Метод BFGS
(Broyden–Fletcher–Goldfarb–Shanno)

Содержание

- теоретические основы метода
- пример практической реализации в «R»
- домашнее задание

Теоретические основы метода

Задача:

$$f(\vec{x}) = f(x_1, \dots, x_d) \to \min_{\vec{x}}.$$

Основная идея метода

На каждой *k*-й итерации метода рассматривается квадратичная модель целевой функции

$m_k(\vec{p}) = f(\vec{x}_k) + \nabla f^T(\vec{x}_k)\vec{p} + \frac{1}{2}\vec{p}^T B_k \vec{p},$ где

$\vec{x}_k$ — значения параметров целевой функции на *k*-й итерации

∇f — градиент целевой функции

B_k — обновляемая на каждой итерации положительно определённая $[d \times d]$-матрица (гессиан)

Определение направления изменения $\vec{p}_k$

На каждой итерации метода BFGS определяется направление $\vec{p}_k$, вдоль которого изменяются значения параметров целевой функции и величина изменения:

$\vec{x}_{k+1} = \vec{x}_k + \alpha_k \vec{p}_k,$ где

$\alpha_k > 0$ называется «длиной шага»

Направление $\vec{p}_k$ определяется путём минимизации квадратичной модельной функции $m_k(\vec{p})$ и находится в виде $\vec{p}_k = -B_k^{-1} \nabla f(\vec{x}_k)$

Определение длины шага α_k

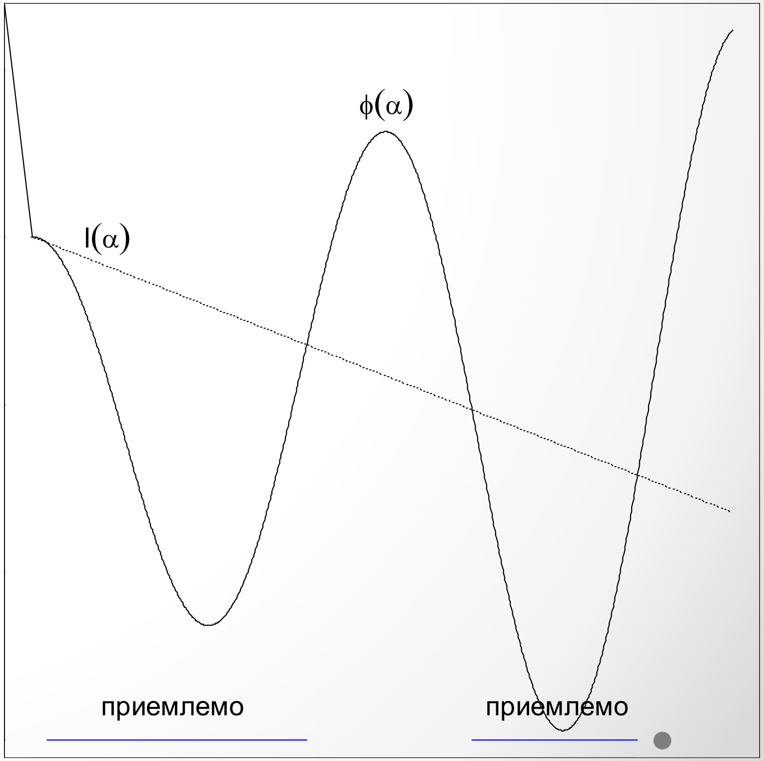
Первое условие Вольфа

Выбор α_k должен гарантировать значительное уменьшение целевой функции:

$$\varphi(\alpha_k) = f(\vec{x}_k + \alpha_k \vec{p}_k) \le f(\vec{x}_k) + c_1 \alpha_k \nabla f^T(\vec{x}_k)\vec{p}_k = l(\alpha_k),$$ где

$c_1 \sim +10^{-4}$ — положительная константа

Величина уменьшения целевой функции должна быть пропорциональна длине шага α_k и градиенту по направлению $\vec{p}_k$



Определение длины шага α_k

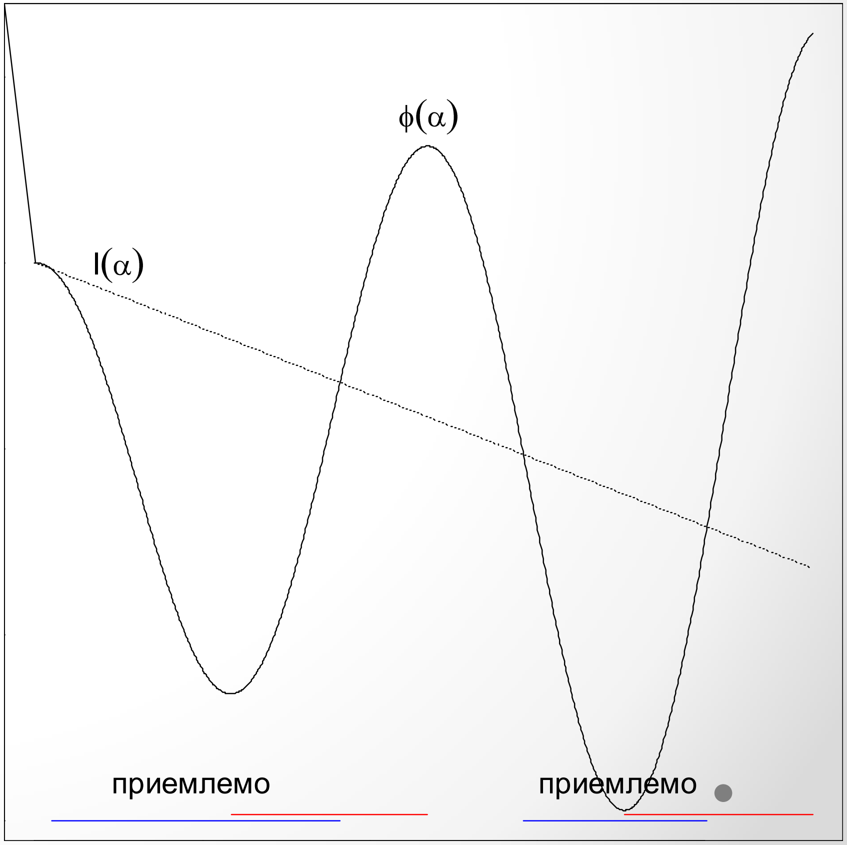
Второе условие Вольфа

Величина α_k не должна быть слишком маленькой:

$\varphi'(\alpha_k) = \nabla f^T(\vec{x}_k + \alpha_k \vec{p}_k)\vec{p}_k \geq c_2 \nabla f^T(\vec{x}_k)\vec{p}_k = c_2 \varphi'(0)$, где

$c_2 \sim +0{,}9$ — положительная константа

Если наклон функции $\varphi(\alpha_k)$ достаточно велик, то разумно продолжить изменять параметры функции в этом направлении, увеличивая α_k



Обновление матрицы B_k

Обновлённая матрица B_{k+1} выбирается так, чтобы градиенты модельной и целевой функции совпадали на двух последних итерациях: $\nabla m_{k+1}(-\alpha_k \vec{p}_k) = \nabla f(\vec{x}_k),\ \nabla m_{k+1}(0) = \nabla f(\vec{x}_{k+1})$

Этому соответствует условие

$$B_{k+1}\vec{s}_k = \vec{y}_k,$$

$$\vec{s}_k = \vec{x}_{k+1} - \vec{x}_k = \alpha_k \vec{p}_k, \qquad \vec{y}_k = \nabla f(\vec{x}_{k+1}) - \nabla f(\vec{x}_k)$$

Среди бесконечного множества вариантов выбирается ближайшая к B_k матрица:

$$\begin{cases} \min\limits_B \|B - B_k\| \\ B = B^T,\ B\vec{s}_k = \vec{y}_k \end{cases}$$

Решением этой задачи является

$$B_{k+1} = \left(\boldsymbol{I} - \rho_k \vec{y}_k \vec{s}_k^T\right) B_k \left(\boldsymbol{I} - \rho_k \vec{s}_k \vec{y}_k^T\right) + \rho_k \vec{y}_k \vec{y}_k^T,$$

$\rho_k = \dfrac{1}{\vec{y}_k^T \vec{s}_k}$, $\boldsymbol{I}$ — единичная матрица

Обновление матрицы $H_k = B_k^{-1}$

При нахождении направления изменения параметров $\vec{p}_k = -B_k^{-1}\nabla f(\vec{x}_k)$ каждый раз приходится инвертировать матрицу B_k, что является ресурсоёмкой процедурой при большом числе параметров

Чтобы избежать этого, на практике обновлению подвергается матрица $H_k = B_k^{-1}$:

$$H_{k+1} = \left(\boldsymbol{I} - \rho_k \vec{s}_k \vec{y}_k^T\right) H_k \left(\boldsymbol{I} - \rho_k \vec{y}_k \vec{s}_k^T\right) + \rho_k \vec{s}_k \vec{s}_k^T$$

Алгоритм метода BFGS

1. Задать начальные значения параметров $\vec{x}_0$, обратного гессиана H_0 и определить параметр конвергенции $\varepsilon > 0$
2. $k \coloneqq 0$
3. Пока верно $\|\nabla f(\vec{x}_k)\| > \varepsilon$, выполнять
   - найти направление поиска $\vec{p}_k \coloneqq -H_k \nabla f(\vec{x}_k)$
   - определить длину шага α_k из условий Вольфа
   - рассчитать новые значения параметров $\vec{x}_{k+1} \coloneqq \vec{x}_k + \alpha_k \vec{p}_k$
   - $\vec{s}_k \coloneqq \vec{x}_{k+1} - \vec{x}_k, \;\; \vec{y}_k \coloneqq \nabla f(\vec{x}_{k+1}) - \nabla f(\vec{x}_k)$
   - обновить матрицу H_k до H_{k+1}
   - $k \coloneqq k + 1$

В качестве H_0 обычно используют обратный гессиан в точке $\vec{x}_0$ или единичную матрицу

Алгоритм определения длины шага α_k (линейный поиск)

1. Задать начальные значения $\gamma \in (0;1)$, $c_1 \in (0;1)$, $\tilde{a} = 1$
2. Пока неверно $f(\vec{x}_k + \alpha_k \vec{p}_k) \leq f(\vec{x}_k) + c_1 \alpha_k \nabla f^T(\vec{x}_k)\vec{p}_k$, выполнять
   - $\tilde{a} \coloneqq \gamma \tilde{a}$
3. $\vec{\alpha}_k \coloneqq \tilde{a}$

Литература

- Nocedal J., Write S. Numerical Optimization. Springler Science+Business Media: N.Y., 2006. Chapter 3, pp. 30–65
- Nocedal J., Write S. Numerical Optimization. Springler Science+Business Media: N.Y., 2006. Chapter 6, pp. 135–143

Пример практической реализации в «R»

Задача:

$$f(\vec{x}) = f(x_1, \dots, x_d) \to \min_{\vec{x}}.$$

Целевая и вспомогательные функции

```
f <- function(x) x[1]^4-x[1]^3+x[1]^2+2*(x[1]-1)*(x[2]-1)+x[2]^2
```

функция градиента f

```
gradf <- function(x) {
  g <- numeric(2)
  g[1] <- 4*x[1]^3 - 3*x[1]^2 + 2*x[1] + 2*x[2] - 2
  g[2] <- 2*x[1] + 2*x[2] - 2
  return(g)
}
```

численная аппроксимация градиента (для сложных функций)

```
library(numDeriv)
gradf <- function(x) grad(func=f,x=x)
```

функция линейного поиска α_k

```
line.search <- function(func,x,p,gradfx=gradf(x),
                        a.wave=1,rho=0.90,c=10^-4) {
  ...
  return(alpha)
}
```

Целевая и вспомогательные функции

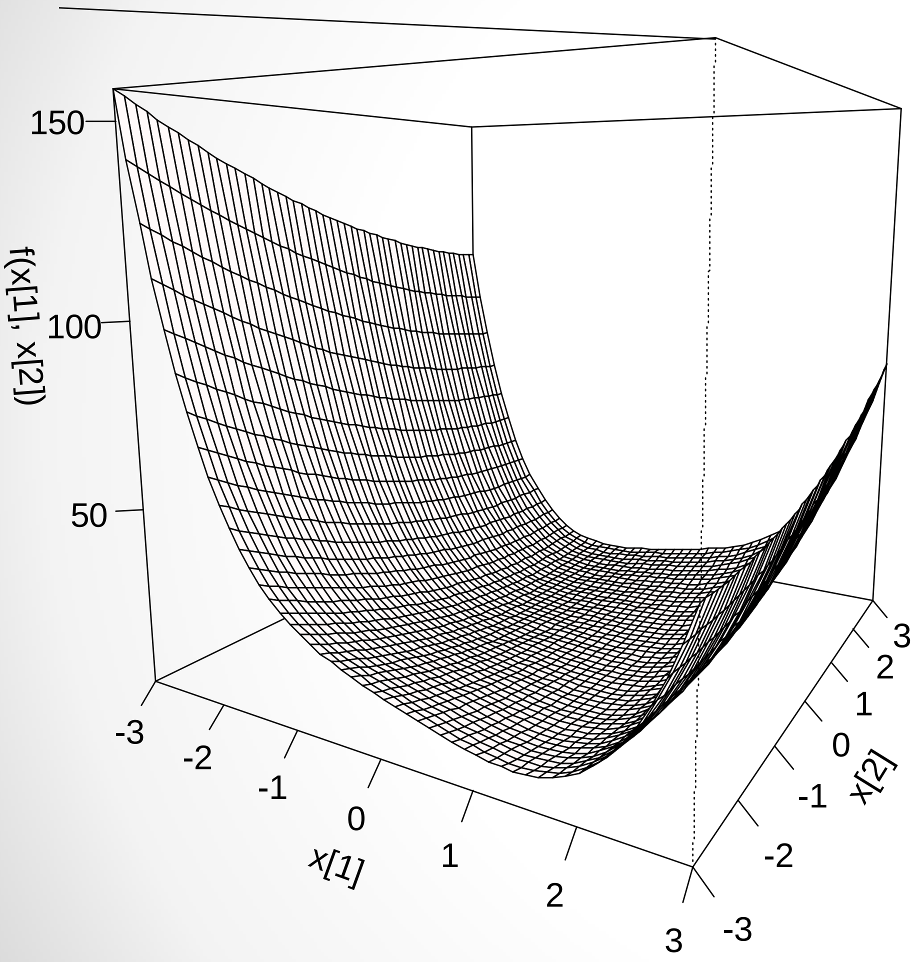
гессиан *B*

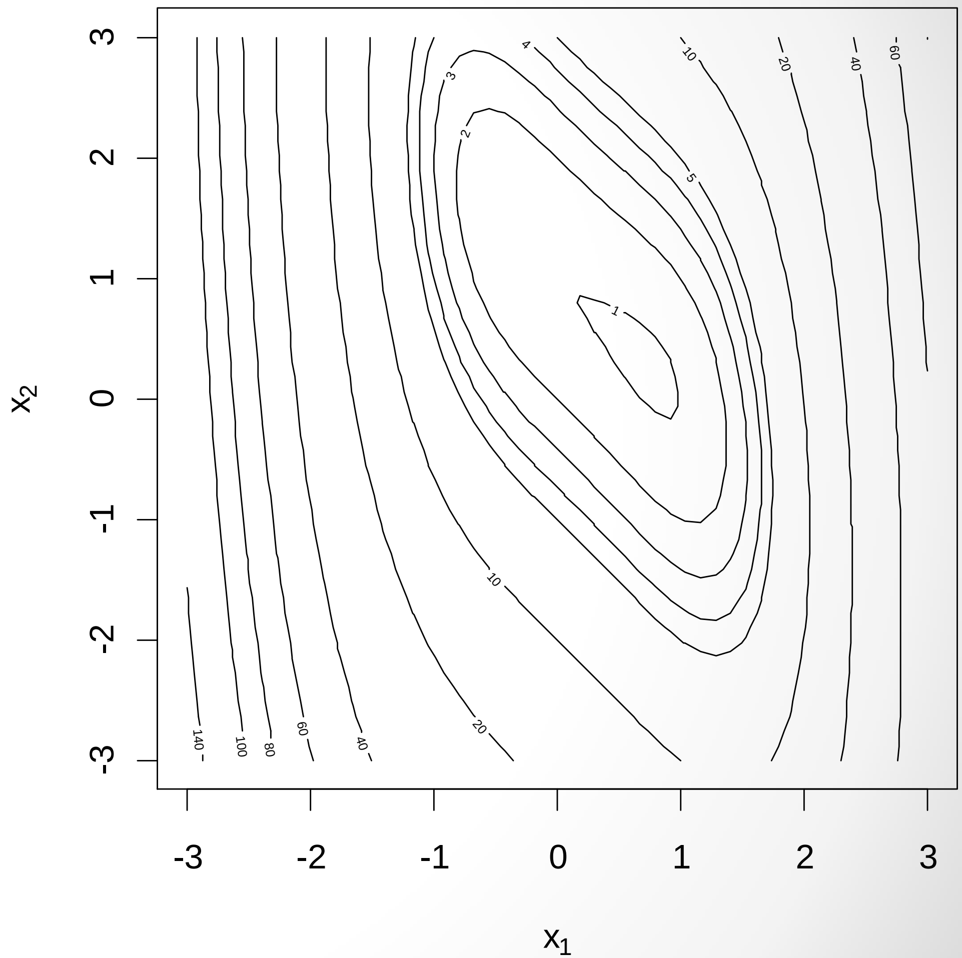
```
hessianf <- function(x) {
  b <- matrix(nrow=2,ncol=2)
  b[1,1] <- 12*x[1]^2 - 6*x[1] + 2
  b[1,2] <- b[2,1] <- b[2,2] <- 2
  return(b)
}
```

численная аппроксимация гессиана

```
hessianf <- function(x) hessian(func=f,x=x)
```

Графики целевой функции





Начальные значения

```
k <- 0
```

значения параметров

```
x <- c(3,3); d <- length(x)
```

градиент

```
gf <- gradf(x); grad.tol <- 10^-12
```

обратный гессиан

```
I <- diag(rep(1,times=d))
H <- try( solve(hessianf(x)) ,silent=TRUE)
if (class(H) == "try-error") H <- I
```

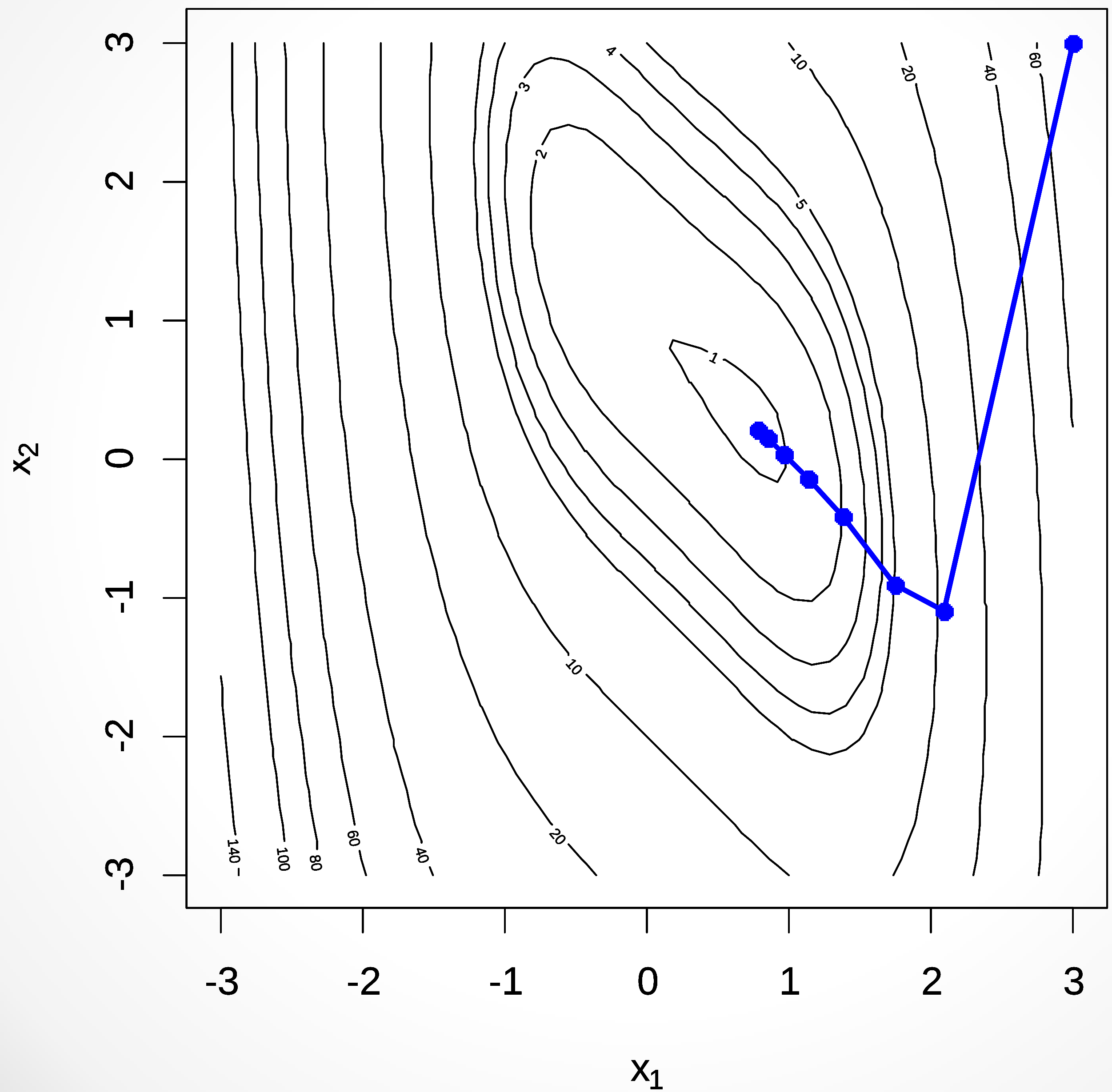
Оптимизационный цикл

```
while (sum(gf^2)^0.5 > grad.tol) {
  k <- k + 1
  # направление поиска p⃗_k = −H_k∇f_k
  p <- as.vector( -H %*% cbind(gf) )
  # длина шага α_k
  alpha <- line.search(f,x,p,gf)
  # новые значения параметров и градиента
  x1 <- x + alpha*p;  gf1 <- gradf(x1)
  # величины s⃗_k, y⃗_k, ρ_k
  s <- cbind(x1 - x);  y <- cbind(gf1 - gf)
  rho <- as.vector( (t(y)%*%s) ) ^ -1
  # матрица H_{k+1}
  H1 <- (I - rho*(s%*%t(y))) %*% H %*% (I - rho*(y%*%t(s))) +
         rho*(s%*%t(s))
  # обновление параметров, градиента и обратного гессиана
  x <- x1; gf <- gf1; H <- H1
  if (rho == Inf) break
}
```

\# ответ

```
x; f(x); k
```

Процесс оптимизации



Домашнее задание

- написать функцию `line.search` для расчёта оптимальной величины изменения параметров целевой функции согласно второму условию Вольфа