Introduction to Optimization

Final Project Report

HAO HAO
Workshop

Contents

Sequence of the presentation

- Introduction
- Data Preparation
- Methodology
- Implementation
- Evaluation
- Conclusion
- Future Work

Introduction

Introduction

Classification Problem

Clustering, one of the most fundamental problems in unsupervised learning, has been widely applied in various fields. However, traditional clustering models may not perform well because of three drawbacks of these models:

1、Non-convexity

2、Initialization

3、Number of clusters

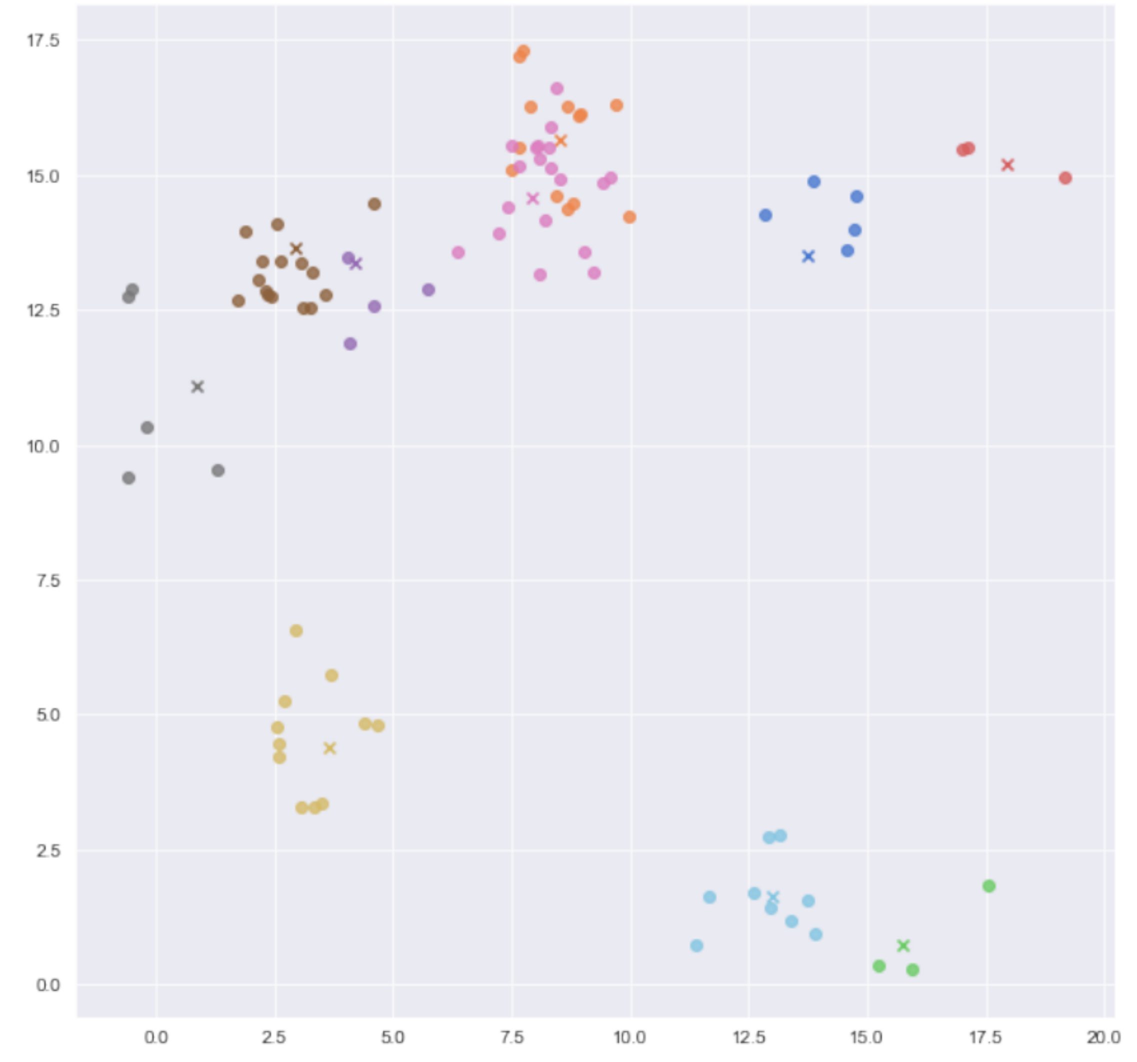
So in our report, we try to realize a convex clustering algorithm to avoid these drawbacks. We tried AGM, Newton-CG, BFGS, L-BFGS and Barzilai-Borwein Method in the original and weighted loss function.

Data Preparation

Data Preparation

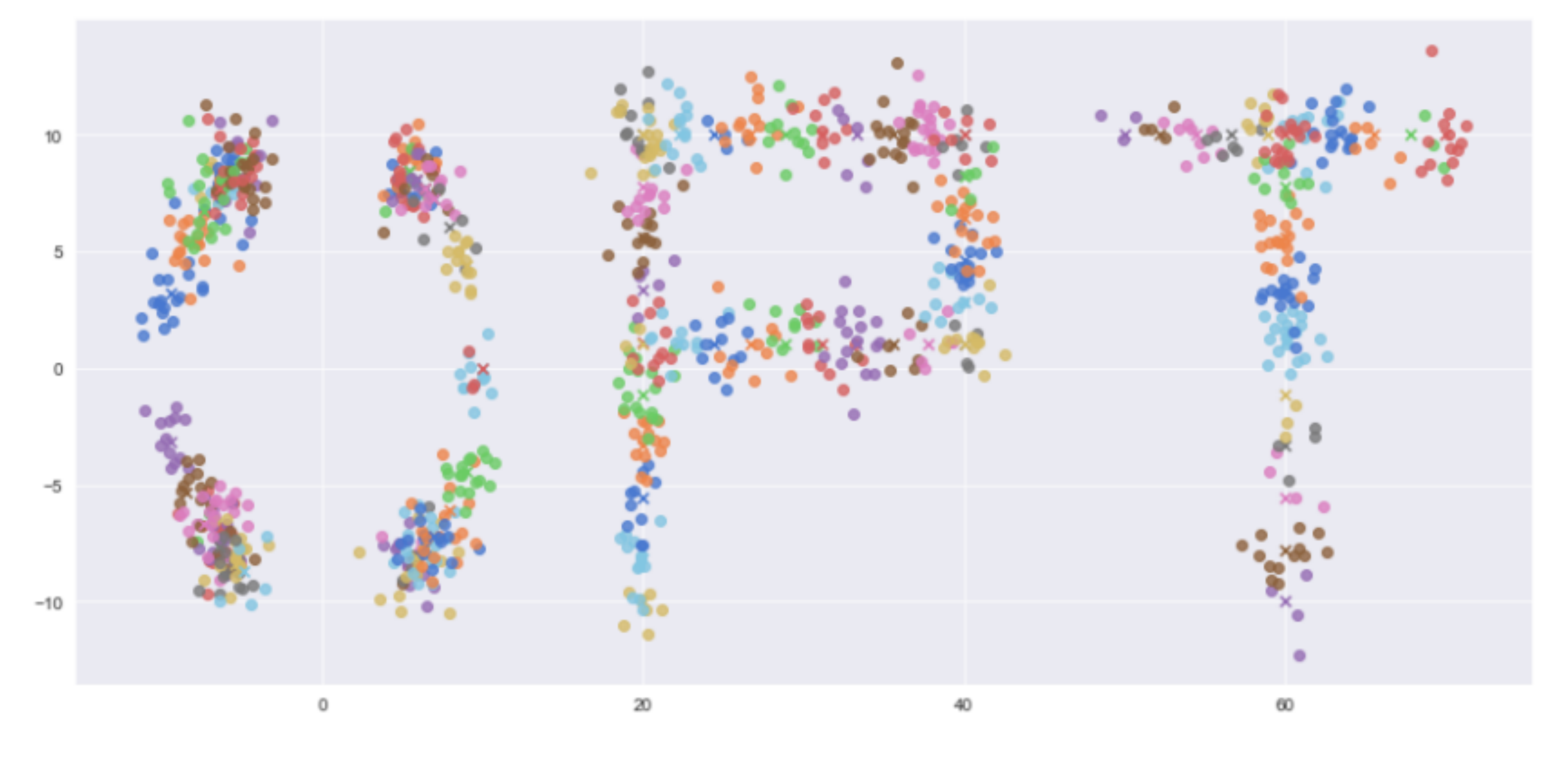
Self generated and real world data

Naive figure



Num of A: 87
Num of Centroids: 10

OPT figure



Num of A: 104
Num of Centroids: 1056

Real world data have higher dimensions and are difficult to plot without PCA.

data set	d	n	classes	description
wine	13	178	3	This data is the result of a chemical analysis of wines grown in the same region in Italy but derived from three different types of wine. The analysis determined the quantities of 13 components found in each of the wines.
vowel	10	528	11	–
segment	19	2310	7	–
mnist	784	60000	10	The MNIST database consists of 60000 handwritten digits. The digits have been normalized and centered in fixed-size 28×28 images.

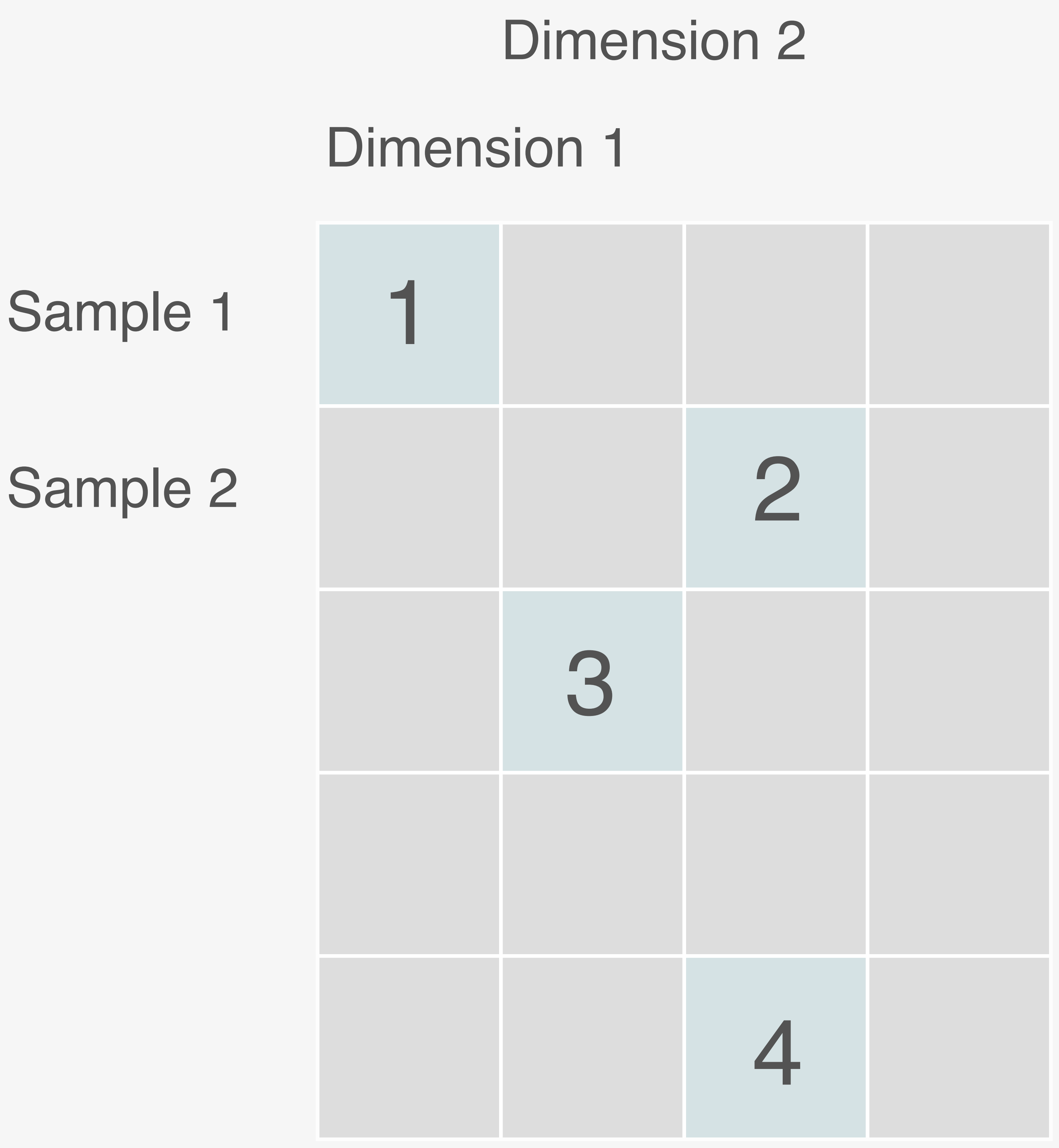
Data Preparation
Storage

We use as much sparse matrix as possible to store the data and intermediate matrix.

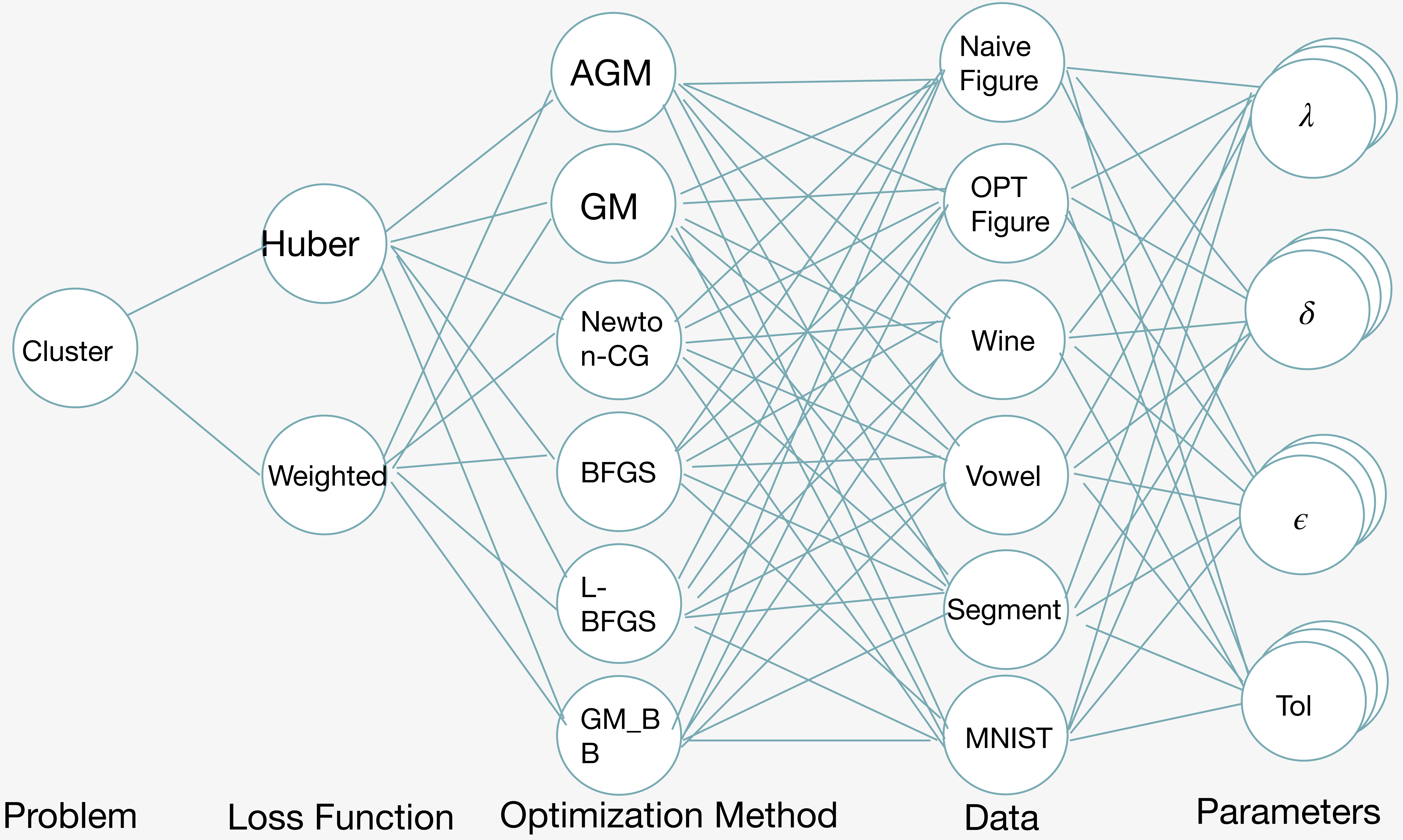
We will introduce our own B and W matrices later, which are also sparse.

However, while computing, the gradient matrix, which is the same shape as the original data, is not sparse anymore.

We use n x d as the shape. (Which is different from the project description)



General Approach



Cluster
Huber
Weighted
AGM
GM
Newto
n-CG
BFGS
L-
BFGS
GM_B
B
Naive
Figure
OPT
Figure
Wine
Vowel
Segment
MNIST
λ
δ
ϵ
Tol
Problem
Loss Function
Optimization Method
Data
Parameters

Methodology

Mathematical Analysis

Math format

Our Problem

$$\min_{X\in\mathbb{R}^{d\times n}} f_{\text{clust}}\,(X) := \frac{1}{2}\sum_{i=1}^{n} \parallel x_i - a_i \parallel^2 + \lambda \sum_{i=1}^{n} \sum_{j=i+1}^{n} \parallel x_i - x_j \parallel_p$$

take Huber-norm as an example

$$\min_{X\in\mathbb{R}^{d\times n}} f_{clust}(X) := \frac{1}{2}\sum_{i=1}^{n} \parallel x_i - a_i \parallel^2 + \lambda \sum_{i=1}^{n} \sum_{j=i+1}^{n} \varphi_{hub}\left(x_i - x_j\right)$$

Gradient

Some notations

- The gradient of x_k is

$$\nabla f_{clust}(x_k) := x_k - a_k + \sum_{j=k+1}^{n} \nabla \varphi_{hub}\left(x_k - x_j\right) - \sum_{i=1}^{k} \nabla \varphi_{hub}\left(x_i - x_k\right)$$

- The gradient of X is (use matrix notation)

$$\nabla f_{clust}(X) = X - A + \lambda B^T \nabla \varphi_{hub}(BX)$$

- What is the magic B?

Gradient

Some notations

$$\nabla f_{clust}(x_k) := x_k - a_k + \sum_{j=k+1}^{n} \lambda \nabla \varphi_{hub}\left(x_k - x_j\right) - \sum_{i=1}^{k} \lambda \nabla \varphi_{hub}\left(x_i - x_k\right)$$

$$\nabla f_{clust}(X) = X - A + \lambda B^T \nabla \varphi_{hub}(BX)$$

$$B = \begin{pmatrix} 1 & -1 & 0 & 0 & \cdots & 0 & 0 \\ 1 & 0 & -1 & 0 & \cdots & 0 & 0 \\ 1 & 0 & 0 & -1 & \cdots & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 1 & -1 & 0 & \cdots & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & 0 \\ 0 & 0 & 0 & 0 & \cdots & 1 & -1 \end{pmatrix}_{\frac{n(n-1)}{2} \times n} \qquad BX = \begin{pmatrix} X_1 - X_2 \\ X_1 - X_3 \\ \cdots \\ X_2 - X_3 \\ \cdots \\ X_3 - X_4 \\ \cdots \\ X_{n-1} - X_n \end{pmatrix}_{\frac{n(n-1)}{2} \times d}$$

$$B^T \nabla \varphi_{hub}(BX) = \begin{array}{r} F_1(X) \\ F_2(X) \\ F_3(X) \\ \cdots \\ \cdots \end{array} \begin{pmatrix} \nabla \varphi_{hub}(X_1 - X_2) + \nabla \varphi_{hub}(X_1 - X_3) + \cdots + \nabla \varphi_{hub}(X_1 - X_n) \\ -\nabla \varphi_{hub}(X_1 - X_2) + \nabla \varphi_{hub}(X_2 - X_3) + \cdots + \nabla \varphi_{hub}(X_2 - X_n) \\ -\nabla \varphi_{hub}(X_1 - X_3) - \nabla \varphi_{hub}(X_2 - X_3) + \cdots + \nabla \varphi_{hub}(X_3 - X_n) \\ \cdots \\ \cdots \end{pmatrix}_{n \times d}$$

- B is sparse and only need to calculate once for a size n

Hessian

$$\begin{pmatrix} \sum_{i=2}^{n} \nabla^2 \varphi_{hub}(X_1 - X_i) & -\nabla^2 \varphi_{hub}(X_1 - X_2) & -\nabla^2 \varphi_{hub}(X_1 - X_3) & \cdots & -\nabla^2 \varphi_{hub}(X_1 - X_n) \\ -\nabla^2 \varphi_{hub}(X_1 - X_2) & \sum_{i \in \{1,3,\cdots,n\}}^{n} \nabla^2 \varphi_{hub}(X_2 - X_i) & -\nabla^2 \varphi_{hub}(X_2 - X_3) & \cdots & -\nabla^2 \varphi_{hub}(X_2 - X_n) \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \end{pmatrix}_{nd \times nd}$$

AGM & GM

AGM & GM

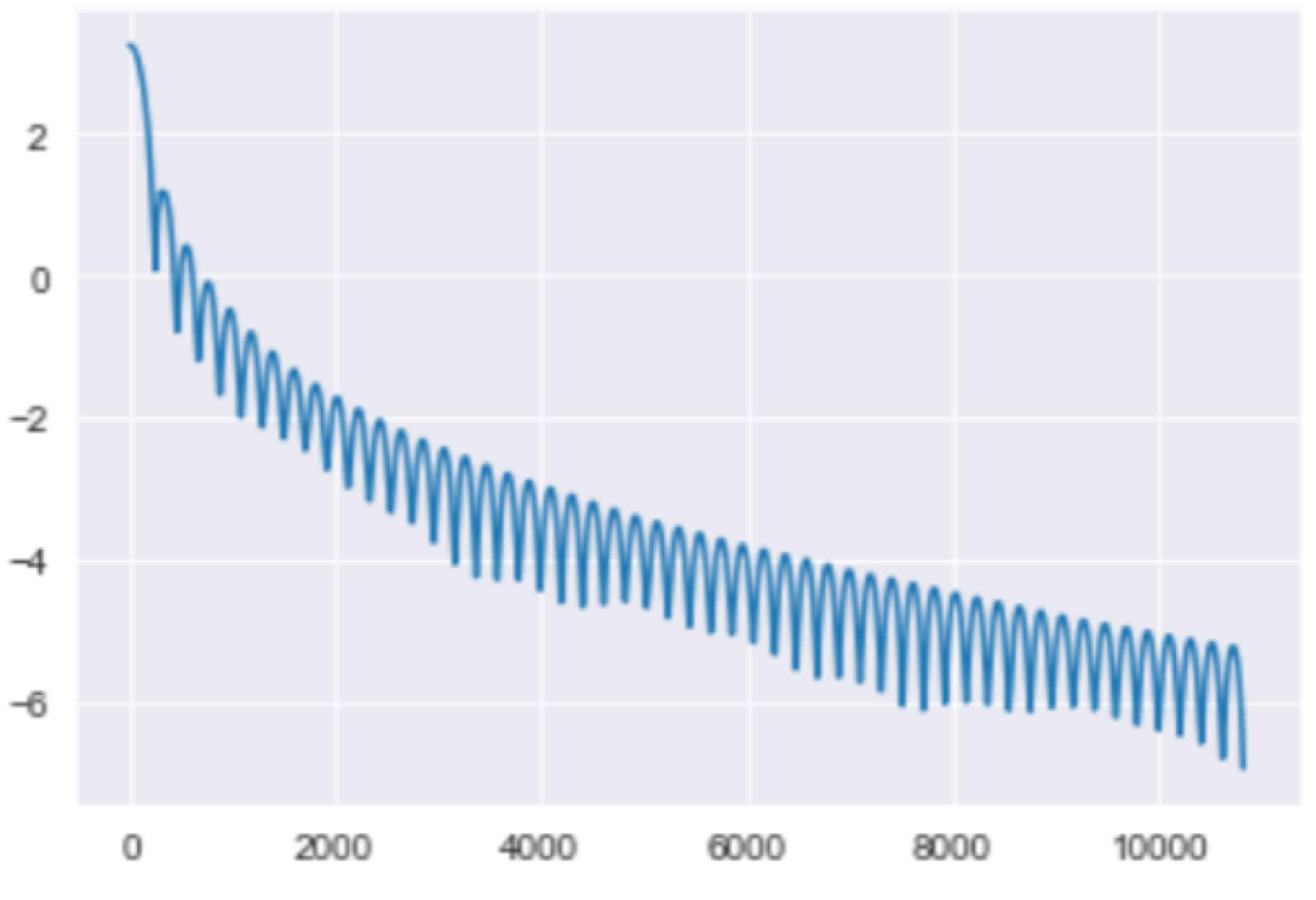


Figure 4: The convergence rate of AGM in 2D data set

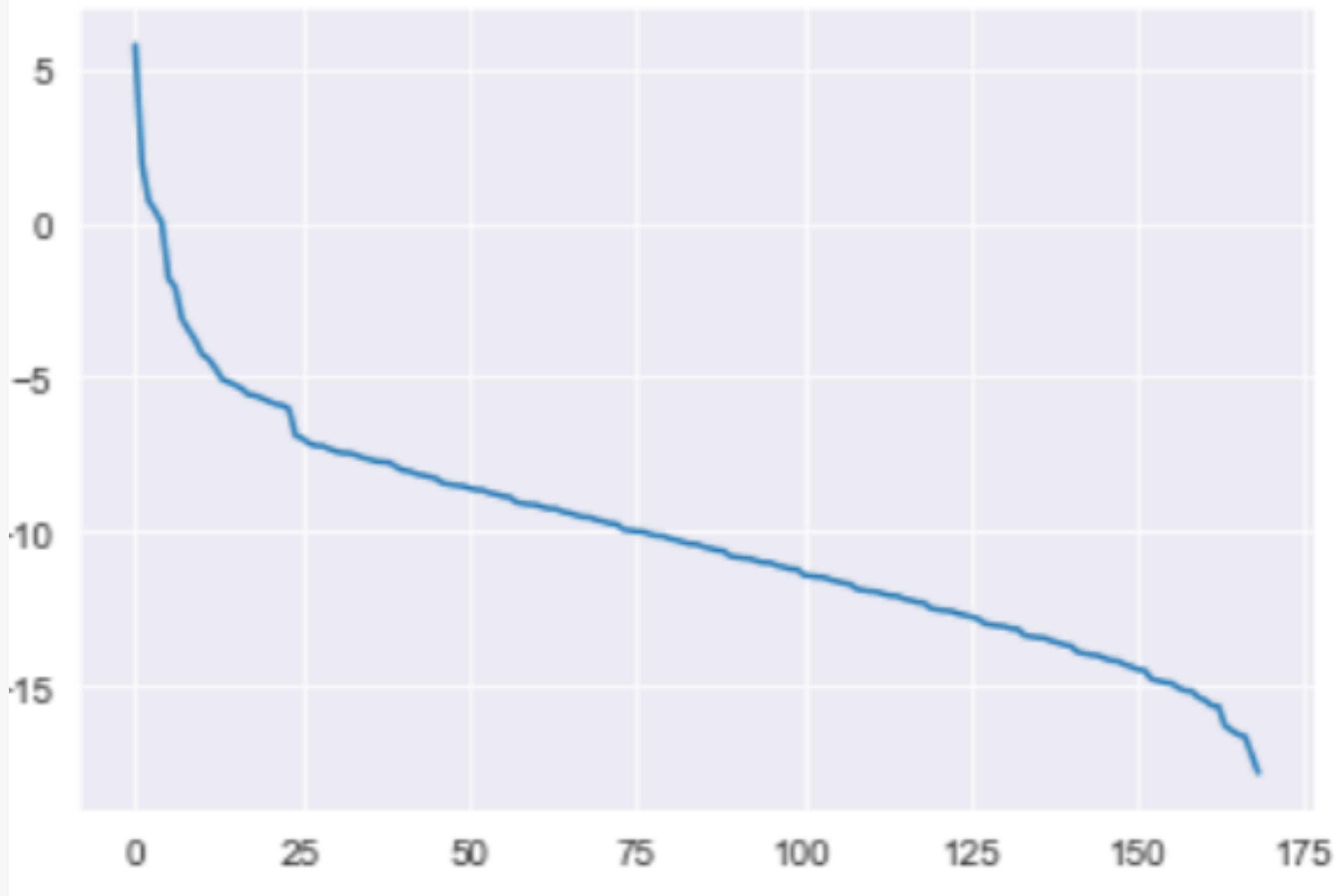


Figure 5: The convergence rate of GM in 2d data set

Newton-CG

Newton-CG

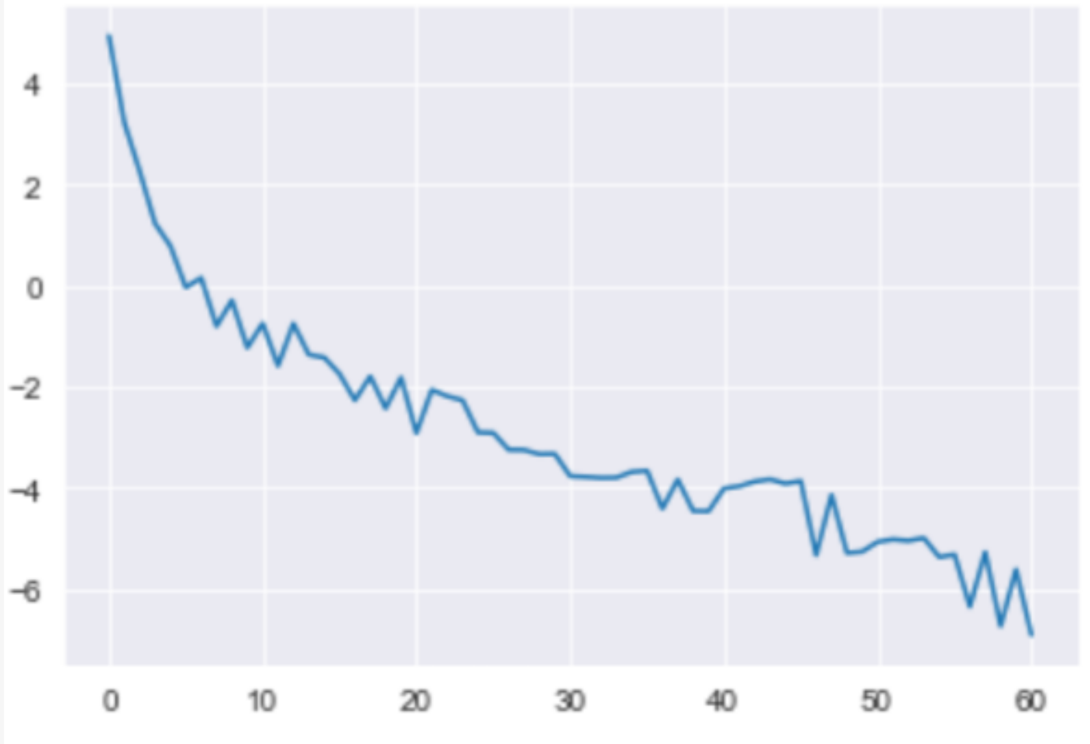


Figure 6: The convergence rate of Newton-CG in 2D data set

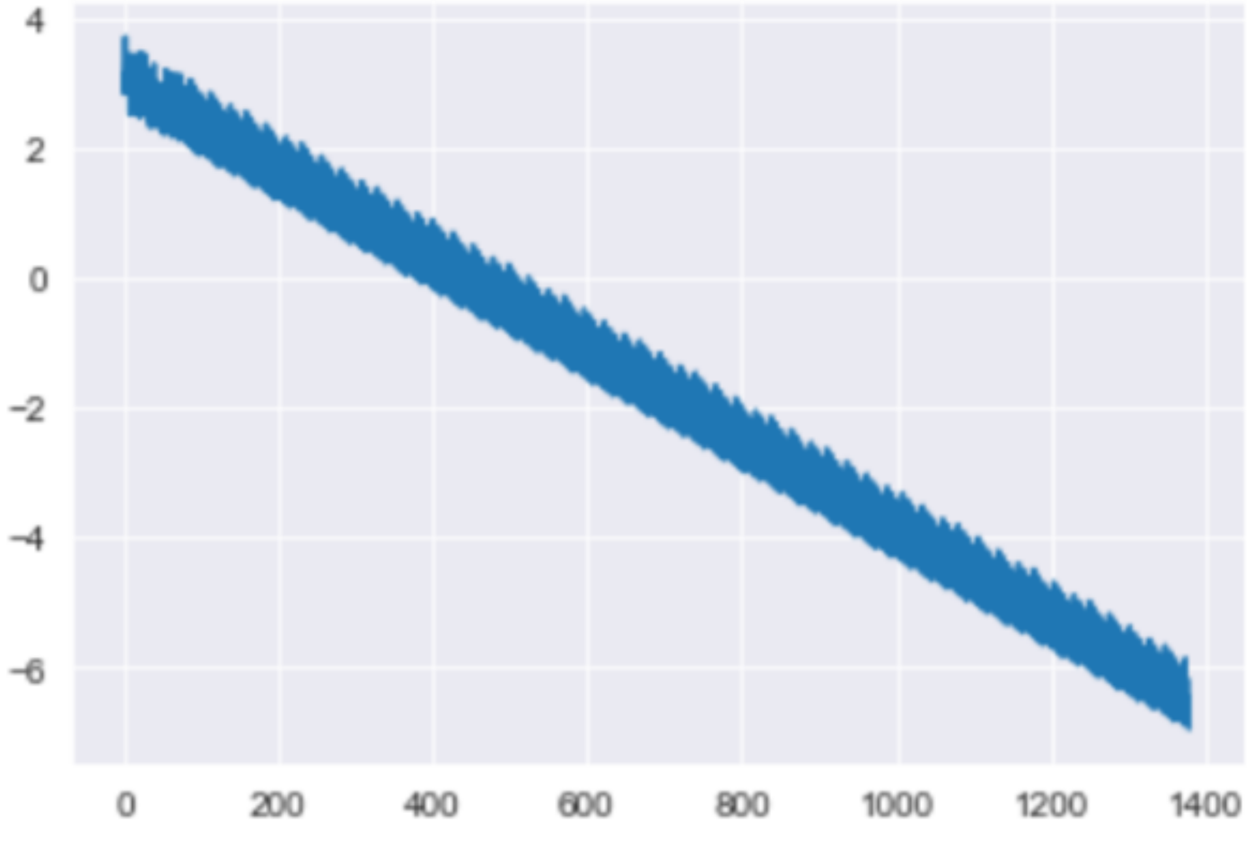


Figure 7: The value of loss function of Newton-CG

Algorithm 1 : Newton-CG

set Ap=Hessian$*p$, $v_0 = 0$, r_0 =gradient, $p_0 = -r_0$
for $j = 0, 1 \ldots n$ **do**
  if $P_k^T * Ap <= 0$ **then**
    return $d_k = v_j$
  else
    $\sigma_j = \frac{||r_j||^2}{P_k^T * Ap}$;
    $v_{j1} = v_j + \sigma_j * p_j$;
    $r_{j1} = r_j + \sigma_j * Ap$;
  end if
  if $||r_j||^2 <= tol$ **then**
    return $d_k = v_j$
  else
    $\beta_j = \frac{||r_{j1}||^2}{||r_j||^2}$;
    $p_{j1} = -r_{j1} + \beta_j * p_j$;
  end if
end for

GM_BB

GM_BB

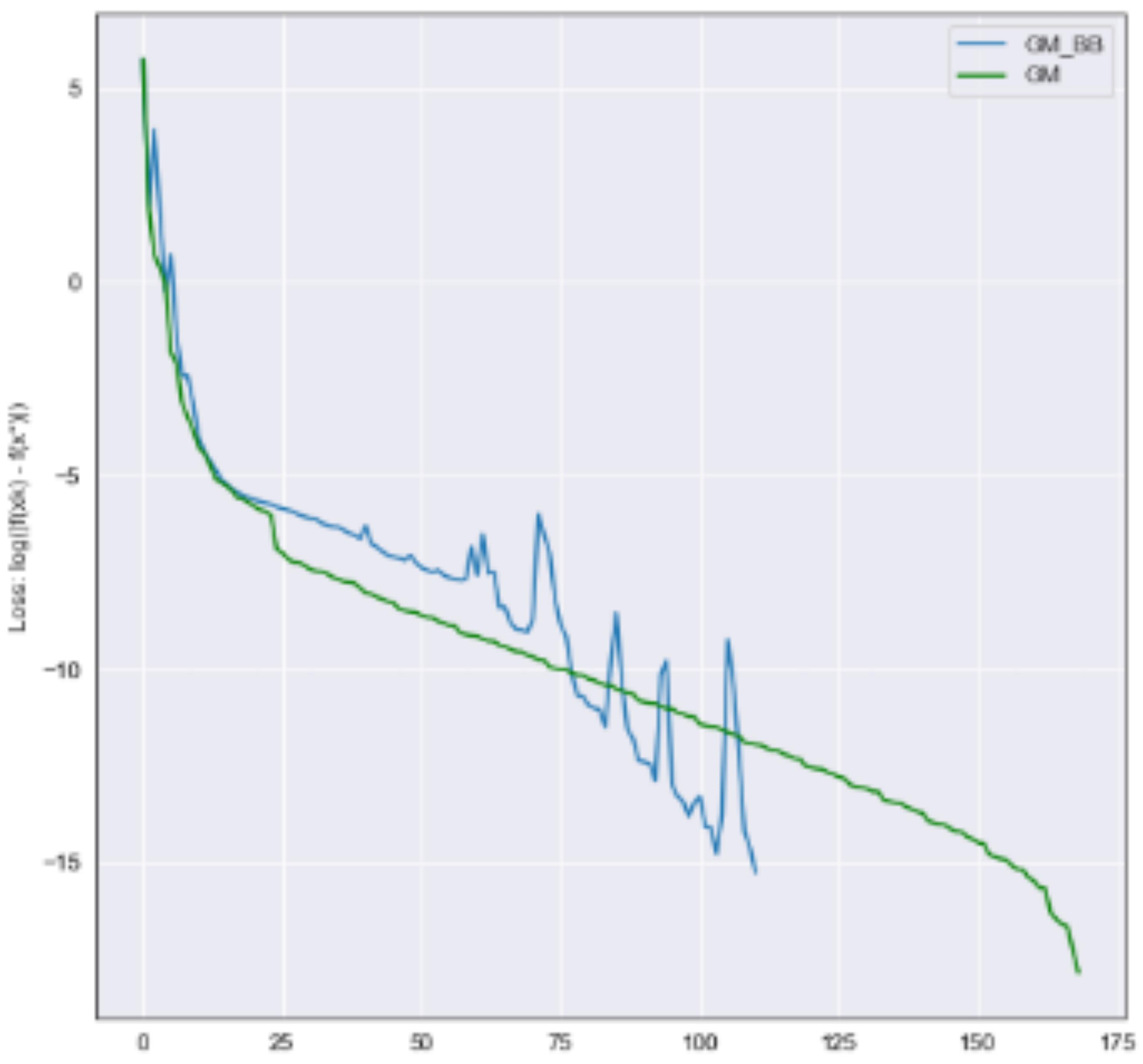


Figure 16: BB GM Convergence Comparison

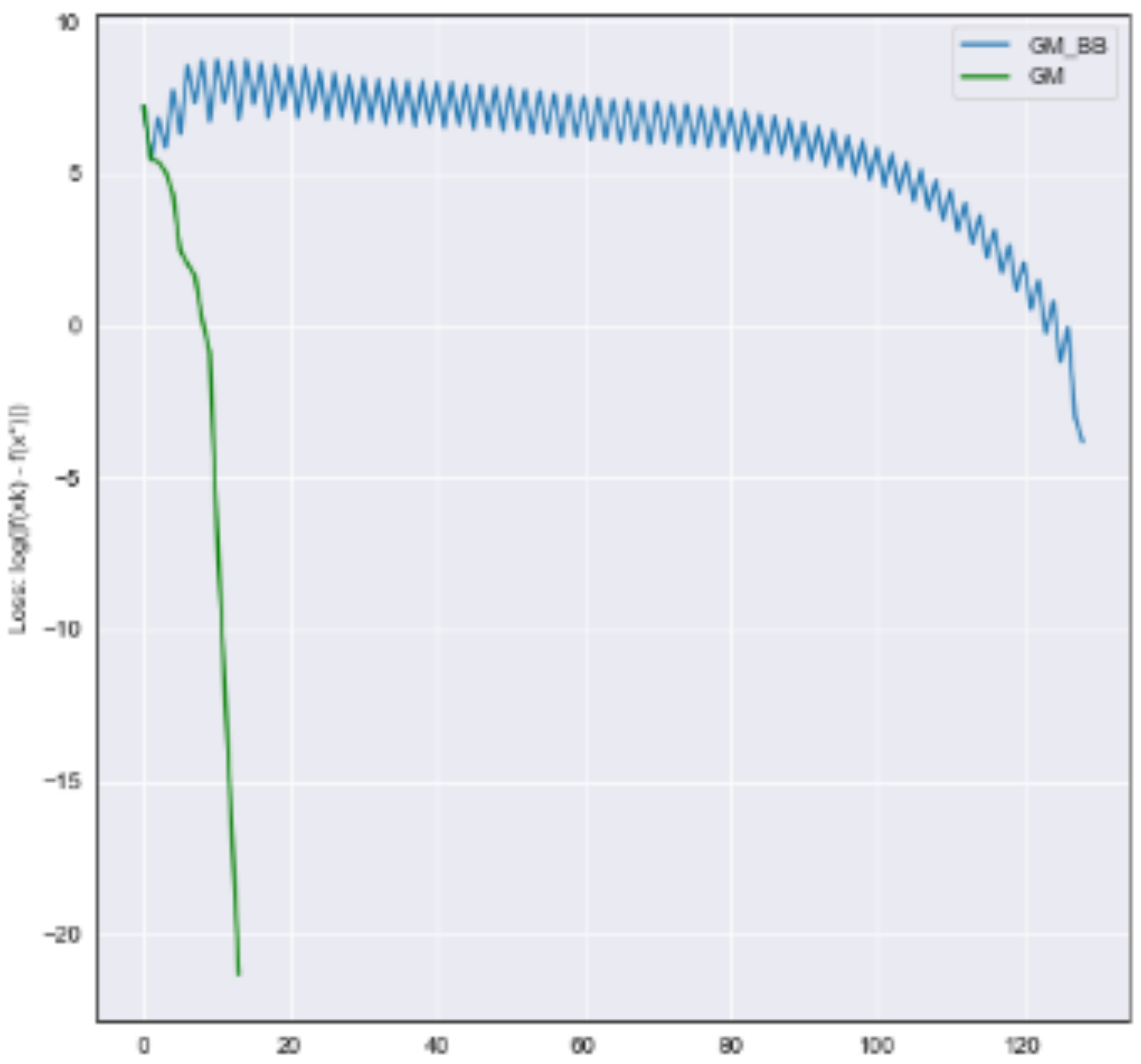


Figure 17: BB GM Wine Data Set Convergence Comparison

Tol/Method	GM	BB
0.1	0.5s	0.3
0.001	2.3s	0.7
1e-5	4.1s	0.8

BFGS & L-BFGS

BFGS & L-BFGS

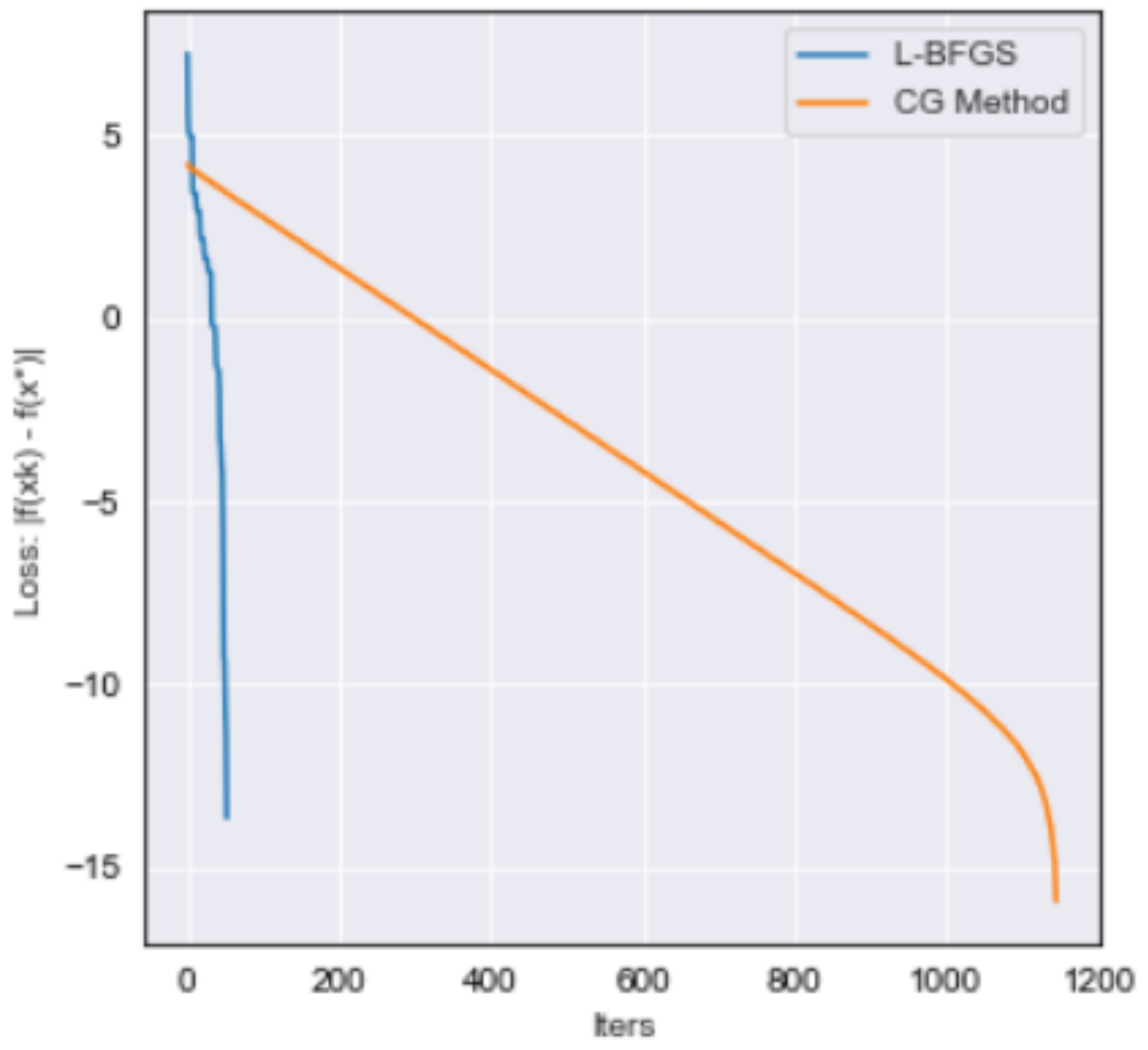
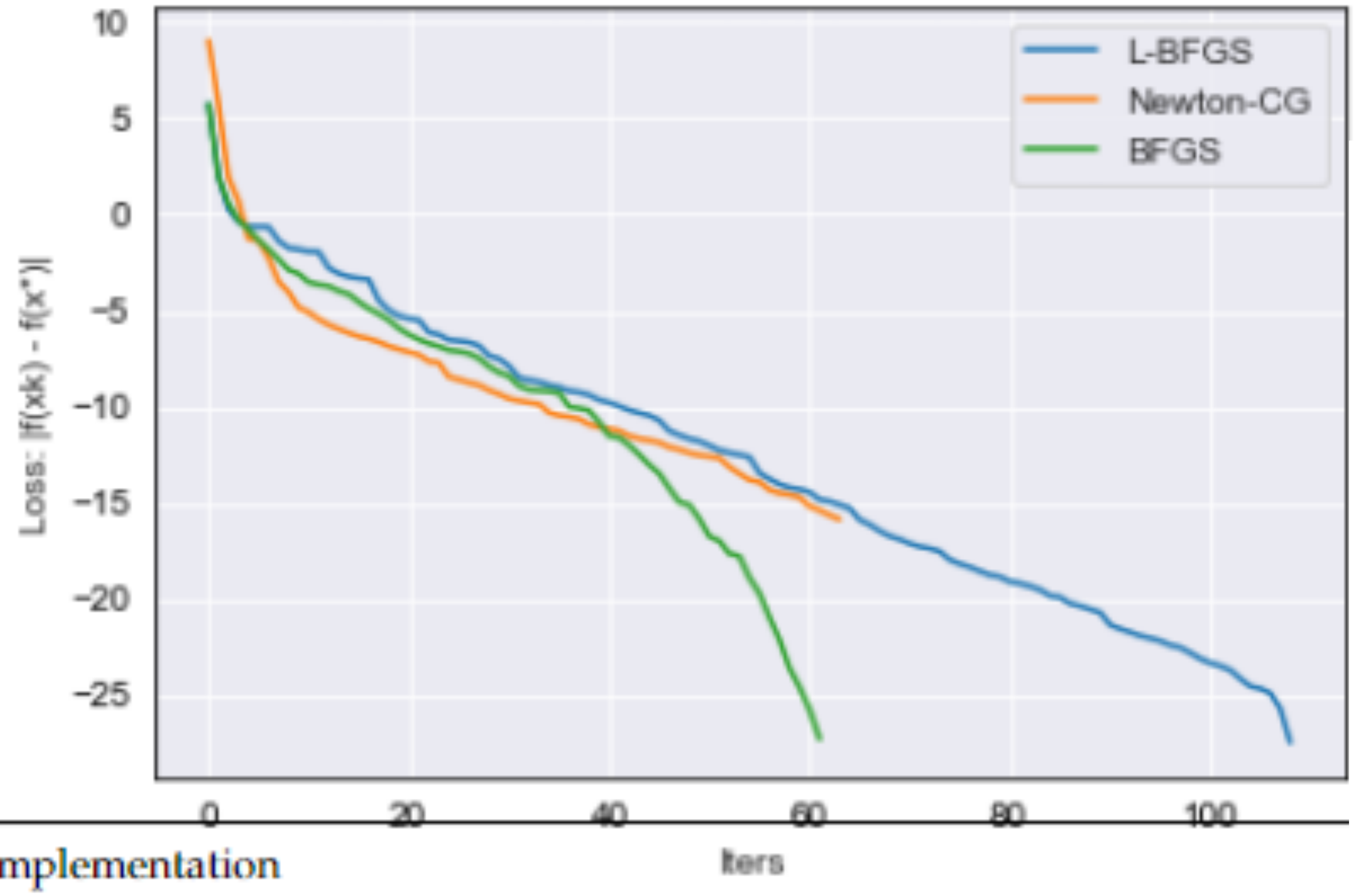


Figure 19: L-BFGS Newton-CG Convergence on Wine data set Comparison



on 4 Implementation

Figure 18: BFGS L-BFGS Newton-CG Convergence Comparison

Algorithm 2 : BFGS Method

set H_0=p*I, x_0
for $k = 0, 1...$ **do**
  $d_k = -H_k * \nabla f(x^k)$
  $\alpha_k = Batrackingstepsize$;
  $x_k + 1 = x_k + \alpha_k d^k$
  if $||\nabla f(x^{k+1})|| <= tol$ **then**
    Stop;
    $s^k = x^{k+1} - xky^k = \nabla f(x^k) - \nabla f(x^{k+1})$;
    if $(s^k)^T y^k < 0$ **then**
      return $H_{k+1} = H_k$
    else
      $H_{k+1}^{BFGS} = H_k + \frac{w^k (s^k)^T + s^k (w^k)^T}{(s^k)^T y^k} - \frac{(w^k)^T y^k}{((s^k)^T y^k)^2} s^k (s^k)^T$, where $w^k = s^k - H_k y^k$
    end if
end for

Weighted Model

Weighted Model

Another loss function

$$\min_{X \in \mathbb{R}^{d \times n}} \frac{1}{2} \sum_{i=1}^{n} \| x_i - a_i \|^2 + \lambda \sum_{i=1}^{n} \sum_{j=i+1}^{n} w_{ij} \| x_i - x_j \|$$

$$w_{ij} = \begin{cases} \exp\left(-\vartheta \| a_i - a_j \|^2\right) & \text{if } (i,j) \in \mathscr{E} \\ 0 & \text{otherwise} \end{cases}$$

$$\mathscr{E} = \bigcup_{i=1}^{n} \left\{ (i,j) : a_j \text{ is among } a_i \text{'s } k\text{-nearest neighbors } i < j \le n \right\}$$

Weighted Model

Construct a W matrice

$$\nabla f_{clust}(x_k) := x_k - a_k - \sum_{j=k+1}^{n} \lambda \frac{x_i - x_j}{||x_i - x_j||} + \sum_{i=1}^{k} \lambda \frac{x_i - x_j}{||x_i - x_j||}$$

$$\nabla f_{clust}(X) := X - A + \lambda W \nabla_{weighted} BX$$

$$W = \begin{pmatrix} w_{12} & \cdots & w_{1n} & w_{23} & \cdots & w_{2n} & w_{34} & \cdots & w_{3n} & w_{nn} \\ -w_{12} & \cdots & -w_{1n} & w_{23} & \cdots & w_{2n} & w_{34} & \cdots & w_{3n} & w_{nn} \\ -w_{12} & \cdots & -w_{1n} & -w_{23} & \cdots & -w_{2n} & w_{34} & \cdots & w_{3n} & w_{nn} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ -w_{12} & \cdots & -w_{1n} & -w_{23} & \cdots & -w_{2n} & -w_{34} & \cdots & -w_{3n} & w_{nn} \end{pmatrix}_{n \times \frac{n(n-1)}{2}}$$

- W matrix is similar to the logic of B matrix
- It is even more sparse

Weighted Model

Drawbacks

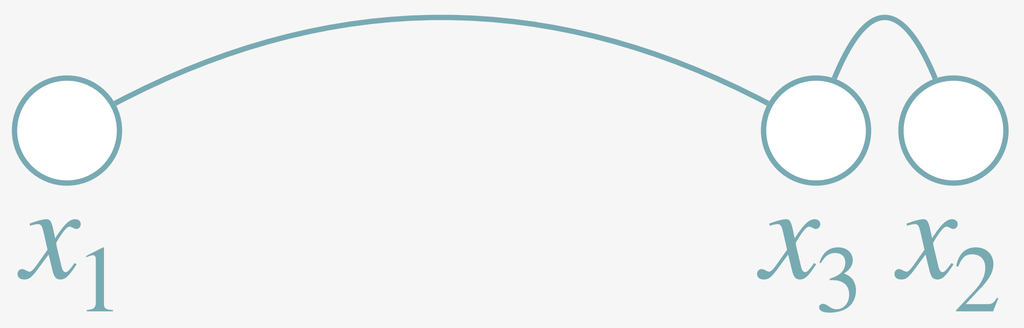
- Sequence of Travel Matters
- Not guarantee to Converge
- X is close to A

Sequence of Travel Matters

The algorithm is not stable. When same data shuffled, the result is different.

Eg.

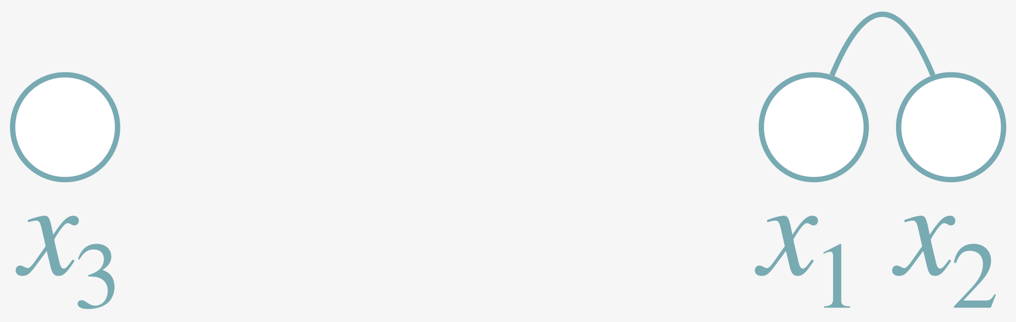
3 points and pick 1 neighbor



$x_1 - x_2$

$\underline{x_1 - x_3}$ $\underline{x_2 - x_3}$

Same points with different order



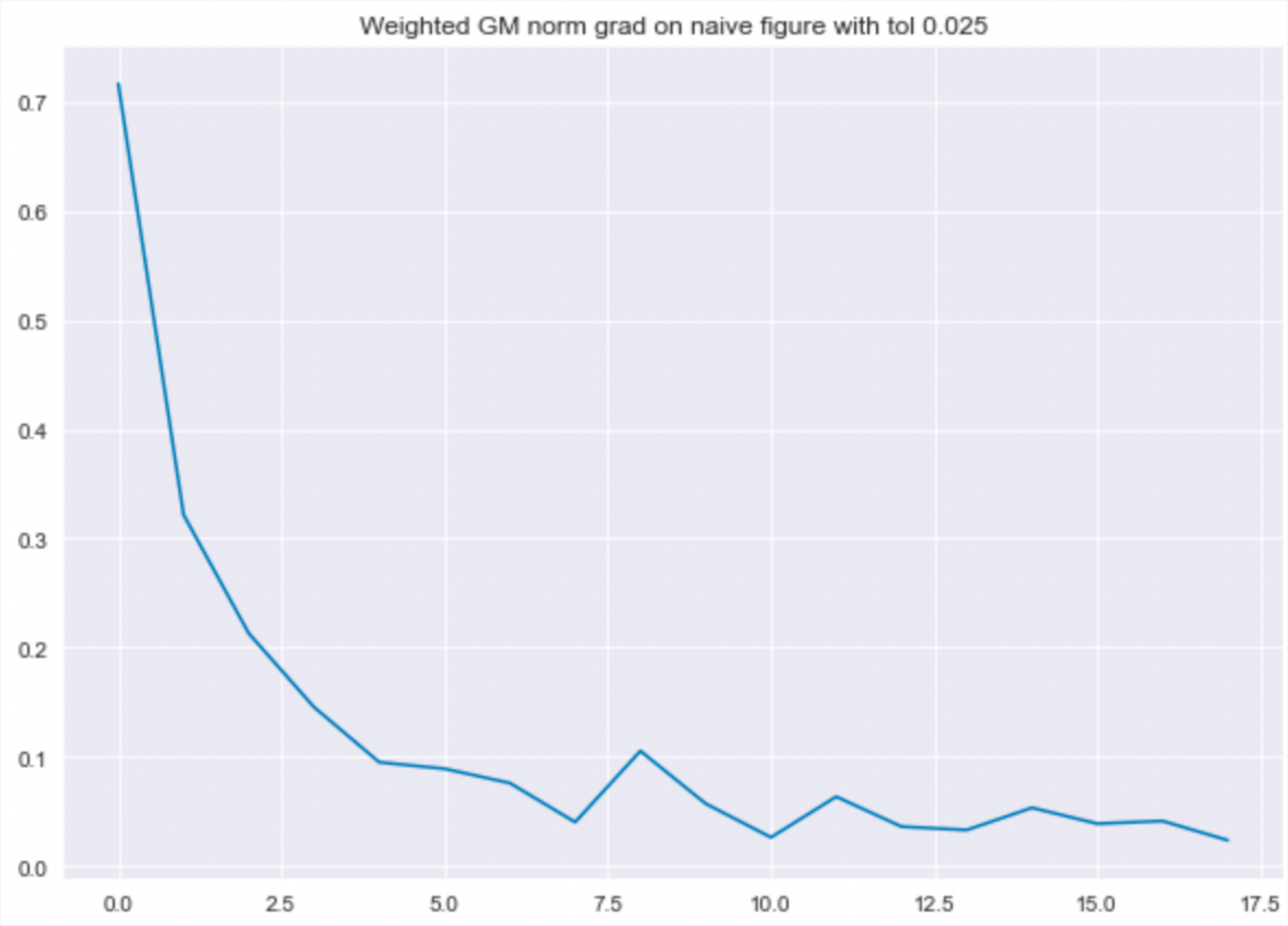
$\underline{x_1 - x_2}$

$x_1 - x_3$ $x_2 - x_3$

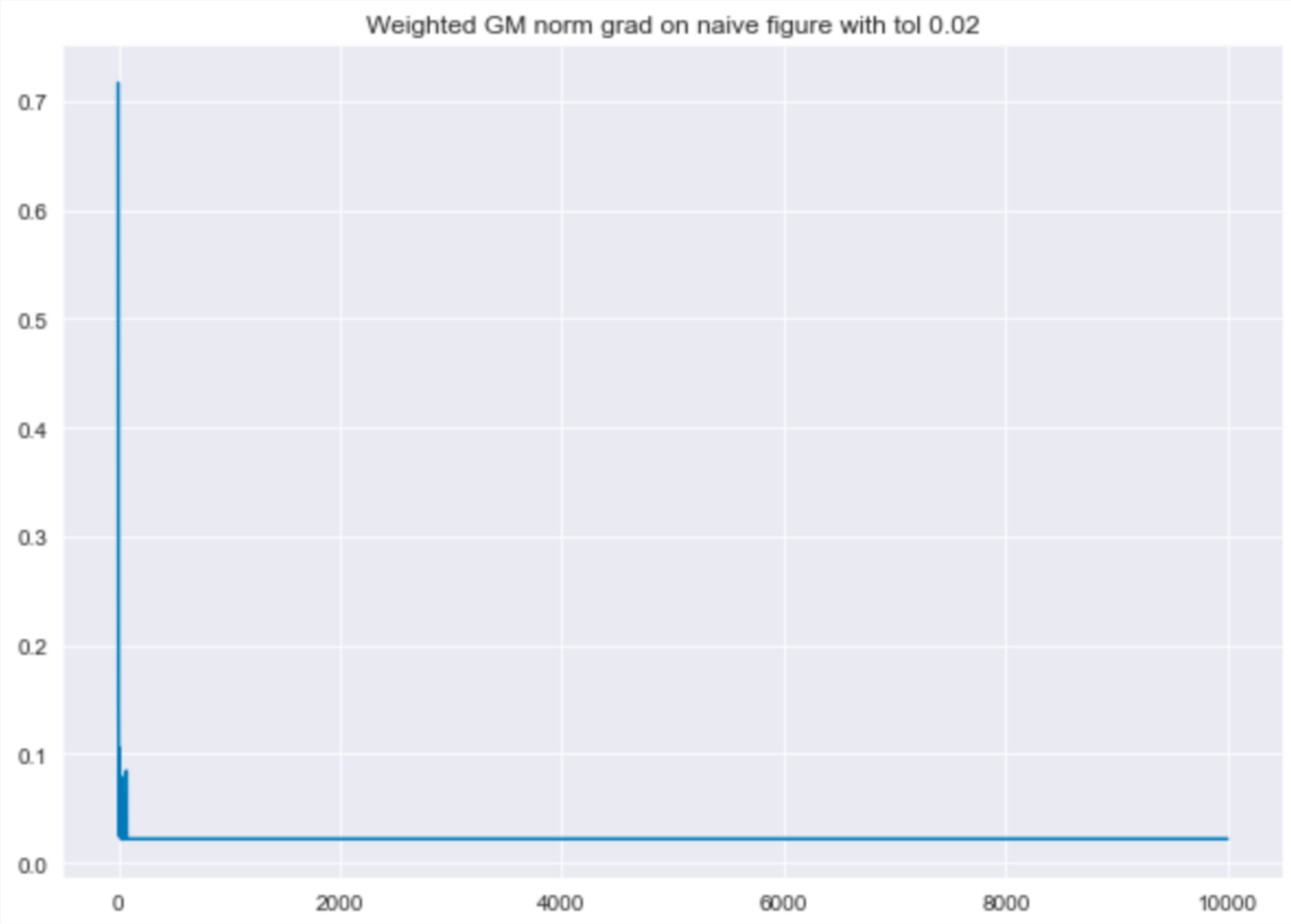
Not guarantee to Converge

The loss function cannot have a global view and there exists a minimum loss it can achieve.

Weighted GM norm grad on naive figure with tol 0.025



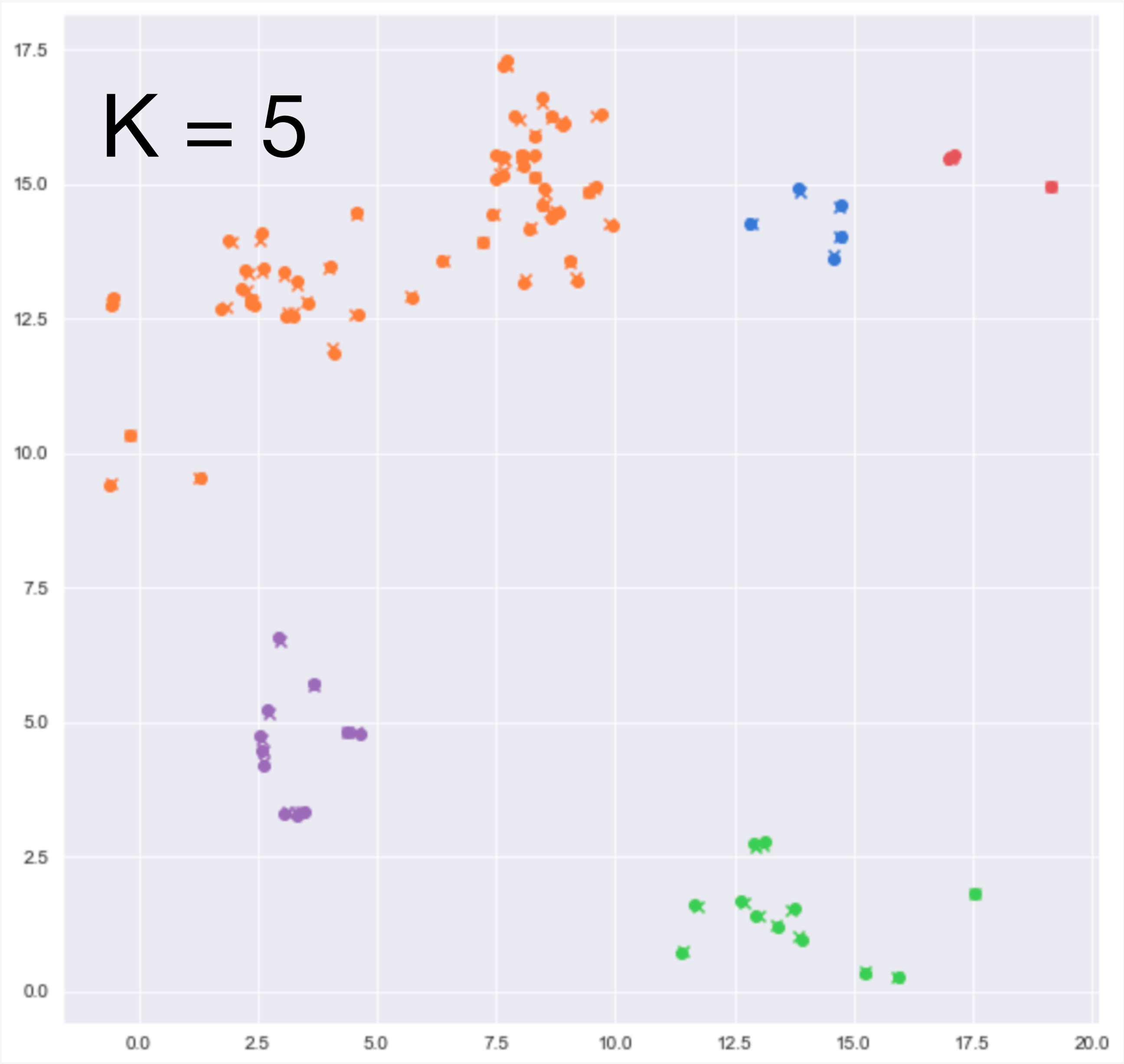
However it cannot achieve tol 0.02



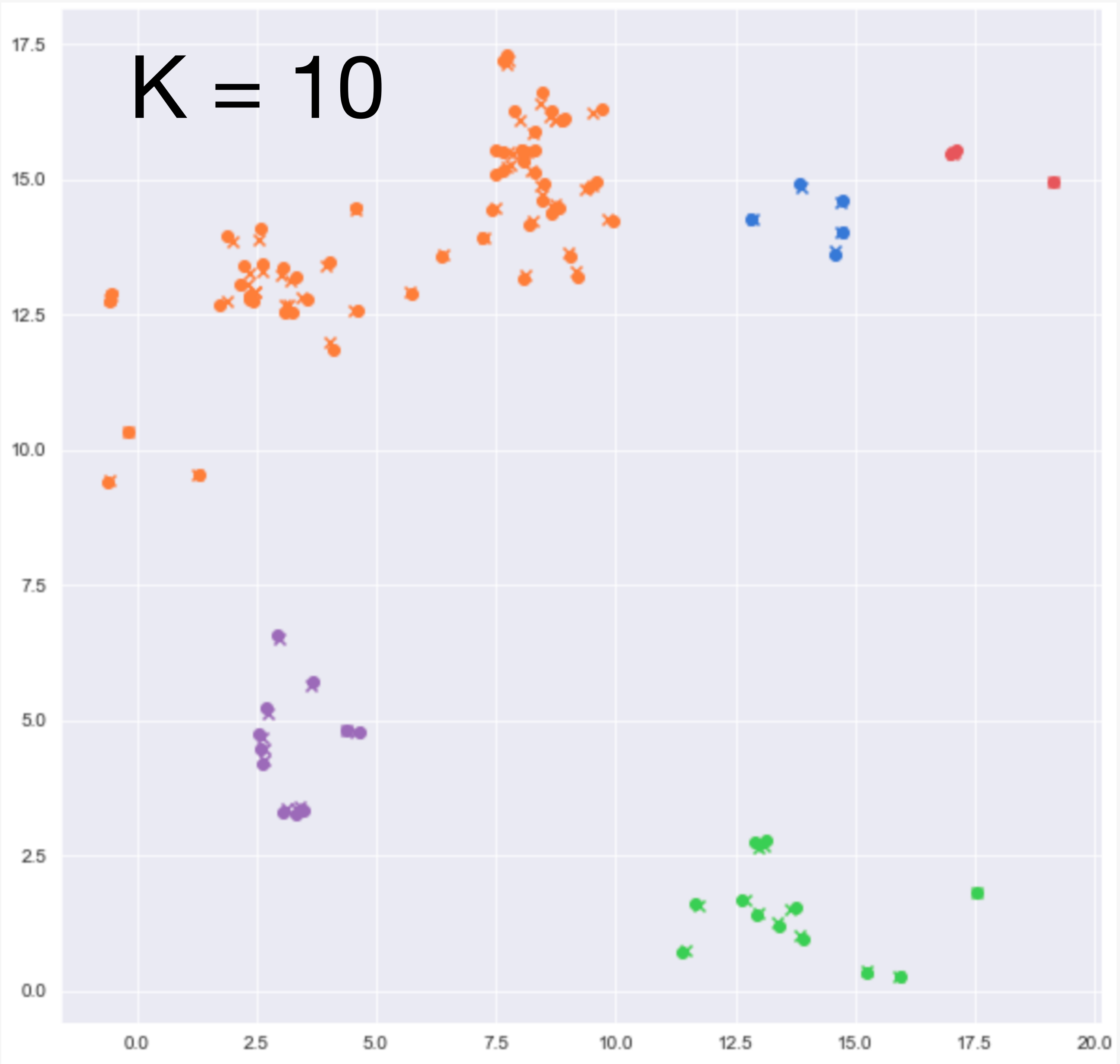
X is close to A

Didn't cluster the original samples. Even change neighbor count doesn't help

Maker O is A and maker X is the X



Maker O is A and maker X is the X



Compare with huber norm

Improvement

Modify the Loss function

$$\min_{X\in\mathbb{R}^{d\times n}} \frac{1}{2}\sum_{i=1}^{n} \| x_i - a_i \|^2 + \underline{\frac{1}{2}}\lambda \sum_{i=1}^{n} \underline{\sum_{j=1}^{n}} w_{ij} \| x_i - x_j \|$$

$x_i - x_j$ will count all pairs of distance.
$x_i - x_i$ doesn't matter because it's 0 in gradient or loss function.

Add $\dfrac{1}{2}$ to make lambda comparable.

It can solve the first drawback. (Sequence of Travel)
Second and third remain unsolved(Hard to Converge and X close to A).

Evaluation

Metrics & Criteria

Classification Problem

- DFS algorithm for finding group belongings.
- Binary Search for finding epsilon
- Purity

Depth First Search

Having the X result, judge the groups belongings by epsilon

Algorithm 4 : DFS Algorithm

```
visited = [False] * len(ans)
groups = [1,2,3,4 ..., N]
for i = 1,2,3,...,N do
    if visited[i] then
        break
    end if
    stack = [i]
    while stack do
        node = stack.pop()
        for j in 1,2,3,...,N do
            if not visited[j] and norm(ans[j]-ans[i]) <= tol then
                stack.append(j)
                [visited[j] = True]
                [groups[j]=groups[i]]
            end if
        end for
    end while
end for
```

The result X is the same shape as A, we can view X as a mapping. Still, we need to cluster X.

Find an unvisited x, allocate it a group, find all its neighbors and so on so forth.

Binary Search

For finding epsilon

Different epsilon varies in diferent scenarios in scale. It's time consuming to manually determine epsilon.

Given the group number, it use binary search to determine the group number.

When comparing the grouping performance of lambda, epsilon should be fixed and this function shouldn't be used!

Algorithm 5 : Auto Grouping

```
set l=0
set r=999
while l < r do
    mid = (l + r)/2
    if get group(l) num < group count then
        r = mid
    else if get group(r) num > group count then
        l = mid
    else
        Return group(mid)
    end if
end while
return group(l)
```

Purity

Compare the accuracy of the classification

$$\text{Purity}(\Omega, C) = \frac{1}{N} \sum_k \max_j \left| w_k \cap c_j \right|$$

Purity is between [0,1].
The higher, the better.

Algorithm 6 : Purity

```
O = classification labels
C = Truth labels
N1 = O's different labels
N2 = C's different labels
sum = 0
for k in 1,2,3,...N1 do
    count = 0
    for c in 1,2,3,...,N2 do
        count = max(count,O[k] intersect c[c])
    end for
    sum += count
end for
return  sum/N
```

Different lambda

λ	0.1	0.2	0.5	0.7	0.8
Number of Groups	56	7	4	2	1

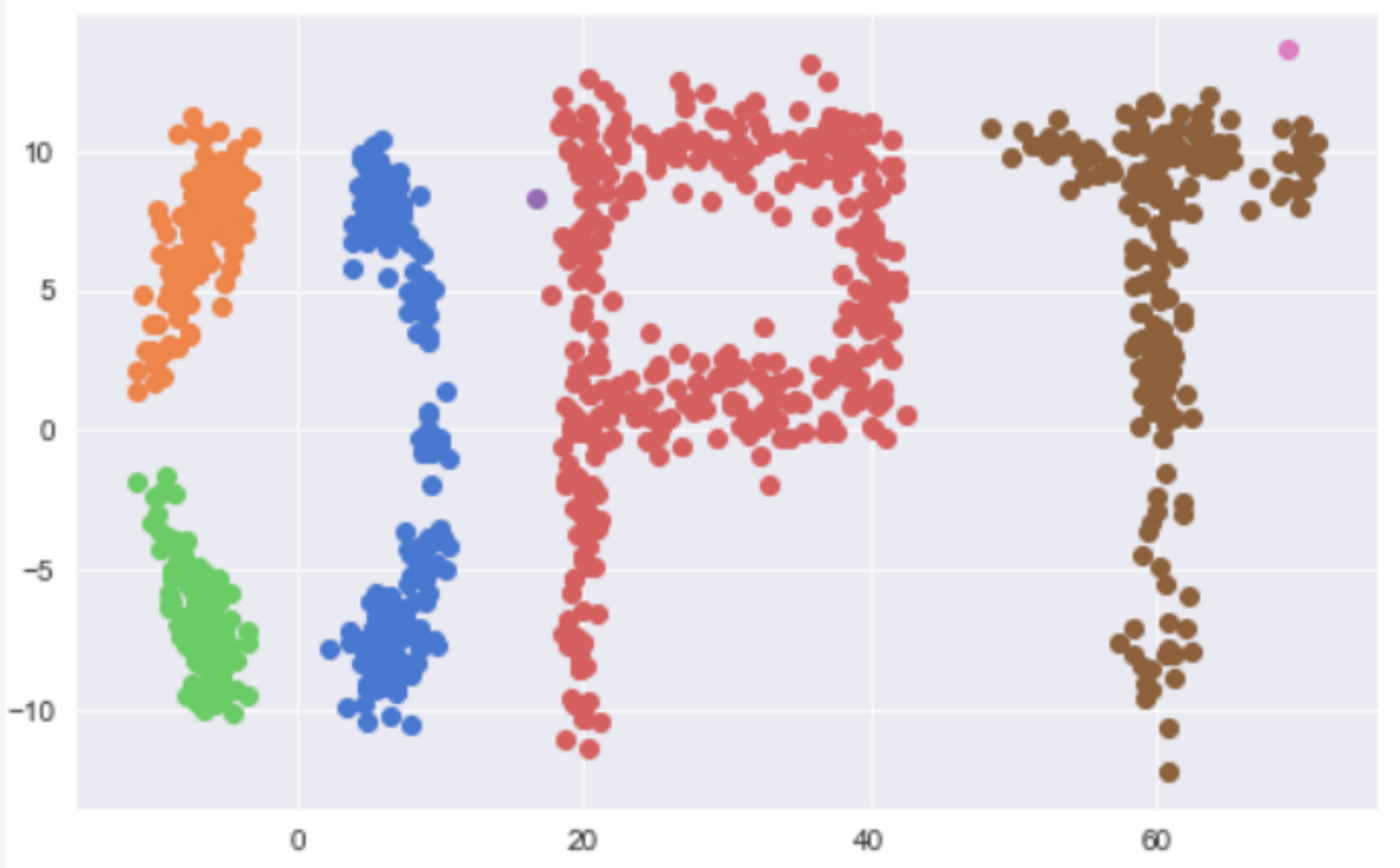


Figure 20: Clustering effect when λ = 0.2

Figure 21: Clustering effect when λ = 0.6

Purity

	Naive Figure	OPT figure	wine	vowel
Weighted Model	0.60929	0.07191	0.39887	0.29014
Original Model	0.60919	0.07196	0.39327	0.19318

Convergence

Weighted

	Naive Figure	OPT figure	wine
AGM	300	780	1750
GM	7	11	17

Huber

	Naive Figure	wine
AGM	10822	347
GM	175	10
CG	62	1420
GM_BB	110	120
BFGS	60	60
L-BFGS	110	60

OPT dataset	Iterations	Time
GM	16	9.7s
AGM	521	26.3s
L-BFGS	61	17.3s

Vowel dataset	Iterations	Time
AGM	74	12s
GM	16	25.7s
L-BFGS	31	30.5s

Summary

Summary

Job has been done

In this project, we use different methods to solve the clustering problems. We firstly prepareself-generate data and read the real-world data.

Then AGM and Newton-CG are use to solve2D optimal problems and we find Newton-CG is better than AGM. Change the original model to weighted one, we observe that gradient method outperforms AGM. We also try to improvethe performance and efficiency by applying different methods, but we lose in improving theperformance in real-word data.

In the last part, we compare the purity and convergence of different method.

Future Work

Future work

Can we do more? Sure.

Block Matrix

Balance between time and space

Test More

on different datasets and parameters

Newton-CG

For Weighted Model

Optimize more

Especially on Newton-CG

THANK YOU

HAO HAO GROUP