Fundamental Seminar

vol.7

Huge set of alternatives & Multi Agent Simulation

2018.5.25

Fukuda lab

Noriko KANEKO

巨大選択肢集合の取り扱い，その前に

行動モデルの構築～交通手段選択モデルの場合～

① 選択肢の列挙
{鉄道、自動車、バス、自転車、徒歩}

←今日のお話

② 選択肢間の相関や誤差構造を考慮し、モデルを選ぶ
{MNL,MNP,NL,MXL,,,}

③ 効用関数を各々定義
$\{V = Btime * time + Bcost * cost + \cdots\}$

④ モデルを推定する
{最尤推定法、シミュレーション推定法、ベイズ推定法}

←今までの
基礎ゼミのお話

巨大選択肢集合の取り扱い，その前に

選択肢の列挙

- 交通手段選択モデル
  →割と列挙可能

- 目的地選択モデル
  →列挙困難

- 立地選択モデル
  →列挙困難

- 経路選択モデル
  →列挙困難

今日のテーマはこのあたりです！！

選択肢の設定方法

① 全選択肢を分析する方法

② サンプリングによる方法
：ある確率分布に従ってランダムに少数の選択肢を取り出して，パラメーター推定に用いる方法

③ 選択肢を集計する方法
：数多くの選択肢をいくつかのグループに集計統合する手法

④ 選択肢をカテゴリー分割する手法
：選択肢をいくつかのカテゴリーに統合する方法

選択肢の設定方法～立地選択モデルの場合～

① 全選択肢を分析する方法

すべての住宅を選択肢として扱う

短所；選択肢列挙が困難,
すべての選択肢に特性値設定が困難,
計算困難

↓

限定的な選択肢しか有さないような場合には有効

選択肢の設定方法～立地選択モデルの場合～

② サンプリングによる方法

ある確率分布に従ってランダムに少数の選択肢を取り出してパラメータ推定に用いる方法

長所；①より効率的
短所；IIA特性が成り立つことを仮定，
サンプル選択肢の数が多くないと歪みが生じる

（注）各個人に対して｛選択したもの，選択しなかったものから１つサンプルし，選択肢集合とすることが多い．

選択肢の設定方法～立地選択モデルの場合～

③　選択肢を集計する方法

数多くの選択肢をいくつかのグループに集計・統合

$$P_n(i) = \frac{\exp(V_{in} + lnM_i)}{\sum_{j \in A_{gn}} (V_{jn} + lnM_j)}$$

M_i:ゾーンi内の個々の住宅の数,
Z_i:ゾーン内の全住宅の平均特性をZ_i,
$A_{gn}t$　:ゾーンiの選択肢集合を$A_{gn}t$

平均効用の修正
選択肢要素数の重み

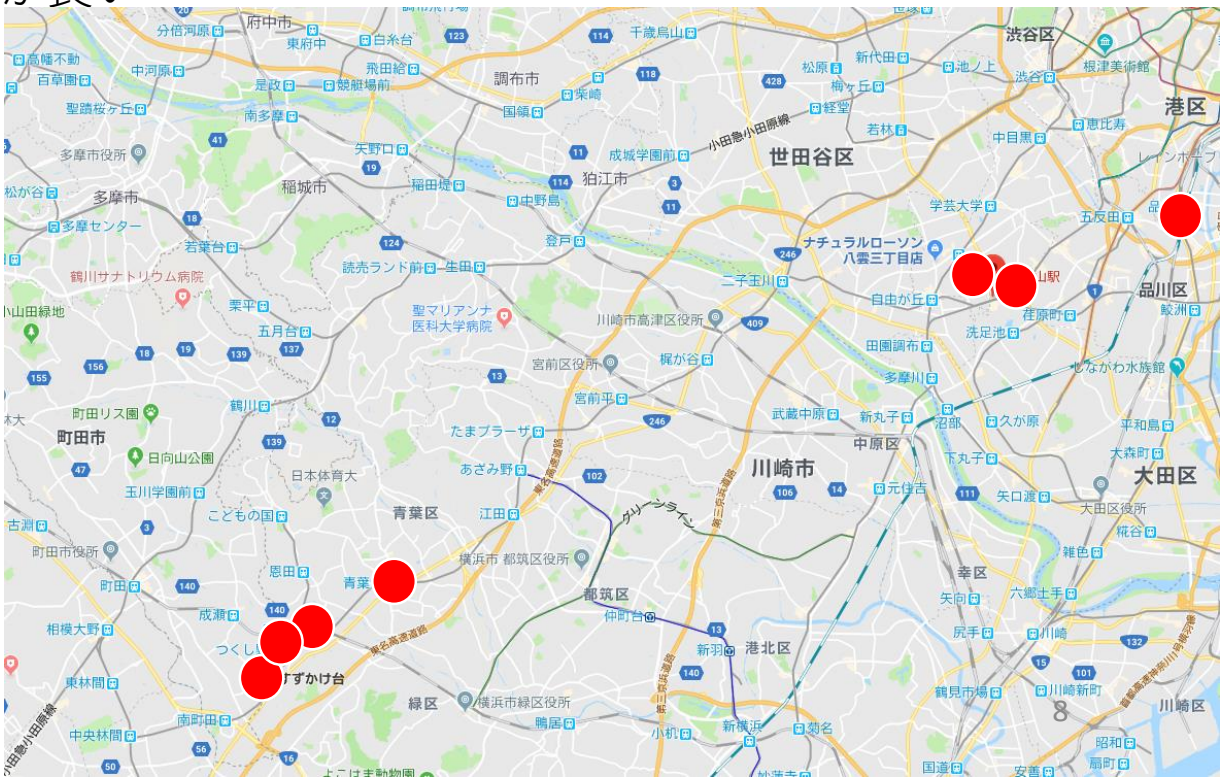
短所；M_i は，利用可能面積などに置換できるが，分析の精度にあった客観的な値の設定が困難

選択肢の設定方法～立地選択モデルの場合～

④　選択肢をカテゴリー分割する方法

各カテゴリーに属する個々の選択肢の数は考慮しない
(EX)　例えばどの寮を選ぶか？どの部屋かは考えない.

→かなり選択肢が限定されていて個々の特性が明らかな場合には有効
　そうでない場合は③のほうが良い



選択肢の利用可能性

① ヒューリスティックに選択肢集合を作る方法
…選択肢の利用可能性に制限を設けることにより選択肢集合を限定する方法
Ex)分析対象地域に境界を設ける，予算制約を設けるetc…
〇分析結果に大きな影響は出にくい
×経験的な面もある

② 各個人の選択候補の情報による方法
…アンケートで最終候補として考慮した選択肢を調査し，選択肢集合を作る
×回答そのものの信頼性
×長期的な予測には使えない

③ 選択肢集合の生成確率を考える方法
…選択肢集合が生成される確率と，選択肢集合から選択肢が選択される確率を考える

④ 検討対象集合を考える方法
…地域全域ではなく，鉄道沿線といった一部具体的な区画の検討を行う競合目的地モデル

選択肢の利用可能性

③ 選択肢集合の生成確率を考える方法

…$P_n(i) = \sum_{B_k} P_n(i|B_k) P_n(B_k)$

選択肢集合B_kからiを選択する確率

↓

ロジットモデル

選択肢の全集合A_nのすべての部分集合B_kが生成される確率

↓

- 選択肢の部分集合を１つの選択肢として求める
- $P_n(B_k) = \prod_{p \in B_k} R_{pm} \prod_{p \notin B_k} (1 - R_{pm})$
  R_{pm}:選択肢pの個人nに対する利用可能確率
  →パラメータ推定が難しい

④ 検討対象集合を考える方法

…地域全域ではなく，鉄道沿線といった一部具体的な区画の検討を行う競合目的地モデル.

$$P_{ij} = \frac{\exp(V_{ij})\, p_i(j \in M)}{\sum_k \exp(V_{ij})\, p_i(j \in M)}$$

選択肢jが比較検討する選択肢集合Mに属する確率

効用関数の推定と選択確率の導出

① ジョイントチョイス分析

② Nested Logit Model

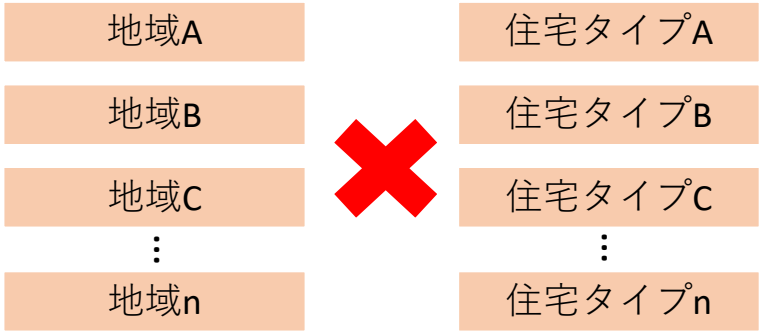
③ 識閾を考慮したNested Logit Model

④ 連続型ロジットモデルによる方法

⑤ 選択確率の動学化

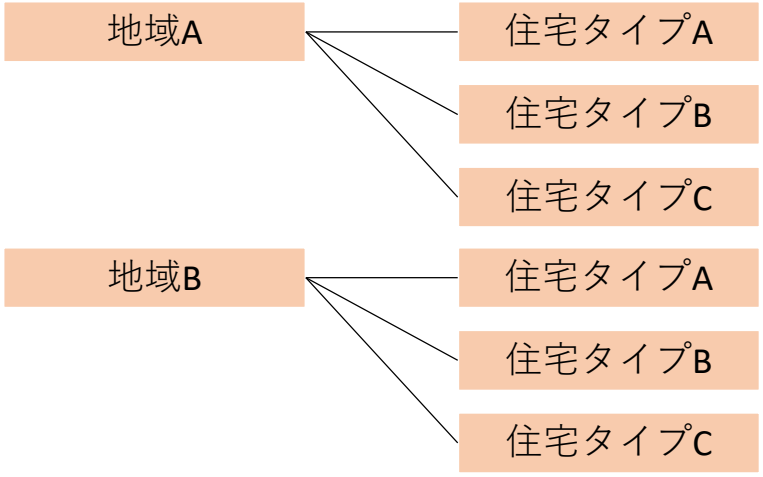
効用関数の推定と選択確率の導出

① ジョイントチョイス分析



それぞれ無関係な選択肢の
全組み合わせを選択肢として
MNLを推定する方法

② Nested Logit Model



それぞれ無関係な選択肢の
全組み合わせを選択肢として
MNLを推定する方法

効用関数の推定と選択確率の導出

③ 識閾を考慮したNested Logit Model
ロジットモデルの確率の定義に

$$\Delta U_{in} = U_{in} - U_{on} \geq \delta$$

を加え，これを満たすとき選択肢iを選択し，そうでないときは状態(0)のままであると仮定する．すると，選択肢iの選択確率は

$$P_n(i) = \frac{1}{\{1 + \exp(V_{on} - V_{in} * \delta)\}}$$

となり，定数項として推定可能となる．

④ 連続型ロジットモデルによる方法
専属的な空間において，個々の地点を選ぶ確率に対して，各選択肢にIIA特性を仮定すれば，

$$P_n(i) = \frac{M_j \exp(V_{in})}{\sum_{j \in A_{gn}} M_j \exp(V_{jn})}$$

ただし，ゾーン内の選択肢の数をM_jとする．確率密度関数は，

$$f_n(i) = \frac{\exp(V_{in})}{\int_{j \in S_n} \exp(V_{in})\, dj}$$

となる．効用関数の説明変数が簡単な連続変数で表されるときのみ有効

効用関数の推定と選択確率の導出

⑤　選択確率の動学化
立地選択行動に選択確率に時間要素を導入すること時間要素を考慮する方法.
Δtの期間に住み替える効用を

$$V_{cn} = V_{ocn} + \ln\left(\frac{\Delta_t}{t}\right)$$

のように置く．tは平均的な移転に要する時間，V_{ocn}は時間に関係しない平均効用．Δtの間にiからjに住み替える確率は

$$P_{ij}(\Delta t) = P(change|i,\Delta t)\ P(j|change,i)$$

iに居住していてΔtの間に住み替える確率

その条件のもと，jを選択する確率

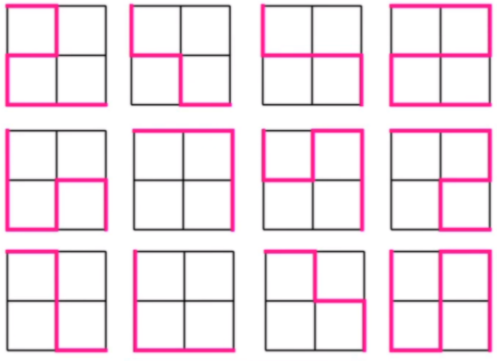
となる.

→理論提案であり，今のところ実用モデルとしては使われておらず.

経路選択モデルに対する巨大選択肢集合の取り扱い

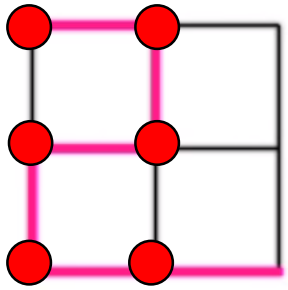
今熱い2つのアプローチ

- ZDD(Zero-suppressed BDD)
  BDD: Binary Decision Diagram
  …ZDDと呼ばれるデータ構造を用いて，
  とにかく全てを数え上げるアプローチ
  （詳しくは基礎ゼミ9にて）



全列挙（同じところを通らない）
2×2　：　12通り
3×3　：　184通り
4×4　：　8512通り

- Recursive Logit Model
  …リンクベースで再帰的にロジットモデルを
  組むことによって，経路選択肢を列挙するこ
  となく，モデルを構築するアプローチ
  （今回はこちらを）



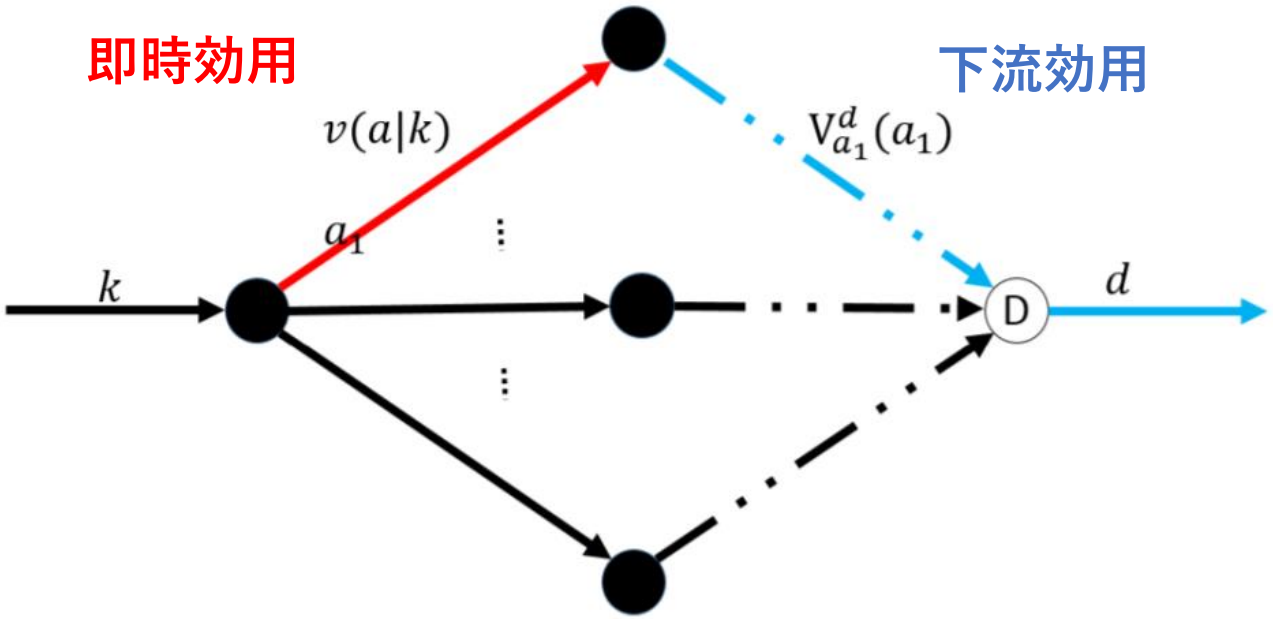
各ノードで，
効用最大化する
ようにロジット
モデルを繰り返す

Recursive Logit Model (RL Model)

- Fosgerau et al. (2013)
- Link Based Model
- 旅行者が各ノードで，即時効用＋下流効用の和を最大化するように行動する

	Path Based Model	Link Based Model
特徴	• 選択肢列挙の必要あり • 旅行者は，目的地までの一貫した効用を最大化するように行動する	• 選択肢列挙の必要なし • 旅行者は，次のリンクから得られる即時効用と，目的地までの期待効用である下流効用を最大化するように行動する
長所	• 説明変数を入れやすい	• 選択肢列挙のコストがない • 動的計画法の枠組みに準拠
短所	• 選択肢列挙が難しい • 変化するネットワークに対応しにくい	• 下流効用を算出の計算負荷が大きい • 説明変数にNon link additiveな項（料金など）を入れにくい

Recursive Logit Model (RL Model)



$G = (A, v)$: ネットワーク　v：ノードのセット　A：リンクのセット$(k, a \in A)$
k： 状態リンク　a： kにおける行動リンク　A(k) : kから出発するリンク
d： 目的地からのダミーリンク　$\tilde{A}$：$\tilde{A} = A \cup d$

- **Recursive Logit Modelの基本概念**
  旅行者nは，即時効用$v(a|k)$と下流効用$V_{a1}^d(a_1)$の和を最大化するようにリンクを選択していく

Recursive Logit Model (RL Model)

◆即時効用：次のリンクの効用

確定効用と確率効用の和

$u_n(a|k) = v_n(a|k) + \tau\varepsilon_n(a) \quad (\tau : scale\ parameter)$

$(ex)\ v(a|k) = \beta_{TT}TT + \beta_{LT}LT_{a|k} + \beta_{LC}LC + \beta_{UT}UT_{a|k}$

$Fosgerau\ et\ al.\ 2013$より

◆下流効用：目的地までの期待効用

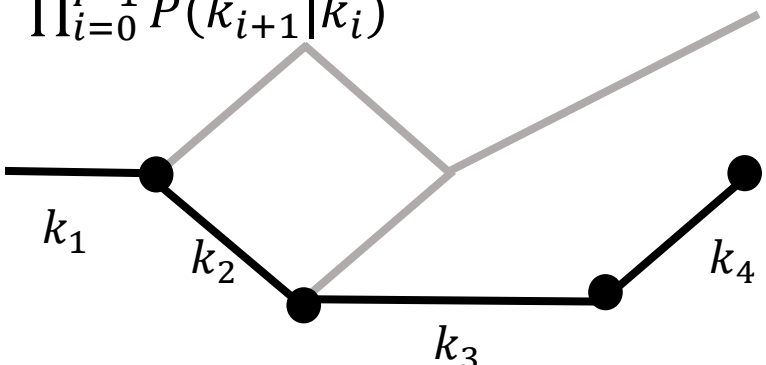
$V_n^d = E[\max(v_n(a|k) + \mu\varepsilon_n(a) + V_n^d(a))]$

◆リンク選択確率：第一種極値分布を仮定

$$P_n^d(a|k) = \frac{\exp\{v_n(a|k) + V_n^d(a)\}}{\sum \exp\{v_n(a|k) + V_n^d(a)\}}$$

◆経路選択確率：リンク選択の繰り返し

$P(\sigma) = \prod_{i=0}^{I-1} P(k_{i+1}|k_i)$



Recursive Logit Model (RL Model)

◆下流効用：目的地までの期待効用

$$V_n^d = E[\max(v_n(a|k) + \mu\varepsilon_n(a) + V_n^d(a))]$$

ログサムをとる

$$V_n^d = \begin{cases} \mu ln \sum \delta(a|k) exp \frac{1}{\mu}\{v_n(a|k) + V_n^d(a)\} & \forall k \in A \\ 0 & k = d \end{cases}$$

指数化する

$$\exp\{\frac{1}{\mu}V(k)\} = \begin{cases} ln \sum \delta(a|k) exp \frac{1}{\mu}\{v_n(a|k) + V(a)\} & \forall k \in A \\ 1 & k = d \end{cases}$$

行列で表現する

$$M_{ka} = \begin{cases} \delta(a|k) exp \frac{1}{\mu}\{v(a|k) + V(a)\} & a \in A(k) \\ 0 & otherwise \end{cases}$$

I: $\left(\left|\tilde{A}\right| \times \left|\tilde{A}\right|\right)$の単位行列　　M: $\left(\left|\tilde{A}\right| \times \left|\tilde{A}\right|\right)$の即時効用を表す行列

z: $\left(\left|\tilde{A}\right| \times 1\right)$の 下流効用を表すベクトル　b: $\left(\left|\tilde{A}\right| \times 1\right)$の 目的地を表すベクトル　とすると,

$$\boldsymbol{z} = (\boldsymbol{I} - \boldsymbol{M})^{-1}\boldsymbol{b}$$　ベルマン方程式

で表せる

Recursive Logit Model (RL Model)

◆経路選択確率：リンク選択の繰り返し

$$P(\sigma) = \prod_{i=0}^{I-1} P(k_{i+1}|k_i)$$
$$= \prod_{i=0}^{I-1} \frac{\exp 1/\mu\{v(k_{i+1}|k_i)+V(k_{i+1}))\}}{\sum_{a\in A(k_i)} \exp 1/\mu\{v(k_{i+1}|k_i)+V(k_{i+1}))\}}$$
$$= \prod_{i=0}^{I-1} exp\frac{1}{\mu}\{v(k_{i+1}|k_i) + V(k_{i+1}) - V(k_i)\}$$
$$= exp - \frac{1}{\mu}V(k_0) \prod_{i=0}^{I-1} exp\frac{1}{\mu}\{v(k_{i+1}|k_i)\}$$

ここで$v(\sigma) = \sum_{i=0}^{I-1} v(k_{i+1}|k_i)$とすると，（Ω：経路の選択肢集合）

$$P(\sigma) = {exp\frac{1}{\mu}v(\sigma)} \Big/ {\sum_{\sigma\prime\in\Omega} exp\frac{1}{\mu}v(\sigma)}$$

となり，選択肢∞のMNLと同様の形を得ることができる.

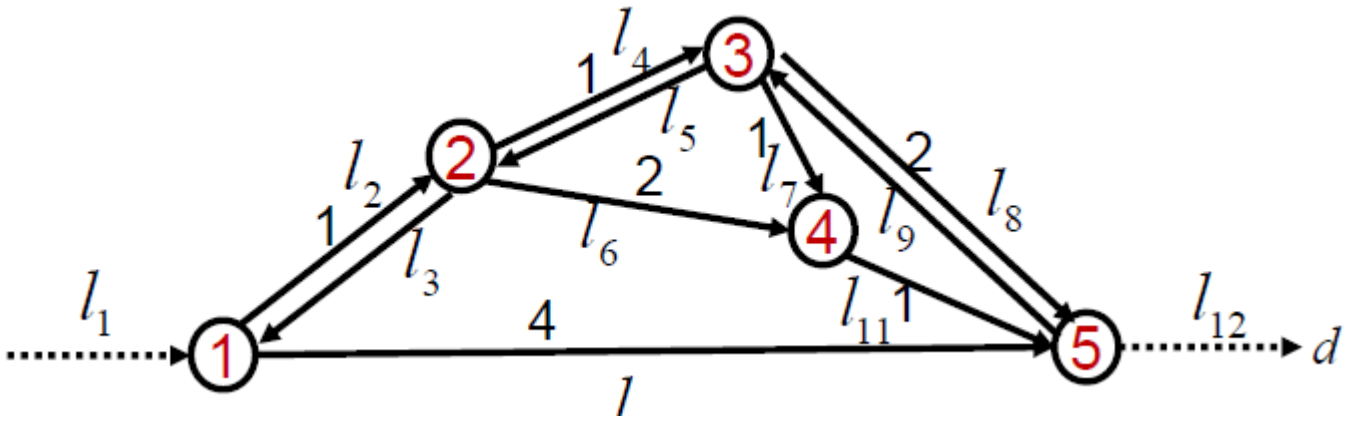
◆ 最尤推定法

対数尤度関数は,

$$LL(\beta) = ln\prod_{n=1}^{N} P(\sigma_n) = \frac{1}{\mu}\sum_{n=1}^{N}\left\{\sum_{i=0}^{I_n-1} v(k_{i+1}|k_i) - V(k_0)\right\}$$

(注)本ゼミでは基本的なRL概念の紹介にとどめておきます.
経路相関の考え方などについては，Fosgerau et al.(2013)を参考にしてください
→coding してみよう！

Coding



- Fosgerau et al. (2013)で使われている上のネットワークについて検証
- 読み込むFile
  ①link.csv
  (ネットワークデータの入ったcsv)
  ①route.csv
  (仮想観測データの入ったcsv)

※本ゼミでは兵頭先生の練習用ネットワークデータを使用しております

①link.csv

mae	ato	ATT1	ATT2
1	2	1	0
1	10	2	100
2	3	999	0
2	4	1	0
2	6	1	0
3	2	999	0
3	10	4	0
4	5	999	0
4	7	0.5	0
4	8	2	0
5	3	1	0
5	4	999	0
5	6	1	0
6	11	1	0
7	11	1	0
8	9	999	0
8	12	0	0
9	5	1	0
9	7	0.5	0
9	8	999	0
10	9	2	0
10	12	0	0
11	9	2	0
11	12	0	0

①route.csv

ID	MAE	ATO
1	1	2
1	2	4
1	4	8
1	8	12
1	12	0
2	1	2
2	2	4
2	4	8
2	8	12
2	12	0
3	1	2
3	2	4
3	4	8
3	8	12
3	12	0
4	1	2

Coding－R編

```
rm(list=ls())
library(maxLik)

link <- read.csv("link3.csv")#networkデータ読み込み
route <- read.csv("routezemi2.csv")#観測データ読み込み

AllLink <- max(link[,2]) #  吸収ノード含む全リンク
PhyLink <- max(link[,1]) #  それ以外の全リンク
DepLink <- AllLink - PhyLink# つまり目的地の数


#  疎行列を用いる（単位行列）
unit <- sparseMatrix(c(1:AllLink), c(1:AllLink), x=1, dims=c(AllLink, AllLink))
# 目的地を示すベクトル b
b <- unit[, AllLink]
b[] <- 0
b[(PhyLink+1):AllLink] <- 1

nume <- route[route[,3]!=0,]  # 最後の吸収状態を除く
deno <- route[route[,2]==1,]  # 初期状態を抽出
```

- Fosgerau(2013)でいう$A, \tilde{A}$の定義

```
# 対数尤度関数を定義
LL <- function(para){
  M <- exp( para[1]*link$ATT1 + para[2]*link$ATT2 )# 効用関数を定義
  SM <- sparseMatrix(link[,1], link[,2], x=M, dims=c(AllLink, AllLink))
  z  <- solve((unit-SM), b, sparse=TRUE)
  deno2 <- log(z[deno[,2]])
  nume2 <- log(diag(SM[nume[,2], nume[,3]]))
  LL <- (sum(nume2)-sum(deno2))
  return(LL)
}
```

- RLの対数尤度関数を記述
- リンクインシデンスマトリクスは，多くは 0 の行列となるため，疎行列 sparseMatrixを使用する
- 今回は練習のため，効用関数の説明変数は ATT1,ATT2の 2 つのみとする

```
b0 <- c(-1,-0.01)#初期値を定義
res <- maxLik(LL, start=b0)#対数尤度最大化
summary(res)
```

逆行列を求め必要があるため，Hawkins-Simon‘s conditionを満足さなければならない．初期値重要

Coding – Python編

In [ ]:
```
# 恐らくおおくの人はこのコードも必要です.
# numdiftoolsというモジュールをインストールします
# パソコンに対して一度行えばOKです
! pip install numdifftools
```

- NumdiftoolsというHessian, Gradientを求めるmoduleを入れる
- 恐らく，もともとは入っていないのでpipを使ってインストール
- コマンドプロンプトで”pip”
  あるいは
  ipython セル上で”! pip”

In [ ]:
```
#モジュールのインポート
import pandas
import numpy
import scipy
from scipy.optimize import minimize
import numdifftools
import time
```

In [ ]:
```
#リンクの性質データの入ったcsv読み込み
link_df = pandas.read_csv("Hyodo/link2.csv", header=None)
link = numpy.array(link_df)
route_df = pandas.read_csv("Hyodo/routezemi2.csv")
route = numpy.array(route_df)
```

In [ ]:
```
#吸収リンク含む
AllLink = max(link_df[1])
#吸収リンク含まない
PhyLink = max(link_df[0])
```

- Fosgerau(2013)でいう$A, \tilde{A}$の定義

In [ ]:
```
#目的地までついたデータをカット
nume = route[(route[:,2]!=0)]
#最初の選択リンクを抽出
deno = route[(route[:,1]==1)]
#前リンク(状態ｋ)
Mrow = link[:,0]
#後リンク(行動a)
Mcol = link[:,1]
```

Coding – Python編

In [ ]:
```
# #### 対数尤度関数の定義
def LL(Beta):
    #パラメータ
    B1, B2 = Beta
    Mdata = numpy.exp(B1*link[:,2] + B2*link[:,3])
    M = scipy.sparse.coo_matrix((Mdata, (Mrow, Mcol)), shape=(AllLink+1, AllLink+1)).tocsr()
    E=scipy.sparse.eye(AllLink+1, format="csr")
    b=numpy.zeros((AllLink+1))
    b[PhyLink:AllLink]=1
    z = scipy.sparse.linalg.spsolve(E-M, b)
    deno2=numpy.log(z[deno[:,1]])
    nume2 = numpy.log(M[nume[:, 1], nume[:, 2]])
    return -(nume2.sum() - deno2.sum())
```

- RL対数尤度関数の定義
- Rと同じく疎行列を使うscipy.sparse
- Rのコードと見比べてみよう！！

In [ ]:
```
#パラメータの初期値を与える
Beta0 = [-1, -0.1]

# 推定結果の追跡
Nfeval =1
def callbackF(Xi):
    global Nfeval
    print("{0:4d}    {1: 3.6f}    {2: 3.6f}    {3: 3.6f}".format(Nfeval, Xi[0], Xi[1], LL(Xi)))
    Nfeval += 1
```

- 初期値の定義
- 初期値が悪いとNan,-infなどになり推定できない

In [ ]:
```
#最尤推定の実行
t=time.time()
print('{0:4s}    {1:9s}    {2:9s}    {3:9s}    '.
      format('Iter', 'Beta1', 'Beta2', 'f(X)'))
output = minimize(LL, Beta0, method='L-BFGS-B', options={'gtol': 1e-18, 'disp': True}, callback=callbackF)
tt=time.time()
print("***********")
print("Maximum Likelihood Computation Time : %.3e sec" % (tt-t))
print("***********")
print(output)
```

In [ ]:
```
# パラメータの標準偏差・t値の計算
optBeta = output.x
print("Estimated Parameter: ", optBeta)
grad = numdifftools.core.Gradient(LL)(optBeta)
print("Gradient: ", grad)
hess = numdifftools.core.Hessian(LL)(optBeta)
print("Hessian: ", hess)
stdev = numpy.sqrt(numpy.diagonal(numpy.linalg.inv(hess)))

# 結果の出力
print('***********')
print('Max LogLL: ', -output.fun)
print('Standard deviation: ', stdev)
print('t-value: ', optBeta/stdev)
print ('Sample-size: ', route[len(route_df)-1,0]+1)
```

- 統計量の計算
- L-BFGS-Bで最適解を探索

参考文献

1. Fosgerau, M., Frejinger, E., & Karlstrom, A. (2013). A link based network route choice model with unrestricted choice set. *Transportation Research Part B: Methodological*, *56*, 70-80.

2. 日集計行動モデルの理論と考察　8章

3. 羽藤研　理論ゼミ資料

4. 兵頭先生ー講義資料，コード

5. 力石先生ーTSUゼミ講義資料，コード