Swendsen-Wang Cuts in Clustering with Hidden Variable

Linfan Zhang

Abstract

In clustering problems, hidden variables are often introduced as the underlying factor that governing the behavior of the observed data. The EM algorithm is a prevalent method to estimate the model parameters and the hidden variable. However, EM can get stuck in local minimum and requires tractable form in E-step and M-step. In this project, we showed how EM fail in these two cases, and employ Swendsen-Wang Cuts [1] for clustering and learning , showing its superiority over EM under certain circumstance.

1 Introduction

1.1 Model-based Clustering

In model-based clustering, by introducing hidden variable, the problem is equivalent to learning parameters for a mixture model. To be more specific, if we assume that the data $\{X_i \in \mathbb{R}^p, i = 1, \dots, n\}$ originate from k different distributions $p(x|\theta_j)$ with parameter θ_j.By introducing a hidden variable $Z_i = \{Z_{i1} \dots, Z_{ik}\} \overset{i.i.d}{\sim} \text{Multinomial}(1, \{\pi_1, \dots, \pi_k\})$, indicating which cluster each observation is in, we have the complete data likelihood:

$$\ell(X, Z|\Theta, \pi) = \sum_{i=1}^{n} \sum_{j=1}^{k} Z_{ij} \log p(x_i|\theta_j) + \sum_{i=1}^{n} \sum_{j=1}^{k} Z_{ij} \log \pi_{ij} \tag{1}$$

EM algorithm can be applied to learn parameter $\{\Theta, \lambda\}$ by maximizing iteratively the expectation of log complete data likelihood given the current estimation on the parameter and the observed data. However, EM is only guaranteed to produce local optimization. Once the density function doesn't have desired property such as convexity, EM can get stuck on local maximum for the likelihood.

1.2 Stochastic block model

Stochastic block model is another model making use of hidden variables. Given the observed adjacency matrix $\mathbf{X}_{n\times n}$, we assume that there are K underlying clusters governing the connectivity of the n nodes. Again, hidden variable $Z_i = \{Z_{i1} \dots, Z_{ik}\} \overset{i.i.d}{\sim}$ Multinomial$(1, \{\pi_1, \dots, \pi_k\})$, indicates which cluster each node is in, and the edge between two nodes is generated by

$$X_{ij}|Z_{im}, Z_{j\ell} \sim f(\cdot; \gamma_{m\ell}) \tag{2}$$

and the complete data likelihood is:

$$\mathbb{P}(X, Z|\gamma, \pi) = \sum_{i,j}\sum_{m,\ell} Z_{im}Z_{j\ell} \log f(X_{ij}; \gamma_{m\ell}) + \sum_{i}\sum_{m,\ell} Z_{im} \log \pi_m \tag{3}$$

Though E-step of EM algorithm is intractable in this case, variational EM [2] is able to maximize the complete data likelihood.

1.3 Potts Model

The Potts model is widely studied in physics, and can also be reparameterized by hidden variable, and thus helps simplify sampling procedure in MCMC based Swendsen-Wang cuts algorithm [1].

Given graph $\mathcal{G} =< V, E >$, each vertex is associated with a state variable X_i that take K different values. If we observe the edges $U_{ij} \in E$, the likelihood is given by:

$$\mathbb{P}(X_1, \dots, X_n) = \frac{1}{C} \exp\left\{\sum_{U_{ij}\in E} -\beta_{ij} 1\{X_i \neq X_j\}\right\} \tag{4}$$

where C is a constant normalizing parameter. This is usually how Potts model is defined. Applying the technique of hidden variable again, we can introducing latent variable U_{ij} on each edges and make the complete likelihood to be:

$$\mathbb{P}(X, U) = \frac{1}{C'} \prod_{U_{ij}\in E} \Big[(1 - q_{ij})1(U_{ij} = 0) + q_{ij}1(U_{ij} = 1)1(X_i = X_j)\Big] \tag{5}$$

where C' is another normalizing constant and $1 - q_{ij} = e^{-\beta_{ij}}$. We can verify that the marginal distribution $\mathbb{P}(X)$ by integrating U is equal to (4).

$$
\begin{aligned}
\mathbb{P}(X) &\propto \sum_{U=0,\in E}^{1} \cdots \sum_{U=0,\in E}^{1} \prod_{U_{ij}\in E} \Big[(1-q_{ij})1(U_{ij}=0) + q_{ij}1(U_{ij}=1)1(X_i = X_j)\Big] \\
&= \prod_{U_{ij}\in E} \Big(\sum_{U_{ij}=0}^{1} [(1-q_{ij})1(U_{ij}=0) + q_{ij}1(U_{ij}=1)1(X_i = X_j)] \Big) \\
&= \prod_{U_{ij}\in E} \Big[(1-q_{ij}) + q_{ij}1(X_i = X_j)\Big] \\
&= \exp\left\{ \sum_{U_{ij}\in E} -\beta_{ij} 1\{X_i \neq X_j\} \right\}
\end{aligned}
$$

And we can also have the conditional distribution of hidden variable as

$$U_{ij}|Z_i, Z_j \sim \mathrm{Ber}(q_{ij}1\{Z_i = Z_j\}) \tag{6}$$

This result turned out to be very useful when we design the sampling scheme in algorithm that we will discuss below.

2 Swendsen-Wang Cuts for Clustering and Learning

From the above discussion, we can see that EM is hard to generalize to all kinds of distributions for clustering problem with hidden variables. We want to compare EM's performcance with a generative model Swendsen-Wang cut [1], which is a MCMC based sampling method that can be used to sample arbitrary posterior probability. We can monitor the behavior of posterior likelihood of $\mathbb{P}(Z_i|X_i)$ to perform clustering and in the meantime, the model parameters in each clustering can be also learned. We first introduce the Potts model as a type of latent space model and then state the algorithm.

It comprises iterates steps:

(a) data-based adjacency matrix construction;

(b) propose new state for the selected connected component;

(c) accept the proposal.

2.1 Construct adjacency graph

Before we begin, we first check the data form. If the input data only contain n observations, which is common is clustering problem. We need to transform the input to graph $\mathcal{G}$ through K-nearest neighbor. First, we calculated the euclidean distance for each pair

(X_i, X_j). Then for fixed i, construct edge on the pair with k smallest distance. Finally, the graph is regarded as the input of the Swendsen-Wang cuts.

After preprocessing data, we assign label $\{1, \dots, K\}$(assume the number of clusters is known) randomly to each node and then remove edges according to (6). q_{ij} can be designed arbitrarily. But approximate q_{ij} with likelihood can be more informative and thus lead to faster convergence. For nodes in the same cluster, connected components can be constructed and we denote them as $CP_1 = \{V_{11}, \dots, V_{1n_1}\}, \dots, CP_K = \{V_{K1}, \dots, V_{Kn_K}\}$.

2.2 Propose a new cluster

The second step is to select a connected component and change its cluster. Consider the probability of transition from state $\mathbf{X}$ to $\mathbf{X}'$, where we change the cluster of connected component V_0 from ℓ to ℓ_0. It can be factorized as the probability of selecting connected component V_0 and the probability of changing the cluster of V_0 from ℓ to ℓ'

$$\mathbb{P}(\mathbf{X} \to \mathbf{X}') = \mathbb{P}(V_0|\mathbf{X})\mathbb{P}(\mathbf{X}_{V_0=\ell}|V_0, \mathbf{X}) \tag{7}$$

2.3 Accept the proposal

The last step to decide whether to accept this proposal or not. Following Metropolis hastings design, let $\mathcal{C}(V_0, V_\ell) = \{< s, t >: s \in V_0, t \in V_\ell \backslash V_0\}$ and apply the theorem 2 in [1],

$$\begin{aligned}
\alpha(\mathbf{X} \to \mathbf{X}') &= min\left\{1, \frac{\mathbb{P}(\mathbf{X}' \to \mathbf{X})}{\mathbb{P}(\mathbf{X} \to \mathbf{X}')}\frac{\pi(\mathbf{X}')}{\pi(\mathbf{X})}\right\} \\
&= min\left\{1, \frac{\prod_{<i,j>\in\mathcal{C}(V_0,V_{\ell'})}(1-q_{ij})}{\prod_{<i,j>\in\mathcal{C}(V_0,V_\ell)}(1-q_{ij})}\frac{\mathbb{P}(\mathbf{X}_{V_0=\ell}|V_0,\mathbf{X}')}{\mathbb{P}(\mathbf{X}_{V_0=\ell'}|V_0,\mathbf{X})}\frac{\mathbb{P}(\mathbf{X}'|Y)}{\mathbb{P}(\mathbf{X}|Y)}\right\} \\
&= min\left\{1, \frac{\prod_{<i,j>\in\mathcal{C}(V_0,V_{\ell'})}(1-q_{ij})}{\prod_{<i,j>\in\mathcal{C}(V_0,V_\ell)}(1-q_{ij})}\frac{\mathbb{P}(\mathbf{X}_{V_0=\ell}|V_0,\mathbf{X}')}{\mathbb{P}(\mathbf{X}_{V_0=\ell'}|V_0,\mathbf{X})}\frac{\mathbb{P}(Y|\mathbf{X}')\mathbb{P}(\mathbf{X}')}{\mathbb{P}(Y|\mathbf{X})\mathbb{P}(\mathbf{X})}\right\} \\
&= min\left\{1, \frac{\prod_{<i,j>\in\mathcal{C}(V_0,V_{\ell'})}(1-q_{ij})}{\prod_{<i,j>\in\mathcal{C}(V_0,V_\ell)}(1-q_{ij})}\frac{\mathbb{P}(\mathbf{X}_{V_0=\ell}|V_0,\mathbf{X}')}{\mathbb{P}(\mathbf{X}_{V_0=\ell'}|V_0,\mathbf{X})}\frac{\mathbb{P}(X_{V_0}=\ell'|\mathbf{X}_{\partial V_0}))}{\mathbb{P}(X_{V_0}=\ell|\mathbf{X}_{\partial V_0}))}\frac{\mathbb{P}(Y_{V_0}|X_{V_0}=\ell')}{\mathbb{P}(Y_{V_0}|X_{V_0}=\ell)}\right\} \\
&= min\left\{1, \frac{\prod_{<i,j>\in\mathcal{C}(V_0,V_{\ell'})}(1-q_{ij})}{\prod_{<i,j>\in\mathcal{C}(V_0,V_\ell)}(1-q_{ij})}\frac{\prod_{<i,j>\in\mathcal{C}(V_0,V_\ell)}e^{-\beta_{ij}}}{\prod_{<i,j>\in\mathcal{C}(V_0,V_\ell)}e^{-\beta_{ij}}}\right. \\
&\qquad \left.\frac{\mathbb{P}(\mathbf{X}_{V_0=\ell}|V_0,\mathbf{X}')}{\mathbb{P}(\mathbf{X}_{V_0=\ell'}|V_0,\mathbf{X})}\frac{\mathbb{P}(Y_{V_0}|X_{V_0}=\ell')}{\mathbb{P}(Y_{V_0}|X_{V_0}=\ell)}\right\}
\end{aligned}$$

As we stated in the first part we can design $q_{ij} = 1 - e^{-\beta_{ij}}$. Therefore, the final acceptance probability would be

$$\alpha(\mathbf{X} \to \mathbf{X}') = min\left\{1, \frac{\mathbb{P}(\mathbf{X}_{V_0=\ell}|V_0, \mathbf{X}')}{\mathbb{P}(\mathbf{X}_{V_0=\ell'}|V_0, \mathbf{X})} \frac{\mathbb{P}(Y_{V_0}|X_{V_0} = \ell')}{\mathbb{P}(Y_{V_0}|X_{V_0} = \ell)}\right\} \tag{8}$$

In this project, we consider two different designs for assigning new label to V_0. The first one, we call it SWC1, is to assign the new label with equal probability $\mathbb{P}(\mathbf{X}_{V_0=\ell}|V_0, \mathbf{X}') = \frac{1}{K}$, the the accept probability is just the ratio of data likelihood.

$$\alpha(\mathbf{X} \to \mathbf{X}') = min\left\{1, \frac{\mathbb{P}(Y_{V_0}|X_{V_0} = \ell')}{\mathbb{P}(Y_{V_0}|X_{V_0} = \ell)}\right\} \tag{9}$$

The second one, we call it SWC2, is a generalized Gibbs sampler, i.e. accept the new cluster with probability one. This requires the assigning probability proportional to the data likelihood:

$$\frac{\mathbb{P}(\mathbf{X}_{V_0=\ell}|V_0, \mathbf{X}')}{\mathbb{P}(\mathbf{X}_{V_0=\ell'}|V_0, \mathbf{X})} = \frac{\mathbb{P}(Y_{V_0}|X_{V_0} = \ell)}{\mathbb{P}(Y_{V_0}|X_{V_0} = \ell')} \tag{10}$$

The complete algorithm is stated below.

Algorithm 1: Swendsen-Wang Cuts for clustering and learning

```
Input  : $(y_1, \ldots, y_n)$ that need to be clustered, number of clusters $K$,
         underlying data likelihood given clusters, design $q_e$ for the prior Potts
         model
Initialize: transform the data into graph $\mathcal{G} =< V, E >$ by K-NN if needed;
construct data likelihood $\mathbb{P}_l^{(0)}$ with random initialized parameter and given
 labels.
for $l$ = 1:L do
    random labeling each observation $X_i^{(0)} \to \{1, \ldots, K\}$;
    construct data likelihood $\mathbb{P}^{(0)}$ wit h random initialized parameter and given
     labels;
    for $t$ = 1:T do
        Construct $B = \{< i, j >\in E : X_i = X_j\}$;
        For $e \in E$, set $U_e = 0$ with probability $q_e$;
        Collect all the connected components according to labels
         $CP = \Big\{CP_1 = \{V_{11}, \ldots, V_{1n_1}\}, \ldots, CP_K = \{V_{K1}, \ldots, V_{Kn_K}\}\Big\}$;
        Select one connected components with probability $\frac{1}{|CP|}$;
        if propose $\ell'$ to $V_0$ with probability $\frac{1}{K}$;
         then
            accept the proposal with probability defined in (9);
        end
        if propose $\ell'$ to $V_0$ with probability proportional to $\mathbb{P}(Y_{V_0}|X_{V_0} = \ell')$ then
            accept the proposal;
        end
        update labels to $\{X_1^{(t)}, \ldots, X_n^{(t)}\}$;
        update parameters estimate in each cluster (e.g. using MLE) $\mathbb{P}^{(t)}$
    end
    record $\mathbf{X}_l = \{X_1^{(T)}, \ldots, X_n^{(T)}\}$, $\mathbb{P}_l^{(T)}$,$\mathbb{P}_l^{(T)}(y_1, \ldots, y_n)$
end
Output: clustering $(y_1, \ldots, y_n)$ with the largest $\mathbb{P}_l^{(T)}(y_1, \ldots, y_n)$, and the
        parameter estimation is given by $\mathbb{P}_l^{(T)}$
```

3 Simulation Results

3.1 Model-based Clustering

The most naive simulation we use is the dataset-1 in homework 2. We know that with designed initial value close to the true value, EM can converge within 10 steps. However, if our initial is far from the truth, for example, if we set $\mu_1^{(0)} = (10, 3), \mu_2^{(0)} = (0, 0), \mu_3^{(0)} = (5, 4)$ as the initial value, EM will get stuck in local minimum and the misclassification rate is high. On the other hand, Swendsen-Wang cuts jumps out of local minimum and achieves high accuracy rate.(Fig.1) The price we pay for high accuracy is more iteration step.

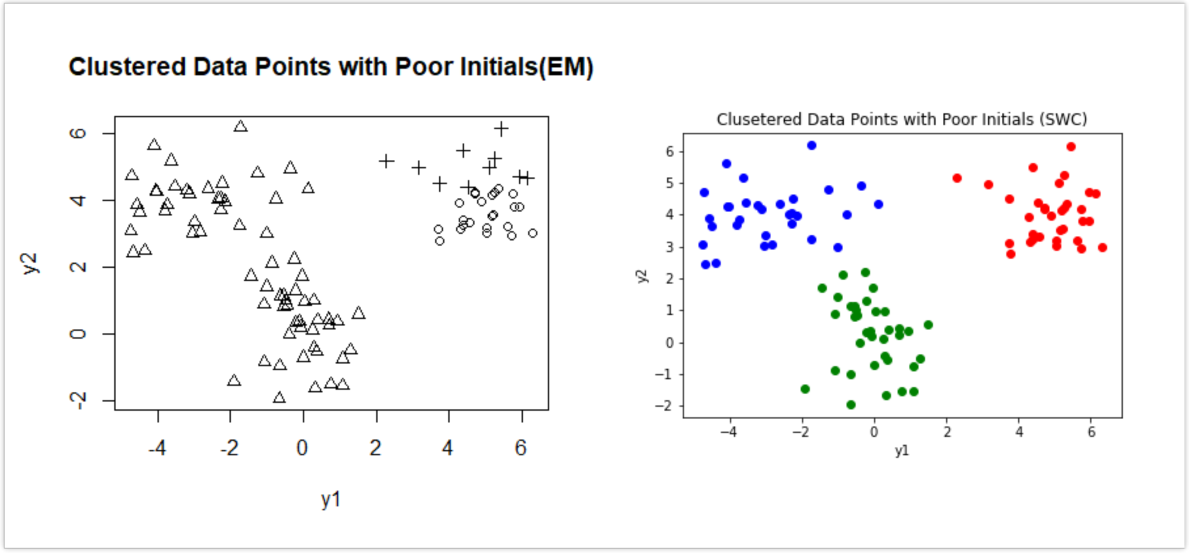

Figure 1: Left: For EM, clustering with poor initials result in poor accuracy; Right: Swendsen-Wang Cuts maintains high accuracy rate with poor start

3.2 Stochastic Block Model

To estimate a stochastic block model, we estimate parameter through variational EM and cluster nodes with community detection. With Swendsen-Wang Cuts, we can simultaneously learn parameters and cluster nodes. In our simulation, we set the true model to be

$$\pi = (0.2, 0.5, 0.3) \qquad Y_{ij} | Z_{im} = 1, Z_{j\ell} = 1 \sim \text{Ber}(\gamma_{m\ell}) \qquad \gamma = \begin{bmatrix} 0.7 & 0.01 & 0.01 \\ 0.01 & 0.5 & 0.02 \\ 0.01 & 0.02 & 0.9 \end{bmatrix}$$

The estimated results with variational EM and Swendsen-Wang Cuts are

$$\hat{\pi}_{V-EM} = (0.24, 0.26, 0.5) \qquad \hat{\gamma}_{V-EM} = \begin{bmatrix} 0.7473 & 0.0085 & 0.0142 \\ 0.0085 & 0.3928 & 0.0281 \\ 0.0142 & 0.0281 & 0.8897 \end{bmatrix}$$

$$\hat{\pi}_{SWC1} = (0.24, 0.54, 0.22) \qquad \hat{\gamma}_{SWC1} = \begin{bmatrix} 0.7424 & 0.0062 & 0 \\ 0.0062 & 0.5157 & 0.0337 \\ 0 & 0.03367 & 0.8727 \end{bmatrix}$$

$$\hat{\pi}_{SWC1} = (0.24, 0.54, 0.22) \qquad \hat{\gamma}_{SWC1} = \begin{bmatrix} 0.7424 & 0.0062 & 0 \\ 0.0062 & 0.5157 & 0.0337 \\ 0 & 0.03367 & 0.8727 \end{bmatrix}$$

$$\hat{\pi}_{SWC2} = (0.24, 0.54, 0.22) \qquad \hat{\gamma}_{SWC2} = \begin{bmatrix} 0.7424 & 0.0062 & 0 \\ 0.0062 & 0.5157 & 0.0337 \\ 0 & 0.03367 & 0.8727 \end{bmatrix}$$

The accuracy rate for clustering is only $(33.33\%, 37.03\%, 90.91\%)$, while the two SWC can cluster the data with 100% accuracy. Though performing community detection separately may improve accuracy, SWC automatically learn and cluster at the same time. However, SWC is not guaranteed to converge to the same value, and that's the reason why we need to include several "iteration cycles" in the algorithm and select the optimal one with data likelihood. Another issue with SWC is that although theoretically we can select any q_e for Potts prior, for example we just set $q_e = 0.8$ in the previous Gaussian clustering setting, q_e still needs to be carefully designed to reflect the connectivity between observations to ensure SWC converges fast. In stochastic block model, SWC would fail if q_e is naively set to be a constant. But if it is set to be the estimated γ in each step, its performance would be much more stable.

4 Real Data Application

4.1 Iris Clustering

Iris is a well test dataset for classification, it can be loaded in Python through *datasets.load_iris()* from *datasets* in *sklearn*. There are 3 classes of 50 observations each with four dimensions. One class can be linearly separated from the other two, but the left two can't be linearly separated. Using only the first two dimensions, SWC can't classify these two very well neither, but with additional information, SWC can classify with high accuracy: only two observations have been misclassified. (Fig.2)

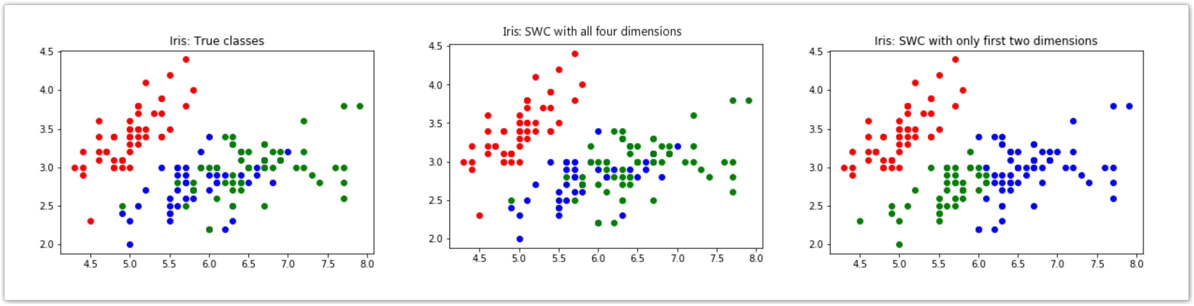

Figure 2: Left: True labels for Iris; Middle: SWC estimates with all four dimensions; Right: SWC estimates with only the first two dimension

4.2 email-Eu-core network

We use data from `https://snap.stanford.edu/data/email-Eu-core.html` to test the performance of SWC in real data. The network was generated using email data from a large European research institution. An edge (u, v) is constructed in the network if person u sent person v at least one email, and there are 1005 nodes and 25571 edges in total. The goal is to label nodes(people) such that the nodes with the same label are from the same department. The true label is given and we assume that it follows $Y_{ij}|Z_{im} = 1, Z_{j\ell} = 1 \sim \text{Ber}(\gamma_{m\ell})$. SWC performs poorly in the sense that the nodes it clustered as a group is not densely connected, but this is true for within each department. (Fig.3) Furthermore, it fails to identify the largest group. For the 109 nodes labeled as 4 in the true largest group, only 6 of them are included in the estimated largest group. (Fig.4)

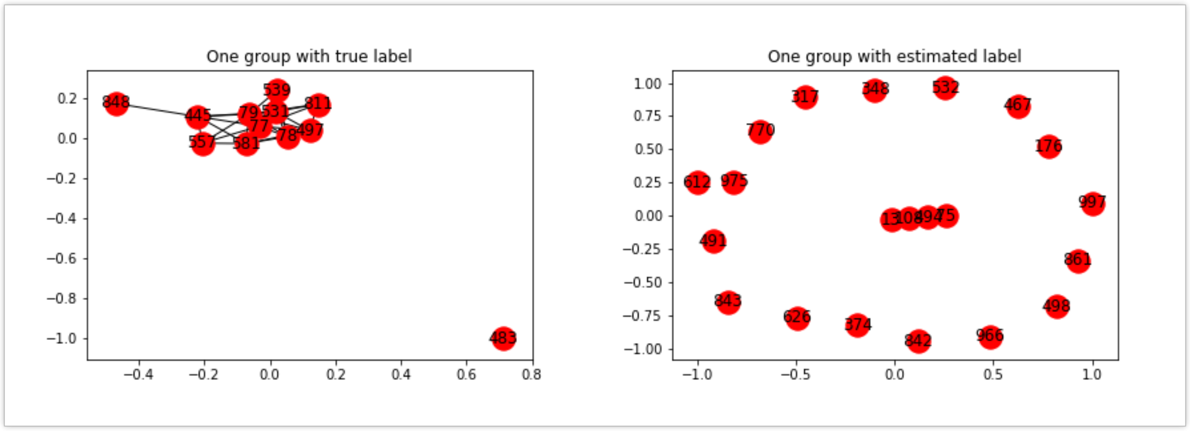

Figure 3: Left: One example of the group in the same department, densely connected; Right: One estimated group using SWC, barely connected

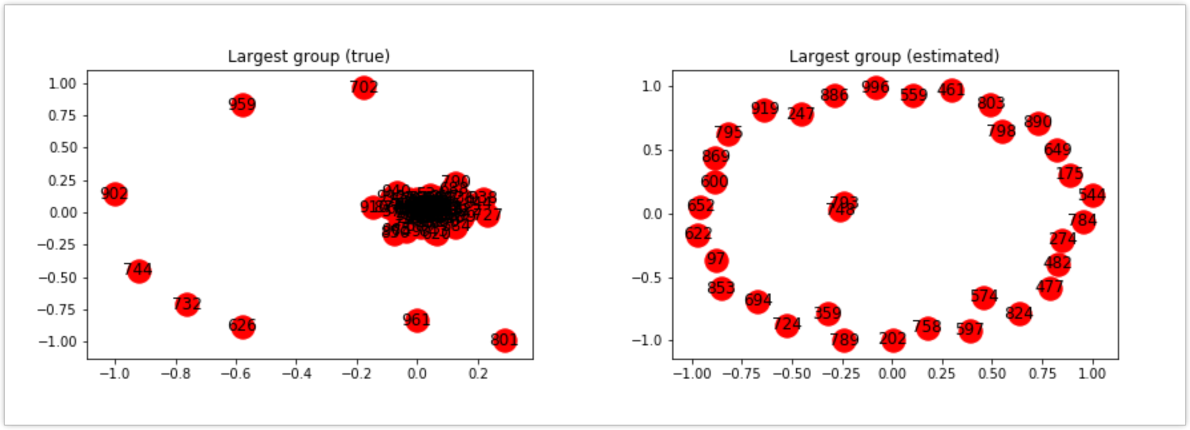

Figure 4: Left: True largest group; Right: Estimated largest group

However, this doesn't imply the failing of SWC in real data. Firstly, it is likely that SWC hasn't converge since we only iterated 2000 times taking 2 hours. But its complexity $O(N^2)$ is still faster than spectral clustering [1]. Another possibility is that the true model is not Bernoulli with respect to γ, some link function (e.g. logistic) may be applied to fix this.

5 Conclusion

In this project, we mainly explore the application of Swendsen-Wang Cuts in the clustering and learning problems with hidden variables. Model-based clustering(Gaussian) and stochastic block model are our focus and both simulation and real data application are performed.

Though EM can converge very fast with proper initials, it may end up in local minimum and produce high classification error. SWC can search the parameter space more exhaustively and find global minimum at the cost of more iteration steps.

EM requires the tractability of E-step and M-step, while SWC just needs correct likelihood identification. Nevertheless, wrong assumption on the likelihood can lead to meaningless estimation. Furthermore, q_e, the probability of turning off edges give graph $\mathcal{G}$ also requires careful design to make SWC work.

The comparison between SWC and EM is the comparison between data-driven and analytical approach in statistical modeling. The advancement in computing allows us to explore more of the data-driven approach, but we still care about the theoretical properties of them. Therefore, further work can focus on the theoretical part of SWC, such as how to design q_e and determine the data likelihood function, to generalize its efficiency in more situations.

Appendex-Python & R Code

Simulation: Model-based Clustering

```
import networkx as nx
import numpy as np
import itertools
import matplotlib.pyplot as plt
from itertools import chain
from sklearn import neighbors
from scipy.stats import multivariate_normal
import os
os.chdir('C:.../201C/project/')

# update parameter:
def para_update(X, pi, mu, sigma, k):
    sigma_new = np.empty([k,2,2])
    mu_new = np.zeros((k, 2))
    for i in range(k):
        mask = pi == i
        idx = np.arange(len(X))[mask]
        nl = sum(mask)
        mu_new[i] = [np.mean(X[mask, 0]), np.mean(X[mask, 1])]
        sigma_e = np.zeros([2,2])
        for j in idx:
            sigma_e += np.dot((X[j, ] -mu[i]).reshape((2,1)), (X[j, ]
            -mu[i]).reshape((1,2)))
        sigma_new[i] = sigma_e/nl
    return mu_new, sigma_new

def likelihood(mu, sigma):
    ll = list()
    for i in range(k):
        ll.append(multivariate_normal(mean=mu[i], cov=sigma[i]))
    return ll

def SWC(G, k, pi, qe, ll):
    G1 = G.copy()
    for e in G.edges(): # remove edges
        if pi[e[0]] == pi[e[1]] and np.random.choice(2, p = [qe, 1-qe]):
            G1.remove_edge(*e)

    V = [None] * k #store CP in each cluster
    CP = list() # store CP as a whole
    for i in range(k):
                # extract subgraph in the same cluster
        G_temp = G1.subgraph(np.array([v for v in G1.nodes()])[pi==i])
        # CP in each cluster
        V[i] = [e for e in nx.connected_components(G_temp)]
        CP = CP + V[i] # construct connected components

    # select a random CP
    idx = np.random.choice(len(CP))
    V0 = CP[idx]

    # propose probability
    q = np.zeros(k)
    for i in range(k):
        pi_temp = pi.copy()
        pi_temp[list(V0)] = i
        # data likelihood
        for j in range(n):
            q[i] += np.log(ll[pi_temp[j]].pdf(X[j, ]))
    q = np.exp(q)
    q = q/sum(q)

    pi_new = pi.copy()
    label_new = np.random.choice(k, p=q)
    pi_new[list(V0)] = label_new
    return pi_new, q[label_new]


qe = 0.5
X = np.loadtxt('Dataset-1.txt')
n = len(X)

k = 3 # number of clusters
temp = neighbors.kneighbors_graph(X, 4, mode='distance')
G = nx.from_numpy_matrix(temp.toarray())
```

```
# initialize cluster
pi = np.random.choice(np.arange(k), n)
# initialize data likelihood function:
mu = np.array([(10,3), (2,2), (3,1)])
sigma = np.array([np.diag([1,1]), np.diag([1,1]), np.diag([1,1])])
ll = likelihood(mu, sigma)

pi_max = np.array([])
prob_max = 0
for l in range(50):
    for i in range(100):
        pi, prob = SWC(G, k, pi, 0.8, ll)

        mu, sigma = para_update(X, pi, mu, sigma, k)
        ll = likelihood(mu, sigma)

    if prob > prob_max:
            prob_max = prob
            pi_max = pi

fig = plt.figure()
plt.plot(X[pi_max == 0, 0], X[pi_max == 0,1],'ro', color="red")
plt.plot(X[pi_max == 1, 0], X[pi_max == 1,1],'ro', color="blue")
plt.plot(X[pi_max == 2, 0], X[pi_max == 2,1],'ro', color="green")
plt.xlabel('y1')
plt.ylabel('y2')
plt.title('Clusetered Data Points with Poor Initials (SWC)')
fig.savefig('2-SWC-cluster.png')
```

Simulation: Stochastic Block Model

```
import networkx as nx
import numpy as np
import itertools
import matplotlib.pyplot as plt
from itertools import chain
from sklearn import neighbors
import os
os.chdir('C:...201C/project/')
```

```
def SWC(G, k, pi, gamma):
    G1 = G.copy()
    for e in G.edges(): # remove edges
        qe = gamma[pi[e[0]], pi[e[1]]]
        if pi[e[0]] == pi[e[1]] and np.random.choice(2, p = [qe, 1-qe]):
            G1.remove_edge(*e)

    V = [None] * k #store CP in each cluster
    CP = list() #store CP as a whole
    for i in range(k):
                # extract subgraph in the same cluster
        G_temp = G1.subgraph(np.array([v for v in G1.nodes()])[pi==i])
        # CP in each cluster
        V[i] = [e for e in nx.connected_components(G_temp)]
        CP = CP + V[i] # construct connected components

    # select a random CP
    idx = np.random.choice(len(CP))
    V0 = CP[idx]
    label_old = pi[list(V0)[0]]
    # propose a new cluster to V0
    #while(True):
    label_new = np.random.choice(k)
        #if label_new != label_old:
         #    break

    q = 1
    pi_new = pi.copy()
    pi_new[list(V0)] = label_new
    #pp = 1
    # proposal
    for s in range(n):
        for t in range(n):
            if t > s:
#                   if gamma[pi_new[s],pi_new[t]] == 0: # if the proposed state
#                       return pi
                p_new = gamma[pi_new[s],pi_new[t]]**X[s,t] *
                (1-gamma[pi_new[s],pi_new[t]])**(1-X[s,t])
                p_old = gamma[pi[s],pi[t]]**X[s,t]
                * (1-gamma[pi[s],pi[t]])**(1-X[s,t])
                q *= (p_new/p_old)

    alpha = min(1,q)


    if alpha > np.random.random(1):
        return pi_new
    else:
        return pi

# update sigma:
def para_update(X, pi, k):
    gamma = np.zeros([k,k])
    count = np.zeros([k,k])
    for i in range(n):
        for j in range(n):
            if j > i: # because of the symmetry
                m = pi[i]
                l = pi[j]
                count[m,l] += 1
                count[l,m] += 1
                if X[i,j] == 1:
                    gamma[m,l] += 1
                    gamma[l,m] += 1
    new_gamma = gamma/count
    new_gamma[np.isnan(new_gamma)] = 0
    return new_gamma

#### simulation set up####
#k = 3 # number of clusters
#p_t = np.array([0.2, 0.5, 0.3])
#gamma_t = np.array([(0.7, 0.01, 0.01), (0.01, 0.5, 0.02),
 (0.01, 0.02, 0.9)])
#X = np.zeros([50,50])
#n = len(X)
#pi_t = np.random.choice(k, size=n, p=p_t)
#for i in range(n):
#    for j in range(n):
#        if j > i: # because of the symmetry
#            temp = gamma_t[pi_t[i], pi_t[j]]
#                X[i,j] = np.random.choice(2, p = [1-temp, temp])
X = np.loadtxt('SBM.txt')
pi_t = np.loadtxt("true_label.txt")
pi_t = pi_t-1
pi_t=np.array(list(map(int, pi_t)))
n = len(X)
k = 3

G = nx.from_numpy_matrix(X)
# initialize cluster
pi = np.random.choice(k, n)
# initialize connectivity
gamma = np.array([(0.5, 0.1, 0.1), (0.1, 0.5, 0.1), (0.1, 0.1, 0.5)])

for i in range(1000):
    pi = SWC(G, k, pi, gamma)
    gamma = para_update(X, pi, k)

rr0 = 0
rr1 = 0
rr2 = 0

for i in range(len(pi)):
    if pi_t[i] == 1:
        rr1 += (pi[i] == 0)
    if pi_t[i] == 2:
        rr2 += (pi[i] == 1)
    if pi_t[i] == 0:
        rr0 += (pi[i] == 2)
rr0/sum(pi_t == 0)
rr1/sum(pi_t == 1)
rr2/sum(pi_t == 2)

sum(pi==2)/len(pi_t) #pi_0
sum(pi==0)/len(pi_t) #PI_1
sum(pi==1)/len(pi_t) #PI_2

library(blockmodels)
k = 3 # number of clusters
p_t = c(0.2, 0.5, 0.3)
```

```
gamma_t = matrix(c(0.7, 0.01, 0.01, 0.01, 0.5, 0.02, 0.01, 0.02, 0.9), byrow
X = matrix( rep( 0, len=50*50), nrow = 50)
n = nrow(X)
pi_t = sample(k, size=n, p=p_t, replace = T)
for (i in 1:n){
    for (j in 1:n){
        if (j > i){# because of the symmetry
            temp = gamma_t[pi_t[i], pi_t[j]]
            X[i,j] = sample(c(0,1), size = 1, p = c(1-temp, temp))
        }
    }
}

write.table(X, file = "SBM.txt", row.names=F, col.names = F)
write.table(pi_t, file = "true_label.txt", row.names=F, col.names = F)

X[lower.tri(X)] = t(X)[lower.tri(X)]

my_model = BM_bernoulli("SBM_sym",X)
my_model

pi = my_model$memberships[[3]]$map()
rr0 = 0
rr1 = 0
rr2 = 0

for(i in 1:length(pi$C)){
    if(pi_t[i] == 1)
        rr0 = rr0 + (pi$C[i] == 2)
    if(pi_t[i] == 2)
        rr1 = rr1 + (pi$C[i] == 1)
    if(pi_t[i] == 3)
        rr2 = rr2 + (pi$C[i] == 3)
}

##accuracy
rr0/sum(pi_t == 1)
rr1/sum(pi_t == 2)
rr2/sum(pi_t == 3)
##pi
```

```
sum(pi$C == 2)/length(pi_t) #pi1
sum(pi$C == 1)/length(pi_t) #pi2
sum(pi$C == 3)/length(pi_t) #pi3
round(my_model$`.->model_parameters`[[3]]$pi,4)
```

Real data application codes are similar to the simulation, so we don't include them here to avoid abundance.

References

[1] Adrian Barbu and Song-Chun Zhu. Generalizing swendsen-wang to sampling arbitrary posterior probabilities. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(8):1239–1253, August 2005.

[2] J.-J. Daudin, F. Picard, and S. Robin. A mixture model for random graphs. *Statistics and Computing*, 18(2):173–183, Jun 2008.