Metodos numéricos avanzados
Mercado Financiero : Informe
Trabajo práctico

Kenny Kevin 49262
Lata Andrea 48102
Pose Jimena 49015
Sal-lari Julieta 49629

29 de septiembre de 2011

Índice

1. Introducción

1.1. Objetivos

El Objetivo del presente trabajo es aproximar mediante cuadrados minimos el indice Standard and Poor's 500 entre los años 1926 y 1993. Para lograr esto utilizamos la herramienta *GNU Octave*

2. Metodo Cuadrados minimos

Para cualquier matriz $\mathbf{A}$ m x n y cualquier vector m $\mathbf{b}$ existe una solución $\vec{x}$ de cuadrados minimos para:

$$\mathbf{A}\vec{x} = \mathbf{b}. \tag{1}$$

$\vec{x}$ es una solución si y solo si $\vec{x}$ es una solución de las ecuaciones normales:

$$\mathbf{A}^T\mathbf{A}\vec{x} = \mathbf{A}^T\mathbf{b}. \tag{2}$$

Si el rango de A es igual al número de columnas , entonces $\mathbf{A}^T\mathbf{A}$ es también de rango completo y existe una sola solución al sistema en la Ec.2. En este caso, existen varios caminos para resolver el problema de cuadrados mínimos:

- **Eliminacion Gaussiana**
- **Cholesky** La matriz $B = \mathbf{A}^T\mathbf{A}$ que aparece en la Ec. 2 tiene dos propiedades interesantes: es simétrica y definida positiva, entonces se puede asegurar que existe G triangular superior con sus elementos en la diagonal positivos tal que:

  $$B = \mathbf{G}\mathbf{G}^T$$

  La idea es resolver el sistema en la Ec. 2 en dos partes:

  1. se encuentra, mediante sustitución hacia adelante, $\vec{y}$ tal que $\mathbf{G}\vec{y} = \mathbf{A}^T\mathbf{b}$.
  2. se busca, mediante sustitución hacia atrás, $\vec{x}$ tal que $\mathbf{G}^T\vec{x} = \vec{y}$.

  Un algoritmo para encontrar G, con $i = 1, ..., n$ y $j = i + 1, ..., n$, es:

  $$\mathbf{G}_{ii} = \sqrt{b_{ii} - \sum_{k=1}^{i-1} \mathbf{G}_{ik}^2}$$

  $$\mathbf{G}_{ji} = (b_{ji} - \sum_{k=1}^{i-1} \mathbf{G}_{jk}\mathbf{G}_{ik})/\mathbf{G}_{ii}$$

- **QR** Si A es una matriz de m x n con columnas linealmente independientes, y si $A = QR$ es una factorización QR, la unica solución $\vec{x}$ de la Ec. 1 se expresa teóricamente con

  $$\vec{x} = R^{-1}Q^Tb$$

  y puede calcularse resolviendo el sistema

  $$R\vec{x} = Q^Tb$$

  Existen diferentes métodos para calcular la factorización QR, en este caso particular explicaremos como hacerlo mediante las transformaciones de Householder.

3. Modelo

Deseamos ajustar el índice Standard and Poor's $\vec{s}$ a la función:

$$y_1(t) = e^{A_0 + A_1 t + A_2 t^2} \tag{3}$$

donde y_1 es el valor del índice y t es el tiempo. Para ello, se realizará el ajuste por cuadrados mínimos del logaritmo natural de índice a la función

$$y_2(t) = A_0 + A_1 t + A_2 t^2 \tag{4}$$

4. Resultados

Con los datos de entrada provistos por la catedra en dos columnas, donde la primera es el vector correspondiente al tiempo y la segunda es el vector correspondiente al indice. Se normalizaron los datos como sugiere el enunciado del trabajo practico y se utilizaron los tres métodos explicados anteriormente para ajustarlos.

4.1. Ajuste con algoritmo propio

Llegamos al mismo valor de ajuste a traves de los métodos de Cholesky y de eliminacion Gaussiana; los valores obtenidos se pueden observar en la tabla 1. Estos valores no presentan diferencias con los resultados obtenidos con los algoritmos que provee octave, por lo tanto omitimos dicha información.

Coeficiente	valores
A_0	3.63807
A_1	1.07534
A_2	0.15304

Cuadro 1: Valores de los coeficientes obtenidos por cuadrados mínimos

El error cuadrático del ajuste $||ln(\vec{s}) - y_2(t)|| = 45{,}115$ y $||\vec{s} - y_1(t)|| = 2703{,}8$

4.1.1. Problemas con la factorización QR

La factorización QR no soporta la cantidad de datos provistos por la catedra, esto sucede tanto en nuestra implementacion del método como la propia de Octave. Pudimos comprobar que el codigo soporta hasta aproximadamente 2000 datos. Con esa cantidad de datos podemos asegurar que nuestro código funciona correctamente, ya que nos retorna el mismo valor de ajuste que los otros métods.

5. Conclusión

5.1. Relacion entre los errores calculados

No se ha encontrado relacion entre los dos errores.Lo que se puede concluir es que al utilizar el método de cuadrados minimos sobre la aproximacion lineal de la funcion original, no se garantiza la minimizacion del error cuadrático. Por eso observamos que el error de aproximacion de la funcion y_2 es menor que el error de y_1 ya que el ajuste se aplico a la funcion linealizada (y_2).

6. Bibliografia

- Fierens, Pablo. *Cuadrados mínimos: repaso*, Métodos Numéricos Avanzados. ITBA, 2do Cuatrimestre 2011
- John H. Mathews. *Métodos Numéricos con MATLAB*
- G. Strang. *Algebra lineal y sus aplicaciones*
- URL: *www.mty.itesm.mx/etie/deptos/m/ma95-843/lecturas/l843-84.pdf*

7. Anexo

7.1. Funciones de Octave

7.1.1. test

La función test carga los datos otorgados por la catedra, los normaliza y calcula el método de cuadrados mínimos por los métodos requeridos (excepto con el método QR, ya que no soporta esta cantidad de datos). Calcula error cuadrático de los dos ajustes y realiza los gráficos de aproximación para las dos funciones.

test.m

```
function test
Y = load("datos.txt");
t = normalizar(Y(:,1));
s = Y(:,2);

printf("Cuadrados minimos con eliminacion Gaussiana.\n")
Xgauss = cuadminGauss(Y)

printf("Cuadrados minimos por Cholesky.\n")
Xchol = cuadminCholesky(Y)

%printf("Cuadrados minimos por QR.\n")
%Xqr = cuadminQR(Y)


printf("Cuadrados minimos con algoritmos de Octave.\n")
A = [t.^0 t.^1 t.^2];
b= log(s);
Xoctave = A\b


X = Xgauss;
printf(" y2(t) = %g + %gt + %gt^2\n", X(1), X(2), X(3));


printf("Error cuadratico del ajuste norm(ln(s)-y2(t)):\n")
E1 = norm(log(s) - y2(X(1), X(2), X(3), t))

printf("Error cuadratico del ajuste norm(s-y1(t)):\n")
E2 = norm(s - y1(X(1), X(2), X(3), t))
log(E2)

plot(t, log(s), ";Original;",  t, y2(X(1), X(2), X(3), t), ";Ajustado;");
title("Ajuste cuadrados minimos del logaritmo natural del indice Standard and Poor's");
print("-dpng", "cuadradosMinimos.png");

plot(t, s, ";Original;",  t, y1(X(1), X(2), X(3), t), ";Ajustado;");
```

```
title("Mercado Financiero");
xlabel("Tiempo normalizado");
ylabel("Indice Standard and Poor's");
print("-dpng", "Ajuste.png");
```

7.1.2. testQR

La función testQR se usa para probar el funcionamiento del método QR. Como éste no soporta la cantidad de datos otorgados, se hace una prueba con solo 2000 datos (archivo datos2.txt).

testQR.m

```
function testQR
Y = load("datos2.txt");
t = normalizar(Y(:,1));
s = Y(:,2);

printf("Cuadrados minimos por QR.\n")
Xqr = cuadminQR(Y)


printf("Cuadrados minimos con algoritmos de Octave.\n")
A = [t.^0 t.^1 t.^2];
b= log(s);
Xoctave = A\b
```

7.1.3. cuadminGauss

cuadminGauss.m

```
function [ X ] = cuadminGauss (Y)
%inciar las variables
T = Y(:,1);
T = normalizar(T);
A = [T.^0  T.^1  T.^2];
b= log(Y(:,2));
B = A'*A;
c = A'*b;
C = [B c];
k = 2;


%eliminacion Gaussiana de B
[rows,cols]=size(C);
for i=1:rows-1
  B1 = C(i,i);
  for j=k:rows
    C(j,:) = B1*C(j,:) - C(j,i)*C(i,:);
  end
  k = k + 1;
end

%Ya tenemos C Gaussiana, vemos si es S.C.D
for i=1:rows
if(C(i,cols) == 0 || C(i,i) == 0)
error("el sistema no es compatible determinado, no tiene solucion\n")
end
end
%suponemos que no tengo nada igual a cero en la diagonal

X = zeros(rows,1);
X(end) = C(rows,cols)/C(rows,rows);
for k=rows-1:-1:1
  X(k) = (C(k,cols)-C(k,k+1:cols-1)*X(k+1:rows))/C(k,k);
end
```

7.1.4. cuadminCholesky

cuadminGauss.m

```
function [ X ] = cuadminCholesky (Y)
%inciar las variables
T = Y(:,1);
T = normalizar(T);
A = [T.^0  T.^1  T.^2];
b= log(Y(:,2));
B = A'*A;
c = A'*b;
k = 2;

%descomposicion de cholesky

G = zeros(3);
for(j=1 : 3)
  G(j,j)=sqrt(B(j,j) - sum(G(j,1:j-1).^2));
  for(i=j+1:3)
    G(i,j)=(B(i,j)-G(i,1:j-1)*G(j,1:j-1)')/G(j,j);
  endfor
endfor
%G


%Ya tenemos G, vemos si es S.C.D y calculamos Z
if(G(1,1) ~= 0 && G(2,2) ~= 0 && G(3,3) ~= 0)
  z1 = c(1)/G(1,1);
  z2 = (c(2)-G(2,1)*z1)/G(2,2);
  z3 = (c(3) - G(3,1)*z1 - G(3,2)*z2)/G(3,3);
  Z = [z1,z2,z3];

%Calculamos X
  x3 = Z(3)/G'(3,3);
  x2 = (Z(2)-G'(2,3)*x3)/G'(2,2);
  x1 = (Z(1) - G'(1,2)*x2 - G'(1,3)*x3)/G'(1,1);
  X = [x1,x2,x3]';
else
  error("el sistema no es compatible determinado, no tiene solucion\n")
end
```

7.1.5. cuadminQR

cuadminGauss.m

```
function [ X ] = cuadminQR (Y)
%inciar las variables
T = Y(:,1);
T = normalizar(T);
A = [T.^0  T.^1  T.^2];
b= log(Y(:,2));
k = 3;
A0=A;

[rows,cols]=size(A);
Q=eye(rows);
for i=1:k
x = A(:,i);
S = sqrt(sum(x(i:rows).^2));
y = zeros(rows,1);
y(1:i-1)= x(1:i-1);
y(i) = S;

w= x-y;
norma = sqrt(sum(w.^2));

Qi = eye(rows) - (2/norma^2)*(w*w');
Q = Q*Qi;
A = Qi*A;

end
R=A;
c=Q'*b;
C=[R c];

[rows,cols]=size(C);
X = zeros(3,1);
X(end) = C(3,cols)/C(3,3);
for k=3-1:-1:1
  X(k) = (C(k,cols)-C(k,k+1:cols-1)*X(k+1:3))/C(k,k);
end
```

7.2. Graficos

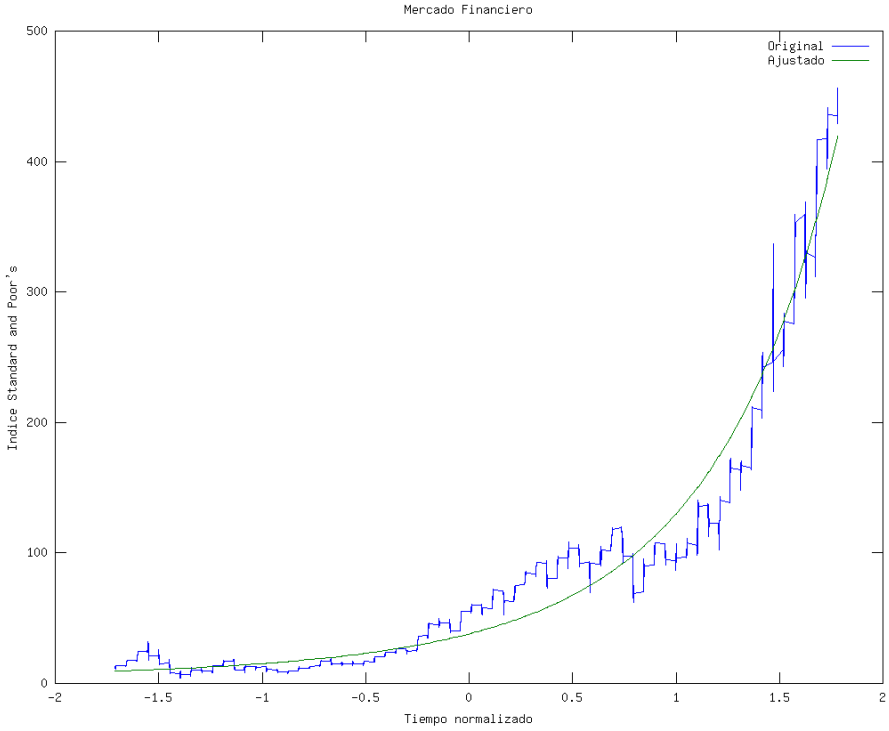

Figura 1: Ajuste por cuadrados mínimos del índice

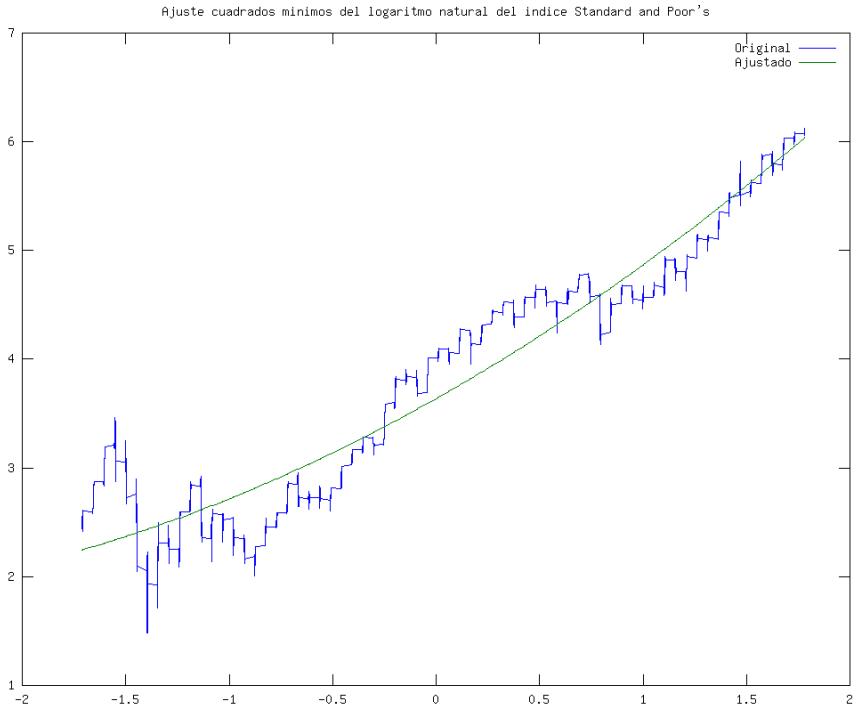

Figura 2: Ajuste por cuadrados mínimos de la función y_2