KMEANS and Kernel KMEANS

A comparative study of classical and kernel kmeans for pixel clustering.

Ezukwoke K.I[1], Zareian S.J[2]

[1, 2] Department of Computer Science
Machine Learning and Data Mining
{ifeanyi.ezukwoke, samaneh.zareian.jahromi}@etu.univ-st-etienne.fr
University Jean Monnet, Saint-Etienne, France

Abstract

Kmeans is a simple and efficient clustering algorithm. In this paper we study kernel Kmeans and compare it classical kmeans. We take an experimental analysis on image dataset for pixel clustering. We briefly introduce multiple kernel learning and its applications in kernel kmeans. We conclude by expanding on the performance of both algorithms to see which is most suited for pixel clustering.

Keywords
Partition clustering, K-Means, kernel K-Means.

1 INTRODUCTION

K-Means clustering is a fast, robust, and simple algorithm that gives reliable results when data sets are distinct or well separated from each other in a linear fashion. It is best used when the number of cluster centers, is specified due to a well-defined list of types shown in the data. However, it is important to keep in mind that K-Means clustering may not perform well if it contains heavily overlapping data, if the Euclidean distance does not measure the underlying factors well, or if the data is noisy or full of outliers

2 PARTITION CLUSTERING

Clustering is the task of gathering samples into groups of similar samples according to some predefined similarity and they find applications in *data compression*, *data summurization for recommender systems*, *similarity grouping of web search result, stock indices and customer profiles.*

KMeans falls under the clustering category called **partition clustering**. This approach used a technique of splitting the datasets into subgroups of $k - clusters$ and iteratively tries to find the best $k - cluster$ that best explains the partition of a data.

2.1 KMEANS

The objective of traditional clustering methods is to partition training vectors by using similarity criteria applied on the basis of Euclidean metrics. More precisely, the Euclidean distance or inner product is used to measure the similarity between the training vectors in the original vector space, $\{x_i, i = 1, \ldots, N\}$. The objective function we try to minimize in KMeans is give by

$$argmin_{w_k} \left\{ \sum_{k=1} K \sum_{X_t \in C_k} \|x_t - \mu_k\|^2 \right\} \quad (1)$$

Where μ_k denotes the *kth* cluster's centroid. In an optimal K-means solution, the centroid, say μ_k is associated with a training vector x_j that yields the minimum distance among all the centroids.

Algorithm 1: Classical K-Means clustering algorithm

Input : x
Output : $x_k \in C_k$

begin
  while *not converged* **do**
    Randomly initialize μ cluster center;
    Compute the distance of each point in x from center μ $\arg\min \sum_{c=1}^{k} \sum_{x_i \in C_k} ||x_i - m_c||^2$
  end
  return x_k
end

2.2 Kernel KMEANS

Kernel Kmeans is a non-lnear version of the classical kmeans algorithm. The data x_i is projected into a higher dimensional subspace $\phi(x_i)$ where we are only concerned in the dot products of the feature vectors.

Algorithm 2: Kernel K-Means clustering algorithm

Input : x
Output : $x_k \in C_k$

begin
  while *not converged* **do**
    Randomly initialize μ cluster center;
    Compute the distance of each point in x from center μ $\arg\min \sum_{c=1}^{k} \sum_{x_i \in C_k} ||\phi(x_i) - m_c||^2$
  end
  return x_k
end

2.3 Kernels

We introduce the commonly used kernels and a brief overview of Multiple kernels used.

- **Linear kernel**

$$\kappa(x_i, x_j) = \mathbf{x}_i \mathbf{x}_j^T \tag{2}$$

- **Polynomial kernel**

$$\kappa(x_i, x_j) = (\mathbf{x}_i \mathbf{x}_j^T + c)^d \tag{3}$$

where $c \geq 0$ and d is the degree of the polynomial usually greated than 2.

- **RBF(Radial Basis Function) kernel**
Sometimes referred to as the **Guassian kernel**.

$$\kappa(x_i, x_j) = \exp(-\gamma ||\mathbf{x}_i - \mathbf{x}_j||^2) \tag{4}$$

where $\gamma = \frac{1}{2\sigma^2}$.

- **Sigmoid kernel**

$$\kappa(x_i, x_j) = \tanh(\gamma \mathbf{x}_i \mathbf{x}_j^T + c) \tag{5}$$

where $c \geq 0$ and $\gamma = \frac{1}{2\sigma^2}$.

- **Cosine kernel**

$$\kappa(x_i, x_j) = \frac{\mathbf{x}_i \mathbf{x}_j^T}{||\mathbf{x}_i|| ||\mathbf{x}_j||} \tag{6}$$

2.4 Muli-kernels

The reason behind the use of multiple kernels is similar to the notion of multiclassification, where cross-validation is used to select the best performing classifier [**?**]. By using multiple kernels, we hope to learn a different similarity uncovered using single kernels.

We can prove from **Mercer's Theorem** that a kernel is **Positive Definite (PD)** if $\kappa(x_i, x_j) \geq 0$. Hence by performing arithmetic or any mathematical operation on two or more kernel matrix, we obtain a new kernel capable of exploiting different property or similarities of training data.

Given a kernel κ, we prove that κ is PD if

$$\langle u, \kappa u \rangle \geq 0 \tag{7}$$

Proposition: *A symmetric function* κ*:* $\times$ $\chi \to \mathbb{R}$ *is positive definite* $\iff$ κ
Proof:
Suppose that κ is a kernel which is the inner product of the mapping functions

$\langle \phi(x_i)\phi(x_j) \rangle$. κ is a kernel if its inner product are positive and the solution of $\kappa u = \lambda u$ gives non-negative eigenvalues.

So that,

$$\langle u, \kappa u \rangle = \sum_{i=1}^{N} u_i (\kappa u)_i \tag{8}$$

$$= \sum_{i=1}^{N} u_i \sum_{j=1}^{N} \langle \phi(x_i)\phi(x_j)_{\mathcal{H}} \rangle u_j \tag{9}$$

Where $\mathcal{H}$ represents the Hilbert space we project the kernel in[**?**].

$$= \left\langle \sum_{i=1}^{N} u_i \phi(x_i), u_j \phi(x_j) \right\rangle_{\mathcal{H}} \tag{10}$$

$$\langle u, \kappa u \rangle = \left\| \sum_{i=1}^{N} u_i \phi(x_i) \right\|_{\mathcal{H}}^2 \geq 0 \tag{11}$$

Therefore κ is positive definite.

Using this property of the kernel κ we are able to create serveral other kernels including

- **LinearRBF**
  Here we combine two kernels, precisely **Linear and RBF kernel** using their inner product.

$$\hat{\mathbf{K}}_{\mathbf{linrbf}} = \kappa(x_i, x_j)^T \kappa(x_i, x_l) \tag{12}$$

- **RBFPoly**
  Here we combine **RBF and Polynomial kernels** using their inner product.

$$\hat{\mathbf{K}}_{\mathbf{rbfpoly}} = \kappa(x_i, x_j)^T \kappa(x_i, x_l) \tag{13}$$

- **RBFCosine**
  RBF and Cosine kernels using their inner product.

$$\hat{\mathbf{K}}_{\mathbf{rbfcosine}} = \kappa(x_i, x_j)^T \kappa(x_i, x_l) \tag{14}$$

- **EtaKernel**
  The **EtaKernel** is a composite combination of **LinearRBF, RBFPoly and RBFCosine** and it is given by

$$\hat{\mathbf{K}}_{\mathbf{etarbf}} = \hat{\mathbf{K}}_{\mathbf{linrbf}}^T \hat{\mathbf{K}}_{\mathbf{rbfpoly}} + \hat{\mathbf{K}}_{\mathbf{rbfpoly}}^T \hat{\mathbf{K}}_{\mathbf{rbfcosine}}$$

3 EXPERIMENTAL RESULT

We experiment on images dataset benchmark.

3.1 Dataset

3.2 Performance analysis

4 CONCLUSION

References

[1] Pearson K. On lineas and Planes of closest fit to systems of points in space. *Philos Mag A*, 6:559–572, 1901.

[2] Zhu J, Hastie T. Kernel logistic regression and import vector machine. *J Comput Graphic Stat*, 14:185–205, 2005.

[3] Saunders C., Gammerman A., Vovk V., Ridge Regression Learning Algorithm in Dual Variables. *ICML 15th Internationl Conforence on Machine Learning.*, 1998.