**プロジェクト最終レポート**

**宣伝効果のシミュレーション**

担当教授: 佐々木晃

提出日: 2017/08/08

所属: 情報科学部ディジタルメディア学科

学年、クラス: ３年C組

氏名: 水村宏輔

**●課題**

人間の生活の中で、どのように物事(音楽、映画、ドラマ、商品など)が流行していくかについてのシミュレーションを行った。情報の伝達方法には様々な要因が考えられるが、今回は、広告と知り合い同士の口コミによる効果に限定した。プログラミング言語はPythonを利用し、TkinterのCanvasを使って可視化を行った。

**●目的**

　人間世界の中で起こる流行のシミュレーションをすることによって、より効率的に宣伝効果をあげるにはどうしたらいいのかを調査する。

●方法

　人間世界での情報伝達には、人間関係が大きく関わってくる。まず、人間関係をシミュレーションすることから始めた。今回は、全員が顔見知りであることとして、250人程度の人間関係を作成した。個人個人は以下の特性を持つ。

・個人番号(ひとりひとり異なり、個人を特定するために利用する。)

・所属グループ(5つのグループのうちいずれかに所属する。可視化の際にグループごとに色分けされる。)

・情報を知っているか否か(可視化の際に知っている人はアウトラインが黒くなる。)

・情報に関する興味(１０段階で設定する。)

・よく行く都市(各々に好きな都市が設定され、その都市によく行く。)

これらの特性を利用して、人間の間の心的距離と物理的距離について考える。

　心的距離は、人と人の間で現在どのくらい心が近づいているか、または離れているかについて表す。簡単化のために、２次元のフィールド上に各々の心的位置を配置した。

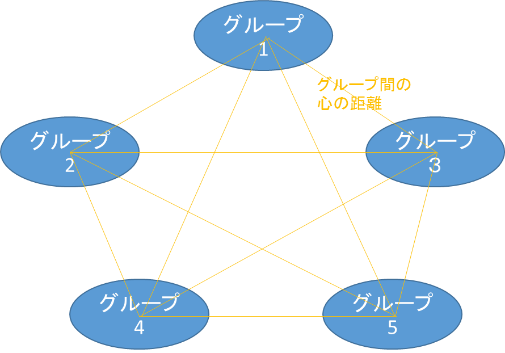


　　図1. グループ間の心の距離

　図1のようにフィールド上に5つのグループを配置する。このフィールド上の距離が、グループ間の心の距離を表す。たとえば、グループ1とグループ3は距離が近い。よって、グループ間交流が比較的多いといえる。それに対して、グループ1とグループ4は距離が遠い。よって、グループ同士はほとんど関わりがないといえる。このようにして、5つのグループ同士の心的な距離を表現した。人間は各々いずれかのグループに所属していて、それぞれグループの中に自分の居場所が存在する。人間はこのフィールド上を動き回るが、一定の確率で自分の場所（所属グループ方向）に戻ろうとする。よって、基本、自分のグループの近くを保って動き回っている。グループごとに所属メンバーが戻りたがる確率が異なる。戻りたがる確率が高いグループは、メンバーがほとんど自分のグループ内を離れないということになる。逆に戻りたがる確率が低いグループは、メンバーは自分の場所を離れ、フィールド上のいろんな場所を動き回る。よって、様々なグループと心的距離が近づくことができる。これによってグループごとのコミュニケーション能力の違いを表現した。

　物理的距離は、実際の人間社会(街中、教室など)での人間間の距離を表す。こちらも2次元のフィールド上の各々の物理的位置を配置した。今回は、都市が二つあるサイズの街を想定した。

　物理的空間上では人々は心的距離の影響を受けずに基本的にランダムに動き回るようにした。しかし、ある一定の確率で、各々の好きな都市の方向に向かうようにした。これによって、都市である地域には人が集まり、都市から離れた地域は、基本的に人が少ない状態が保たれる。このフィールド上に場所とサイズを指定した広告を配置する。

　以上で作成した、心的距離と物理的距離を利用して、広告効果と口コミ効果による流行のシミュレーションを行った。

　広告効果は、物理的空間に配置した広告の場所やサイズによってどのように情報が広まっていくかということである。人々は広告を何回も見ることによって情報を認知していく。その際、各々の情報に関する興味を利用して、興味が大きい人は、少ない数広告の前を通るだけで情報を認知する。また、興味があまりない人は、何度も何度も同じ広告の前を通ることによって、情報を認知していくことになる。

　口コミ効果は、人間同士で情報をやり取りすることによって、伝わっていくことである。今回は、物理的空間における実際の会話の中における口コミに限定した。つまり、ネットの口コミなどは想定していない。人間同士の物理的距離が0になったとき(街中で出会ったとき)、心的距離が近ければ会話をする確率が高くなる。逆に、遠い場合はほとんどが無視する。次に、会話が成立したとき、お互いの情報に関する興味が大きいほど、その情報に関する話が出てくる確率が高くなる。逆に、いずれかが興味のない話はあまりしないはずである。さらに、その情報に関する話をしたときに、一定の確率で情報が伝わらない場合がある。会話中にその情報が頭にないこともあるし、聞き手がちゃんと聞いてないこともあるためである。これらの3つの確率をクリアした場合に口コミが成立し、情報が伝達する。

●実行結果

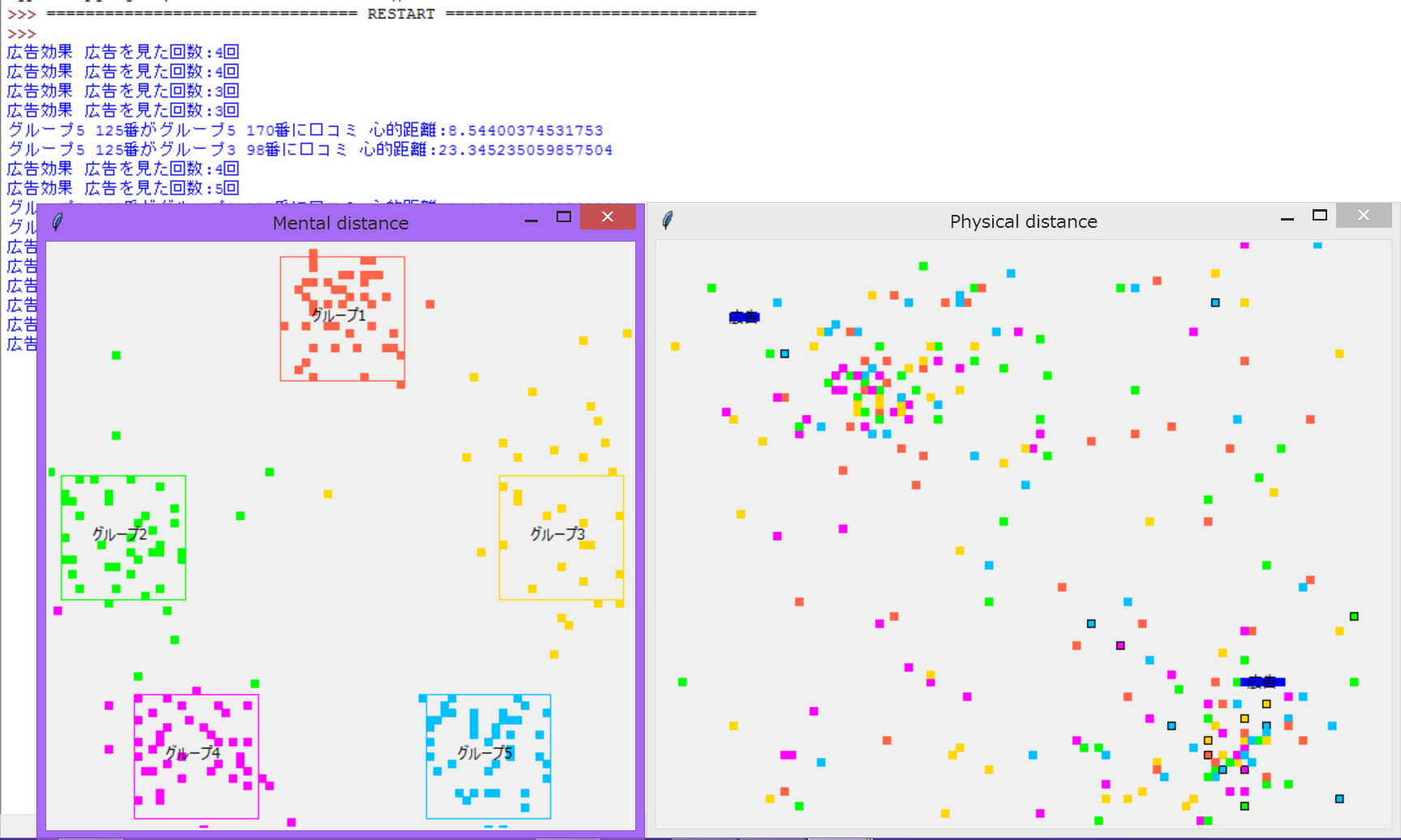


図２. シミュレーションのスクリーンショット

　心的距離と物理的距離をCanvas上に表示し、広告効果と口コミが成立するたびに情報を表示するようにした。

　広告は都市から遠い場所に置くとなかなか効果は表れなかった。逆に都市の中心部に置くとすぐに効果が表れた。

　口コミは同じグループ同士で最もよく起こった。

●考察

　今回、宣伝効果のシミュレーションを考える上で、人間関係を自分の考える範囲でシミュレーションしてみた。しかし、実際の人間関係はもっと複雑なものである。さらに現実に近づけるために、人間関係に関してさらに詳しく調べる必要がある。

**●付録（ソースプログラム）**

**promotion.py**

from tkinter import \*

import time, random, math

pic=6

FIELD\_X, FIELD\_Y = 80, 80

FIELD\_Xp, FIELD\_Yp = 100, 80

tk = Tk()

tk.title("Mental distance")

canvas = Canvas(tk, width=FIELD\_X\*pic, height=FIELD\_Y\*pic)

canvas.pack()

tkp = Tk()

tkp.title("Physical distance")

canvasp = Canvas(tkp, width=FIELD\_Xp\*pic, height=FIELD\_Yp\*pic)

canvasp.pack()

class World:

def \_\_init\_\_(self,population):

#population:グループごとの人数

self.people = []

self.bases = []

self.advertise = []

self.population = population

self.communication = []

def add\_people(self, group, personal\_number):

self.people.append(Human(group,personal\_number))

def add\_base(self, size, group):

self.bases.append(Base(size,group))

def add\_advertise(self, x, y, wide):

self.advertise.append(Advertise(x,y,wide))

def step(self):

people\_sample = []

people\_sample\_p = []

for human in self.people:

human.move()

human.movep()

people\_sample.append(human)

people\_sample\_p.append(human)

for A in self.people: #心的接触判定

for B in people\_sample:

if (A.x == B.x) and (A.y == B.y):

if A.group != B.group:

self.communication.append((A,B))

people\_sample.pop(0)

for A in self.people:

for advertise in self.advertise:

advertise.effect(A)

for B in people\_sample\_p:

if (A.xp == B.xp) and (A.yp == B.yp):

if A.number != B.number:

if (A.known != 1000) or (B.known != 1000):

A.reviews(B)

people\_sample\_p.pop(0)

for human in self.people:

human.render()

for base in self.bases:

base.render()

for advertise in self.advertise:

advertise.render()

tk.update()

tkp.update()

tk.update\_idletasks()

tkp.update\_idletasks()

canvas.delete("all")

canvasp.delete("all")

time.sleep(0.05)

def start(self, n\_steps):

current = 0

for i in range(self.population):

for j in range(5):

current = current+1

self.add\_people(j+1,current)

for i in range(5):

self.add\_base(16,i+1)

self.add\_advertise(10,10,4)

self.add\_advertise(80,60,6)

for x in range(n\_steps):

self.step()

class Human:

def \_\_init\_\_(self, group, personal\_number):

#group:所属グループ名 personal\_number:個人番号(１人１つ)

self.group = group

self.number =personal\_number

self.known = 0 #情報を知っているかどうか(この値が1000のとき知っている)

self.base\_size = 16

self.s = self.base\_size/2

self.colors = ["tomato","lime","gold","magenta","deepskyblue"]

self.cities = [("Shinjuku",5,30,20),("Ikebukuro",6,80,70)] #都市のリスト(都市名, 都市レベル(0~10), x, y)

city = random.choice(self.cities)

self.like\_city = city

favo = random.randint(1,10)

self.favorite\_level = favo

if self.group == 1: #グループ1

self.center\_x,self.center\_y = 40,10 #グループの本拠地の中心

self.return\_p = 40 #戻り率:小さいほどコミュニケーション能力が高いグループ

elif self.group == 2: #グループ2

self.center\_x,self.center\_y = 10,40

self.return\_p = 20

elif self.group == 3: #グループ3

self.center\_x,self.center\_y = 70,40

self.return\_p = 10

elif self.group == 4: #グループ4

self.center\_x,self.center\_y = 20,70

self.return\_p = 30

elif self.group == 5: #グループ5

self.center\_x,self.center\_y = 60,70

self.return\_p = 50

for i in range(5):

if self.group == i+1:

rx = random.randint(self.center\_x-self.s,self.center\_x+self.s)

ry = random.randint(self.center\_y-self.s,self.center\_y+self.s)

self.base\_x, self.base\_y = rx,ry #個人の定位置

self.color = self.colors[i] #個体の色

self.x, self.y = self.base\_x, self.base\_y #個人の心理的位置

self.xp, self.yp = self.base\_x, self.base\_y #個人の物理的位置

self.vx, self.vy = 1, 1

self.vxp, self.vyp = 1, 1

def move(self):

self.change\_dir()

if self.x == FIELD\_X:

self.vx = -self.vx

self.x = self.x + self.vx

if self.y == FIELD\_Y:

self.vy = -self.vy

self.y = self.y + self.vy

def change\_dir(self):

dirs = [(1,1),(-1,1),(-1,-1),(1,-1),

(1,0),(0,1),(-1,0),(0,-1),(0,0)]

ind = dirs.index((self.vx, self.vy))

r = random.random()

if r > 0.95:

newInd = (ind + 1) % len(dirs)

self.vx, self.vy = dirs[newInd]

elif r <= 0.01\*self.return\_p:

#本拠地方向に進む

if self.x < self.base\_x:

self.vx = 1

elif self.x == self.base\_x:

self.vx = 0

elif self.x > self.base\_x:

self.vx = -1

if self.y < self.base\_y:

self.vy = 1

elif self.y == self.base\_y:

self.vy = 0

elif self.y > self.base\_y:

self.vy = -1

else:

newInd = ind

self.vx, self.vy = dirs[newInd]

def movep(self):

self.change\_dirp()

self.xp = (self.xp + self.vxp)%FIELD\_Xp

self.yp = (self.yp + self.vyp)%FIELD\_Yp

def change\_dirp(self):

dirs = [(1,1),(-1,1),(-1,-1),(1,-1),

(1,0),(0,1),(-1,0),(0,-1),(0,0)]

ind = dirs.index((self.vxp, self.vyp))

r = random.random()

if r < 0.1:

newInd = (ind + 1) % len(dirs)

self.vxp, self.vyp = dirs[newInd]

elif r >= 0.1 and r< 0.2:

if r < self.like\_city[1]\*0.1:

if self.xp < self.like\_city[2]:

self.vxp = 1

elif self.xp == self.like\_city[2]:

self.vxp = 0

elif self.xp > self.like\_city[2]:

self.vxp = -1

if self.yp < self.like\_city[3]:

self.vyp = 1

elif self.yp == self.like\_city[3]:

self.vyp = 0

elif self.yp > self.like\_city[3]:

self.vyp = -1

else:

newInd = ind

self.vxp, self.vyp = dirs[newInd]

def reviews(self,human): #口コミ

dist = math.sqrt(pow(self.x-human.x,2)+pow(self.y-human.y,2))

Field\_size = math.sqrt(pow(FIELD\_X,2)+pow(FIELD\_Y,2))

p = dist/Field\_size + 0.5

rr = random.random()

if rr >= p:

if (rr < self.favorite\_level\*0.1) and (rr < human.favorite\_level\*0.1):

if self.known == 1000:

human.known = 1000

print("グループ{} {}番がグループ{} {}番に口コミ 心的距離:{}".format(self.group,self.number,human.group,human.number,dist))

elif human.known == 1000:

self.known = 1000

print("グループ{} {}番がグループ{} {}番に口コミ 心的距離:{}".format(human.group,human.number,self.group,self.number,dist))

def render(self):

canvas.create\_rectangle(self.x\*pic, self.y\*pic,

self.x\*pic+pic, self.y\*pic+pic,

fill=self.color, outline=self.color)

if self.known == 1000:

out = "Black"

else:

out = self.color

canvasp.create\_rectangle(self.xp\*pic, self.yp\*pic,

self.xp\*pic+pic, self.yp\*pic+pic,

fill=self.color, outline=out)

class Base:

def \_\_init\_\_(self, size, group):

#size:本拠地のサイズ group:グループ名

self.size = size

self.group = group

self.human = Human(self.group,None) #グループの情報を知りたいだけなので

#個人番号はなんでもよし

def render(self):

canvas.create\_rectangle((self.human.center\_x-(self.size/2))\*pic,

(self.human.center\_y-(self.size/2))\*pic,

(self.human.center\_x+(self.size/2)+1)\*pic,

(self.human.center\_y+(self.size/2)+1)\*pic,

outline=self.human.color)

canvas.create\_text(self.human.center\_x\*pic, self.human.center\_y\*pic,

text="グループ{}".format(self.group))

class Advertise: #広告

def \_\_init\_\_(self, x, y, wide):

self.x, self.y = x, y

self.wide = wide

def effect(self, human): #広告による効果

if human.known != 1000:

if (human.xp >= self.x) and (human.xp <= self.x+self.wide) and (human.yp >= self.y) and (human.yp <= self.y+1):

human.known = human.known + 1

if human.known == 13 - human.favorite\_level:

print("広告効果 広告を見た回数:{}回".format(human.known))

human.known = 1000

def render(self):

canvasp.create\_rectangle(self.x\*pic, self.y\*pic,

(self.x+self.wide)\*pic, (self.y+1)\*pic,

fill="blue", outline="blue")

canvasp.create\_text((self.x+self.wide/2)\*pic, (self.y+1/2)\*pic,

text="広告")

world = World(50)

world.start(3000)