Fast Discrete Distribution Clustering Using Wasserstein Barycenter with Sparse Support

IEEE TRANSACTIONS ON SIGNAL PROCESSING(2017)

Jianbo Ye; Panruo Wu; James Z. Wang; Jia Li
Reporter: Fengjiao Gong

April 14, 2022

Outline

1. Discrete Distribution Clustering(D2-clustering)
   1.1 Wasserstein Distance & Barycenter
   1.2 Clustering Criterion
   1.3 Algorithm
2. Alternating Direction Method of Multipliers(ADMM)
   2.1 Precursors
   2.2 Algorithm
3. Bregman ADMM(B-ADMM)
   3.1 Introduction
   3.2 Algorithm
4. Experiments

Discrete Distribution Clustering(D2-clustering)

Wasserstein Distance

Consider two distributions

$$P^{(a)} = \left\{ \left(w_i^{(a)}, x_i^{(a)} \right), i = 1, \ldots, m_a \right\}$$
$$P^{(b)} = \left\{ \left(w_i^{(b)}, x_i^{(b)} \right), i = 1, \ldots, m_b \right\}$$

Then

$$\left(W_p \left(P^{(a)}, P^{(b)} \right) \right)^p := \min_{\{\pi_{i,j} \geqslant 0\}} \sum_{i \in \mathcal{I}_a, j \in \mathcal{I}_b} \pi_{i,j} c \left(x_i^{(a)}, x_j^{(b)} \right)$$
$$\text{s.t.} \sum_{i=1}^{m_a} \pi_{i,j} = w_j^{(b)}, \forall j \in \mathcal{I}_b, \tag{1}$$
$$\sum_{j=1}^{m_b} \pi_{i,j} = w_i^{(a)}, \forall i \in \mathcal{I}_a$$

Discrete Distribution Clustering(D2-clustering)
Wasserstein Distance

where

$$c\left(x_i^{(a)}, x_j^{(b)}\right) = \left\|x_i^{(a)} - x_j^{(b)}\right\|_p^p$$

$$\mathcal{I}_a = \{1, \ldots, m_a\} \qquad \text{(index set)}$$

$$\mathcal{I}_b = \{1, \ldots, m_b\} \qquad \text{(index set)}$$

$$\{\pi_{i,j}\} \qquad \text{matching weights}$$

$$\text{optimal coupling}$$

Here

$$W = W_2 = W_p|_{p=2}$$

Discrete Distribution Clustering(D2-clustering)
Wasserstein Barycenter

Suppose $\left\{P^{(1)}, \ldots, P^{(N)}\right\}$ is a set of discrete distributions

$$\min_{P} \frac{1}{N} \sum_{k=1}^{N} W^2 \left(P, P^{(k)}\right) \tag{2}$$

where

$$P : \{(w_1, x_1), \ldots, (w_m, x_m)\}$$

$$\sum_{i=1}^{m} w_i = 1, w_i \geqslant 0$$

So P is centroid

Discrete Distribution Clustering(D2-clustering)
Clustering Criterion

Criterion:

- minimize total within-cluster variation under the Wasserstein distance
- find a set of centroid distributions $\{Q^{(i)}, i = 1, \ldots, K\}$

$$\min_{\{Q^{(i)}\}} \sum_{k=1}^{\bar{N}} \min_{i=1,\ldots,K} W^2 \left(Q^{(i)}, P^{(k)} \right)$$

Discrete Distribution Clustering(D2-clustering) Algorithm

Alternating Optimization

- **Assignment:** assign each instance to the nearest centroid
- **Update Centroids:** optimize centroids

Discrete Distribution Clustering(D2-clustering) Algorithm

Algorithm 1 D2 Clustering

procedure D2CLUSTERING($\{P^{(k)}\}_{k=1}^{M}$, K)
  Denote the label of each objects by $l^{(k)}$.
  Initialize K random centroid $\{Q^{(i)}\}_{i=1}^{K}$.
  repeat
    for $k = 1, \ldots, M$ **do** ▷ Assignment Step
      $l^{(k)} := \text{argmin}_i \, W(Q^{(i)}, P^{(k)})$;
    for $i = 1, \ldots, K$ **do** ▷ Update Step
      $Q^{(i)} := \text{argmin}_Q \sum_{l^{(k)}=i} W(Q, P^{(k)})$ (*)
  until the number of changes of $\{l^{(k)}\}$ meets some stopping criterion
  return $\{l^{(k)}\}_{k=1}^{M}$ and $\{Q^{(i)}\}_{i=1}^{K}$.

Figure 1: Outer Loop

Computional challenge $\rightarrow$ optimal centroid *for each cluster at each iteration*

Discrete Distribution Clustering(D2-clustering) Algorithm - Main question

$$\min_{P} \frac{1}{N} \sum_{k=1}^{N} W^2 \left(P, P^{(k)} \right) \tag{2}$$

Variables:

- weights in centroid $\{w_i \in \mathbb{R}^+\}$
- support points $\left\{x_i \in \mathbb{R}^d\right\}$
- optimal coupling between P and $P^{(k)}$ $\left\{\pi_{i,j}^{(k)}\right\}$

Discrete Distribution Clustering(D2-clustering) Algorithm - Main question

$$\min_{P} \frac{1}{N} \sum_{k=1}^{N} W^2 \left(P, P^{(k)}\right) \tag{2}$$

Alternating Optimization:

- Fix $\boldsymbol{w}$ and Π, cost function (2) is quadratic in terms of $\boldsymbol{x}$:

$$x_i := \frac{1}{Nw_i} \sum_{k=1}^{N} \sum_{j=1}^{m_k} \pi_{i,j}^{(k)} x_j^{(k)}, \quad \boldsymbol{i} \in \mathcal{I}' = \{1, \ldots, m\} \tag{3}$$

Matrix form

$$\boldsymbol{x} := \frac{1}{N} X \Pi^T \operatorname{diag}(1./\boldsymbol{w})$$

Discrete Distribution Clustering(D2-clustering) Algorithm - Main question

$$\min_{P} \frac{1}{N} \sum_{k=1}^{N} W^2 \left(P, P^{(k)}\right) \tag{2}$$

Alternating Optimization:

- Fix $\boldsymbol{x}$, updating $\boldsymbol{w}$ and Π is challenging (large LP):

$$\begin{aligned} \min_{\Pi \in \mathbb{R}^{+}_{m \times n}, \boldsymbol{w} \in \Delta_m} \sum_{k=1}^{N} & \left\langle C\left(\boldsymbol{x}, \boldsymbol{x}^{(k)}\right), \Pi^{(k)} \right\rangle \\ s.t. \quad & \mathbf{1} \cdot \left(\Pi^{(k)}\right)^T = \boldsymbol{w} \\ & \mathbf{1} \cdot \Pi^{(k)} = \boldsymbol{w}^{(k)} \\ \forall k \in & \mathcal{I}^c = \{1, \ldots, N\} \end{aligned} \tag{4}$$

Scalability

Discrete Distribution Clustering(D2-clustering) Algorithm

Algorithm 2 Centroid Update with Full-batch LP

```
procedure CENTROID({P^(k)}_{k=1}^N)
    repeat
        Updates {x_i} from Eq. (3);
        Updates {w_i} from solving full-batch LP (4);
    until P converges
    return P
```

Figure 2: Inner Loop

Eq(4) is not end result but just one round of centroid update in outer loop

Scalable Methods: not accurate, but fast approximation

Alternating Direction Method of Multipliers(ADMM)

Precursors

- Dual Ascent
- Dual Decomposition
- Augmented Lagrangian & Method of Multipliers

Alternating Direction Method of Multipliers

- Algorithm
- Scaled Form

ADMM to solve Eq(4)

Alternating Direction Method of Multipliers(ADMM)
Precursors - Dual Ascent

Convex optimization problem:

$$\begin{aligned} &\textit{minimize} && f(x) \\ &\textit{subject to} && Ax = b \end{aligned} \tag{5}$$

Lagrangian function:

$$L(x, y) = f(x) + y^T(Ax - b)$$

Strong duality:

same optimal values

Dual problem:

$$\textit{maximize} \qquad g(y) = \inf_x L(x, y)$$

Alternating Direction Method of Multipliers(ADMM)
Precursors - Dual Ascent

Lagrangian function

$$L(x, y) = f(x) + y^T(Ax - b)$$

$x^\star$ can be recovered by $y^\star$

$$x^\star = \arg\min_x L(x, y^\star)$$

[reference] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers", Foundations and Trends R in Machine Learning, vol. 3, no. 1, pp. 1–122, 2011.

Alternating Direction Method of Multipliers(ADMM)
Precursors - Dual Ascent

$$x^\star = \arg\min_x L(x, y^\star)$$

Dual Ascent Method:

$$\begin{aligned} x^{k+1} &:= \arg\min_x L(x, y^k) \\ y^{k+1} &:= y^k + \alpha^k (Ax^{k+1} - b) \\ \alpha^k &> 0 \end{aligned} \tag{6}$$

Alternating Direction Method of Multipliers(ADMM) Precursors - Dual Decomposition

Suppose objective f is separable

$$f(\boldsymbol{x}) = \sum_{i=1}^{N} f_i(x_i)$$
$$\boldsymbol{x} = (x_1, \ldots, x_N)$$
$$A = [A_1, \ldots, A_N]$$

Then

$$L(x, y) = \sum_{i=1}^{N} L_i(x_i, y) = \sum_{i=1}^{N} f_i(x_i) + y^T(A_i x_i - \frac{1}{N} b)$$

Eq(6) can be solved in N separate problems in parallel

Alternating Direction Method of Multipliers(ADMM) Precursors - Augmented Lagrangian & Method of Multipliers

Augmented Lagrangian

$$L_\rho(x, y) = f(x) + y^T(Ax - b) + \frac{\rho}{2}\|Ax - b\|_2^2, \quad \rho > 0 \ (penalty)$$

Apply Dual Ascent:

$$\begin{aligned} x^{k+1} &:= \arg\min_x L_\rho(x, y^k) \\ y^{k+1} &:= y^k + \rho(Ax^{k+1} - b) \\ \rho &> 0 \end{aligned} \tag{7}$$

Method of Multipliers

Alternating Direction Method of Multipliers(ADMM) Algorithm

Problem form:

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

suppose f and g are convex

$$\mathbf{x}_{before} \Rightarrow (x, z)$$

Alternating Direction Method of Multipliers(ADMM) Algorithm

Augmented Lagrangian

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

Iteration:

$$\begin{aligned} x^{k+1} &:= \arg\min_x L_\rho(x, z^k, y^k) \\ z^{k+1} &:= \arg\min_z L_\rho(x^{k+1}, z, y_k) \\ y^{k+1} &:= y^k + \rho(Ax^{k+1} + Bz^{k+1} - c) \end{aligned}$$

Alternating Direction Method of Multipliers(ADMM) Algorithm - Scaled Form

Denote $u = \frac{1}{\rho}y$

$$
\begin{aligned}
x^{k+1} &:= \arg\min_x (f(x) + \tfrac{\rho}{2}\|Ax + Bz^k - c + u^k\|_2^2) \\
z^{k+1} &:= \arg\min_z (g(z) + \tfrac{\rho}{2}\|Ax^{k+1} + Bz - c + u^k\|_2^2) \\
u^{k+1} &:= u^k + Ax^{k+1} + Bz^{k+1} - c
\end{aligned}
$$

Formulas shorter

ADMM to solve Eq(4)

Discrete Distribution Clustering(D2-clustering) Algorithm - Main question

$$\min_{P} \frac{1}{N} \sum_{k=1}^{N} W^2 \left(P, P^{(k)}\right) \tag{2}$$

Alternating Optimization:

- Fix $\boldsymbol{x}$, updating $\boldsymbol{w}$ and Π is challenging (large LP):

$$\begin{aligned} \min_{\Pi \in \mathbb{R}^{+}_{m \times n}, \boldsymbol{w} \in \Delta_m} & \sum_{k=1}^{N} \left\langle C\left(\boldsymbol{x}, \boldsymbol{x}^{(k)}\right), \Pi^{(k)} \right\rangle \\ s.t. \quad & \mathbf{1} \cdot \left(\Pi^{(k)}\right)^T = \boldsymbol{w} \\ & \mathbf{1} \cdot \Pi^{(k)} = \boldsymbol{w}^{(k)} \\ & \forall k \in \mathcal{I}^c = \{1, \ldots, N\} \end{aligned} \tag{4}$$

Scalability

Alternating Direction Method of Multipliers(ADMM)

- Scaled Augmented Lagrangian

$$
\begin{aligned}
L_\rho(\Pi, \boldsymbol{w}, \Lambda) = \sum_{k=1}^{N} \left\langle C\left(\boldsymbol{x}, \boldsymbol{x}^{(k)}\right), \Pi^{(k)} \right\rangle + \rho \sum_{i\in\mathcal{I}' k\in\mathcal{I}^c} \lambda_{i,k} \left(\sum_{j=1}^{m_k} \pi_{i,j}^{(k)} - w_i \right) \\
+ \frac{\rho}{2} \sum_{i\in\mathcal{I}' k\in\mathcal{I}^c} \left(\sum_{j=1}^{m_k} \pi_{i,j}^{(k)} - w_i \right)^2
\end{aligned}
$$

- Iteration

$$
\begin{aligned}
\Pi^{n+1} &:= \underset{\Pi\in\Delta_\Pi}{\operatorname{argmin}} L_\rho\left(\Pi, \boldsymbol{w}^n, \Lambda^n\right), \\
\boldsymbol{w}^{n+1} &:= \underset{\boldsymbol{w}\in\Delta_m}{\operatorname{argmin}} L_\rho\left(\Pi^{n+1}, \boldsymbol{w}, \Lambda^n\right), \\
\lambda_{i,k}^{n+1} &:= \lambda_{i,k}^{n} + \sum_{j=1}^{m_k} \pi_{i,j}^{(k),n+1} - w_i^{n+1}, \boldsymbol{i} \in \mathcal{I}', \boldsymbol{k} \in \mathcal{I}^c.
\end{aligned}
\tag{8}
$$

Alternating Direction Method of Multipliers(ADMM)

- Update $\pi_{i,j}^{k}$: equivalent form:

$$\min_{\pi_{i,j}^{(k)} \geqslant 0} \left\langle C\left(\boldsymbol{x}, \boldsymbol{x}^{(k)}\right), \Pi^{(k)} \right\rangle + \frac{\rho}{2} \sum_{i=1}^{m} \left(\sum_{j=1}^{m_k} \pi_{i,j}^{(k)} - w_i^n + \lambda_{i,k}^n \right)^2 \tag{9}$$
$$\text{s.t. } \mathbf{1} \cdot \Pi^{(k)} = \boldsymbol{w}^{(k)}, k \in \mathcal{I}^c.$$

- Update $\boldsymbol{w_i}$: denote $\tilde{w}_i^{(k),n+1} = \sum_{j=1}^{m_k} \pi_{i,j}^{(k),n+1} + \lambda_{i,k}^n,\ \boldsymbol{i} = 1, \ldots, m$

$$\min_{\boldsymbol{w} \in \Delta_m} \sum_{i=1}^{m} \sum_{k=1}^{N} (\tilde{w}_i^{(k),n+1} - w_i)^2 \tag{10}$$

Alternating Direction Method of Multipliers(ADMM) Algorithm

Algorithm 3 Centroid Update with ADMM

procedure CENTROID($\{P^{(k)}\}_{k=1}^{N}$, P, Π)
    Initialize $\Lambda^0 = 0$ and $\Pi^0 := \Pi$.
    repeat
        Updates $\{x_i\}$ from Eq.(3);
        Reset dual coordinates Λ to zero;
        for $iter = 1, \ldots, T_{admm}$ **do**
            for $k = 1, \ldots, N$ **do**
                Update $\{\pi_{i,j}\}^{(k)}$ based on QP
            Update $\{w_i\}$ based on QP
            Update Λ based on Eq. (8);
    until P converges
    return P

Figure 3: Centroid Update with ADMM

Computation limitation:

$$\text{solve N of LP} \Rightarrow \text{solve N of QP}$$

B-ADMM replace the quadratic penalty term by **Bregman divergence**:

$$L_\rho^\phi(\mathbf{x}, \mathbf{z}, \mathbf{y}) = f(\mathbf{x}) + g(\mathbf{z}) + \langle \mathbf{y}, \mathbf{Ax} + \mathbf{Bz} - \mathbf{c} \rangle + \rho B_\phi(\mathbf{c} - \mathbf{Ax}, \mathbf{Bz})$$

where

$$B_\phi(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x}) - \phi(\mathbf{y}) - \langle \nabla\phi(\mathbf{y}), \mathbf{x} - \mathbf{y} \rangle \geq 0$$

as $\phi : \Omega \to \mathbb{R}$ is a continuously differentiable and **strictly convex** function on the relative interior of a convex set Ω

Here $B_\phi(\mathbf{x}, \mathbf{y})$ is not necessarily convex w.r.t $y \Rightarrow$ *Not directly from ADMM*

Bregman ADMM(B-ADMM)
Introduction

B-ADMM Update:

$$
\begin{aligned}
\mathbf{x}_{t+1} &= \text{argmin}_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x}) + \langle \mathbf{y}_t, \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z}_t - \mathbf{c}\rangle + \rho B_\phi\left(\mathbf{c} - \mathbf{A}\mathbf{x}, \mathbf{B}\mathbf{z}_t\right) \\
\mathbf{z}_{t+1} &= \text{argmin}_{\mathbf{z}\in\mathcal{Z}} g(\mathbf{z}) + \langle \mathbf{y}_t, \mathbf{A}\mathbf{x}_{t+1} + \mathbf{B}\mathbf{z} - \mathbf{c}\rangle + \rho B_\phi\left(\mathbf{B}\mathbf{z}, \mathbf{c} - \mathbf{A}\mathbf{x}_{t+1}\right) \\
\mathbf{y}_{t+1} &= \mathbf{y}_t + \rho\left(\mathbf{A}\mathbf{x}_{t+1} + \mathbf{B}\mathbf{z}_{t+1} - \mathbf{c}\right)
\end{aligned}
$$

Ensure new updates do not violate the equality constraint significantly

Bregman ADMM(B-ADMM) Application

Optimal Transport Problem:

$$\min \langle \mathbf{C}, \mathbf{X} \rangle \text{ s.t. } \mathbf{X}\mathbf{e} = \mathbf{a}, \mathbf{X}^T\mathbf{e} = \mathbf{b}, \mathbf{X} \geq 0 \qquad (\textit{Linear Program})$$

where $\langle \mathbf{C}, \mathbf{X} \rangle = \operatorname{Tr}\left(\mathbf{C}^T\mathbf{X}\right)$,$\mathbf{C} \in \mathbb{R}^{m\times n}$ is cost matrix,$\mathbf{X} \in \mathbb{R}^{m\times n}$, $\mathbf{a} \in \mathbb{R}^{m\times 1}$, $\mathbf{b} \in \mathbb{R}^{m\times 1}$, e is a column vector of ones.

Bregman ADMM(B-ADMM)
Application

Introduce variable $\mathbf{Z}$ to split the constraints into two simplex such that

$$\mathbf{\Delta_x} = \{\mathbf{X} \mid \mathbf{X} \geq 0, \mathbf{Xe} = \mathbf{a}\}$$
$$\mathbf{\Delta_z} = \left\{\mathbf{Z} \mid \mathbf{Z} \geq 0, \mathbf{Z}^T\mathbf{e} = \mathbf{b}\right\}$$

Then

$$\min\langle \mathbf{C}, \mathbf{X}\rangle \text{ s.t. } \mathbf{X} \in \mathbf{\Delta_x}, \mathbf{Z} \in \mathbf{\Delta_z}, \mathbf{X} = \mathbf{Z}$$

Bregman ADMM(B-ADMM) Application

B-ADMM Updates:

$$\begin{aligned}
\mathbf{X}^{t+1} &= \mathrm{argmin}_{\mathbf{X}\in\mathbf{\Delta_x}} \langle \mathbf{C}, \mathbf{X}\rangle + \left\langle \mathbf{Y}^t, \mathbf{X}\right\rangle + \rho\mathbf{KL}\left(\mathbf{X}, \mathbf{Z}^t\right), \\
\mathbf{Z}^{t+1} &= \mathrm{argmin}_{\mathbf{Z}\in\mathbf{\Delta_z}} \left\langle \mathbf{Y}^t, -\mathbf{Z}\right\rangle + \rho\mathbf{KL}\left(\mathbf{Z}, \mathbf{X}^{t+1}\right), \\
\mathbf{Y}^{t+1} &= \mathbf{Y}^t + \rho\left(\mathbf{X}^{t+1} - \mathbf{Z}^{t+1}\right).
\end{aligned}$$

Closed Form:

$$X_{ij}^{t+1} = \frac{Z_{ij}^t \exp\left(-\frac{C_{ij}+Y_{ij}^t}{\rho}\right)}{\sum_{j=1}^n Z_{ij}^t \exp\left(-\frac{C_{ij}+Y_{ij}^t}{\rho}\right)} a_i, \quad Z_{ij}^{t+1} = \frac{X_{ij}^{t+1} \exp\left(\frac{Y_{ij}^t}{\rho}\right)}{\sum_{i=1}^m X_{ij}^{t+1} \exp\left(\frac{Y_{ij}^t}{\rho}\right)} b_j$$

BADMM can be faster than ADMM by a factor of $O(n/\ln n)$

Bregman ADMM(B-ADMM) Algorithm

Consider two sets of variables

$$\Delta_{k,1} := \left\{ \pi_{i,j}^{(k,1)} \geqslant 0 : \sum_{i=1}^{m} \pi_{i,j}^{(k,1)} = w_j^{(k)}, j \in \mathcal{I}_k \right\}$$
$$\Delta_{k,2}(\boldsymbol{w}) := \left\{ \pi_{i,j}^{(k,2)} \geqslant 0 : \sum_{j=1}^{m_k} \pi_{i,j}^{(k,2)} = w_i, i \in \mathcal{I}' \right\}$$

Then $\Pi^{(k,1)} \in \Delta_{k,1}$ and $\Pi^{(k,2)} \in \Delta_{k,2}(\boldsymbol{w})$

Bregman ADMM(B-ADMM) Algorithm

Notation

$$\bar{\Pi}^{(1)} = \left\{\Pi^{(1,1)}, \Pi^{(2,1)}, \ldots, \Pi^{(N,1)}\right\}$$
$$\bar{\Pi}^{(2)} = \left\{\Pi^{(1,2)}, \Pi^{(2,2)}, \ldots, \Pi^{(N,2)}\right\}$$
$$\bar{\Pi} = \left\{\bar{\Pi}^{(1)}, \bar{\Pi}^{(2)}\right\}$$
$$\Lambda = \left\{\Lambda^{(1)}, \ldots, \Lambda^{(N)}\right\}$$
$$\Lambda^{(k)} = \left(\lambda_{i,j}^{(k)}\right), i \in \mathcal{I}', j \in \mathcal{I}_k$$

Formulation

$$\min_{\bar{\Pi}, \boldsymbol{w}} \sum_{k=1}^{N} \left\langle C\left(\boldsymbol{x}, \boldsymbol{x}^{(k)}\right), \Pi^{(k,1)} \right\rangle$$
$$\text{s.t. } \boldsymbol{w} \in \Delta_m$$
$$\Pi^{(k,1)} \in \Delta_{k,1}, \quad \Pi^{(k,2)} \in \Delta_{k,2}(\boldsymbol{w})$$
$$\Pi^{(k,1)} = \Pi^{(k,2)}, \quad k = 1, \ldots, N$$

Bregman ADMM(B-ADMM) Algorithm

B-ADMM Updates:

$$\bar{\Pi}^{(1),n+1} := \underset{\left\{\Pi^{(k,1)} \in \Delta_{k,1}\right\}}{\operatorname{argmin}} \sum_{k=1}^{N} \left(\left\langle C\left(\boldsymbol{x}, \boldsymbol{x}^{(k)}\right), \Pi^{(k,1)}\right\rangle + \left\langle \Lambda^{(k),n}, \Pi^{(k,1)}\right\rangle + \rho \mathrm{KL}\left(\Pi^{(k,1)}, \Pi^{(k,2),n}\right)\right)$$

$$\bar{\Pi}^{(2),n+1}, \boldsymbol{w}^{n+1} := \underset{\substack{\left\{\Pi^{(k,2)} \in \Delta_{k,1}(\boldsymbol{w})\right\} \\ \boldsymbol{w} \in \Delta_m}}{\operatorname{argmin}} \sum_{k=1}^{N} \left(-\left\langle \Lambda^{(k),n}, \Pi^{(k,2)}\right\rangle + \rho \mathrm{KL}\left(\Pi^{(k,2)}, \Pi^{(k,1),n+1}\right)\right)$$

$$\Lambda^{n+1} := \Lambda^{n} + \rho\left(\bar{\Pi}^{(1),n+1} - \bar{\Pi}^{(2),n+1}\right)$$

Kullback-Leibler divergence(KL divergence):

$$B_{\phi}(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} x_i \log \frac{x_i}{y_i}$$

Bregman ADMM(B-ADMM) Algorithm

Algorithm 4 Centroid Update with B-ADMM

---

procedure CENTROID($\{P^{(k)}\}_{k=1}^{N}$, P, Π).
  $\Lambda := 0$; $\bar{\Pi}^{(2),0} := \Pi$.
  repeat
    Update $\boldsymbol{x}$ from Eq.(3) per τ loops;
    for $k = 1, \ldots, N$ **do**
      Update $\Pi^{(k,1)}$
      Update $\{\tilde{\pi}_{i,j}^{(k,1)}\}$
    Update $\boldsymbol{w}$
    for $k = 1, \ldots, N$ **do**
      Update $\Pi^{(k,2)}$
      $\Lambda^{(k)} := \Lambda^{(k)} + \rho(\Pi^{(k,1)} - \Pi^{(k,2)})$;
  until P converges
  return P

Figure 4: Centroid Update with B-ADMM

where $\tilde{\pi}_{i,j}^{(k,1),n+1} := \pi_{i,j}^{(k,1),n+1} \exp\left[\frac{1}{\rho}\lambda_{i,j}^{(k),n}\right] + eps$, eps is floating-point tolerence.

Experiments

Data	$\bar{N}$	d	m	K
synthetic	2,560,000	$\geq$16	$\geq$32	256
image color	5,000	3	8	10
image texture	-	-	-	-
USPS digits	11,000	2	80	360
BBC news abstract	2,225	300	16	15
Wiki events abstract	1,983	400	16	100
20newsgroups GV	18,774	300	64	40
20newsgroups WV	-	400	100	-

Figure 5: DATASETS IN THE EXPERIMENTS

$\bar{N}$: DATA SIZE
d: DIMENSION OF THE SUPPORT VECTORS
m: NUMBER OF SUPPORT VECTORS IN A CENTROID
K: MAXIMUM NUMBER OF CLUSTERS TESTED
ENTRY WITH SAME VALUE AS IN PREVIOUS ROW IS "-"

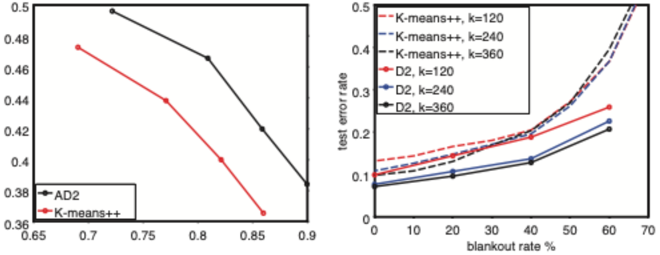
Experiments

# processors	32	64	128	256	512
SSE (%)	93.9	93.4	92.9	84.8	84.1
WSE on $\bar{N}$ (%)	99	94.8	95.7	93.3	93.2
WSE on m (%)	96.6	89.4	83.5	79.0	-

Figure 6: SCALING EFFICIENCY OF AD2-CLUSTERING IN PARALLEL IMPLEMENTATION

- Strong scaling efficiency(SSE):
  speed-up gained from using more and more processors when the problem is fixed in size
- Weak scaling efficiency(WSE):
  how stable the real computation time can be when proportionally more processors are used as the size of the problem grows

Experiments



(a) Homogeneity vs. completeness (b) Test error rate vs. blankout rate

Figure 7: Comparisons between Kmeans++ and AD2-clustering on USPS dataset.

Thanks!