Optical Character Recognition Via Unsupervised Learning (K-means & SOM)

Amir M. Mousavi _3971242026

Department of Computer Engineering
Shahid Rajayi University

Tehran, Iran

AmirMahmood.Mousavi@yahoo.com

***Abstract*— Clustering is a type of unsupervised machine learning which aims to find homogeneous subgroups such that objects in the same group (clusters) are more similar to each other than the others.**

K-Means 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.

in this paper , I'll be using the given dataset which is a collection of labelled handwritten digits and use K-Means and SOM to find clusters within the dataset and discuss the results...

Keywords—Unsupervised Learning , SOM , Self-Organization Map , K-Means , OCR , Optical Character Recognition , Clustering , handwritten digits , Machine Learning .

.

I. INTRODUCTION

K-means is the most popular clustering algorithm, because it is very simple and easy to implement and it has shown good performance on different tasks. It belongs to the class of partition algorithms that simultaneously partition data points into distinct groups called clusters. An alternative group of methods, which we will not cover in this book, are hierarchical clustering algorithms. These find an initial set of clusters and divide or merge them to form new ones.

The main idea behind k-means is to find a partition of data points such that the squared distance between the cluster mean and each point in the cluster is minimized. Note that this method assumes that you know a priori the number of clusters your data should be divided into.

We will show in this section how k-means works using a motivating example, the problem of clustering handwritten digits. So, we will apply unsupervised algorithms to solve the OCR problem , in order to cluster the digits in this paper we are going to apply K-Means and SOM ...

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.

The recognition of handwritten characters by computer has been a topic of intensive search for many years. Handwritten numeral recognition is always the research focus in the field of image process and pattern recognition. The numeral varieties in size, shape, slant and the writing style make the research harder. The numeral character recognition is the most challenging field, because the big research and development effort that has gone into it has not solved all commercial and intellectual problems. Handwritten numeral character recognition is an important step in many document processing applications. Digital document processing is gaining popularity for application to office and library automation, bank and postal services, publishing houses and communication technology. The complexity of the problem is greatly increased by noise in the data and infinite variability of handwriting as a result of mood of writer and nature of writing. Recognition of handwritten digits has been popular topic of research for many years. The recognition of handwritten numeral character has been considerable interest to researchers working on OCR

In this paper we have proposed to provide a comprehensive tutorial and survey about review of the unsupervised learning and its algorithm including K-Means and SOM , etc. which is discussed in sections II to VI Then a brief explanation of OCR problem and feature extraction methods are in sections VII and V . Finally. The results of the MATLAB implementation are presented VIII and the conclusions are discussed in section IX .

II. UNSUPERVISED LEARNING

Unsupervised learning studies how systems can learn to represent particular input patterns in a way that reflects the statistical structure of the overall collection of input patterns. By contrast with SUPERVISED LEARNING or REINFORCEMENT LEARNING, there are no explicit target outputs or environmental evaluations associated with each input; rather the unsupervised learner brings to bear prior biases as to what aspects of the structure of the input should be captured in the output.

Unsupervised learning is important since it is likely to be much more common in the brain than supervised learning. For instance there are around 10 ^{6} photoreceptors in each eye whose activities are constantly changing with the visual world and which provide all the information that is available to indicate what objects there are in the world, how they are presented, what the lighting conditions are, etc. Developmental and adult plasticity are critical in animal vision (see VISION AND LEARNING) – indeed structural and physiological properties of synapses in the neocortex are known to be substantially influenced by the patterns of activity in sensory neurons that occur. However, essentially none of the information about the contents of scenes is available during learning. This makes unsupervised methods essential, and, equally, allows them to be used as computational models for synaptic adaptation.

The only things that unsupervised learning methods have to work with are the observed input patterns xi, which are often assumed to be independent samples from an underlying unknown probability distribution PI [x], and some explicit or implicit a priori information as to what is important. One key notion is that input, such as the image of a scene, has distal independent causes, such as objects at given locations illuminated by particular lighting. Since it is on those independent causes that we normally must act, the best representation for an input is in their terms. Two classes of method have been suggested for unsupervised learning. Density estimation techniques explicitly build statistical models (such as BAYESIAN NETWORKS) of how underlying causes could create the input. Feature extraction techniques try to extract statistical regularities (or sometimes .irregularities) directly from the inputs

a) Short Review of the The Unsupervised Learning :

Unsupervised learning in general has a long and distinguished history. Some early influences were Horace Barlow (see Barlow, 1992), who sought ways of characterising neural codes, Donald MacKay (1956), who adopted a cybernetic-theoretic approach, and David Marr (1970), who made an early unsupervised learning postulate about the goal of learning in his model of the neocortex. The Hebb rule (Hebb, 1949), which links statistical methods to neurophysiological experiments on plasticity, has also cast a long shadow. Geoffrey Hinton and Terrence Sejnowski in inventing a model of learning called the Boltzmann machine (1986), imported many of the concepts from statistics that now dominate the density estimation methods (Grenander, 1976-1981). Feature extraction methods have generally been less extensively explored.

b) Clustering :

Clustering (or cluster analysis) aims to organize a collection of data items into clusters, such that items within a cluster are more “similar” to each other than they are to items in the other clusters. This notion of similarity can be expressed in very different ways, according to the purpose of the study, to domain-specific assumptions and to prior knowledge of the problem.
Clustering provides a convenient example. Consider the case in which the inputs are the photoreceptor activities created by various images of an apple or an orange. In the space of all possible activities, these particular inputs form two clusters, with many fewer degrees of variation than 106 , ie lower dimension. One natural task for unsupervised learning is to find and characterise these separate, low dimensional clusters.

Clustering is usually performed when no information is available concerning the membership of data items to predefined classes. For this reason, clustering is traditionally seen as part of unsupervised learning. We nevertheless speak here of unsupervised clustering to distinguish it from a more recent and less common approach that makes use of a small amount of supervision to “guide” or “adjust” clustering. To support the extensive use of clustering in computer vision, pattern recognition, information retrieval, data mining, etc., very many different methods were developed in several communities.

III. K-MEANS

K-Means Clustering is a machine learning technique for classifying , In k-means clustering, the data set is partitioned into k regions corresponding to the k clusters. The algorithm finds a local minimum of the total squared Euclidean distance from the points to their respective cluster means. That is, the loss function is given by :

$$L_{\text{km}} = \sum_{h=1}^{k} \sum_{n:c_n=h} \|\mathbf{x}_n - \boldsymbol{\mu}_h\|^2 .$$

The partition is a local minimum of this cost function when two criteria are met: first, each datum is assigned to its nearest cluster, which minimizes the total distance for a fixed µ ; and second, each µh is at the centroid (mean) of all the points in its cluster, which minimizes the squared error for fixed cluster assignments. The k-means algorithm is equivalent to the Lloyd algorithm for vector quantization. Given an initial set of means, the algorithm iterates over two steps, each of which optimizes for one of the two criteria described above:

١) Assign each datum to the cluster with the closest centroid :

$$c_n = \arg \min_{1 \le h \le k} \|\mathbf{x}_n - \boldsymbol{\mu}_h\|^2 .$$

٢) Assign each centroid to the mean of the points in its cluster:

$$\boldsymbol{\mu}_h = \frac{\sum_{n:c_n=h} \mathbf{x}_n}{\sum_{n:c_n=h} 1} .$$

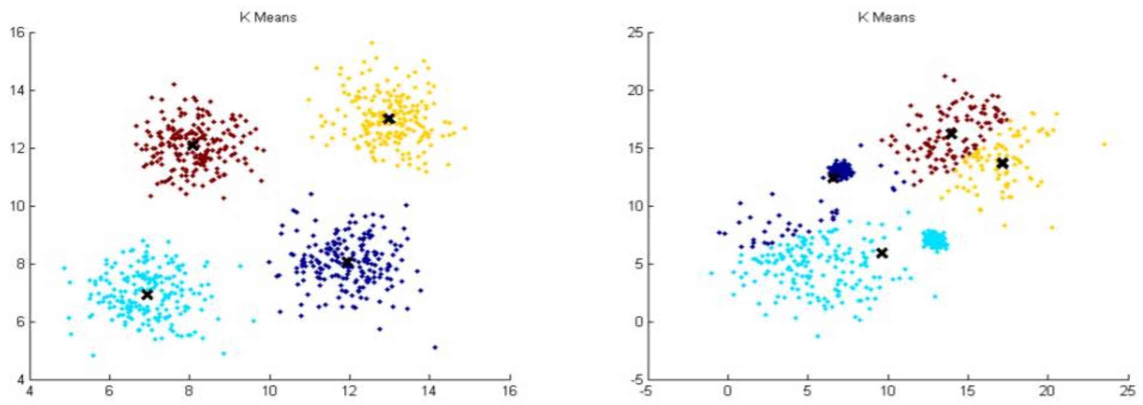


Figure 1: Data sets clustered by the k-means algorithm. The k-means clusters tend to be similar in size.

These steps are repeated a fixed number of times or until the changes in the parameters are smaller than some threshold.

Because each point is assigned to the nearest centroid, the clusters produced by k-means are Vornoi cells with piecewise linear boundaries. The cells tend to be similar in size, but may contain very different numbers of elements. The algorithm uses Euclidean distances, so each dimension is given equal weight. If the elements of the data vectors have different scales, then the data should first be normalized so that Euclidean distance accurately quantifies the discrepancy between points.

Figure 1 shows the k-means clusters for two data sets, each generated from four bivariate Gaussian distributions. A reasonable human observer would classify the points in each set into four clusters. The data set on the left contains separable clusters of similar size, so they are easily categorized. The data set on the right has clusters of dierent sizes that cannot be separated by Voronoi cells, so the k-means clusters do not match the classes used to generate the points. Nevertheless, the clusters do achieve the k-means optimality criteria by minimizing the total squared distance.

to explain it with a simple example. Below is some (fictitious) data comparing elephants and penguins. We've plotted 20 animals, and each one is represented by a (weight, height) coordinate.

to explain it with a simple example. Below is some (fictitious) data comparing elephants and penguins. We've plotted 20 animals, and each one is represented by a (weight, height) coordinate.
You can see that the coordinate points of the elephants and penguins form two clusters in figure 2 : elephants are bigger and heavier, penguins are smaller and lighter. Now suppose we've got one more data point, but we've forgotten whether it's an elephant or a penguin. Let's plot it, too. We've marked it in orange .

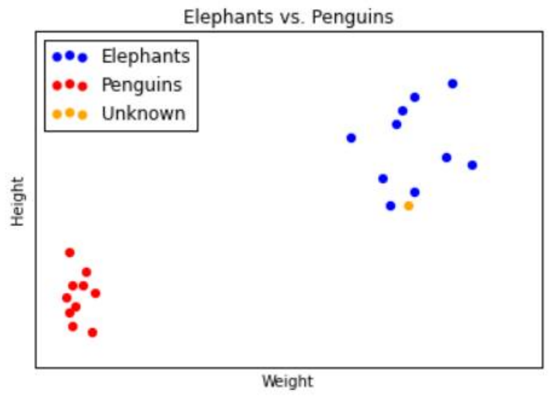


Figure 2 . coordinate points of the elephants and penguins

If you were to make a guess, you'd say that the orange datapoint probably belongs to an elephant, and not to a penguin. We say this because the orange datapoint seems to belong to the elephant cluster, not to the penguin cluster.

This is the essence of clustering. We take some labelled data — like heights and weights of animals, where each animal is labeled as either a penguin or an elephant. We use an algorithm to figure out which datapoints belong to which (weight, height) clusters. We look at the labels of the clusters to understand what label each cluster corresponds to. Then we take an unlabelled datapoint, see into which cluster it fits best, and thereby assign the unlabelled datapoint a label.

We'll use k-means clustering on the given dataset of handwritten digits, which consists of 20,000 handwritten digits (0-9) that have been scanned in and scaled to 25 ×25 pixels. For every digit, each pixel can be represented as an integer in the range [0,255] where 0 corresponds to the pixel being completely white, and 255 corresponds to the pixel being completely black. This gives us a 25 ×25 matrix of integers. We can then flatten this matrix into a 625 ×1 vector, which is like a coordinate pair, except for that instead of 2 coordinates it has 625. Now the data is in coordinate form, then we can run k-means clustering.

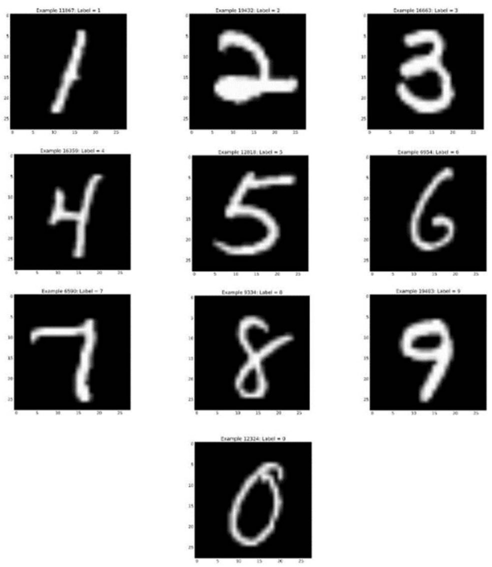


Figure 2. digits are as a 25*25 grayscale matrix

IV. SOM (Self Organizing Maps)

The SOM was proposed in 1984 by Teuvo Kohonen, a Finnish academician. It is based in the process of task clustering that occurs in our brain; it is a kind of neural network used for the visualization of high-dimensional data. It compresses the information of high-dimensional data into geometric relationships onto a low-dimensional representation. In a SOM algorithm, the neurons are ordered in two layers: the input layer and the competition layer. The input layer is composed of N neurons, one for each input variable. The competition layer is composed of a topological low-dimensional grid of neurons (usually 2-dimensional) geometrically ordered. The number of neurons in the output layer depends on the problem. It may fluctuate from a few to several thousands depending on the complexity of the problem.

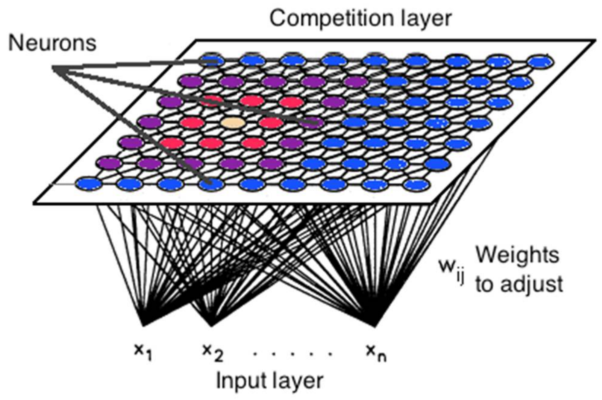


Figure 3 . Self-Organizing Map scheme

Each input layer unit is connected with every neuron of the competition layer and next, an N-dimensional weight vector is assigned to each competition layer unit.

Thereby, a model of some observations is associated with each second layer unit. The algorithm calculates the models that best describe the domain of observations. The models are arranged in the two-dimensional grid so that similar models are closer each other than the different ones. The SOM keeps a neighborhood relation between the original distribution of N-dimensional data and the topological low-dimensional grid.

Concerning to the nuts and bolts of the algorithm, a couple of choices have to be made: the map type (hexagonal or rectangular grid, which indicates the neighborhood relation or topology), and the number of neurons (which defines the size of the low-dimensional grid). These choices depend on the size of the input data and their dispersion.

Afterwards, a learning algorithm is used in order to calculate the associated weights for each competition layer neuron. The first step in this algorithm is the weight initialization (at random, usually). The following step is to approach them to the optimum values using an iterative procedure. In each training step, a sampled input observation x is randomly chosen, and the distance within its N-dimensional space position and the weights vectors associated with each competition layer unit, is calculated using some distance measurement (usually the Euclidean distance). The neuron whose weight vector is nearest to the input observation x is called winning neuron or Best-Matching Unit (BMU hereafter), denoted by c :

$$\|\mathbf{x} - \mathbf{m_c}\| = min_i d(\mathbf{x} - \mathbf{m_i}),$$

where d() is the distance measure function.

Right after, a weight update is performed so that the winning neuron is approached to the input observation. The BMU’s topological neighbors are trated likewise. This adaptation method pulls the BMU and its topological neighbors towards the sampled observation (look at the figure 4).

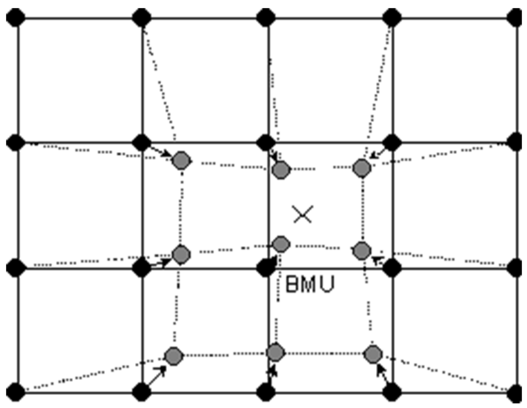


Figure.4. Updating the best matching unit (BMU) and its neighbours towards the input sample marked with x. The solid and dashed lines correspond to the situation before and after updating, respectively.

where x(t) is the sampled observation randomly drawn from the input data, hci(t) is the neighborhood function that is centered in the winning neuron c, and α(t) is the learning rate at each iteration

The neighborhood function or kernel is a iteration-dependent shrinking function. It also depends on distance of unit i from the winning unit c. The influence that the input sample has in the SOM is determined in the equation below. It defines a bell-shaped influence region.

$$h_{c(\mathbf{x}),i} = \alpha(t) \cdot e^{\frac{-\|x_i, c_i\|^2}{2*\sigma^2(t)}} ,$$

where 0<α(t)<1 is the learning rate factor which decreases monotonically with each iteration, xi and ci are the vectorial locations of the sampled observation and the BMU in the grid, respectively, and σ(t) corresponds at the width of the neighborhood kernel, which also shrinks with the iterations.

The training phase is usually performed in two stages (as it is told in the Haykin's book; or must I tell bible? :D): the rough training (in that a relatively large initial learning rate and neighborhood radius are used) and the fine-tune (in that, small learning rate and neighborhood radius are applied from the beginning).

Once the map training is finished, the visualization of the two-dimensional map can be performed. It is also called the «components plane» and provides a qualitative information about how the input variables are related to each other for the given dataset. Furthermore, a «hits map» can be performed. The topological structure of this map is the same as a component of the components plane. It represents the number of times each map unit, or neuron, was the BMU for each input register, so that the distribution of the BMU for a given data is represented. This gives an idea of the number of input observations that gather in each neuron. This makes possible to compare the importance of each unit in the components plane.

V. OCR

Optical character recognition, usually abbreviated to OCR is the mechanical or electronic translation of images of handwritten, typewritten or printed text (usually captured by a scanner) into machine-editable text or computer process-able format, such as ASCII code. Whenever a page is scanned, it is stored as a bit-mapped file. When the image is displayed on the screen, we can read it. But it is just a series of dots for the computer. The computer does not recognize any "words" on the image.

OCR makes the computer read these words. It looks at each line of the image and determines which particular character is represented by dots. OCR is a field of research in pattern recognition, artificial intelligence and machine vision. Optical character recognition (using optical techniques such as mirrors and lenses) and digital character recognition (using scanners and computer algorithms) were originally considered separate fields. Because very few applications survive that use true optical techniques, the OCR term has now been broadened to include digital image processing as well.

OCR uses more than one approach when it comes to recognising text. The most basic way the technology distinguishes characters from pictures is through a technique known as pattern recognition. This involves a computer comparing objects within an image to letters already stored within its software. In other words, the software is equipped with a library of characters and the computer will search for the same patterns within your work and recognize when it finds a match.

The problem with pattern recognition, at least for our purposes, is that it cannot detect handwritten text. No one writes in Times New Roman, after all. Thankfully, as the technology has become more sophisticated, it increasingly relies on a different tactic known as feature extraction.

Rather than trying to recognize full letters, feature extraction occurs when a computer detects certain features (lines and loops, for example) and understands that they signify a character. The letter 'H', for instance, will be picked up by the software whenever it detects two vertical lines joined in the middle by a smaller, horizontal line.

This technique means that a computer's ability to recognize characters is not constrained to a limited number of fonts. From here, it can be trained to detect even handwritten text.

VI. Feature Extraction

In machine learning, pattern recognition and in image processing, feature extraction starts from an initial set of measured data and builds derived values (features) intended to be informative and non-redundant, facilitating the subsequent learning and generalization steps, and in some cases leading to better human interpretations. Feature extraction is a dimensionality reduction process, where an initial set of raw variables is reduced to more manageable groups (features) for processing, while still accurately and .completely describing the original data set

When the input data to an algorithm is too large to be processed and it is suspected to be redundant (e.g. the same measurement in both feet and meters, or the repetitiveness of images presented as pixels), then it can be transformed into a reduced set of features (also named a feature vector). Determining a subset of the initial features is called feature selection. The selected features are expected to contain the relevant information from the input data, so that the desired task can be performed by using this reduced representation instead of the complete initial data. to extract the OCR's . feature we used PCA and HOG

A. : HOG (histogram of oriented gradients)

The histogram of oriented gradients (HOG) is a feature descriptor used in computer vision and image processing for the purpose of object detection. The technique counts occurrences of gradient orientation in localized portions of an image. This method is similar to that of edge orientation histograms, scale-invariant feature transform descriptors, and shape contexts, but differs in that it is computed on a dense grid of uniformly spaced cells and uses overlapping local contrast normalization for improved accuracy.

The essential thought behind the histogram of oriented gradients descriptor is that local object appearance and shape within an image can be described by the distribution of intensity gradients or edge directions. The image is divided into small connected regions called cells, and for the pixels within each cell, a histogram of gradient directions is compiled. The descriptor is the concatenation of these histograms. For improved accuracy, the local histograms can be contrast-normalized by calculating a measure of the intensity across a larger region of the image, called a block, and then using this value to normalize all cells within the block. This normalization results in better invariance to changes in illumination and shadowing.

The HOG descriptor has a few key advantages over other descriptors. Since it operates on local cells, it is invariant to geometric and photometric transformations, except for object orientation. Such changes would only appear in larger spatial regions. Moreover, as Dalal and Triggs discovered, coarse spatial sampling, fine orientation sampling, and strong local photometric normalization permits the individual body movement of pedestrians to be ignored so long as they maintain a roughly upright position. The HOG descriptor is thus particularly suited for human detection in images , we used it in our OCR system to extract the features.

The data used to train the classifier are HOG feature vectors extracted from the training images. Therefore, it is important to make sure the HOG feature vector encodes the right amount of information about the object. The extract HOG Features function returns a visualization output that can help form some intuition about just what the "right amount of information" means. By varying the HOG cell size parameter and visualizing the result, you can see the effect the cell size parameter has on the amount of shape information encoded in the feature vector:

The visualization shows that a cell size of [8 8] does not encode much shape information, while a cell size of [2 2] encodes a lot of shape information but increases the dimensionality of the HOG feature vector significantly. A good compromise is a 4-by-4 cell size. This size setting encodes enough spatial information to visually identify a digit shape while limiting the number of dimensions in the HOG feature vector, which helps speed up training. In practice, the HOG parameters should be varied with repeated classifier training and testing to identify the optimal parameter setting .

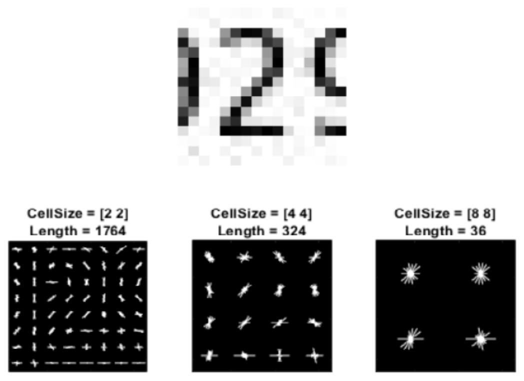


Figure.6. cell size of applied HOG on digits

The visualization on the figure 5 shows that a cell size of [8 8] does not encode much shape information, while a cell size of [2 2] encodes a lot of shape information but increases the dimensionality of the HOG feature vector significantly. A good compromise is a 4-by-4 cell size. This size setting encodes enough spatial information to visually identify a digit shape while limiting the number of dimensions in the HOG feature vector, which helps speed up training. In practice, the HOG parameters should be varied with repeated classifier training and testing to identify the optimal parameter setting.

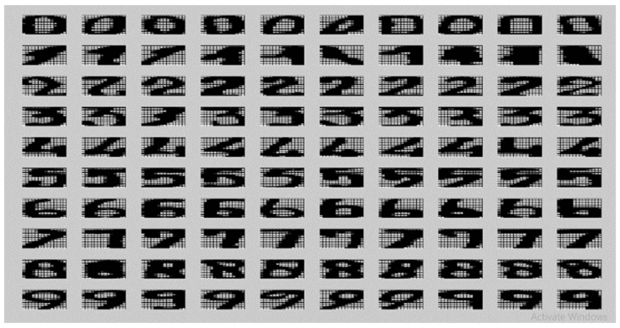


Figure 5. Extracted feature

B. PCA (Principal Component Analysis):

In statistics, principal components analysis (PCA) is a technique that can be used to simplify a dataset., more formally it is a linear transformation that chooses a new coordinate system for the data set such that the greatest variance by any projection of the data set comes to lie on the first axis (then called the first principal component), the second greatest variance on the second axis, and so on. PCA can be used for reducing dimnesionalty in a dataset while retaining those characteristics of the dataset that contribute most to its variance by eliminating the later principal components (by a more or less heuristic decision). These characteristics may be the "most important", but this is not necessarily the case, depending on the application.

PCA has the speciality of being the optimal linear transformation subspace that has largest variance. However this comes at the price of greater computational requirement. Unlike other linear transforms, the PCA does not have a fixed set of basis vectors, Its basis vectors depend on the data set.

Assuming zero empirical mean (the empirical mean of the distribution has been subtracted from the data set), the principal component w_i of a dataset x can be calculate by finding the eigenvalues and eigenvectors of the covariance matrix of x, we find that the eigenvectors with the largest eigenvalues correspond to the dimensions that have the strongest correlation in the dataset. The original measurements are finally projected onto the reduced vector space.

PCA can be thought of as fitting an n-dimensional ellipsoid to the data, where each axis of the ellipsoid represents a principal component. If some axis of the ellipsoid is small, then the variance along that axis is also small, and by omitting that axis and its corresponding principal component from our representation of the dataset, we lose only a commensurately small amount of information.

To find the axes of the ellipsoid, we must first subtract the mean of each variable from the dataset to center the data around the origin. Then, we compute the covariance matrix of the data, and calculate the eigenvalues and corresponding eigenvectors of this covariance matrix. Then we must normalize each of the orthogonal eigenvectors to become unit vectors. Once this is done, each of the mutually orthogonal, unit eigenvectors can be interpreted as an axis of the ellipsoid fitted to the data. This choice of basis will transform our covariance matrix into a diagonalised form with the diagonal elements representing the variance of each axis. The proportion of the variance that each eigenvector represents can be calculated by dividing the eigenvalue corresponding to that eigenvector by the sum of all eigenvalues.

This procedure is sensitive to the scaling of the data, and there is no consensus as to how to best scale the data to obtain optimal results.

PCA is mathematically defined as an orthogonal linear transformation that transforms the data to a new coordinate system such that the greatest variance by some projection of the data comes to lie on the first coordinate (called the first principal component), the second greatest variance on the second coordinate, and so on.

Consider a data matrix, X, with column-wise zero empirical mean (the sample mean of each column has been shifted to zero), where each of the n rows represents a different repetition of the experiment, and each of the p columns gives a particular kind of feature (say, the results from a particular sensor).

Mathematically, the transformation is defined by a set of p-dimensional vectors of weights or coefficients {\displaystyle \mathbf {w} _{(k)}=(w_{1},\dots ,w_{p})_{(k)}} that map each row vector {\displaystyle \mathbf {x} _{(i)}} of X to a new vector of principal component scores {\displaystyle \mathbf {t} _{(i)}=(t_{1},\dots ,t_{l})_{(i)}} , given by :

$$t_{k(i)} = \mathbf{x}_{(i)} \cdot \mathbf{w}_{(k)} \qquad \text{for} \qquad i = 1, \dots, n \qquad k = 1, \dots, l$$

in such a way that the individual variables {\displaystyle t_{1},\dots ,t_{l}} of t considered over the data set successively inherit the maximum possible variance from x, with each coefficient vector w constrained to be a unit vector.

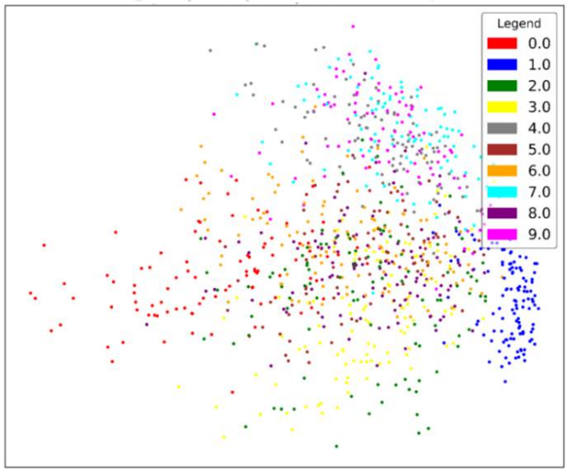
VII. IMPLEMENTATION RESULTS

In this section we are going focus on checking our assumptions. So far we have learned how to perform a K Means Cluster and SOM . When running a K Means Cluster, you first have to choose how many clusters you want. But what is the optimal number of clusters? This is the “art” .part of an algorithm like this
One thing you can do is check the distance from you points .to the cluster center

: Let’s start to see the results

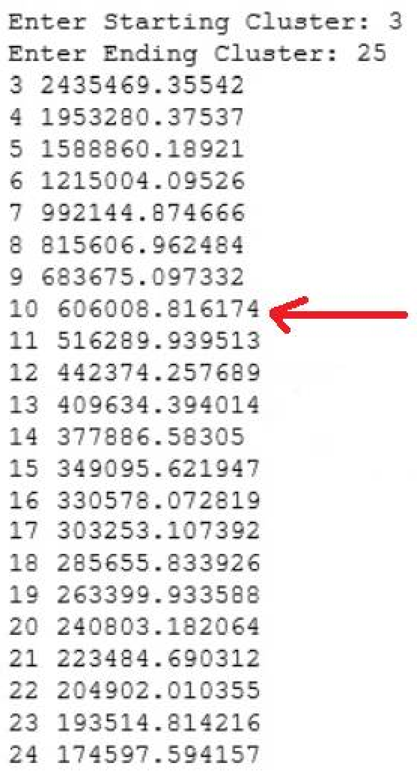
✓ : K-Means

Figure ٧ / Figure ٧ shows the variety of Scatter plot of :embedding digits



.Figure ٧ Scatter plot of embedding

First of all we applied different K to see in which cluster we : have the best performance



```
Enter Starting Cluster: 3
Enter Ending Cluster: 25
3 2435469.35542
4 1953280.37537
5 1588860.18921
6 1215004.09526
7 992144.874666
8 815606.962484
9 683675.097332
10 606008.816174
11 516289.939513
12 442374.257689
13 409634.394014
14 377886.58305
15 349095.621947
16 330578.072819
17 303253.107392
18 285655.833926
19 263399.933588
20 240803.182064
21 223484.690312
22 204902.010355
23 193514.814216
24 174597.594157
```

. Figure ٨

As you can observe in figure ٨ ’ in K=١٠ (١٠ cluster) there was a sudden reduction in error rate , which makes sense , . because we have ١٠ digits , and it works very well

: Figure ٩ is showing the clustering of digits

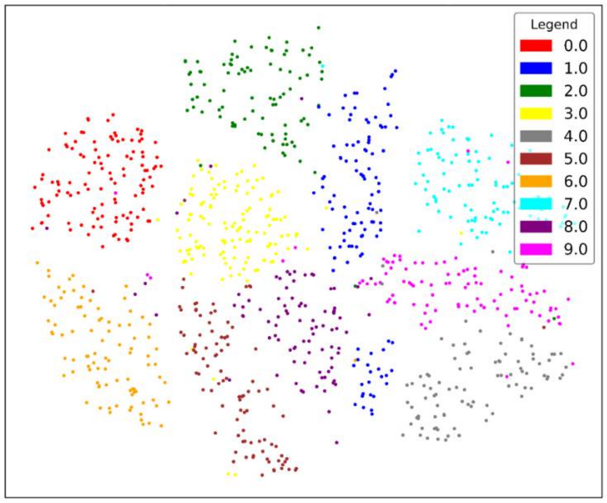


Figure ٩ Scatter plot of embedding

Digits	of Digits #
١	١١٠٠
٢	٧٠٠
٣	٦٠٠
٤	٥٠٠
٥	٥٥٠
٦	٣٥٠
٧	٢٠٠
٨	٤٠٠
٩	٣٠٠
٠	١٢٠٠

Table ١ / number of each cluster

✓ : SOM

. Then we applied SOM and got a better result
: Look at figure ١٠/ It shows the error rate

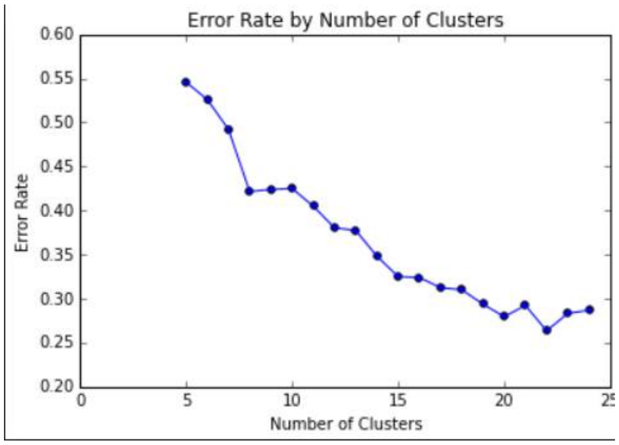


Figure ١٠

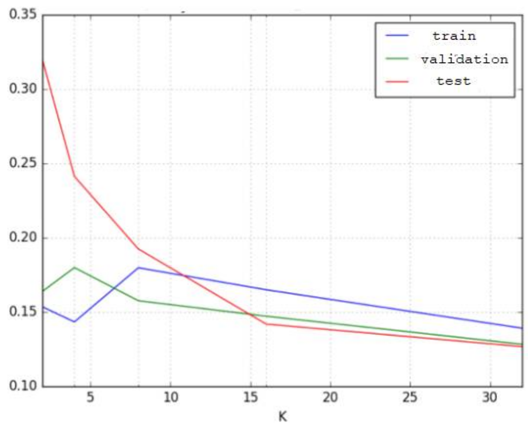


Figure ١١/ error rate of training , validation and test phases

VIII. Conclusion

in this paper we have discussed unsupervised learning and OCR and some of the feature extraction techniques , also we studied about K-Means and SOM we observed that in K-means the nodes (centroids) are independent from each other. The winning node gets the chance to adapt each self and only that. In SOM the nodes (centroids) are placed onto a grid and so each node is consider to have some neighbors, the nodes adjacent or near to it in repspect with their position on the grid. So the winning node not only adapts itself but causes a change for its neighbors also. K-Means can be considered a special case of SOM were no neighbors .are taken into account when modifing centroids vectors

References

[1] Vuokko Vuori, Erkki Oja, "Analysis of different writings styles with the self organizing map". In Proceedings of the 7th International Conference on neural information processing. Vol. 2., 2000, pp 1243-1247. November 2000.

[2] A.Ultsch, H.P.Siemon(1990), Kohonen's Self Organizing Feature Maps for Exploratory Data Analysis, Proc. Intern. Neural Networks, Kluwer Academic Press, Paris, 305-308.

[3] Mumford, D (1994). Neuronal architectures for pattern-theoretic problems. In C Koch and J Davis, editors, Large-Scale Theories of the Cortex. Cambridge, MA: MIT Press, 125-152.

[4] Kohonen, T.: Self-organizing maps. Springer Series in Information Sciences (2001),

[5] R. C. Gonzalez and R. E. Woods, Digital Image Processing (2nd Edition). Prentice Hall, January 2002. .

[6] Marinai, S., Gori, M., Soda, G.: Artificial neural networks for document analysis and recognition. IEEE Transactions on PAMI 27(1) (2005) 23–35

[7] Avi-Itzhak, H., Diep, T., Garland, H.: High accuracy optical character recognition using neural networks with centroid dithering. IEEE Transaction on PAMI 17(2) (1995) 218–224 .

Feature selection using principal component analysis Fengxi Song, Zhongwei Guo, Dayong Mei Department of Automation and Simulation New Star Research Inst. of Applied Tech. in Hefei City Hefei, China

[8] H.W. Liu, J. G. Sun, L. Liu, H. J. Zhang. Feature selection with dynamic mutual information. Pattern Recognition, 42 (7):1330 1339, July 20.