CSE-574: MACHINE LEARNING PROJECT 3 CLUSTER ANALYSIS USING UNSUPERVISED LEARNING

Madhumitha Sivasankaran
Department of Computer Science
University at Buffalo
Buffalo, NY 14214
UB Id:50310290
msivasan@buffalo.edu

Abstract

This project is about performing cluster analysis using unsupervised learning. We are provided with Fashion MNIST dataset. The task here is to cluster the given images in the dataset and identify it as one of many clusters. The three tasks to be performed are:

- Use KMeans algorithm to cluster original data space of Fashion-MNIST dataset using Sklearns library.
- Build an Auto-Encoder based K-Means clustering model to cluster the condensed representation of the unlabeled fashion MNIST dataset using Keras and Sklearns library.
- Build an Auto-Encoder based Gaussian Mixture Model clustering model to cluster the condensed representation of the unlabeled fashion MNIST dataset using Keras and Sklearns library.

The clustering accuracy is recorded for the baseline KMeans Algorithm after clustering the dataset into 10 clusters. Using Keras library an Auto-encoder network is built and trained. We have the encoder learned to compress each image into latent floating point values. We are then going to use KMeans and Gaussian Mixture model to generate the cluster centroids which are the 10 cluster centers in the latent feature space. We generate the confusion matrix and report the clustering accuracy of each of the clustering methods.

Introduction:

Clustering as a method of finding subgroups within observations is used widely in applications like market segmentation wherein we try and find some structure in the data. Although an unsupervised machine learning technique, the clusters can be used as features in a supervised machine learning model. And all the algorithms will partition the data into a group of similar objects by discovering the structure or pattern in the data. And good clustering algorithm should give you minimal similarity across clusters and maximum similarity within the cluster

KMeans is a clustering algorithm which divides observations into k clusters. Since we can dictate the amount of clusters, it can be easily used in classification where we divide data into clusters which can be equal to or more than the number of classes.

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. One can think of mixture models as generalizing k-means clustering to incorporate information about the covariance structure of the data as well as the centers of the latent Gaussians.

An **autoencoder (AE)** is a specific kind of unsupervised artificial neural network that provides compression and other functionality in the field of machine learning. The specific use of the autoencoder is to use a feedforward approach to reconstitute an output from an input. The input is compressed and then sent to be decompressed as output, which is often similar to the original input

Dataset:

For training and testing of our classifiers, 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, associated with a label from 10 classes. 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 785 columns. The first column consists of the class labels, and represents the type of clothing. The rest of the columns contain the pixel-values of the associated image. Each training and test example is assigned to one of the labels as follows:

1. T-shirt/top
2. Trouser
3. Pullover
4. Dress
5. Coat
6. Sandal
7. Shirt
8. Sneaker
9. Bag
10. Ankle Boot

.

Architecture:

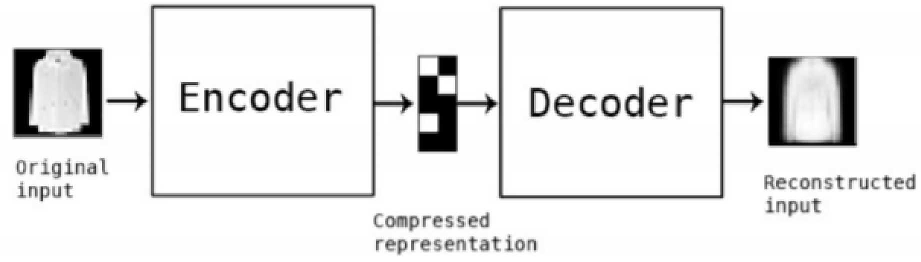
The **KMeans** algorithm clusters data by trying to separate samples in n groups of equal variance, minimizing a criterion known as the *inertia* or within-cluster sum-of-squares (see below). This algorithm requires the number of clusters to be specified. It scales well to large number of samples and has been used across a large range of application areas in many different fields.
The k-means algorithm divides a set of N samples X into K disjoint clusters C, each described by the mean μj of the samples in the cluster. The means are commonly called the cluster "centroids"; note that they are not, in general, points from X, although they live in the same space.
The K-means algorithm aims to choose centroids that minimise the **inertia**, or **within-cluster sum-of-squares criterion**:

$$\sum_{i=0}^{n} \min_{\mu_j \in C}(||x_i - \mu_j||^2)$$

Inertia can be recognized as a measure of how internally coherent clusters are. It suffers from various drawbacks:
Inertia makes the assumption that clusters are convex and isotropic, which is not always the case. It responds poorly to elongated clusters, or manifolds with irregular shapes.
Inertia is not a normalized metric: we just know that lower values are better and zero is optimal. But in very high-dimensional spaces, Euclidean distances tend to become inflated (this is an instance of the so-called "curse of dimensionality"). Running a dimensionality reduction algorithm such as Principal component analysis (PCA) prior to k-means clustering can alleviate this problem and speed up the computations.

Autoencoders (AE) are neural networks that aims to copy their inputs to their outputs. They work by compressing the input into a **latent-space representation**, and then reconstructing the output from this representation. This kind of network is composed of two parts:

1. **Encoder:** This is the part of the network that compresses the input into a latent-space representation. It can be represented by an encoding function *h=f(x)*.
2. **Decoder:** This part aims to reconstruct the input from the latent space representation. It can be represented by a decoding function *r=g(h)*.



The autoencoder as a whole can thus be described by the function *g(f(x)) = r* where you want ***r*** as close as the original input ***x***.

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. One can think of mixture models as generalizing k-means clustering to incorporate information about the covariance structure of the data as well as the centers of the latent Gaussians.

The **GaussianMixture** object implements the expectation-maximization (EM) algorithm for fitting mixture-of-Gaussian models. It can also draw confidence ellipsoids for multivariate models, and compute the Bayesian Information Criterion to assess the number of clusters in the data. A **GaussianMixture.fit** method is provided that learns a Gaussian Mixture Model from train data. Given test data, it can assign to each sample the Gaussian it mostly probably belong to using the **GaussianMixture.predict** method.

Results and Observations:

1. Baseline KMeans Clustering Model

The dataset was clustered into 10 clusters using the KMeans Algorithm. The Clustering Accuracy was reported to be 52.03 %
Inertia: 1936071.8096
Homogeneity: 0.5019

2. Training using the Auto Encoder network

The dataset was split into training and validation set and the plots for the respective Losses vs No: of Epochs are as follows:

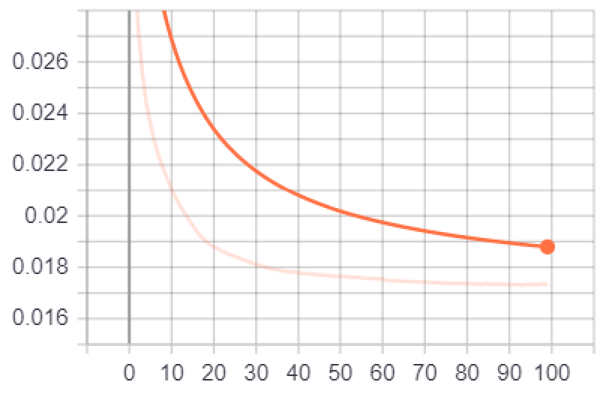


Figure 1: Training Loss vs No: of epochs

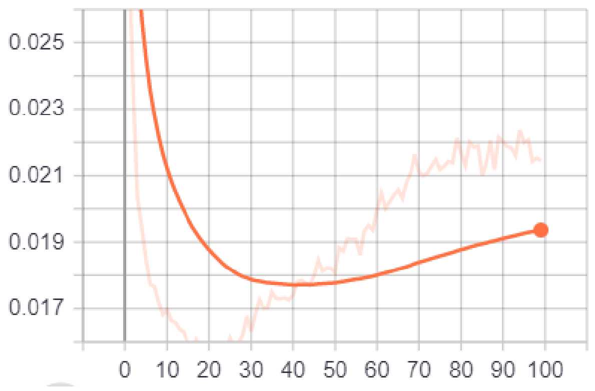


Figure 2: Validation Loss vs No: of epochs

3. Auto-Encoder based K-Means clustering prediction

The confusion matrix for the predicted cluster values is:

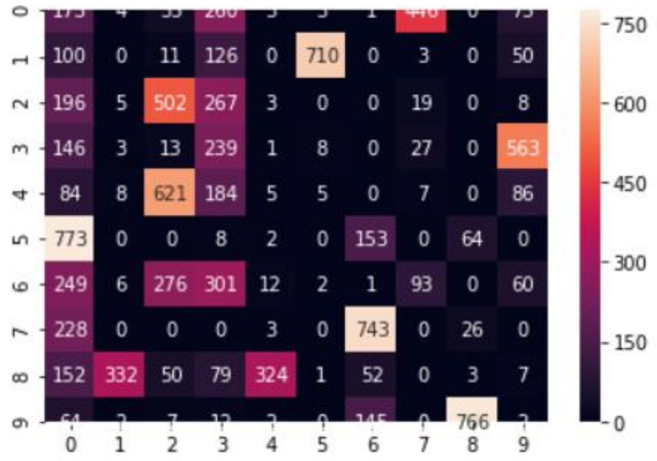


Figure 3: Confusion matrix for Auto encoder based K Means clustering prediction

And the clustering accuracy is reported to be 52.54%

4. Auto-Encoder based Gaussian Mixture Model prediction

The confusion matrix for the predicted cluster values is:

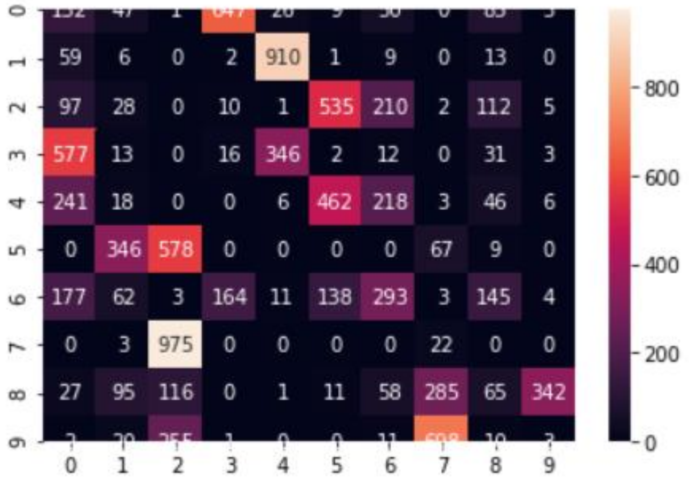


Figure 4: Confusion Matrix for Auto-Encoder based Gaussian Mixture Model

And the clustering accuracy is reported to be 54.68%

Conclusion:

Thus from the above results, it can be concluded that the Auto-Encoder based Gaussian Mixture Model proved to be the most efficient clustering model with the Accuracy of 54.68%.

References:

1) https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html#sklearn.cluster.KMeans.fit
2) https://stackoverflow.com/questions/42112260/how-do-i-use-the-tensorboard-callback-of-keras?noredirect=1
3) https://keras.io/models/sequential/#sequential-model-methods
4) https://scikit-learn.org/stable/modules/mixture.html#mixture
5) https://ramhiser.com/post/2018-05-14-autoencoders-with-keras/
6) https://towardsdatascience.com/kmeans-clustering-for-classification-74b992405d0a