MATLABプログラミング入門 Dec.9 2011 SOFT SNS

市橋　秀友

目次

1　はじめに

本資料では、SNS で報告してきた筆者の研究（最近傍探索）や動画の投稿に用いた MATLAB の簡単な使い方を、1）k-Means クラスタリングの MATLAB プログラム、2）画像ファイルを読み込んで主成分分析で圧縮する MATLAB プログラム、3）最近傍探索（全探索）の C 言語プログラム、4）主成分分析に関連した多変量解析 [2, 3] の簡単なプログラムなどを例として説明する。3）は MATLAB から C 言語のプログラムを呼び出して使用するための MEX の説明である。なお、[14036] のように記載しているのは引用文献ではなく SNS の url http://sns.j-soft.org/890060/reference/14036 の最後の 5 ケタの番号であるので、補足資料としてそちらもご覧いただきたい。

認知科学における「例からの学習」は、学習者は解説よりも例から学ぶことを好むことを言っている。計算機のプログラムは例を利用して作成すると，はるかに短時間で容易にできる。MATLAB の入門書としては文献 [1] がお勧めであるが、次章以降はできるだけ例を示すことで簡単な MATLAB をすぐにマスターしていただけるようにした。

2　k-Means クラスタリングの簡単プログラミング

まず簡単な MATLAB プログラミングの例として k-Means（Hard c-Means）クラスタリングを取り上げる [7439]。MATLAB プログラミングの練習問題であるが、いろいろな変形版も可能で、楕円の検出に用いたプログラム [11380] とその動画 [/community/67/video/11242] を SNS に掲載している。

```
 function Hcm
   Y=csvread('CsvData0.csv'); % データの読み込み
   [V]=Hcm(Y) % サブルーチン Hcm の呼び出し
% 行の最後に ';' を付けていないので変数 V の値をコマンドウインドウに表示
% Hard c-means (k-Means) clustering
 function [U]=Hcm(X)
   C=4; T=20; % クラスター数と反復回数
   n=size(X,1); p=size(X,2);
   U=rand(n,C); % メンバシップの初期値
   [Max,I]=max(U'); U=zeros(n,C);
   for j=1:n; U(j,I(j))=1; end
   D=zeros(n,C);
% 繰り返しルーチン
   for t=1:T
     for c=1:C
       if sum(U(:,c)) > 0
% クラスター中心の計算
       V(c,:)=U(:,c)'*X./sum(U(:,c));
     end
     end
     for c=1:C
       DistM=(X-ones(n,1)*V(c,:)).^ 2;
       D(:,c)=sum(DistM,2)';
     end
     [Min,I]=min(D,[ ],2);
% 距離最小のクラスターへ
     U=zeros(n,C);
     for j=1:n; U(j,I(j))=1; end
   end
```

3　主成分分析 (PCA) による高次元データの圧縮

筆者が学会 SNS に投稿している研究報告の資料では、そのデータの準備（高次元データの圧縮）のみならず、FCM 識別器 [10049]、[7457]、[13313] や近似最近傍探索の提案アルゴリズム [13174]、[13809]、[14270] に主成分分析 (PCA) を用いている。そこでの主な改善点は PCA の適用方法を工夫したことである。データセットを PCA で最大固有値に対応する固有ベクトル（主成分ベクトル）に直交する超平面で 2 分割する。このことを繰り返して完全 2 分木を構成している。細長く分布している方向 (主成分ベクトルの方向）の垂直方向に 2 分割することを繰り返すので、データの固まりの分割面（マイナー成分）の面積は小さくなり、近くにあるデータは出来るだけ同一のクラスターに含

まれる様になる。また、各クラスターに含まれるデータサンプルの数はほぼ等しくなる。

画像や高次元データの圧縮 (reduction) は MATLAB で PCA のプログラムを作成すれば非常に簡単である [9484]。画像データは、まず下記の 3 行のようにすれば 3 次元配列 Y に読み込まれる。

```
Label1=[char('PapaIchy'), num2str(i,'%02d')];
Imf=fullfile('D:¥GUIPAPA',Label1);
Y=imread(Imf,'JPG');
```

ただし、i は画像（写真）の番号であるので for ループで次々と読み込めば良い。jpeg のデジカメ写真がプログラムと同じファイルにあるなら直接 Y=imread('P1010364.jpg') ; のように指定するだけで良い。これは jpeg 画像の例であるが、tiff、gif、bmp、pmg など様々な画像ファイルが読み込める。書き方を忘れた場合は MATLAB のコマンドウインドウで help imread と入力すれば簡単な説明が表示される。画像を表示するための簡単な GUI プログラムは [9288] に記載している。MATLAB には GUI のためのツールが用意されているが、筆者は [9288] のプログラムを用途に合わせて修正して用いている。動画投稿の [11203]、[6878]、[6924] などのデモ用プログラムはすべてこれで作成したものである。プッシュボタンしか用いていないが、Slider、Check box、Radio Button なども使用できる。「Matlab GUI を作成」で検索すれば MathWorks 社の GUI 開発ツールの解説が用意されているが、筆者の投稿動画のように画像処理結果をデモとして表示するだけなら筆者のプログラムを修正して用いるのが簡単で便利である。また、動画の avi ファイルを作成する方法は粒子群最適化法 (PSO) を例に [9446] に掲載している。その動画は [/video/6853] にある。

もし何らかの計算に用いる数値データのテーブルが csv ファイルで与えられるなら

```
ImM=csvread('abc.csv');
```

とすれば 2 次元配列 ImM に読み込まれる。この場合の ImM は事前に宣言しておく必要がない。

ただし、たとえば大きな 2 次元配列 A の (i,j) 成分に for ループで順に代入していくような場合は、事前に

```
A=zeros(1000,500);
```

のように宣言しておかないと、毎回配列が大きくなっていくので計算が非常に遅くなる。したがって、大規模データセットを扱う場合は必須であるが、単に読み込むだけならいくら大きな配列でも宣言は不要で、いきなり

```
A=csvread('abc.csv');
```

でよい。

csv ファイルは、人手で入力した数値データを読み込む場合や C 言語など他の言語で開発されたプログラムにハードディスク経由でデータを渡す場合に用いることができる。2 次元配列 A を csv ファイルに書き込むには

```
csvwrite('Abc.csv', A);
```

とすればよいが、csv ファイルの読み込みや書き出しは時間がかかるので、何度も処理するファイルは MATLAB で

```
save('Abc','A','B');
```

のようにすれば、配列（変数）の A、B が Abc.mat ファイルとして save される。読み込む場合は

```
load Abc;
```

だけで、配列 A、B が一度に読み込まれる。mat ファイルはディスクスペースも効率よく使われ、読み込み (load) や書き出し (save) はかなり高速である。一般に開発段階では何度もデータの読み書きを繰り返すのが通常である。C 言語で csv ファイルや txt ファイルの読み書きをするより save や load の方が処理スピードがかなり速いので、この点からだけでも MATLAB を利用する価値がある。

以下の例は乱数でテストデータを作成しているが、画像データと同じ 3 次元配列として、画像データの扱いが分かりやすいようにしている。

```
% 乱数の初期化
rand('state',0);
% 配列の宣言とゼロクリアー
ImM=zeros(1000,32*32*3);
% 1000 件の画像の読み込み
for i=1:1000
% 画像の読み込みは imread であるが、代わりに乱数で 256 階調の画像データのように作成
    Y=floor(rand(128,128,3).*255);
% 一つ飛びに間引いて 32×32 の画像データに
```

```
    X=Y(11:2:74,11:2:74,:);
% 実数値のベクトルに
    ImM(i,:)=reshape(double(X./255),1,[ ]);
  end
```

ここからが PCA で、下の 1),2),3) のどれでやっても同じである。画像のサイズや件数でどの方法がメモリー使用量を少なくできるかを考えて使い分けるとよい。まず PCA の主成分ベクトルや平均ベクトルを求めるのに、大量の訓練データ全てを用いる必要がないことに注意すべきである。大量データの圧縮は求めた主成分ベクトルを用いて行うのでいくら大量でも計算できる。 Random Projection とよばれる PCA によらない射影でも効果的であることが多く、正確な主成分ベクトルでなくてもよい。 2) はデータセットを分割して、行列（ImC*ImC'）を計算して後で足す様にできるので件数がいくら多くても計算可能である。3) は、データ件数がたとえば 1000 件程なら、データの次元数が非常に大きくても行列（ImC*ImC'）をブロックに分割して計算できる。

```
% 50 次元に圧縮
  r=50;   Mean=mean(ImM);   n=size(ImM,1);
% 平均を引く
  ImC=ImM-ones(n,1)*Mean;
```

1) 特異値分解

主成分ベクトル P を求めるだけなので ImC は sqrt(n) で割っていない。Z と P はその各縦ベクトルは長さが１である。V は特異値の対角行列である。

```
  [Z,V,P]=svds(ImC,r);
```

2) 散布行列の固有値分解

```
  [P,V2]=eigs(ImC'*ImC,r);
```

V2=V^ 2 で、特異値の対角行列の 2 乗が固有値の対角行列である。ImC'*ImC で行列 ImC を転置して ImC に掛ける。 r をたとえば 3000 と大きくすれば復元したときの誤差は小さくなる。

3) 内積の行列の固有値分解（高次元データの場合に用いる。ただし、メモリーの制約があるのでデータ件数は 1000 以下ぐらいにし余り多くしない。）

```
  [Z,V2]=eigs(ImC*ImC',r);
  P=ImC'*Z*V2^ (-1/2);
```

次はデータ圧縮の手順で、上で用いた画像データ以外に大量の画像データがあっても以下のように処理する。下記の Y は画像を Y=imread(abc,'jpeg') で読み込んでいるとする。圧縮すべきデータが大量でも以下を繰り返すだけである。

```
% 切り取って，間引いて 32×32 の画像データにする
  Xtst=Y(11:2:74,11:2:74,:); % 矩形でないときなどは変数で指定して for ループを用いる
% ベクトルにする
  ImTst=reshape(double(Xtst./255),1,[ ]);
% 平均を引く。Mean は前述。
  ImTc=ImTst-Mean;
% 50 次元に圧縮
  Ztst=ImTc*P;
% 復元してどの程度誤差があるかを見たい時は
  ImCre=Ztst*P';
  Xre=reshape(ImCre+Mean,32,32,3);
% 復元画像を描くには
  image(Xre);
% double を含む行列に対して，カラーの強度は [0.0, 1.0] の範囲で，uint8 または uint16 を含む 行列に対して，カラーの強度は [0, 255] の範囲である。
% 平均誤差の表示
  sqrt(sum(sum(sum((Xtst-Xre.*255).^ 2)))
  ./(128*128*3))
% 固有値計算中の warning を消したいときは
  opts.disp=0;
  [P,V2]=eigs(ImC*ImC',r,'lm',opts);
```

とする。

なお、RGB を白黒のグレースケールに変換するには上記の ImM や ImTst は

```
  MonoIm=rgb2hsv(Y); % RGB を HSV に
  ImTst=reshape(double(MonoIm(:,:,3)),1,[ ]);
```

のようにして、二次元配列をベクトル化する。

4 MATLAB で C 言語のコンパイル

MATLAB の MEX を用いて C 言語のプログラムをコンパイルして MATLAB のプログラムから呼び出して使うようにすれば、計算時間やメモリー使用量を改

善できる場合が多くある。前処理のように計算時間がそれほど重要でないところはMATLABで開発工数を減らして、計算時間が性能評価に欠かせない箇所だけをC言語にして用いればよい。そのような格好の例として近似最近傍探索があげられる。Cとのデータの受け渡しをハードディスクを経由させてcsvファイルなどで行うことも可能であるが、大量データの場合は受け渡しに時間がかかる。

MEXでのMATLABとC言語とのデータの受け渡しは、行列の縦ベクトルをつないだ1次元配列として行われる。従って1件のデータベクトルが縦ベクトルになるように、必要ならば行列を転置してからCのプログラムを呼び出す必要がある。ここでは計算機のメモリー上に読み込まれているデータベースの中から、クエリー（テストデータ）に最も近いデータを探索する最近傍探索 [10957]、[13826]、[13773] のプログラムを例にして説明する。まずMATLABのプログラムを

```
% 最近傍探索（全探索）プログラム
  function BruteForce( )
  load DataF;
  [Dnum]=linearsearch(Ddg,Que');
```

とする。DataFの中にDdgとQueが入れてあるとする。Ddgは30次元300万件の訓練データ（データベース）で、1000件のクエリーがQueであるが、横ベクトルの行列であるので「Que'」として転置している。ただし、非常に大きな行列の場合は転置に時間がかかるので、事前に転置した行列を準備しておくべきである。配列Dnumには探索された最近傍データの番号が返される。

このプログラムで呼び出されるC言語のプログラムは下記のようになる。mexFunctionで1次元配列が受け渡しされ、linearsearch( ) が全探索のプログラムである。簡単なプログラムであるが、for(d=1;d<Ddim && dis<min;d++){ となっているので距離計算の打ち切り (Early Break) を用いた全探索である。minにその時点で見つかっている最も近いデータまでの距離が保存される。nod=3000000; notstd=1000; Ddim=31; はそれぞれ、訓練データ数、クエリーの数、データの次元数をプログラム中に指定しているが、MATLABから渡すようにもできる。ststd+=Ddim; は1次元配列中の該当データベクトルの位置を計算しているが、掛算よりもこのように足し算にする方が高速である。配列の要素は、Cでは0番地から指定するが、MATLABでは1番地からである。コメント行はMATLABでは % で示されるが、Cでは/* と//である。

```
/*linearsearch.c */
#include"mex.h"
void linearsearch(double *dnum, const double *ddg,
const double *que){
int j, k, d, nod, notstd, Ddim, nndata, cnt, std, ststd;
double dis, min;
nod=3000000;  notstd=1000;  Ddim=31;  cnt=0;
ststd=0-Ddim;
for(k=1;k<=notstd;k++){
      min=90000000.0; std=0-Ddim; ststd+=Ddim;
      for(j=1;j¡=nod;j++){
        dis=0; std+=Ddim;
// 次の行が距離計算で Early Break を用いている
        for(d=1;d<Ddim && dis<min;d++){
          dis+=(ddg[std+d]-que[ststd+d])*
                (ddg[std+d]-que[ststd+d]);
        }
        if(dis<min){
          min=dis; nndata=j;
        }
      } // for j
      dnum[k-1]=nndata;
} // for k
}
// 以下が MATLAB との配列受け渡しのためのプログラムである
void mexFunction( int Nreturned, mxArray
 *returned[ ], int Noperand, const mxArray
 *operand[ ]){
double *ddg, *que, *dnum;
returned[0] = mxCreateDoubleMatrix(1000,1,
              mxREAL);
ddg = mxGetPr(operand[0]);
que = mxGetPr(operand[1]);
```

```
dnum= mxGetPr(returned[0]); // return variable
linearsearch(dnum, ddg, que);
}
```

C のコンパイルは簡単で、MATLAB のコマンドウインドウから

```
>> mex linearsearch.c
```

を実行する [8531]。ただし、パスが通っていないといけないので、Make.m ファイルとして mex linearsearch.c の１行だけの MATLAB プログラム（スクリプト）を作成しておいて、コンパイルするときはそれを実行するようにした方が便利である。以下は MATLAB 7.10(R2010a) での結果であるが、初めて MEX を使用した時は、次のようにコンパイラーを選ぶようにプロンプトが出るので選択する。1 を選べば MATLAB に入っている lcc コンパイラーが使われる。Microsoft Visual C がインストールされてあればそれも選択肢に表示される。

```
>> mex linearsearch.c
Select a compiler:
[1] Lcc-win32 C 2.4.1 in C：¥ PROGRAM¥ MATLAB¥ R2010a¥ sys¥ lcc
[0] None
Compiler：1
```

Microsoft の Visual C/C++は一般に lcc よりも計算が高速になる．無料 (Available at no charge) のインストールのやり方は [11041] 参照。ただし v.7.0 のように古い MATLAB の場合は無料の Visual Studio Express Microsoft Visual C/C++は使えない.

5　計算時間とメモリー使用量を調べる

MATLAB での計算時間の表示は

```
t00 = clock; % スタート時間
Time=etime(clock,t00) % 秒単位
```

C 言語の場合は

```
t1=clock(); % スタート時間
t2=clock(); % 終了時間
 printf("elapse time= %7.0f¥n", t2-t1);
 //  単位はミリ秒
```

とする。

次に計算機のメモリー使用量を調べる [8872]。研究テーマによっては、プログラムの実行中にどれだけメモリーを使用しているかも性能評価には重要である [14270]。まず、自分の MATLAB プログラムの中で次のように書くことで、使用されているメモリー量の初期値を取得する。配列を宣言したり，load する前にまず以下の２行を書く。

```
[usr, sys] = memory;
S=usr.MemUsedMATLAB
```

そして、最後またはメモリー使用料を調べようとする箇所に

```
[usr, sys] = memory;
E=usr.MemUsedMATLAB
```

として、最後に

```
MemoryUsage=E-S
```

としてコマンドウインドウに表示する。MATLAB では実数値は倍精度で 8 バイトとられる。

6　多変量データ解析の簡単プログラミング

多変量解析 [2, 3] の MATLAB プログラミングとして、画像データを主成分分析で圧縮する方法は前述の通りである。PCA には，固有値分解や特異値分解（SVD）が用いられるが、線形最小２乗法（線形回帰）にも SVD を用いることができる。誤差の最小２乗法で回帰式（$Y = Xb$）の係数ベクトル b を求める際にも， SVD はプログラムを簡単にする強力な方法である。k-Means クラスタリングの変形版である筆者提案の楕円検出法のプログラム [11380] にも用いている。また、以下のように、いくつかの代表的な多変量解析は簡単な MATLAB プログラミングで可能である。

6.1　線形最小２乗法（線形回帰）

以下の三つの方法がある。下記の 1) は論文などに書かれている式としては最もよく見るものであるが,

実際のプログラムは，2) のように正規方程式（連立方程式 $X^\top Xb = X^\top Y$）を解くプログラムを用いる。MATLAB の ¥（バックスラッシュ）は SVD を用いて連立方程式を解く関数である。 正規方程式を満たす b は通常は存在する。しかし、観測データ（従属変数）Y について $Y = Xb$ を満たすような偏回帰係数 b は存在しない。b は最小２乗誤差となるような b としてしか求まらない。それを SVD で $Y = Xb$ からいきなり求めることができる。すなわち、MATLAB で B=X¥Y として B を Y 割る X として求める。¥ はバックスラッシュ（\) であるがプログラミングには円マークを用いる。.

以下の 1) から 3) の三つの B の計算結果はすべて同じ値になる。まず，説明のために乱数でテストデータを準備する。

```
% 乱数発生器を初期状態にリセットしサンプルデータ準備（正規乱数）
  randn('state',0);    X=randn(5,3);
% X のサンプル平均を 0 に
  Xc=(eye(5,5)-ones(5,5)./5)*X;
  Y=Xc*[2;3;6]; % サンプルデータ準備
% 平均 0，分散 0.01 の誤差をプラス
  Y=Y+randn(5,1)./10
% Y のサンプル平均を 0 に
  Yc=(eye(5,5)-ones(5,5)./5)*Y;
```

ここからが回帰係数 b の計算で、以下の 3) がお勧めである。

```
% 1) 逆行列を計算して偏回帰係数を求める
  B=inv(Xc'*Xc)*Xc'*Yc;
% 2) SVD で正規方程式を解き偏回帰係数を求める
  B=Xc'*Xc¥Xc'*Yc;
% 3) SVD で簡単に偏回帰係数を求める
  B=Xc¥Yc;
% Y の推定値を計算する。
  YY=Xc*B;
```

6.2 対応分析 (数量化分析 III 類)

ソーシャルフィルタリングや協調フィルタリング（お勧めを出すシステム）の研究では共起関係行列などとも呼ばれるが、多変量解析ではクロス集計表や分割表と呼ばれる次のような行列を対象とする。壷の発掘データ（行:発見場所, 列:壷のタイプ）として発見件数が次のように与えられているとする。MATLAB では、このように行列（２次元配列）の成分を「,」と「;」を使い分けて直接プログラム内で入力できる。

```
  Q=[ 30, 10, 10, 39;
      53,  4, 16,  2;
      73,  1, 41,  1;
      20,  6,  1,  4;
      46, 36, 37, 13;
      45,  6, 59, 10;
      16, 28,169,  5]
```

なお、同様に行列 B〜G を

```
  A=[B, C; D, E; F, G]
```

として大きなサイズの一つの行列 A にもできる。

行列 Q の対応分析の目的は生起した度数から壷と場所の似た者同士を近くにプロットして視覚的に分かりやすく分類することである。

```
  R=diag(sum(Q,2)); % 行の和
  S=diag(sum(Q,1)); % 列の和
% 中心化の代わりの基準化
  X=R^ (-0.5)*Q*S^ (-0.5);
  [Z_ast,L,P_ast]=svds(X,3); % 特異値分解
% 発見場所のプロット用，1 列目は使わない
  Z=R^ (-0.5)*Z_ast
% 壷のタイプのプロット用，1 列目は使わない
  P=S^ (-0.5)*P_ast
```

6.3 計量的多次元尺度構成法

対象相互間の距離の行列 R が与えられていている例は、ソーシャルフィルタリングやインターネット関連のテーマでもよく現れる。まず、R の非対角成分に加算定数を加えて固有値が非負になるようにしておく。

```
  load R; n=size(R,1);
```

C=eye(n)-(1/n)*ones(n); % 中心化のための行列
X=(-1/2)*C*R*C; % 内積の行列に変換
[Z,L2]=eigs(X,2) % 固有値分解

内積の行列から固有ベクトルを求めるところは前述のPCA の 3) と同じである。R から内積の行列 X を求めてその固有値分解を行っているので、得られる主成分得点 (Z*L2^ (1/2)) が多次元空間内の相対的な位置を復元している。

6.4 数量化分析 IV 類

対象相互間の類似度 (similarity) の行列として R が与えられていている場合も上記の計量的多次元尺度構成法を援用することができるが、数量化分析 IV 類を用いても良い。

R=csvread('Proximity.csv'); % 類似度のデータ
% 絶対値最大の固有値２つが正になるように定数を引く
R=R-0.2;
X=R+R'-diag(sum(R,2))-diag(sum(R,1));
[Z,L2]=eigs(X,2) % 固有値分解

6.5 正準相関分析

たとえば，食物と病気の罹患率の相関関係を知りたい場合，食物には色々な種類があり，病気も色々ある．そこで食物の種類別摂取量の線形関数と病気の種類別罹患率の線形関数を求め，その両者の値の相関が最大になるように線形関数の係数を求めれば，どの食物がどの病気に大きく関係しているかを推察することができる．プログラムは次のようになる。

% データの読み込み
X=csvread('X.csv');Y=csvread('Y.csv');
% 中心化のための行列
n=size(X,1); C=eye(n)-(1/n)*ones(n);
X=C*X; Y=C*Y; % 中心化
% 一般化固有値問題
[A,L2]=eigs(X'*Y*inv(Y'*Y)*Y'*X,X'*X,3);
B=L2(1,1)^ (-1/2)*inv(Y'*Y)*Y'*X*A(:,1);
F=X*A(:,1); G=Y*B;
Correlation=F'*G % 相関係数は L2^ (1/2) に同じ

Y がベクトルの場合は，固有ベクトル A(:,1) は回帰式の係数に等しく、例えば上記の線形回帰の 3) の特異値分解による X の pseudo inverse を用いて求められる回帰係数

A=X¥Y

とはベクトルの方向が等しい。

7 おわりに

学会 SNS では MATLAB の解説と筆者自身の研究への応用を報告してきた。本稿はそのまとめであるが、k-Means クラスタリングと最近傍探索の実際のプログラムも掲載してそこに説明を加えた。MATLAB は「例に基づく学習」の有効性を示すための格好の材料であると考えている。MATLAB を初めて使う方がそのことを実感して頂けたら幸いである。

参考文献

[1] 上坂吉則：MATLAB プログラミング入門 [改訂版]，牧野書店，2000

[2] 河口至商：多変量解析入門 I, II，森北出版, 1973

[3] 柳井晴夫：多変量データ解析法-理論と応用、朝倉書店、1994