Background on the c-means Algorithm for CPSC/ECE 3520

RJ Schalkoff

Fall 2018

Contents

1 c-means Algorithm Background

The c-means algorithm is an interesting and useful algorithm for multidimensional (often high-dimensional) data analysis, data mining and visualization. This application includes clustering and unsupervised learning. **Most importantly, c-means is the focus of SDEs 2 amd 3 in ECE/CPSC 3520 for Fall 2018. A prerequisite for success in this SDE is a good understanding of the algorithm and accompanying notation.** That is the objective of this document.

1.1 Notation

All vectors are denoted with an underbar and considered column vectors. The superscript (^{T}) indicates transpose. In what follows, most column vectors will be represented by an `ocaml` list.

1.2 J_{SSE}, The Sum of Squared Error (SSE) Criterion

We briefly show one of the more popular clustering metrics. Given n_i samples in subset H_i, with sample mean $\underline{\mu}_i$, where

$$\underline{\mu}_i = \frac{1}{n_i} \sum_{\underline{x}_j \in H_i} \underline{x}_j \tag{1}$$

the so-called SSE criterion, J_{SSE}, is defined as

$$J_{SSE}(P) = \sum_{i=1}^{c} \sum_{\underline{x}_j \in H_i} \| \underline{x}_j - \underline{\mu}_i \|^2 \tag{2}$$

J_{SSE} thus indicates the total 'variance' for a given partition. It also serves as an indicator of how well the c means, $\underline{\mu}_i$, represent the members of H.

1.3 Computational Complexity and The Role of c-means

There are approximately $\frac{c^n}{c!}$ possible partitions of n vectors into c nonempty subsets. For example, given the apparently innocuous case of $n = 100$ vectors and $c = 10$ sets, there are approximately $\frac{10^{100}}{10!} \approx 3 \times 10^{93}$ possible partitions. Clearly, exhaustive search procedures are impractical.

As we shall see, given c and H, c-means is a computationally efficient way to compute P and minimize $J_{SSE}(P)$.

2 Implementing the c-means Algorithm

2.1 Algorithm Description

Given a set of input vectors, $H = \{\underline{x}_p\} \quad p = 1, 2, \ldots, n$ the c-means algorithm processes H as follows:

1. Input the desired number of clusters, c.
2. Determine the c initial exemplars for each cluster in H, denoted $\underline{\mu}_i(0)$. Often the $\underline{\mu}_i$ are randomly chosen. This is addressed in SDE2.
3. Classify each input vector from H into one of c clusters. Input vector $\underline{x}_i$ is assigned to the class (or subset H_j) of H represented by mean or exemplar $\underline{\mu}_j$ if

$$d(\underline{x}_i, \underline{\mu}_j) = \overset{min}{q} \{d(\underline{x}_i, \underline{\mu}_q)\} \tag{3}$$

   in Equation 3, d is a distance measure.
4. Recompute the estimates for the cluster means using the results of step 3.
5. If the exemplars are consistent, STOP, otherwise go to step 3.

Notice the essence of this approach is to achieve a self- consistent partitioning of the data. Choice of initial parameters (c and $\underline{\mu}_i(0)$) is still a challenging area of study.

2.2 Notes on the Algorithm Implementation

1. The above algorithm is described in a procedural or imperative framework. The significant challenge in SDE2 is to achieve a purely functional implementation.
2. Stopping of the algorithm is discussed in SDE2.

2.3 Distance

While a number of distance measures could be used, we will restrict our interest to Euclidean distance. This may be formulated as:

$$d(\underline{x}_i, \underline{x}_j) = \sqrt{(\underline{x}_i - \underline{x}_j)^T(\underline{x}_i - \underline{x}_j)} \tag{4}$$

One important observation is that if two vectors are closest in terms of the distance measure, they are also closest if the measure used is distance squared.

2.4 A Simple 2-D c-means Example and Visual Interpretation

We begin with a simple, easy to visualize, two-cluster example, and investigate the behavior of c-means with varying values of c.

2.4.1 Input Vectors

To illustrate the process, a set of 200 randomly generated 2-D training or input vectors was used. In this set, there are randomly generated Gaussian vectors from two classes with mean vectors and covariance matrices as follows:

Class 1: $\underline{\mu}_1 = \begin{pmatrix} 50 \\ 50 \end{pmatrix}$ and $\Sigma_1 = \begin{pmatrix} 100 & 0 \\ 0 & 100 \end{pmatrix}$.

Class 2: $\underline{\mu}_2 = \begin{pmatrix} -50 \\ -50 \end{pmatrix}$ and $\Sigma_1 = \begin{pmatrix} 100 & 0 \\ 0 & 100 \end{pmatrix}$.

2.4.2 Results and Means Trajectory

Figure 1 indicates the quick and direct convergence of the $c = 2$ means to the cluster centers in this example. As shown in the Figure, initial means were:

$$\underline{\mu}_1(0) = \begin{pmatrix} 31.3488 \\ 2.6328 \end{pmatrix} \text{ and } \underline{\mu}_2(0) = \begin{pmatrix} 19.2116 \\ -38.6717 \end{pmatrix}$$

and the final (converged) means are:

$$\underline{\mu}_1(2) = \begin{pmatrix} 50.855 \\ 50.866 \end{pmatrix} \text{ and } \underline{\mu}_2(2) = \begin{pmatrix} -50.679 \\ -51.600 \end{pmatrix}$$

Interestingly, each final mean represents a cluster comprised of (the nearest) 100, i.e., half of the input vector set. Only two iterations were required for convergence. The favorable results provided by this example are somewhat optimistic due to a number of factors, including well separated clusters, favorable choices of initial means and accurate knowledge of c.

We should also note that this simple graphical illustration is impractical when there are thousands of vectors of high (e.g., 100×1) dimension. SDEs 2 and 3 will be designed to accommodate vectors of arbitrary dimension.

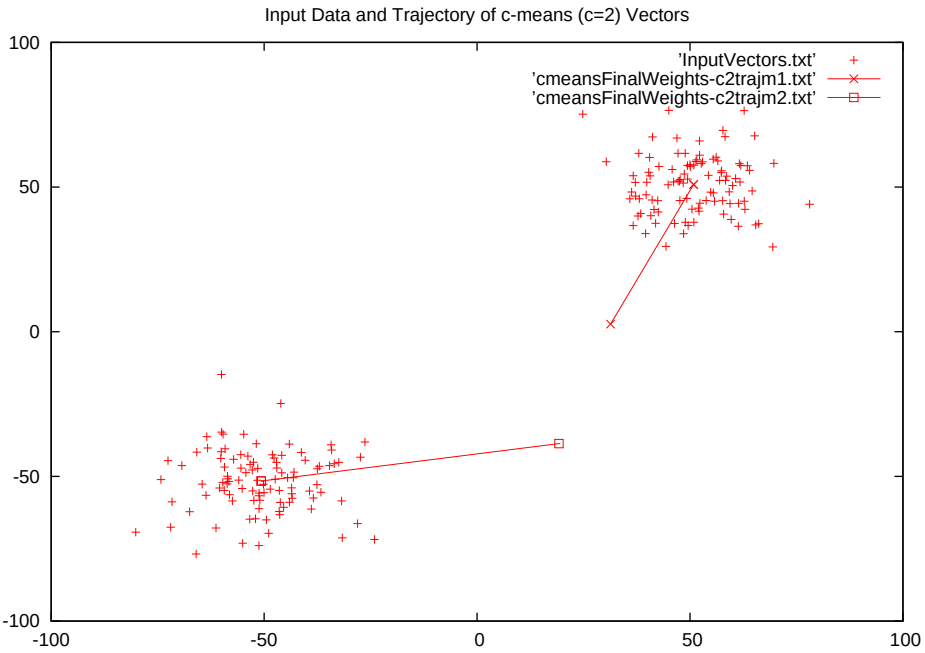

Figure 1: $c = 2$ Means Showing Initial to Final Means Trajectory. 2 Iterations Required for Convergence.