Logistická regrese pomocí odhadu směsi

Logistickou regresí máme na mysli odhad modelu s diskrétním výstupem, který je závislý na smíšených (spojitých i diskrétních) veličinách. Standardně se uvažuje logistický model ve tvaru

$$f\left(y_t|x_t,\Theta\right)=\frac{\exp\left\{y_t x_t\Theta\right\}}{1+\exp\left\{x_t\Theta\right\}}$$

Pomocí tohoto modelu se konstruuje likelihood a numericky se maximalizuje. Ta numerická maximalizace je nepříjemná, i když konvergence většinou není dlouhá.

Model směsi se statickými komponentami má tvar

$$f\left(y_t,c_t|\alpha,\Theta\right)$$

kde y_t jsou data (výstup) a c_t je ukazovátko, které zařazuje data y_t do některé třídy klasifikace. Tedy do modelu vstupují data y_t a výstupem je diskrétní veličina c_t - přesně to, co dělá logistická regrese. Modeluje diskrétní výstup pomocí smíšených dat.

Připomeneme si, jak funguje odhad směsi. Je tvořen třemi částmi

1. Výpočet vah w pro jednotlivé komponenty, kde hlavním prvkem jsou tzv. proximity - tj. hodnoty hustoty pravděpodobnosti příslušné komponenty s dosazenými existujícími bodovými odhady parametrů a aktuálně změřenými daty, tj. $f_c\left(y_t|\hat{\Theta}_{c;t-1}\right)$, kde $c=1,2,\cdots,nc$ (pro všechny komponenty).
2. Přepočet statistik, kde data se přidávají s příslušnou váhou.
3. Výpočet bodových odhadů.

Úloha logistické regrese má ale dvě části (i) učení, (ii) testování. V prvé úloze se odhadují parametry modelu na trénovací množině dat (kde známe jak hodnotu regresního vektoru x_t (u nás jsou to data y_t), tak i výstupu y_t (u nás je to ukazovátko c_t). V druhé části výstup y_t (c_t) neznáme, měříme jen data x_t (y_t) a hodnotu výstupu odhadujeme s pomocí modelu.

(i) fáze učení

V bodu 1. podle hodnoty ukazovátka vybereme odpovídající komponentu a pro ní uděláme body 2. a 3. To opakujeme pro všechna trénovací data.

Poznámka

Parametry komponent jsou střední hodnoty komponent. Jejich odhadem jsou průměry z dat patřících dané komponentě. Takže výpočet je následující:

V trénovacích datech najdeme všechny položky, kde výstup je 1.

Z těchto položek určíme průměr a máme bodový odhad parametrů komponenty 1.

To opakujeme pro všechny různé hodnoty výstupu.

(*ii*) fáze testování

Pro změřená data provádíme jen bod 1. Výsledkem je odhad ukazovátka, což pro nás je odhadnutá hodnota výstupu.

Poznámka

Bod 2. a 3. můžeme samozřejmě provádět dále, ovšem s tím, že neznáme skutečnou hodnotu výstupů. Nicméně odhad funguje jako odhad směsi a parametry se mohou průběžně upřesňovat.

Aplikaci tohoto postupu ukážeme pro modely s normálním a rovnoměrným rozdělením a také pro čistě diskrétní modely.

Program pro normální komponenty je

```
// Logistic regression based on mixtures
// -------------------------------------
[u,t,n]=file();                          // find working directory
chdir(dirname(n(2)));                    // set working directory
clear("u","t","n")                       // clear auxiliary data
exec("ScIntro.sce",-1),mode(0)           // intro to sesion

nL=1000;                      // number of data for learning
nT=150;                       // number of data for testing

// Simulation --------------------------
nd=nT+nL;
x=rand(2,nd,'n');
x1=[x; ones(1,nd)];
ys=[1,1,2]*x1+.001*rand(1,nd,'n');
y=zeros(1,nd);
s1=find(ys<0);                y(s1)=1;
s2=find((ys>=0) & (ys<3));    y(s2)=2;
s3=find(ys>=3);               y(s3)=3;

// POZOR - xL musí mít veličiny v řádcích
yL=y(1,1:nL); xL=x(:,1:nL);
// Learning ----------------------------
[Est,al]=lrLearn(yL,xL);

tT=(1:nT)+nL;
// POZOR - xT musí mít veličiny v řádcích
xT=x(:,tT);
// Testing -----------------------------
yT=lrTest(xT,Est,al);

// Results -----------------------------
yR=y(tT)';
s=1:nT;
plot(s,yR,'bo',s,yT,'rx','markersize',8)
```

```
set(gcf(),'position',[700 100 600 400])

printf('Wrong %d from %d\n',sum(yR~=yT),nT)

//tx=['r';'b';'g';'m';'c';'y'];
//for i=1:3
//  s=find(y==i);
//  plot(x(s,1),x(s,2),'.'+tx(i))
//end


function [Est,al]=lrLearn(y,x)
  // [Est,al]=lrLearn(y,x)
  //           learning of Mixture logistic regression model
  // y    class label (column - nd x 1)
  // x    data vectors (vectors in rows - nd x nx)
  // Est  estimation structure
  // al   stationary probs.

  nc=max(y);
  n1=min(y);
  if n1~=1,
    disp('Error: y must start with 1');
    Est=[]; al=[];
    return
  end
  [n1,n2]=size(x);
  if n1>n2, disp('Variables in x should be rows'), end
  for i=1:nc
    c=find(y==i);
    Y=y(c);
    X=x(:,c);
    th=mean(X,'c');
    cv=cov(X');
    if det(cv)<1e-5,
      cv=.1*eye(cv)+cv;
    end
    Est(i).th=th;
    Est(i).cv=cv;
  end
  ga=vals(y);
  al=ga(2,:)/sum(ga(2,:));
endfunction

function ct=lrTest(x,Est,al)
  // ct=lrTest(x,Est,al)      test data with model from "lrLearn"
  //                          Mixture logistic regressiom
  // x    tested data item  (row)
  // Est  estimation structure  Est.th, Est.cv
```

```
  // al   stationary komponents probabilities
  // ct   estimated class labels

  [n1,n2]=size(x);
  if n1>n2, disp('Variables in x should be its rows'), end

  nc=max(size(Est));
  nd=size(x,2);
  md=zeros(nc,nd);
  for t=1:nd
    xt=x(:,t);
    dL=zeros(1,nc);
    for i=1:nc
      [xxx dL(1,i)]=GaussN(xt,Est(i).th,Est(i).cv);
    end
    dL=dL-max(dL);
    d=exp(dL);
    mm=(d(:)').*(al(:)');
    md(:,t)=mm'/sum(mm);
  end
  [xxx ct]=max(md,'r');
endfunction
```

Popis programů

Hlavní program je velice jednoduchý. V prvé části se provádí simulace dvou spojitých veličin (matice x) a ta je dále doplněna jedničkami (pro uvažování konstanty modelu). Z nich se potom konstruuje diskrétní výstup y. Potom se volá funkce provádějící učení modelu a díle funkce s testováním.

Funkce lrLearn

Po testu, zda hodnoty y začínají jedničkou se v cyklu pro každou komponentu určí data, příslušející komponentě, spočtou se parametry: průměr (centrum komponenty) a kovarianční matice.

Poznámka

Algoritmus je velice rychlý. Stačí tolik průchodů, kolik je komponent. V každém kroku se vždy zpracují všechna data, příslušející komponentě.

Funkce lrTest

Zde se na základě odhadnuté hustoty pravděpodobnosti normálního rozdělení pro každý testovaný datový vektor spočtou proximity (hodnota hustoty pravděpodobnosti v bodě daném aktuálním datovým vektorem). Ty se znormalizují na součet jedna a tak dostaneme aktuální váhy komponent. Maximální hodnoty v každém váhovém vektoru určují hodnotu odhadu výstupu.

Program pro rovnoměrné komponenty je následující

```
// Logistic regression based on mixtures (uniform distribution)
// -------------------------------------------------------------
[u,t,n]=file();                          // find working directory
chdir(dirname(n(2)));                    // set working directory
```

```
clear("u","t","n")                      // clear auxiliary data
exec("ScIntro.sce",-1),mode(0)          // intro to sesion

nL=10000;
nT=30;
nd=nT+nL;
x=rand(nd,2,'n');
x1=[x ones(nd,1)];
ys=x1*[.1,-3,2]'+.001*rand(nd,1,'n');
mi=min(ys);
ma=max(ys);
mm=ma-mi;
m1=mi+9*mm/20;
m2=mi+11*mm/20;
y=zeros(nd,1);
s1=find(ys<m1);                y(s1)=1;  x(s1,:)=x(s1,:)+2; X1=x(s1,:);
s2=find((ys>=m1) & (ys<m2));   y(s2)=2;                     X2=x(s2,:);
s3=find(ys>=m2);               y(s3)=3;  x(s3,:)=x(s3,:)-2; X3=x(s3,:);

yL=y(1:nL); xL=x(1:nL,:);
[Est,al]=lrLearnU(yL,xL);

tT=(1:nT)+nL;
xT=x(tT,:);
[yT,wt]=lrTestU(xT,Est,al);

yR=y(tT)';
yP=[1 2 3]*wt;
s=1:nT;
plot(s,yP,'o',s,yT,'x','markersize',8)
set(gcf(),'position',[700 100 600 400])

printf('Wrong %d from %d\n',sum(yR~=yT),nT)

tx=['r';'b';'g';'m';'c';'y'];
set(figure(2),'position',[50 150 600 400])
set(gcf(),'background',8)
for i=1:3
  s=find(y==i);
  plot(x(s,1),x(s,2),'.'+tx(i))
end


function [Est,al]=lrLearnU(y,x)
  // [Est,al]=lrLearn(y,x)    learning of Mixture logistic
  //                          regression model - pyrymida
  // y    class label (column - nd x 1)
  // x    data vectors (vectors in rows - nd x nx)
  // Est  estimation structure
  // al   stationary probs.
```

```
  nc=max(y);
  n1=min(y);
  if n1~=1,
    disp('Error: y must start with 1');
    return
  end
  for i=1:nc
    c=find(y==i);
    Y=y(c);
    X=x(c,:);
    thU=max(X,'r');
    thL=min(X,'r');
    Est(i).thL=thL;
    Est(i).thU=thU;
    Est(i).Cen=(thL+thU)/2;
    Est(i).Bas=prod(thU-thL); // normaizing constant od distribution
    ga(i)=length(Y);          // number of data in a component
  end
  al=ga/sum(ga);
endfunction

function [ct,md]=lrTestU(x,Est,al)
  // ct=lrTest(x,Est,al)      test data with model from "lrLearn"
  //                          Mixture logistic regressiom
  // x    tested data item
  // Est  estimation structure  Est.th, Est.cv
  // al   stationary komponents probabilities
  // ct   estimated class labels

  nc=max(size(Est));
  nd=size(x,1);
  md=zeros(nc,nd);
  for t=1:nd
    xt=x(t,:);
    b=unifdst(xt,Est);
    mm=b.*al;
    sm=sum(mm);
    for c=1:nc
      if sm>0
        mq(c)=mm(c)/sm;
      else
        mq(c)=%nan;
      end
    end
    md(:,t)=mq;
  end
  [xxx ct]=max(md,'r');
endfunction
```

```
function f=unifdst(x,Est)
  // Distsance of x from centers of individual component
  nc=max(size(Est));
  n=length(x)+1;
  for c=1:nc
    V=n/Est(c).Bas;      // normalization
    L=Est(c).thL;        // lower border
    U=Est(c).thU;        // uuper border
    C=Est(c).Cen;        // center
    Z=(U-C);             // new center
    xt=x;
    xn=(xt-C)./Z         // relative distance of x from center
    xa=abs(xn);          // absolute distsnces
    if sum(xa>1)>0       // computation of distances
      f(c)=0;
    else
      [xxx i]=max(xa);
      f(c)=(1-xa(i))*V;
    end
  end
endfunction
```

Popis programu

Hlavní program je prakticky stejný jako pro normální komponenty. Liší se jen v tom, že volá procedury pro rovnoměrné komponenty.

Odhad je založen na hledání maxim a minim dat.

Při testování se počítá proximity (a to v proceduře unifdst) jako vzdálenost dat od centra komponenty. Aktivní komponenta se určí jako ta, co aktuálním datům nejblíže.

Program pro diskrétní komponenty je

```
// Logistic regression based on mixtures
// -------------------------------------
[u,t,n]=file();                          // find working directory
chdir(dirname(n(2)));                    // set working directory
clear("u","t","n")                       // clear auxiliary data
exec("ScIntro.sce",-1),mode(0)           // intro to sesion
rand('seed',0)

nL=1000;                      // number of data for learning
nT=150;                       // number of data for testing

// Simulation --------------------------
nd=nT+nL;
x=fix(3*rand(nd,2,'u'))+1;
x1=[x ones(nd,1)];
ys=x1*[1,1,2]'+.001*rand(nd,1,'u');
y=zeros(nd,1);
```

```
s1=find(ys<6);                y(s1)=1;
s2=find((ys>=6) & (ys<7));    y(s2)=2;
s3=find(ys>=7);               y(s3)=3;

// POZOR - xL musí mít veličiny v řádcích
yL=y(1:nL)'; xL=x(1:nL,:)';
// Learning ---------------------------
[Est,al,tab,bx]=lrLearnD(yL,xL);

tT=(1:nT)+nL;
// POZOR - xT musí mít veličiny v řádcích
xT=x(tT,:)';
// Testing ----------------------------
[yT,wt]=lrTestD(xT,Est,al,tab,bx);

// Results ----------------------------
yR=y(tT)';
s=1:nT;
plot(s,yR,'bo',s,yT,'rx','markersize',8)
set(gcf(),'position',[700 100 600 400])

printf('Wrong %d from %d\n',sum(yR~=yT),nT)

function [Est,al,tab,bx]=lrLearnD(y,x)
  // [Est,al]=lrLearnD(y,x)    learning of Mixture logistic
  //                           regression model - categorical
  // y    class label (column - nd x 1)
  // x    data vectors (vectors in rows - nd x nx)
  // Est  estimation structure
  // al   stationary probs.

  nc=max(y);
  n1=min(y);
  if n1~=1,
    disp('Error: y must start with 1');
    return
  end
  bx=max(x,'c');
  nv=prod(bx);
  tab=[];
  for i=1:nc
    c=find(y==i);
    Y=y(c);
    X=x(:,c);
    nx=size(X,2);
    hs=1e-8*ones(nv,1);
    for t=1:nx
      k=psi2row(X(:,t),bx);
      hs(k)=hs(k)+1;
    end
```

```
    tab=[tab hs];
  end
  for i=1:nc
    Est(i).th=fnorm(tab(:,i)');
  end

  ga=vals(y);
  al=ga(2,:)/sum(ga(2,:));
endfunction

function [ct,wt]=lrTestD(x,Est,al,tab,bx)
  // ct=lrTestD(x,Est,al)      test data with model from "lrLearnD"
  //                          Mixture logistic regressiom - categ.
  // x    tested data item
  // Est  estimation structure  Est.th, Est.cv
  // al   stationary komponents probabilities
  // ct   estimated class labels

  nc=max(size(Est));
  nd=size(x,2);
  ct=zeros(1,nd);
  wt=zeros(nc,nd);
  tbN=fnorm(tab+1e-8,2);
  for t=1:nd
    k=psi2row(x(:,t),bx);
    wt(:,t)=tbN(k,:)';
    [xxx, ct(t)]=max(tbN(k,:));
  end
endfunction
```

Popis programu

Hlavní program generuje jen diskrétní veličiny.

Učení spočívá v odhadu diskrétního modelu.

Testování je v podstatě jednokroková predikce výstupu z diskrétního modelu.