Breif Introduction to MSSC Solving Using DCA, Math 515

Ian Picklo[1]

[1] College of Engineering and Mines, The University of North Dakota

1 Introduction

Mean sum of squares clustering (MSSC) is one formulation of the clustering problem. The solution is able to group like data in so called cluster. These are useful in a variety of applications, from data science to machine learning. The difficulty is computing the solution to the MSSC formulation effectively and efficiently.

The following report follows the implementation of a DC algorithm for solving MSSC problems. Specifically, it provides an introduction to clustering of data, K-means clustering, and implementing a DCA formulation from [1] to compare with. [?]

2 Background

This section is designed to give an the numerical tools for construction optimizations for MSSC. It will first cover on of the more traditional methods before giving an overview on how DC programming may be adapted as well.

2.1 Mean Sum of Squares

Square error algorithms, such as Mean Sum of Squares (MSSC), are some of the most frequently used classical methods for clustering data of distinct sets. These usually take the form of minimizing the distance between a set of n points/entities within a c clusters to each cluster's centroid. For convenience, we will present using the notation present in [1]. Mathematically, this can be formulated as having n entities be denoted as $X := \{x_1, x_2, \ldots, x_n\} \in \mathbb{R}^d$. These are partitioned into $c (2 \leq c \leq n)$ clusters. The identification of which cluster element n is part of can be easily represented as $U = (u_{i,k}) \in \mathbb{R}^{c \times n}$ with $i = 1, ..., c$ and $k = 1, ..., n$. or as

$$u_{i,k} := \begin{cases} 1, & \text{if } x_k \in C_i, \\ 0, & \text{otherwise.} \end{cases}$$

Further, with centriods $v_i \in \mathbb{R}^d$ and the matrix $V \in \mathbb{R}^{c \times d}$ where each column is v_i, MSSC can be formulated as

$$\begin{cases} \min f(U, V) := \sum_{k=1}^{n} \sum_{i=1}^{c} u_{i,k} \left\| x_k - v_i \right\|^2, \\ \text{s.t.} \quad u_{i,k} \in \{0, 1\}, \quad i = 1, \ldots, c, \ k = 1, \ldots, n, \\ \sum_{i=1}^{c} u_{i,k} = 1, \quad k = 1, \ldots, n. \end{cases} \tag{1}$$

These formulations can be uses with several optimization algorithms such as K-means and DCA.

2.2 K-means

One of the benchmark algorithms in clustering is the K-means algorithm introduced in [2], and is time complexity of $O(n)$ [3]. One problem with this algorithm is that it is sensitive to its initialization and does not guarantee convergence to a global minimum. The basic implementation is shown in Algorithm 1. Example results shown in figure 1.

One way of looking at this algorithm is following the traditional "guess and check" approach. After one round of centroid selection, entities are assigned to each centroid. Then the centroids are selected to minimize distance between the entities in a group and gradually that reach positions where they converge.

Algorithm 1 K-Means Clustering

Require: Data points $X = \{x_1, \ldots, x_n\} \in \mathbb{R}^d$, number of clusters c, maximum iterations T

Ensure: Cluster assignments C, centroids $\{v_1, \ldots, v_c\}$

Initialize centroids $\{v_1^{(0)}, \ldots, v_c^{(0)}\}$ (e.g. random from X)
for $t = 1$ to T **do**
  Assignment step:
  for $k = 1$ to n **do**
    $u_{i,k}^{(t)} \leftarrow \arg\min_{i \in \{1,\ldots,c\}} \| x_k - v_i^{(t-1)} \|^2$
  end for
  Update step:
  for $i = 1$ to c **do**
    $v_i^{(t)} \leftarrow \dfrac{\sum_{k=1}^{n} u_{i,k}^{(t)} x_k}{\sum_{k=1}^{n} u_{i,k}^{(t)}}$
  end for
  Convergence check:
    if $\max_k \| v_k^{(t)} - v_k^{(t-1)} \| < \varepsilon$ **then break**
end for
return $U = \{u_{i,k}^{(t)}\}_{i=1,k=1}^{c,k}, V = \{v_k^{(t)}\}_{k=1}^{k}$

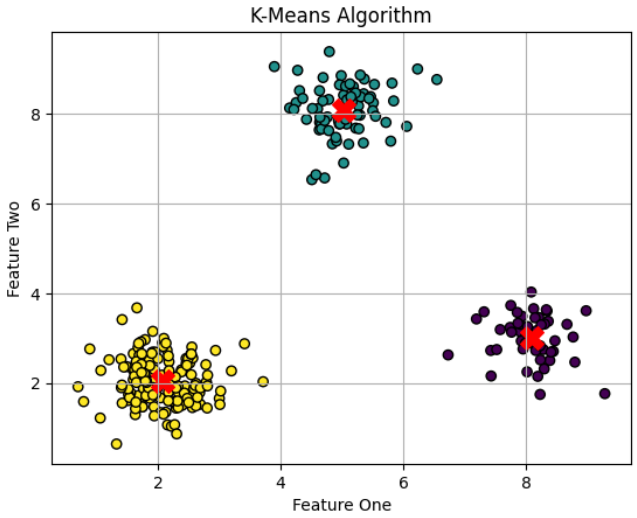

Figure 1. Example of K-Means clustering on three distinct groups.

2.3 DC Algorithms

Difference of Convex Programming describes the Optimization Problem

$$min \quad f(x) := g(x) - h(x), \quad \text{subject to } x \in R^d \tag{2}$$

where $g, h : R^d \rightarrow (-\infty, \infty]$ are convex. First introduced in [4] and expanded in many prolific works such has [5], the resulting methods for solving these problems (know as DC algorithms(DCA)) have had use in many applications in machine learning and data science. While it it outside the scope of this work to provide a complete review of DCA, there are the following conditions.

Theorem 2.1 (Necessary condition for DCA) *Suppose that $\bar{x}$ is a local minimizer of f in 2 then*

$$\partial h(\bar{x}) \subset \partial g(\bar{x}) \tag{3}$$

Theorem 2.2 (Sufficient condition for DCA) *Suppose that there exists a neighborhood $\mathcal{N}$ of $\bar{x}$ such that*

$$\varnothing \neq \partial h(x) \subset \partial g(\bar{x}) \quad \forall\, x \in \mathcal{N}. \tag{4}$$

The proofs and basic implementation of DCA for these are already covered in other works [5].

2.4 DCA for MSSC

The works adapting DCA to MSSC include [6],[7], and many others outside the scope of this report. These are specifically related to rewriting equation 1, as a difference of convex functions. [1] gives one natural adaptation. Note first that $u_{i,k} = u_{i,k}^2$ as $u_{i,k} \in \{0, 1\}$.

Using the equation $f_1^2 + f_2^2 = (f_1 + f_2)^2 - 2f_1f_2$ equation 1 can be rewritten as

$$f(U,V) = \sum_{k=1}^{n}\sum_{i=1}^{c} u_{i,k}^2 \|x_k - v_i\|^2 =$$

$$\frac{1}{2}\sum_{k=1}^{n}\sum_{i=1}^{c}(u_{i,k}^2 + \|x_k - v_i\|^2)^2 - \frac{1}{2}(u_{i,k}^4 + \|x_k - v_i\|^4).$$

Thus,

$$f(U,V) := G_1(U,V) - H_1(U,V) \tag{5}$$

where

$$G_1(U,V) = \frac{1}{2}\sum_{k=1}^{n}\sum_{i=1}^{c}(u_{i,k}^2 + \|x_k - v_i\|^2)^2 \tag{6}$$

and

$$H_1(U,V) = \frac{1}{2}\sum_{k=1}^{n}\sum_{i=1}^{c}(u_{i,k}^4 + \|x_k - v_i\|^4). \tag{7}$$

Note that each is the sum of convex functions and therefore also convex. This shows that $f(U,V)$ is a DC function. While the work [1] provides further information on how to adapt for implementation it comments, "this DC decomposition is not interesting because it requires an iterative algorithm for solving a convex program at each iteration."

3 Methods

In this section a reformation of DCA for MSSC is reviewed. It's main contribution is its convex subproblem is already known, reducing the total computation time. Additionally, some remarks about initialization processes for initial centroid selection are covered.

3.1 DCA Reformation

In [1] an alternative formulation of DCA for MSSC is provided that solves the issues of having to resolve a convex subproblem at each step. In that work, equation 2 is rewritten as

$$\min\left\{\chi_{\Delta\times C}(U,V) + \frac{\rho}{2}\left\|(U,V)\right\|^2 - H_2(U,V) : (U,V) \in \mathbb{R}^{c\times n}\times\mathbb{R}^{c\times d}\right\}. \tag{8}$$

where

$$H_2(U,V) := \frac{\rho}{2}\|(U,V)\|^2 - f(U.V), \quad G_2(U,V) := \frac{\rho}{2}\|(U,V)\|^2 \tag{9}$$

This first requires computation of the convex subprogram

$$\min\left\{\frac{\rho}{2}\left\|(U,V)\right\|^2 - \left\langle(U,V),\,(Y^l,\,Z^l)\right\rangle : (U,V) \in \Delta\times C\right\}. \tag{10}$$

where

$$\rho \geq \alpha^2 + n + n\sqrt{[\frac{1}{n}\alpha^2 + 1]^2 + \frac{16}{n}\alpha^2}] \tag{11}$$

and

$$Y^l = \left(\left(\rho u_{i,k}^2 - 2u_{i,k}^2\right)\|x_k - V_k^l\|^2 + 2tu_{i,k}^l - t\right)_{i=1,\ldots c}^{k=1,\ldots n} \tag{12}$$

$$Z^l = \left(\rho V_i^l - 2\sum_{k-=1}^{n}(V_i^l - x_k)(u_{i,k}^2)\right)_{i=1,\ldots c} \tag{13}$$

The solution is conveniently [1] provided as

$$(U^{l+1})^k = Proj_{\Delta k}\left((Y^l)^k\right) \text{ for } k = 1, \ldots n, \tag{14}$$

$$(V^{l+1})^k = Proj_{R_i}\left(\frac{1}{\rho}(Z^l)_i\right) \text{ for } i = 1, \ldots c. \tag{15}$$

$$= \quad \left\{\frac{(Z^l)_i}{\rho} \text{ if } \|(Z^l)_i\| \leq \rho r, \text{else } \frac{(Z^l)_i r}{\|(Z^l)_i\|}, (i = 1, \ldots, c)\right.$$

Essentially, equation 14 is saying that the projection of Y^k onto a simplex gives us the label assignment for the entities. While 15 is projecting the results back into the outside edge of the ball containing all realistic possibilities. This is a significant result in terms of computation time for DCA as the convex subproblem that could potential take an unknown number of iterations is removed. This is perhaps the most interesting portion of this formulation. Together, the following DCA for the MSSC can be formulated in alg. 2.

3.2 Centroid initialization

Initialization of optimization problems is always a prime issue. There are many works specifically looking at initialization procedures. For brevity, two basic initializations for an MSSC problem are going to be discussed here. The first is random initialization within the solution space. In the case of MSSC, this involves calculating the min α_i and max β_i for each dimension in $\mathbb{R}^d$. From here, the initial c centroids can be guessed randomly within the space.

Table 1
Experimental results for comparing the two clustering algorithms and initialization strategies. Results shown are for the average of 200 cycles. Each the initialization type *INIT* number of elements n, the dimension in $\mathbb{R}^d$ and the number of clusters c

Experimental Results					K-Means			DCA		
Dataset	n	c	d	*INIT*	Accuracy	Time(s)	Iterations	Accuracy	Time(s)	Iterations
VOTE	435	2	17	Random	87.23	4.11E-04	4.05	76.27	0.5978	229.41
VOTE	435	2	17	Heuristic	87.70	1.66E-03	4.19	84.56	0.5250	201.21
2D	200	2	2	Random	77.14	4.74E-04	5.06	71.12	0.3794	318.53
2D	200	2	2	Heuristic	77.11	5.69E-04	5.79	73.9	0.3116	260.89

Algorithm 2 DCA for MSSC

Require: Data points $X = \{x_1, \ldots, x_n\} \in \mathbb{R}^d$, number of clusters c, maximum iterations L, convergence threshold ϵ

Ensure: Cluster assignments C, centroids $\{v_1, \ldots, v_c\}$

Initialize centroids $\{v_1^{(0)}, \ldots, v_c^{(0)}\}$ (e.g. random from X) **Assign** X labels and store them in U
for $l = 1$ to L **do**
  Compute subgradients
  Y^l and Z^l using equations 13 and 12
  Define set
  U^l and V^l U^{l+1} and V^{l+1} using equations 15 and 14
  if $\max_k \|v_k^{(t)} - v_k^{(t-1)}\| < \varepsilon$ **then break**
end for
return $U = \{u_{i,k}^{(t)}\}_{i=1,k=1}^{c,k}, V = \{v_k^{(t)}\}_{k=1}^{k}$

Other options take a heuristic approach. Instead, c centroids are picked randomly from entities X. This can be further compounded by requiring a minimum distance d to separate the centroids. A comparison of these two approaches will be covered in the results section.

4 Experiments

The following experiments were constructed to compare the K-means algorithm 1 and the discussed DCA 2. Experiment one tests the ability of each clustering algorithm to cluster in $\mathbb{R}^2$. Experiment two used the 1984 Congressional House-Votes (VOTES) dataset [8]. In each case, the number of correctly grouped entities of the total number are reported.

4.1 Clustering in $\mathbb{R}^2$

For the following experiment, 200 points were normally distributed about 2 randomly chosen centroids in a box $[-5, 5]$. The number of points in each were randomly selected between $[50 - 150]$ at each iterations.

4.2 Party assignment

The VOTES dataset consist of 435 data points with representative voting results on 17 actions and two possible political parties. This is formulated as each data point being a size 17 array with 1, −1, and 0 representing voting yay, nay, or unassigned.

5 Discussion

In table 1 the averages of 200 cycles with random seeds 0-199 are shown and a number of interesting results can be gleamed. The first is that it would appear that in both cases a heuristic based initialization improved the accuracy and time to completing of each of DCA on each of the test datasets. I believe this has to do with preventing ill-conditioned starting centroids such as have two place too close to each other and/or away from the rest of the entities. The second is that it would appear my implementation of the DCA from [1] appears to contain some mistake as my results do not agree with [1] on the same dataset VOTE.

6 Conclusion

This paper provide a brief introduce to the MSSC problem. It gave one formulation for thinking about the minimization problem as well as some algorithms. Overall, the results showed a benefit to using heuristic style initialization. The comparison between DCA and K-Means did not show an advantage to DCA, however this may be do to some computational mistake.

References

[1] L. T. H. An and P. D. Tao, "Minimum sum-of-squares clustering by DC programming and DCA," in *Proceedings of the Intelligent computing 5th international conference on Emerging intelligent computing technology and applications*, ICIC'09, (Berlin, Heidelberg), pp. 327–340, Springer-Verlag, Sept. 2009.

[2] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, vol. 5.1, pp. 281–298, University of California Press, Jan. 1967.

[3] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: a review," *ACM Computing Surveys*, vol. 31, pp. 264–323, Sept. 1999.

[4] P. D. Tao and E. B. Souad, "Algorithms for Solving a Class of Nonconvex Optimization Problems. Methods of Subgradients," in *North-Holland Mathematics Studies* (J. B. Hiriart-Urruty, ed.), vol. 129 of *Fermat Days 85: Mathematics for Optimization*, pp. 249–271, North-Holland, Jan. 1986.

[5] P. D. Tao, "CONVEX ANALYSIS APPROACH TO D. C. PROGRAMMING: THEORY, ALGORITHMS AND APPLICATIONS," 1997.

[6] B. Ordin and A. M. Bagirov, "A heuristic algorithm for solving the minimum sum-of-squares clustering problems," 2015.

[7] F. J. A. Artacho and P. T. Vuong, "The Boosted DC Algorithm for nonsmooth functions,"

[8] "Cluster Analysis of Roll Call Votes in U.S. House."