Surrogates 7020

Chapter 1: Introduction to Response Surface

Dr. Alex Bledar Konomi

Department of Mathematical Sciences
University of Cincinnati

Response Surface: What is it?

- Response surface methodology (RSM) is a collection of statistical and mathematical tools useful for developing, improving, and optimizing processes.
- Applications historically come from industry and manufacturing, focused on design, development, and formulation of new products and the improvement of existing products but also from (national) laboratory research, and with obvious military application.
- The overarching theme is a study of how *input variables* controlling a product or process potentially influence a *response* measuring performance or quality characteristics.

Response surface

Consider a response Y that depends on controllable input variables $\xi_1, \xi_2, \ldots, \xi_m$. Write

$$Y = f(\xi_1, \xi_2, \ldots, \xi_m) + \epsilon$$

- $f(\xi_1, \xi_2, \ldots, \xi_m) = E(Y) = \eta$
- where ϵ is treated as zero mean idiosyncratic noise possibly representing inherent variation, or the effect of other systems or variables not under our purview at this time.
- A simplifying assumption that $\epsilon \sim N(0, \sigma^2)$ is typical.
- We seek estimates for f and σ^2 from noisy observations Y at inputs ξ.

Response surface

- Inputs $\xi_1, \xi_2, \ldots, \xi_m$ above are called natural variables because they're expressed in their natural units of measurement, such as degrees Celsius (C), pounds per square inch (psi), etc.
- We usually transform these to coded variables $x_1, x_2, \ldots, x_m$ to mitigate hassles and confusion that can arise when working with a multitude of scales of measurement.
- Transformations offering dimensionless inputs $x_1, x_2, \ldots, x_m$ in the unit cube, or scaled to have a mean of zero and standard deviation of one, are common choices.

Simple Function

Consider the relationship between the response variable yield (y) in a chemical process and two process variables: reaction time (ξ_1) and reaction temperature (ξ_2). R code below synthesizes this setting for the benefit of illustration.

```
yield <- function(xi1, xi2)
 {
  xi1 <- 3*xi1 - 15
  xi2 <- xi2/50 - 13
  xi1 <- cos(0.5)*xi1 - sin(0.5)*xi2
  xi2 <- sin(0.5)*xi1 + cos(0.5)*xi2
  y <- exp(-xi1^2/80 - 0.5*(xi2 + 0.03*xi1^2 - 40*0.03)^2)
  return(100*y)
 }
```

Visualization

```
xi1 <- seq(1, 8, length=100)
xi2 <- seq(100, 1000, length=100)
g <- expand.grid(xi1, xi2)
y <- yield(g[,1], g[,2])
persp(xi1, xi2, matrix(y, ncol=length(xi2)), theta=45, phi=45,
  lwd=0.5, xlab="xi1 : time", ylab="xi2 : temperature",
  zlab="yield", expand=0.4)

############################

cols <- heat.colors(128)
image(xi1, xi2, matrix(y, ncol=length(xi2)), col=cols,
  xlab="xi1 : time", ylab="xi2 : temperature")
contour(xi1, xi2, matrix(y, ncol=length(xi2)), nlevels=4, add=TRUE)
```

Low Order Polynomial

- Learning about f is lots easier if we make some simplifying approximations.
- Appealing to Taylor's theorem, a low-order polynomial in a small, localized region of the input (x) space is one way forward. Classical RSM focuses on disciplined application of local analysis and sequential refinement of "locality" through conservative extrapolation. It's an inherently hands-on process.

First Order Polynomial

A first-order model, or sometimes called a main effects model, makes sense in parts of the input space where it's believed that there's little curvature in f.

$$\eta = \beta_0 + \beta_1 x_1 + \beta_2 x_2$$

Let assume we know this function. *In practice, such a surface would be obtained by fitting a model to the outcome of a designed experiment.*

```
# Define a function
first.order <- function(x1, x2)
 { 50 + 8*x1 + 3*x2 }
#### Simulate data into a grid
x1 <- x2 <- seq(-1, 1, length=100)
g <- expand.grid(x1, x2)
eta1 <- matrix(first.order(g[,1], g[,2]), ncol=length(x2))
### Visualization
par(mfrow=c(1,2))
persp(x1, x2, eta1, theta=30, phi=30, zlab="eta", expand=0.75, lwd=0.25)
image(x1, x2, eta1, col=heat.colors(128))
contour(x1, x2, matrix(eta1, ncol=length(x2)), add=TRUE)
```

First Order Polynomial

A first-order model with interactions induces a limited degree of curvature via different rates of change of y as x_1 is varied for fixed x_2, and vice versa.

$$\eta = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_{12} x_1 x_2$$

```
# Define a function
first.order.i <- function(x1, x2)
 {50 + 8*x1 + 3*x2 - 4*x1*x2}
#### Simulate data into a grid
eta1i <- matrix(first.order.i(g[,1], g[,2]), ncol=length(x2))
### Visualization
par(mfrow=c(1,2))
persp(x1, x2, eta1i, theta=30, phi=30, zlab="eta", expand=0.75, lwd=0.25)
image(x1, x2, eta1i, col=heat.colors(128))
contour(x1, x2, matrix(eta1i, ncol=length(x2)), add=TRUE)
```

Second Order Polynomial

A second-order model may be appropriate near local optima where f would have substantial curvature.

$$\eta = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_{11} x_1^2 + \beta_{22} x_2^2 + \beta_{12} x_1 x_2$$

```
# Define a function
simple.max <- function(x1, x2)
 {50 + 8*x1 + 3*x2 - 7*x1^2 - 3*x2^2 - 4*x1*x2}

eta2sm <- matrix(simple.max(g[,1], g[,2]), ncol=length(x2))

par(mfrow=c(1,2))
persp(x1, x2, eta2sm, theta=30, phi=30, zlab="eta", expand=0.75, lwd=0.25)
image(x1, x2, eta2sm, col=heat.colors(128))
contour(x1, x2, eta2sm, add=TRUE)
```

Second Order Polynomial

Box and Draper (2007) & Myers, Montgomery, and AndersonCook (2016) provide a beautiful diagram categorizing all of the kinds of second-order surfaces one can encounter in an RSM analysis.

General models, inference and sequential design

The general first-order model on m process variables $x_1, \ldots, x_m$ is

$$\eta = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_m x_m$$

and the general second-order model thus:

$$\eta = \beta_0 + \sum_{i=1}^{m} \beta_i x_i + \sum_{i=1}^{m} \beta_{ii} x_i^2 + \sum_{i=1}^{m} \sum_{k=1}^{i} \beta_{ik} x_i x_k$$

Inference from data can be carried out by ordinary least squares (OLS) or maximum likelihood estimators (MLEs). In this case are exactly the same. Check your course on linear regression or for an good review including R examples, see Sheather (2009).

General models, inference and sequential design

- Besides serving to illustrate RSM methods in action, we shall see how important it is to organize the data collection phase of a response surface study carefully.
- A design is a choice of x's where we plan to observe y's, for the purpose of approximating f. Analysis and designs need to be carefully matched.
- When using a first-order model, some designs are preferred over others. When using a second-order model to capture curvature, a different sort of design is appropriate.
- Design choices often contain features enabling modeling assumptions to be challenged, e.g., to check if initial impressions are supported by the data ultimately collected.

Computer experiments

- Mathematical models implemented in computer codes are now commonplace as a means of avoiding expensive field data collection.
- Codes can be computationally intensive, solving systems of differential equations, finite element analysis, Monte Carlo quadrature/approximation, individual/agent based models (I/ABM), and more.
- Highly nonlinear response surfaces, high signal-to-noise ratios (often deterministic evaluations) and global scope demands a new approach to design and modeling compared to a classical RSM setting.
- As computing power has grown, so too has simulation fidelity, adding depth in terms of both accuracy and faithfulness to the best understanding of the physical, biological, or social dynamics in play.

Example: weight wings

The following equation has been used to help understand the weight of an unpainted light aircraft wing as a function of nine design and operational parameters Forrester et al..

$$W = 0.036 S_w^{.758} W_{fw}^{0.0035} (\frac{A}{\cos^2 \Lambda})^{0.6} q^{0.006} \lambda 0.04 (\frac{100 R_{tc}}{cos \Lambda})^{-0.3} (N_z W_{dg})^{0.49} \quad (1)$$

where: S_w is Wing area (ft2), W_{fw} Weight of fuel in wing (lb), A Aspect ratios, Λ Quarter-chord sweep (deg), q Dynamic pressure at cruise (lb/ft^2),λ Taper ratio, R_{tc} Aerofoil thickness to chord ratio, N_z Ultimate load factor, W_{dg} Final design gross weight (lb)

Wing Example

```
wingwt <- function(Sw=0.48, Wfw=0.4, A=0.38, L=0.5, q=0.62, l=0.344,
  Rtc=0.4, Nz=0.37, Wdg=0.38)
 {
  ## put coded inputs back on natural scale
  Sw <- Sw*(200 - 150) + 150
  Wfw <- Wfw*(300 - 220) + 220
  A <- A*(10 - 6) + 6
  L <- (L*(10 - (-10)) - 10) * pi/180
  q <- q*(45 - 16) + 16
  l <- l*(1 - 0.5) + 0.5
  Rtc <- Rtc*(0.18 - 0.08) + 0.08
  Nz <- Nz*(6 - 2.5) + 2.5
  Wdg <- Wdg*(2500 - 1700) + 1700

  ## calculation on natural scale
  W <- 0.036*Sw^0.758 * Wfw^0.0035 * (A/cos(L)^2)^0.6 * q^0.006
  W <- W * l^0.04 * (100*Rtc/cos(L))^(-0.3) * (Nz*Wdg)^(0.49)
  return(W)
 }
```

Computer Model

Weight response as a function of N_z and A with an imagecontour plot.

```
x <- seq(0, 1, length=100)
g <- expand.grid(x, x)

W.A.Nz <- wingwt(A=g[,1], Nz=g[,2])


cs <- heat.colors(128)
bs <- seq(min(W.A.Nz), max(W.A.Nz), length=129)

image(x, x, matrix(W.A.Nz, ncol=length(x)), col=cs, breaks=bs,
  xlab="A", ylab="Nz")
contour(x, x, matrix(W.A.Nz, ncol=length(x)), add=TRUE)
```

Apparently an aircraft wing is heavier when aspect ratios A are high, and designed to cope with large g-forces (large N_z), with a compounding effect.

Computer Model

Weight response as a function of taper ratio λ and fuel weight W_{fw}.

```
W.l.Wfw <- wingwt(l=g[,1], Wfw=g[,2])

image(x, x, matrix(W.l.Wfw,ncol=length(x)), col=cs, breaks=bs,
  xlab="l", ylab="Wfw")
contour(x,x, matrix(W.l.Wfw,ncol=length(x)), add=TRUE)
```

Need for Surrogates

- For each pair we evaluated wingwt $10,000$ times. Doing the same for all pairs would require $360K$ evaluations, not a reasonable number with a real computer simulation that takes any non-trivial amount of time to evaluate.
- Even at just 1s per evaluation, presuming speedy but not instantaneous numerical simulation in a slightly more realistic setting, we're talking > 100 hours.
- Many solvers take minutes/hours/days to execute a single run. Even with great patience, or distributed evaluation in an HPC setting, wed only really know about pairs.
- How about main effects or three-way interactions? A different strategy is needed.

Surrogate modeling and design

- Computer model $f(x) : R^p \to R$ is expensive to evaluate.
- To economize on expensive runs, avoiding a grid in each pair of coordinates say, the typical setup instead entails choosing a small design $X_n = x_1, \ldots, x_n$ of locations in the full m-dimensional space, (where $m = 9$ for wingwt).
- Runs at those locations complete a set of n example evaluation pairs (x_i, y_i), where $y_i \sim f(x_i)$ for $i = 1, \ldots, n$. Collect the n data pairs as $D_n = (X_n, Y_n)$ where X_n is an $n \times m$ matrix and Y_n is an n-vector.
- Use these data to train a statistical (regression) model, producing an emulator $\hat{f}_n \equiv \hat{f}|D_n$ whose predictive equations may be used as a surrogate $\hat{f}_n(x')$ for $f(x')$ at novel x' locations in the m-dimensional input space.

Surrogate modeling and design

a Provide a predictive distribution $\hat{f}(x')$ whose mean can be used as a surrogate for $f(x')$ at new x' locations and whose variance provides uncertainty estimates intervals for $f(x')$ - that have good coverage properties;

b may interpolate when computer model f is deterministic;

c can be used in any way f could have been used, qualified with appropriate uncertainty quantification (i.e., bullet a mapped to the intended use, say to optimize);

d and finally, fitting $\hat{f}$ and making predictions $\hat{f}(x')$ should be much faster than working directly with $f(x')$.

Latin Hypercube & First-order Model

Just to see what might come up, let's fit a parsimonious first-order model with interactions, cheating with the log response, using backward step-wise elimination with BIC. Other alternatives include AIC with $k = 2$ below, or F-testing.

```
library(lhs)
n <- 1000
X <- data.frame(randomLHS(n, 9))
names(X) <- names(formals(wingwt))

plot(X[,1:2], pch=19, cex=0.5)
abline(h=c(0.6, 0.8), col=2, lwd=2)

inbox <- X[,1] > 0.6 &  X[,1] < 0.8
sum(inbox)/nrow(X)

Y <- wingwt(X[,1], X[,2], X[,3], X[,4], X[,5], X[,6], X[,7], X[,8], X[,9])

fit.lm <- lm(log(Y) ~ .^2, data=data.frame(Y,X))
fit.lmstep <- step(fit.lm, scope=formula(fit.lm), direction="backward",
  k=log(length(Y)), trace=0)
```

Gaussian Process

```
library(laGP)

fit.gp <- newGPsep(X, Y, 2, 1e-6, dK=TRUE)
mle <- mleGPsep(fit.gp)

baseline <- matrix(rep(as.numeric(formals(wingwt)), nrow(g)),
  ncol=9, byrow=TRUE)
XX <- data.frame(baseline)
names(XX) <- names(X)
XX$A <- g[,1]
XX$Nz <- g[,2]

p <- predGPsep(fit.gp, XX, lite=TRUE)

image(x, x, matrix(p$mean, ncol=length(x)), col=cs, breaks=bs,
  xlab="A", ylab="Nz")
contour(x, x, matrix(p$mean, ncol=length(x)), add=TRUE)
```

Gaussian Process

```
meq1 <- meq2 <- me <- matrix(NA, nrow=length(x), ncol=ncol(X))
for(i in 1:ncol(me)) {
  XX <- data.frame(baseline)[1:length(x),]
  XX[,i] <- x
  p <- predGPsep(fit.gp, XX, lite=TRUE)
  me[,i] <- p$mean
  meq1[,i] <- qt(0.05, p$df)*sqrt(p$s2) + p$mean
  meq2[,i] <- qt(0.95, p$df)*sqrt(p$s2) + p$mean
}
```