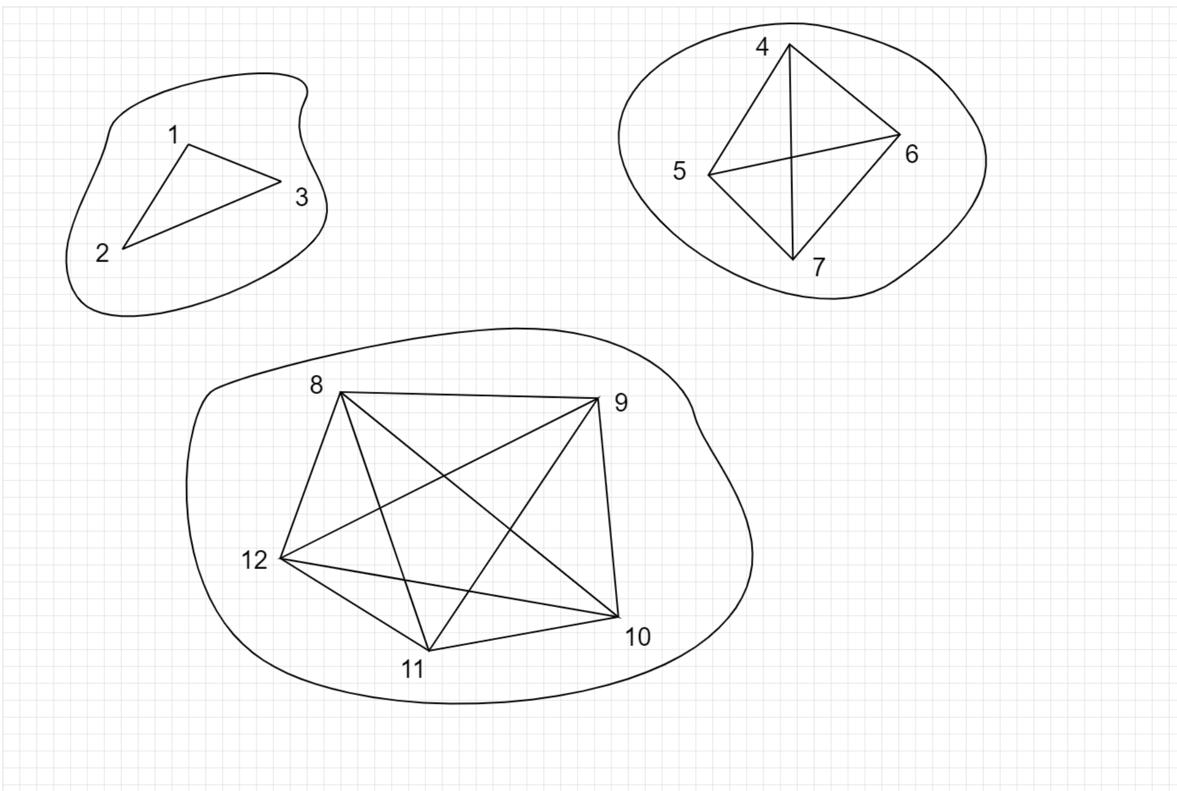
$In\ the\ Name\ of\ God$

$Probability\ and\ Statistics\ Project$

$Phase 1\ Section 2$

$Theory\ Section\ 1:$

In this case, Adjacency matrix would be Symmetric with 0 in it's main diagonal. An example would make this crystal clear, Consider:



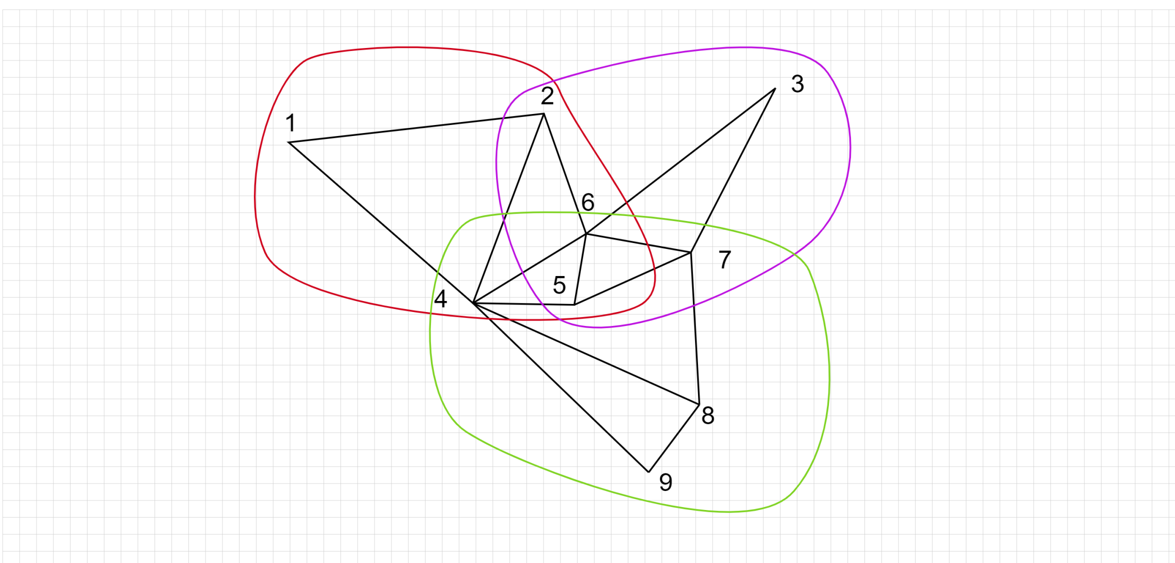
Intuitively, 3 clusters and 12 people(points) are defined and each cluster's people have nothing in common with people in other clusters. We derive the adjaceny matrix of the indicated graph blindly:

$$\begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{pmatrix}$$

Since clusters are compeletly separate, it seems that three adjaceny matrix of these separate clusters are connected diagonally to makr a bigger matrix.

$Theory\ Section\ 2:$

In this case, Adjacency matrix would be Symmetric with 0 in it's main diagonal again. An example would make this crystal clear, Consider:



Intuitively, 3 clusters and 12 people(points) are defined and each cluster's people have Something in common with people in other clusters. We derive the adjaceny matrix of the indicated graph blindly:

$$\begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

We observe that the adjaceny matrix is symmetric, But as we expected it is not made by connecting three separate adjaceny matrices.

$Theory\ Section\ 3:$

u , v -> People c -> community(cluster)
$P(u,v) = 1 - \Pi_{c\in M}(1 - P_c(u,v)) = 1 - \Pi_i^M(1 - P_i)$
$--> P(u,v) = 1 - (1 - P_1)(1 - P_2)\ldots(1 - P_M) = 1 - \Pi_{i=1}^M(1 - P_i) = 1 - A$

When two people are members of many special communities , $P_i(i \in M)$ increses , $1 - P_i(i \in M)$ decreases, thus A also decreases and P(u,v) = 1 - A increases accordingly.

Intersection => $P_i(i \in M) \uparrow$ => $1 - P_i(i \in M) \downarrow$ => $1 - \Pi_{i=1}^M(1 - P_i) \uparrow$ => $p(u,v) \uparrow$

Consequently , people's connection get wider as they become members of different particular communities. Intuitively, Being at the intersection of relationships causes more connection between people. Deficiency of this model is mostly due to being an all-or-nothing approach (0 or 1)! (i.e) There is no defined parameter which can describe relativity of people (Having a same taste), so people in a cluster are either connected to each other(1) or not connected to each other(0 - being isolated). However, in a real world having the same taste is relative.(Not 0 or 1)

$Theory\ Section\ 4:$

$$P(u,v) = 1 - \Pi_{c\in M}(1 - P_c(u,v))M = c_1, c_2, \ldots, c_n$$

We khonw that :
$P_c u, v = 1 - e^{-F_{uc}F_{vc}} => P(u,v) = 1 - \Pi_{c\in M}(1 - (1 - exp(-F_{uc}Fvc)))$

$P(u,v) = 1 - \Pi_{c\in M}exp(-F_{uc}F_{vc}) = 1 - exp(\sum_{c\in M} F_{uc}Fvc)$
(Definition of inner product)
$P(u,v) = 1 - exp(-F_u^T F_v)$

$Theory\ Section\ 5:$

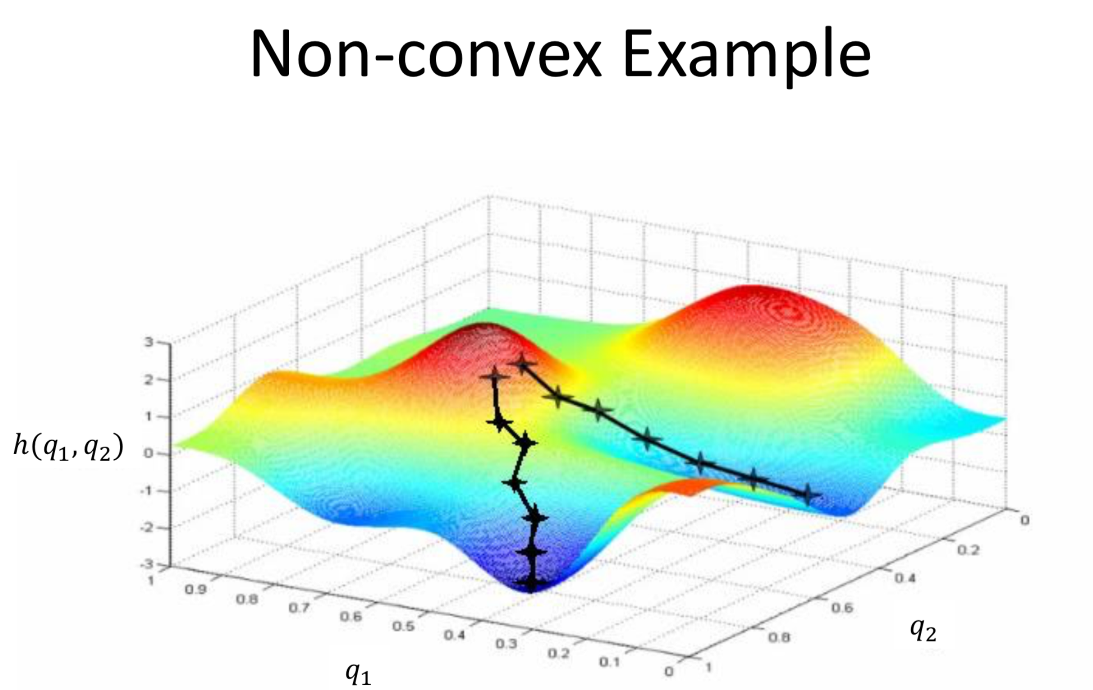
According to the definition of likelehood function, we have:
$P(A|F) = \Pi_{(u,v)\in A} p(u,v) \Pi_{(u,v)\notin A}(1-p(u,v))$

Note that in Theory Section 4 we calculated P(u,v) in terms of F matrix elements:
$P(u,v) = 1 - exp(-F_u^T F_v)$

Thus we obtain:
$l(F) = log(P(A|F)) = log(\Pi_{(u,v)\in A} p(u,v) \Pi_{(u,v)\notin A}(1-p(u,v)))$
$= \sum_{(u,v)\in A} log(P(u,v)) + \sum_{u,v)\notin A} log(1-P(u,v))$

$Theory\ Section\ 6:$

From calculus 2, we remember that the gradient of a fenced field is a vector that its components illustrate the changing rate of the field in different directions. The direction of greadient vector field is the the maxmimum changing rate diection ever found. Gradient field's vector shows a path with maximum changing rate(either positive or negative) but after a while, changing rate decreases and the multiple variable function arrives at its extreme point. This issue is so unpleasant! Beacause by following the gradient of a specific F , after some iterations, we can reach the optimized value of F (F* = argmax(l(F))).



$Theory\ Section\ 7:$

Assume N(u) is the set of neighbouring points of u:

$l(F_u) = \sum_{v \in N(u)} log(1 - e^{(-F_u F_v^T)}) - \sum_{v \notin N(u)} F_u F_v^T$

$\nabla l(F_u) = \sum_{v \in N(u)} F_v \frac{exp(-F_u F_v^T)}{1-exp(-F_u F_v^T)} - \sum_{v \notin N(u)} F_v$

$Simulating\ Section\ 1:$

$l(F) = \sum_{(u,v) \in A} log(1 - e^{-F_u F_v^T}) - \sum_{(u,v) \notin A} F_u F_v^T$

In [ ]:
```
import numpy as np


def log_likelihood(F, A):


    B = F.dot(F.T)  #  F . F(Transpose)

    neighbouring_part = A*np.log(1.-np.exp(-1.*B)) # matrix multiplication
    sum_Neighbours = np.sum(neighbouring_part)
    notNeighbouring_part = (1-A)*B # matrix multiplication
    sum_notNeighbours = np.sum(notNeighbouring_part)


    log_likelihoodEstimation = sum_Neighbours - sum_notNeighbours
    return log_likelihoodEstimation

def gradient(F, A, i):

    rowF, columnF = F.shape

    myNeighbours = np.where(A[i])
    notNeighbours = np.where(1-A[i])

    sumNeighbours = np.zeros((columnF,))
    for neighbor in myNeighbours[0]:
        B = F[neighbor].dot(F[i])
        sumNeighbours += F[neighbor]*(np.divide(np.exp(-1.*B),1.-np.exp(-1.*B)))


    sumNotNeighbour = np.zeros((columnF,))

    for NotNeighbor in notNeighbours[0]:
        sumNotNeighbour += F[NotNeighbor]

    Gradient = sumNeighbours - sumNotNeighbour
    return Gradient
```

```
def train(A, C, iterations = 100):
    # initialize an F
    N = A.shape[0]
    F = np.random.rand(N,C)

    for n in range(iterations):
        for person in range(N):
            grad = gradient(F, A, person)
            F[person] += 0.005*grad                    # updating F
            F[person] = np.maximum(0.001, F[person]) # F should be nonnegative
        ll = log_likelihood(F, A)
        print('At step %4i logliklihood is %5.4f'%(n,ll))

    return F
```

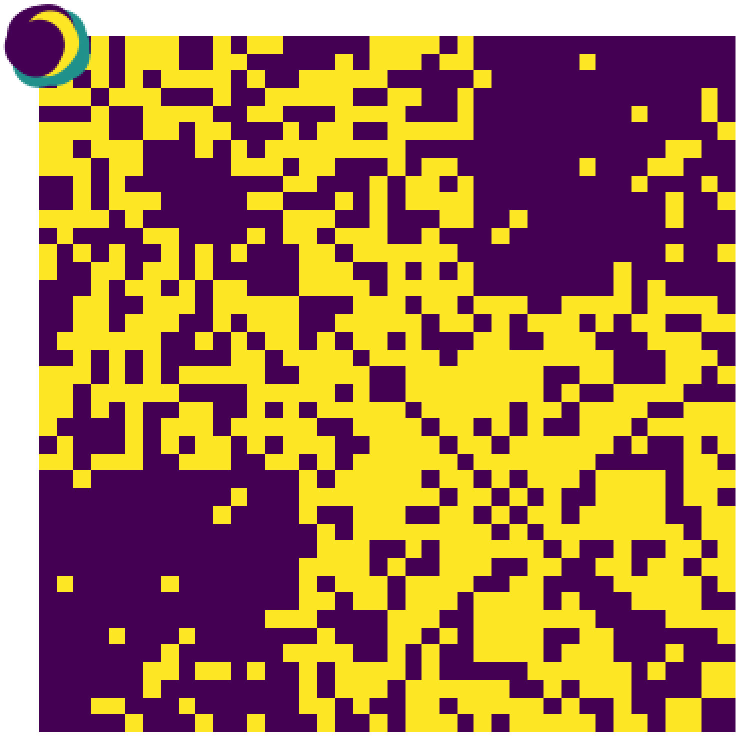
In [ ]:

```
import numpy as np
#testing in two small groups
A=np.random.rand(40,40)
A[0:15,0:25]=A[0:15,0:25]>1- 0.6 # connection prob people with 1 common group
A[0:15,25:40]=A[0:15,25:40]>1-0.1 # connection prob people with no common group
A[15:40,25:40]=A[15:40,25:40]>1-0.7 # connection prob people with 1 common group
A[15:25,15:25]=A[15:25,15:25]>1-0.8 # connection prob people with 2 common group
for i in range(40):
    A[i,i]=0
    for j in range(i):
        A[i,j]=A[j,i]

import matplotlib.pyplot as plt
import networkx as nx
plt.imshow(A)
delta=np.sqrt(-np.log(1-0.1)) # epsilon=0.1
F=train(A, 2, iterations = 120)
print(F>delta)
G=nx.from_numpy_matrix(A)
#G=nx.from_numpy_array(A)
C=F>delta # groups members
nx.draw(G,node_color=10*(C[:,0])+20*(C[:,1])) ## Because of version of networkx probab
```

```
At step    0 logliklihood is -1136.4110
At step    1 logliklihood is -1105.1729
At step    2 logliklihood is -1083.5183
At step    3 logliklihood is -1067.2899
At step    4 logliklihood is -1054.7426
At step    5 logliklihood is -1044.8897
At step    6 logliklihood is -1037.0735
At step    7 logliklihood is -1030.8146
At step    8 logliklihood is -1025.7481
At step    9 logliklihood is -1021.5906
At step   10 logliklihood is -1018.1202
At step   11 logliklihood is -1015.1607
At step   12 logliklihood is -1012.5680
At step   13 logliklihood is -1010.2202
At step   14 logliklihood is -1008.0109
At step   15 logliklihood is -1005.8445
At step   16 logliklihood is -1003.6330
At step   17 logliklihood is -1001.3571
At step   18 logliklihood is -998.9340
At step   19 logliklihood is -996.2789
At step   20 logliklihood is -993.3205
At step   21 logliklihood is -989.9860
At step   22 logliklihood is -986.2008
At step   23 logliklihood is -981.8888
At step   24 logliklihood is -976.9730
At step   25 logliklihood is -971.4502
At step   26 logliklihood is -965.4143
At step   27 logliklihood is -959.5312
At step   28 logliklihood is -953.8284
At step   29 logliklihood is -948.0214
At step   30 logliklihood is -941.9773
At step   31 logliklihood is -935.8802
At step   32 logliklihood is -930.0341
At step   33 logliklihood is -924.2183
At step   34 logliklihood is -918.8230
At step   35 logliklihood is -914.0473
At step   36 logliklihood is -909.4985
At step   37 logliklihood is -905.2158
At step   38 logliklihood is -901.2922
At step   39 logliklihood is -897.8512
At step   40 logliklihood is -894.8209
At step   41 logliklihood is -892.0553
At step   42 logliklihood is -889.5274
At step   43 logliklihood is -887.2165
At step   44 logliklihood is -885.5151
At step   45 logliklihood is -884.1152
At step   46 logliklihood is -882.9018
At step   47 logliklihood is -881.8462
At step   48 logliklihood is -880.9228
At step   49 logliklihood is -880.1032
At step   50 logliklihood is -879.3706
At step   51 logliklihood is -878.7115
At step   52 logliklihood is -878.1144
At step   53 logliklihood is -877.5682
At step   54 logliklihood is -877.0743
At step   55 logliklihood is -876.6315
At step   56 logliklihood is -876.2219
At step   57 logliklihood is -875.8385
At step   58 logliklihood is -875.4753
At step   59 logliklihood is -875.1275
```

```
At step  60 logliklihood is -874.7908
At step  61 logliklihood is -874.4755
At step  62 logliklihood is -874.2113
At step  63 logliklihood is -873.9710
At step  64 logliklihood is -873.7464
At step  65 logliklihood is -873.5342
At step  66 logliklihood is -873.3318
At step  67 logliklihood is -873.1367
At step  68 logliklihood is -872.9470
At step  69 logliklihood is -872.7608
At step  70 logliklihood is -872.5764
At step  71 logliklihood is -872.3923
At step  72 logliklihood is -872.2071
At step  73 logliklihood is -872.0197
At step  74 logliklihood is -871.8568
At step  75 logliklihood is -871.7093
At step  76 logliklihood is -871.5660
At step  77 logliklihood is -871.4252
At step  78 logliklihood is -871.3097
At step  79 logliklihood is -871.2057
At step  80 logliklihood is -871.1087
At step  81 logliklihood is -871.0177
At step  82 logliklihood is -870.9316
At step  83 logliklihood is -870.8498
At step  84 logliklihood is -870.7716
At step  85 logliklihood is -870.6966
At step  86 logliklihood is -870.6242
At step  87 logliklihood is -870.5539
At step  88 logliklihood is -870.4854
At step  89 logliklihood is -870.4183
At step  90 logliklihood is -870.3523
At step  91 logliklihood is -870.2870
At step  92 logliklihood is -870.2223
At step  93 logliklihood is -870.1578
At step  94 logliklihood is -870.0933
At step  95 logliklihood is -870.0285
At step  96 logliklihood is -869.9633
At step  97 logliklihood is -869.9034
At step  98 logliklihood is -869.8559
At step  99 logliklihood is -869.8137
At step 100 logliklihood is -869.7756
At step 101 logliklihood is -869.7411
At step 102 logliklihood is -869.7098
At step 103 logliklihood is -869.6813
At step 104 logliklihood is -869.6553
At step 105 logliklihood is -869.6314
At step 106 logliklihood is -869.6095
At step 107 logliklihood is -869.5894
At step 108 logliklihood is -869.5709
At step 109 logliklihood is -869.5538
At step 110 logliklihood is -869.5380
At step 111 logliklihood is -869.5233
At step 112 logliklihood is -869.5098
At step 113 logliklihood is -869.4972
At step 114 logliklihood is -869.4856
At step 115 logliklihood is -869.4747
At step 116 logliklihood is -869.4647
At step 117 logliklihood is -869.4553
At step 118 logliklihood is -869.4465
At step 119 logliklihood is -869.4383
```

```
[[False  True]
 [False  True]
 [False  True]
 [False  True]
 [False  True]
 [False  True]
 [False  True]
 [False  True]
 [False  True]
 [False  True]
 [False  True]
 [False  True]
 [False  True]
 [False  True]
 [False  True]
 [ True  True]
 [ True  True]
 [ True  True]
 [ True  True]
 [ True  True]
 [ True  True]
 [ True  True]
 [ True  True]
 [ True  True]
 [ True  True]
 [ True False]
 [ True False]
 [ True False]
 [ True False]
 [ True False]
 [ True False]
 [ True False]
 [ True False]
 [ True False]
 [ True False]
 [ True False]
 [ True  True]
 [ True False]
 [ True False]
 [ True False]]
```



$Theory\ Section\ 8:$

The objective here is to model the connection probability between a pair of nodes based on the similarity in their learned affiliations towards communities. To do this, individual nodes are connected with communities with some number of links, with more links from a node to a community indicating that the node has a higher 'affilation' to that group. For a network with N nodes and c communities, the affiliation between nodes and communities is encoded by a matrix, F, where Fuc is the learned count of links (again encoding the affiliation), between node i and community c. Similarly, let Fu and Fv be the community affiliations for nodes u and v. Then the probability that an edge exists between nodes u and v, or P(Auv = 1) is modeled as

$P(A_{uv} = 1) = 1 - exp(-F_u F_v^T)$

The node to community affiliations can be used as a proxy for the total amount of interaction between a pair of nodes u and v with a Poisson distribution. This modeling paradigm will allow for the straightforward modeling of the probability that an edge exists between the node pair. To do this, the total amount of interaction between nodes u and v is modeled as,
$X_{uv}^{(c)} \sim Poisson(\lambda = F_{uc}F_{vc}) => X_{uv} = \sum_c X_{uv}^{(c)}$

Note that we also know how the sum of Poisson random variables are distributed:
$X_{uv} \sim Poisson(\sum_c F_{uc}F_{vc}$

Thus:

$P(X_{uv} > 0) = 1 - P(X_{uv} = 0) = 1 - exp(-\sum_c F_{uc}F_{vc})$

log-likelihood function and its gradient can be derived similar to Theory section 7:

$l(F_u) = \sum_{v \in N(u)} log(1 - e^{(-F_u F_v^T)}) - \sum_{v \notin N(u)} F_u F_v^T$

$\nabla l(F_u) = \sum_{v \in N(u)} F_v \frac{exp(-F_u F_v^T)}{1-exp(-F_u F_v^T)} - \sum_{v \notin N(u)} F_v$

- Helped by

  -ADAPTING COMMUNITY DETECTION APPROACHES TO LAGRE,MULTILAYER, AND ATTRIBUTED NETWORKS(Natalie Stanley)

بخش۳ :گناه بخت پریشان و دست کوته ماست!

پرسش تئوری ۹

ماتریس A ماتریسی است که درایه های آن به صورت تصادفی هستند به این صورت که اگر دو فرد عضو یک خوشه باشند احتمال وجود رابطه دوستی بین آن دو نفر p و اگر عضو یک خوشه نباشند q است. در واقع هر درایه این ماتریس یک متغییر تصادفی با توزیع برنولی است.
با توجه به این توضیحات می توان آن را به صورت زیر در نظر گرفت:

$$A_{i,j} \sim Bernoulli(Q_{z_i,z_j})$$

که در آن از ماتریس Q و بردار z که دستور کار تعریف شدند استفاده شده است.

پرسش تئوری ۱۰

اگر دو فرد i و j با هم دوست باشند آنگاه هر دو $\mathrm{A_{i,j}}$ و $\mathrm{A_{j,i}}$ باید برابر با یک باشند و اگر رابطه دوستی نداشته باشند هر دو $\mathrm{A_{i,j}}$ و $\mathrm{A_{j,i}}$ باید صفر باشند. بنابراین ماتریس A باید ماتریسی متقارن باشد. با توجه به گراف هایی که در دستور کار پروژه رسم شده اند، گراف حاصل حلقه ندارد. بنابراین درایه های روی قطر آن باید صفر باشند. در غیر اینصورت، حلقه به وجود می آید.

بنابراین ماتریس A ماتریسی متقارن با قطر اصلی صفر است.

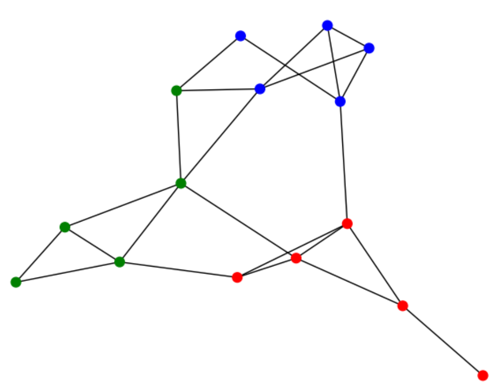
پرسش شبیه سازی ۲

با توجه به توضیحات درمورد ماتریس مجاورت در پرسش تئوری ۱۰، ده ماتریس مجاورت مختلف می سازیم. یک نمونه از آنها به صورت زیر است:

$$A = \begin{pmatrix} 0 & 0 & 0 & & 0 & 0 & 0 \\ 0 & 0 & 1 & \cdots & 1 & 1 & 0 \\ 0 & 1 & 0 & & 0 & 1 & 0 \\ & \vdots & & \ddots & & \vdots & \\ 0 & 1 & 0 & & 0 & 0 & 1 \\ 0 & 1 & 1 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 0 & & 1 & 0 & 0 \end{pmatrix}$$

پرسش شبیه سازی ۳

یک ماتریس مجاورت مشابه قسمت قبل به صورت تصادفی می سازیم و گراف آن را تشکیل می دهیم. این گراف به صورت زیر است:



شکل ۳_۱: یک نمونه گراف براساس یک ماتریس مجاورت تصادفی

در گراف افرادی که در یک خوشه قرار دارند یا به عبارت دیگر روی یک میز نشستند با رنگ یکسان مشخص شدند.

پرسش شبیه سازی ۴

طبق توضیحات تابعی مینویسیم که فاصله همینگ دو ورودی z_1 و z_2 را محاسبه کند. برای مثال اگر دو ورودی به صورت زیر داشته باشیم:

$$z_1 = [2\ 3\ 3\ 3\ 3\ 2\ 1\ 2\ 2\ 2\ 3\ 1\ 1\ 1\ 1]^T$$
$$z_2 = [1\ 1\ 3\ 2\ 3\ 3\ 1\ 3\ 2\ 1\ 2\ 3\ 2\ 1\ 2]^T$$

فاصله همینگ برابر است با:

$$d_H = 10$$

پرسش شبیه ساز ۵

در این قسمت پرسش فاصله همینگ را به ازای جایگشت های مختلف z محاسبه کرده و کمترین مقدار آن را به دست می آوریم. برای مثال اگر دو ورودی به صورت زیر داشته باشیم:

$$z_1 = [2\ 3\ 3\ 3\ 3\ 2\ 1\ 2\ 2\ 2\ 3\ 1\ 1\ 1\ 1]^T$$
$$z_2 = [1\ 1\ 3\ 2\ 3\ 3\ 1\ 3\ 2\ 1\ 2\ 3\ 2\ 1\ 2]^T$$

کمینه فاصله همینگ برابر است با:

$$d = 9$$

پرسش تئوری ۱۱

دو نوع درایه برای ماتریس توصیف شده در پرسش تئوری ۱۰ وجود دارد. روی قطر اصلی باشد و یا نباشد. اگر روی قطر اصلی بود از تمامی درایه های دیگر مستقل است واگر روی قطر اصلی نبود از تمامی درایه ها مستقل است به جز درایه قرینه خود نسبت به قطر اصلی که با آن یکسان است و با دانستن یکی، دیگری به طور کامل مشخص می شود.

پرسش تئوری ۱۲

با توجه به تعریف ارائه شده در صورت سوال داریم:

$$L(z) = P[A|z]$$

با توجه به پرسش تئوری ۱۱، می توان از مستقل بودن درایه ها از یکدیگر، تابع درست نمایی را می توان به صورت زیر نوشت:

$$\begin{aligned} L(z) = P[A|z] &= \prod_{i=1}^{n}\prod_{j=i}^{n} P\left[A_{i,j}\middle|z\right] = \prod_{i=1}^{n}\prod_{j=i+1}^{n} P\left[A_{i,j}\middle|z\right] \\ &= \prod_{i=1}^{n}\prod_{j=i}^{n}\left(A_{i,j}Q_{z_i,z_j} + \left(1 - A_{i,j}\right)\left(1 - Q_{z_i,z_j}\right)\right) \\ &= \prod_{i=1}^{n}\prod_{j=i}^{n}\left(1 + 2A_{i,j}Q_{z_i,z_j} - A_{i,j} - Q_{z_i,z_j}\right) \end{aligned}$$

پرسش تئوری ۱۳

با توجه به قسمت قبل لگاریتم تابع درست نمایی به صورت زیر خواهد بود:

$$l(z) = \log(L(z)) = \log\left(\prod_{i=1}^{n}\prod_{j=i}^{n}\left(1 + 2A_{i,j}Q_{z_i,z_j} - A_{i,j} - Q_{z_i,z_j}\right)\right)$$
$$= \sum_{i=1}^{n}\sum_{j=i}^{n}\log\left(1 + 2A_{i,j}Q_{z_i,z_j} - A_{i,j} - Q_{z_i,z_j}\right)$$

که آن را به صورت زیر نیز می توان نمایش داد:

$$l(z) = \sum_{i=1}^{n}\sum_{j=i}^{n}\log\left(A_{i,j}Q_{z_i,z_j} + \left(1 - A_{i,j}\right)\left(1 - Q_{z_i,z_j}\right)\right)$$

پرسش شبیه سازی ۶

طبق پرسش تئوری ۱۳ تابعی مینویسیم که $\tilde{l}(z) = -\log(L(z))$ را محاسبه کند. این مقدار برای ماتریسی که گراف آن در پرسش شبیه سازی ۳ رسم شد برابر است با:

$$\tilde{l}(z) = 41.275$$

پرسش شبیه سازی ۷

ماتریس A را با خوشه بندی اولیه زیر در نظر میگیریم:

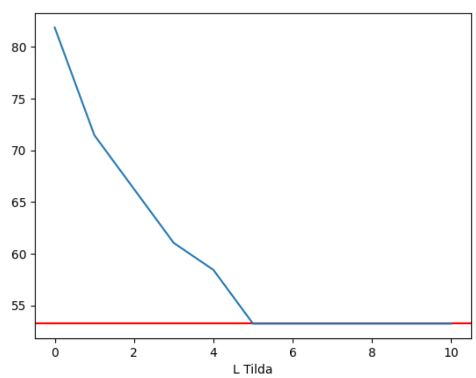
$$z_0 = [3\ 1\ 2\ 1\ 3\ 1\ 2\ 2\ 3\ 3\ 2\ 1\ 1\ 3]^T$$

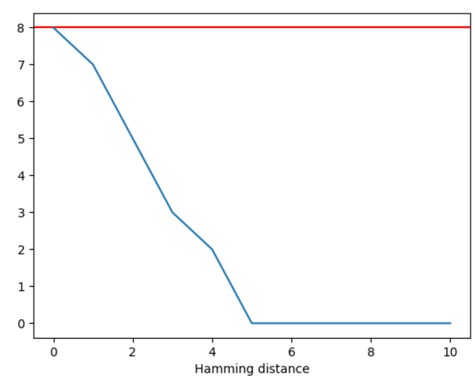
و به صورت تصادفی ماتریس A را میسازیم.

الگوریتم را طبق توضیحات پیاده سازی میکنیم. T را برابر با ۱۰ در نظر میگیریم و تخمین اولیه را به صورت زیر درنظر میگیریم:

$$\hat{z}_T = [3\ 3\ 3\ 3\ 3\ 2\ 2\ 2\ 2\ 2\ 1\ 1\ 1\ 1\ 1]^T$$

در هر مرحله $\tilde{l}(\hat{z}_t)$ و $d(\hat{z}_T, z_0)$ را روی نمودار نمایش میدهیم. نمودارها به صورت زیر هستند:





شکل ۳_۲: الف) کمینه فاصله همینگ مقدار واقعی خوشه بندی اولیه و تنیجه تخمین در هر مرحله و ب) قرینه لگاریتم درست یابی در هر مرحله از تخمین

خط قرمز رنگ در دو نمودار به ترتیب از راست به چپ نشان دهنده $\tilde{l}(z_0)$ و $d(\hat{z}_0, z_0)$ هستند.

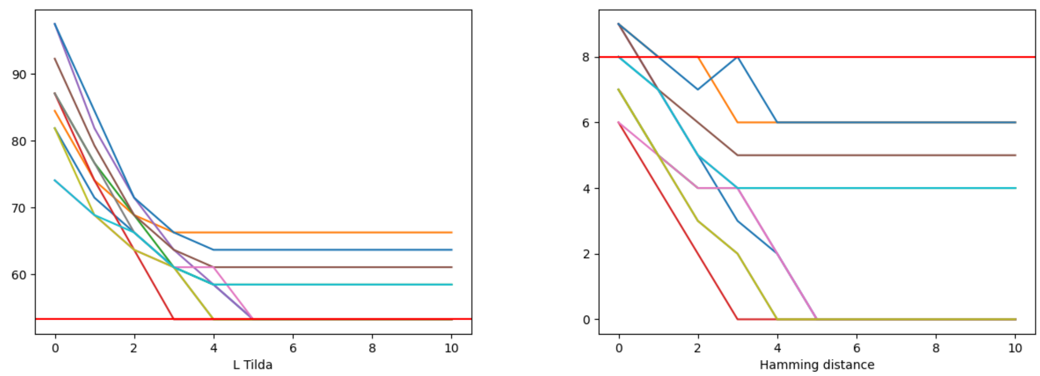
پرسش شبیه سازی ۸

ده تخمین اولیه متفاوت به صورت زیر در نظر میگیریم:

$$\hat{z}_0^{(1)} = [3\ 3\ 3\ 3\ 3\ 2\ 2\ 2\ 2\ 2\ 1\ 1\ 1\ 1\ 1]^T$$

$$\hat{z}_0^{(2)} = [3\ 2\ 1\ 3\ 2\ 1\ 3\ 2\ 3\ 2\ 1\ 2\ 1\ 3\ 1]^T$$
$$\hat{z}_0^{(3)} = [3\ 1\ 3\ 3\ 2\ 2\ 1\ 2\ 2\ 3\ 1\ 1\ 3\ 2\ 1]^T$$
$$\hat{z}_0^{(4)} = [1\ 2\ 3\ 1\ 3\ 2\ 3\ 2\ 2\ 3\ 1\ 3\ 1\ 2\ 1]^T$$
$$\hat{z}_0^{(5)} = [3\ 1\ 3\ 2\ 1\ 3\ 2\ 2\ 3\ 2\ 1\ 3\ 1\ 2\ 1]^T$$
$$\hat{z}_0^{(6)} = [2\ 3\ 2\ 3\ 1\ 3\ 1\ 3\ 2\ 3\ 2\ 2\ 1\ 1\ 1]^T$$
$$\hat{z}_0^{(7)} = [1\ 1\ 3\ 2\ 3\ 3\ 1\ 3\ 2\ 1\ 2\ 3\ 2\ 1\ 2]^T$$
$$\hat{z}_0^{(8)} = [2\ 3\ 3\ 3\ 3\ 2\ 1\ 2\ 2\ 2\ 3\ 1\ 1\ 1\ 1]^T$$
$$\hat{z}_0^{(9)} = [1\ 3\ 3\ 2\ 2\ 3\ 2\ 2\ 1\ 3\ 3\ 1\ 2\ 1\ 1]^T$$
$$\hat{z}_0^{(10)} = [1\ 3\ 2\ 3\ 2\ 1\ 3\ 1\ 2\ 1\ 3\ 2\ 3\ 2\ 1]^T$$

مراحل پرسش قبل را برای این ده تخمین اولیه تکرار میکنیم. نمودار ها به صورت زیر است:



شکل ۳_۳: الف) کمینه فاصله همینگ مقدار واقعی خوشه بندی اولیه و تنیجه تخمین در هر مرحله به ازاری ۱۰ تخمین اولیه متفاوت و ب) قرینه لگاریتم درست یابی در هر مرحله از تخمین به ازاری ۱۰ تخمین اولیه متفاوت

خط قرمز رنگ در دو نمودار به ترتیب از راست به چپ نشان دهنده $\tilde{l}(z_0)$ و $d(\hat{z}_0, z_0)$ هستند.

همانطور که از روی نمودار ها نیز مشخص است، در هر مرحله $\tilde{l}(z_t)$ کاهش یافته و $d(\hat{z}_t, z_0)$ اکثر اوقات کاهش می یابد و از مرحله ای به بعد به ازای بعضی از تخمین های اولیه، $d(\hat{z}_T, z_0)$ صفر میشود و $\tilde{l}(z_T)$ برابر با $\tilde{l}(z_0)$ میشود، یعنی خروجی الگوریتم تخمین برابر با خوشه بندی اولیه میشود و به ازای بعضی از تخمین های اولیه، $d(\hat{z}_T, z_0)$ و $\tilde{l}(z_T)$ به مقدار ثابتی می رسند، یعنی خروجی الگوریتم تخمین برابر با خوشه بندی اولیه نمیشود و هر قدر هم که T بزرگ باشد، باز هم خطا وجود خواهد داشت.

پرسش شبیه سازی ۹

به ازای برخی از تخمین های اولیه، لگاریتم تابع درست نمایی نحوه صحیح خوشه بندی ($\tilde{l}(z_0)$) و خروجی الگوریتم تخمین ($\tilde{l}(\hat{z}_T)$) برابر صفر میشود. روی نمودار های رسم شده در پرسش تئوری ۸ این موضوع مشخص است.

به ازای شش تخمین اولیه از بین ده تخمین اولیه پرسش قبل، این اتفاق می افتد یعنی شش j مختلف وجود دارد که $\tilde{l}\left(\hat{z}_T^{(j)}\right) = \tilde{l}(z_0)$ میشود. این شش تخمین اولیه عبارتند از:

$$\hat{z}_0^{(1)} = [3\ 3\ 3\ 3\ 3\ 2\ 2\ 2\ 2\ 2\ 1\ 1\ 1\ 1\ 1]^T$$
$$\hat{z}_0^{(2)} = [1\ 3\ 2\ 3\ 2\ 1\ 3\ 1\ 2\ 1\ 3\ 2\ 3\ 2\ 1]^T$$
$$\hat{z}_0^{(3)} = [3\ 1\ 3\ 3\ 2\ 2\ 1\ 2\ 2\ 3\ 1\ 1\ 3\ 2\ 1]^T$$
$$\hat{z}_0^{(4)} = [1\ 2\ 3\ 1\ 3\ 2\ 3\ 2\ 2\ 3\ 1\ 3\ 1\ 2\ 1]^T$$

$$\hat{z}_0^{(5)} = [3\ 1\ 3\ 2\ 1\ 3\ 2\ 2\ 3\ 2\ 1\ 3\ 1\ 2\ 1]^T$$
$$\hat{z}_0^{(6)} = [2\ 3\ 2\ 3\ 1\ 3\ 1\ 3\ 2\ 3\ 2\ 2\ 1\ 1\ 1]^T$$

به ازای تمام این مقادیر اولیه تخمین، فاصله همینگ نحوه صحیح خوشه بندی و خروجی الگوریتم صفر میشود. بنابراین میتوان نتیجه گرفت که هرگاه $\tilde{l}(z_0) = \tilde{l}\left(\hat{z}_T^{(j)}\right)$ شد، خروجی الگوریتم به درستی نحوه صحیح خوشه بندی را نشان میدهد.

پرسش شبیه سازی ۱۰

به ازای برخی از تخمین های اولیه کمینه فاصله همینگ نحوه صحیح خوشه بندی (z_0) و خروجی الگوریتم تخمین ($\hat{z}_T$) برابر صفر میشود. روی نمودار های رسم شده در پرسش تئوری ۸ این موضوع مشخص است.

یک نمونه تخمین اولیه که به ازای آن فاصله همینگ نحوه صحیح خوشه بندی و خروجی الگوریتم صفر است به صورت زیر است:

$$\hat{z}_0 = [1\ 3\ 2\ 3\ 2\ 1\ 3\ 1\ 2\ 1\ 3\ 2\ 3\ 2\ 1]^T$$

پرسش شبیه سازی ۱۱

سه ماتریس مجاورت تصادفی دیگر با خوشه بندی اولیه زیر در نظر میگیریم:

$$z_0 = [3\ 1\ 2\ 1\ 3\ 1\ 2\ 2\ 3\ 3\ 2\ 1\ 1\ 3]^T$$

بهترین تخمین،تخمینی است که کمترین که کمترین فاصله همینگ را داشته باشد. بهترین تخمین ها به ترتیب برای هر ماتریس به صورت زیر هستند:

$$\hat{z}_T = [2\ 3\ 1\ 3\ 2\ 3\ 1\ 1\ 2\ 2\ 1\ 1\ 3\ 3\ 2]^T \quad , \quad d(\hat{z}_T, z_0) = 2$$
$$\hat{z}_T = [3\ 1\ 2\ 1\ 3\ 1\ 2\ 2\ 2\ 3\ 3\ 2\ 1\ 1\ 3]^T \quad , \quad d(\hat{z}_T, z_0) = 0$$
$$\hat{z}_T = [2\ 3\ 2\ 3\ 1\ 3\ 2\ 2\ 2\ 1\ 1\ 1\ 3\ 3\ 1]^T \quad , \quad d(\hat{z}_T, z_0) = 2$$

بخش۴: او نه خیال است و نه طیف!

پرسش تئوری ۱۴

ماتریس Q و z را مانند بخش سه تشکیل میدهیم به این صورت که Q یک ماتریس با قطر اصلی p و دیگر درایه های آن q است و z یک بردار است که هر مولفه آن نشان دهنده دسته ای است که فرد با اندیس به آن تعلق دارد.

طبق توضیحات اگر $A_{i,j}$ یک باشد معنی آن این است که دو فرد i و j با هم هم سلیقه هستند و اگر این دو فرد در یک خوشه قرار داشته باشند، احتمال آن p و اگر در یک خوشه نباشند احتمال q است. بنابراین ماتریس A را میتوان به این صورت نشان داد:

$$A_{i,j} \sim Bernoulli(Q_{z_i,z_j})$$

این ماتریس باید یک ماتریس متقارن باشد، بنابراین درایه ها را به طور دقیق تر به صورت زیر است:

$$A_{i,j} \begin{cases} Bernoulli(Q_{z_i,z_j}) & i \geq j \\ A_{j,i} & j < i \end{cases}$$

پرسش تئوری ۱۵

طبق توضیحات بخش بخش قبل درایه $W_{i,j}$ یک متغییر برنولی است بنابراین ماتریس $A_{i,j}$ در حالت کلی به صورت زیر است:

$$W_{i,j} = Q_{z_i,z_j}$$

پرسش تئوری ۱۶

طبق بخش قبل ماتریس W در حالت ذکر شده به صورت زیر است:

$$W = \begin{pmatrix} p & p & q & q \\ p & p & q & q \\ q & q & p & p \\ q & q & p & p \end{pmatrix}$$

بردار X را بردار ویژه و λ را مقدار ویژه در نظر میگیریم. بنابراین:

$$WX = \lambda X$$

بردار X را به صورت زیر در نظر میگیریم:

$$X = (x_1\ x_2\ x_3\ x_4)^T$$

چهار معادله به صورت زیر خواهیم داشت:

$$p(x_1 + x_2) + q(x_3 + x_4) = \lambda x_1$$

$$p(x_1 + x_2) + q(x_3 + x_4) = \lambda x_2$$

$$q(x_1 + x_2) + p(x_3 + x_4) = \lambda x_3$$

$$q(x_1 + x_2) + p(x_3 + x_4) = \lambda x_4$$

در نتیجه:

$$\lambda(x_1 - x_2) = 0$$

$$\lambda(x_3 - x_4) = 0$$

دو حالت وجود دارد، یکی اینکه $\lambda = 0$ و دیگری اینکه $x_1 - x_2 = 0$ و $x_3 - x_4 = 0$.

حالت اول: $x_1 - x_2 = 0$ و $x_3 - x_4 = 0$:

$$p(2p-\lambda)x_1 + 2qx_3 = 0$$
$$p(2p-\lambda)x_3 + 2qx_1 = 0$$

در صورتی جواب دارد که:

$$\frac{2p-\lambda}{2q} = \frac{2q}{2p-\lambda} \longrightarrow \lambda = 2p \pm 2q$$

به ازای این دو مقدار ویژه، بردار ویژه ها به صورت هستند:

$$X_1 = \begin{pmatrix}1\\1\\1\\1\end{pmatrix}, \quad X_2 = \begin{pmatrix}1\\-1\\1\\-1\end{pmatrix}$$

حالت دوم: $\lambda = 0$:

$$p(x_1+x_2) + q(x_3+x_4) = 0$$
$$q(x_1+x_2) + p(x_3+x_4) = 0$$

در نتیجه:

$$x_1 + x_2 = 0$$
$$x_3 + x_4 = 0$$

بنابراین بردار ویژه ها به صورت زیر خواهند بود:

$$X_3 = \begin{pmatrix}1\\-1\\0\\0\end{pmatrix}, \quad X_4 = \begin{pmatrix}0\\0\\1\\-1\end{pmatrix}$$

بنابراین چهار بردار ویژه و مقدار ویژه به صورت زیر هستند:

$$\lambda_1 = 2p + 2q, X_1 = \begin{pmatrix}1\\1\\1\\1\end{pmatrix}, \qquad \lambda_2 = 2p - 2q, X_2 = \begin{pmatrix}1\\-1\\1\\-1\end{pmatrix}$$

$$\lambda_3 = 0, X_3 = \begin{pmatrix}1\\-1\\0\\0\end{pmatrix}, \qquad \lambda_4 = 0, X_4 = \begin{pmatrix}0\\0\\1\\-1\end{pmatrix}$$

پرسش تئوری ۱۷

در این بخش n_1 را مجموعه افراد دسته اول و n_2 را مجموعه افراد دسته دوم در نظر میگیریم و بردار X را بردار ویژه و λ را مقدار ویژه در نظر میگیریم. بنابراین:

$$WX = \lambda X$$

بردار X را به صورت زیر در نظر میگیریم:

$$X = (x_1\ x_2 \ldots x_{n-1}\ x_n)^T$$

برای افراد دسته اول و دوم معادلات به صورت زیر هستند:

$$p\sum_{i\epsilon n_1} x_i + q\sum_{i\epsilon n_2} x_i = \lambda x_k \qquad k\epsilon n_1$$
$$p\sum_{i\epsilon n_1} x_i + q\sum_{i\epsilon n_2} x_i = \lambda x_k \qquad k\epsilon n_2$$

از معادلات دسته اول و دوم به ترتیب داریم:

$$\lambda x_i = \lambda x_j \qquad i \text{ و } j\ \epsilon n_1$$
$$\lambda x_i = \lambda x_j \qquad i \text{ و } j\ \epsilon n_2$$

دو حالت وجود دارد، یکی اینکه $\lambda = 0$ باشد که جزو حالات مورد بررسی نیست و دیگری اینکه

$$x_i = x_j = a \qquad i \text{ و } j\ \epsilon n_1$$
$$x_i = x_j = b \qquad i \text{ و } j\ \epsilon n_2$$

باشد که در این صورت معادلات دسته اول و دوم به صورت زیر هستند:

$$\frac{n(pa+qb)}{2} = \lambda a \longrightarrow \left(p - \frac{2\lambda}{n}\right)a + qb = 0$$
$$\frac{n(pb+qa)}{2} = \lambda b \longrightarrow \left(p - \frac{2\lambda}{n}\right)b + qa = 0$$

برای اینکه این a و b جواب داشته باشند داریم:

$$\frac{p - \frac{2\lambda}{n}}{q} = \frac{q}{p - \frac{2\lambda}{n}} \longrightarrow \lambda = \frac{n}{2}(p \pm q)$$

بنابراین ماترس W تنها دو مقدار ویژه غیر صفر دارد و بردار ویژه آنها را نیز به شکل زیر است:

$$\lambda_1 = 2p + 2q\ , X_1 = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} \quad , \qquad \lambda_2 = 2p - 2q\ , [X_2]_i = \begin{cases} 1 & i\epsilon n_1 \\ -1 & i\epsilon n_2 \end{cases}$$

پرسش تئوری ۱۸

تعداد افراد دو دسته را برابر در نظر میگیریم. در این صورت ماتریس D_W به صورت زیر است:

$$D_W = \frac{n(p+q)}{2}\ I_n$$

که در آن I_n ماتریس همانی n تایی است.

پرسش تئوری ۱۹

در این بخش n_1 را مجموعه افراد دسته اول و n_2 را مجموعه افراد دسته دوم در نظر میگیریم و بردار X را بردار ویژه و λ را مقدار ویژه در نظر میگیریم. بنابراین:

$$L_W X = \lambda X$$

بردار X را به صورت زیر در نظر میگیریم:

$$X = (x_1\ x_2 \dots x_{n-1}\ x_n)^T$$

برای افراد دسته اول و دوم معادلات به صورت زیر هستند:

$$\frac{n(p+q)}{2}\ x_k - p\sum_{i\epsilon n_1} x_i - q\sum_{i\epsilon n_2} x_i = \lambda p x_k \qquad k\epsilon n_1$$
$$\frac{n(p+q)}{2}\ x_k - p\sum_{i\epsilon n_1} x_i - q\sum_{i\epsilon n_2} x_i = \lambda p x_k \qquad k\epsilon n_2$$

از معادلات دسته اول و دوم به ترتیب داریم:

$$\frac{n(p+q)}{2}\ (x_i - x_j) = \lambda p(x_i - x_j) \qquad i \text{ و } j\ \epsilon n_1$$
$$\frac{n(p+q)}{2}\ (x_i - x_j) = \lambda p(x_i - x_j) \qquad i \text{ و } j\ \epsilon n_2$$

دو حالت وجود دارد، یکی اینکه $\lambda = \frac{n(p+q)}{2}$ باشد و دیگری اینکه

$$x_i = x_j = a \qquad i \text{ و } j\ \epsilon n_1$$
$$x_i = x_j = b \qquad i \text{ و } j\ \epsilon n_2$$

باشد که در این صورت معادلات دسته اول و دوم به صورت زیر هستند:

$$\frac{n(p+q)}{2}\ a - \frac{n(pa+qb)}{2} = \lambda a \longrightarrow \left(\frac{nq}{2} - \lambda\right) a = \frac{nq}{2} b$$
$$\frac{n(p+q)}{2}\ b - \frac{n(pb+qa)}{2} = \lambda b \longrightarrow \left(\frac{nq}{2} - \lambda\right) b = \frac{nq}{2} a$$

برای اینکه این a و b جواب داشته باشند داریم:

$$\frac{\frac{nq}{2} - \lambda}{\frac{nq}{2}} = \frac{\frac{nq}{2}}{\frac{nq}{2} - \lambda} \longrightarrow \lambda = \begin{cases} 0 \\ \\ nq \end{cases}$$

بردار ویژه برای این دو مقدار به صورت زیر است:

$$\lambda_1 = 0\ , X_1 = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} \quad , \qquad \lambda_2 = nq\ , [X_2]_i = \begin{cases} 1 & i\epsilon n_1 \\ -1 & i\epsilon n_2 \end{cases}$$

اگر $\lambda = \frac{n(p+q)}{2}$ باشد آنگاه داریم:

$$p\sum_{i\epsilon n_1} x_i + q\sum_{i\epsilon n_2} x_i = 0$$
$$q\sum_{i\epsilon n_1} x_i + p\sum_{i\epsilon n_2} x_i = 0$$

برای اینکه بردار ویژه غیر صفر باشد داریم:

$$\sum_{i\epsilon n_1} x_i = 0$$
$$\sum_{i\epsilon n_2} x_i = 0$$

بردار $T_{n,k}$ را برداری در نظر میگیریم که طول آن n و مولفه اول آن یک و مولفه k آن منفی یک باشد. بردار $Vzeros_n$ را برداری در نظر میگیریم که طول آن n و همه مولفه ها آن صفر است. با این تعریف بردار ویژه ها به صورت زیر در می آیند:

$$X_{2+i} = \begin{pmatrix} T_{\frac{n}{2},i} \\ Vzeros_{\frac{n}{2}} \end{pmatrix} \qquad \forall i : n\epsilon N , 0 < i < \frac{n}{2}$$

$$X_{n+1-i} = \begin{pmatrix} Vzeros_{\frac{n}{2}} \\ T_{\frac{n}{2},i} \end{pmatrix} \qquad \forall i : n\epsilon N , 0 < i < \frac{n}{2}$$

بنابراین به ازای $\lambda = \frac{n(p+q)}{2}$ تعداد $n-2$ بردار ویژه خواهیم داشت.

پرسش تئوری ۲۰

با توجه به پرسش قبلی، برای حالت چهارتایی پرسش ۱۶ مقادیر و بردارهای ویژه به صورت زیر هستند:

$$\lambda_1 = 0 , X_1 = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \quad , \qquad \lambda_2 = 4q, X_2 = \begin{pmatrix} 1 \\ 1 \\ -1 \\ -1 \end{pmatrix}$$

$$\lambda_3 = 2(p+q) , X_3 = \begin{pmatrix} 1 \\ -1 \\ 0 \\ 0 \end{pmatrix} \quad , \qquad \lambda_4 = 2(p+q) , X_4 = \begin{pmatrix} 0 \\ 0 \\ 1 \\ -1 \end{pmatrix}$$

دو مقدار ویژه کوچکتر $\lambda_1 = 0$ و $\lambda_2 = 4q$ هستند. یک ماتریس چهار در دو با بردار ویژه های متناظر با این دو مقدار ویژه میسازیم. این ماتریس به صورت زیر است:

$$R = (u_1, u_2) = \begin{pmatrix} 1 & 1 \\ 1 & 1 \\ 1 & -1 \\ 1 & -1 \end{pmatrix}$$

طبق این ماتریس باید نقاط را به صورت زیر برای این چهار فرد در نظر بگیریم:

$$r_1 = r_2 = (1,1)$$

$$r_3 = r_4 = (1,-1)$$

فرد یک و دو در یک دسته و سه و چهار هم در دسته دیگر قرار داشتند. بنابراین با این روش می توان افراد را به دو گروه تقسیم کرد.

پرسش شبیه سازی ۱۲

ابتدا به صورت تصافی دسته بندی اولیه را انجام میدهیم. سپس با استفاده ار این دسته بندی و پرسش های ۱۴ و ۱۵ ماتریس ها L_A و L_W را تشکیل میدهیم. سپس طبق تعریف A و W را محاسبه کرده و مقادیر ویژه و بردارهای ویژه آنها را به دست می آوریم.

در تقسیم بندی براساس L_W مشابه قسمت قبل، آنهایی که مولفه بردار ویژه دوم مربوط به آنها مثبت است را در یک دسته و آنهایی که به مولفه متناظر با آنها منفی است را در دسته دیگر میگذاریم. از آنجایی که ماتریس L_A بردار ویژه هایی نزدیک به بردار ویژه های L_W دارد و در تقسیم بندی با استفاده از بردار ویژه های L_W بردار ویژه اول تاثیری نداشت و دسته بندی براساس مثبت و یا منفی بودن مولفه های بردار ویژه دوم انجام میشد، در تقسیم بندی با بردار ویژه های L_A نیز بردار ویژه اول تاثیری نخواهد داشت و براساس مثبت و یا منفی بودن مولفه های بردار ویژه دوم تقسیم بندی انجام میشود.

به این ترتیب تقسیم بندی را انجام میدهیم. برای بررسی خطا، دسته بندی تخمین زده شده را با دسته بندی اولیه مقایسه میکنیم به این شکل که تعداد افرادی که در تخمین در گروه اشتباه قرار گرفتند را شمارش کرده و به عنوان خطا گزارش میکنیم.

به ازای $n = 10000$ و $p = 0.1$ و $q = 0.01$ خطا هم تخمین به وسیله A و هم در تخمین به وسیله W برابر صفر شد.

پرسش تئوری ۲۱

با توجه به نتایج پرسش تئوری ۱۹، بردار متناظر با دومین مقدار کوچک مقدار ویژه ماتریس L_W یک و منفی یک هستند و دسته بندی بر همین اساس انجام میشود. در دسته بندی این بردار برای L_A، براساس مثبت یا منفی بودن مولفه ها تقسیم بندی کردیم مانند بخش قبل.

بردار متناظر با دومین مقدار کوچک مقدار ویژه ماتریس L_A نسخه آغشته به نویز این بردار برای L_W است. بنابراین هر مولفه آن را می توان به صورت مولفه بردار برای L_W به علاوه مقداری خطا به صورت تصادفی در نظر گرفت. با توجه به نحوه دسته بندی، برای اینکه خطا رخ دهد باید اندازه مقدار خطا برای هر مولفه، از یک بیشتر باشد.

$u_2(L)$ بردار ویژه ی متناظر با دومین مقدار ویژه ی ماتریس L و $[u_2(L)]_j$ را مولفه jام آن در نظر میگیریم. طبق توضیحات داریم :

$$[u_2(L_A)]_j = [u_2(L_W)]_j + \varepsilon_j$$

n_1 را مجموعه مولفه های بدون خطا و n_2 را مجموعه مولفه ها خطا دار در نظر میگیریم. داریم:

$$\sum_{j=1}^{n} |[u_2(L_W)]_j - [u_2(L_A)]_j| = \sum_{j\in n_1} |\varepsilon_j| + \sum_{j\in n_2} |\varepsilon_j|$$

برای مولفه های بدون خطا داریم:

$$0 < |\varepsilon_j| < 1$$

برای مولفه های خطا دار داریم:

$$|\varepsilon_j| \geq 1$$

کران پایین دو نوع خطا را در نظر میگیریم. داریم:

$$\sum_{j=1}^{n} |[u_2(L_W)]_j - [u_2(L_A)]_j| = \sum_{j\in n_1} |\varepsilon_j| + \sum_{j\in n_2} |\varepsilon_j| \geq |n_2|$$

که در آن $|n_2|$ تعداد داده های خطا دار است.

تا اینجا اینطور در نظر گرفتیم که به طور کلی علامت دو بردار ویژه $u_2(L_W)$ و $u_2(L_A)$ یکسان است اما امکان دارد که اینطور نباشد و علامت این دو بردار به طور کلی تفاوت داشته باشد. می توان با ضرب کردن منفی در $u_2(L_A)$ این حالت را به حالت هم علامت بودن در حالت کلی تبدیل کرد چون منفی $u_2(L_A)$ نیز بردار ویژه L_A است. بنابراین حالت کلی تر معادله بالا به صورت زیر است:

$$\exists\theta \in \{-۱,۱\}: \quad \sum_{j=1}^{n} |[u_2(L_W)]_j - \theta[u_2(L_A)]_j| = \sum_{j\in n_1} |\varepsilon_j| + \sum_{j\in n_2} |\varepsilon_j| \geq |n_2|$$

طبق معادله نامساوی صورت سوال داریم:

$$\exists\theta \in \{-۱,۱\}: \quad \sum_{j=1}^{n} \left|[u_2(L_W)]_j - \theta[u_2(L_A)]_j\right| = \sum_{j\in n_1} \left|\varepsilon_j\right| + \sum_{j\in n_2} \left|\varepsilon_j\right| \leq \frac{C}{\mu^2}$$

از ترکیب این دو معادله داریم:

$$|n_2| \leq \frac{C}{\mu^2}$$

اگر تعداد داده های خطا دار را با e نشان دهیم خواهیم داشت:

$$e \leq \frac{C}{\mu^2}$$

بنابراین الگوریتم خوشه بندی طیفی حداکثر تعداد ثابتی خطا خواهد داشت و این تعداد مستقل از اندازه ماتریس ها است. از آنجایی که نامساوی صورت سوال با احتمال $1 - 4e^{-n}$ برقرار است، میتوان گفت که با احتمال $1{-}\in(n)$ الگوریتم خوشه بندی طیفی حداکثر تعداد ثابتی خطا خواهد داشت که در آن

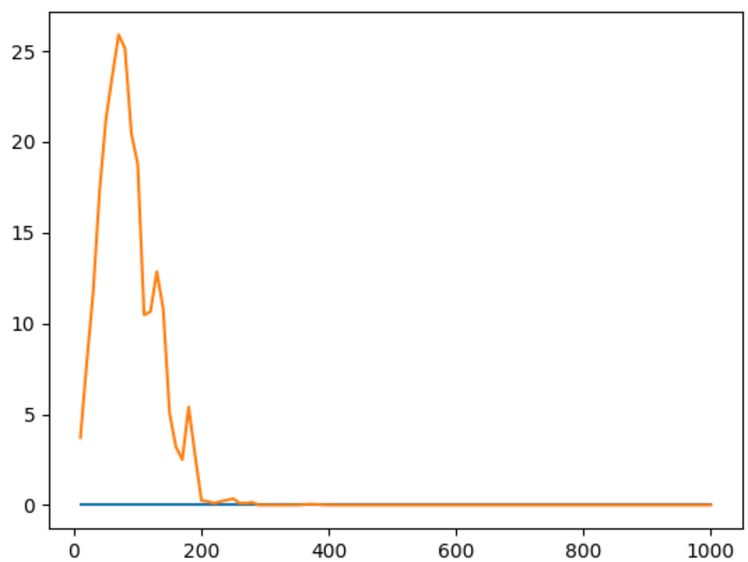
$$\in(n) = 4e^{-n}$$

است. بنابراین تعداد داده های مورد نیاز برای رسیدن به خطا $1-\epsilon$ برابر است با:

$$n_{min} = \ln\left(\frac{4}{\epsilon}\right)$$

پرسش شبیه سازی ۱۳

کاری که در پرسش شبیه سازی ۱۲ برای $n = 10000$ انجام دادیم را برای چند n بین یک تا هزار انجام میدهیم و نمودار آن را رسم میکنیم. نمودار آن به صورت زیر است که در آن خط نارنجی خطای ناشی از تخمین با L_A را نشان می دهد و خط آبی خطا ناشی از L_W را نشان می دهد که همانطور که انتظار داشتیم همواره صفر است. خطای ناشی از تخمین با L_A نیز با افزایش n کاهش می یابد و تقریبا برای nهای بزرگ تر از ۳۰۰ صفر می شود و این یعنی بردار ویژه های L_A و L_W تقریبا به هم نزدیک هستند.



شکل ۴_۱: مقدار خطا الگوریتم خوشه بندی طیفی به ازای nهای مختلف

پرسش تئوری ۲۲

از الگوریتم خوشه بندی طیفی برای دسته بندی داده ها استفاده می شود. زمانی که بخواهیم به دو دسته تقسیم بندی کنیم، مشابه پرسش تئوری ۲۰ و پرسش شبیه سازی ۱۲ عمل میکنیم اما برای بیشتر از دو دسته باید ابتدا ماتریس شباهت یا همسانی را تشکیل دهیم. برای این کار معیارهای مختلفی را می توان در نظر گرفت. مثلا برای داده های مورد بررسی تا به اینجا، احتمال هم سلیقه بودن معیار شباهت بود. می توان معیار های دیگر را نیز در نظر گرفت. مثلا اگر داده مورد بررسی متوسط درآمد باشد، میتوان ماتریس مشابهت را با رابطه

$$S_{i,j} = (Inc_i - Inc_j)^2$$

و یا رابطه

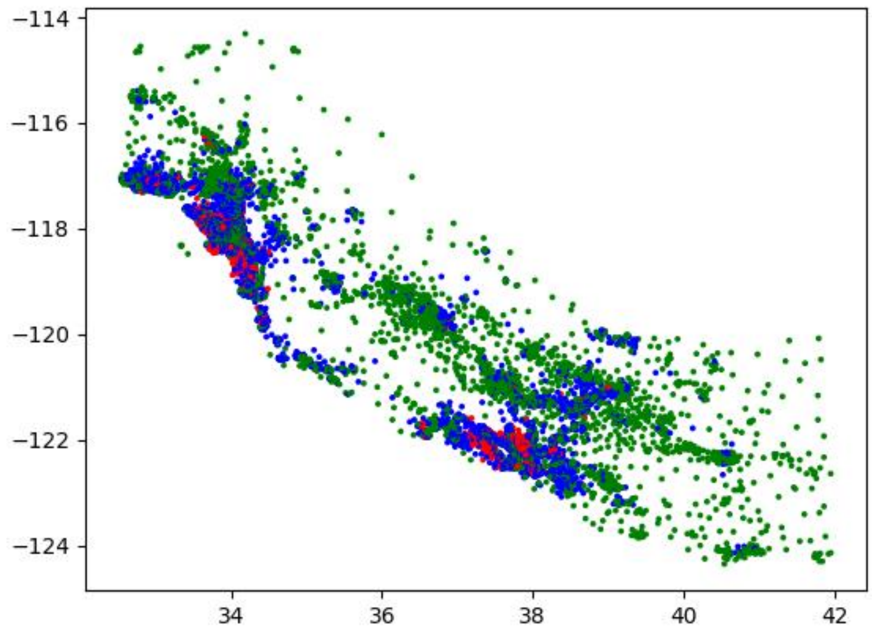
$$S_{i,j} = 1 - e^{-(Inc_i - Inc_j)^2}$$

در نظر گرفت. سپس ماتریس L_S را تشکیل میدهیم و بردار ویژه ها و مقدار ویژه های آن را محاسبه می کنیم. این ماتریس n بردار ویژه خواهد داشت که ما k کوچکترین مقدار ویژه ها را در نظر میگیریم. سپس بردار ویژه های متناظر با این k مقدار ویژه را در نظر گرفته و یک

ماتریس n در k تشکیل میدهیم که k تعداد خوشه های مورد نظر است. ماتریس جدید را U در نظر میگیریم و y_i را سطر iام آن در نظر میگیریم. حالا n نقطه در فضای k بعدی داریم که مختصات هر یک با بردار y_i نشان میدهیم. این نقاط را با استفاده از الگوریتم k_mean دسته بندی میکنیم و خوشه بندی به این صورت انجام میشود.

پرسش شبیه سازی ۱۴

مجموعه داده های sing Calfornia را به سه دسته با الگوریتم k_mean خوشه بندی میکنیم. خوشه بندی به صورت زیر خواهد بود:

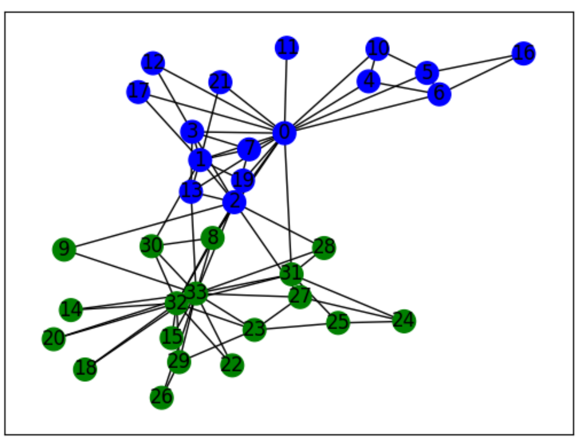


شکل ۴_۲: دسته بندی داده های sing Calfornia به سه خوشه با الگوریتم k_mean

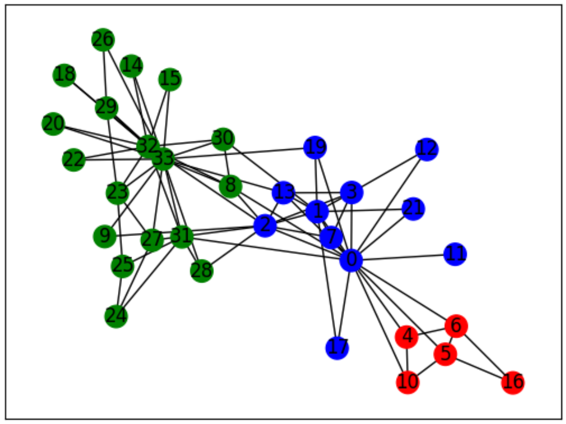
نقاط سبز درآمد متوسط کم، نقاط آبی درآمد متوسط متوسط و نقاط قرمز درآمد متوسط زیاد را نشان می دهند.

پرسش شبیه سازی ۱۵

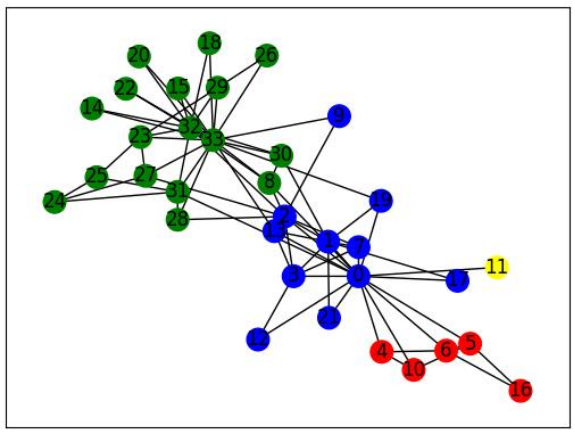
مجموعه داده های $Zachary's\ Karate\ Club$ را به سه دسته با الگوریتم k_mean خوشه بندی میکنیم. خوشه بندی برای تعداد خوشه های دو و سه و چهار به صورت زیر خواهد بود:



شکل ۴_۳: دسته بندی داده های $Zachary's\ Karate\ Club$ به دو خوشه با الگوریتم k_mean و خوشه بندی طیفی



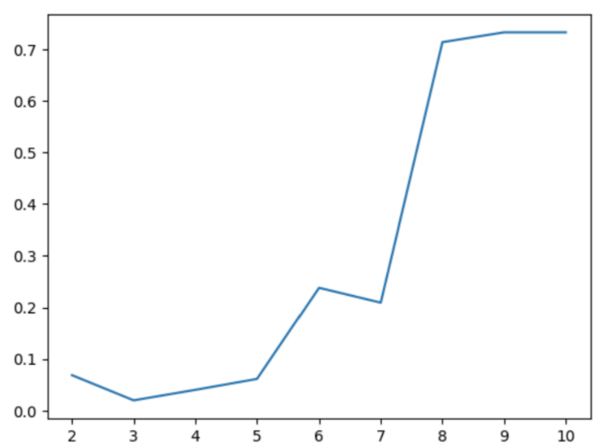
شکل ۴_۴: دسته بندی داده های $Zachary's\ Karate\ Club$ به سه خوشه با الگوریتم k_mean و خوشه بندی طیفی



شکل ۴_۵: دسته بندی داده های $Zachary's\ Karate\ Club$ به چهار خوشه با الگوریتم k_mean و خوشه بندی طیفی

پرسش شبیه سازی ۱۶

مقدار متوسط تابع $f(S)$ که در صورت سوال تعریف شده است را به ازای دو تا ده خوشه رسم میکنیم. نمودار خروجی به صورت زیر است:



شکل ۴_۶: مقدار متوسط تابع $f(S)$ به ازای دو تا ده خوشه

با توجه به نمودار فوق، تعداد خوشه مناسب برای مجموعه داده سه خوشه است.

$Probability\ and\ Statistics\ Project$

$Phase 1\ Section 5$

Theoretical 23:

Assuming that ther are $m \leq \frac{n(n-1)}{2}$ friendship among all people in this cluster and the probability of two people being friends is p, The probabality of getting all the friendship true is $\frac{p^m(1-p)^{\frac{n(n-1)}{2}-m}}{\sum_{k=0}^{n} C(\frac{n(n-1)}{2},k)p^k(1-p)^{\frac{n(n-1)}{2}-k}}$

Theoretical 24:

Assuming that this time we are aware of the value of m, the probability will become $\frac{p^m(1-p)^{\frac{n(n-1)}{2}-m}}{C(\frac{n(n-1)}{2},m)p^m(1-p)^{\frac{n(n-1)}{2}-m}} = \frac{1}{C(\frac{n(n-1)}{2},m)}$

Theoretical 25:

Assuming that we want to foretell 20% of all friendships true and the rest of all m frienships falsely is:

Simulating 17:

First we will define a funtion that will create and store the Adjacency matrix $A \in \mathbb{R}^{n \times n}$ where each element fo the matrix is a $a_{i,j} \approx Bernoulli(p)$

In [ ]:
```
from scipy.stats import bernoulli, binom
import numpy as np

def generate_sample_bernoulli(n,p):
    mlist = a = [[0] * n] * n
    c2n = int(n*(n-1)/2)
    p = 0.0034
    c = 0
    X = bernoulli(p)
    X_samples = X.rvs(c2n)
    for i in range(0, n):
        for j in range(0, i):
            mlist[i][j] = X_samples[c]
            c += 1
            mlist[j][i] = mlist[i][j]
    return mlist
```

```
def generate_Adjacency_Bernoulli(n,p):
    mlist = np.zeros([n,n])
    for i in range(0,n):
        mlist[i,i] = 0
        for j in range(0,n):
            mlist[j,i] = np.random.binomial(1,p)
            mlist[j,i] = mlist[j,i]
    return mlist
```

In [ ]:
```
n1 = 1000
m1 = 3000
p1 = 0.0034
list1 = []
h= 0
for N in range(0,10):
    Adj1 = generate_Adjacency_Bernoulli(n1,p1)
    list1.append(sum(sum(Adj1))/2)
    #for row in range(0, n1):
        #if(sum(row)>L)
        #h += Adj1[x][:].count(1)

list1
```

Out[ ]:
```
[1640.0,
 1705.0,
 1682.0,
 1711.0,
 1695.0,
 1709.5,
 1727.5,
 1699.0,
 1693.0,
 1696.0]
```

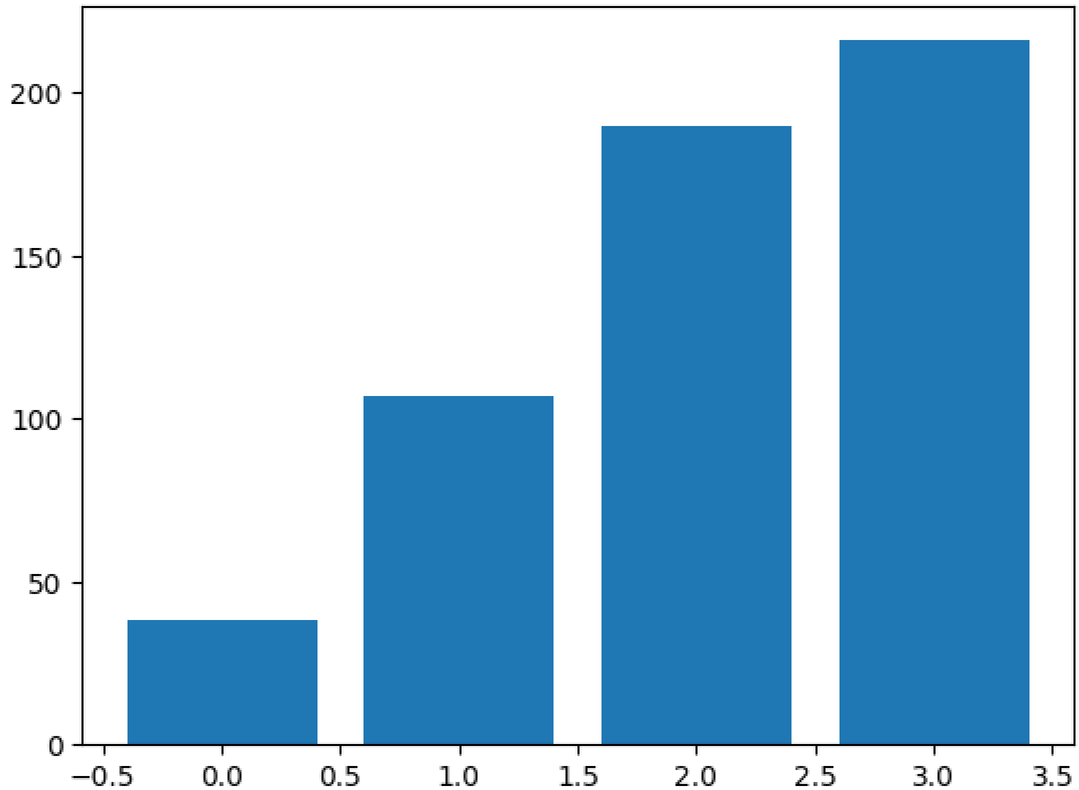
In [ ]:
```
m = 3000
mean= sum(list1)/len(list1)
print("mean of all friendships = "+str(mean))
error = (mean-m)/mean
print("error with respect to m = "+str(error))
```

```
mean of all friendships = 1695.8
error with respect to m = -0.7690765420450525
```

In [ ]:
```
import matplotlib.pyplot as plt

ss = 4
list_ = [0,0,0,0]
data = {}
data[0]= 0
data[1]=0
data[2]=0
data[3]=0
for item in sum(Adj1):
    if item<ss:
        data[item] +=1
```

```
        #list_[item] +=1

#plt.hist(list_)
plt.bar(data.keys(),data.values())
plt.show()
```

Out[ ]: <BarContainer object of 4 artists>



Theoretical 26:

Let X_{ij} be the random variable for person i and person j be friends. Since all X_{ij}'s are all i.i.d. $X_{i,j} \approx Bernoulli(p)$. Let the expected value of all friendships be named $M = \mathbb{E}|edges|$. We know that if we count all degrees of people in graph that would be: $2|edges| = \sum_{k=0}^{n} deg(v_k) = \sum i = 0^n X_{ii} + 2\sum_{1\leq i<j\leq n} X_{i,j}$. Getting Expecrd value from both RHS and LHS implies that:

$$2\mathbb{E}|edges| = 2M = \sum_{k=0}^{n} deg(v_k) = \sum_{i=0}^{n} \mathbb{E}X_{ii} + 2\sum_{1\leq i<j\leq n} \mathbb{E}X_{i,j} \rightarrow M = \frac{1}{2}np + C(n,$$

. But if we consider X_{ii} to be zero then we would have : $M = \frac{n(n-1)p}{2}$

Simulating 18:

In this simulating firstly, we wish to find L which is the mean of all friendship of a person. Then with the help of that we will continue to find the mean of This so called "Hamrang" group's number.

In [ ]:
```
def Hamrang_finder_by_adjmat(adj_mat):
    counter = 0
    n = len(adj_mat[0])
    #L = (adj_mat).count(1)/n
    L = sum(sum(adj_mat))/n
    for row in adj_mat:
        #if adj_mat[x][:].count(1)> L:
        if (sum(row)>L):
            counter+=1
    return counter

list2 = []
n2 =1000
p2 = 0.00016
for N in range(0,10):
    adj_mat2 = generate_Adjacency_Bernoulli(n2,p2)
    list2.append(Hamrang_finder_by_adjmat(adj_mat2))
list2
```

Out[ ]:
```
[151, 124, 135, 149, 131, 147, 146, 140, 141, 165]
```

In [ ]:
```
xx = sum(list2)/len(list2)
print("the mean of Hamrang group among all People is = " + str(xx))
```

```
the mean of Hamrang group among all People is = 142.9
```

Theoretical 27:

Let Y_i be the number of neighbors of person i,since $Y_i = \sum_{j=1}^{n} X_{i,j} \approx Binomial(n,p)$ and $\mathbb{E}X_{i,j} = p$ which will result in $\mathbb{E}Y_i = \sum_{j=1}^{n} \mathbb{E}X_{i,j} = np = 0.016$.

Theoretical 28:

if each person is having L friends on average, we choose someone randomly, then he will be "Hamrang" with Probavility: $\sum_{i=\lfloor L+1 \rfloor}^{n} C(n-1,i)p^i(1-p)^{n-i-1}$ Since in this case $L < 1$ then $\lfloor L \rfloor = 0$ which suggests:
$z = \sum_{i=1}^{n} C(n-1,i)p^i(1-p)^{n-i-1} = (p+1-p)^{n-1} - C(n-1,0)(1-p)^{n-1} = 1-(1-$
. Finally letting $n = 1000$ and $p = 0.000016$ will lead to: $z \approx 0.15$. So we can deduce that for each person being "Hamrang" is a Bernoulli random variable with parameter $z = 0.15$ whcih his named h_i after the i-th person. => $h_i \approx Bernoulli(p = 0.15)$, which finally suggest the expected value of "Hmarang" people among all will be : $H = \sum_{i=1}^{n} h_i$ getting expexted value from both sides implies that : $\mathbb{E}H = n\mathbb{E}h_i = nz \approx 147.73$

Simulating 19:

In [ ]:
```
import itertools
# from itertools import combination
```

```
def meshgrid(n):
    i =0
    j =0
    res = [i for j in range(0,n)]
    return res

def find_subsets(set_, n):
    res = list(itertools.combinations(set_,n))
    return res

def find_num_of_triagles(adj_mat):
    res = 0
    n = len(mat)
    indices = meshgrid(n)
    all_3sized_subsets = find_subsets(indices,3)


    for x,y,z in all_3sized_subsets:
        if ((mat[x,y] + mat[y,z] + mat[z,x] ==2)):
                counter += 1

    return res



def find_num_of_taragozari(adj_mat):
    res = 0
    n = len(mat)
    indexes = [i for i in range(n)]
    all_3sized_subsets = find_subsets(indexes,3)
    for x,y,z in all_3sized_subsets:
        if (mat[x,y] + mat[y,z] + mat[z,x]) == 3 :
            res += 1

    return res
```

In [ ]:
```
# in order to save our computer we need to take a smaller n with respect to n=3000
n3 = 300
p3 = 0.01
adj_mat3 = generate_Adjacency_Bernoulli(n3,p3)
taragozari_num = find_num_of_taragozari(adj_mat3)
triagle_num = find_num_of_triagles(adj_mat3)
print("number of all trargozari : " + str(taragozari_num))
print("number of all triangles(chains) : "+str(triagle_num))
```

```
number of all trargozari : 10176
number of all triangles(chains) : 0
```

In [ ]:
```
n3 = 100
p3 = 0.01
N = 5
taragozari = []
zangiri = []
for temp in range(0,N):
    adj_mat3 = generate_Adjacency_Bernoulli(n3,p3)
    zangiri.append(find_num_of_triagles(adj_mat3))
```

```
    taragozari.append(find_num_of_taragozari(adj_mat3))
taragozari
```

Out[ ]: [0, 0, 0, 0, 0]

In [ ]:
```
zangiri
```

Out[ ]: [0, 0, 0, 0, 0]

In [ ]:
```
tara = sum(taragozari)/len(taragozari)
mosalas = sum(zangiri)/len(zangiri)
print("the mean of taragozari is : "+ str(tara))
print("the mean of chains is : "+ str(mosalas))
```

```
the mean of taragozari is : 0.0
the mean of mosalas is : 32340.0
```

Simulating 20:

First we will compute number friendship having a confied set of all vertices by this function down below. Then we will find each person's friends(neighbors) and find the friendships inside that set of friends using the function we defined.

In [ ]:
```
def find_num_of_edges(adj_mat, vertics):
    s = 0
    for i in vertics:
        for j in vertics:
            if adj_mat[i][j] ==1 :
                s += 1
    return int(s/2)

n4 = 1000
p4 = 0.003
adj_mat4 = generate_Adjacency_Bernoulli(n4,p4)
total_relations = 0
for i in range(n4):
    temp_neighbors = []
    for j in range(n4):
        if adj_mat4[i][j] == 1:
            temp_neighbors.append(j)
    total_relations += find_num_of_edges(adj_mat4,temp_neighbors)
yyy = total_relations/n4
print("Average friendships among the set of each person's friends = "+ str(yyy))
```

```
Average friendships among the set of each person's friends = 2.0
```

Simulating 21:

In this sectio n we will use networkx library in order to find the shortest path between nodes in the graph. The we will find the average distance between two arbitrary vertices.

In [ ]:
```
import networkx as nx
```

```
n5 = 1000   # ////////////////////////////////////////////////////////// n5 = 1000
p5 = 0.0033
adj_mat5 = generate_Adjacency_Bernoulli(n5,p5)

my_graph = nx.path_graph(n5)
i = 0
total_path = 0
j =0
# making the graph
while(i<n5):
    j = i
    if adj_mat5[i,j] ==1 :
        my_graph.add_edge(i,j)
        j +=1
    i+=1

# finding distances
for i in range(0,n5):
    for j in range(0,n5):
        total_path += nx.shortest_path_length(my_graph, i,j)

c2n = n5*(n5 -1 )/2
result = total_path*2/c2n
print(result)
```

```
134.66666666666666
```

Simulating 22:

Here we will repeat the same thing but for 100 times ... i n order to find the longest path each time and finally calculate its mean(expected value)

In [ ]:
```
N = 100
n6 = 50
p6 = 0.34
longest_paths = []
for i in range(n6):

    adj_mat6 = generate_Adjacency_Bernoulli(n6,p6)
    my_gr = nx.path_graph(n6)
    i = 0
    while(i<n6):
        j = i
        while(j<n6):
            if adj_mat6[i][j] ==1 :
                my_gr.add_edge(i,j)
            j +=1
        i+=1
temp_max =0
for i in range(0,n6):
    for j in range(0,n6):
        if nx.shortest_path_length(my_gr,i,j)> temp_max:
            temp_max =  nx.shortest_path_length(my_gr,i,j)
```

```
    longest_paths.append(temp_max)
zzzz = sum(longest_paths)/len(longest_paths)
print("The mean of longest path between two arbitrary nodes is = "+ str(zzzz))
```

```
The mean of longest path between two arbitrary nodes is = 2.0
```

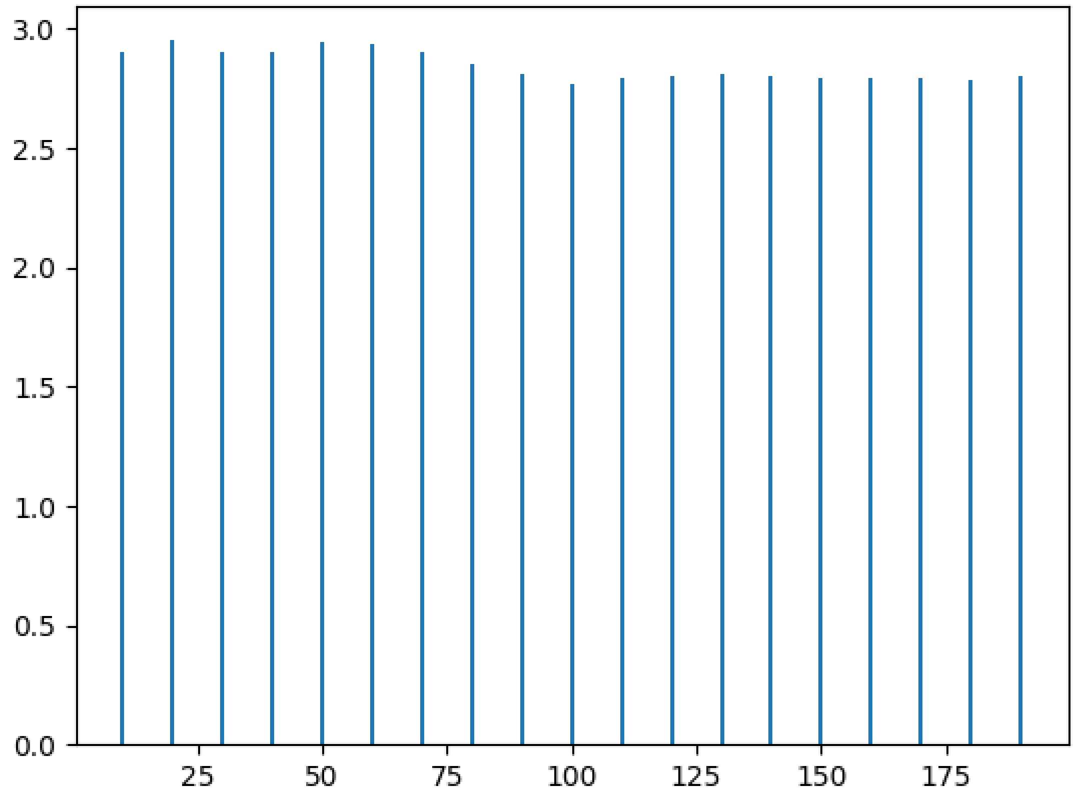
Simulating 23:

In [ ]:

```
chains_num = []
data_input = [i*10 + 10 for i in range(19)]
steps = (200 - 10)//10
N = 100   # ///////////////////////////////////////////////   N =10
p7 = p6
LongestPaths = []

for z in range(steps):
    for _ in range(N):
        n7 = z*10 + 10
        adj_mat = generate_Adjacency_Bernoulli(n7,p7)
        graph = nx.path_graph(n7)

        i = 0
        while (i < n):
            j = i
            while (j < n):
                if (adj_mat[i,j] == 1):
                    graph.add_edge(i, j)
                j = j+1
            i = i+1

        temp_max=0
        for i in range(n):
            for j in range(n):
                if( nx.shortest_path_length(graph, i, j)>temp_max):
                    temp_max = nx.shortest_path_length(graph, i, j)
        LongestPaths.append(temp_max)

    chains_num.append(sum(LongestPaths)/len(LongestPaths))

plt.figure()
plt.bar(data_input,chains_num)
```

Out[ ]:

```
<BarContainer object of 19 artists>
```



as n grows larger, the expected vlaue of chains will convege to 2.5

Simulation 24:

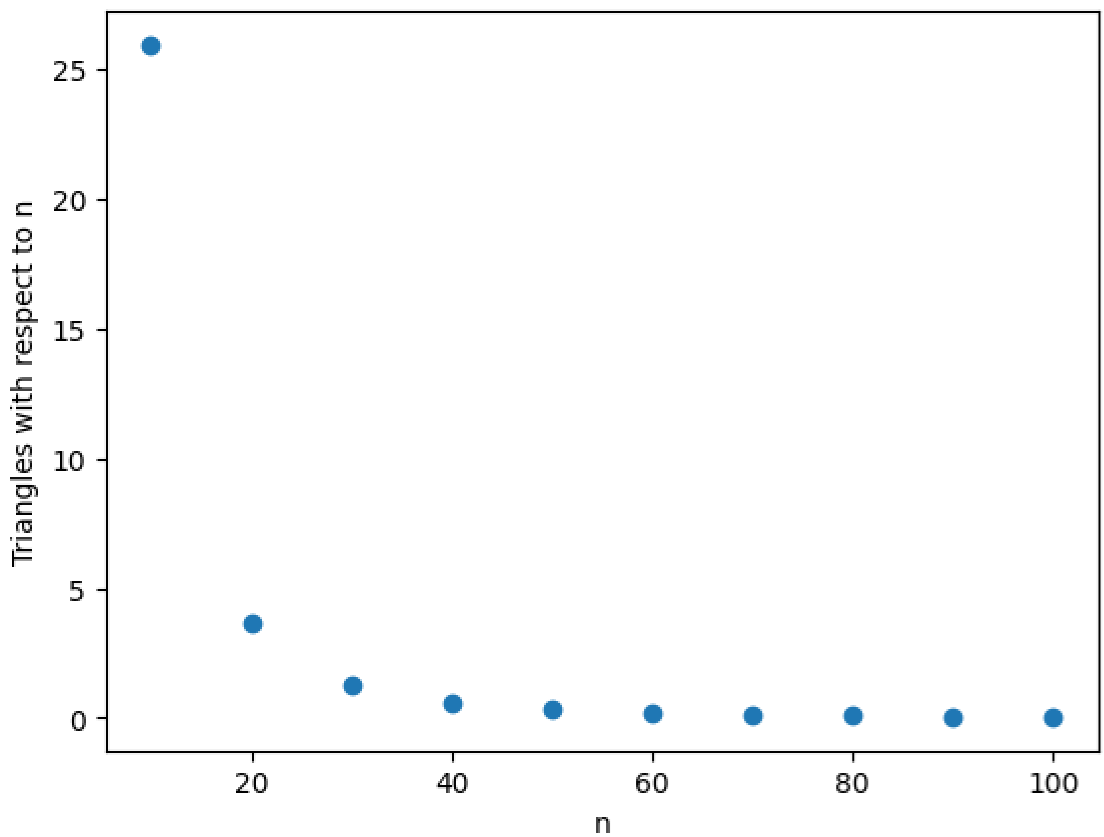
In [ ]:
```
def Triangle(n,p):
    res = 0
    for N in range(100):
        adj_mat = generate_Adjacency_Bernoulli(n,p)
        res += cal_number_of_taragozari(adj_mat)
    return res/100


n8= 100
p8 = 0.34
y = 0
x = Triangle(n8,p8)
print("number of triangles = "+str(z))
```

```
number of triangles = 101.76000000000016
```

Simulation 25:

In [ ]:
```
def Triangle(n,p):
    res = 0
    for N in range(100):
        adj_mat = generate_Adjacency_Bernoulli(n,p)
        res += cal_number_of_taragozari(adj_mat)
    return res/100
```

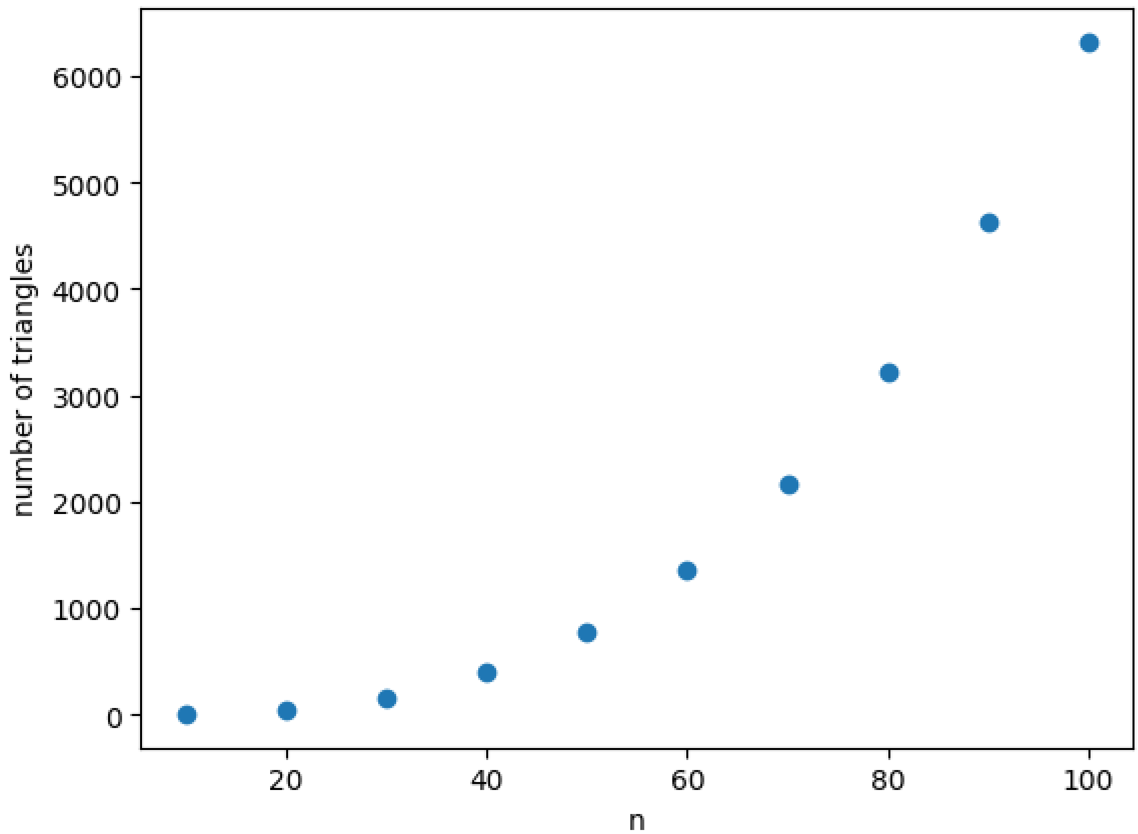
```
def get_number_of_taragozari(adj_mat):
    n = len(adj_mat)
    res = 0
    indices = [i for i in range(n)]
    all_3subset = find_subsets(indices,3)
    for x,y,z in all_3subset:
        if mat[x,y]==1 and mat[y,z]==1 and  mat[z,x]==1 :
            res += 1
    return res


y = []
x = [i*10 + 10 for i in range(10)]
steps = 10

for t in range(steps):
    n = t*10 + 10
    p = 60/(n*n)
    y.append(Triangle(n,p))

plt.scatter(x,y)
plt.xlabel("n")
plt.ylabel("Triangles with respect to n");
plt.show()
```



As you can see since $p = \frac{60}{n^2}$ is decreasing with n the probability if two peple being frrinnds is decreasing which results in less and less Expectation if triangles.

Simulation 26:

In [ ]:
```
import matplotlib.pyplot as plt

y2 = []
x2 = [i*10 + 10 for i in range(10)]
steps = 10
p = 0.34
for t in range(steps):
    n = t*10 + 10
    y2.append(Triangle(n,p))

plt.scatter(x2,y2)
plt.xlabel("n",)
plt.ylabel("number of triangles");
```



it ca be deduced that if p is constant, increasing of n will result in rising of Expectation of number of tringles.

Simulation 27:

In [ ]:
```
y3 = []
x3 = [i*50 + 50 for I in range(20)]
steps = 20
for t in range(steps):
    n = 50*t
y3 = []
```

```
x3 = [i*50 + 50 for I in range(20)]
steps = 20
for t in range(steps):
    n = 50*t +50
    p = 1/n
    y.append(Triangle(n,p))
plt.scatter(x,y)
plt.xlabel("n")
plt.ylabel("number of triangles with respect to n")
Xcumulate = np.cumsum(x, axis =0)
Ycumulate = np.cumsum(y, axis =0)
Nn = len( Ycumulate)
for i in range(Nn):
    Ycumulate[i] = Ycumulate[i]/Xcumulate[i]

plt.scatter(Xcumulate,Ycumulate)
plt.xlabel("n")
plt.ylabel("cumulative mean with respect to n")
```

As we an see cumulative mean goes to 0 as n grows larger and larger.

Theoretical 29:

In taragozari(T_0) two out of three nodes must be connected, wheras in triangle(T_1) all nodes are connected to each other. $T_0 = \mathbb{E}|taragozari| = C(n,3)C(3,2)p^2(1-p)$ while $T_1 = \mathbb{E}|triangles| = \frac{1}{3}C(n,3)p^3$

Theoretical 30:

if x, y, z are im a cluster and also each is connected to at least one node. The probabaility of having "taragozai" is : $\frac{C(n,3)C(3,2)p^2(1-p}{C(n,3)C(3,2)p^2(1-p)+C(3,n)p^3}$. However the probability of having a "triangle" is : $\frac{C(n,3)p^3}{C(n,3)C(3,2)p^2(1-p)+C(n,3)p^3}$

Theoretical 32:

let u, v be two nodes. if we want them not having mutual friendship with person x_k the probability will be $(1-p)^2$. if we want no x_k have relatins with both u, v;since these aer independant the probabaility will become $\mathbb{P}(I_{u,v} = 1) = (1-p)^{2n-4}$

Theoretical 33:

Let $d(i)$ be the number of connected nodees to the i -th person. We know that if $m = |edges|$ then $2m = \sum_{k=0}^{n} d(k)$ since each edge was counted twice in the summation. Let us define $\bar{d}\,(i) = C(n,2) - d(i)$ which is clearly the number of disconnected nodes to the i-th person. We know that: $X_n = \sum_{i=0}^{n} C(\bar{d}\,(i), 2) = \frac{1}{2}\sum_{i=0}^{n}(\bar{d}\,(i)^2 - \bar{d}\,)$ which taking Expectation implies that: $\mathbb{E}X_n = \frac{1}{2}\sum_{i=0}^{n}(\mathbb{E}\bar{d}\,(i)^2 - \mathbb{E}\bar{d}\,)$. but we knew

$\mathbb{E}m = \frac{n^2p}{2} = \frac{1}{2}\mathbb{E}d(i) \to \mathbb{E}d(i) = n^2p \to \mathbb{E}\bar{d}\,(i) = n - n^2p$.Now comuting $\mathbb{E}\bar{d}\,(i)^2 = \sum_{k=0}^{n} k^2p^k(1-p)^{n-k-1}C(n-1,k) = n(n-1)p((n-2)p+1)$ which leads to $\mathbb{E}X_n = n(n(n-1)p(n-2)p+1)) - (n-n^2p)$

Theoretical 34:

using Markov inequality we can deduce that:
$\mathbb{P}(X_n \geq 1) \geq \frac{\mathbb{E}X_n}{1} = n(n(n-1)p(n-2)p+1)) - (n-n^2p)$

Probability and Statistics Project

*Phase*1 *Section*6

Theoretical 36:

Here is the probavility of going from node i to node j at one step $P_{i,j} = \frac{A_{i,j}}{d(i)}$. If $A_{i,j} = 0$, so does $P_{i,j}$; and if $A_{i,j} = 1$ since there are $d(i)$ possible routes, $P_{i,j} = \frac{1}{d(i)}$

Theoretical 37:

We can rewrite $P_{i,j}$ as $P_{i,j} = \frac{A_{i,j}}{d(i)}$ and D is a diagonal matrix, we can deduce that:
$D \times P = A \to P = D^{-1}A \to DP^{(t)}D^{-1} = (P^{(t)})^T$

Theoretical 38:

first we consider the case $t = 3$ (The base of the induction). Similar to prior aproach we obtain that: $[P^3]_{i,j} = \sum_{k=0}^{n}\sum_{s=0}^{n} P_{ik}P_{ks}P_{sj} = \sum_{k=0}^{n} P_{ik}\sum_{s=0}^{n} P_{ks}P_{sj} = \sum_{k=0}^{n} P_{ik}[P^2]_{k,j}$. Now we suppose that $[P^{t-1}]_{h,j}$ is the probability of going form h to j in $t-1$ steps. Thus we get $[P^t]_{i,j} = \sum_{k=0}^{n} P_{ih}[P^{t-1}]_{h,j} = [P^t]_{i,j}$. Thus the induction's step is prove since $f(3)$ is true and $f(n-1)$ will lead to $f(n)$. T

Theoretical 39:

we can deduce that $P_{ij}^{t} = \sum_{k=1}^{n} P_{ik}^{t-1}P_{kj}$ from the proof we did in Theoretical 38.

Theoretical 40:

Suppose that $V = \{v_1, v_2, \ldots, v_{h-1}\}$ are between nodes i, j. Moreover assume that $v_0 = v_{h+1} = i, v_h = j$. We can deduce that in going from i to j =:The probability will be : $[P^h]_{i,j} = \prod_{k=0}^{h}[P]_{v_j,v_{k+1}} = \prod_{k=0}^{h} \frac{A_{v_k,v_{k+1}}}{d(v_k)}$. The same approach for going from j to i will result in: $[P^h]_{j,i} = \prod_{k=0}^{h} [P]_{v_{h-k}, v_{h-k-1}} = \prod_{k=0}^{h} \frac{A_{v_{h-k}, v_{h-k-1}}}{d(v_k)}$. which since $A_{i,j} = A_{j,i}$ will finally leads to : $\frac{P_{ij}^{(t)}}{P_{ji}^{(t)}} = \frac{1/d(i)}{1/d(j)} = \frac{d(j)}{d(i)}$

Theoretical 41:

Since i, j are friends in the cluster, we can deduce that $[P^t]_{i,j} \approx [P^t]_{j,k}$. because the intersts are similar and there is good chance if the random path is similar too, then $[P^t]_{i,j} \approx [P^t]_{j,k}$.

Theoretical 42:

From adding Principle we can deduce that: $P_{C_1,k} = \frac{1}{\|C_1\|} \sum_{v_i \in C_1} P_{ik}^{(t)}$. letting this into the geiven formula implies that:

$$d(C_1, C_2) = \sqrt{\sum_{k=0}^{n} r_{ij}^2} = \sqrt{\sum_{k=1}^{n} \frac{(P_{C_1,k} - P_{C_2,k})^2}{d(k)}}$$

Theoretical 43:

We can use an aproach similar to "K-means" methos. Assume that the vector $X = \{x_1, x_2, x_3, .., x_n\}$ are the data we wish to cluster and $S = \{S_1, .., S_k\}$ is the set of clusers. S_k's must be chosen such that within-cluster sum of squares is minimized. Therefore we wish to minimize this variable $\sum_{i=1}^{k} \sum_{x \in S_i} ||x - \mu_i||^2$ i among all data. Thus, we get: $\arg\min_s \sum_{i=1}^{k} \sum_{x \in S_i} ||x - \mu_i||^2 = \arg\min_s \sum_{i=1}^{k} |S_i| Var(S_i)$ where μ_i is the mean of i-th cluster S_i. So we will continue on clustering the data if we take two arbitrary S_i, S_j, then the combination of these two will have larger " within-cluster sum of squares".