

Симплексный метод Нелдера–Мида (Nelder–Mead)

Содержание

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

Теоретические основы метода

Задача:

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

Основная идея метода

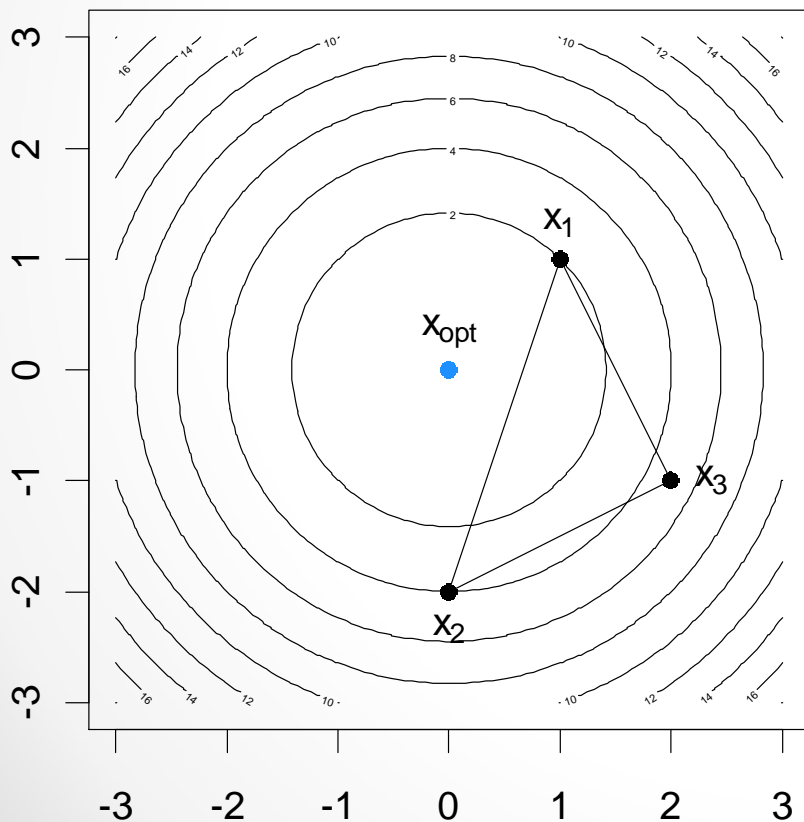
Название метода восходит к тому факту, что на каждом его этапе рассматривается $d + 1$ точка в пространстве R_d , чья выпуклая линейная комбинация образует симплекс S

На каждой итерации метода Нелдера–Мида мы стараемся заменить вершину с наибольшим значением целевой функции на другую точку, положение которой находится путём отражения, расширения или сжатия симплекса вдоль прямой, соединяющей эту вершину с центроидом остальных вершин

Одна итерация метода (1:6)

Пример: $f(\vec{x}) = x_1^2 + x_2^2$

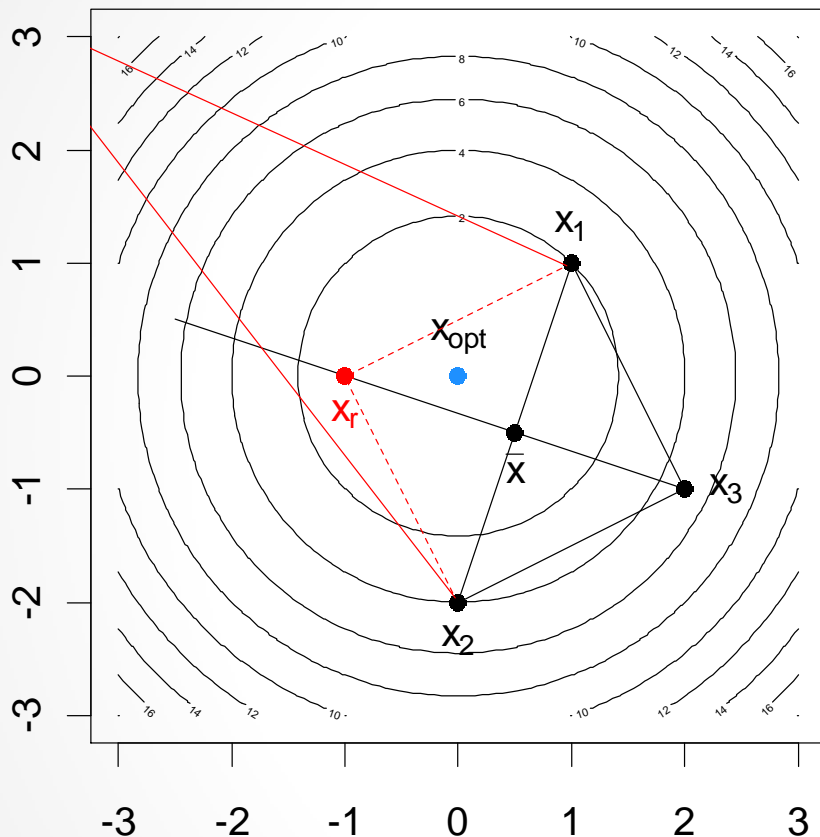
1. Рассчитать функцию f в $d+1$ вершине симплекса и отсортировать вершины, чтобы $f(\vec{x}_1) \leq f(\vec{x}_2) \leq \dots \leq f(\vec{x}_{d+1})$



Одна итерация метода (2:6)

2. Найти «отражённую» точку $\vec{x}_r = \bar{x} + \alpha(\bar{x} - \vec{x}_{d+1})$, где

$\bar{x} = \frac{1}{d} \sum_{i=1}^d \vec{x}_i$ — центроид
 d наилучших вершин

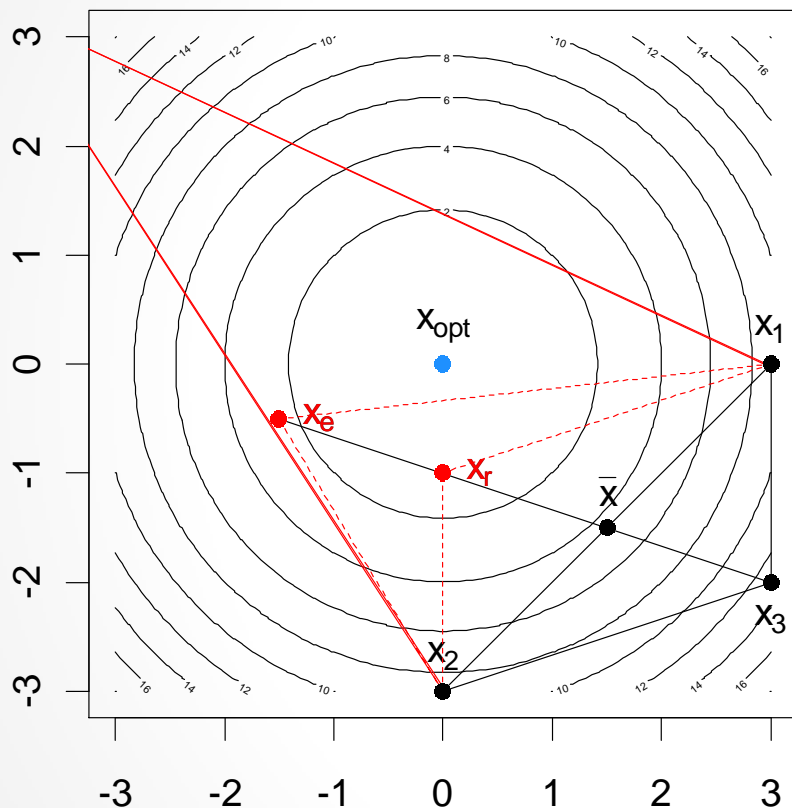


Если $f(\vec{x}_1) \leq f(\vec{x}_r) < f(\vec{x}_d)$, то заменить худшую вершину \vec{x}_{d+1} на \vec{x}_r и перейти к следующей итерации

Одна итерация метода (3:6)

3. Если $f(\vec{x}_r) < f(\vec{x}_1)$, то продвинуться далее в этом направлении, найдя «продвинутую» точку

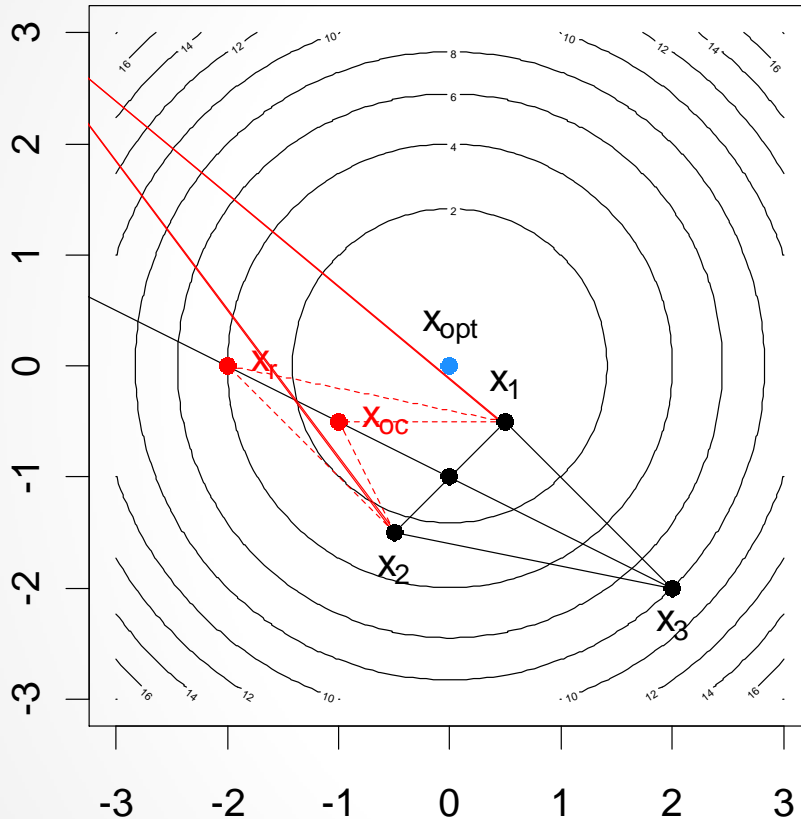
$$\vec{x}_e = \vec{x} + \beta(\vec{x}_r - \vec{x})$$



Если $f(\vec{x}_e) < f(\vec{x}_1)$, то заменить \vec{x}_{d+1} на \vec{x}_e , иначе заменить \vec{x}_{d+1} на \vec{x}_r ; перейти к следующей итерации

Одна итерация метода (4:6)

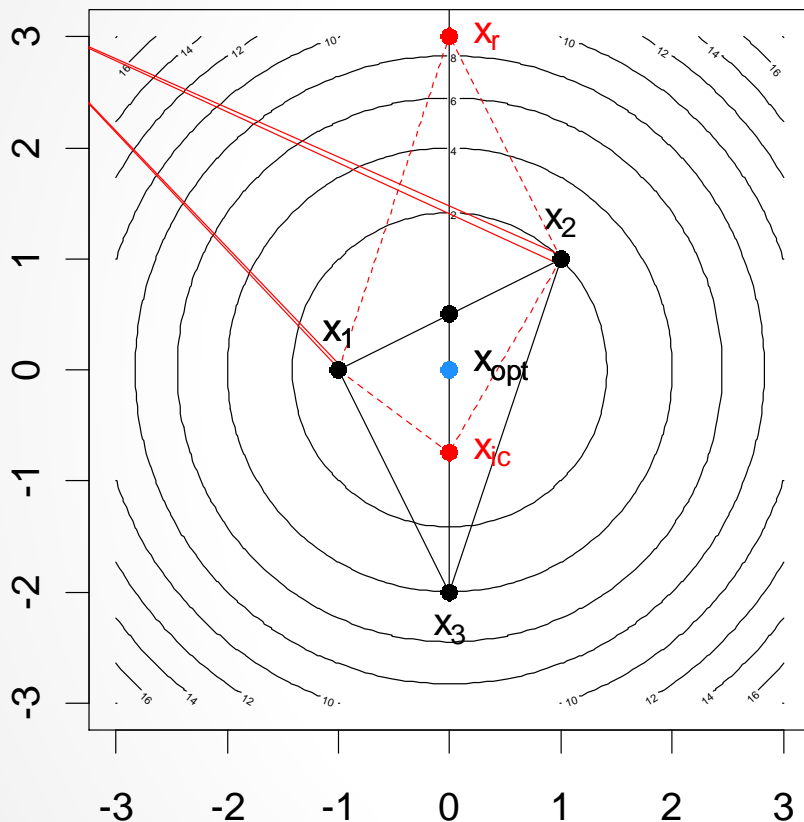
4. Если $f(\vec{x}_d) \leq f(\vec{x}_r) < f(\vec{x}_{d+1})$, то осуществить «внешнее сжатие»: $\vec{x}_{oc} = \vec{x} + \gamma(\vec{x}_r - \vec{x})$



Если $f(\vec{x}_{oc}) \leq f(\vec{x}_r)$, то заменить \vec{x}_{d+1} на \vec{x}_{oc} , иначе осуществить «сокращение»

Одна итерация метода (5:6)

5. Если $f(\vec{x}_r) \geq f(\vec{x}_{d+1})$, то осуществить «внутреннее сжатие»:
 $\vec{x}_{ic} = \vec{x} - \gamma(\vec{x}_r - \vec{x})$

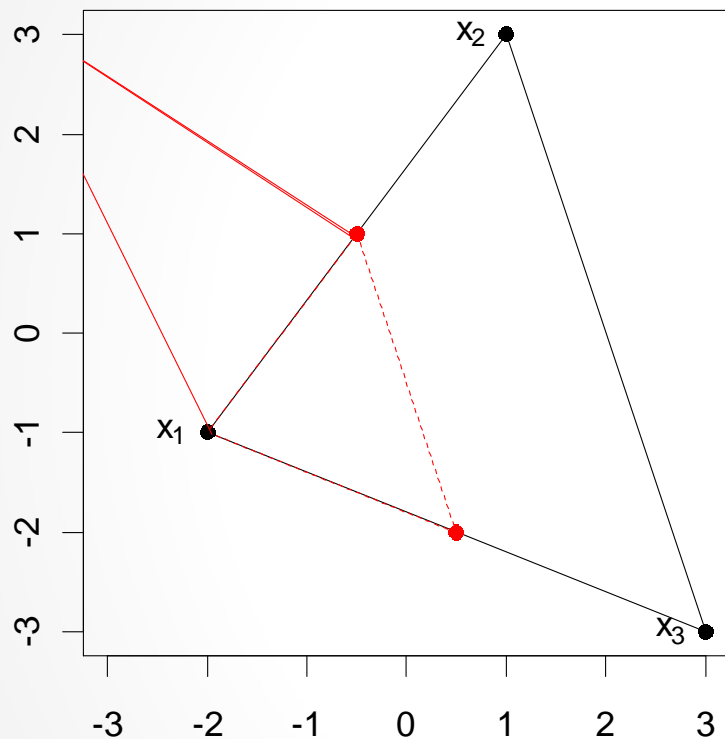


Если $f(\vec{x}_{ic}) < f(\vec{x}_{d+1})$, то заменить \vec{x}_{d+1} на \vec{x}_{ic} , иначе осуществить «сокращение»

Одна итерация метода (6:6)

6. «Сокращение»

$$\forall i \in \{2; \dots; d + 1\} \quad \vec{x}_i := \vec{x}_1 + \delta(\vec{x}_i - \vec{x}_1)$$



$$\{\alpha; \beta; \gamma; \delta\} = \left\{1; 2; \frac{1}{2}; \frac{1}{2}\right\} \quad (\text{Nelder, Mead, 1965})$$

$$\{\alpha; \beta; \gamma; \delta\} = \left\{1; \frac{d+2}{d}; \frac{3d-2}{4d}; \frac{d-1}{d}\right\} \quad (\text{Gao, Han, 2010})$$

Начальный симплекс

Построение начального симплекса на основе заданного вектора параметров \vec{x}_0 :

$$S_0 = S(\vec{x}_0, \vec{x}_1, \dots, \vec{x}_d)$$

$$\forall i \in \{1; \dots; d\} \quad \vec{x}_i = \vec{x}_0 + \tau_i \vec{e}_i, \text{ где}$$

\vec{e}_i — единичный вектор с единицей на i -м месте,

$$\tau_i = \begin{cases} 5 \cdot 10^{-2}, & x_{0,i} \neq 0 \\ 2.5 \cdot 10^{-4}, & x_{0,i} = 0 \end{cases}$$

Литература

- Nocedal J., Wright S. Numerical Optimization. Springer Science+Business Media: N.Y., 2006. Chapter 9, pp. 238–240
- Gao F., Han L. Implementing the Nelder-Mead simplex algorithm with adaptive parameters. Computational Optimization and Applications, Vol. 51, № 1, 2012, pp. 259–277
- Nelder J.A., Mead R. A simplex method for function minimization. The Computer Journal, 1965, № 8, pp. 308–313

Пример практической реализации в «R»

Задача:

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

Целевая и вспомогательные функции

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

функция для определения начального симплекса

```
init.simplex <- function(x0) {  
# матрица nrow == length(x0) + 1  
# по строкам - вершины симплекса  
}
```

сортировка вершин по значению целевой функции

```
sort.vertex <- function(simplex,fn) {  
# список из двух элементов: vertex и vertex.value  
# vertex - матрица вершин симплекса, отсортированных по  
#           возрастанию значений функции на них  
# vertex.value - вектор значений функций на вершинах, отсортированный  
#               по возрастанию  
}
```

Начальные значения и параметры ОПТИМИЗАЦИИ

начальная точка и начальный симплекс

```
x0 <- c(2.5, 2.5)
z <- sort.vertex(init.simplex(x0), f)
s <- z$vertex; vert.value <- z$value
```

параметры оптимизации

```
d <- length(x0)
nm.par <- c(1, (d+2)/d, (3*d-2)/(4*d), (d-1)/d)
names(nm.par) <- c("alpha", "beta", "gamma", "delta")
val <- mean(vert.value)
delta.val <- Inf; delta <- 10^-12
```

Оптимизационный цикл

```
while (delta.val>=delta) {  
  
  tmp <- val # сохраняем предыдущее значение  
  # центроид наилучших вершин  
  x.bar <- apply(s[1:d,],2,mean)  
  # отражённая точка  
  x.r <- x.bar + nm.par["alpha"]*(x.bar-s[d+1,])  
  f.r <- f(x.r)  
  
  shrunked <- FALSE # индикатор, было ли сокращение симплекса  
  # проверка условий 2. и 3.  
  if ((f.r>=vert.value[1]) & (f.r<vert.value[d])) {  
    s[d+1,] <- x.r; f.new.vert <- f.r  
  } else { if (f.r<vert.value[1]) {  
    # продвинутая точка  
    x.e <- x.bar + nm.par["beta"]*(x.r-x.bar)  
    f.e <- f(x.e)  
    if (f.e<vert.value[1]) {  
      s[d+1,] <- x.e; f.new.vert <- f.e  
    } else {  
      s[d+1,] <- x.r; f.new.vert <- f.r  
    }  
  }  
}}
```

• продолжение на следующем слайде •

начало на предыдущем слайде

```
# проверка условия 4.
if ((f.r>=vert.value[d]) & (f.r<vert.value[d+1])) {
  # внешнее сжатие
  x.oc <- x.bar + nm.par["gamma"]*(x.r-x.bar)
  f.oc <- f(x.oc)
  if (f.oc<=f.r) {
    s[d+1,] <- x.oc; f.new.vert <- f.oc
  } else {
    # сокращение симплекса
    for (i in 2:(d+1))
      s[i,] <- s[1,]+nm.par["delta"]*(s[i,]-s[1,])
    shrunked <- TRUE
  }
}
```

продолжение на следующем слайде

начало на предыдущем слайде

проверка условия 5.

```
if (f.r>=vert.value[d+1]) {  
  # внутреннее сжатие  
  x.ic <- x.bar - nm.par["gamma"]*(x.r-x.bar)  
  f.ic <- f(x.ic)  
  if (f.ic<vert.value[d+1]) {  
    s[d+1,] <- x.ic; f.new.vert <- f.ic  
  } else {  
    # сокращение симплекса  
    for (i in 2:(d+1))  
      s[i,] <- s[1,]+nm.par["delta"]*(s[i,]-s[1,])  
    shrunked <- TRUE  
  }  
}
```

переоценка значений функции на вершинах, пересортировка симплекса

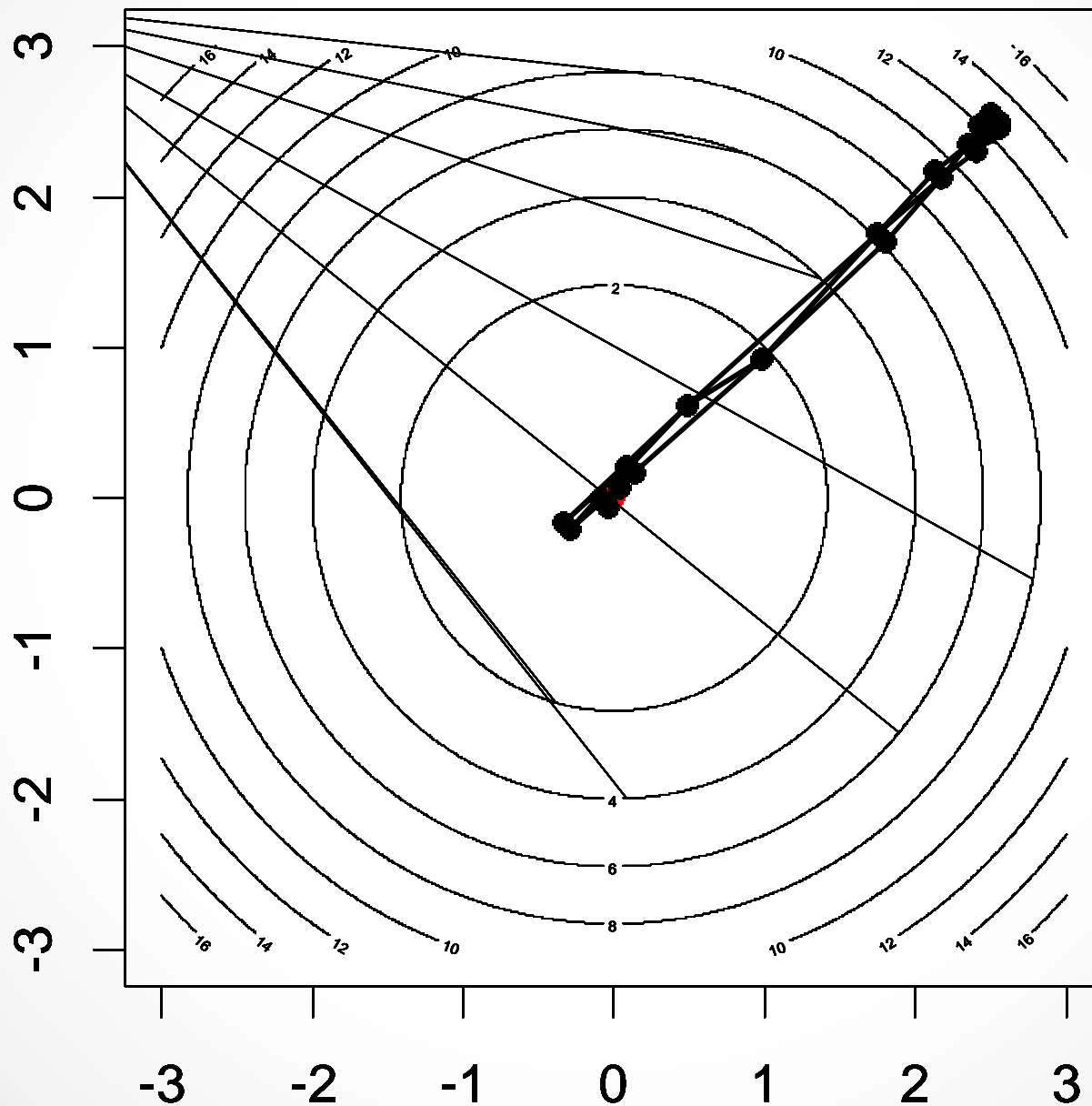
```
if (shrunked) {  
  z <- sort.vertex(s,f)  
  s <- z$simplex; vert.value <- z$value  
} else {  
  vert.value[d+1] <- f.new.vert  
  z <- order(vert.value)  
  s <- s[z,]; vert.value <- vert.value[z]  
}
```

```
}
```

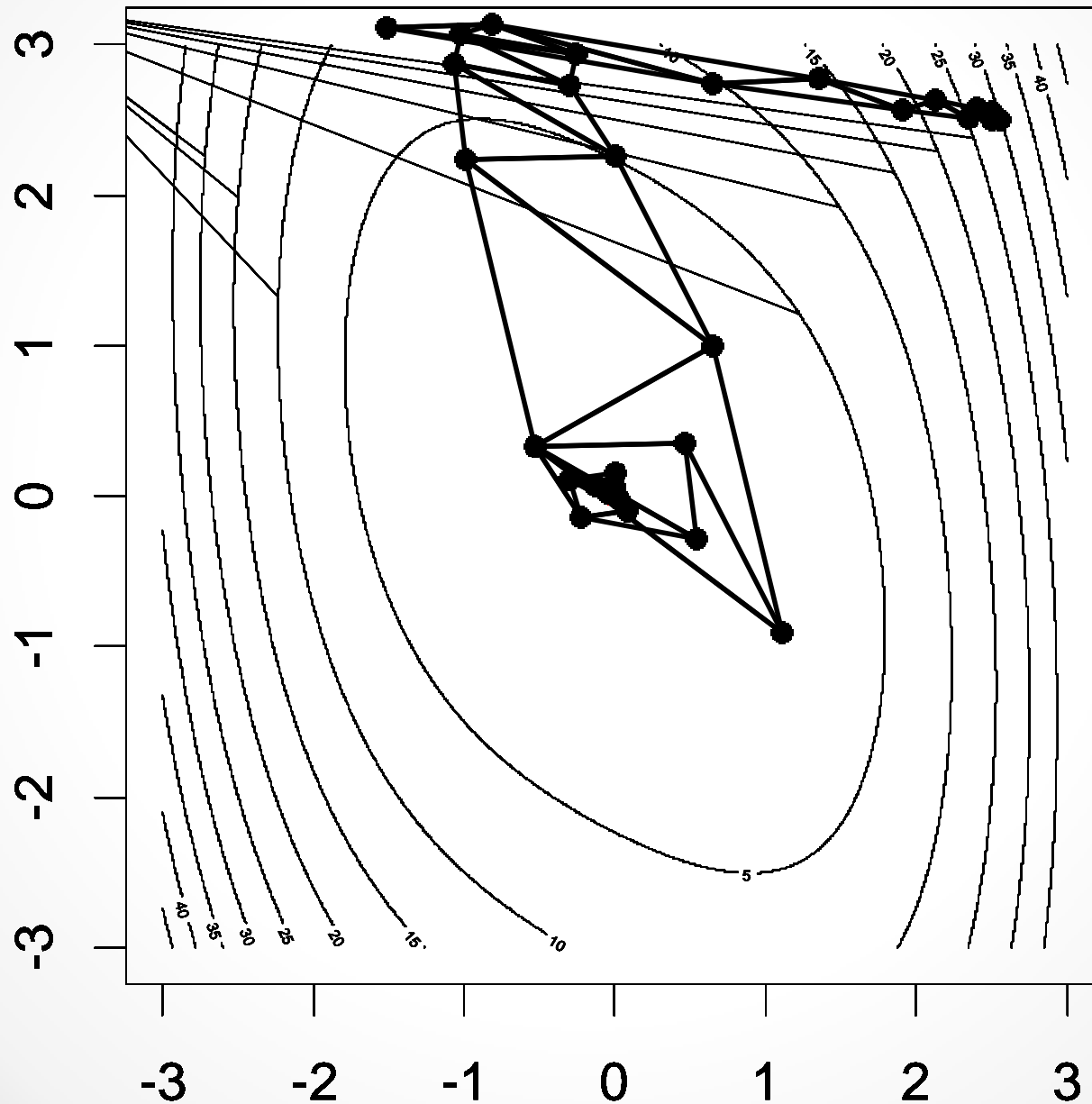
Ответ

```
opt <- s[1,]  
opt.value <- vert.value[1]
```

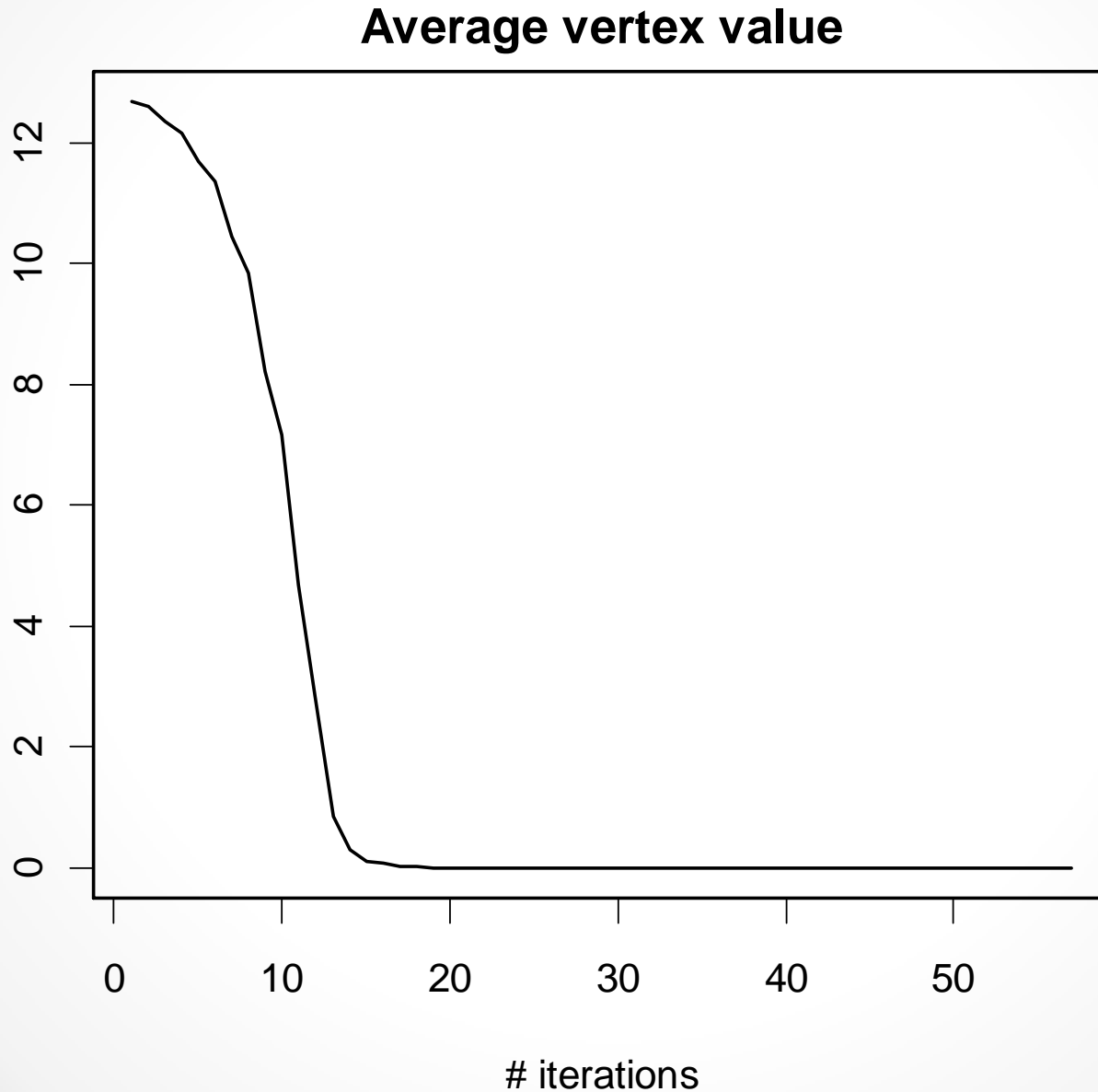
Процесс оптимизации



Ещё один процесс оптимизации



Изменение среднего значения на вершинах



Домашнее задание

- написать функцию `init.simplex` для задания начального множества вершин
- написать функцию `sort.vertex` для перекалибровки вершин и расчёта значений минимизируемой функции на них