スタートアップゼミ
課題発表

交通研B4 出原昇馬

#0 基本事項

初期尤度と最終尤度

初期尤度

：パラメータをすべて 0 としたときの尤度

$$L(0) = -N \cdot \ln A$$

N: サンプル数，　A: 選択肢数

最終尤度

：推定パラメータを用いて計算したときの尤度

尤度比

尤度比（ρ^2値）

：推定したパラメータによって尤度がどのくらい向上したかを表す指標.

$$\rho^2 = 1 - \frac{L(\hat{\beta})}{L(0)}$$

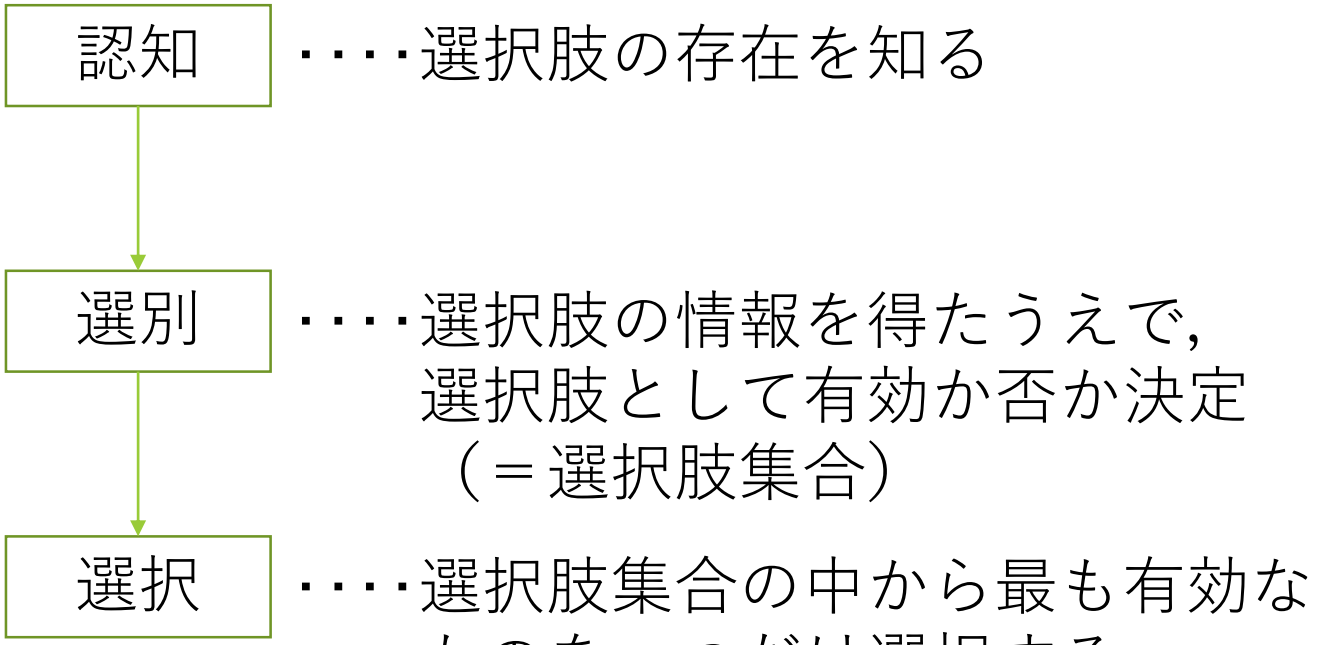
尤度比は$0 < \rho^2 < 1$の値をとり，1に近いほど好ましい.

修正済み尤度比

パラメータ数が増えると見かけ上説明力が向上してしまうため，自由度を調整して尤度比を修正する.

選択肢集合の扱い方

個人が実際に選択するまでのプロセス



参考：交通工学研究会(1993)『やさしい非集計分析』

選択肢集合の扱い方

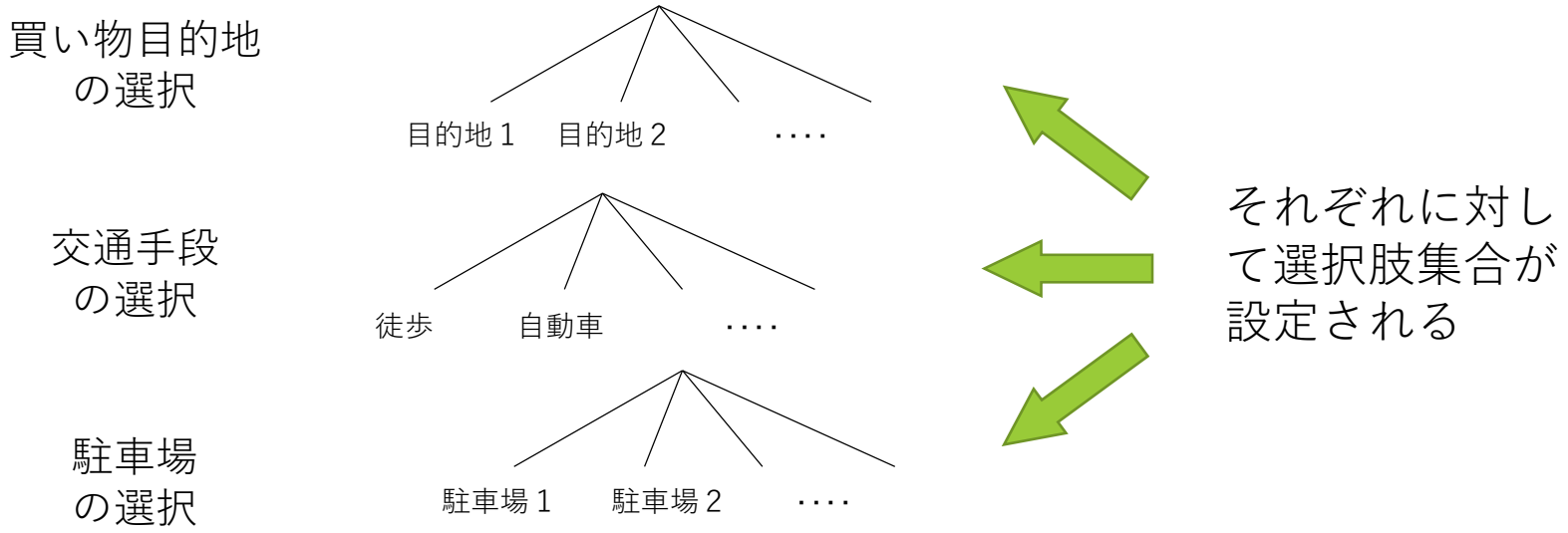
行動モデルでは，個人の利用可能な選択肢の集まり（＝選択集合）を基にして，その中から個人が効用最大となるものを選択する.

選択肢集合を如何にして設定するかというのが，重要な作業になる.

例：買い物目的地の選択問題で，500m離れたスーパーと10km離れた郊外型大型店が設定された場合，もしこの人が郊外大型店の存在を知らず，いつもスーパーに行っていたとしたら，推定されるモデルでは距離が過大評価されてしまい，この人は距離の近いほうを選んだと説明されてしまう（実際は選択肢集合になかっただけ）.

選択肢集合の扱い方

例：買い物行動のモデル（NL）



選択肢集合の扱い方

①買い物目的地

１か月間に最低１回は出かけている買い物目的地を選択肢として設定し，１度も利用していない目的地は選択肢集合に含めない←個人の買い物実績が基準

選択肢が１つしかない個人は固定層として推定から除外.

実績データがない場合は，ある地区に住んでいる人たちはみな同一の選択肢をもつと仮定し，地区ごとに選択肢集合を設定する．ただし自動車が利用可能かどうかなど，個人属性に留意する.

選択肢集合の扱い方

②交通手段

実際の利用実績を集計し，距離帯別の各交通手段の利用分布曲線を作成．これを目安に個人属性ごとに判断していく．

徒歩

：利用分布曲線から得られる歩行限界距離

自動車

：運転免許の有無・自動車を保有しているかどうか

バス

：ODがバス利用圏域にあるかどうか・バスでたどりつけるかどうか

鉄道

：駅利用限界距離（駅勢圏）

選択肢集合の扱い方

③駐車場

最終目的地に対して歩行可能な距離範囲にあるかどうかを基準にする.

郊外型SCの場合等は，通常駐車場は一つなので考慮せず，駐車場選択は中心市街地に限られる.

選択肢集合の扱い方

経路集合について

1）事前に経路集合を明示するアプローチと,

2）明示的な経路の列挙を必要としないアプローチの２つがある.

1）・k番目最短経路探索
・選択肢サンプリング
・ベイズアプローチ　etc.

2）・Dial(1971)'sアプローチ
・マルコフ連鎖アプローチ

ベイズアプローチ

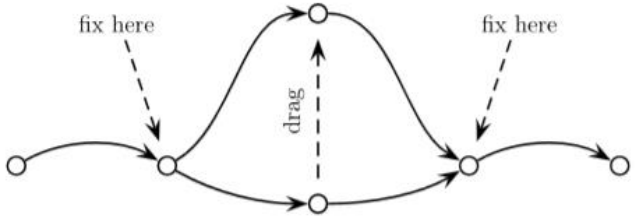


Fig. 1. "Rubber band"-like variation of a path.

- 二箇所の固定点(fix)を選定
- 一箇所の移動点を選定し，新しいポジションに移動(drag)
- すなわち，輪ゴム(rubber band)のように経路を変位させるが，その変位がネットワーク上に制限される

参照　http://bin.t.u-tokyo.ac.jp/model14/lecture/Fukuda.pdf

#1 MNL推定

MNL推定

松山都市圏PT調査データのうち，以下の８つの商業施設を目的地とする目的地選択モデルを考える.

1. ﾌｼﾞｸﾞﾗﾝ松山店
2. ﾌｼﾞｸﾞﾗﾝ重信SC
3. ﾊﾟﾙﾃｨ姫原SC
4. ﾊﾟﾙﾃｨ衣山SC
5. 伊予鉄高島屋
6. 三越　松山店
7. ジャスコ松山店
8. ジョー・プラ

MNL推定　説明変数と効用関数の設定

効用関数を以下のように設定した.

$$V_1 = \beta_1 \cdot 距離 \cdot 0.001 + \beta_3 \cdot \delta_{walk} + \beta_4 \cdot \delta_{car} + \alpha_1$$

$$V_2 = \beta_1 \cdot 距離 \cdot 0.001 + \alpha_2$$

$$V_3 = \beta_1 \cdot 距離 \cdot 0.001 + \alpha_3$$

$$V_4 = \beta_1 \cdot 距離 \cdot 0.001$$

$$V_5 = \beta_1 \cdot 距離 \cdot 0.001 + \beta_3 \cdot \delta_{walk} + \beta_4 \cdot \delta_{car} + \alpha_5$$

$$V_6 = \beta_1 \cdot 距離 \cdot 0.001 + \beta_3 \cdot \delta_{walk} + \beta_4 \cdot \delta_{car} + \alpha_6$$

$$V_7 = \beta_1 \cdot 距離 \cdot 0.001 + \beta_2 \cdot \delta_{Tu} + \beta_4 \cdot \delta_{car} + \alpha_7$$

$$V_8 = \beta_1 \cdot 距離 \cdot 0.001 + \beta_4 \cdot \delta_{car} + \alpha_8$$

δ_{Tu}:火曜日ダミー（火曜日なら１,それ以外は０）
δ_{walk}:徒歩ダミー（代表交通手段が徒歩なら１，それ以外は０）
δ_{car}:自動車ダミー（代表交通手段が自動車なら１，それ以外なら０）

MNL推定　モデルの設定

Availabilityを設定し，歩行距離5km以上の目的地は選択肢集合に含めないようにした.
収束法にはBFGSを用いた.

```
a1<-(Data[,21]!=420)+(Data[,21]==420)*(Data[,15]<5000)
a2<-(Data[,21]!=420)+(Data[,21]==420)*(Data[,16]<5000)
a3<-(Data[,21]!=420)+(Data[,21]==420)*(Data[,17]<5000)
a4<-(Data[,21]!=420)+(Data[,21]==420)*(Data[,18]<5000)
a5<-(Data[,21]!=420)+(Data[,21]==420)*(Data[,13]<5000)
a6<-(Data[,21]!=420)+(Data[,21]==420)*(Data[,14]<5000)
a7<-(Data[,21]!=420)+(Data[,21]==420)*(Data[,19]<5000)
a8<-(Data[,21]!=420)+(Data[,21]==420)*(Data[,20]<5000)
des5202<-a1*exp(d1*Data[,15]/1000                    +d7*walk +d8*car+b1*matrix(1,nrow=hh,ncol=1))
des5204<-a2*exp(d1*Data[,16]/1000                                    +b2*matrix(1,nrow=hh,ncol=1))
des5206<-a3*exp(d1*Data[,17]/1000                                    +b3*matrix(1,nrow=hh,ncol=1))
des5208<-a4*exp(d1*Data[,18]/1000                                          )
des5212<-a5*exp(d1*Data[,13]/1000                    +d7*walk        +d8*car+b4*matrix(1,nrow=hh,ncol=1))
des5213<-a6*exp(d1*Data[,14]/1000                    +d7*walk        +d8*car+b5*matrix(1,nrow=hh,ncol=1))
des5272<-a7*exp(d1*Data[,19]/1000          +d5*Tue                   +d8*car+b6*matrix(1,nrow=hh,ncol=1))
des5303<-a8*exp(d1*Data[,20]/1000                                    +d8*car+b7*matrix(1,nrow=hh,ncol=1))
deno<-(des5202+des5204+des5206+des5208+des5212+des5213+des5272+des5303)
```

MNL推定　推定結果

	距離	火曜日ダミー	徒歩ダミー	自動車ダミー	定数項						
					1	2	3	5	6	7	8
パラメータ推定値	-0.849	1.812	0.956	1.567	1.130	0.623	0.341	1.600	0.731	0.958	1.071
t 値	-10.610	3.960	2.595	2.281	2.057	0.701	0.533	2.928	1.308	1.682	1.893

$\rho^2 = 0.297$

```
> print(L0)
[1] -568.6038
> print(LL)
[1] -388.8343
> print((L0-LL)/L0)
[1] 0.3161596
> print((L0-(LL-length(b)))/L0)
[1] 0.296814
> print(b)
 [1]  1.1302188  0.6226336  0.3414119  1.6000092  0.7310363  0.9580380  1.0706952 -0.8491117  1.8119226
[10]  0.9559079  1.5670082
> print(tval)
 [1]   2.0567256   0.7013097   0.5332919   2.9278287   1.3080057   1.6821966   1.8933705 -10.6097502
 [9]   3.9597570   2.5954942   2.2806020
```

MNL推定　推定結果

hessian

1	-34.74666202	0.069881409	1.462953449	17.4257397	7.427084	3.7880344	2.9289158	0.6064951	1.1263900	-1.63868066	-1.3845267
2	0.06988141	-2.160318672	0.004118561	0.3516836	0.148806	0.8127100	0.7695416	4.3709935	0.1123285	0.03553505	1.7558029
3	1.46295345	0.004118561	-4.379595893	1.3141811	0.682750	0.3252066	0.2203257	-0.1712555	0.1095740	0.52254299	1.4277087
4	17.42573966	0.351683596	1.314181141	-56.6988697	16.387401	10.9088513	8.8631829	-1.8849124	2.6269373	-5.81433375	-1.1757564
5	7.42708441	0.148806038	0.682750027	16.3874015	-33.903035	4.8990356	3.7200426	3.6914185	1.2147383	-2.30956735	-0.5232680
6	3.78803441	0.812710013	0.325206571	10.9088513	4.899036	-29.7113975	8.6523703	-4.0977606	-6.2339255	4.82975484	-0.8636799
7	2.92891579	0.769541572	0.220325710	8.8631829	3.720043	8.6523703	-25.3829169	-2.8958598	0.9214991	3.62956990	-0.8237982
8	0.60649514	4.370993523	-0.171255500	-1.8849124	3.691418	-4.0977606	-2.8958598	-176.2465277	-4.7848016	3.03808616	-7.5954774
9	1.12638995	0.112328465	0.109573989	2.6269373	1.214738	-6.2339255	0.9214991	-4.7848016	-6.2339255	1.45677730	-0.1249758
10	-1.63868066	0.035535052	0.522542990	-5.8143337	-2.309567	4.8297548	3.6295699	3.0380862	1.4567773	-9.76258224	0.0000000
11	-1.38452667	1.755802870	1.427708710	-1.1757564	-0.523268	-0.8636799	-0.8237982	-7.5954774	-0.1249758	0.00000000	-4.7710295

MNL推定　結果と考察

- 距離パラメータが特に有意な負値となった
- 中心部に位置する3施設について徒歩ダミーを加えたところ，有意な正値となった.
- ジャスコ（イオン）松山に火曜日ダミーを加えたところ有意な正値となった.

→距離による効用逓減，および各目的地の属性による利用者の選択結果の差が示された

MNL推定　結果と考察


フジグラン 松山
フジグラン重信
フジ姫原店
パルティ・フジ衣山
伊予鉄高島屋
松山三越
イオンスタイル松山
ジョー・プラ
徒歩でアクセス
しやすい目的地
潮見山
観音山
松山市
松山自動車道

MNL推定　今回の推定の課題

- 選択肢共通変数が少なく，規模や料金等を含めた分析に至らない
- 修正済み尤度比が低く，モデルの改善の余地がある
- 近隣に位置する目的地や，個人が交通手段ごとに目的地を選択することを踏まえ，NLモデルの適用を検討できる

#2 Dijkstra法

Dijkstra法

サンプルデータをもとに

- Dijkstra法による最短経路探索
- 得られた最短経路距離をもとに目的地選択のMNL推定

を行う.

Dijkstra法

2通りのコードでDijkstraアルゴリズムを実行する.

1. if式による最小値の比較と更新

```
def search(m, ori, des):
    # 初期設定
    max_cost = float('inf')  # 初期コストは無限大
    unchecked = [False] * m_node  # 未確定ノードの集合
    cost = [max_cost] * m_node  # 起点から各ノードへの最小費用
    f_prev = [None] * m_node  # 最短経路を列挙するための配列
    cost[ori] = 0  # 起点のノードの距離は0とする
    f_prev[ori] = ori  # 起点前のノードは起点とする
    now = ori  # 現在地を起点とする

    while True:
        min = max_cost  # 変数minは現段階の最小費用を表す（初期は無限大）
        next = -1  # nextは起点から最小費用のあるノードを表す(初期－1)
        unchecked[now] = True
        for i in range(m_node):  # 変数iはノード（0~4）
            if unchecked[i]: continue  # ノードが確定していない場合ループが続く
            if m[now][i]:  # 今のノードiと起点(now=ori)が接続しているかどうか
                tmp_cost = m[now][i] + cost[now]  # 一時的の費用を計算し、最小費用の更新
                if cost[i] > tmp_cost:
                    cost[i] = tmp_cost
                    f_prev[i] = now  # 直前のノードに更新
            if min > cost[i]:  # 現段階の最小費用と最小費用を持つノードを更新
                min = cost[i]
                next = i
        now = next  # 確定された最小費用持つノードが新しい”起点”となる
        if next == -1: break  # ノード番号が－1になるまで(すべてノードが確認される)

    #print_path(f_prev, cost)  # 各接続しているノード間ルートの費用
    print(get_path(ori, des, f_prev), cost[des])
    return [get_path(ori, des, f_prev), cost[des]]
```

2. heapモジュールによる最小値探索

```
class Dijkstra(object):
    def dijkstra(self, adj, start, goal=None):
        num = len(adj)  # グラフのノード数
        dist = [float('inf') for i in range(num)]  # 始点から各頂点までの最短距離を格納する
        prev = [float('inf') for i in range(num)]  # 最短経路における，その頂点の前の頂点のIDを格納する
        dist[start] = 0
        q = []  # プライオリティキュー．各要素は，(startからある頂点vまでの仮の距離，頂点vのID)からなるタプル
        heapq.heappush(q, (0, start))  # 始点をpush

        while len(q) != 0:
            prov_cost, src = heapq.heappop(q)  # pop
            # プライオリティキューに格納されている最短距離が，現在計算できている最短距離より大きければ，distの更新をする必要はない
            if dist[src] < prov_cost:
                continue
            # 他の頂点の探索
            for dest in range(num):
                cost = adj[src][dest]
                if cost != float('inf') and dist[dest] > dist[src] + cost:
                    dist[dest] = dist[src] + cost  # distの更新
                    heapq.heappush(q, (dist[dest], dest))  # キューに新たな仮の距離の情報をpush
                    prev[dest] = src  # 前の頂点を記録
        if goal is not None:
            return self.get_path(goal, prev)
        else:
            return dist

    def get_path(self, goal, prev):

        '''
        始点startから終点goalまでの最短経路を求める
        '''
        path = [goal]  # 最短経路
        dest = goal
        # 終点から最短経路を逆順に辿る
        while prev[dest] != float('inf'):
            path.append(prev[dest])
            dest = prev[dest]
        # 経路をreverseして出力
        return list(reversed(path))
```

Dijkstra法

Dijkstra法による経路探索により，以下の結果が得られた.

配布コードによる結果

```
[0, 0] 0
[0, 1] 142.2065
[0, 1, 2] 456.6726
[0, 3] 89.68804
[0, 1, 4] 347.0229
[0, 1, 2, 5] 538.04425
[0, 3, 6] 463.57014
[0, 3, 6, 7] 702.78314
[0, 3, 6, 7, 8] 831.93554

Process finished with exit code 0
```

ヒープ構造化による結果

```
[0] 0
[0, 1] 142.2065
[0, 1, 2] 456.6726
[0, 3] 89.68804
[0, 1, 4] 347.0229
[0, 1, 2, 5] 538.04425
[0, 3, 6] 463.57014
[0, 3, 6, 7] 702.78314
[0, 3, 6, 7, 8] 831.93554

Process finished with exit code 0
```

Dijkstra法　MNL推定

効用関数を以下のように設定した.

$i = 1, \cdots, 8$

$$V_i = \beta_1 \cdot \delta_1 \cdot \frac{\text{距離}}{100} + \beta_2 \cdot \delta_2 \cdot \frac{\text{距離}}{100} + \beta_3 \cdot \delta_1 \cdot \text{規模} + \beta_4 \cdot \delta_2 \cdot \text{規模} + \alpha_i$$

$i = 8$については定数項なし

δ_1：50歳未満であれば 1，50歳以上であれば 0 を返す値

δ_2：50歳未満であれば 0，50歳以上であれば 1 を返す値

Dijkstra法　説明変数

年齢によって距離抵抗の度合いが異なると考え，50歳未満とそれ以上とでパラメータを分けた.

また，規模に関しても年齢によりパラメータを分けた.

Dijkstra法　推定結果

	距離		規模	
	50歳未満	50歳以上	50歳未満	50歳以上
パラメータ推定値	-0.097912	-0.9112354	0.30412457	0.430363
t 値	-4.85E-05	-4.51E-04	1.32E-04	1.86E-04

定数項						
1	2	3	4	5	6	7
0.21206463	-0.093127	0.34787195	0.1432542	-0.1130782	-0.535632	-0.833199
2.11E-05	-1.74E-05	3.27E-05	1.46E-05	-1.09E-05	-7.20E-05	-1.56E-04

$\rho^2 = 0.110$

```
> print(L0)
[1] -103.9721
> print(LL)
[1] -81.52935
> print((L0-LL)/L0)
[1] 0.2158534
> print((L0-(LL-length(b)))/L0)
[1] 0.1100558
> print(b)
 [1]  0.21206463 -0.09312711  0.34787195  0.14325415 -0.11307823 -0.53563163 -0.83319878 -0.09791203 -0.91123544
[10]  0.30412457  0.43036300
> print(tval)
 [1]  2.110415e-05 -1.737396e-05  3.268545e-05  1.462480e-05 -1.091157e-05 -7.198356e-05 -1.561034e-04
 [8] -4.847103e-05 -4.511052e-04  1.315009e-04  1.860853e-04
```

Dijkstra法　推定結果

hessian

1	-9.0165430	0.59737988	2.51430790	3.6868328	0.17563650	1.0560655	0.166716866	13.285339	5.500634e+00	7.241565e+00	1.376085e+00
2	0.5973799	-2.76466488	0.29352664	0.8932110	0.08380832	0.3784701	0.085836941	-1.027057	-9.797845e-01	3.922564e+00	5.775902e-02
3	2.5143079	0.29352664	-5.73099249	1.8384474	0.08559759	0.5207131	0.080948325	7.661282	5.244349e+00	1.052027e+01	1.468575e+01
4	3.6868328	0.89321105	1.83844744	-10.0289145	0.30287836	1.4754229	0.304816485	6.462710	-6.976871e+00	-1.377490e+01	-1.372761e+01
5	0.1756365	0.08380832	0.08559759	0.3028784	-0.96974026	0.1337486	0.031112808	-1.146798	-1.660416e-01	4.277460e+00	1.557266e-01
6	1.0560655	0.37847008	0.52071305	1.4754229	0.13374859	-4.3887709	0.136623473	-1.918626	-2.207920e+00	-6.232945e+00	-2.407295e+00
7	0.1667169	0.08583694	0.08094833	0.3048165	0.03111281	0.1366235	-0.967868239	-2.797994	-9.762696e-02	1.470646e+00	2.925781e-03
8	13.2853392	-1.02705701	7.66128187	6.4627098	-1.14679830	-1.9186258	-2.797994473	-161.931181	0.000000e+00	-6.746773e+01	0.000000e+00
9	5.5006338	-0.97978448	5.24434897	-6.9768713	-0.16604161	-2.2079204	-0.097626955	0.000000	-3.060975e+01	-3.552714e-09	-4.373947e+01
10	7.2415648	3.92256390	10.52027252	-13.7749034	4.27745967	-6.2329447	1.470646406	-67.467732	-3.552714e-09	-1.266907e+02	-3.552714e-09
11	1.3760848	0.05775902	14.68574628	-13.7276087	0.15572660	-2.4072948	0.002925781	0.000000	-4.373947e+01	-3.552714e-09	-9.335907e+01

Dijkstra法　結果と考察

- 十分な t 値,尤度比が出たとは言えない
- 50歳未満に比べ50歳以上の人の方が距離に対する抵抗が大きくなっている

Dijkstra法　今回の課題

- ｔ値,尤度比が十分でなく，モデル改善の余地が大いにある
- PythonによるDijkstra法の実行についても，データ処理手法の改善を検討できる