Experimentation with Clustering Algorithms

Corinne Curcie and Shivam Naik

March 5, 2017

Abstract

Clustering algorithms are a popular tool for making sense of big data. Our project involved implementing various algorithms, focusing specifcally on high-dimensional data, to gain better understanding .

1. Introduction

2. Related Work

3. Algorithms Implemented

3.1 K-Means

One of the algorithms we implemented is a variant on K-Means. Before getting into the details of that algorithm, we will go over the basics of K-Means, since we also used it as a benchmark when scoring preformances of algorithms we implemented. The well-known K-Means algorithm starts by choosing k points in the dataset to be the initial cluster center points, and then updates on an "expectation-maximization mechanism" (EM).
The "E-step" is the cluster assignment step, where points are labelled with a cluster based on which of the center points they are closest to. The "M-step" is the update step, where the k cluster center points are recalculated to be some average of all of the points that were labeled as belonging to that cluster during the previous "E-step" round. The algorithm stops when the change in centers from one round to the next is less than some predetermined threshold. The objective function that K-Means tries to minimize is the sum of squared Euclidean distances from each point to its assigned cluster center:

$$\arg\min_{\mathbf{S}} \sum_{i=1}^{k} \sum_{\mathbf{x} \in S_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|^2$$

3.2 K-Subspaces

We were intrigued by a 2009 paper that proposed a "K-Subspaces" algorithm similar to K-Means (Wang, Ding, and Li 2009). Instead of using Euclidean distances for our objective function, multiple distance measures are used during the "E-step" to determine which centers the points are closest to. The authors of the algorithm decided to focus on three possible subspaces – 1D lines, 2D planes, and 3D spheres – and determines distance functions based on those subspaces. By calculating all three distances for each pair of point and cluster center, the algorithm is better able to determine when a point is within a cluster of a non-standard space. To extend this performance not only to the entire dataset but also subspaces within the dataset, PCA is used during distance calculations to isolate meaningful subspaces

K-Subspaces Initialization

For initializing cluster centers before beginning the EM steps, we used the standard "K-means++" algorithm, which probabilistically selects initial clusters. Specifically, the algorithm goes through the following steps: (1) Select an input data point uniformly at random to be the first center. (2) Calculate the distance between every point x and its nearest center point that has already been chosen. (3) Select a new data point as a next center, using a probability distribution where a point is chosen with probability proportional to the distance calculated in step 2. (4) Repeat the steps 2 and 3 until k centers have been chosen. The "K-means++" initialization is also used in scikit-learns implementation of K-Means.

Cluster Assignment Step

In our cluster assignment step is when we use the new distance measures proposed by the authors. Instead of finding the direct distance to the center of a cluster, they aim to estimate a *surface* of a cluster by using perpendicular distances. For the following equations, let x be a point and C_k represent a cluster where c_k is the center point of the cluster.

Line Distance:

$$Dist(\mathbf{x}, C_k) = ||\mathbf{x}^{\perp}||^2 = ||\mathbf{x} - \mathbf{c}_k - \alpha\mathbf{a}_k||^2$$

where

$$\alpha = (\mathbf{x} - \mathbf{c}_k)^T\mathbf{a}_k.$$

Here, a_k represents the first principal component vector found using PCA.

Plane Distance:

$$Dist(\mathbf{x}, C_k) = ||\mathbf{x}^{\perp}||^2 = ||\mathbf{x} - \mathbf{c}_k - \alpha\mathbf{a}_k - \beta\mathbf{b}_k||^2$$

where

$$\alpha = (\mathbf{x} - \mathbf{c}_k)^T \mathbf{a}_k, \quad \beta = (\mathbf{x} - \mathbf{c}_k)^T \mathbf{b}_k.$$

Here, a_k again represents the first principal component vector found using PCA, and b_k is the second principal component.

Sphere Distance:

$$\min_{\mathbf{c}_k} \sum_{i \in C_k} \max \left(0, ||\mathbf{x}_i - \mathbf{c}_k||^2 - \eta \sum_{j \in C_k} ||\mathbf{x}_j - \mathbf{c}_k||^2 \right)$$

DBScan

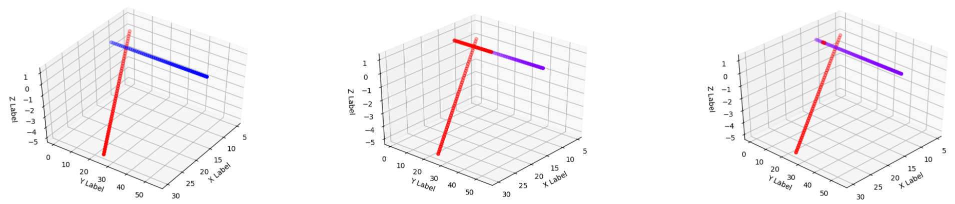
DBScan stands for Density-Based Algorithm for Discovering Clusters and was proposed by Martin Ester, Hans-Peter Kriegel, Jiirg Sander, Xiaowei Xu in 1996. The algorithm tries to find different clusters in the data based on two main parameters which are epsilon, the threshold for distance between two points, and min points, which is the threshold for minimum number of points required for a cluster. The samples from the dataset are classified in the clusters under 3 types of points: core points, border points, and noise points. Core points are within the epsilon of a center point, border points lie on the edges and can be flipped to either of the close clusters, and noise points, which dont belong to any clusters. The clusters are created by finding points closest to each other. We start with one point and find all the clusters closest to it and then find all the points closest to that cluster.

The algorithm: We have a list of cluster ids which is how we keep track of which sample belongs to which cluster. We go through each point and if it hasnt been classified yet, then we ExpandCluster. Expand points starts by checking to see whether the current sample has at least min points number of neighbors by checking to see if the distance between the two samples is less than epsilon. If it fails to have enough neighbors with a short distance, all points are marked as NOISE. Otherwise, it goes through all the neighbors and finds all the points that it can reach to swallow into this its cluster by marking all cluster ids as the same.

How to determine eps: Find the k nearest(for our purposes 3) points and get the distance. Sort all distances and set the eps to the midway point. Will keep min points set to 4 according to the paper even though our data has a higher dimension than 2.

4. Datasets Used

5. Performance



(a) Original Data (b) K-Means (c) K-Subspaces

Figure 1: Performance on two 1-D Subspaces

K-Subspaces on Synthetic Data

For our synthetic data, we ran a very similar synthetic experiment to the one presented in the 2009 paper. We wanted to demonstrate the ability of K-subspaces to cluster for a variety of shapes. Data includes 1D lines, 2D planes, and 3D spheres, and the goal is for K-Subspaces to cluster those shapes together separately even when they are close together. We are comparing the performance with scikit-learn's implementation of K-Means.