GPfit – An R package for fitting GP model

Blake MacDonald, Pritam Ranjan and Hugh Chipman

Acadia University, Nova Scotia, Canada

Introduction

Gaussian process (GP) models are popular surrogates for emulating deterministic computer simulator outputs.

Fitting a GP model can be computationally unstable due to near-singularity of the spatial correlation matrix R. We use the nugget based approach in Ranjan et al. (2011).

GP model fitting procedure requires numerous evaluations of determinant and inverse of R (i.e., every likelihood evaluations is expensive)

Maximum likelihood approach: the log-likelihood function of the GP model can have multiple local optima.

We follow a clustering based multi-start BFGS algorithm for optimizing the log-likelihood. This is faster than genetic algorithm and more accurate than *mlegp*.

Gaussian process model

Assume the simulator is deterministic, process is stationary, and the outputs are scalar.

Data: $\{(x_i, y_i), i = 1, \dots, n\}$, where $x_i \in [0,1]^d$.

Model:

$$y_i = \mu + z(x_i), \quad i = 1, \dots, n,$$

where μ is constant mean, $z(x_i)$ is a GP. That is, $E(Z(x_i) = 0$, and $Cov\left(Z(x_i), Z(x_j)\right) = \sigma^2 R_{ij}$. We use Gaussian correlation

$$R_{ij} = \prod_{k=1}^{d} exp\left(-\theta_k |x_{ik} - x_{jk}|^2\right),$$

where $\theta_k \in [0, \infty)$. The closed form estimators of μ and σ^2 are given by

$$\hat{\mu}(\theta) = (1'R^{-1}1)'(1'R^{-1}Y)$$

and

$$\hat{\sigma}^2(\theta) = \frac{(Y - 1\hat{\mu})'R^{-1}(Y - 1\hat{\mu})}{n}.$$

The deviance $(-2\log(L_\theta))$ to be optimized is

$$\log(|R|) + n\log[(Y - 1\hat{\mu})'R^{-1}(Y - 1\hat{\mu})].$$

Near-singularity of R

An $n \times n$ correlation matrix R is said to be ill-conditioned or near-singular if its condition number

$$\kappa(R) = ||R|| * ||R^{-1}||$$

is too large.

This is a common problem in fitting GP models which occurs if any pair of design points in the input space are close together (Neal 1997).

Popular approach: replace R by $R_\delta = R + \delta \cdot I$.

To minimize over-smoothing, Ranjan et al. (2011) suggests using the lower bound on δ, i.e.,

$$\delta_{lb} = \max\left\{\frac{\lambda_n(\kappa(R) - e^a)}{\kappa(R)(e^a - 1)}, 0\right\},$$

Where λ_n is the maximum eigenvalue of R, and e^a is the threshold of $\kappa(R)$ that ensures a well-conditioned R.

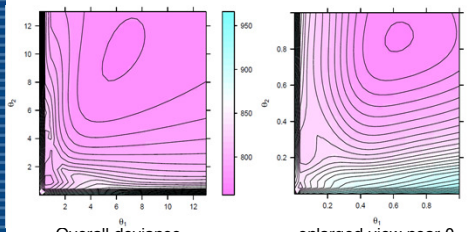
That is, we use $R_{\delta_{lb}} = R + \delta_{lb} \cdot I$ in place of R in the log-likelihood expression.

Multiple local optima of $-2\log(L_\theta)$

For the 2-d GoldPrice function, $x \in [-2,2]^2$,

$$y(x) = [1 + (x_1 + x_2 + 1)^2\{19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2\}] * [30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)].$$

The inputs were scaled to $[0,1]^2$. Generated a 30-point maximin LHD, and evaluated the log-likelihood function



Overall deviance

enlarged view near 0

Optimization near zero is tricky.

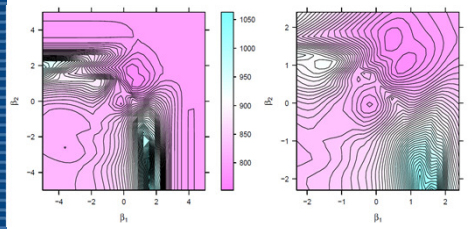
Reparametrization of R

Let $\beta_k = \log_{10}(\theta_k)$, for $k = 1, \dots, d$. Then, the Gaussian correlation is

$$R_{ij} = \prod_{k=1}^{d} exp\left(-10^{\beta_k}|x_{ik} - x_{jk}|^2\right),$$

where $\beta_k \in (-\infty, \infty)$.

For the 2-d GoldPrice function, the deviance function is easier to optimize (local optima are in the middle now).



Overall deviance

enlarged view near 0

Optimization algorithm

Plausible values of β_k's

$$(\exp(-5) \approx) 0 \leq R_{ij} \leq 1 \ (\approx \exp(-10^{-4}))$$

Assuming isotropic correlation, $x \in [0,1]^d$ and $n = 10 \cdot d$, let Ω_0 is given by

$$-2 - \log_{10}(d) \leq \beta_k \leq \log_{10}(500) - \log_{10}(d).$$

Algorithm

1. Choose $200d$ –point maximin LHD for $\beta \in \Omega_0^d$, and evaluate $-2\log(L_\beta)$ for each β.
2. Choose $80d$ values of β that gives smallest $-2\log(L_\beta)$
3. Use k-means clustering on these $80d$ values of β to find $2d$ groups and the cluster means.
4. For $d \geq 2$, run BFGS along the diagonal (starting at 25%, 50% and 75%) to find the best solution.
5. Use these $2d\ (or, 2d + 1)$ points as the starting points of BFGS to find the best minimizer of $-2\log(L_\beta)$.

➢ BFGS instead of genetic algorithm makes it a bit faster

➢ Multiple starting points make the algorithm robust

GPfit package

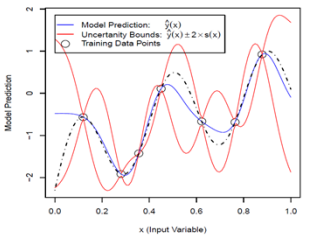
A more complete simulation study showed:

```
> GPmodel = GP_fit(X, Y, control=c(200*d,80*d,2*d),
                   nug_thres=20, trace=FALSE, maxit=100)

> Model_pred = predict.GP(GPmodel, xnew)

> plot.GP(GPmodel, range=c(0,1), resolution=50,
                   surf_check=FALSE, response=TRUE)
```

Examples

```
R> library(GPfit)
R> library(lhs)
R> n = 7
R> x = maximinLHS(n,1)
R> y = matrix(0,n,1)
R> for(i in 1:n){ y[i] = computer_simulator(x[i]) }
R> GPmodel = GP_fit(x,y)

        Number Of Observations: n = 7
        Input Dimensions: d = 1

        Correlation: Exponential (power = 2)
        Correlation Parameters:
            beta_hat
        [1]    1.977

        sigma^2_hat: [1] 0.7444

        delta_lb(beta_hat): [1] 0
        nugget threshold parameter: 20
```

> plot.GP(Gpmodel)



References

- Ranjan, P., Haynes, R. and Karsten, R. (2011). A computationally stable approach to Gaussian process interpolation of deterministic computer simulation data, *Technometrics*, 53(5), 366-378.
- MacDonald, K.B., Ranjan, P. and Chipman, H. (2012). GPfit: An R package for Gaussian process model fitting using a new optimization algorithm, *Journal of Statistical Software* (submitted).