# Co-clustering through Optimal Transport

SMAI Project - Team 9

## Introduction

- Co-clustering is an unsupervised learning approach that aims to discover homogeneous groups of data instances and features by grouping them simultaneously.
- Many co-clustering methods need number of clusters as input but our method automatically detects the number of co-clusters.
- We use entropy regularized optimal transport between empirical measures defined on data instances and features in order to obtain an estimated joint probability density function represented by the optimal coupling matrix.
- This matrix is further factorized to obtain the induced row and columns partitions using multiscale representations approach.
- The algorithm derived for the proposed method and its kernelized version based on the notion of Gromov-Wasserstein distance are fast, accurate and can determine automatically the number of both row and column clusters.

## Optimal Transport

- Optimal transportation theory was first introduced to study the problem of resource allocation.
- Assuming that we have a set of n factories and a set of n mines, the goal of optimal transportation is to move the ore from mines to factories in an optimal way, i.e., by minimizing the overall transport cost.
- If M is the set of mines and F is the set of factories.*Transport plan* is a bijection $T : M \rightarrow F$. In other words, each mine $m \in M$ supplies precisely one factory $T(m) \in F$ and each factory is supplied by precisely one mine. For optimal transport plan,we need to minimize

$$c(T) := \sum_{m \in M} c(m, T(m))$$

# Optimal Transport

- More formally, given two empirical probability measures $\hat{\mu}_S = \frac{1}{N_S}\sum_{i=1}^{N_S}\delta_{x_i^S}$ and $\hat{\mu}_T = \frac{1}{N_T}\sum_{i=1}^{N_T}\delta_{x_i^T}$ defined as uniformly weighted sums of Dirac with mass at locations supported on two point sets $X_S = \{\boldsymbol{x}_i^S \in \mathbb{R}^d\}_{i=1}^{N_S}$ and $X_T = \{\boldsymbol{x}_i^T \in \mathbb{R}^d\}_{i=1}^{N_T}$, the Monge-Kantorovich problem consists in finding a probabilistic coupling $\gamma$ matrix defined as a joint probability measures $X_S \times X_T$ with marginals $\hat{\mu}_S$ and $\hat{\mu}_T$ that minimizes the cost of transport w.r.t. some $l : X_s \times X_t \to \mathbb{R}^+$:

$$\min_{\gamma \in \Pi(\hat{\mu}_S, \hat{\mu}_T)} \langle M, \gamma \rangle_F$$

Where $\langle \cdot, \cdot \rangle_F$ is the Frobenius dot product

## Optimal Transport

- $\Pi(\hat{\mu}_S, \hat{\mu}_T) = \{\gamma \in \mathbb{R}_+^{N_S \times N_T} | \gamma \mathbf{1} = \hat{\mu}_S, \gamma^T \mathbf{1} = \hat{\mu}_T\}$ is a set of doubly stochastic matrices and M is a dissimilarity matrix i.e , $M_{ij} = l(\boldsymbol{x}_i^S, \boldsymbol{x}_j^T)$ defining the energy needed to move a probability mass from $\boldsymbol{x}_i^S$ to $\boldsymbol{x}_j^T$

- This problem admits a unique solution γ* and defines a metric on the space of probability measures (called the Wasserstein distance) as follows:

$$W(\hat{\mu}_S, \hat{\mu}_T) = \min_{\gamma \in \Pi(\hat{\mu}_S, \hat{\mu}_T)} \langle M, \gamma \rangle_F$$

- The success of algorithms based on this distance is also due to (Cuturi, 2013) who introduced an entropy regularized version of optimal transport that can be optimized efficiently using matrix scaling algorithm.

## Entropic Regularization

- Entropic regularization found its application to the optimal transportation problem through the following objective function:

$$\min_{\gamma \in \Pi(\hat{\mu}_S, \hat{\mu}_T)} \langle M, \gamma \rangle_F - \frac{1}{\lambda} E(\gamma)$$

- Second term $E(\gamma) = -\sum_{i,j}^{N_S, N_T} \gamma_{i,j} \log(\gamma_{i,j})$ is the entropy

**Advantages :**

- Allows to obtain smoother and more numerically stable solutions compared to the original case and converges to it at the exponential rate.
- It allows to solve optimal transportation problem efficiently using Sinkhorn-Knopp matrix scaling algorithm which can be parallelised.

## Why Optimal Transport for Co-Clustering ???

We pose the co-clustering problem as the task of transporting the empirical measure defined on the data instances to the empirical measure defined on the data features. The intuition behind this process is very natural to co-clustering and consists in capturing the associations between instances and features of the data matrix. And the solution of optimal transportation problem can be decomposed into 3 terms as

$$\gamma_\lambda^* = \mathrm{diag}(\boldsymbol{\alpha})\xi_\lambda \mathrm{diag}(\boldsymbol{\beta})$$

where ξ is the approximated joint probability distribution and remaining are diagonal matrices which can be approximated to distributions of data instances and features.

## Why Optimal Transport ? - Mathematical View

The optimal  co-clustering is taken as the one that minimizes the difference in mutual information between the original random variables and the mutual information between the  clustered random variables i.e., I(X,Y) - I(X',Y').The loss in mutual information  can be expressed as the distance of p ( X; Y ) to an approximation q ( X; Y ) i.e., KL(p(x,y)/q(x,y) where q(x,y) is of the form

$$q(x,y) = p(x/x')p(x',y')p(y/y').$$

This solution is similar to the solution we obtain in Optimal Transport using Sinkhorp Algorithm. If we assume p(x',y') follows Gibbs Distribution then p(x',y') can be approximated to $\xi$.

## Co Clustering - Problem Setup

- Let X and Y be two random variables taking values in the sets $\{x_r\}^n_{r=1}$ and $\{y_c\}^d_{c=1}$ , respectively, where subscripts r and c correspond to rows (instances) and columns (features).
- Let the joint probability distribution between X and Y be denoted by p(X, Y ) is estimated from the data matrix $A \in R^{n\times d}$. We further assume that X and Y consist of instances that are distributed w.r.t. probability measures $\mu_r$ , $\mu_c$ supported on $\Omega_r$ , $\Omega_c$ where $\Omega_r \subseteq R^d$ and $\Omega_r \subseteq R^n$ respectively.
- The problem of co-clustering consists in jointly grouping the set of features and the set of instances into homogeneous blocks by finding two assignment functions $C_r$ and $C_c$ that map as follows: $C_r : \{x_1 , \ldots , x_n \} \rightarrow \{\hat{x}_1 , \ldots , \hat{x}_g \}$, $C_c : \{y_1 , \ldots , y_d \} \rightarrow \{\hat{y}_1 , \ldots , \hat{y}_m \}$ where g and m denote the number of row and columns clusters, and discrete random variables $\hat{X}$ and $\hat{Y}$ represent the partitions induced by X and Y , i.e., $\hat{X} = C_r(X) =$ and $\hat{Y} = C_c(Y )$.

$$\hat{\mu}_r = \frac{1}{n}\sum_{i=1}^{n} \delta_{\boldsymbol{x}_i} \text{ and } \hat{\mu}_c = \frac{1}{m}\sum_{i=1}^{m} \delta_{\boldsymbol{y}_i}$$

# Optimal Transportation - Proposed Approach

- We use optimal transportation to find a probabilistic coupling of the empirical measures defined based on rows and columns of a given data matrix. More formally, for some fixed $\lambda > 0$ we solve the co-clustering problem through the following optimization procedure:

$$\gamma_\lambda^* = \mathrm{argmin}_{\gamma \in \Pi(\hat{\mu}_\mathrm{r}, \hat{\mu}_\mathrm{c})} \langle \mathrm{M}, \gamma \rangle_\mathrm{F} - \frac{1}{\lambda}\mathrm{E}(\gamma) \qquad \text{--(1)}$$

- Matrix M is computed using the Euclidean distance, i.e., $M_{ij} = \|x_i - y_j\|^2$. The elements of the resulting matrix $\gamma_\lambda^*$ provides us with the weights of associations between instances and features: similar instances and features correspond to higher values in $\gamma_\lambda^*$. Our intuition is to use these weights to identify the most similar sets of rows and columns that should be grouped together to form co-clusters.

# Optimal Transportation - Proposed Approach

- Following (Benamou et al., 2015), this optimization problem can be rewritten as:

$$\min_{\gamma\in\Pi(\hat{\mu}_r,\hat{\mu}_c)} \langle M,\gamma\rangle_F - \frac{1}{\lambda}E(\gamma) = \frac{1}{\lambda}\min_{\gamma\in\Pi(\hat{\mu}_r,\hat{\mu}_c)} \mathrm{KL}(\gamma\|\xi_\lambda)$$

- where $\xi_\lambda = e^{-\lambda M}$ is the Gibbs kernel. Finally, we can rewrite the last expression as follows:

$$\min_{\gamma\in\Pi(\hat{\mu}_r,\hat{\mu}_c)} \mathrm{KL}(\gamma\|\xi_\lambda) = \min_{\gamma\in\mathcal{C}} \mathrm{KL}(\gamma\|\xi_\lambda)$$

- where $C = C_1 \cap C_2$ is the intersection of closed convex subsets given by $C_1 = \{\gamma \in R^{d\times d} \mid \gamma 1 = \hat{\mu}_r\}$ and $C_2 = \{\gamma \in R^{d\times d} \mid \gamma^T 1 = \hat{\mu}_c\}$. The solution of the entropy regularized optimal transport can be obtained using Sinkhorn-Knopp algorithm and has the following form

$$\gamma_\lambda^* = \mathrm{diag}(\boldsymbol{\alpha})\xi_\lambda \mathrm{diag}(\boldsymbol{\beta})$$

where α and β are the scaling coefficients of the Gibbs kernel $\xi_\lambda$

## Sinkhorn-Knopp algorithm

Notation :

For two probability vectors r and c

$$U(r,c) := \{P \in \mathbb{R}_+^{d\times d} \mid P\mathbf{1}_d = r, P^T\mathbf{1}_d = c\}.$$

We propose a solution for

$$\text{For } \lambda > 0,\ d_M^\lambda(r,c) := \langle P^\lambda, M\rangle, \text{ where } P^\lambda = \underset{P\in U(r,c)}{\operatorname{argmin}} \langle P, M\rangle - \frac{1}{\lambda}h(P).$$

## Sinkhorn-Knopp algorithm

**Algorithm 1** Computation of $\mathbf{d} = [d_M^\lambda(r, c_1), \cdots, d_M^\lambda(r, c_N)]$, using Matlab syntax.

```
**Input** $M, \lambda, r, C := [c_1, \cdots, c_N]$.
$I = (r > 0); r = r(I); M = M(I, :); K = \exp(-\lambda M)$
$u = \texttt{ones}(\texttt{length}(r), N)/\texttt{length}(r);$
$\tilde{K} = \texttt{bsxfun}(\texttt{@rdivide}, K, r)$ % equivalent to $\tilde{K} = \mathbf{diag}(1./r)K$
**while** $u$ changes or any other relevant stopping criterion **do**
    $u = 1./(\tilde{K}(C./(K'u)))$
**end while**
$v = C./(K'u)$
$\mathbf{d} = \texttt{sum}(u. * ((K. * M)v)$
```

## Sinkhorn-Knopp algorithm

```
def sinkhorn(M,mu,nu,lam,iterations):
    d=len(M)
    a=np.ones(d)
    tolerance=1e-10
    Err=np.empty([iterations,2])
    k=np.exp(np.multiply(M,-1*lam))
    # print k
    for i in range(0,iterations):
        new_k=np.matmul(np.transpose(k),a)
        b=np.divide(nu,new_k)
        Err[i][0]=LA.norm(a*np.matmul(k,b)-mu,1)
        a=np.divide(mu,np.matmul(k,b))
        Err[i][1]=LA.norm(b*np.matmul(np.transpose(k),a)-nu,1)
        if max(Err[i][0],Err[i][1]) < tolerance:
            break

    gamma=np.matmul(np.diag(a),k)
    gamma=np.matmul(gamma,np.diag(b))
    return [a,b,gamma]
```

# Kernelized version and Gromov-Wasserstein distance

- We'll look at a kernelized version of our method. We first define two similarity matrices $K_r \in R^{n\times n}$ and $K_c \in R^{d\times d}$ associated to empirical measures $\hat{\mu}_r$ , $\hat{\mu}_c$ .
- Matrices $K_r$ and $K_c$ are defined by calculating the pairwise distances or similarities between rows and columns, respectively, without the restriction of them being positive or calculated based on a proper distance function satisfying the triangle inequality.
- The entropic Gromov-Wasserstein discrepancy is defined as follows:

$$\begin{aligned}\mathbf{GW}(K_r, K_c, \hat{\mu}_r, \hat{\mu}_c) &= \min_{\gamma\in\Pi_{\hat{\mu}_r,\hat{\mu}_c}} \Gamma_{K_r,K_c}(\gamma) - \lambda E(\gamma)\\ &= \min_{T\in\Pi_{\hat{\mu}_r,\hat{\mu}_c}} \sum_{i,j,k,l} L(K_{r_{i,j}}, K_{c_{k,l}})\gamma_{i,j}\gamma_{k,l} - \lambda E(\gamma).\end{aligned}$$

- where γ is a coupling matrix between two similarity matrices and $L : R \times R \rightarrow R_+$ is an arbitrary loss-function, usually the quadratic-loss or Kullback-Leibler divergence.

## Kernelized version and Gromov-Wasserstein distance

- Based on this definition, one may define the problem of the entropic Gromov-Wasserstein barycenters for similarity or distance matrices $K_r$ and $K_c$ as follows:

$$\min_{K,\gamma_r,\gamma_c} \sum_{i=\{r,c\}} \varepsilon_i \Gamma_{K,K_i}(\gamma_i) - \lambda E(\gamma_i) \qquad \text{----(2)}$$

- K is the computed barycenter and $\gamma_r \in \Pi_{\hat{\mu},\hat{\mu}_r}$ , $\gamma_c \in \Pi_{\hat{\mu},\hat{\mu}_c}$ are the coupling matrices that align it with $K_r$ and $K_c$ , respectively.
- $\varepsilon_i$ are the weighting coefficients summing to one, i.e., $\Sigma_{i=\{r,c\}}\varepsilon_i = 1$ that determine our interest in more accurate alignment between $K_r$ and K or $K_c$ and K.
- In (1) we align rows with columns directly, in (3) our goal is to do it via an intermediate representation given by the barycenter K that is optimally aligned with both $K_r$ and $K_c$.

## Kernelized version and Gromov-Wasserstein distance

- In this case, we obtain the solutions $\gamma_r$ and $\gamma_c$ that, similar to previous method and can be decomposed as follows:
- $\gamma_r^* = \text{diag}(\alpha_r)\xi_r \, \text{diag}(\beta_r)$, $\gamma_c^* = \text{diag}(\alpha_c)\xi_c\text{diag}(\beta_c)$, where $\xi_r = e^{-\lambda Mr}$ and $\xi_c = e^{-\lambda Mc}$ are Gibbs kernels calculated between the barycenter and row and column similarity matrices using any arbitrary loss-function L as explained before.
- Finally, based on the analysis presented above, we further use vectors $\beta_r$ and $\beta_c$ to derive row and column partitions.

# Computing GW barycenters

**Notation :**

The simplex of histograms with N bins is $\Sigma_N \overset{\text{def.}}{=} \{p \in \mathbb{R}_N^+ ; \sum_i p_i = 1\}.$

The entropy of $T \in \mathbb{R}_+^{N \times N}$ is defined as $H(T) \overset{\text{def.}}{=} -\sum_{i,j=1}^{N} T_{i,j}(\log(T_{i,j}) - 1)$

The set of couplings between histograms $p \in \Sigma_{N_1}$ and $q \in \Sigma_{N_2}$ is

$$\mathcal{C}_{p,q} \overset{\text{def.}}{=} \{T \in (\mathbb{R}_+)^{N_1 \times N_2} ; \; T\mathbb{1}_{N_2} = p, \; T^\top \mathbb{1}_{N_1} = q\}$$

For any tensor L = $(L_{i,j,k,l})_{i,j,k,l}$ and matrix $(T_{i,j})_{i,j}$ , we define the tensor-matrix multiplication as

$$\mathcal{L} \otimes T \overset{\text{def.}}{=} \Big( \sum_{k,\ell} \mathcal{L}_{i,j,k,\ell} T_{k,\ell} \Big)_{i,j}.$$

## Computing GW barycenters

If the loss function L can be written as

$$L(a,b) = f_1(a) + f_2(b) - h_1(a)h_2(b)$$

*for functions* $(f_1, f_2, h_1, h_2)$*, then, for any* $T \in \mathcal{C}_{p,q}$*,*

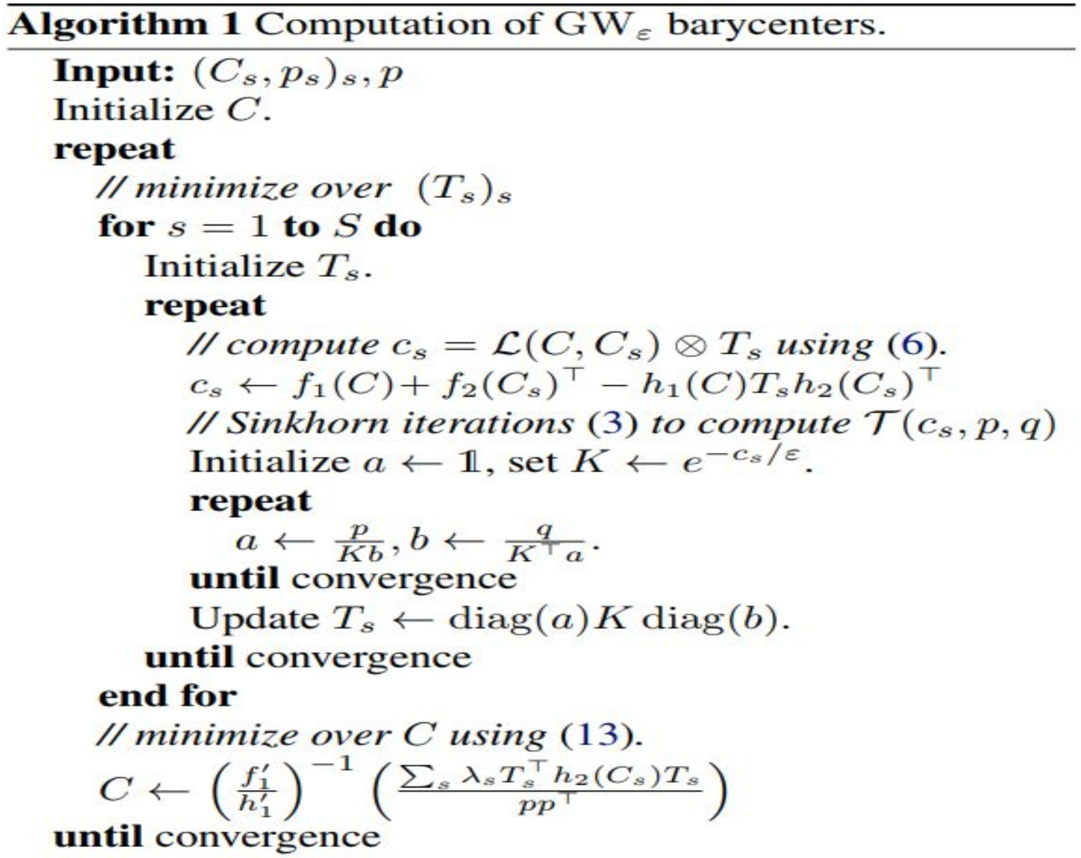
$$\mathcal{L}(C, \bar{C}) \otimes T = c_{C,\bar{C}} - h_1(C) T h_2(\bar{C})^\top. \qquad (6)$$

*where* $c_{C,\bar{C}} \overset{\text{def.}}{=} f_1(C) p \mathbb{1}_{N_2}^\top + \mathbb{1}_{N_1} q^\top f_2(\bar{C})^\top$ *is independent of* $T$.

For square loss $L = L_2$ , $f_1(a) = a^2$ , $f_2(b)=b^2$ , $h_1(a)=a$ , $h_2(b)=2b$ satisfies

For KL loss L = KL , $f_1(a) = a*\log(a) - a$ , $f_2(b)=b$ , $h_1(a)=a$ , $h_2(b)=\log(b)$ satisfies

# Computing GW barycenters



**Algorithm 1** Computation of $\mathrm{GW}_\varepsilon$ barycenters.

```
Input: $(C_s, p_s)_s, p$
Initialize $C$.
repeat
    // minimize over $(T_s)_s$
    for $s = 1$ to $S$ do
        Initialize $T_s$.
        repeat
            // compute $c_s = \mathcal{L}(C, C_s) \otimes T_s$ using (6).
            $c_s \leftarrow f_1(C) + f_2(C_s)^\top - h_1(C) T_s h_2(C_s)^\top$
            // Sinkhorn iterations (3) to compute $\mathcal{T}(c_s, p, q)$
            Initialize $a \leftarrow \mathbb{1}$, set $K \leftarrow e^{-c_s/\varepsilon}$.
            repeat
                $a \leftarrow \frac{p}{Kb}, b \leftarrow \frac{q}{K^\top a}$.
            until convergence
            Update $T_s \leftarrow \mathrm{diag}(a) K \,\mathrm{diag}(b)$.
        until convergence
    end for
    // minimize over $C$ using (13).
    $C \leftarrow \left(\frac{f_1'}{h_1'}\right)^{-1} \left(\frac{\sum_s \lambda_s T_s^\top h_2(C_s) T_s}{pp^\top}\right)$
until convergence
```

If $(C_s)_s$ are positive semi-definite matrices

using $L^2$ loss ,eq(13) becomes

$$C \leftarrow \frac{1}{pp^\top} \sum_s \lambda_s T_s^\top C_s T_s.$$

Using KL loss, eq (13) becomes

$$C \leftarrow \exp\left(\frac{1}{pp^\top} \sum_s \lambda_s T_s^\top \log(C_s) T_s\right)$$

## Jump Detection - Detecting number of Clusters

- In order to detect the steps (or jumps) in the approximated marginals, we use a method described in Matei & Meignen, 2012 for multiscale denoising of piecewise smooth signals.
- It determines the significant jumps in the vectors α and β without knowing their number and location. As the proposed procedure deals with non-decreasing functions, we first sort the values of α and β in the ascending order. Since the procedure is identical for both vectors, we only describe it for the vector α.
- We consider that the elements $\{\alpha_i\}^n_{i=1}$ of α are the local averages of a piecewise continuous function $v : [0, 1] \subset R \rightarrow R$ on the intervals $= I_i^n = [i/n, (i + 1)/n]$ ( defined by the uniform subdivision R of step 1/n of the interval [0, 1]).
- More precisely: $\alpha_i^n = n \int_{Ii} v(t)dt, \quad i = 0, . . . , n - 1.$

## Jump Detection

- The detection strategy is based on the cost function: $F(I_i^n) = \Sigma^{i}_{l=i-1} \; |\alpha^n_{l+1} - \alpha^n_l|$ defined for each interval. Therefore, we get the list of the interval suspicious to contain a jump for the subdivision of order n as follows:

$$L^n = \{i^* \, ; \, i^* = \mathrm{argmax}_i F(I^n_i)\}$$

- This detection should be refined in order to get only significant jumps in our vector α. To this end we use the multi-scale representation of α as in (Harten, 1989) and we perform this detection on each scale. On the first scale, we get a coarse version of α by averaging:

$$\alpha^{n/2}_i = 0.5*(\alpha^n_{2i} + \alpha^n_{2i+1}), \quad i=0, ..., n/2 - 1.$$

## Jump Detection

- Now, by considering the coarse version of α, we obtain a second list $L^{n/2}$ of suspicious intervals as before.
- After that, these two lists merge in the list L jumps as follows: a jump will be considered in the interval $I^{n}_{2i}$ or $I^{n}_{2i+1}$ if the interval $I^{n/2}_{i}$ is also detected as suspicious at the coarse scale. This procedure is iterated $[\log_2 n]$ times and a jump is observed if a chain of detection exists from fine to coarse scales. Finally, the number of clusters is obtained by

$$g = | L_{jumps} | + 1.$$

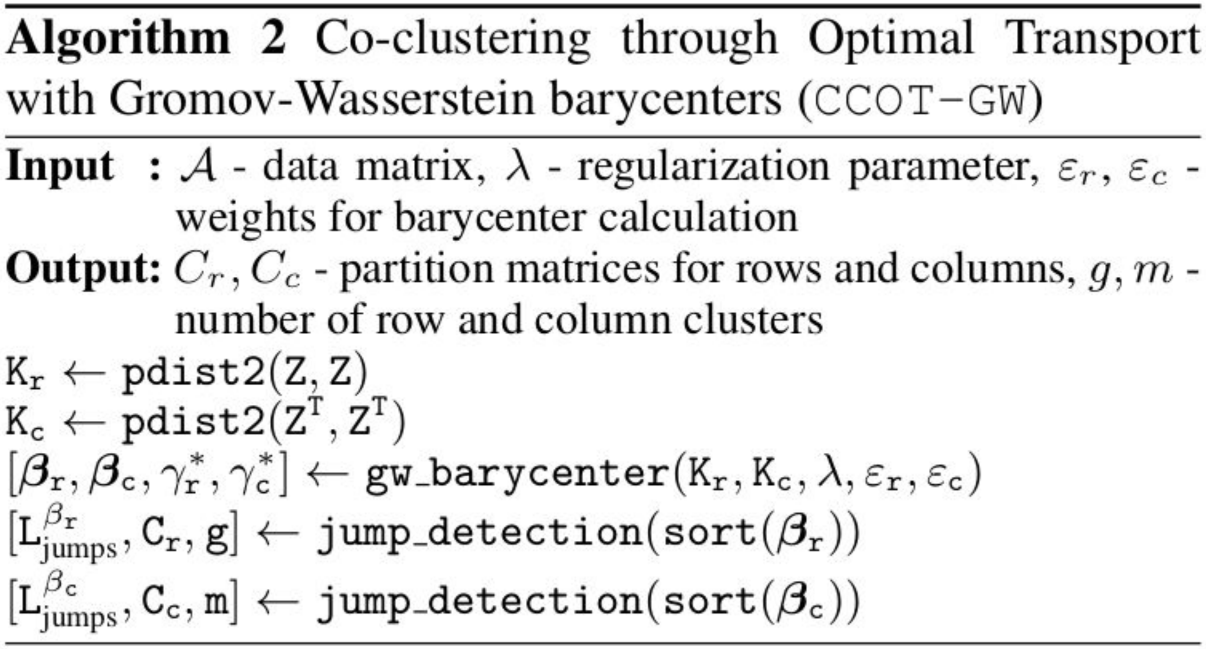
## Algorithmic implementation

**Algorithm 1** Co-clustering through Optimal Transport (CCOT)

```
Input  : $\mathcal{A}$ - data matrix, $\lambda$ - regularization parameter, $n_s$ - number of sampling
Output: $C_r, C_c$ - partition matrices for rows and columns, $g, m$ - number of row and column clusters
$[\mathtt{n}, \mathtt{d}] = \mathtt{size}(\mathtt{Z})$
for $i = 1$ to $n_s$ do
    $\mathtt{D_i} = \mathtt{datasample}(\mathtt{Z}, \mathtt{d})$
    $\mathtt{M_i} \leftarrow \mathtt{pdist2}(\mathtt{D_i}, \mathtt{D_i^T})$
    $[\boldsymbol{\alpha}_\mathtt{i}, \boldsymbol{\beta}_\mathtt{i}, \gamma^*] \leftarrow \mathtt{optimal\_transport}(\mathtt{M_i}, \lambda)$
    $[\mathtt{L}^{\alpha_\mathtt{i}}_{\text{jumps}}, \mathtt{C^i_r}, \mathtt{g}] \leftarrow \mathtt{jump\_detection}(\mathtt{sort}(\boldsymbol{\alpha}_\mathtt{i}))$
    $[\mathtt{L}^{\beta_\mathtt{i}}_{\text{jumps}}, \mathtt{C^i_c}, \mathtt{m}] \leftarrow \mathtt{jump\_detection}(\mathtt{sort}(\boldsymbol{\beta}_\mathtt{i}))$
$\mathtt{C_r} \leftarrow \mathtt{mode}(\mathtt{C^i_r})$
$\mathtt{C_c} \leftarrow \mathtt{mode}(\mathtt{C^i_c})$
```

$$C_r^i(\boldsymbol{x}_r) = \begin{cases} 1, & r \leq L^{\alpha_i}_{\text{jumps}}(1) \\ \dots & \\ k, & L^{\alpha_i}_{\text{jumps}}(k-1) < r \leq L^{\alpha_i}_{\text{jumps}}(k) \\ \dots & \\ |L^{\alpha_i}_{\text{jumps}}| + 1, & r > L^{\alpha_i}_{\text{jumps}}(|L^{\alpha_i}_{\text{jumps}}|). \end{cases}$$

## Algorithmic Implementation



**Algorithm 2** Co-clustering through Optimal Transport with Gromov-Wasserstein barycenters (`CCOT-GW`)

**Input :** $\mathcal{A}$ - data matrix, $\lambda$ - regularization parameter, $\varepsilon_r$, $\varepsilon_c$ - weights for barycenter calculation

**Output:** $C_r, C_c$ - partition matrices for rows and columns, $g, m$ - number of row and column clusters

```
$\mathtt{K_r} \leftarrow \mathtt{pdist2}(\mathtt{Z}, \mathtt{Z})$
$\mathtt{K_c} \leftarrow \mathtt{pdist2}(\mathtt{Z}^\mathtt{T}, \mathtt{Z}^\mathtt{T})$
$[\boldsymbol{\beta}_\mathtt{r}, \boldsymbol{\beta}_\mathtt{c}, \gamma^*_\mathtt{r}, \gamma^*_\mathtt{c}] \leftarrow \mathtt{gw\_barycenter}(\mathtt{K_r}, \mathtt{K_c}, \lambda, \varepsilon_\mathtt{r}, \varepsilon_\mathtt{c})$
$[\mathtt{L}^{\beta_\mathtt{r}}_{\text{jumps}}, \mathtt{C_r}, \mathtt{g}] \leftarrow \mathtt{jump\_detection}(\mathtt{sort}(\boldsymbol{\beta}_\mathtt{r}))$
$[\mathtt{L}^{\beta_\mathtt{c}}_{\text{jumps}}, \mathtt{C_c}, \mathtt{m}] \leftarrow \mathtt{jump\_detection}(\mathtt{sort}(\boldsymbol{\beta}_\mathtt{c}))$
```

## Experiment

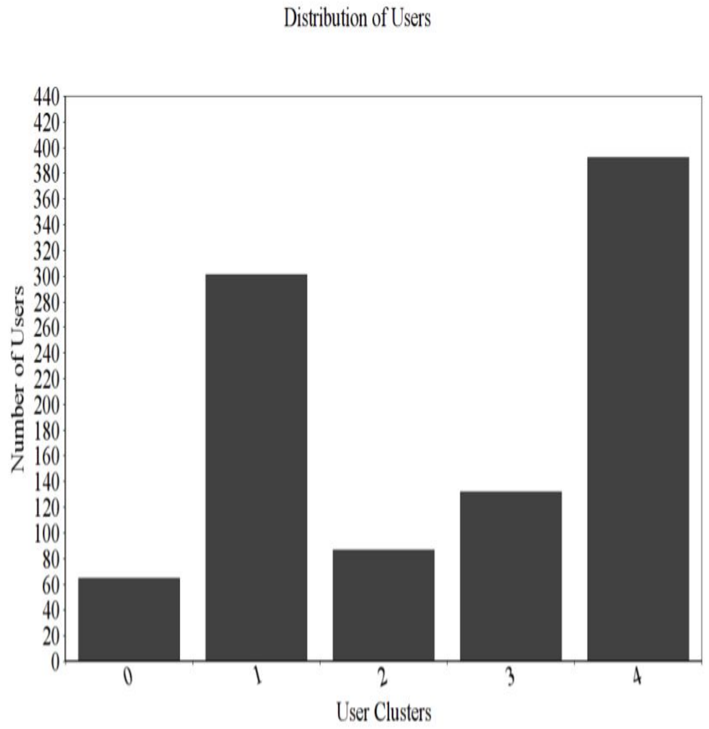
- We used Movie-Lens dataset which is a popular benchmark data set that consists of user-movie ratings, on a scale of one to five, collected from a movie recommendation service gathering 100,000 ratings from 943 users on 1682 movies.
- Our goal is to find homogeneous subgroups of users and films in order to further recommend previously unseen movies that were highly rated by the users from the same group.
- We set the regularisation parameters for the CCOT algorithm as $\lambda = 0.001$ and for CCOT-GW algorithm as $\varepsilon = (0.5,0.5)$.

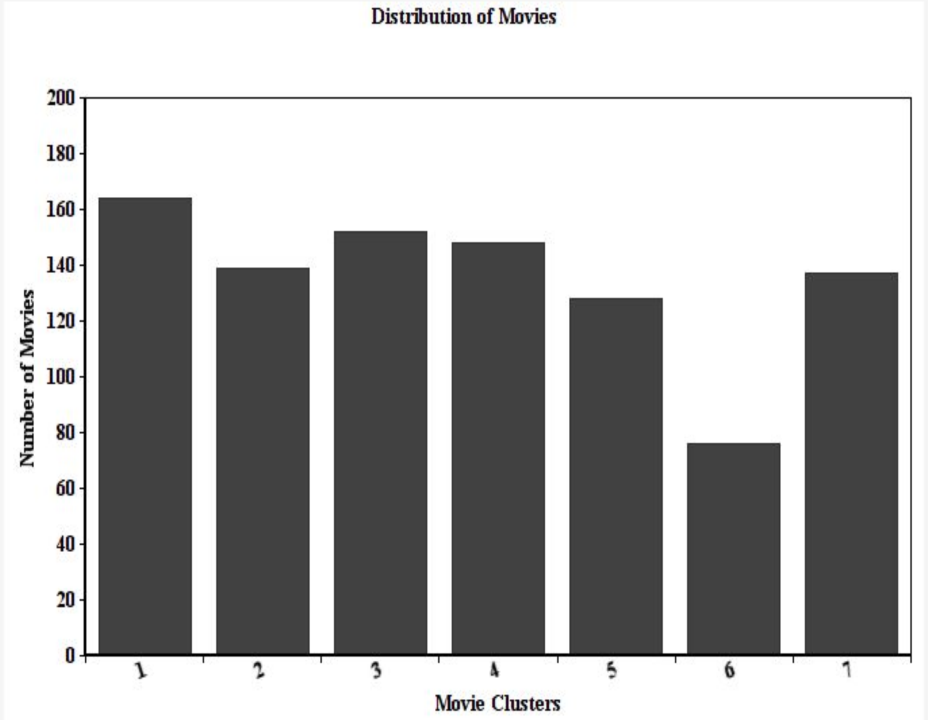
## Results

| MC5 | MC6 |
|---|---|
| Dead man Walking | Diabolique |
| Golden Eye | All Dogs go to Heaven 2 |
| Usual Suspects | Theodore Rex |
| CopyCat | Sgt.Bilko |

In the results we got the movies in the top 4 rated movies in the 5th and 6th clusters are shown. The movies in 5th cluster are similar in type of genre(all are Thriller Movies) and that of in 6th cluster consists of movies which are less critically acclaimed.

# Distribution of Users and Movies in various Clusters





## Source Code

The Source code is available at https://github.com/tarungavara/smai-project

**Thank You**