Approximating K-Means using an ADMM Approach

Denise Cerna and Tejes Srivastava

University of California, Davis

October 12, 2021

K-Means Algorithm

K-Means is a clustering algorithm that tries to find the centroids of clusters using an iterative method that eventually converges to actual centroid of the clusters.

The K-Means algorithm is the following:

1. Randomly choose the centers of the k clusters that lies in the domain of the set of points.
2. Assign each point in the set to a cluster, depending on how close it is to one of the randomly chosen centers.
3. For each of the k clusters, calculate the new center.
4. Repeat step 2 and 3 until convergence is met.

K-Means

We would like to formalize this problem as an optimization problem:

Consider a set of points $S = \{(x_1^i, x_2^i, \ldots, x_n^i) : i \in \{0, 1, \ldots, n\}\}$. Then we would like to divide S into clusters of $(S_1, \ldots, S_k)$ each with centers at $(c_1, c_2, \ldots c_k)$. Then we wish to minimize the function

$$f(S, S) = \sum_{j=1}^{k} \sum_{i=1}^{|S_j|} \left|\left| s_i^j - c_j \right|\right|^2$$

which describes the total sum of squared Euclidean distances from each point x_i to its centroid c_i

K-Means

A different way to model the problem would be to let

$$X = [x_{ij}] \in \mathbb{R}^{n \times k}$$

Such that

$$x_{ij} = \begin{cases} 1 & \text{if } s_i \text{ is in cluster } S_j \\ 0 & \text{if if } s_i \text{ is not in cluster } S_j \end{cases}$$

K-Means

In this case, the cluster centroids can be rewritten as

$$c_j = \frac{\sum_{l=1}^{n} x_{lj} s_l}{\sum_{l=1}^{n} x_{lj}}$$

and the MSSC (minimum sum of squared Euclidean distance) is represented as the following optimization problem:

1. $\min \sum_{j=1}^{k} \sum_{i=1}^{n} x_{ij} \left|\left| s_i - \frac{\sum_{l=1}^{n} x_{lj} s_l}{\sum_{l=1}^{n} x_{lj}} \right|\right|^2$
2. s.t. $\sum_{j=1}^{k} x_{ij} = 1$ for $(i = 1, ..., n)$.
3. $\sum_{i=1}^{n} x_{ij} \geq 1$ for $(j = 1, ..., k)$.
4. $x_{ij} \in 0, 1 (i = 1, ..., n; j = 1, ..., k)$

Problems with K-Means

The issue that arises when trying to solve K-Means is that the algorithm that has been mentioned is NP-Hard, or in other words, inefficient to solve. Therefore, it is more viable to use approximations of the classical K-Means algorithm which are more efficient, and can be solved in polynomial time. One such solution is by approximating the K-Means problem as a semidefinite programming problem (SDP), which can be solved efficiently.

SDP

A standard SDP problem is of the following form

$$\begin{cases} \min \ \ \mathrm{Tr}(WZ) \\ \text{s.t.} \ \ \mathrm{Tr}(B_i Z) = b_i \text{ for } i = 1, \ldots, k \\ \qquad Z \succcurlyeq 0 \end{cases} \tag{1}$$

0-1 SDP

Alternatively, a 0-1 SDP problem has slightly different constraints and takes the following form

$$\begin{cases} \min \ \ \mathrm{Tr}(WZ) \\ \text{s.t.} \ \ \mathrm{Tr}(B_i Z) = b_i \text{ for } i = 1, \ldots, k \\ \qquad Z^2 = Z, Z = Z^T \end{cases} \tag{2}$$

meaning that Z is a symmetric matrix and its elements can only be 0 and 1

Equivalence of Classical K-Means to SDP

Recall the objective function given previously

$$f(S,S) = \sum_{j=1}^{k}\sum_{i=1}^{n} x_{ij}\left|\left|s_i - \frac{\sum_{l=1}^{n} x_{lj}s_l}{\sum_{l=1}^{n} x_{lj}}\right|\right|^2$$

Then we expand this term and yield

$$f(S,S) = \sum_{i=1}^{n} ||s_i||^2 \sum_{j=1}^{k} x_{ij} - \sum_{j=1}^{k} \frac{||\sum_{i=1}^{n} x_{ij}s_i||^2}{\sum_{i=1}^{n} x_{ij}}$$

Let W_S be a matrix that denotes a matrix with the ith row being s_i^T. So, we have that the matrix $W_S W_S^T$ gives us the norms of each point s_i in each of the diagonal entries, so the trace of this matrix yields the norm of the point. Hence, we have

$$f(S,S) = \mathrm{Tr}(W_S W_s^T) - \sum_{j=1}^{k} \frac{||\sum_{i=1}^{n} x_{ij}s_i||^2}{\sum_{i=1}^{n} x_{ij}}$$

Equivalence of Classical K-Means to SDP

Since X is an assignment matrix, we have that the rows depict which points belong to which cluster, so every row contains only one 1. On the other hand, for each column there is at least 1 since the columns can be thought of clusters. It follows that since X has elements that are either 0 or 1,

$$X^T X = \text{diag}\left(\sum_{i=1}^{n} x_{i1}^2, \ldots, \sum_{i=1}^{n} x_{ik}^2\right) = \text{diag}\left(\sum_{i=1}^{n} x_{i1}, \ldots, \sum_{i=1}^{n} x_{ik}\right).$$

Now, let

$$Z = [z_{ij}] = X(X^T X)^{-1} X^T$$

then we have that

$$\begin{aligned} f(S, S) &= \text{Tr}(W_S W_s^T) - \sum_{j=1}^{k} \frac{||\sum_{i=1}^{n} x_{ij} s_i||^2}{\sum_{i=1}^{n} x_{ij}} \\ &= \text{Tr}(W_S W_s^T) - \text{Tr}(W_S W_s^T Z) \\ &= \text{Tr}(W_S W_s^T (I - Z)) \end{aligned}$$

Equivalence of Classical K-Means to SDP

Recall that there is a constraint in our original optimization problem:

$$\sum_{j=1}^{k} x_{ij} = 1$$

We can reformulate this by writing that

$$Xe^k = e^n$$

where e^k is the vector with only 1's (same with e^n). It follows

$$Ze^n = ZXe^k = X(X^TX)^{-1}X^TXe^k = Xe^k = e^n.$$

So, we have a constraint given by $Ze = e$.

Equivalence of Classical K-Means to SDP

It can be shown that in Z the term

$$\frac{\sum_{i=1}^{n} x_{ij}^2}{\sum_{i=1}^{n} x_{ij}}$$

appears in each of the diagonal entries. So,

$$\begin{aligned} \mathrm{Tr}(Z) &= \sum_{j=1}^{k} \frac{\sum_{i=1}^{n} x_{ij}^2}{\sum_{i=1}^{n} x_{ij}} \\ &= \sum_{j=1}^{k} 1 \\ &= k \end{aligned}$$

Equivalence of Classical K-Means to SDP

Finally, we are left with the following 0-1 SDP problem to solve:

$$\begin{cases} \min \ \ \mathrm{Tr}(W(I - Z)) \\ \text{s.t.} \quad Ze = e, Tr(Z) = k \\ \qquad\quad Z \geq 0, Z^2 = Z, Z = Z^T \end{cases} \tag{3}$$

Reformulating K-Means to Manifold Optimization

Let us consider another relaxation of the K-Means problem given by

$$\begin{cases} \min \ \mathrm{Tr}(DYY^T) + \lambda ||Y_-||_F^2 \\ \text{s.t.} \ \ Y \in M \end{cases} \tag{4}$$

We let

$$M = \{Y \in \mathbb{R}^{n \times k} : Y^T Y = I_k, YY^T 1 = 1\}.$$

Y_- denotes a matrix from the manifold M which only contains the negative entries and all the positive entries are simply set to 0. Note that $||Y_-||_F$ denotes the Frobenius norm.

Solving K-Means Manifold Optimization

The following is the algorithm to solve the relaxation of K-Means in (4)

1. Set $\lambda_0 \to 0$
2. $Y_{n+1} \leftarrow$ Gradient Descent(f_λ) Initialized at Y_n
3. $\lambda_{n+1} \leftarrow 2\lambda_n + 1$
4. Repeat steps 2-3 until $||Y_-||_F < \epsilon$

Issues for Manifold Optimization

The above algorithm may seem simplistic and easy to implement, implying that the given problem is ideal for solving. However, note that this solves a relaxation of the problem, without a way to generalize back to the original K-Means problem. Moreover, the above algorithm was not provided an assured guarantee to converge because Gradient Descent will not necessarily remain in M.

Alternating Direction Method of Multipliers (ADMM)

This is a particular class of problems that are well studied in convex optimization.

$$\begin{aligned} \text{minimize} \quad & f(x) + g(z) \\ \text{subject to} \quad & Ax + Bz = C \end{aligned}$$

We define the Langrangian to be the following function:

$$L_\rho(x, z, y) = f(x) + g(z) + y^T(Ax + Bz - C) + \frac{\rho}{2}||Ax + Bz - C||_2^2$$

Alternating Direction Method of Multipliers (ADMM)

The algorithm proposed to solve ADMM Problems is of the following form:

$$x^{k+1} = \text{argmin}_x L_\rho(x, z^k, y^k)$$
$$z^{k+1} = \text{argmin}_x L_\rho(x^{k+1}, z^k, y^k)$$
$$y^{k+1} = y^k + \rho(Ax^{k+1} + Bz^{k+1} - C)$$

ADMM Manifold Optimization

In order to address these aforementioned problems with manifold optimization, we formulate a new relaxation using ADMM.

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\text{Tr}(DXX^T) + I_+(Y) \\ \text{subject to} \quad & X = Y, Y \in M \end{aligned}$$

where $\mathcal{M} = \{Y \in \mathbb{R}^{x \times k} : Y^T Y = I_k, YY^T 1 = 1\}$. I_+ is the positive indicator function, given by

$$I_+(Y) = \begin{cases} \infty, & \text{if } Y \text{ contains any negative values} \\ 0, & \text{else} \end{cases} \tag{5}$$

Algorithm to solve ADMM Manifold Optimization

The algorithm used to solve this problem is given by

$$X^{k+1} = \text{Retr}_{X_k}(-\eta_k \text{Proj}_{T_{X^k}\mathcal{M}}(2DX + \Lambda^k + \rho(X^k - Z^k)) \tag{6}$$

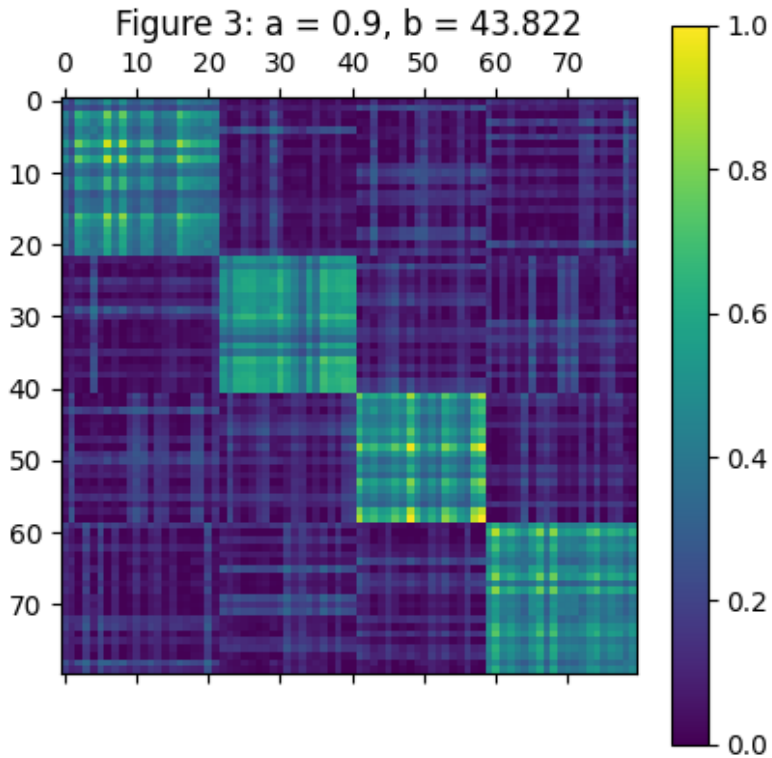
$$Y^{k+1} = \max\left\{X^{k+1} + \frac{1}{\rho}\Lambda^k, 0\right\} \tag{7}$$

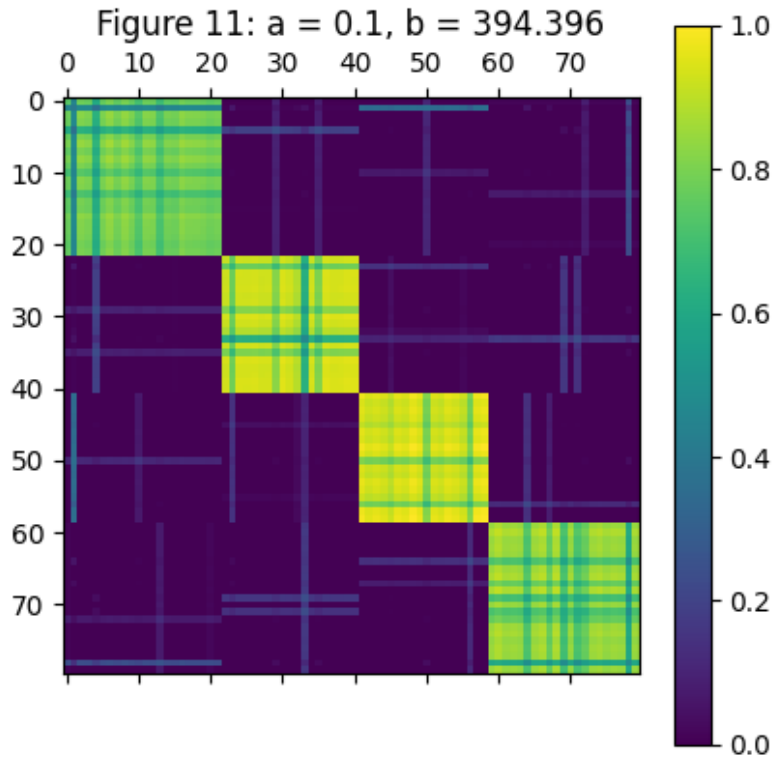
$$Z^{k+1} = \frac{1}{1 + \frac{1}{\gamma} + \rho}\left(\frac{1}{\gamma}Y^{k+1} + \Lambda^k + \rho X^{k+1}\right) \tag{8}$$

$$\Lambda^{k+1} = \Lambda^k + \rho(X^{k+1} - Z^{k+1}) \tag{9}$$

Numerical Experiments

First we ran the original gradient descent algorithm using a synthetic data set of 80 points and 4 clusters:

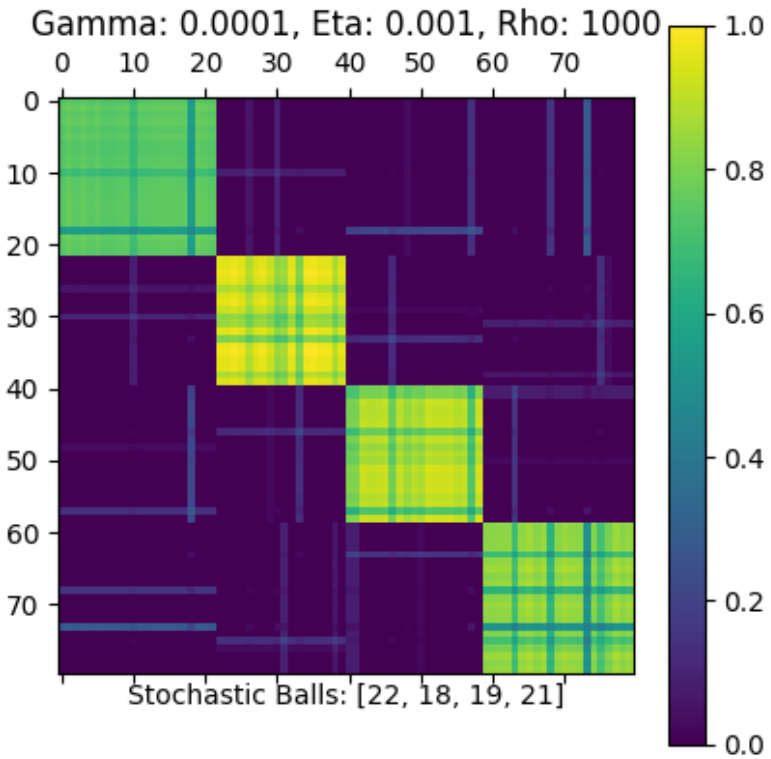




Minimization of $a\mathrm{Tr}(DYY^T) + b||Y_-||^2$

Numerical Experiments

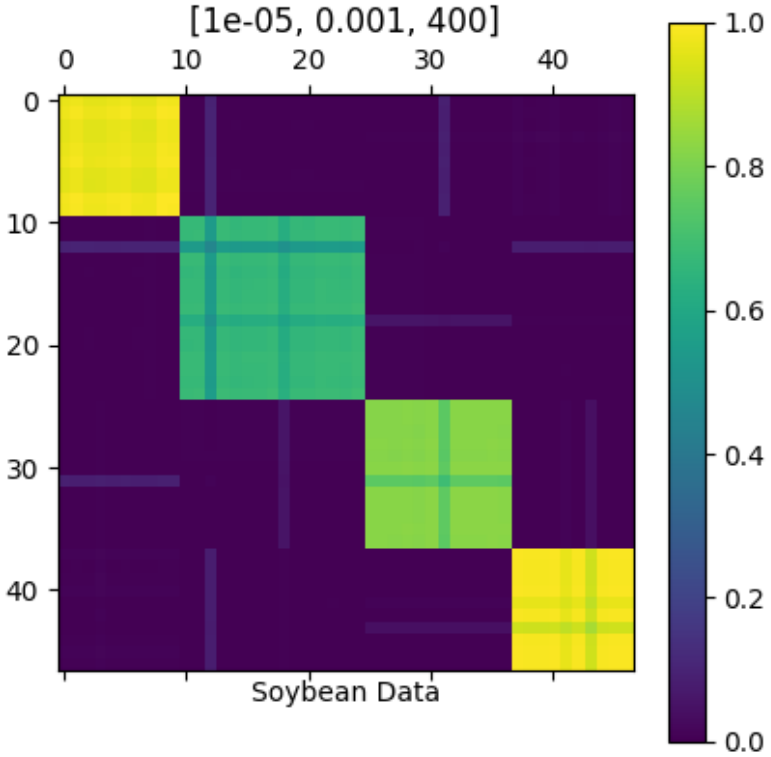
Next, we considered the ADMM Algorithm for different data sets.



Stochastic Balls: 80 points, 4 planted clusters

NMI = 1.0

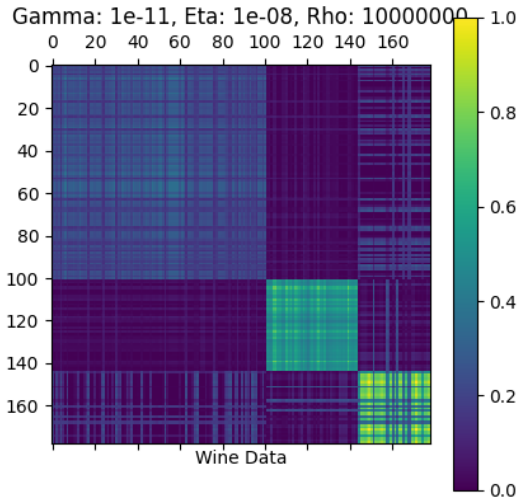
Numerical Experiments

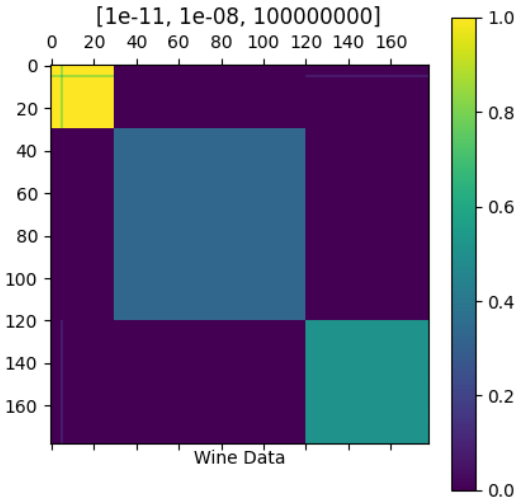


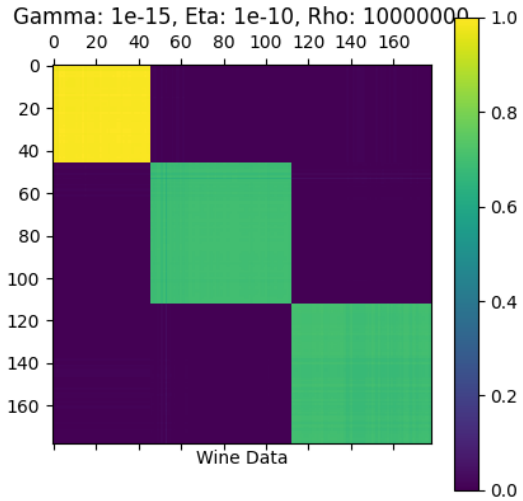
Soybean Small-Data: 46 points, 4 clusters

NMI = 0.718

Numerical Experiments







Wine Data: 178 points, 3 clusters

The NMI values from left to right are 0.461, 0.365, 0.011

Conclusions

Challenges when testing different parameter values

- Block structure appears despite low NMI scores
- Parameters leading to singular matrices
- Results varied upon initialization
- Larger data was more unstable

Applictions to Future Work

- Identifying a more objective way to compare similar results
- Exploring new initialization methods
- Determining a proper terminal condition ofr the algorithm
- Considering a relaxation for the indicator function

References

T. Carson, D. G. Mixon and S. Villar, *Manifold Optimization for K-means Clustering*, 2017 International Conference on Sampling Theory and Applications (SampTA), 2017, pp. 73-77, doi: 10.1109/SAMPTA.2017.8024388.

Peng, Jiming and Wei, Yu. (2007). *Approximating K-means-type Clustering via Semidefinite Programming*. SIAM Journal on Optimization. 18. 186-205. 10.1137/050641983.

Thank You! Any Questions?

https://github.com/tejess/manifold-optimization