Optimización bayesiana (OB), introducción: ejemplo completo sencillo

Objetivo: Programar y comprender el funcionamiento de la metodología completa de optimización bayesiana: adquirir datos, actualizar posterior, decidir muestra y repetir.

Presentación en vídeo: https://personales.upv.es/asala/YT/V/boloop1.html

Tabla de Contenidos

Motivación, discusión preliminar

Objetivo de BO: obtener el ***mínimo*** global de $f(x)$ a través de un "modelo estadístico" del mismo (un **proceso gaussiano**, usualmente).

Metodología:

1. Establecer un modelo de GP, el "prior". $k=0$ (o número de muestras iniciales disponibles)
2. Decide cuál es la mejor "siguiente x", x_k para probar.
3. Adquirir muestras (generalmente una por una, proceso costoso), pares $(x_k,\ y_k = f(x_k)+\epsilon_k)$.
4. Actualizar el modelo GP (obtener el "posterior")
5. Evaluar los criterios de terminación e ir al paso 2 si no se cumplen.

Paso 2, detalle: asociamos a cada x una "función de adquisición" $a(x)$, rápida de evaluar, relacionada con lo "bueno" que se supone que es el punto de muestreo candidato, basado en determinadas propiedades estadísticas. El "mejor" valor de $a(x)$ es el que se propone probar en el paso 2.

En este ejemplo usaremos el VALOR INFERIOR DEL INTERVALO DE CONFIANZA (2.5%); otras opciones más complejas se verán en vídeos posteriores.

Inicialización del Proceso gaussiano ejemplo

Función "real" a optimizar (desconocida):

```
TrueF=@(x) 0.9*sin(1.8*x+1.35)+.5*x.^2-.6+0.2.*x; %perfecto
MaxSamples=12;
Data.priormean=@(x) zeros(size(x));
```

Prior: función media y kernel, intervalo de confianza

```
Data.X=[]; Data.Y=[]; %To test "prior" model, empty data
Data.K=@K;
STDMeasureNoise=4e-2;
VarMeasureNoise=STDMeasureNoise^2;
Xtest=-2:(1/50):2; %uniform grid to test stuff
TrueData=TrueF(Xtest);
[TrueOptimum,TrueOptIdx]=min(TrueData);
TrueXOpt=Xtest(TrueOptIdx);
LL=length(Xtest);
[MeanPrior,CovMatrixPrior]=predictGP(Xtest,Data,VarMeasureNoise);
sg=sqrt(diag(CovMatrixPrior));
PredPrior=[MeanPrior-1.96*sg MeanPrior MeanPrior+1.96*sg];
```

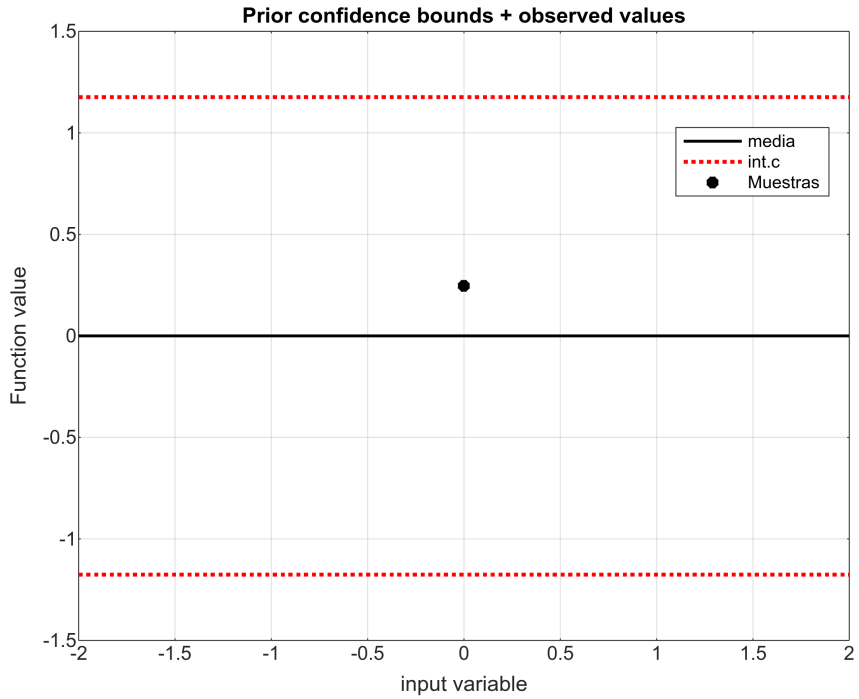
Cargado de datos experimentales de observaciones "pasadas"

```
Data.X=0;
Data.Y=TrueF(Data.X)+randn(size(Data.X))*STDMeasureNoise;Data.Y
```

```
ans = 0.2460
```

```
plot(Xtest,PredPrior(:,2),'k',LineWidth=1.5), grid on, hold on
plot(Xtest,PredPrior(:,[1 3]),'r:',LineWidth=2)
plot(Data.X,Data.Y,'*k',LineWidth=3),
%plot(Xtest,TrueData,'-.',LineWidth=3,Color=[.4 .5 .2])
hold off
xlabel("input variable"), ylabel("Function value")
title("Prior confidence bounds + observed values")
legend("media","int.c","","Muestras","F verdadera",Location="best")
```

```
Warning: Ignoring extra legend entries.
```



Bucle de optimización bayesiana

```
ybestDataPLOT=[];
while length(Data.X)<MaxSamples
    [post_mean,post_var]=predictGP(Xtest,Data,VarMeasureNoise);
    [ybestData,idxbest]=min(Data.Y);
    ybest=ybestData
    ybestDataPLOT=[ybestDataPLOT ybest];
```

Nota: minimizar buscando un punto de un "mallado muy fino" obviamente NO es la mejor forma de hacerlo, porque no escala a varias dimensiones de la variable de entrada y dependemos de la "granularidad" del mallado. Debería usarse software de optimización (fmincon o similar), pero aquí no lo hacemos por claridad didáctica en la presentación.

```
    sg=sqrt(diag(post_var));
    Pred=[post_mean-1.96*sg post_mean post_mean+1.96*sg];
    [min_lcb,lcb_idx]=min(post_mean-1.96*sg);

    AFname='LCB';%ESTA es la función de adquisición elegida
    NewSample=Xtest(lcb_idx) %ESTA es la función de adquisición
elegida
    NewMeasure=TrueF(NewSample)+randn()*STDMeasureNoise
```

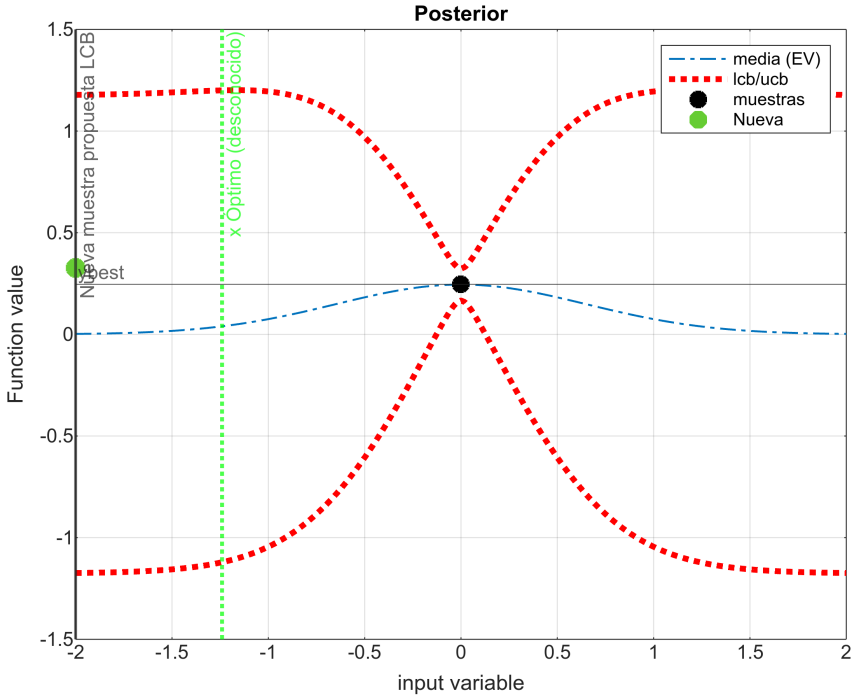
Dibujemos... (esto es cosmético)

```
    figure()
    plot(Xtest,Pred(:,2),"-.",LineWidth=1)
    hold on
```

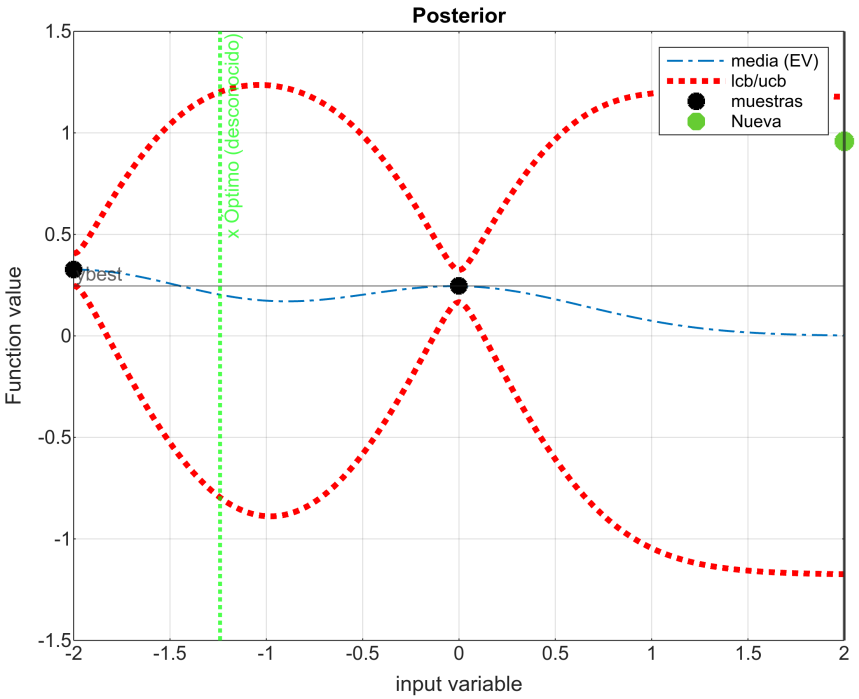
```
    plot(Xtest,Pred(:,[1 3],:),'r:',LineWidth=3)
    plot(Data.X,Data.Y,'*k',LineWidth=3,MarkerSize=9);
    %plot(Xtest,TrueData,'-.',LineWidth=2,Color=[.4 .5 0.2])
    plot(NewSample,NewMeasure,'*',Color=[.4 .8
.2],LineWidth=3.5,MarkerSize=10);
    hold off
    grid on, xlabel("input variable"), ylabel("Function value")
    title("Posterior")
    yline(ybest,label="ybest",LabelHorizontalAlignment="left")
    xline(NewSample,LineWidth=1.5,label=['Nueva muestra propuesta '
AFname])
    xline(TrueXOpt,':g',LineWidth=2,label="x Óptimo (desconocido)")
    %legend("media (EV)","lcb/ucb","","muestras","F real
(desconocida)","Nueva")
    legend("media (EV)","lcb/ucb","","muestras","Nueva")
    Data.X=[Data.X NewSample]; %for next iter
    Data.Y=[Data.Y NewMeasure]; %for next iter

end %end of Bayesian Optimization loopmedia
```

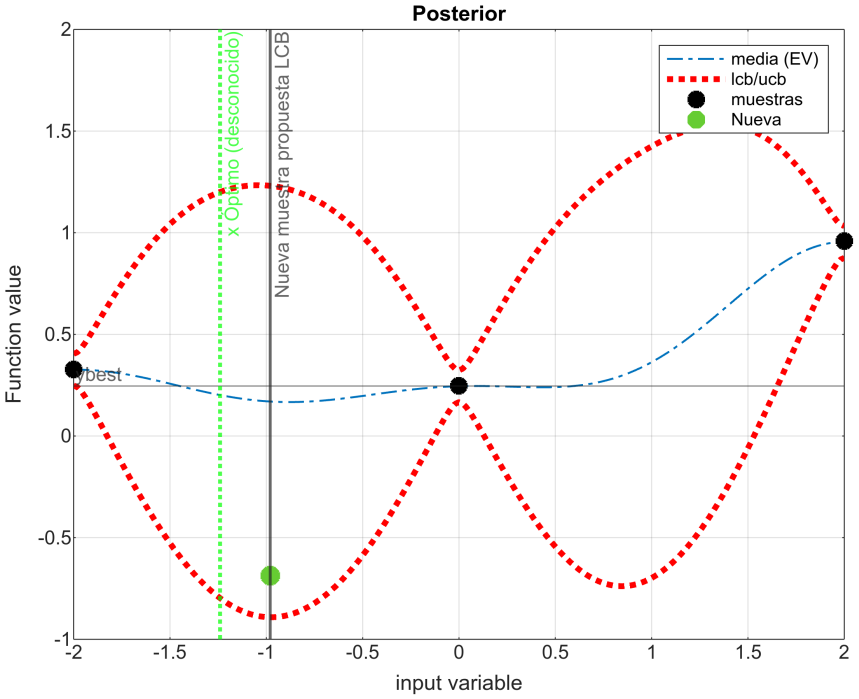
```
ybest = 0.2460
NewSample = -2
NewMeasure = 0.3276
```



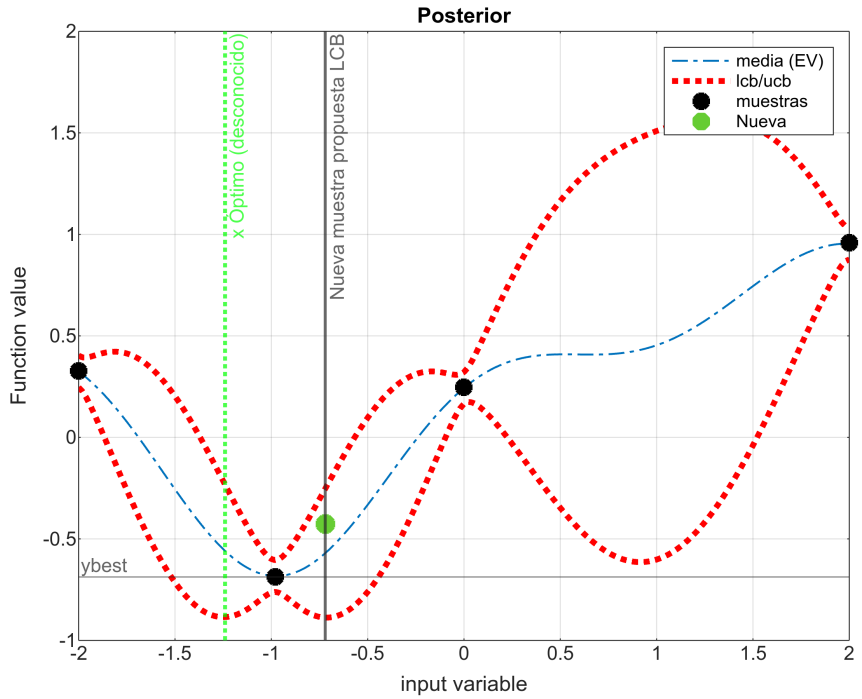

```
ybest = 0.2460
NewSample = 2
NewMeasure = 0.9587
```



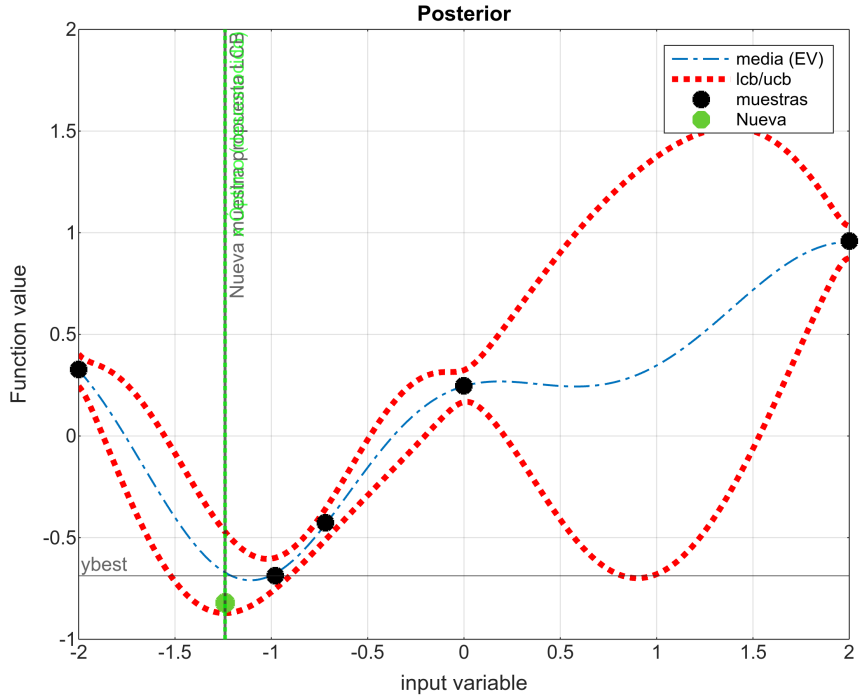

```
ybest = 0.2460
NewSample = -0.9800
NewMeasure = -0.6876
```



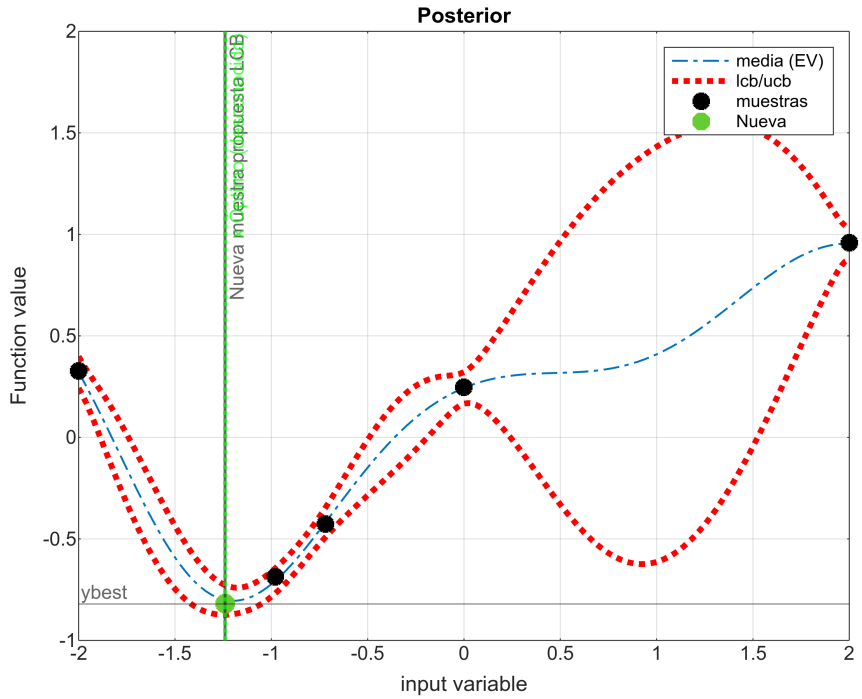

```
ybest = -0.6876
NewSample = -0.7200
NewMeasure = -0.4276
```



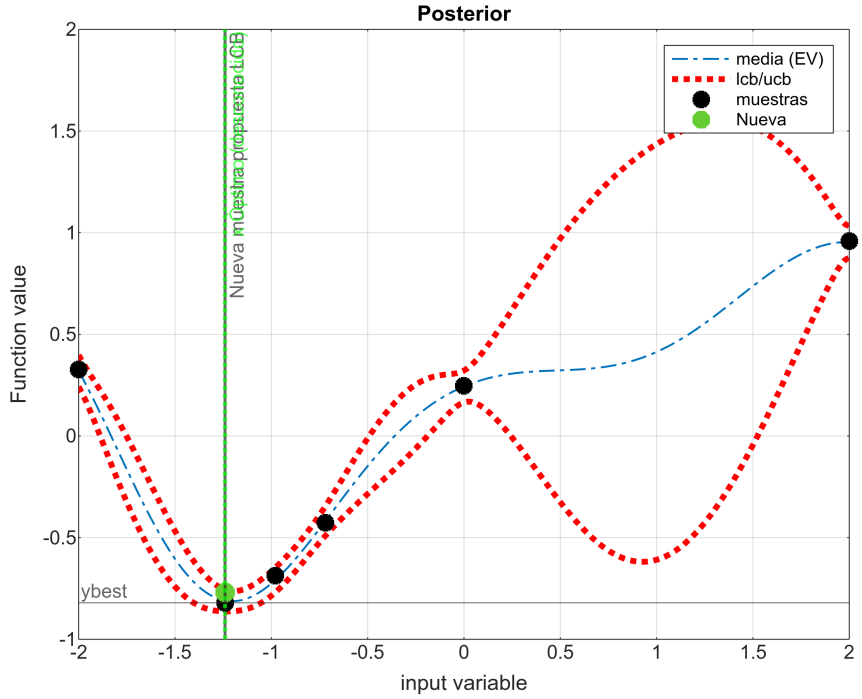

```
ybest = -0.6876
NewSample = -1.2400
NewMeasure = -0.8206
```



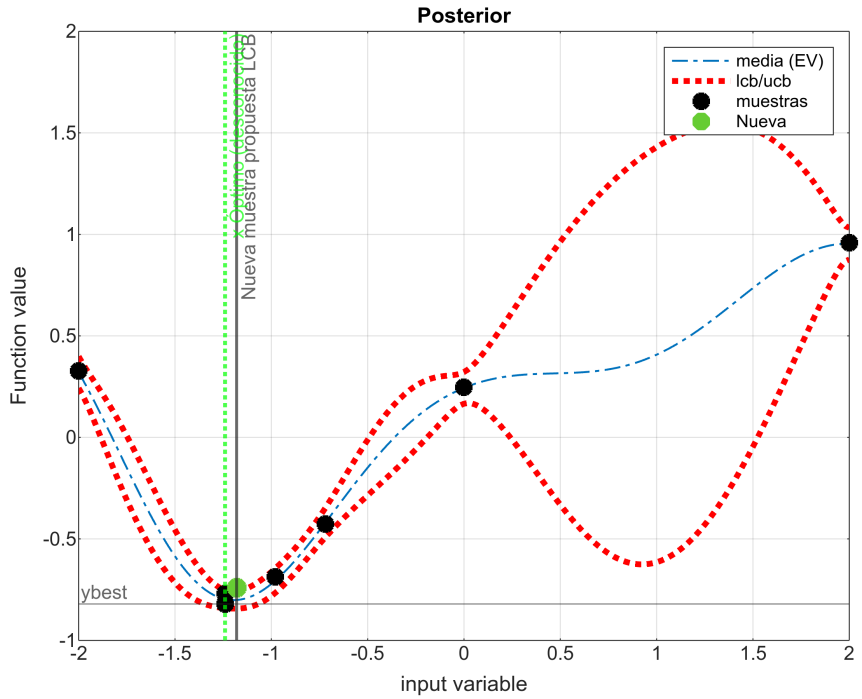

```
ybest = -0.8206
NewSample = -1.2400
NewMeasure = -0.8199
```



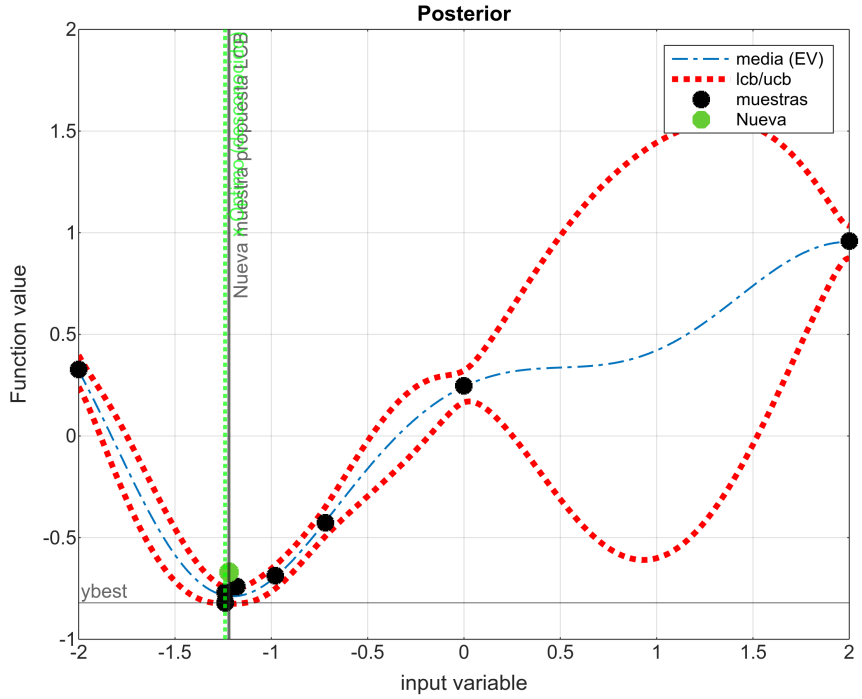

```
ybest = -0.8206
NewSample = -1.2400
NewMeasure = -0.7698
```



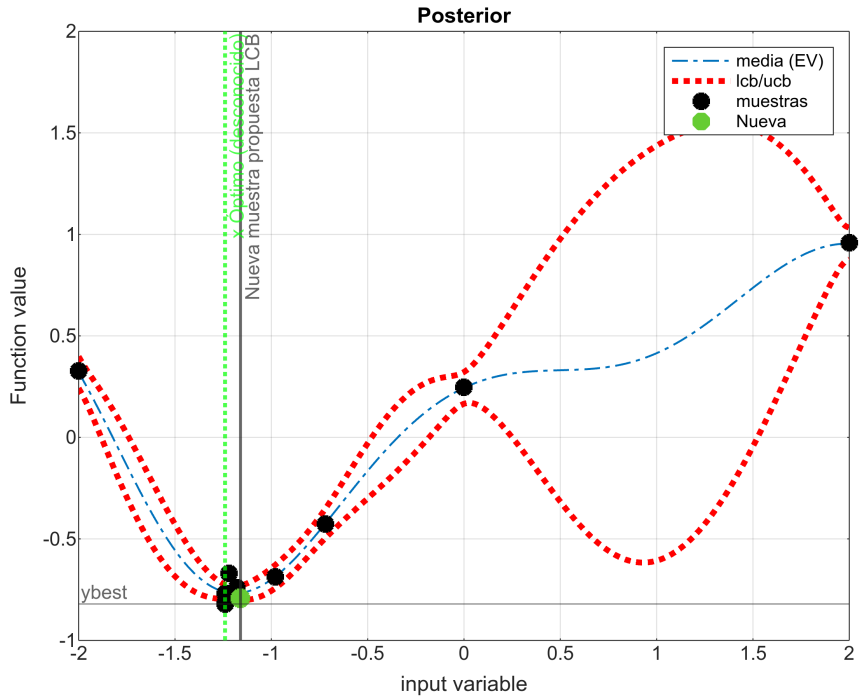

```
ybest = -0.8206
NewSample = -1.1800
NewMeasure = -0.7400
```




```
ybest = -0.8206
NewSample = -1.2200
NewMeasure = -0.6702
```

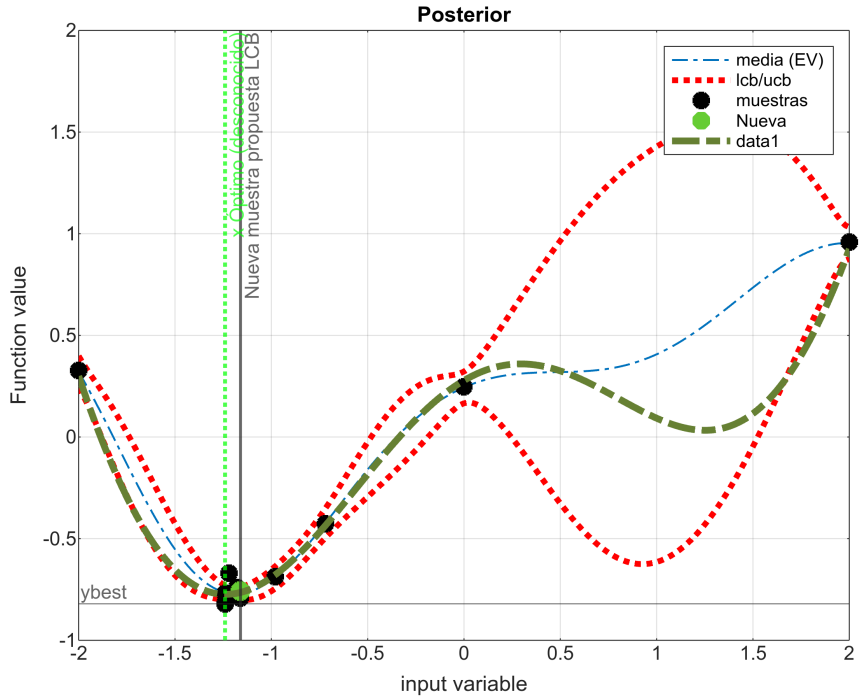



```
ybest = -0.8206
NewSample = -1.1600
NewMeasure = -0.7914
```



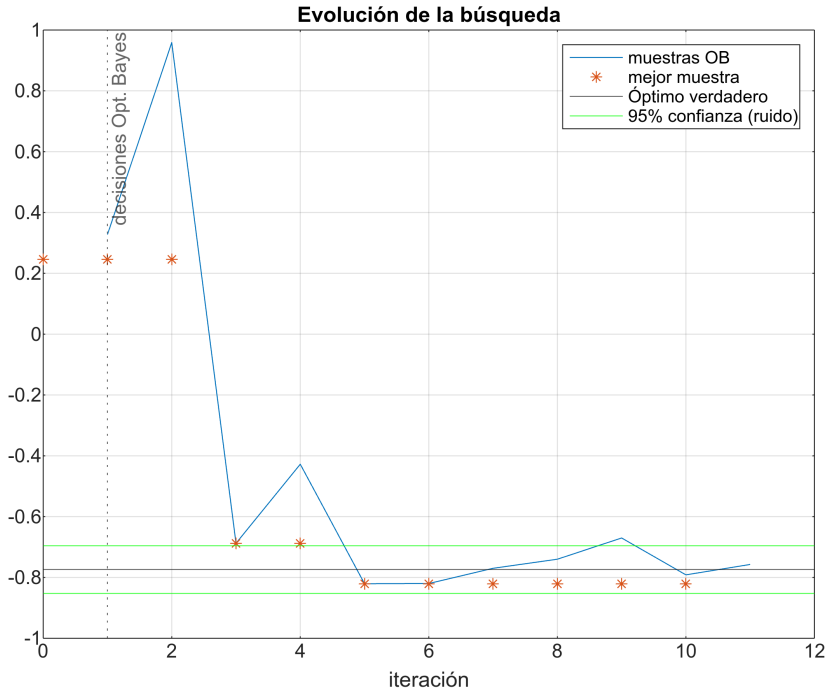

```
ybest = -0.8206
NewSample = -1.1600
NewMeasure = -0.7572
```

```
hold on
plot(Xtest,TrueData,'-.',LineWidth=3.5,Color=[.4 .5 0.2])
hold off
```



Vamos ahora a ver el progreso de la búsqueda en una nueva gráfica.

```
plot(Data.Y(2:end)), hold on
plot(0:(MaxSamples-2), ybestDataPLOT,'*'),
hold off
yline(TrueOptimum)
yline(TrueOptimum+1.96*STDMeasureNoise,'g')
yline(TrueOptimum-1.96*STDMeasureNoise,'g'),
xline(1,':',label="decisiones Opt. Bayes")
title("Evolución de la búsqueda"), grid on, xlabel("iteración")
legend("muestras OB","mejor muestra","Óptimo verdadero","95%
confianza (ruido)")
```




```
function [mu,Var]=predictGP(x1,Data,VarMeasureNoise)
priormean=Data.priormean;
K=@Data.K;
N=length(Data.X);
mu=priormean(x1)';
Var=K(x1,x1);
if(N>0)
    K1X=K(x1,Data.X);
    Gain=K1X/(K(Data.X,Data.X)+VarMeasureNoise*eye(N));
    mu=mu+Gain*(Data.Y-priormean(Data.X))';
    Var=Var-Gain*K1X';
    Var=0.5*(Var+Var'); %roundoff-errors, correct them
end
end

function km=K(x1,x2)
M=0.6; sg=0.65;%hyperparameters stddev x and y
```

```
kernel=@(x1,x2)   M^2*exp(-norm(x2-x1)^2/2/sg^2); %covariance
kernel function
N1=length(x1); N2=length(x2); %1D case
km=zeros(N1,N2);
for i=1:N1
    for j=1:N2
        km(i,j)=kernel(x1(:,i),x2(:,j)); %do it for all pairs
(x1_i,x2_j)
    end
end
end
```