DACE R Practice

2024-05-31

Use of DiceOptim Package

Load package

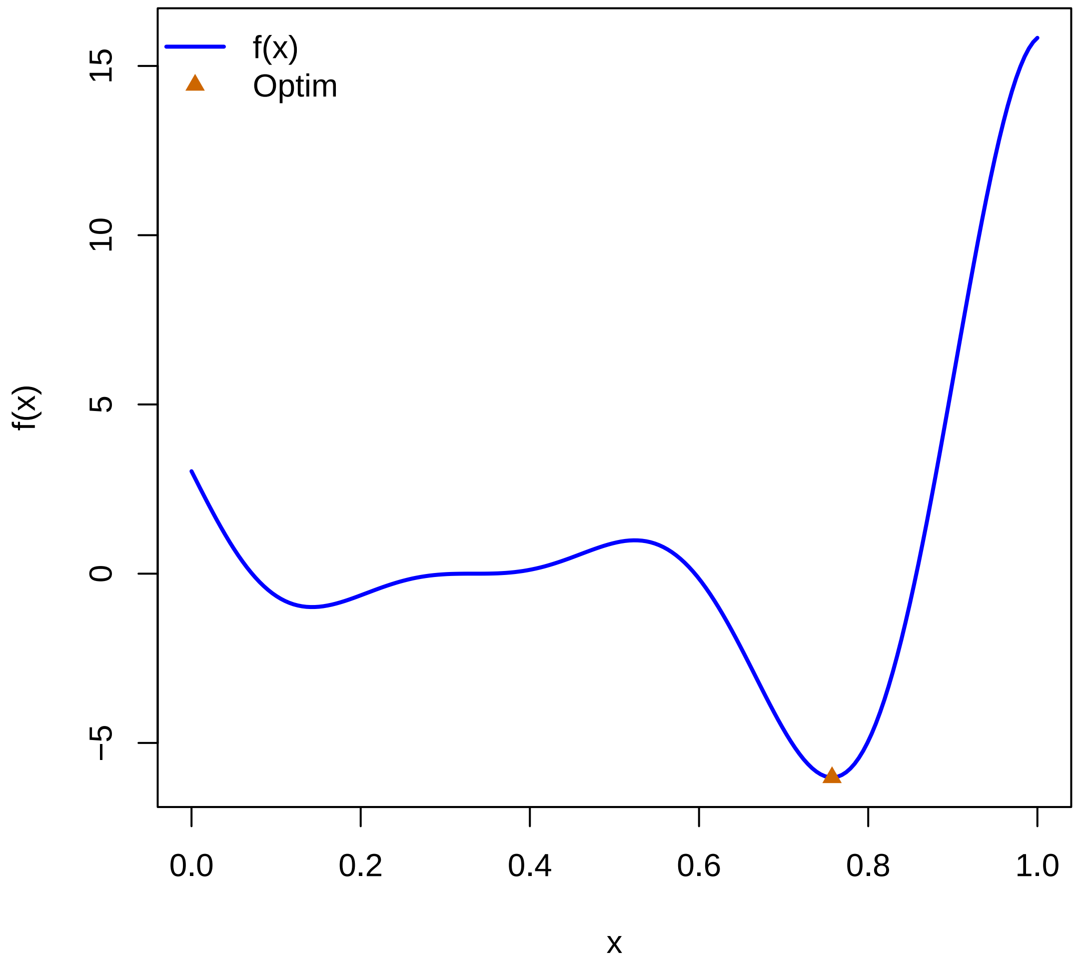
```
library(DiceOptim)
library(lhs)
```

Define the objective function from FORRESTER ET AL. (2008) FUNCTION.

```
forretal08 <- function(x) {
  fact1 <- (6*x - 2)^2
  fact2 <- sin(12*x - 4)
  y <- fact1 * fact2
  return(y)
}
```

It is known that the optimal solution is at $x^* = 0.7572477$ with objective function value $f(x^*) = -6.02074$. Draw the true curve of the objective function and point the optimal solution.

```
optimSol <- list(x = 0.7572477, y = -6.02074)
x_grid <- as.matrix(seq(0, 1, length = 200))
y_grid <- apply(x_grid, 1, forretal08)
plot(x_grid, y_grid, type = "l", col = "blue", lwd = 2,
     xlab = "x", ylab = "f(x)")
points(optimSol$x, optimSol$y, pch = 17, col = "darkorange3")
legend("topleft", c("f(x)", "Optim"), bty = "n",
       pch = c(NA, 17), col = c("blue", "darkorange3"),
       lty = c(1, NA), lwd = c(2, NA))
```



Generate the initial design of 5 experiment points. To fit the Gaussian Process, a.k.a. Kriging, model, we use the space-filling design. One of the popularly used space-filling design is the Latin hypercube design.

```
n <- 5 # Initial Sample size
d <- 1 # Dimension of input variable
#
set.seed(1)
X <- lhs::optimumLHS(n, d)
train_y <- as.matrix(apply(X, 1, forretal08))
train_x <- data.frame(x = X)
```

Fit the Gaussian Process model (Kriging model) using `km()`. Turn on the input argument `nugget.estim = TURE` for stable estimates.

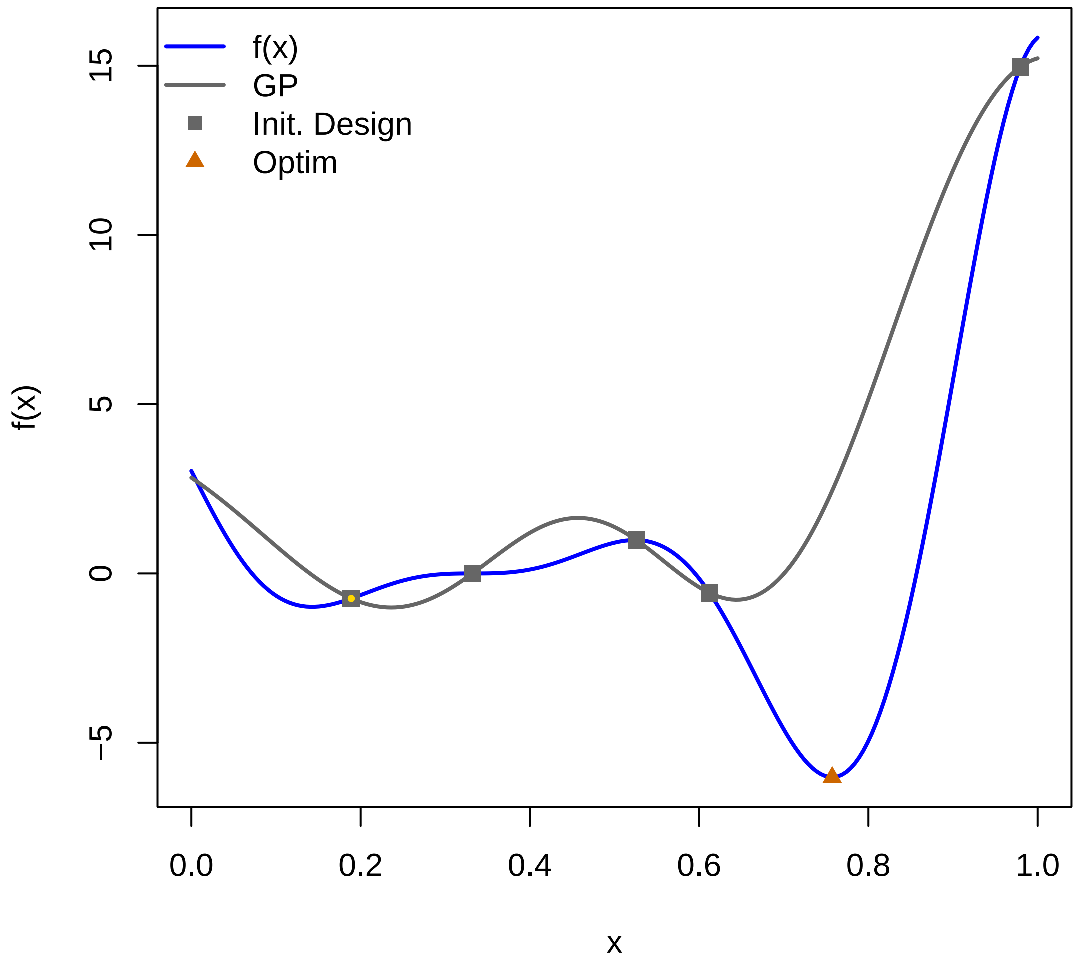
```
fit <- km(design = train_x, response = train_y, covtype = "gauss",
          nugget.estim = TRUE)
```

```
##
## optimisation start
## ------------------
## * estimation method   : MLE
## * optimisation method : BFGS
```

```
## * analytical gradient : used
## * trend model : ~1
## * covariance model :
##   - type :  gauss
##   - nugget : unknown homogenous nugget effect
##   - parameters lower bounds :  1e-10
##   - parameters upper bounds :  1.581486
##   - upper bound for alpha   :  1
##   - best initial criterion value(s) :  -15.12244
##
## N = 2, M = 5 machine precision = 2.22045e-16
## At X0, 0 variables are exactly at the bounds
## At iterate     0  f=       15.122  |proj g|=      0.29981
## At iterate     1  f =       15.105  |proj g|=       0.23257
## At iterate     2  f =        14.96  |proj g|=       0.22498
## At iterate     3  f =       14.843  |proj g|=       0.44419
## At iterate     4  f =       14.836  |proj g|=       0.15904
## At iterate     5  f =       14.836  |proj g|=     0.0010965
## At iterate     6  f =       14.836  |proj g|=    8.3181e-07
##
## iterations 6
## function evaluations 10
## segments explored during Cauchy searches 8
## BFGS updates skipped 0
## active bounds at final generalized Cauchy point 1
## norm of the final projected gradient 8.31812e-07
## final function value 14.8364
##
## F = 14.8364
## final  value 14.836437
## converged
```

Predict the response surface.

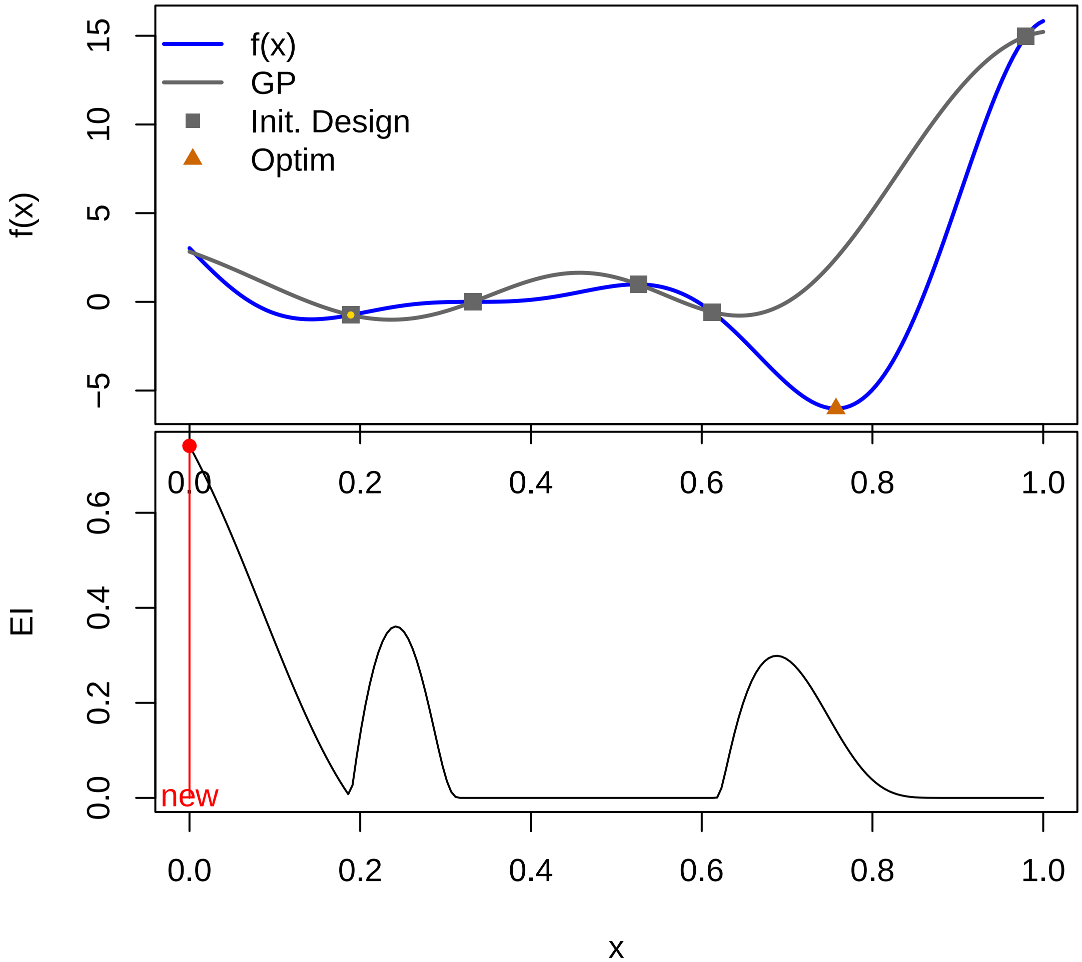
```
pred_y <- predict(fit, data.frame(x = x_grid), type = "UK")
#
bestloc <- which.min(train_y)
#
plot(x_grid, y_grid, type = "l", col = "blue", lwd = 2,
     xlab = "x", ylab = "f(x)")
points(optimSol$x, optimSol$y, pch = 17, col = "darkorange3")
#
points(x_grid, pred_y$mean, type = "l", col = "#666666", lwd = 2)
points(train_x$x, train_y, pch = 15, cex = 1.2, col = "#666666")
points(train_x$x[bestloc], train_y[bestloc], pch = 16, cex = 0.5, col = "gold1")
#
legend("topleft", c("f(x)", "GP", "Init. Design", "Optim"), bty = "n",
       pch = c(NA, NA, 15, 17), col = c("blue", "#666666", "#666666", "darkorange3"),
       lty = c(1, 1, NA, NA), lwd = c(2, 2, NA, NA))
```



Compute the Expected-Improvement criterion values on the grid of the design space $x \in [0, 1]$.

```
EI_values <- apply(x_grid, 1, EI, fit, type = "UK")
mloc <- which.max(EI_values)
#
par(mfrow = c(2, 1))
par(mar = c(.1, 4, 4, 2))
plot(x_grid, y_grid, type = "l", col = "blue", lwd = 2,
     xlab = "x", ylab = "f(x)")
points(optimSol$x, optimSol$y, pch = 17, col = "darkorange3")
#
points(x_grid, pred_y$mean, type = "l", col = "#666666", lwd = 2)
points(train_x$x, train_y, pch = 15, cex = 1.2, col = "#666666")
points(train_x$x[bestloc], train_y[bestloc], pch = 16, cex = 0.5, col = "gold1")
#
legend("topleft", c("f(x)", "GP", "Init. Design", "Optim"), bty = "n",
       pch = c(NA, NA, 15, 17), col = c("blue", "#666666", "#666666", "darkorange3"),
       lty = c(1, 1, NA, NA), lwd = c(2, 2, NA, NA))
#
par(mar = c(5, 4, .1, 2))
plot(x_grid, EI_values, type = "l", col = "black",
     xlab = "x", ylab = "EI")
points(x_grid[mloc], EI_values[mloc], pch = 16, col = "red")
```

```r
arrows(x_grid[mloc], EI_values[mloc], x_grid[mloc], 0, code = 0, col = "red")
text(x_grid[mloc], 0, "new", col = "red")
```




```r
par(mar = c(5, 4, 4, 2) + .1)
par(mfrow = c(1, 1))
```

By maximizing the Expected-Improvement criterion, we now have a new experiment point that the GP model expects an improvement or experiences a lack of prediction accuracy.

```r
new_x <- x_grid[mloc]
new_y <- forretal08(new_x)
# Add new points and its experiment result into the data
train_x <- rbind(train_x, new_x)
train_y <- c(train_y, new_y)
```

Put Them All Together

Define the function of GP-fitting and EI-maximizing.

```
seqOptim <- function(design, response, candidates) {
  model <- km(design = design, response = response, covtype = "gauss",
              nugget.estim = TRUE)
  EI_values <- apply(candidates, 1, EI, model, type = "SK")
  mloc <- which.max(EI_values)
  return(list(design = design, response = response,
              model = model, EI_values = EI_values,
              new_x = candidates[mloc]))
}
```

Optimization Steps:

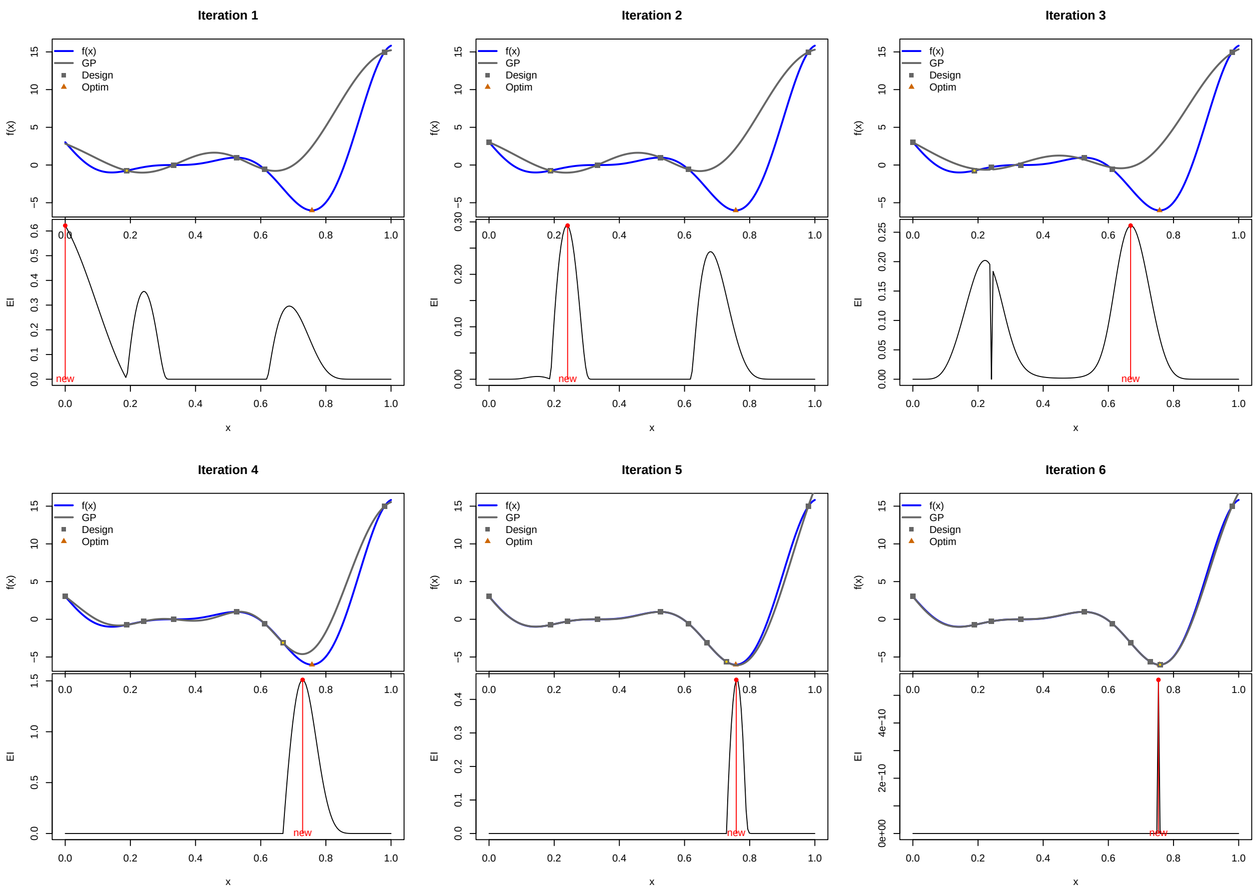
1. Randomly select an initial design of n points
2. Run experiments to obtain the response values
3. Fit GP model
4. Maximize EI criterion
5. For the new design, run one experiment for its response value
6. Add new data to the training set
7. Repeat steps 3 to 6 until observing a satisfactory experiment result or running out the budget.

```
n <- 5 # Initial sample size
d <- 1 # Dimension of input variable
#
set.seed(1)
init_X <- lhs::optimumLHS(n, d)
init_y <- as.matrix(apply(init_X, 1, forretal08))
train_x <- data.frame(x = init_X)
train_y <- init_y
#
bestloc <- which.min(train_y)
#
nIter <- 6
results <- vector("list", nIter)
bestHist <- matrix(0, nIter+1, d + 1)
colnames(bestHist) <- c(sprintf("x%d", 1:d), "y")
bestHist[1,] <- c(train_x[bestloc,], train_y[bestloc])
for (i in 1:nIter) {
  out <- seqOptim(design = train_x, response = train_y, candidates = x_grid)
  results[[i]] <- out
  # Obtain the new point
  new_x <- out$new_x
  new_y <- forretal08(new_x)
  # Add new point and its experiment result into the data
  train_x <- rbind(train_x, new_x)
  train_y <- c(train_y, new_y)
  #
  bestloc <- which.min(train_y)
  bestHist[i+1,] <- c(train_x[bestloc,], train_y[bestloc])
}
```

```
lay <- layout(rbind(matrix(1:6, 2, 3), matrix(7:12, 2, 3)))
for (i in 1:nIter) {
```

```r
  pred_y <- predict(results[[i]]$model, data.frame(x = x_grid), type = "UK")
  EI_values <- results[[i]]$EI_values
  curr_x <- results[[i]]$design
  curr_y <- results[[i]]$response
  bestloc <- which.min(curr_y)
  mloc <- which.max(EI_values)

  par(mar = c(.1, 4, 4, 2))
  plot(x_grid, y_grid, type = "l", col = "blue", lwd = 2,
       xlab = "x", ylab = "f(x)", main = sprintf("Iteration %d", i))
  points(optimSol$x, optimSol$y, pch = 17, col = "darkorange3")
  #
  points(x_grid, pred_y$mean, type = "l", col = "#666666", lwd = 2)
  points(curr_x$x, curr_y, pch = 15, cex = 1.2, col = "#666666")
  points(curr_x$x[bestloc], curr_y[bestloc], pch = 16, cex = 0.5, col = "gold1")
  #
  legend("topleft", c("f(x)", "GP", "Design", "Optim"), bty = "n",
         pch = c(NA, NA, 15, 17), col = c("blue", "#666666", "#666666", "darkorange3"),
         lty = c(1, 1, NA, NA), lwd = c(2, 2, NA, NA))
  #
  par(mar = c(5, 4, .1, 2))
  plot(x_grid, EI_values, type = "l", col = "black",
       xlab = "x", ylab = "EI")
  points(x_grid[mloc], EI_values[mloc], pch = 16, col = "red")
  arrows(x_grid[mloc], EI_values[mloc], x_grid[mloc], 0, code = 0, col = "red")
  text(x_grid[mloc], 0, "new", col = "red")
  par(mar = c(5, 4, 4, 2) + .1)
}
```


Iteration 1
Iteration 2
Iteration 3
Iteration 4
Iteration 5
Iteration 6
f(x)
GP
Design
Optim
EI
x
new