Convex Is All You Need In Clustering

Gu, Yilin
The Chinese University of Hong Kong, Shenzhen
School of Management and Economics (SME)
221025012@link.cuhk.edu.cn

Jin, Xin
The Chinese University of Hong Kong, Shenzhen
School of Management and Economics (SME)
221025013@link.cuhk.edu.cn

Abstract

Convex clustering, a convex method in ameliorating K-means clustering and hierarchical clustering, greatly improves the performances for traditional non-convex clustering methods. To solve the instabilities and difficulties of non-convex clustering, we address a few convex objective functions (including smooth and non-smooth) and solve with Accelerated Gradient Method (AGM), CG-Newton Method, Proximal Gradient Method (PGM), Alternating Direction Method of Multipliers (ADMM) and Stochastic Gradient Method (SGM). Compared all the convex clustering methods with traditional K-means clustering method, we ultimately obtain a better convergence and accuracy result from convex clustering.

1 Introduction

Clustering is a fundamental unsupervised learning problem and aims to assign observations into clusters such that observations in the same group are similar to each other. Traditional clustering methods such as K-means clustering, hierarchical clustering, and Gaussian mixture models apply greedy searching method and suffer from instabilities due to their non-convex optimization formulations. Also, traditional non-convex clustering models require prior knowledge about the number of clusters which is not available in many applications. To overcome the instability issues of these traditional clustering methods, a clustering algorithm based on optimization methods, *Convex Clustering*, has been proposed (Hocking et al., 2011; Lindsten et al., 2011; Pelckmans et al., 2005).

For the basic idea of convex clustering, suppose we have a sample data matrix $S_{n\times p} = \{s_1, s_2, ..., s_n\}$ with n observations and p features, we can construct two basic target objectives in solving clustering problem and figure out the clusters we need. Assume that initially each data point corresponds to a cluster c_i. Then, the first target is that, we need to minimize the distances between clusters and all the sample points to make data points cluster.

$$\min \ \frac{1}{2}\Sigma_{i=1}^{n}||c_i - s_i||^2 \tag{1}$$

Given we need to specify all the data points into several clusters and not let each data point categorize into a cluster to increase the complexity of algorithm, we can assign data points s_i and s_j into the same cluster if the *convex* distance of the corresponding clusters of 2 data points $||c_i^* - c_j^*|| \leq \varepsilon$. And the second target is that, when we minimize $||c_i^* - c_j^*||$ according to the tolerance ε that we set, we can obtain the optimal number of clusters and optimal cluster choices based on minimization (suppose the distance applies p norm, $p = 1, 2, ..., \infty$).

$$\min \ \Sigma_{i=1}^{n}\Sigma_{j=i+1}^{n}||c_i - c_j||_p \tag{2}$$

To solve both two objective functions together, we can add a regularization term γ on the Equation (2) and sum up Equation (1) and (2) with regularization term.

$$\min \ \frac{1}{2}\Sigma_{i=1}^{n}||c_i - s_i||^2 + \gamma\Sigma_{i=1}^{n}\Sigma_{j=i+1}^{n}||c_i - c_j||_p \tag{3}$$

Nevertheless, given Equation (2) in Equation (3) is non-smooth, there are basically two approaches to solve Equation (3). The first approach is to transform Equation (2) into a smooth term (e.g. Huber norm, $l2$ norm square, etc.) so that gradient methods and Newton Methods can be used; the second approach is to calculate the proximity operator of Equation (2) to generate a new direction which can be used in Proximal Gradient Method (PGM) or Proximal Newton Method (PNM) with Armijo condition. With proximity operator, another method we can apply is Alternating Direction Method of Multipliers (ADMM), in which we can withdraw the non-smooth term as an epigraph term, and apply ADMM on this constraint problem.

$$\min \ \frac{1}{2}\Sigma_{i=1}^{n}||c_i - s_i||^2 + \gamma\Sigma_{i=1}^{n}t_i \tag{4}$$

$$s.t. \quad \begin{cases} \sum_{j=i+1}^{n} ||c_i - c_j||_p \leq t_i \\ epi||c||_p := \{(c,t) \in R^n \times R : ||c||_p \leq t\} \end{cases} \tag{5}$$

Moreover, we can also add a weight on the Equation (2) so that we can focus more on the points with large distance in which we add a larger weight on it. Also, we can solve the Equation (6) with PGM and PNM.

$$min \ \frac{1}{2}\Sigma_{i=1}^{n}||c_i - s_i||^2 + \gamma\Sigma_{i=1}^{n}\Sigma_{j=i+1}^{n}w_{ij}||c_i - c_j||_p \tag{6}$$

In this paper, we mainly solve these approaches with Accelerated Gradient Method (AGM), CG-Newton Method, Proximal Gradient Method (PGM), Alternating Direction Method of Multipliers (ADMM) and Stochastic Gradient Method (SGM) and compare all these results with traditional K-means clustering method.

2 Literature Review

A related paper on convex clustering is efficient implementations proposed by Chi and Lange (2015) and its extension to convex biclustering has been developed by Chi et al. (2016). Two efficient algorithms PGM and ADMM are introduced while they are mainly designed for clustering low-dimensional data. In order to address high dimensionality problem, one key ingredient of our sparse convex clustering method is a regularization penalty built upon their PGM and ADMM algorithms to encourage clusters. As will be shown in experimental studies, such regularization step is able to significantly improve the clustering accuracy in clustering problems compared with K-means Clustering.

Another line of research focuses on simultaneous clustering and feature selection. Some approaches are model-based clustering methods, such as Raftery and Dean (2006), Pan and Shen (2007), Wang and Zhu (2008), Xie et al. (2010), and Guo et al. (2010). In contrast, some approaches are model-free, such as Witten and Tibshirani (2010), Sun et al. (2012), and Wang et al. (2013). One common building block of these clustering approaches is the usage of a lasso type penalty for feature selection. For example, Witten and Tibshirani (2010) developed a unified framework for feature selection in clustering using the lasso penalty (Tibshirani, 1996). Sun et al. (2012) proposed a sparse K-means using the group-lasso penalty (Yuan and Lin, 2006). We refer readers to Alelyani et al. (2013) for a thorough overview. In spite of their good numeric performance, these sparse clustering procedures still suffer from instabilities due to the non-convex optimization formulations. To overcome it, our constructions on objective function solves a convex optimization problem with AGM, CG-Newton, PGM, ADMM and SGM which ensure a unique global solution.

3 Convex Clustering

For the rest part of this paper, section 3.1 will briefly introduce how we prepare the data. With respect to section 3.2, 3.3, 3.4 and 3.5, we'll mainly introduce and analyze the AGM, CG-Newton algorithm on Equation (3) with Huber norm in section 3.2; PGM will be applied to solve Equation (3) with non-smooth term in section 3.3; With the proximity operator obtained from section 3.3, we'll apply it into ADMM method in section 3.4; ultimately, we choose $m(m < n)$ sample points from n points and apply SGM on these sample data points.

3.1 Data Preparation

In this part, we mainly concern data generation and data preparation for our optimization problems. We prepare and discuss two categories of data: self-generated data points in two-dimensions and datasets derived from real-world large-scale applications. With respect to self-generated data points, we use `np.random.normal` in python numpy to generate 3 different datasets (pictures of data points can be shown as follows).

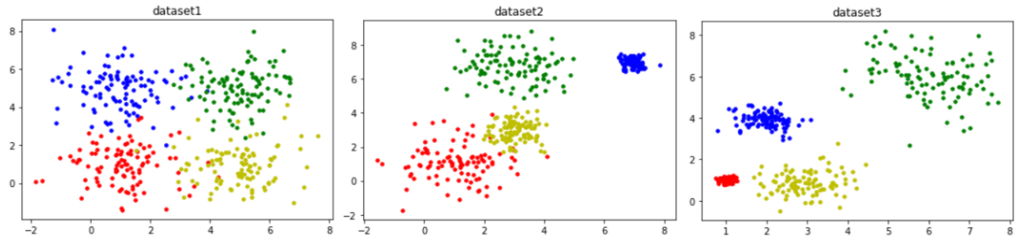

Figure 1: Self-generated Data Points

With regards to the real-world large-scale dataset, we download `wine.mat` from website `https://www.csie.ntu.edu.tw/cjlin/libsvm/` to make analysis.

3.2 Huber Norm: AGM and CG-Newton's Method

To transform the p-norm in Equation (3) to a term that can calculate the gradient and hessian, we consider $l2$ norm square and Huber norm to make transformation. In comparison, huber norm has a better result than $l2$ norm square, so we ultimately take Huber norm as transformation term. After we apply Huber norm, the objective function can be written as follows,

$$\min \frac{1}{2}\Sigma_{i=1}^{n}||c_i - s_i||^2 + \gamma\Sigma_{i=1}^{n}\Sigma_{j=i+1}^{n}\varphi_{hub}(c_i - c_j) \tag{7}$$

While the Huber norm and its gradient can be written as follows,

$$\varphi_{hub}(y) = \begin{cases} \frac{1}{2\delta}||y||^2 & if||y||^2 \leq \delta \\ ||y|| - \frac{\delta}{2} & if||y||^2 > \delta \end{cases} \tag{8}$$

$$\nabla\varphi_{hub}(y) = \begin{cases} \frac{y}{\delta} & if||y||^2 \leq \delta \\ \frac{y}{||y||} & if||y||^2 > \delta \end{cases} \tag{9}$$

According to Equation (8) and (9), we first apply AGM algorithm (because especially for the large scale data in the real world, speed is essential) with the following parameters: $\alpha_k = \frac{1}{L}$, $\beta_k = \frac{t_{k-1}-1}{t_k}$, $t_k = \frac{1}{2}(1 + \sqrt{1 + 4t_{k-1}^2})$, and set $t_{-1} = t_0 = 1$. Also, we calculate the Lipschitz constant L for Equation (7),

$$L = 1 + n\frac{\gamma}{\delta} \tag{10}$$

According to Equation (7), (8), (9), (10), we write AGM algorithm with `python`, the main codes shown as follows,

```
def acc_gradient_method_huber(f, f_grad, x0, tol):
    start=time.time()
    x_minus_1=x0
    t_minus_1=1
    x=[x_minus_1,x0]
    t=[t_minus_1]


```

```
    while np.abs(np.linalg.norm(f_grad(x[-1]))>=tol) and len(x)<
    max_iter:
        alphak = 1/
        if len(x) !=2:
            tk = (1/2)*(1+(1+4*(t[-1])**2)**(1/2))
        else:
            tk = t_minus_1 #t(-1)=t0
        t.append(tk)#add tk into t list
        betak = (t[-2]-1)/t[-1]# here t[-1] means tk for tk has been
    added.
        y = x[-1]+betak*(x[-1]-x[-2])
        x.append(y-alphak*f_grad(y))

    end=time.time()
    duration=end-start
    return {'converge_point':x[-1],'iter_num':len(x)-1,'time':duration
    }
```

Listing 1: AGM algorithm

And we can obtain the convergence plot and result shown as follows,

Table 1: Results of Accelerated Gradient Method with Backtracking

dataset	tol	iter	time
dataset_1	10^{-3}	101	5.5s
dataset_2	10^{-3}	232	10.3s
dataset_3	10^{-3}	136	7.1s
wine	10^{-3}	558	23.7s

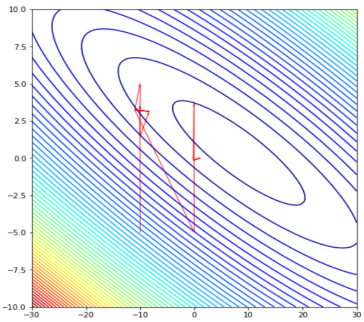

Figure 2: Convergence Plot for AGM Algorithm

According to the convergence result in Figure 2 and results in Table 1, we find that although AGM does not converge well under tol = 10^{-3}, in such tolerance, it reaches tol $\leq 10^{-3}$ in a fast speed in all the 4 datasets. Also, we use the same method to obtain the convergence plot and results from CG-Newton Method.

Table 2: Results of CG-Newton Method

dataset	tol	iter	time
dataset_1	10^{-3}	57	2.3s
dataset_2	10^{-3}	89	4.3s
dataset_3	10^{-3}	78	3.7s
wine	10^{-3}	228	13.2s

Ultimately, with CG-Newton Method, we obtain the final clustering results shown as follows.

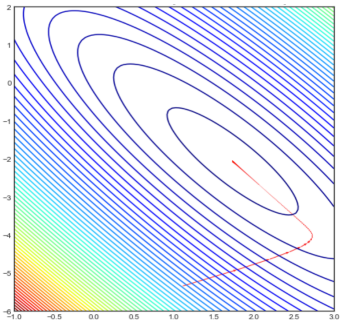

Figure 3: Convergence Plot for CG-Newton Method

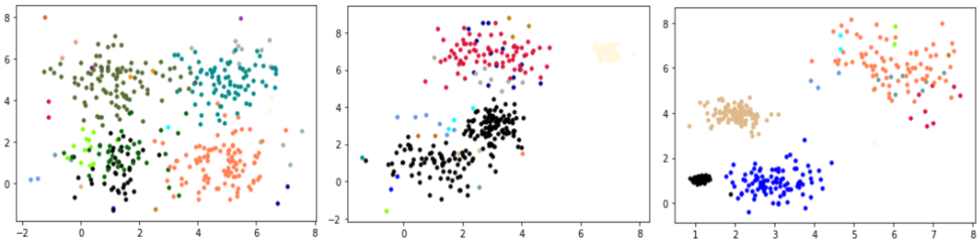

Figure 4: Results for CG-Newton Method

3.3 Non-smooth Term: Proximal Gradient Method

With respect to Proximal Gradient Method (PGM), we choose the $l1$ norm and add weights on different distances. To make a higher weight added on a larger distance, we can generally put a non-zero weight only for a pair of points which are nearby each other, and we write our weights as the follows (ρ is a set in which it contains the groups of (i, j), while i and j derive from the data point of s_i and its k-nearest neighbors s_j),

$$w_{ij} = \begin{cases} e^{-\theta||s_i - s_j||^2} & if(i,j) \in \rho \\ 0 & otherwise \end{cases} \tag{11}$$

According to the weights proposed, we then calculate and obtain the proximity function for the $l1$ norm term as function $f(.)$ in Equation (6), ($\sigma > 0$ and norm $\tau(.)$, $C^k = c_i^k - c_j^k$

$$prox_{\sigma,\tau}(C^k - \lambda\nabla f(C^k)) - C^k = arg\min[\sigma\tau(v) + \frac{1}{2}||C^k - \lambda\nabla f(C^k) - v||_1^2] \tag{12}$$

In Equation (12), we also calculate v within proximity operator,

$$v = arg\min\frac{1}{2}||v - (c_i - c_j - v^{-1}\lambda)||_1^2 + \frac{\gamma w}{v}||v||_2 \tag{13}$$

With Equation (13), we write our main python codes as follows,

```
import warnings
import numpy as np
from scipy import optimize
from scipy import linalg

def fmin_cgprox(f, f_prime, g_prox, x0, rtol=1e-6,
                maxiter=1000, verbose=0, default_step_size=1.):

    xk = x0
    fk_old = np.inf

```

```
    fk, grad_fk = f(xk), f_prime(xk)
    success = False
    for it in range(maxiter):
        # .. step 1 ..
        # Find suitable step size
        step_size = default_step_size  # initial guess
        grad_fk = f_prime(xk)
        while True:  # adjust step size
            xk_grad = xk - step_size * grad_fk
            prx = g_prox(xk_grad, step_size)
            Gt = (xk - prx) / step_size
            lhand = f(xk - step_size * Gt)
            rhand = fk - step_size * grad_fk.dot(Gt) + \
                (0.5 * step_size) * Gt.dot(Gt)
            if lhand <= rhand:
                # step size found
                break
            else:
                # backtrack, reduce step size
                step_size *= .5

        xk -= step_size * Gt
        fk_old = fk
        fk, grad_fk = f(xk), f_prime(xk)

        if verbose > 1:
            print("Iteration %s, Error: %s" % (it, linalg.norm(Gt)))

        if np.abs(fk_old - fk) / fk < rtol:
            if verbose:
                print("Achieved relative tolerance at iteration %s" %
it)
                success = True
            break
    else:
        warnings.warn(
            "fmin_cgprox did not reach the desired tolerance level",
            RuntimeWarning)

    return optimize.OptimizeResult(
        x=xk, success=success, fun=fk, jac=grad_fk, nit=it)
```

Listing 2: PGM algorithm

So, with Equation (13), we can calculate the new direction for the gradient method with Armijo condition. We'll compare the final result of PGM and ADMM in Section 3.4.

3.4 Non-smooth Term: Alternating Direction Method of Multipliers

This subsection discusses two efficient optimization approaches to solve the convex clustering by adopting a similar computational strategy used in Chi and Lange (2015). Our aproach is based on the alternating direction method of multipliers (ADMM) algorithm (Boyd et al., 2011; Gabay and Mercier, 1976; Glowinski and Marroco, 1975). So, firstly, we rewrite our function to create constraints with epigraph as follows,

$$\min \ \frac{1}{2}\Sigma_{i=1}^{n}||c_i - s_i||^2 + \gamma\Sigma_{i=1}^{n} w_i t_i \tag{14}$$

$$s.t. \quad \begin{cases} \sum_{j=i+1}^{n} ||c_i - c_j||_p \leq t_i \\ epi||c||_p := \{(c,t) \in R^n \times R : ||c||_p \leq t\} \end{cases} \tag{15}$$

In the 2-block structure, our ADMM structure for Equation (14) and (15) can be written as,

$$c^{k+1} = arg\min_{c} L_p(c, t^k, y^k) \tag{16}$$

$$t^{k+1} = arg\min_{t} L_p(c^{k+1}, t, y^k) \tag{17}$$

$$y^{k+1} = y^k + \rho(c^{k+1} + t^{k+1}) \tag{18}$$

With Equation (13), we update c, t and y in Equation (16), (17), (18) for each iteration. And finally we obtain the results for PGM and ADMM algorithm shown in Table 3 and Table 4.

It seems that Proximal Gradient Method performs better than the Alternating Direction Method of Multipliers, but we consider it might exist some errors on the process of calculation or coding because it spend too many iterations and too much time. For the time constraint, we'll check this process with more cautious and try to solve it in the future.

Table 3: Results of Proximal Gradient Method

dataset	tol	iter	time
dataset_1	10^{-3}	457	212.4s
dataset_2	10^{-3}	689	435.6s
dataset_3	10^{-3}	578	323.8s
wine	10^{-3}	-	-

Table 4: Results of Alternating Direction Method of Multipliers

dataset	tol	iter	time
dataset_1	10^{-3}	778	533.7s
dataset_2	10^{-3}	-	-
dataset_3	10^{-3}	963	1628.4s
wine	10^{-3}	-	-

3.5 Huber Norm: Stochastic Gradient Method

Ultimately, we choose Stochastic Gradient Method (SGM), in which we choose m sample points from total n $(m < n)$ points to minimize the objective function. With respect to this method, we basically apply AGM as our baseline and compare the results of SGM-AGM and SGM. The results are shown as follows in Table 5.

Table 5: Results of Stochastic Gradient Method

dataset	tol	iter	time
dataset_1	10^{-3}	36	1.3s
dataset_2	10^{-3}	57	2.2s
dataset_3	10^{-3}	39	1.3s
wine	10^{-3}	88	2.5s

Although the speed seems well in the SGM, the convergence and accuracy is not good as those in AGM.

4 Acknowledgment

We want to show our deep gratefulness and express our gratitude to Prof. Andre Milzarek and Instructor Xuanye. Without their descent help in this semester, we are not able to finish this work right now. Although we only have 2 members in our group, we pay all of our efforts possible to finish this work. Really wish you will feel enjoyable after reading this work, thank you!

References

[1] Hocking, T. D., Joulin, A., Bach, F., and Vert, J.-P. (2011). Clusterpath: An Algorithm for Clustering using Convex Fusion Penalties. *Proceedings of the 28th International Conference on Machine Learning (ICML)*.

[2] Lindsten, F., Ohlsson, H., and Ljung, L. (2011). Clustering sum-of-norms regularization: With application to particle filter output computation. In *Statistical Signal Processing Workshop (SSP)*, pages 201–204. IEEE.

[3] Pelckmans, K., De Brabanter, J., Suykens, J., and De Moor, B. (2005). Convex clutering shrinkage. In *PASCAL Workshop on Statistics and Optimization of Clustering Workshop*.

[4] Chi, E. C. and Lange, K. (2015). Splitting Methods for Convex Clustering. *Journal of Computational and Graphical Statistics*, 24(4):994–1013.

[5] Chi, E. C., Allen, G. I., and Baraniuk, R. G. (2016). Convex biclustering. *Biometrics*.

[6] Raftery, A. and Dean, N. (2006). Variable selection for model-based clustering. *Journal of the American Statistical Association*, 101:168–178.

[7] Pan, W. and Shen, X. (2007). Penalized model-based clustering with application to variable selection. *Journal of Machine Learning Research*, 8:1145–1164.

[8] Wang, S. and Zhu, J. (2008). Variable selection for model-based high-dimensional clustering and its application to microarray data. *Biometrics*, 64:440–448.

[9] Xie, B., Pan, W., and Shen, X. (2010). Variable selection in penalized model-based clustering via regularization on grouped parameters. *Biometrics*, 64:921–930.

[10] Guo, J., Levina, E., Michailidis, G., and Zhu, J. (2010). Pairwise variable selection for high-dimensional model-based clustering. *Biometrics*, 66:793–8

[11] Sun, W., Wang, J., and Fang, Y. (2012). Regularized k-means clustering of high-dimensional data and its asymptotic consistency. *Electronic Journal of Statistics*, 6(April 2011):148–167.

[12] Wang, Y., Fang, Y., and Wang, J. (2013). Sparse optimal discriminant clustering. *Statistics and Computing*, pages 1–11.

[13] Witten, D. and Tibshirani, R. (2010). A framework for feature selection in clustering. *Journal of the American Statistical Association*, 105:713–726.

[14] Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society*, Series B, 58:267–288.

[15] Yuan, M. and Lin, Y. (2006). Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67.

[16] Alelyani, S., Tang, J., and Liu, H. (2013). Feature slection for clustering: review. In *In Data Clustering: Algorithms and Applications. Edited by: Charu A, Chandan* R. CRC Press.