Bayesian Optimization (motivation): probabilistic information about the optimal point implicitly available in a Gaussian process

Objectives: assess the implicitly available information on where (abscissa) is the global minimum of a GP (posterior to some data), and which is the value (ordinate) it attains.

Presentation in video:

https://personales.upv.es/asala/YT/V/boinopEN.html

Table of Contents

Gaussian Process setup

Prior mean and kernel, confidence interval

```
Data.X=[]; Data.Y=[]; %To test "prior" model, empty data
Data.K=@K;
Xtest=-2:(1/50):2; %uniform grid to test stuff
LL=length(Xtest)
```

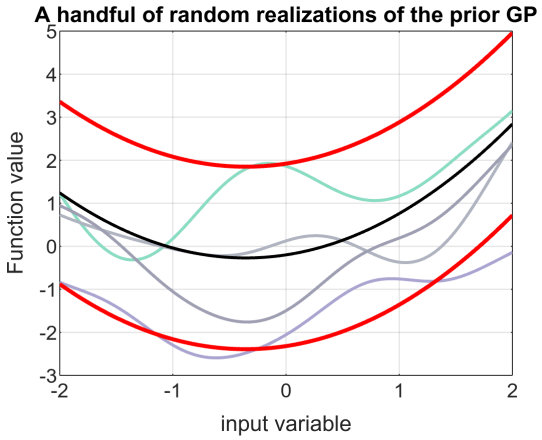
```
LL = 201
```

```
[Mean,CovMatrix]=predictGP(Xtest,Data);
size(CovMatrix)
```

```
ans = 1×2
   201   201
```

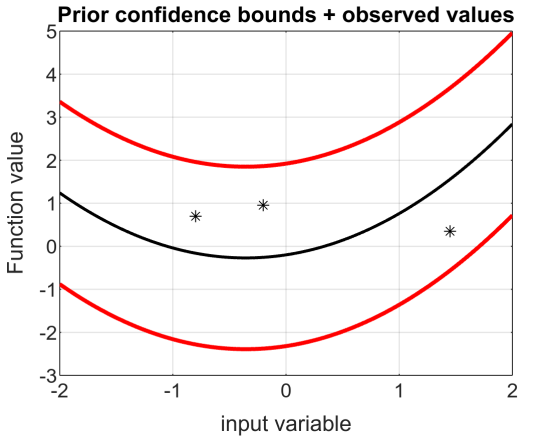
```
sg=sqrt(diag(CovMatrix)); %standard deviation at each test point
PredPrior=[Mean-1.96*sg Mean Mean+1.96*sg]; %to plot conf. intervals
NTrials=4;
FunctionRealization=mvnrnd(Mean,CovMatrix,NTrials);
for i=1:NTrials
    if(NTrials<5)
        LW=1.5; %Line width, cosmetic
    else
        LW=1; %Line width, cosmetic
    end
```

```
    col=[.5 .6 .65]+rand()*[.3 0 0]+rand()*[0 .3 0]+rand()*[0 0
0.2]; %color, cosmetic
    plot(Xtest,FunctionRealization(i,:),Color=col,LineWidth=LW),
hold on
end
plot(Xtest,PredPrior(:,2),'k',LineWidth=1.5), grid on
plot(Xtest,PredPrior(:,[1 3]),'r',LineWidth=2), hold off
title("A handful of random realizations of the prior GP")
xlabel("input variable"), ylabel("Function value")
```



Load with experimental data from past observations

```
Data.X=[-0.8 -.2 1.45];
Data.Y=[0.7 0.95 0.35];
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'), hold off, xlabel("input variable"),
ylabel("Function value")
title("Prior confidence bounds + observed values")
```

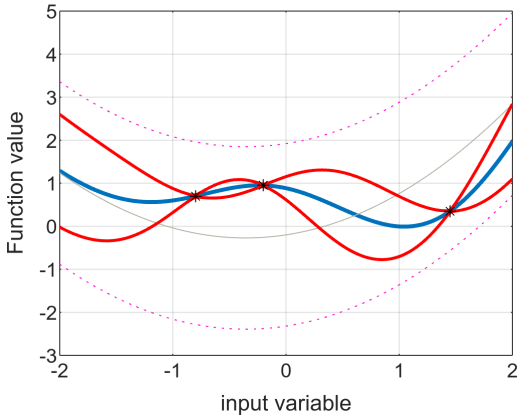


Hyperparameter optimization?: all samples so close/outside confidence bounds... maybe we might wish to fiddle with the covariance kernel parameters. Well, just remember it for later refinements but forget it for the moment being.

Computing and sampling the posterior

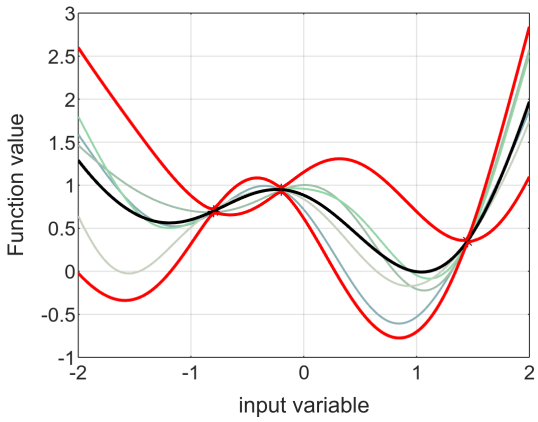
Let's compute the posterior given data:

```
[mm,sss]=predictGP(Xtest,Data);
sg=sqrt(diag(sss)); %standard deviation at each test point
Pred=[mm-1.96*sg mm mm+1.96*sg];
plot(Xtest,Pred(:,2),LineWidth=2)
hold on
plot(Xtest,PredPrior(:,2),Color=[.7 .7 .65])
plot(Xtest,Pred(:,[1 3],:),'r',LineWidth=1.5)
plot(Xtest,PredPrior(:,[1 3]),':',Color=[1 .0 .8])
plot(Data.X,Data.Y,'*k');
hold off, grid on, xlabel("input variable"), ylabel("Function
value")
```



Sample the posterior

```
figure()
NTrials=4;
FunctionRealization=mvnrnd(mm,sss,NTrials);%clean
for i=1:NTrials
    if(NTrials<4)
        LW=1.5;
    else
        LW=1;
    end
    col=[.5 .6 .65]+rand()*[.3 0 0]+rand()*[0 .3 0]+rand()*[0 0
0.2];
    plot(Xtest,FunctionRealization(i,:),Color=col,LineWidth=LW),
hold on
end
hold on
plot(Data.X,Data.Y,'*k');
plot(Xtest,Pred(:,2),'k',LineWidth=1.5)
plot(Xtest,Pred(:,[1 3]),'r',LineWidth=1.5)
hold off, grid on, xlabel("input variable"), ylabel("Function
value")
```



Information for Bayesian optimization: distribution of optimal points and optimal values

Experimental tests are "costly". We can do whatever we wish "in silico" (simulation, see below) but obtaining one "actual" sample takes time in laboratory tests, man hours, raw materials, etc. So we wish our "statistical model" on how the function behaves to guide us towards obtaining something close to the actual optimum value with a reduced number of experimental samples.

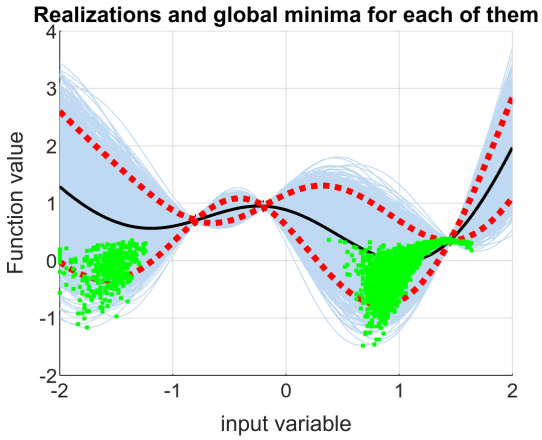
Say, we wish to carry out next sample at a point which "is highly likely to be optimal", or at a point that is "highly informative on the location of the optimal value", or variations of that, and depending on the available sample budget we might be more "risky" (exploratory).

Let us throw the dice with the GP and find where the global minimum is each of the times:

```
figure()
hold on
plot(Data.X,Data.Y,'*k');
NTrials=4000;
FunctionRealizations=mvnrnd(mm,sss,NTrials);
size(FunctionRealizations)
```

```
ans = 1×2
        4000         201
```

```
[fval,xmn]=min(FunctionRealizations,[],2); %min of each trial,
i.e., min of each ROW.
plot(Xtest,FunctionRealizations,Color=[.75 .85 .95])
plot(Xtest,Pred(:,2),'k',LineWidth=1.5)
plot(Xtest,Pred(:,[1 3]),':r',LineWidth=3)
plot(Xtest(xmn),fval,'.g')
hold off, grid on
title("Realizations and global minima for each of them")
xlabel("input variable"), ylabel("Function value")
```



Approximate probability plots

```
plot(Xtest,Pred(:,2),'k',LineWidth=1.5)
hold on
plot(Xtest,Pred(:,[1 3]),':r',LineWidth=3)
plot(Xtest(xmn),fval,'.g')
grid on
title("global minima for each of them + marginals")
xlabel("input variable"), ylabel("Function value")
```

Compute **input** histogram

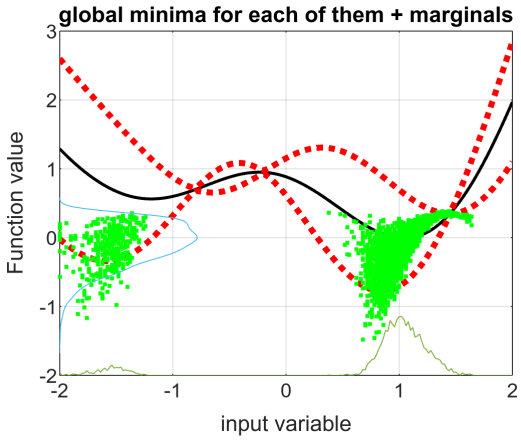
```
hist1p=zeros(LL,1);
for k=1:LL
    hist1p(k)=length(find(xmn==k));
end
Pmin=hist1p/sum(hist1p);
plot(Xtest,Pmin*20-2)
```

Compute **output** histogram:

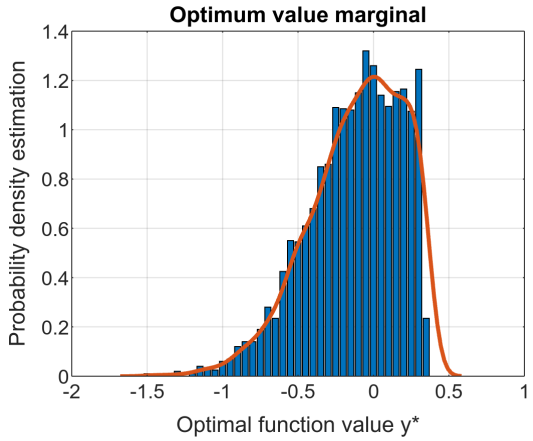
```
[Ncounts,edges]=histcounts(fval,Normalization="pdf");
```

Smoothed histogram via "kernel density estimation" (it has nothing to do with covariance kernel of the GP):

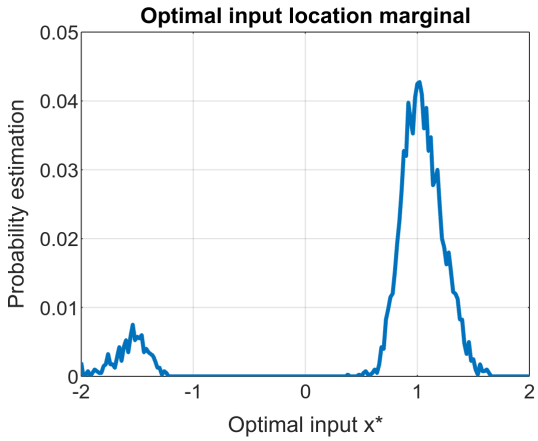
```
[pro,ff]=ksdensity(fval);
plot(pro-2,ff);
grid on, hold off
```




```
bar(edges(1:(end-1)),Ncounts), hold on
plot(ff,pro,LineWidth=2), xlabel("Optimal function value
y*"),ylabel("Probability density estimation"), grid on,
title("Optimum value marginal")
hold off
```



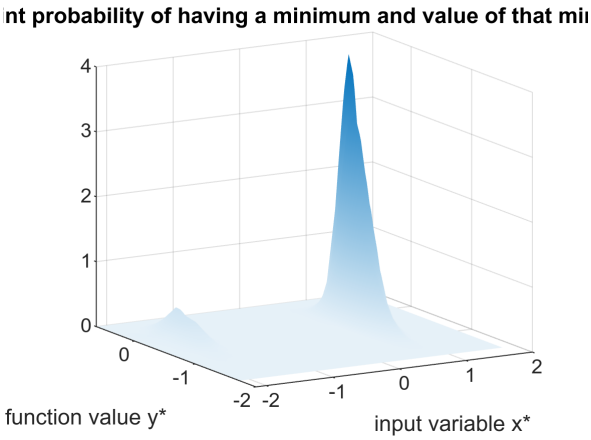

```
plot(Xtest,Pmin,LineWidth=2), xlabel('Optimal input x*'),
ylabel("Probability estimation")
hold off, grid on, title("Optimal input location marginal")
```



*Probability "density" estimation would require dividing by grid separation interval, just proportional to the plot we see.

Joint plot

```
ksdensity([Xtest(xmn)'  fval]);
shading interp
view([-30 15])
colormap sky
xlabel("input variable x*"), ylabel("function value y*")
title("Joint probability of having a minimum and value of that
minimum")
```



Appendix (utility, auxiliary functions)

```
function [mu,Var]=predictGP(x1,Data)
arguments
    x1
    Data
end
priormean=@(x) 0.56*x.^2+0.4*x-.2;
lambda=1.5e-2^2; %measurement noise variance
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)+lambda*eye(N));
    mu=mu+Gain*(Data.Y-priormean(Data.X))';
    Var=Var-Gain*K1X';
    Var=0.5*(Var+Var')+1e-15*eye(size(Var,1)); %round-off and
numerical issues just in case
end
end

function km=K(x1,x2)
M=0.6; sg=0.65;%hyperparameters: correlation distance sg (x) and
std dev (y)
kernel=@(x1,x2)   M^2*exp(-norm(x2-x1)^2/2/sg^2)+.9^2;
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)); %we form the matrix
    end
end
end
```