Some Applications Based On KMeans and Principle Components Analysis

1st Xiong Hongwei
School Of System Design and Intelligent Manufacturing
Southern University of Science and Technology
Shenzhen, China
12212203@mail.sustech.edu.cn

***Abstract*—This is the final project report of SDM274. Including some analysis on the experiment based on K-means and Principle Components Analysis.**

***Index Terms*—K-means, Soft K-means, Principle Components Analysis, Linear Auto Encoder**

I. Introduction

Both K-means and Principle Components Analysis are popular unsupervised learning methods. The experiments explore the effect of the two learning methods.

II. Problem and formulation

A. Clustering Problems

In our daily life, we may meet with many clustering problems. We have to divide things into different groups based on their characters. When using supervised learning methods, the algorithm have a clear goal to produce a desired output for given input. This need labels in training data set. But in clustering problems, the labels are unknown. So we need unsupervised learning methods such as K-means and Principle Components Analysis.

B. K-means

K-means clustering is the most basic and commonly used clustering algorithm. The basic idea is to find a partition scheme for K clusters through iteration, so that the loss function corresponding to the clustering result is minimized.

C. Principle Components Analysis

It is one of the most widely used data dimensional reduction algorithms. The main idea of Principle Components Analysis is to map n-dimensional features to k-dimension, which is a new orthogonal feature also known as principal component, which is a reconstructed k-dimensional feature on the basis of the original n-dimensional feature.

D. Formulation in K-means algorithm

K-means Objective: Find cluster $\mathbf{m}$ and assignment $\mathbf{r}$ to minimize the sum of squared distances of data points $\mathbf{x}^{(n)}$ to their assigned cluster centers

$$\min_{\{\mathbf{m}\},\{\mathbf{r}\}} J(\{\mathbf{m}\},\{\mathbf{r}\}) = \min_{\{\mathbf{m}\},\{\mathbf{r}\}} \sum_{n=1}^{N}\sum_{k=1}^{K} r_k^{(n)} ||\mathbf{m}_k - \mathbf{x}^{(n)}||^2$$

$$\text{s.t.} \sum_k r_k^{(n)} = 1, \forall n, \quad \text{where} \quad r_k^{(n)} \in \{0,1\}, \forall k, n$$

where $r_k^{(n)} = 1$ means that $x^{(n)}$ is assigned to cluster k (with center m_k)

E. Formulation in Principle Components Analysis

To find Principle Components we calculate the empirical covariance matrix:

$$C = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}^{(n)} - \bar{\mathbf{x}})(\mathbf{x}^{(n)} - \bar{\mathbf{x}})^T$$

Minimize the reconstruction error:

$$J(\mathbf{u}, \mathbf{z}, \mathbf{b}) = \sum_n ||\mathbf{x}^{(n)} - \tilde{\mathbf{x}}^{(n)}||^2$$

where

$$\tilde{\mathbf{x}}^{(n)} = \sum_{j=1}^{M} z_j^{(n)}\mathbf{u}_j + \sum_{j=M+1}^{D} b_j \mathbf{u}_j$$

Objective minimized when first $\mathbf{M}$ components are the eigenvectors with the maximal eigenvalues

$$z_j^{(n)} = \mathbf{u}_j^T \mathbf{x}^{(n)}; \quad b_j = \bar{\mathbf{x}}^T \mathbf{u}_j$$

Auto Encoder

Define

$$\mathbf{z} = f(W\mathbf{x}); \quad \hat{\mathbf{x}} = g(V\mathbf{z})$$

Goal:

$$\min_{\mathbf{W},\mathbf{V}} \frac{1}{2N}\sum_{n=1}^{N} ||\mathbf{x}^{(n)} - \hat{\mathbf{x}}^{(n)}||^2$$

If g and f are linear

$$\min_{\mathbf{W},\mathbf{V}} \frac{1}{2N}\sum_{n=1}^{N} ||\mathbf{x}^{(n)} - VW\mathbf{x}^{(n)}||^2$$

III. Method and algorithm

A. K-means

The algorithm iterative alternates between two steps:

Assignment step: Assign each data point to the closest cluster. **Refitting step:** Move each cluster center to the center of gravity of the data assigned to it.

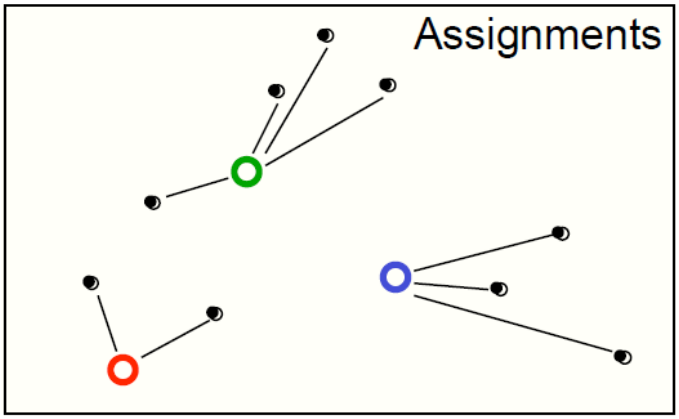


Fig. 1.

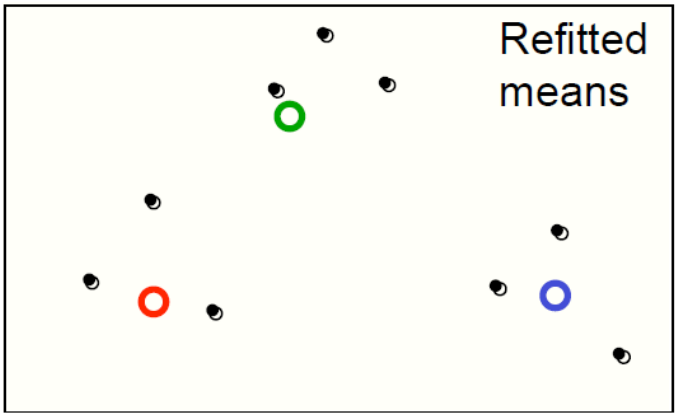


Fig. 2.

Initialization: Set K cluster means $\mathbf{m}_1, \ldots, \mathbf{m}_K$ to random va
Repeat until convergence (until assignments do not change):
Assignment: Each data points $x^{(n)}$ assigned to nearest mean

$$\hat{k}^n = \arg\min_k d(\mathbf{m}_k, \mathbf{x}^{(n)})$$

and **Responsibilities**(1 of k encoding):

$$r_k^{(n)} = 1 \longleftrightarrow \hat{k}^{(n)} = k$$

Update: Model parameters, means are adjusted to match sample means of data points they are responsible for:

$$\mathbf{m}_k = \frac{\sum_n r_k^{(n)} \mathbf{x}^{(n)}}{\sum_n r_k^{(n)}}$$

***Soft K-means algorithm* Initialization:**

Set K means $\{m_k\}$ to random values Repeat until convergence (until assignments do not change):

Assignment:

Each data point n given soft "degree of assignment" to each cluster mean k, based on responsibilities

$$r_k^{(n)} = \frac{\exp[-\beta d(\mathbf{m}_k, \mathbf{x}^{(n)})]}{\sum_j \exp[-\beta d(\mathbf{m}_j, \mathbf{x}^{(n)})]}$$

Update:

Model parameters, means, are adjusted to match sample means of data points they are responsible for:

$$\mathbf{m}_k = \frac{\sum_n r_k^{(n)} \mathbf{x}^{(n)}}{\sum_n r_k^{(n)}}$$

B. Principle Components Analysis

Algorithm: to find M components underlying D-dimensional data

1. Select the top M eigenvectors of C (data covariance matrix):

$$C = \frac{1}{N} \sum_{n=1}^{N} (\mathbf{x}^{(n)} - \bar{\mathbf{x}})(\mathbf{x}^{(n)} - \bar{\mathbf{x}})^T = U \Sigma U^T \approx U_{1:M} \Sigma_{1:M} U_{1:M}^T$$

where U is orthogonal, columns are unit-length eigenvectors

$$U^T U = U U^T = 1$$

and Σ is a matrix with eigenvalues on the diagonal, representing the variance in the direction of each eigenvector.

2. Project each input vector x into this subspace, e.g.,

$$z_j = \mathbf{u}_j^T \mathbf{x}; \quad \mathbf{z} = U_{1:M}^T \mathbf{x}$$

An auto encoder is a neural network whose outputs are its own inputs.

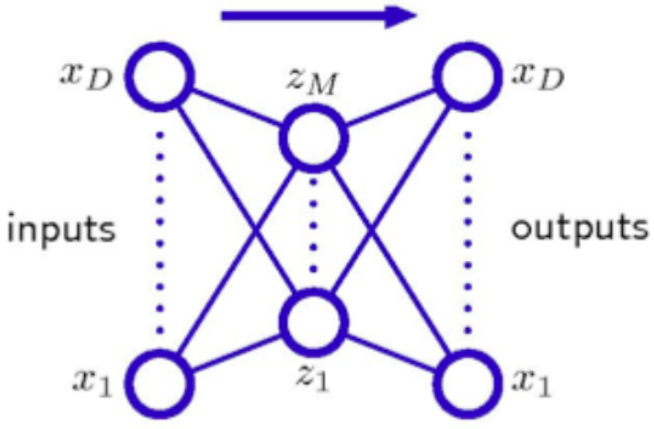


Fig. 3. Number 3

IV. Experiment results and analysis

A. KMeans and Soft KMeans

In the experiments, we first use KMeans and Soft KMeans with K=3 for clustering. We will use both objective function and precision in the first experiment. In the second experiment, we will only use objective function to estimate the performance of KMeans and Soft KMeans with K=10. The data sets used in these two experiments include 210 samples with seven input feature. More information about the data sets are conclude in the following picture:

列号	列名	含义	特征 / 类标记	可取值
1	area	区域	特征	实数
2	perimeter	周长	特征	实数
3	compactness	紧密度	特征	实数
4	length of kernel	籽粒长度	特征	实数
5	width of kernel	籽粒宽度	特征	实数
6	asymmetry coefficient	不对称系数	特征	实数
7	length of kernel groove	籽粒腹沟长度	特征	实数
8	class	类别	类标记	1,2,3

Fig. 4. Information about data sets

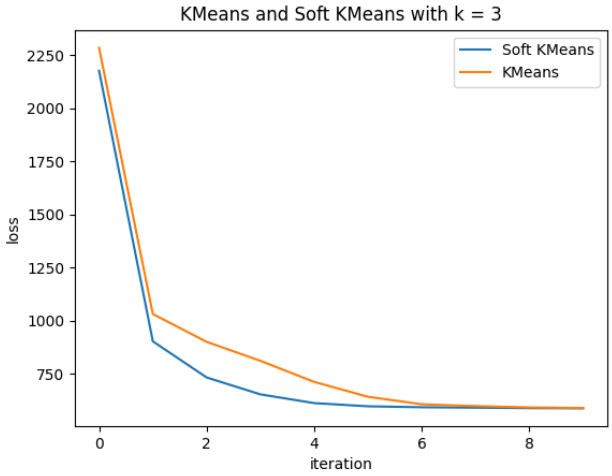
1) KMeans and Soft KMeans with K=10:

```
KMeans with k = 3:
precision:
cluster 1:  0.8142857142857143
cluster 2:  0.8428571428571429
cluster 3:  0.9857142857142858
```

Fig. 5. Precision of KMeans with K=3

```
Soft KMeans with k = 3:
precision:
cluster 1:  0.8142857142857143
cluster 2:  0.8428571428571429
cluster 3:  0.9857142857142858
```

Fig. 6. Precision of Soft KMeans with K=3



We can learn from the figure that when k=3, the performance of KMeans and Soft KMeans are very close. What's more, the Soft KMeans loss may descent faster than the KMeans. They both converge in near 10 epochs. And the precision of them are both well on the dataset.

2) KMeans and Soft KMeans with K=10

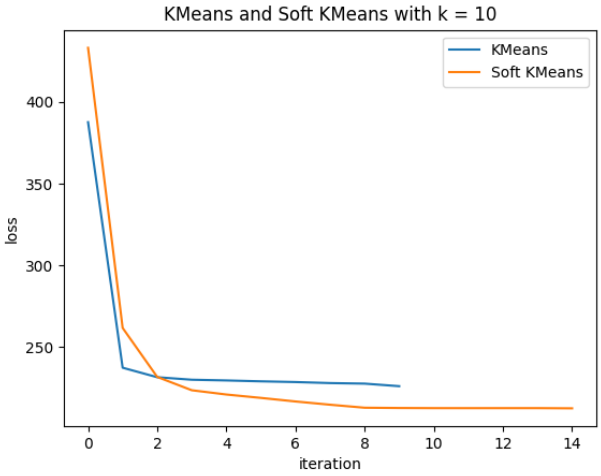


Fig. 7.

In this experiment, Soft KMeans performances better than KMeans with faster loss descent and lower converged loss. This will become more obvious with K bigger. This is the advantage of Soft KMeans algorithm.

3) KMeans and Modified KMeans with K=10

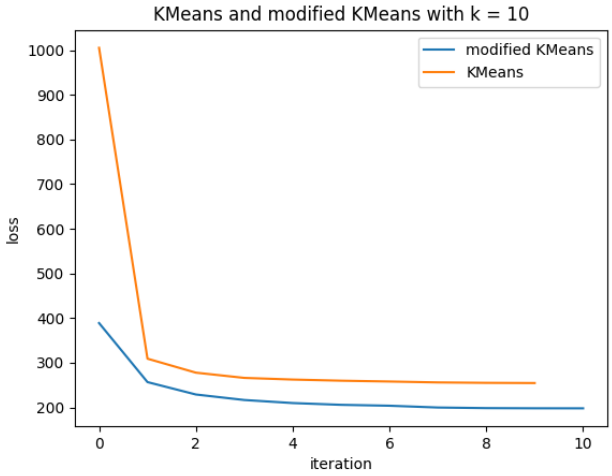


Fig. 8.

In this experiment, the original KMeans and the modified KMeans(adding non-local split-and-merge moves). From the figure we can know that the KMeans algorithm have got into the local minimum and cannot get out of it. But with non-local split-and-merge moves, the modified KMeans algorithm can get out of the local minimum and have a lower loss to converge.

4) Soft KMeans and Modified Soft KMeans with K=10

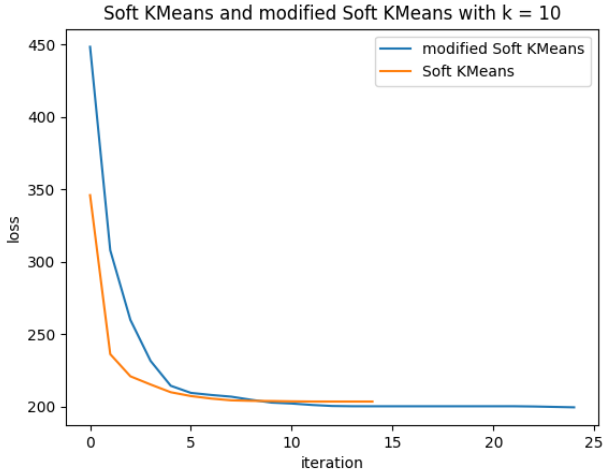


Fig. 9.

In this experiment, the original Soft KMeans and the modified Soft KMeans(adding non-local split-and-merge moves). From the figure we can know that the Soft KMeans algorithm have got into the local minimum and cannot get out of it. But with non-local split-and-merge moves, the modified Soft KMeans algorithm can get out of the local minimum and have a lower loss to converge. But the difference of their loss is a little bit.

5) Modified KMeans and Modified Soft KMeans with K=10

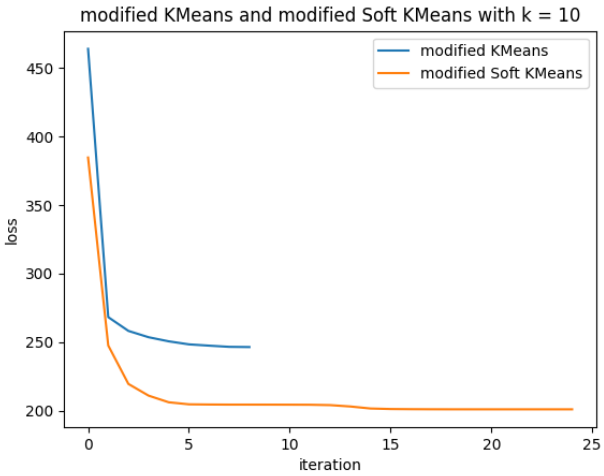


Fig. 10.

From the figure, the modified KMeans had converged to a number faster than the modified Soft KMeans. But the modified KMeans had a much higher loss than the modified Soft KMeans.

B. PCA and Linear Auto Encoder on an image

The following experiments choose a picture of sun flower as the input image. And for calculation convenient, the image are resized to (100,100) and converted to gray-scale graph.

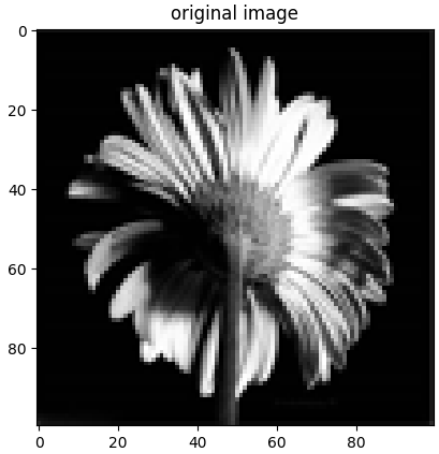


Fig. 11. Original image

1) PCA In the PCA algorithm, we have to set the number of principle components which is a hyper parameter. Different number of principle components are tried in the experiment to show the effect of this key hyper parameter.

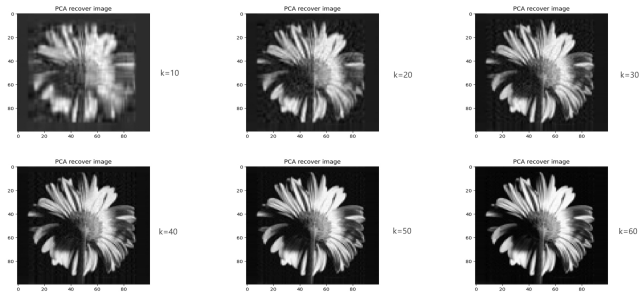


Fig. 12. PCA

From the figure, the k=30 should be a suitable number of the principle components. And the k=30 will be used in the following Linear Auto Encoder and KMeans experiments as their hyper parameter hidden layer sizes and clusters.

2) Linear Auto Encoder

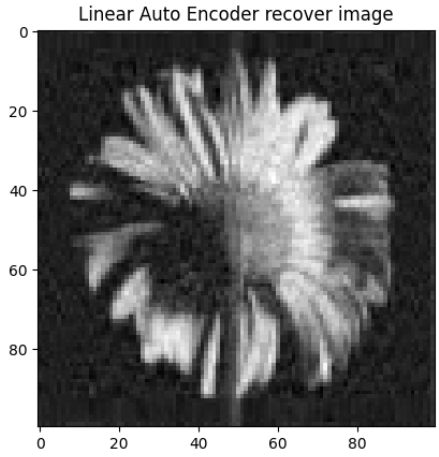


Fig. 13. Linear Auto Encoder

The hidden layer's units was set to 30 (due to the last experiments result). Because of the big gradient at first, we have to set a very little learning rate to keep the nan far away. And a small learning rate also means a much higher epoch for iterating. In this experiments, we set learning rate 0.00001 and epoch 1000000 to get a better result.

Compare this figure with the figure in PCA experiment, we find they are very similar, actually from the perspective of math, the PCA is a optimal solution of a linear auto encoder.

C. KMeans on an image

1) KMeans

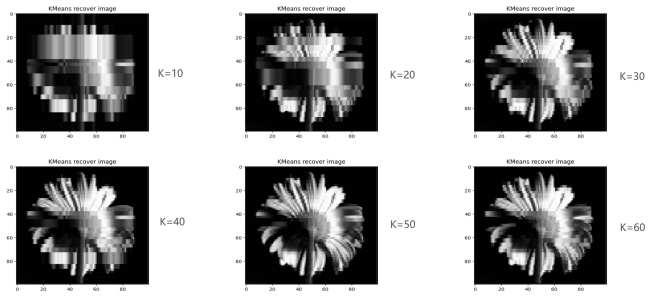


Fig. 14. KMeans on image

In this experiment, we can find the KMeans on image have a very different effect that it will use a single number to represent a cluster of point nearby it so the image would look like make up of some square with different colors (or gray scale).

V. Conclusion and future problems

Conclusion

- We show the KMeans and Soft KMeans models' performance on the clusterng problems with K=3 and K=10. And we analyze the reults and reasons. What's more, the non-local split-and-merge was added into the two algorithm and bring better performance.
- We show PCA and KMeans algorithm on image with different k and compare them from landscape and portrait. We also use a linear auto encoder to encode and decode the same image and valid that PCA is a optimal solution to linear auto encoder from the experiments.

Future problems

- In the experiments, we can use some colorful image by separating them into RGB three dimensions. Using colorful image maybe can show the effect more obvious.
- We can add non-linear activation functions to the auto encoder, or add more hidden layers to test its performance and compare with the result of PCA.

References

[1] SDM274 lectures