confusion Matrix:
[[  2   0   3 319  93   0   2  66   0   0]
 [  0 379   2   1  10   0   9  70   0   0]
 [  3   0  35   2 236   0 231   5   0   0]
 [  0   9   0  13  48   0  13 435   0   0]
 [  0   0  13   0  74   0 343  56   0   0]
 [  1   0   1   0  10 266   0   0  73 145]
 [  8   1  22 103 206   0 153  35   0   0]
 [  0   0   0   0   0 414   0   0  90   6]
 [207   0 232   0  33   2   2  21   0   3]
 [  0   0   0   0   0  14   0   0 278 202]]
The reordered confusion matrix
[[319   0  93  66   2   0   2   0   3   0]
 [  1 379  10  70   9   0   0   0   2   0]
 [  2   0 236   5 231   0   3   0  35   0]
 [ 13   9  48 435  13   0   0   0   0   0]
 [  0   0  74  56 343   0   0   0  13   0]
 [  0   0  10   0   0 145   1 266   1  73]
 [103   1 206  35 153   0   8   0  22   0]
 [  0   0   0   0   0   6   0 414   0  90]
 [  0   0  33  21   2   3 207   2 232   0]
 [  0   0   0   0   0 202   0  14   0 278]]
Accuracy for the clustering model using auto-encoder: 55.779999999999994 %
```

III. Accuracy and confusion matrix obtained for Auto-Encoder based Gaussian Mixture clustering:

```
Gaussian Mixture Clutering Model:
Clustering Score using NMI: 0.6470197538610604
Accuracy for the clustering model using auto-enncoder: 61.71 %
Confusion Matrix:
[[  0   0 139  50   0  29  19   0   0 763]
 [  0   0 109   9   0   9   2 867   0   4]
 [  0   0  17 584   0 388   5   0   0   6]
 [  0   0 898  22   0  41   6   6   0  27]
 [  0   0 116 700   0 181   3   0   0   0]
 [ 28 453   0   0 464   0   4   0  51   0]
 [  0   0 128 324   0 322  23   0   0 203]
 [  6  10   0   0 637   0   0   0 347   0]
 [  2   1  14  50   5   7 920   1   0   0]
 [545  33   0   0   3   0   1   0 418   0]]
```

```
The reordered confusion matrix
[[763   0  29 139  50   0   0   0  19   0]
 [  4 867   9 109   9   0   0   0   2   0]
 [  6   0 388  17 584   0   0   0   5   0]
 [ 27   6  41 898  22   0   0   0   6   0]
 [  0   0 181 116 700   0   0   0   3   0]
 [  0   0   0   0   0 453  51 464   4  28]
 [203   0 322 128 324   0   0   0  23   0]
 [  0   0   0   0   0  10 347 637   0   6]
 [  0   1   7  14  50   1   0   5 920   2]
 [  0   0   0   0   0  33 418   3   1 545]]
```

5. CONCLUSION

This project helps get a clear idea of the working of unsupervised learning algorithms. It has also helped in understanding the k-Means clustering, auto-encoder and Gaussian mixture model. From the experiment, it is observed that the best result is obtained from an auto-encoder based on Gaussian Mixture Model. This project helped get a deeper considering of the impact of the auto-encoder in unsupervised learning. Thus,

cluster analysis is performed and evaluated on fashion MNIST dataset using unsupervised learning and best results are obtained.

References:

[1] Auto-encoder - https://towardsdatascience.com/deep-inside-autoencoders-7e41f319999f

[2] K-Means - https://scikit-learn.org/stable/modules/clustering.html#k-means

[3] Accuracy Evaluation for Clustering- https://smorbieu.gitlab.io/accuracy-from-classification-to-clustering-evaluation/

[4] Gaussian Mixture Model - https://jakevdp.github.io/PythonDataScienceHandbook/05.12-gaussian-mixtures.html