Kernel Ridge Regression: Interpretación estadística/filtro de Wiener

Este código ejecutó correctamente en Matlab R2018b.

Table of Contents

1.- Generación de los datos

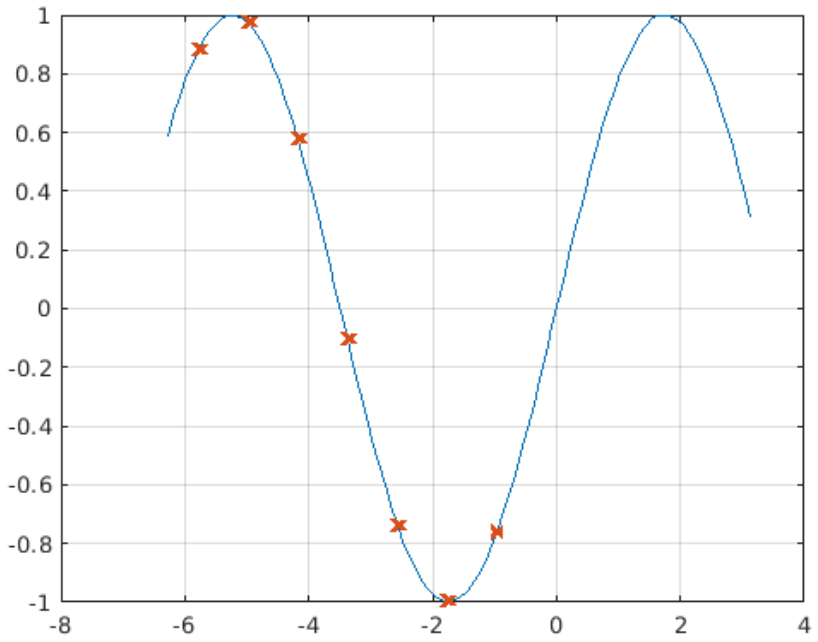
```
Ntodos=95;
x_todos=(0:(Ntodos-1))*3*pi/(Ntodos-1)-2*pi;
y_todos=sin(0.9*x_todos); %
%y_todos=sinc(1.05/pi*x_todos); %inversa exacta de la ventana [0,1.05] en frecuencia.
```

Entrenamos con un subconjunto de los datos, contaminados con un poco de ruido de medida

```
X=x_todos(6:8:(end-36));
Y=y_todos(6:8:(end-36));
N=length(X)
```

```
N =
     7
```

```
Y=Y+randn(1,N)*0.000002;
plot(x_todos,y_todos)
hold on
plot(X,Y,'x','LineWidth',3), grid on, hold off
```



2.- Regresión

Hagamos distintas pruebas de Kernels:

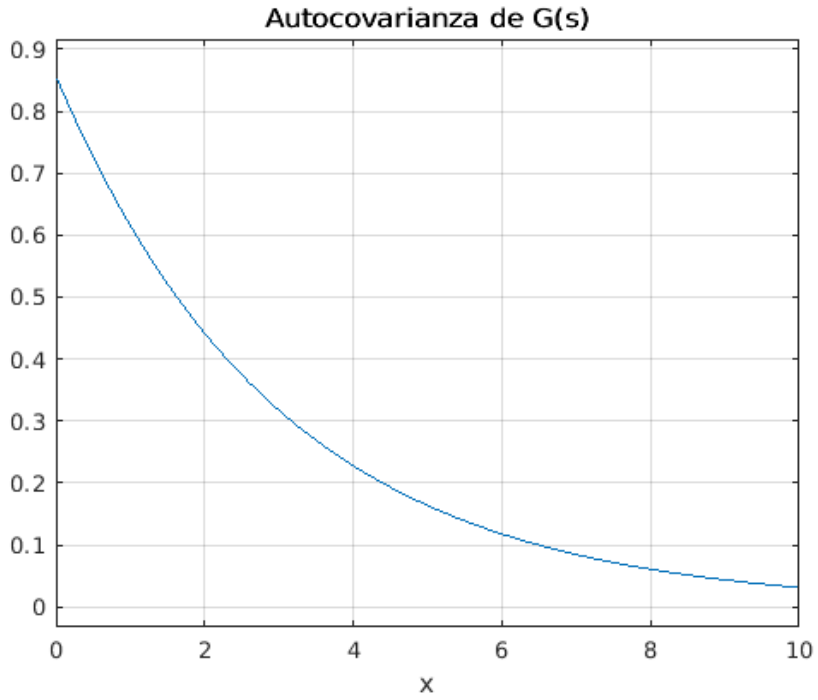
```
prueba=0;
c=.005^2; %parámetro de regularización/ruido medida
switch prueba
    case 0
        %kernel exponencial, filtro primer orden
%         polo=-.75;
%         kappa=@(x,y) exp(norm(x-y)*polo);
        syms w real
        G=@(s) .75/(s+.33);%primer orden, Kernel exponecial.
        %bw=1.05;
        %G=@(s) 1.5*bw^2/(s^2+sqrt(2)*bw*s+bw^2);%Butterworth
        PowerSpectralDensity=simplify(G(1i*w)*G(-1i*w))
        AutoCovarianza=ifourier(PowerSpectralDensity)
        ezplot(AutoCovarianza,[0,10]), grid on, title('Autocovarianza de G(s)')
        cosa=matlabFunction(AutoCovarianza);
        desvtipprocesogaussiano=sqrt(cosa(0)) %desvtip estacionaria
        norma2deG=norm(G(tf('s')))
        kappa=@(x,y) cosa(norm(x-y));
    case 1
        % Kernel que recupera la fórmula famosa de reconstrucción
        % muestreada de una serie de datos con frecuencia finita "bw".
        % Shannon interpolation, con añadido de intervalos de confianza
        bw=1.05;
        kappa=@(x,y) sinc(norm(x-y)*bw/pi);
end
```

```
PowerSpectralDensity =
```

$$\frac{5625}{10000\,w^2 + 1089}$$

```
AutoCovarianza =
```

$$\frac{75\,e^{-\frac{33\,|x|}{100}}}{88}$$




```
desvtipprocesogaussiano =
   0.923186182344995
norma2deG =
   0.923186182344996
```

```
K=generaKernel(X,kappa)
```

```
K = 7×7
   0.852272727272727   0.654067584528809   0.501957168687451   0.385221657756110 ···
   0.654067584528809   0.852272727272727   0.654067584528809   0.501957168687451
   0.501957168687451   0.654067584528809   0.852272727272727   0.654067584528809
   0.385221657756110   0.501957168687451   0.654067584528809   0.852272727272727
   0.295634239057488   0.385221657756110   0.501957168687451   0.654067584528809
   0.226881333236030   0.295634239057488   0.385221657756110   0.501957168687451
   0.174117651375789   0.226881333236030   0.295634239057488   0.385221657756110
```

```
% comprobar K>=0 para que se pueda interpretar como matriz
% varianzas-covarianzas
min(eig(K))
```

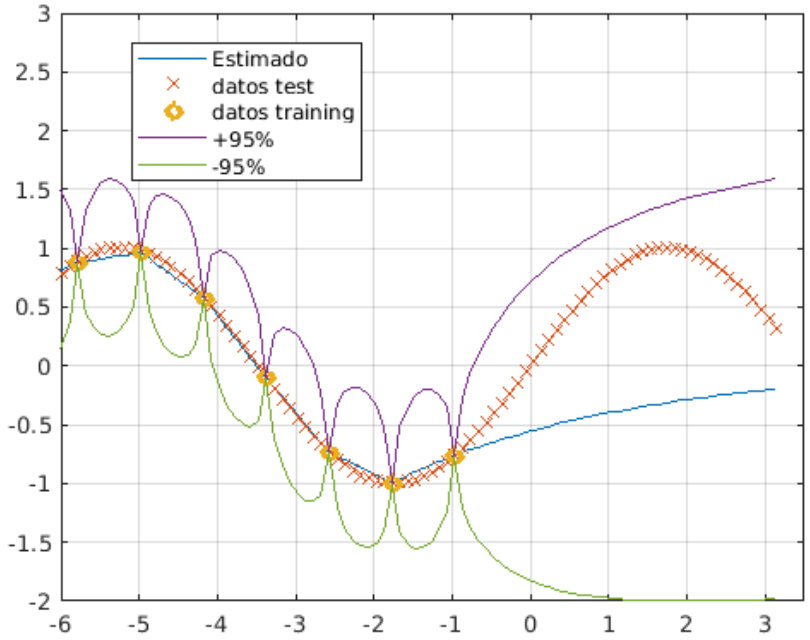
```
ans =
   0.117658448497374
```

```
w=Y/(K+c*eye(N)) %solución óptima ridge regression
```

```
w = 1×7
   0.384266277883144   1.214228662846576   0.722970082278094  -0.128990100276995 ···
```

3.- Comprobación del ajuste resultante

```
Ntodo=length(x_todos);
lista=zeros(1,Ntodo);
listadt=zeros(1,Ntodo);
for i=1:Ntodo
     lista(i)=modeloestimado(x_todos(i),X,kappa,w);
     listadt(i)=sqrt(varianzaestimada(x_todos(i),X,kappa,c));
end
plot(x_todos,lista)
hold on
plot(x_todos,y_todos,'x')
plot(X,Y,'o','LineWidth',3)
plot(x_todos,lista+listadt*2)
plot(x_todos,lista-listadt*2)
hold off, grid on, axis([-6 3.5 -2 3])
legend('Estimado','datos test','datos training','+95%','-95%','Location','best')
```




```
function K=generaKernel(X,kappa)
N=size(X,2);
K=zeros(N,N);
for i=1:N
    K(i,i)=kappa(X(i),X(i));
    for j=(i+1):N
        K(i,j)=kappa(X(i),X(j));
        K(j,i)=K(i,j);
    end
end
end
```

```
function K=KappaXY(X,Y,kappa)
Nx=size(X,2);
Ny=size(Y,2);
K=zeros(Nx,Ny);
for i=1:Nx
    for j=1:Ny
        K(i,j)=kappa(X(:,i),Y(:,j));
    end
end
end

function yestimada=modeloestimado(x,X,kappa,w)
yestimada=0;
N=size(X,2);
 for i=1:N
     yestimada=yestimada+w(i)*kappa(X(i),x);
 end
end

function V=varianzaestimada(x,X,kappa,c)
    Cov=KappaXY(X,x,kappa);
    K=KappaXY(X,X,kappa);
    V=kappa(x,x)-Cov'*inv(K+c*eye(size(K,2)))*Cov;
end
```