Optimisation

Qinqing Li

2024-08-29

```r
StatCompLab::optimisation()
```

For m-dimensional problems and derivatives approximated by finite differences, a gradient calculation costs at least m extra function evaluations, and a Hessian costs at least 2m^2 extra function evaluations.

For the 2D Rosenbrock function, *count the total number of function* evaluations required by each optimisation method (under the assumption that finite differences were used for the gradient and Hessian) until convergence is reached.

Answer:

m = 2, the number of function evaluations is given by #f + 2 * #gradient + 8 * #hessian

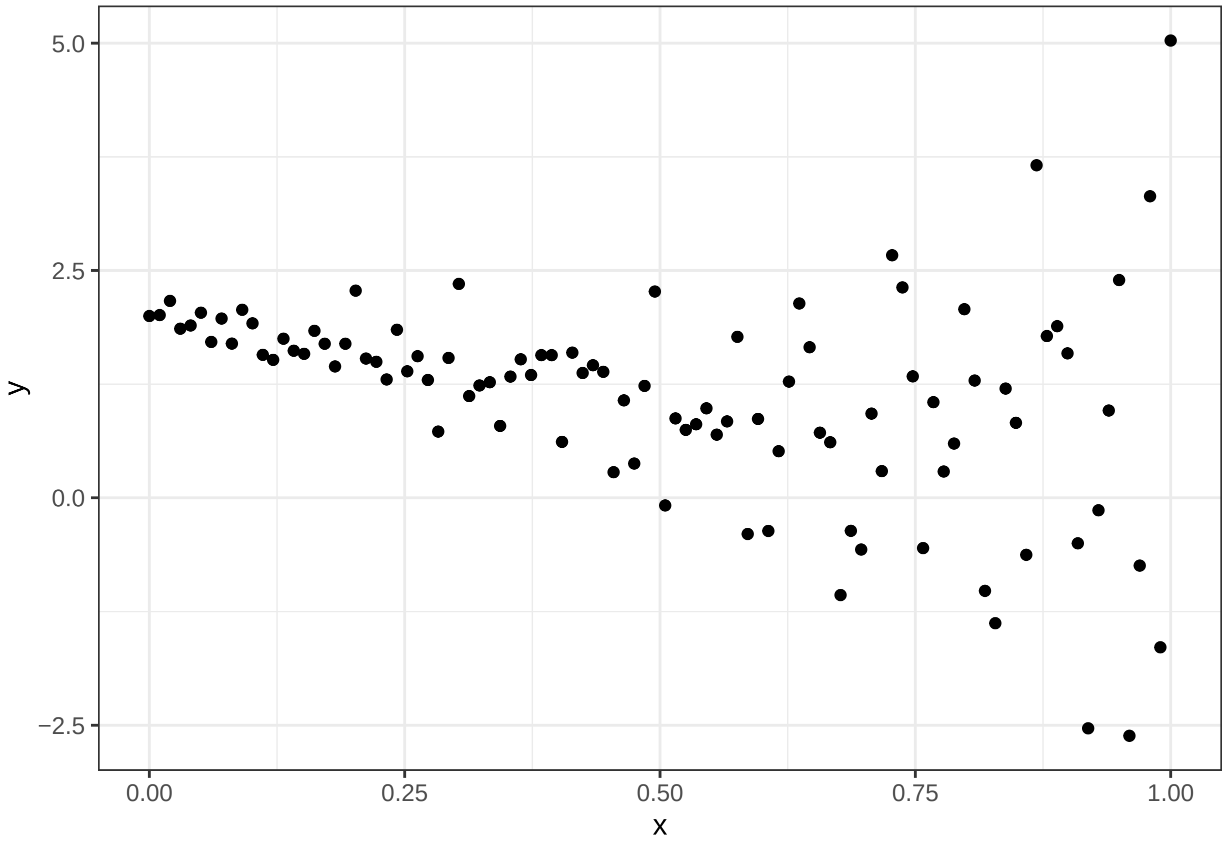
Nelder-Mead Simplex: 202+0+0=202 Gradient Descent: 14132+2*9405+0=32942 Newton(with LS): 29+2* 23+8*22=251 Newton(BFGS): 54+2* 41+8*1=144

Estimating a complicated model

The true parameter of this model is (2,-2,-2,3). In other words, true param for mu is (2,-2) and true param for sigma (sd) is (-2,3). Now, suppose we don't know this information and we want to estimate the true param of this synthetic model using MLE.

```r
n <- 100
theta_true <- c(2, -2, -2, 3)
X <- cbind(1, seq(0, 1, length.out = n))
y <- rnorm(n = n,
           mean = X %*% theta_true[1:2],
           sd = exp(X %*% theta_true[3:4]))
```

```r
ggplot(data.frame(x = X[, 2], y = y)) +
  geom_point(aes(x, y))
```



Function that evaluates the negative log-likelihood for the model.

```
neg_log_lik <- function(theta, y, X) {
  sd <- exp(X %*% theta[3:4])
  mu <- X %*% theta[1:2]
  neg_log_like <- -sum(log(dnorm(y, mean = mu, sd = sd)))
  return(neg_log_like)
}
```

Find the maximum likelihood parameter estimates for our statistical model using the BFGS method with numerical derivatives. Use (0,0,0,0) as the starting point for the optimisation. Did the optimisation converge?

```
#y and X already defined in the above code chunk
opt <- optim(c(0,0,0,0), fn = neg_log_lik, y = y, X=X, method = "BFGS")
print(opt)
```

```
## $par
## [1]  2.006171 -1.949222 -2.008320  2.965715
##
## $value
## [1] 89.34746
##
## $counts
## function gradient
##       54       17
##
## $convergence
## [1] 0
##
```

```
## $message
## NULL
```

Ans: This vector contains the parameter estimates that minimize the negative log-likelihood function. These are your MLEs:

1.982227 -1.952263 are the MLEs for the mean parameters.

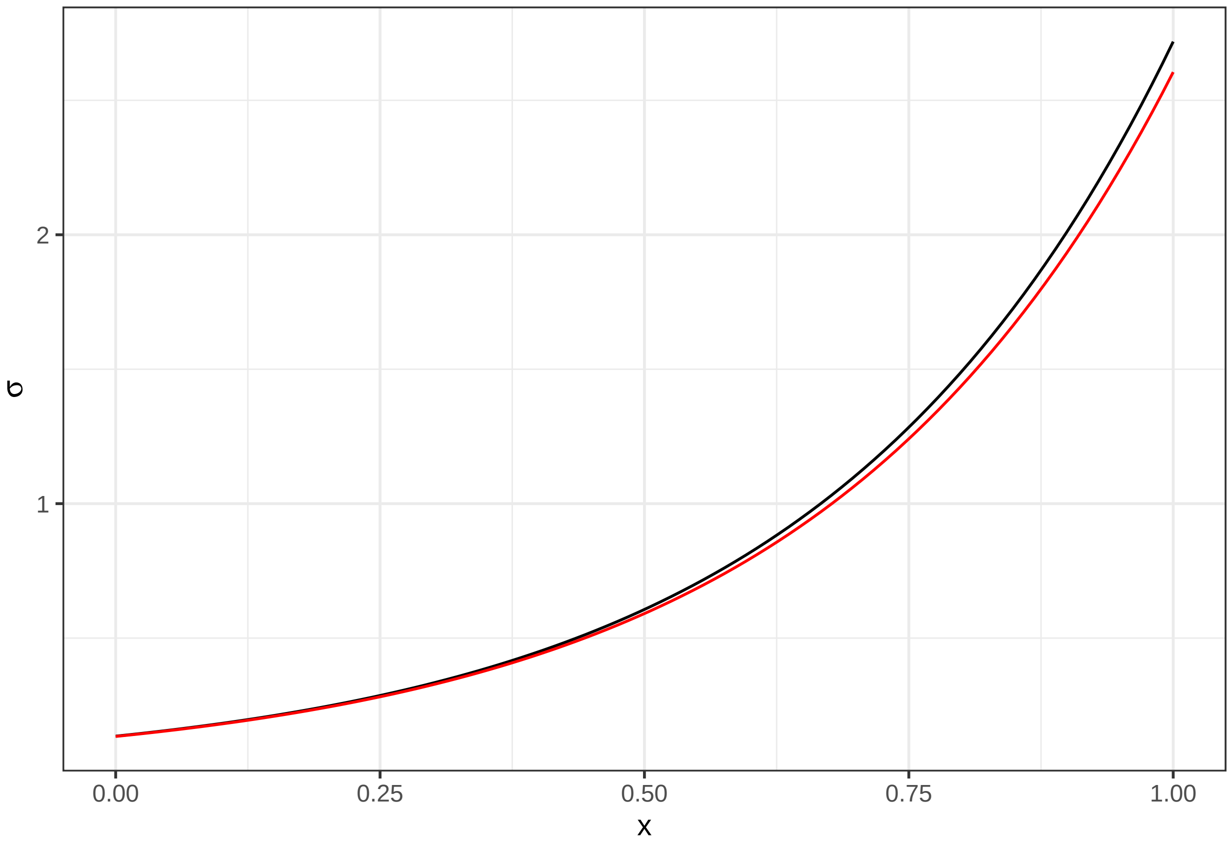
-2.156395 3.045014 are the MLEs for the sd parameters.

which is pretty close to true param (2,-2) and (-2,3) seperately.

Yes converge to 0.

In the plot, estimated values of sigma (sd) is shown in red.

```r
ggplot(data) +
  geom_line(aes(x, sigma_true)) +
  geom_line(aes(x, sigma_est), col = "red") +
  xlab("x") +
  ylab(expression(sigma))
```

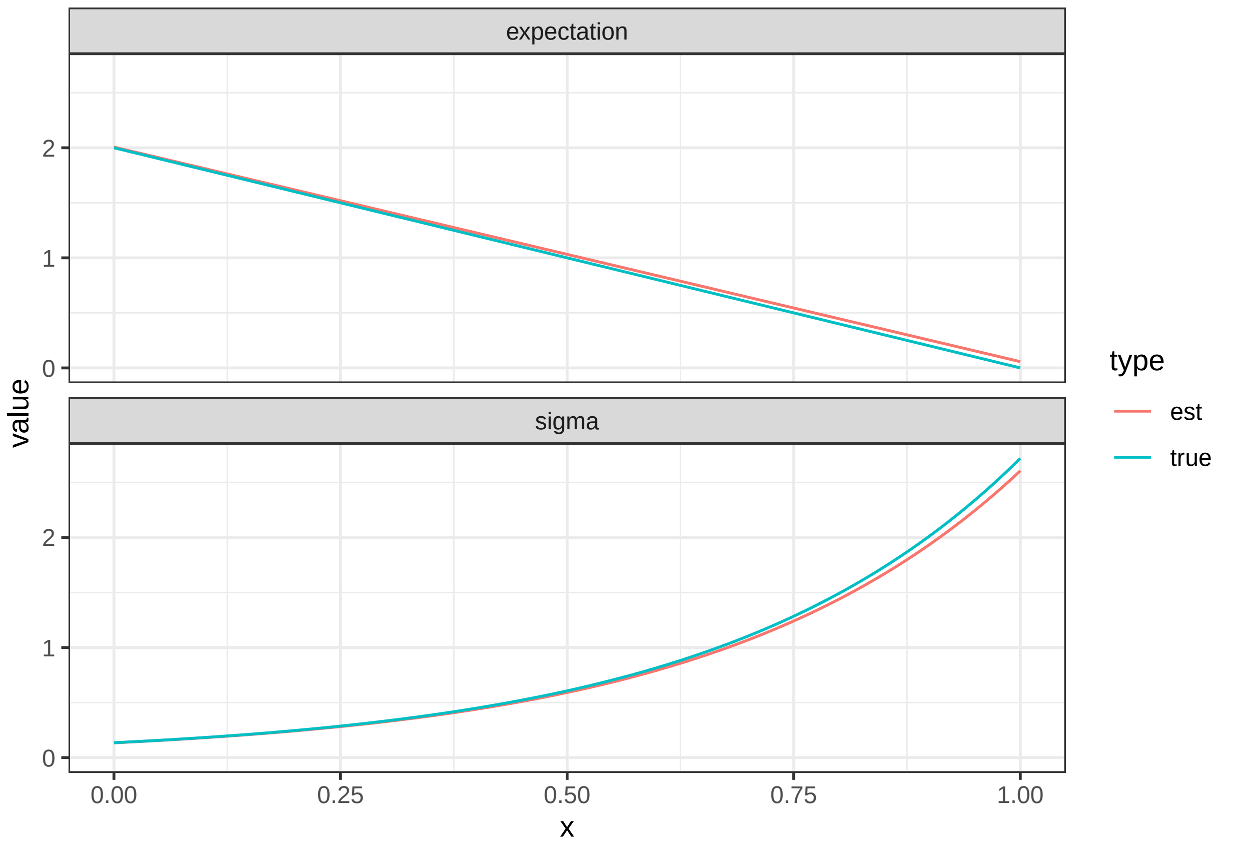


Data wrangling

```r
suppressPackageStartupMessages(library(tidyverse))
data_long <-
  data %>%
  pivot_longer(cols = -c(x, y),
               values_to = "value",
               names_to = c("property", "type"),
               names_pattern = "(.*)_(.*)")
data_long
```

```
## # A tibble: 400 x 5
##         x     y property    type  value
##     <dbl> <dbl> <chr>       <chr> <dbl>
##   1 0       2.00 expectation true  2
##   2 0       2.00 sigma       true  0.135
##   3 0       2.00 expectation est   2.01
##   4 0       2.00 sigma       est   0.134
##   5 0.0101  2.01 expectation true  1.98
##   6 0.0101  2.01 sigma       true  0.139
##   7 0.0101  2.01 expectation est   1.99
##   8 0.0101  2.01 sigma       est   0.138
##   9 0.0202  2.17 expectation true  1.96
## 10 0.0202  2.17 sigma       true  0.144
## # ... with 390 more rows
```

```
ggplot(data_long, aes(x, value)) +
  geom_line(aes(col = type)) +
  facet_wrap(vars(property), ncol=1)
```



Rerun the optimisation with the extra parameter hessian = TRUE, to obtain a numeric approximation to the Hessian of the target function at the optimum, and compute its inverse, which is an estimate of the covariance matrix for the error of the parameter estimates.

```r
opt1 <- optim(c(0,0,0,0), fn = neg_log_lik, y = y, X=X, method = "BFGS",hessian = TRUE)
print(opt1)
```

```
## $par
## [1]  2.006171 -1.949222 -2.008320  2.965715
##
## $value
## [1] 89.34746
##
## $counts
## function gradient
##       54       17
##
## $convergence
## [1] 0
##
## $message
## NULL
##
## $hessian
##                [,1]          [,2]          [,3]          [,4]
## [1,]  9.522226e+02  1.533664e+02 -6.139853e-04 -0.0004040608
## [2,]  1.533664e+02  4.931767e+01 -4.060201e-04  2.3114982639
## [3,] -6.139853e-04 -4.060201e-04  1.999998e+02 99.9998787101
## [4,] -4.040608e-04  2.311498e+00  9.999988e+01 65.9229178162
```

Estimate of the covariance matrix for the error of the parameter estimates.

```r
inv_hes <- solve(opt1$hessian)
print(inv_hes)
```

```
##                [,1]         [,2]          [,3]          [,4]
## [1,]  0.0021185484 -0.006633330 -0.0004815267  0.0009630398
## [2,] -0.0066333295  0.041185079  0.0029897076 -0.0059792911
## [3,] -0.0004815267  0.002989708  0.0209175803 -0.0318351563
## [4,]  0.0009630398 -0.005979291 -0.0318351563  0.0636703201
```