Surrogate Optimization: Sequential Design

BIOE 498/598 PJ

Spring 2021

Optimizing a surrogate model

- Nonlinear optimization requires repeatedly evaluating an *objective function*.
- If the gradient is available, the solvers use it to find good descent directions.
- If the gradient is unavailable, solvers may approximate it using additional objective evaluations.
- Some "gradient-free" algorithms use search methods for objectives with discontinuous or undefined gradients.

Optimizing a surrogate model

- Nonlinear optimization requires repeatedly evaluating an *objective function*.
- If the gradient is available, the solvers use it to find good descent directions.
- If the gradient is unavailable, solvers may approximate it using additional objective evaluations.
- Some "gradient-free" algorithms use search methods for objectives with discontinuous or undefined gradients.

- The R function `optim` implements many nonlinear optimization algorithms.
- For surrogate optimization, we use L-BGFS-B, an efficient quasi-Newton method.
- L-BGFS-B allows *box constraints* to limit the solution space.

Objective functions for optimization

By default, `optim` *minimize* functions, so we negate our objective to find *maxima*.

$$\max_x f(x) \Leftrightarrow \min_x -f(x)$$

```
library(laGP)

total_obj_evals <- 0

obj_mean <- function(x,gp) {
  total_obj_evals <<- total_obj_evals + 1
  -predGP(gp, matrix(x,nrow=1), lite=TRUE)$mean
}
```

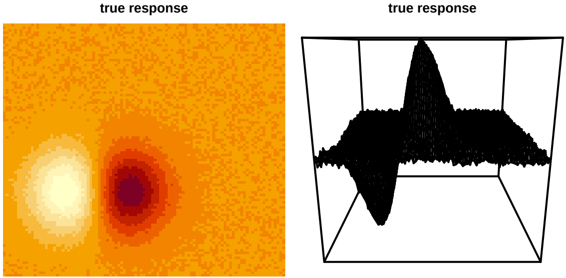
The `lite=TRUE` tells `laGP` to not compute the entire covariance matrix, just the `mean` and variance `s2`.

Our search function

```r
gp_search <- function(obj, gp, nstarts, Xn) {
  Xstart <- maximin::maximin(nstarts, 2, Xorig=Xn)$Xf[nrow(Xn)+(1:nstarts), ]
  X <- matrix(NA, nrow=nstarts, ncol=2)
  y <- numeric(nstarts)
  for (i in 1:nstarts) {
    out <- optim(Xstart[i, ], obj,
                 method="L-BFGS-B", lower=0, upper=1,
                 gp=gp)
    X[i, ] <- out$par
    y[i] <- out$value
  }
  return(list(Xstart=Xstart, X=X, y=y))
}
```

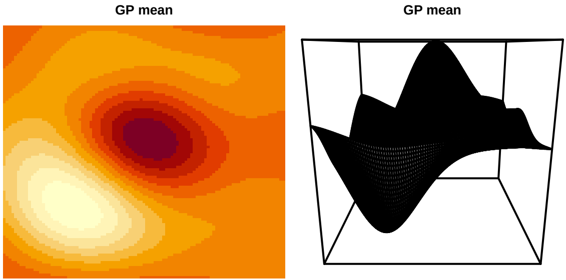
The *true* function f

```
f <- function(X, sd=0.01) {
  X[,1] <- (X[,1] - 0.5)*6 + 1
  X[,2] <- (X[,2] - 0.5)*6 + 1
  X[,1] * exp(-X[,1]^2 - X[,2]^2) + rnorm(nrow(X), sd=sd)
}
plot_f2(f)
```



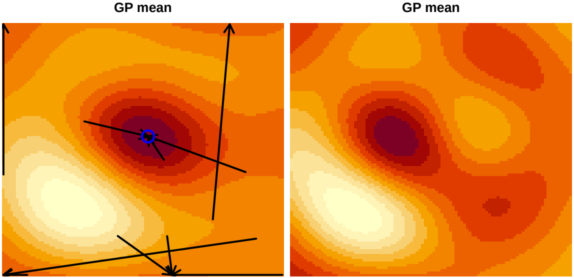
An initial design

```
Xn <- maximin::maximin(n=16, p=2, T=100)$Xf
yn <- f(Xn)
gp <- laGP::newGP(Xn,yn,d=0.1,g=0.1*var(yn),dK=TRUE)
plot_gp2(gp)
```

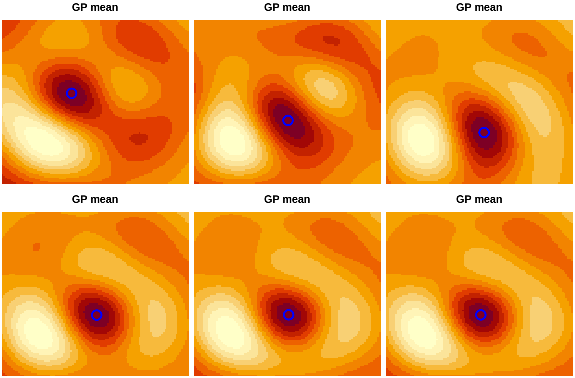


Searching the surrogate for a maximum

```r
result <- gp_search(obj_mean, gp, 10, Xn)
argmax <- which.max(-result$y)
Xnew <- matrix(result$X[argmax, ], ncol=2)

plot_result(gp,result)
points(Xnew[ ,1], Xnew[ ,2], col="blue")

updateGP(gp, Xnew, f(Xnew))
Xn <- rbind(Xn, Xnew)
plot_gp2(gp, type="image")
```



Iterative design by surrogate optimization



How many functional evaluations did it take?

True function evaluations: $16 + 7 = 23$.

How many functional evaluations did it take?

True function evaluations: $16 + 7 = 23$.

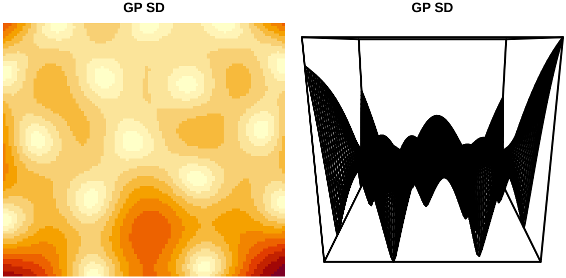
Surrogate function evaluations:

```
total_obj_evals
```

```
## [1] 3235
```

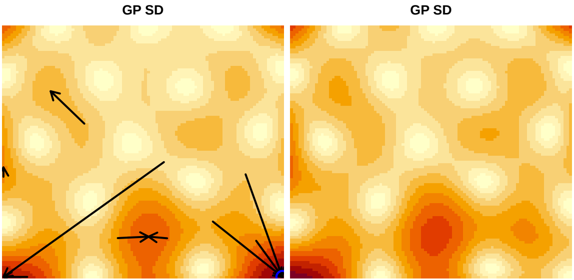
What about uncertainty?

```
Xn <- maximin::maximin(n=16, p=2, T=100)$Xf
yn <- f(Xn)
gp <- laGP::newGP(Xn,yn,d=0.1,g=0.1*var(yn),dK=TRUE)
plot_gp2(gp,"sd")
```

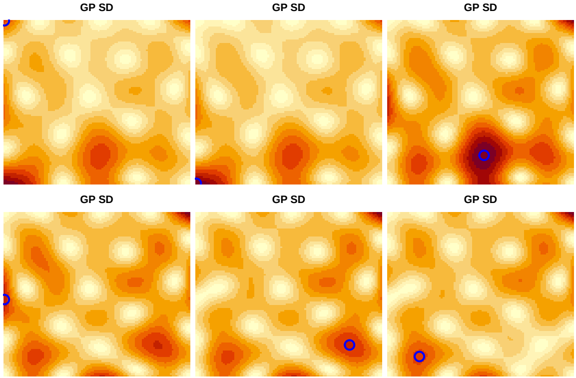


Searching for locations of maximum uncertainty

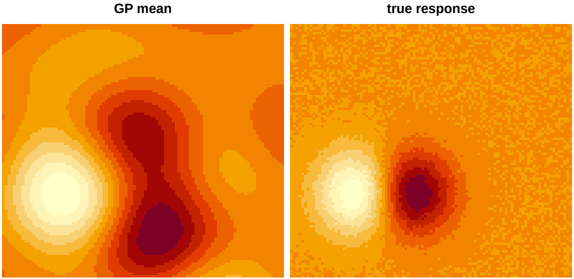
```
obj_sd <- function(x,gp) {
  -sqrt(predGP(gp, matrix(x,nrow=1), lite=TRUE)$s2)
}

result <- gp_search(obj_sd, gp, 10, Xn)
argmax <- which.max(-result$y)
Xnew <- matrix(result$X[argmax, ], ncol=2)

plot_result(gp,result,"sd")
points(Xnew[ ,1], Xnew[ ,2], col="blue")

updateGP(gp, Xnew, f(Xnew))
Xn <- rbind(Xn, Xnew)
plot_gp2(gp,"sd",type="image")
```



Iterative model improvement by surrogate optimization



The response surface after minimizing uncertainty



Exploration vs. exploitation

There is a fundamental tradeoff in global optimization:

- **Exploration** searches areas of high uncertainty to find *new* regions of interest.
- **Exploitation** refines existing optima by adding points to *known* regions of interest.

Exploration vs. exploitation

There is a fundamental tradeoff in global optimization:

- **Exploration** searches areas of high uncertainty to find *new* regions of interest.
- **Exploitation** refines existing optima by adding points to *known* regions of interest.

Should we explore or exploit?

- **Both.** Good algorithms balance discovery and refinement.
- The *best* balance is an open problem. Some solutions:
  - Always explore some (small) percent of the time.
  - Explore early, exploit later.
  - Alternate between batches of exploration and exploitation.

Summary

- We use `optim` to optimize the surrogate function.
- Using `optim` directly on the true function would be far too expensive.
- We can *exploit* by maximizing the predicted GPR mean.
- We can *explore* by maximizing the predicted GPR standard deviation.

Summary

- We use `optim` to optimize the surrogate function.
- Using `optim` directly on the true function would be far too expensive.
- We can *exploit* by maximizing the predicted GPR mean.
- We can *explore* by maximizing the predicted GPR standard deviation.

- **Next time:** Combining exploitation and exploration into a single search criterion.