数理生物学演習

第6回　疫学モデル

第6回：疫学モデル

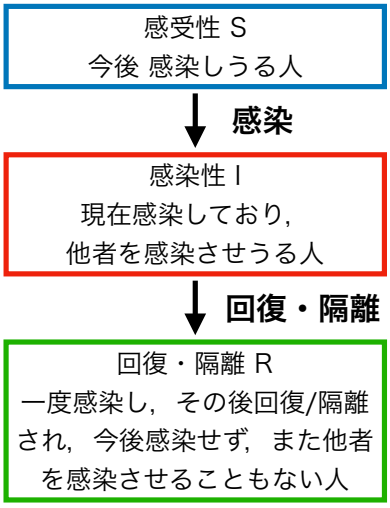
本日の目標

- SIRモデルの解析
- 基本再生産数
- 最小二乗法
- （C言語における）関数

Kermack-McKendricのSIRモデル

仮定

- 人々を感受性(susceptible, S)，感染性(infectious, I)，回復・隔離(recovered/removed, R)の３状態のいずれかにある
- 感染症は感染している人と未感染の人が接触したとき，ある確率でうつる
- 感染から回復すると免疫をもち，再び感染することはない
- 移入・移出，出生・死亡などによる“人口の増減”はない



$$\begin{cases} \dfrac{dS}{dt} = -\beta S(t)I(t) \\ \dfrac{dI}{dt} = \beta S(t)I(t) - \gamma I(t) \\ \dfrac{dR}{dt} = \gamma I(t) \end{cases}$$

β: 伝達係数
γ: 回復率・隔離率

“人口の増減なし”

$N(t) = S(t) + I(t) + R(t)$ とすると $\dfrac{dN}{dt} = 0$

基本再生産数

- １人の感染者が，感染期間中に再生産する２次感染者の期待値のこと
- 基本再生産数を r_0 とすれば，もし $r_0>1$ ならば感染症の流行が起こる

SIRモデルを仮定して，感染初期について基本再生産数を考えてみる

初期条件を $I(0)=I_0, S(0)=S_0, R(0)=R_0$ とする．
感染症が出現したごく初期において全人口のほとんどは感受性Sで占められているとすれば，感染性Iのダイナミクスは

$$\frac{dI}{dt} = (\beta S_0 - \gamma)I(t)$$

となる．

これを解くと

$$I(t) = I_0 e^{\lambda_0 t}$$

ただし，$\lambda_0 = \beta S_0 - \gamma$

よって，$\lambda_0 > 0$ の場合に感染症の流行が起こる．

整理すると $\dfrac{\beta S_0}{\gamma} > 1$

つまり，この左辺が基本再生産数 r_0 .

別の表現もできる．例えば，γは回復・隔離率なので逆数$T=1/\gamma$は回復・隔離までの期間の期待値になる．これを使って書き直すと

$$r_0 = 1 + \lambda_0 T$$

となる．

λ_0とTは実際のデータから推定しやすいケースが多いので，便利かも知れない．

ニュートン法 Newton's method

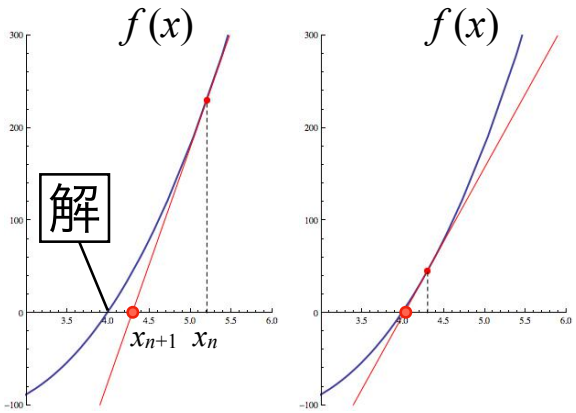
方程式を解くためのアルゴリズム

解を求めたい方程式を $f(x)=0$ とすれば,

解は $f(x)$ と x 軸との交点の x 座標になる.

では, どうやって「数値的に」求めるか？

1.解の近似値を x_n とし, 適当なその初期値 x_0 を決める
2.解の近似値 x_n での接線を求める
3.この接線と x 軸との交点を求める
4.交点の x 座標を新たに近似値 x_{n+1} として採用する
以後, 近似値が収束するまで
2.～4.を繰り返す.



x_n の漸化式を求めてみてください

最小二乗法

- 得られた測定データに対して, あるモデルを当てはめたい
- しかし, 測定データは何らかの誤差を含んでおり, パラメータを推定する必要がある
- 「残差の二乗和を最小にする」という基準に従いモデルのパラメータ推定を行う

n 個のデータの組 (t_1, x_1) , (t_2, x_2), …, (t_n, x_n) に対し,

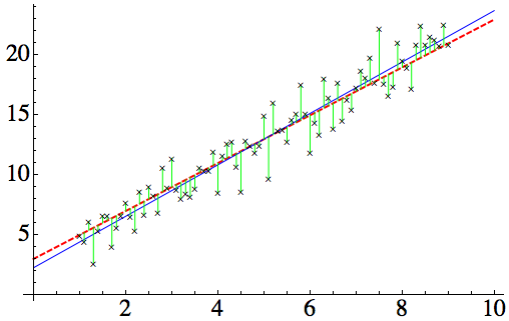
$x_i = f(t_i) + \varepsilon_i$ というモデルを当てはめるとする.

誤差 ε_i に関する仮定
- 不偏性をもつ
- 等分散である
- 無相関である
- 正規分布に従う

この $f(t)$ は m 個のパラメータ $a_1, a_2, \ldots, a_m$ で特徴づけられるとする. この $\mathbf{a} = (a_1, a_2, \ldots, a_m)$ を推定したい.

最小二乗法では
残差の二乗和 $V(\mathbf{a}|\mathbf{x}) = \sum_{i=1}^{n} [x_i - f(t_i|\mathbf{a})]^2$

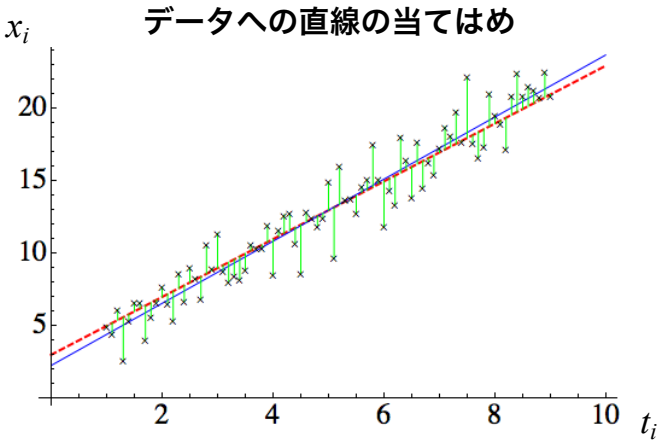
を評価関数として, これを最小にするパラメータ $\mathbf{a}$ をみつけることになる.



つまり, $\begin{cases} \frac{\partial V}{\partial a_1} = 0 \\ \frac{\partial V}{\partial a_2} = 0 \\ \vdots \\ \frac{\partial V}{\partial a_m} = 0 \end{cases}$ という連立方程式を満たす $\mathbf{a}$ を求めればよい.

非線形の連立方程式ならばニュートン法などを使う！

最小二乗法：一次方程式の例



データに対して,

$$x_i = at_i + b + \varepsilon_i$$

というモデルを当てはめる場合,
評価関数は

$$V(a,b) = \sum_{i=1}^{n} [x_i - (at_i + b)]^2$$

になる.

誤差ε_iに関する仮定
- 不偏性をもつ
- 等分散である
- 無相関である
- 正規分布に従う

この場合, $V(a,b) = 0$ は a と b について解ける.

解いてみよう！

$$a = \frac{n\sum_{i=1}^{n} t_i x_i - \sum_{i=1}^{n} t_i \sum_{i=1}^{n} x_i}{n\sum_{i=1}^{n} t_i^2 - \left(\sum_{i=1}^{n} t_i\right)^2}$$

$$b = \frac{\sum_{i=1}^{n} t_i^2 \sum_{i=1}^{n} x_i - \sum_{i=1}^{n} t_i \sum_{i=1}^{n} t_i x_i}{n\sum_{i=1}^{n} t_i^2 - \left(\sum_{i=1}^{n} t_i\right)^2}$$

実際にプログラムを組んでみよう！

関数

- C言語における関数とは，ある処理を行うモジュールのこと
- これまで使ってきた，pow()やsqrt()はもちろん関数だし，main()も関数
- 使うときは基本的に定義する必要がある.

関数プロトタイプ

```
戻り値の型　関数名(仮引数リスト);
```

関数本体

```
戻り値の型　関数名(仮引数リスト){
  処理1;
  処理2;
  …
  処理n;
  return 戻り値;
}
```

```
6-1. 関数
#include <stdio.h>

void myprint(int i);

int main(void){
  int i;
  for(i=0;i<10;i++){
    myprint(i);
  }

  return 0;
}

void myprint(int i){
  if(i%2==0){
    printf("even\n");
  }
  else{
    printf("odd\n");
  }
}
```

関数プロトタイプ

関数本体

int型の引数に対して，
偶数ならeven，奇数ならoddを
コンソールに表示させる関数

main関数の前に関数プロトタイプによる宣言をおこない，
関数の本体をmain関数の後に書くことが多い.

ファイル入出力を鍛える（1）

- ファイルへの出力だけでなく，ファイルからの入力も可能
- 例えば以下のような場合に使う
  - パラメータを変えてシミュレーションをおこなうときに毎回コンパイルするのは面倒
  - データを読み込み解析する場合に必要

ファイルを開く

```
fp=fopen(“ファイル名”，モード);
```

モード
“w”:：書き込み用
“r”：読み込み用

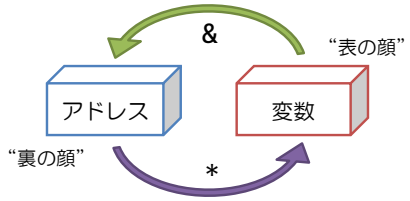
ファイルからの入力

```
fscanf(fp，入力したい内容，入力を受け取る変数のアドレス);
```

ファイルの終端 (End Of File)

```
EOF
```

別の変数等のアドレスを“値”としてもつ変数を**ポインタ**という



ポインタ演算子
*：後に続くアドレスの指す変数を返す
&：後に続く変数のアドレスを返す

ファイル入出力を鍛える（2）

6-1. データの入力

```
#include <stdio.h>

int main(void){

  FILE *fp;

  int ret;

  char name1[100],name2[100];
  int no;
  double value;

  fp=fopen(“6-1.csv","r");

  fscanf(fp,"%[^,], %s", name1,name2);
  printf("%s, %s\n",name1,name2);

  while((ret=fscanf(fp,"%d, %lf",&no, &value))!=EOF){
    printf("%d, %f\n",no, value);
  }

  fclose(fp);

  return 0;
}
```

char型の配列

注意：6-1.csvを実行ファイルと同じディレクトリに移動させる

出力例

```
No,random
1,0.668684002
2,0.689774025
3,0.648388984
4,0.785198763
5,0.515038867
⋮
```

ファイルの終端（EOF）に到達するまで，ファイルを読み込む

変換指定子(フォーマット指定子)
%s　：　文字列形式
%[^_]:　_以外を文字列として
%lf　：　倍精度小数点形式

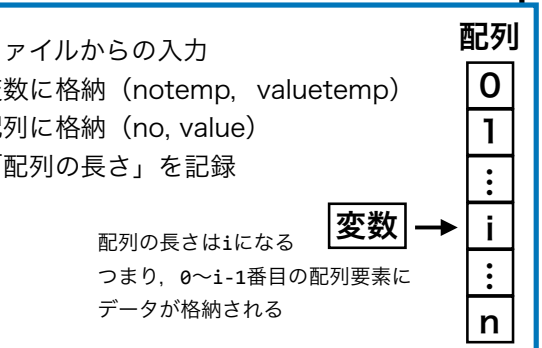
ファイル入出力を鍛える（3）

6-2. データを読み込み，利用する

```
#include <stdio.h>

int main(void){

  FILE *fp;
  int ret;

  int i,j;
  char name1[100],name2[100];
  int no[100],notemp;
  double value[100],valuetemp;

  //データを読み込む
   fp=fopen(“6-1.csv","r");
  fscanf(fp,"%[^,], %s", name1,name2);

  i=0;
  while((ret=fscanf(fp,"%d, %lf",&notemp,
&valuetemp))!=EOF){
    no[i]=notemp;
    value[i]=valuetemp;
    i++; //データを何列読み込んだかカウント
  }
  fclose(fp);

  //データを書き込む
  fp=fopen(“6-2.csv","w");
  for(j=0;j<i;j++){
    fprintf(fp,"%d,
%f\n",no[j],value[j]);
  }
  fclose(fp);

  return 0;
}
```

1. ファイルからの入力
2. 変数に格納（notemp，valuetemp）
3. 配列に格納（no, value）
4. 「配列の長さ」を記録

配列の長さはiになる
つまり，0〜i-1番目の配列要素にデータが格納される



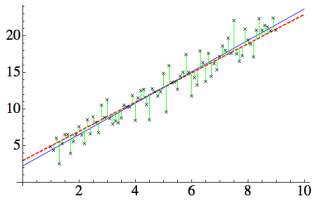
6-3. 直線当てはめ

```
#include <stdio.h>

int main(void){
  int i,n;
  double sumt,sumx,sumtt,sumtx;
  char name[100];
  double t[100],x[100];
  double a,b;
  int ret;

  FILE *fp;

  //ファイルからデータを読み込む
  fp=fopen("6-3.csv","r");
  fscanf(fp,"%[^,],%s",name,name);
  i=0;
  while((ret=fscanf(fp,"%lf,
%lf",&t[i],&x[i]))!=EOF){
    i++;
  }
  fclose(fp);

  //総和をそれぞれ計算する
   n=i;
  sumt=0;
  sumx=0;
  sumtt=0;
  sumtx=0;
  for(i=0;i<n;i++){
    sumt=sumt+t[i];
    sumx=sumx+x[i];
    sumtt=sumtt+t[i]*t[i];
    sumtx=sumtx+t[i]*x[i];
  }
  //導出した式に従い，パラメータを推定する
  a=(n*sumtx-sumt*sumx)/(n*sumtt-sumt*sumt);
  b=(sumtt*sumx-sumt*sumtx)/(n*sumtt-
sumt*sumt);

  printf("a: %f, b: %f\n",a,b);

  return 0;
}
```

$\sum_{i=1}^{n} t_i \quad \sum_{i=1}^{n} x_i \quad \sum_{i=1}^{n} t_i^2 \quad \sum_{i=1}^{n} t_i x_i$

の4つがあれば，

a, bの推定量を計算できる



$$a = \frac{n\sum_{i=1}^{n} t_i x_i - \sum_{i=1}^{n} t_i \sum_{i=1}^{n} x_i}{n\sum_{i=1}^{n} t_i^2 - \left(\sum_{i=1}^{n} t_i\right)^2} \qquad b = \frac{\sum_{i=1}^{n} t_i^2 \sum_{i=1}^{n} x_i - \sum_{i=1}^{n} t_i \sum_{i=1}^{n} t_i x_i}{n\sum_{i=1}^{n} t_i^2 - \left(\sum_{i=1}^{n} t_i\right)^2}$$

6-3.csvは $x_i = at_i + b + \varepsilon_i$ で，a=2, b=3に従うデータ．誤差ε_iは$N(\mu=0, \sigma^2=4)$に従う．

感受性 S
今後 感染しうる人

↓ **感染**

感染性 I
現在感染しており，
他者を感染させうる人

↓ **回復・隔離**

回復・隔離 R
一度感染し，その後回復/隔離され，今後感染せず，また他者を感染させることもない人

$$\begin{cases} \dfrac{dS}{dt} = -\beta S(t)I(t) \\ \dfrac{dI}{dt} = \beta S(t)I(t) - \gamma I(t) \\ \dfrac{dR}{dt} = \gamma I(t) \end{cases}$$

β: 伝達係数
γ: 回復率・隔離率

初期値
$I(0)=I_0$
$S(0)=S_0$
$R(0)=R_0$

6-4. オイラー法で離散化して，プログラムを組んでみよう．

完成したら，パラメータを様々に変化させて
モデルの挙動を見てみましょう．

発展的内容

課題の「ハード」に取り組みたい人向け

- Box-Muller法
- ニュートン法

正規分布に従う乱数の生成：Box-Muller法

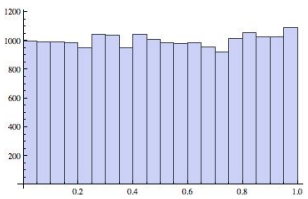
一様乱数から正規分布する乱数を生成したい

x_1, x_2は$[0, 1)$で一様分布する独立な乱数とする.
これを以下ように変換すると正規分布に従う乱数となる.

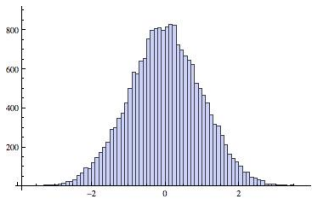
$$\begin{cases} y_1 = \sqrt{-2\log x_1}\cos(2\pi x_2) \\ y_2 = \sqrt{-2\log x_1}\sin(2\pi x_2) \end{cases}$$

y_1, y_2は$[0, 1)$で正規分布する独立な乱数

一様乱数



Box-Muller法により
生成した正規乱数



これにσ倍して，μを加えれば
$N(\mu, \sigma^2)$の正規分布に従う乱数列を生成できる.

```
6-5. Box-Muller法
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>

int main(void){
 double x1,x2,y1,y2,i,N;
 srand(time(NULL));
 N=1;

 FILE *fp;
 fp=fopen("output_rand.csv","w");

 for(i=0;i<10000;i++){
  x1=(((double)rand()+1) / ((double)RAND_MAX + 1)) * N;
  x2=(((double)rand()+1) / ((double)RAND_MAX + 1)) * N;

  y1=sqrt(-2*log(x1))*cos(2*M_PI*x2);
  y2=sqrt(-2*log(x1))*sin(2*M_PI*x2);

  fprintf(fp,"%f, %f\n%f, %f\n",x1,y1,x2,y2);
 }

 fclose(fp);
 return 0;
}
```

詳しく知りたい人は
調べてみてね

ニュートン法

$f(x)$=0を満たすxを数値的に得る

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

例．ロジスティック成長モデルの場合

$$\frac{dx}{dt} = r\left(1 - \frac{x}{K}\right)x$$

右辺の導関数は

$$f'(x) = r\left(1 - \frac{2}{K}x\right)$$

よって,

$$x_{i+1} = \frac{x_i^2}{2x_i - K}$$

を適当な初期値x0から始めて次々と求めていく.

```
6-6．ニュートン法による
ロジスティク成長モデルの平衡点の計算
#include <stdio.h>

int main(void){
  int i;
  double x,xx,K;

  //パラメータ設定
  K=100.0;

  //初期値
  x=10.0;

  for(i=0;i<1000;i++){
    xx=x*x/(2*x-K);
    x=xx;
  }

  printf("%f\n",x);

  return 0;
}
```

もう少し詳しいテキストを用意したので興味のある人は目を通してみてね

本日の課題

1. SIRモデルにおいて，回復・隔離率γの逆数Tが回復・隔離までの期間の期待値になるのは何故か．説明せよ.
2. SIRモデルを解析し，どうすれば感染症の流行を防げるか考察せよ.

ハード 3. SIRモデルのシミュレーションを改良し，感染性の個体数I(t) をファイルに出力する際に平均０の正規分布に従う偶然誤差が生じるとしてデータを生成せよ．（分散は様々に試す）

ハード 4. 3で出力したデータについて，感染初期に注目しλ_0を推定し，シミュレーションの結果と比較し考察せよ.

5. 質問，意見，要望等をどうぞ.

課題をPDFファイルにまとめて，Google フォームにて提出すること

課題4のイメージ

1. 評価関数Vを決める

データに $I_i = I_0 e^{\lambda_0 t_i} + \varepsilon_i$
というモデルを当てはめるとする

2. 評価関数Vを最小にするパラメータλ_0を見つける

   →しかし，直接解くことが出来ない

$\frac{dV}{d\lambda_0} = 0$ を満たすλ_0は？

3. ニュートン法により近似的に数値解を得る

   →満たすべき方程式を$f(\lambda_0)=0$の形にして$f'(\lambda_0)$を求める.

   →$f(\lambda_0)$と$f'(\lambda_0)$を利用して，適当な初期値から始め，λ_0の近似解を得る.

ニュートン法

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

次回予告

第7回：Raupモデル

5月28日

復習推奨

- 回転行列（行列の計算）
- Pythonの環境