An Improved K-Means Clustering Algorithm for One Dimensional Data

Ryan Froese
Dept. of Computer Science
University of Manitoba
Winnipeg, MB, Canada
froeser5@myumanitoba.ca

James Klassen
Dept. of Computer Science
University of Manitoba
Winnipeg, MB, Canada
klass167@myumanitoba.ca

Tyler Loewen
Dept. of Computer Science
University of Manitoba
Winnipeg, MB, Canada
loewent4@myumanitoba.ca

***Abstract*—This document is a model and instructions for LaTeX. This and the IEEEtran.cls file define the components of your paper [title, text, heads, etc.]. *CRITICAL: Do Not Use Symbols, Special Characters, Footnotes, or Math in Paper Title or Abstract.**

***Index Terms*—component, formatting, style, styling, insert**

I. Introduction

This document is a model and instructions for LaTeX. Please observe the conference page limits.

II. Body

A. Algorithm Overview

B. Pseudo Code

C. Proof of Correctness

This algorithm needs a couple requirements in order to function properly:

1) There must be no duplicate cluster centroids
2) Every cluster must contain at least one item in it

Note: if a data point is the same distance from two cluster centroids, this algorithm favours the cluster with the larger centroid. This case is ignored in this proof, as the chance of this happening is vanishingly small in real world datasets due to floating point precision.

To prove correctness of the algorithm, we must prove the following properties

Property 1. *Each item in the dataset will be assigned to the cluster with the closest centroid, and*

Property 2. *The cluster borders cannot cross each other (even if the algorithm is done in parallel), as this would cause sum/count calculations to be incorrect.*

Setup

Algorithm inputs:

- $D = \{d_{11}, d_{12}, \ldots, d_{1n_1}, \ldots, d_{kn_k}\}$ where the dataset D contains elements d_{ij} where i is the cluster and j is the index within the cluster.
- k, the number of cluster where $k \geq 2$
- $C = \{c_1, \ldots, c_k\}$ where C is a set containing the centroids of each cluster where each c is unique and $c_1 < c_2 < \cdots < c_k$. This sequence is constant until the very end of the algorithm.
- $S = \{s_1, \ldots, s_k\}$ where S is a set containing the sums of data points in each cluster
- $N = \{n_1, \ldots, n_k\}$ where N is a set containing the number of data points in each cluster
- $N_{\text{sum}} = n_1 + \cdots + n_k$ where N_{sum} is the total number of data points in D

In this proof we use the concept of cluster borders separating the data set into its clusters. This is an integral concept as the algorithm leans heavily on it. In order to better visualize the intuition of the algorithm, we model the data and associated clusters as a sorted list of data points in clusters separated by cluster borders, i.e. the first border separates clusters 1 and 2, etc.. The set of all cluster borders positions is $B = b_1, \ldots, b_{k-1}$. Visually, this looks like the following diagram:

$$D = \underbrace{d_{11}, d_{12}, \ldots, d_{1x_1}}_{\substack{s_1 = d_{11} + \cdots + d_{1x_1} \\ c_1 = \frac{s_1}{x_1}}} \overbrace{\quad | \quad}^{\substack{b_1 = \\ x_1}} \underbrace{d_{21}, d_{22}, \ldots, d_{2x_2}}_{\substack{s_2 = d_{21} + \cdots + d_{2x_2} \\ c_2 = \frac{s_2}{x_2}}} \overbrace{\quad | \quad}^{\substack{b_2 = \\ x_1 + x_2}} \ldots \overbrace{\quad | \quad}^{\substack{b_{k-1} = \\ x_1 + x_2 + \cdots + x_{k-1}}} \underbrace{d_{k1}, d_{k2}, \ldots, d_{kx_k}}_{\substack{s_k = d_{k1} + \cdots + d_{kx_k} \\ c_k = \frac{s_k}{x_k}}} \tag{1}$$

Proof outline for Property 1

1.

Property 1.1. *Data points can only be closest to one of the clusters they're adjacent to or inside of, i.e.* d_{ij} *is closest to either* c_{i-1}, c_i *or* c_{i+1}

Definition 1.2. *Definition: a cluster border* b_i *is in the correct location if all data points before* b_i *are closer to* c_i *than* c_{i+1}*, and all data points after* b_i *are closer to* c_{i+1} *than* c_i*.*

Property 1.3. *An equivalent statement: If a cluster border* b_i *is in the correct location, the item* x *immediately before* b_i *satisfies* $x < (c_i + c_{i+1})/2$*, and the item* y *immediately after* b_i *satisfies* $y > (c_i + c_{i+1})/2$

Property 1.4. *Once the first* i *cluster borders have been put in the correct location, it follows that all data points before* b_i *have been assigned to the correct cluster*

Property 1.5. *Once the last cluster border* b_{k-1} *has been put in the correct location, all data points after* b_{k-1} *have been correctly assigned to the last cluster.*

Conclusion: By properties 1.4 and 1.5, once the algorithm terminates, all data points have been assigned to the correct cluster.

Proof of Property 1

Recall that:

1) $c_1 < c_2 < \cdots < c_k$
2) $d_{11} \le d12 \le \cdots \le d_{1n_1} \le \cdots \le d_{k1} \le \cdots \le d_{kn_k}$
   Due to the properties of means, we know that for any cluster i: $d_{i1} \le c_i \le din_i$

Property 1.1 We are now going to prove that data points can only be closest to one of the clusters they're adjacent to or inside of, i.e. d_{ij} is closest to either c_{i-1}, c_i, or c_{i+1}.

Given an item d_{ij} in a cluster i where j is the index of the item inside the cluster, which cluster centroids could d_{ij} conceivably be closest to? If $i = 1$, then it should make sense that d_{ij} is either closest to c_1 or c_2. It wouldn't make sense for it to be closer to c_3 (or any centroid higher than that), since $c_3 > c_2$. Putting it more formally: We want to minimize the L2 Norm in 1 dimension $|d_{1i} - c_a|$ over the variable a. By the algebraic definition of absolute values, $|d_{1i} - c3| = c_3 - d_{1i}$ because $c_3 > d_{1i}$ and $|d_{1i} - c_2| = c_2 - d_{1i}$ because $c_2 \ge d_{1i}$. Then it follows that: $|d_{1i} - c_3| > |d_{1i} - c_2| \iff c_3 - d_{1i} > c_2 - d_{1i} \iff c_3 > d_2$ which is one of our given properties.

This same argument applies to any cluster i: Data points can only be closest to one of the clusters they're adjacent to or inside of i.e. d_{ij} is closest to either c_{i-1}, c_i, or c_{i+1}

Definition 1.2 For a cluster border b_i to be in the correct position, all data points to the left of b_i are closer to ci than c_{i+1}, and all data points to the right of bi are closer to c_{i+1} than ci.

Property 1.3 For convenience, define $m_i = (c_i + c_{i+1})/2$ as the mean of two adjacent cluster centroids.

Lemma: if an arbitrary element d_{ij} is less than m_i, then it's closer to c_i than c_{i+1}. Additionally, if an arbitrary element d_{ij} is greater than m_i, then it's closer to c_{i+1} than c_i.

- If $d_{ij} \le c_i$ (and hence automatically closer to c_i than c_{i+1}):
  - Then $d_{ij} < m_i$ since $m_i > c_i$
- If $d_{ij} > c_i$:
  - Then $|d_{ij} - c_i| = d_{ij} - c_i$ and $|d_{ij} - c_{i+1}| = c_{i+1} - d_{ij}$ (since $d_{ij} < c_{i+1}$ must be true), after which it follows that: $(d_{ij} - c_i) < (c_{i+1} - d_{ij}) \iff 2d_{ij} < c_i + c_{i+1} \iff d_{ij} < (c_i + c_{i+1})/2 \iff d_{ij} < m_i$.
  - This also applies to $(d_{ij} - c_i) > (c_{i+1} - d_{ij})$:
  - $(d_{ij} - c_i) > (c_{i+1} - d_{ij}) \iff 2d_{ij} > c_i + c_{i+1} \iff d_{ij} > (c_i + c_{i+1})/2 \iff d_{ij} > m_i$

This extends without loss of generality to an arbitrary element $d_{(i+1)j}$ in cluster $i + 1$. Then by the lemma, our condition for a cluster border being in the correct position is then that the data point x immediately before the border satisfies $x < m_i$, and the data point y immediately after the border satisfies $y > m_i$, ensuring that both elements have the shortest distance to the centroid of the cluster it belongs to. Then by the transitive property, all elements before x are also closer to c_i than c_{i+1}, and all elements after y are also closer to c_{i+1} than c_i.

Property 1.4 We will now prove that once the first i cluster borders have been put in the correct location, it follows that all data points before border b_i have been assigned to the correct cluster

Our algorithm starts with the first cluster border at position b_1, moving it left/right until it's in the right place. Since every element before b_1 is closer to c_1 than c_2, and each item must be closest to either c_1 or c_2, we can conclude each item before b_1 has been assigned to the correct cluster, i.e. the first cluster.

We then move onto the second cluster border at position b_2, and adjust that to the left/right until it's in the right place. The same argument as before applies; since every item before b_2 is closer to c_2 than c_1 or c_3, and since every item before b_2 must be closest to either c_1, c_2 or c_3, everything before b_2 must be assigned to the correct cluster. We can continue this process for i cluster borders, resulting in every data point before b_i being assigned to the correct cluster.

Property 1.5 We will now prove that once the last cluster border b_{k-1} has been moved into the correct position, all data points after b_{k-1} have been assigned to the correct cluster.

Following from property 1.1 and definition 1.2, all data points after b_{k-1} must be closer to ck than ck-1, and all data points after b_{k-1} must be closer to either ck or ck-1. Therefore, once the last cluster border b_{k-1} has been moved into the correct position, all data points after b_{k-1} have been assigned to the correct cluster.

Conclusion: Therefore, by properties 1.4 and 1.5, once the algorithm terminates, all data points have been assigned to the correct cluster.

Proof outline for Property 2

2.

Property 2.1. *As a cluster border is being adjusted, it cannot cross an unadjusted border to the left or right.*

Property 2.2. *As a cluster border is being adjusted, it cannot cross an already-adjusted border to the left or right.*

Conclusion: By properties 2.1 and 2.2, cluster borders cannot cross each other, even if the f-cluster algorithm is executed in parallel.

Proof of Property 2 Property 2.1: Let us start with an unadjusted cluster border bi surrounded by unadjusted cluster borders b_{i-1} and b_{i+1}. If bi is moved to the left to get to its correct position, it cannot cross to the left of d_{i1} (the data point just to the right of b_{i-1}), since $d_{i1} \leq c_i < c_{i+1}$. This means that d_{i1} is closer to c_i than c_{i+1}, but if b_i were to cross to the left of d_{i1}, that would imply it was closer to c_{i+1}. This also means that b_i can move at most n_{i-1} data points to the left.

If b_i is instead moved to the right to get to its correct position, it cannot cross to the right of $d_{i+1n_{i+1}}$ (the data point just to the left of b_{i+1}). This is because $d_{i+1n_{i+1}} >= c_{i+1} > c_i$, meaning that $d_{i+1n_{i+1}}$ is closer to $c_i + 1$ than c_i, however if b_i crossed to the right of $d_{i+1n_{i+1}}$ that would imply it was closer to c_i than c_{i+1}. This also means that b_i can move at most $n_{i+1} - 1$ data points to the right.

Combining the fact that b_i can move at most n_{i-1} spots to the left, or move at most $n_{i+1} - 1$ spots to the right, the absolute maximum a cluster border can move in one iteration is $\max(n_{i-1}, n_{i+1} - 1)$.

Property 2.2: Let us start with an unadjusted cluster border b_i surrounded by adjusted cluster borders b_{i-1} and b_{i+1}. If b_i is moved to the left to get to its correct position, it cannot cross to the left of b_{i-1}, as that would imply that an element left of b_{i-1} was closer to c_{i+1} than c_{i-1}, when we already know that the element is closest to c_{i-1}.

If b_i is instead moved to the right to get to its correct position, it cannot cross to the right of b_{i+1}, as that would imply that an element to the right of b_{i+1} was closer to c_{i-1} than c_{i+1}, when we already know that the element is closest to c_{i+1}.

Conclusion: By properties 2.1 and 2.2, cluster borders cannot cross each other, even if the f-cluster algorithm is executed in parallel, and the maximum number of spots a cluster border b_i can be moved is $\max(n_{i-1}, n_{i+1} - 1)$