222800-1358 Nieklasyczne Metody Optymalizacji **Letni 2022/2023**

Ćwiczenia 2: Metoda Spadku Newtona

Daniel Kaszyński *7-14 Marca 2023 r.*

2.1 Spadek Newtona (Newton Descent)

Wzór iteracyjny metody Spadku Newtona jest następujący:

$$x_{k+1} = x_k - \mathrm{H}_f^{-1}(x_k)\nabla_f(x_k)$$

Skąd to wiemy? Rozważmy wzór aproksymacyjny Taylora:

$$f(x+h) \approx f(x) + \nabla_f(x)h + \frac{1}{2}h^T H_f(x)h$$

oznacza to, że dla FOC ze względu na parametr h, tj. $\frac{df}{dh}(x+h)$ mamy

$$\frac{df}{dh}(x+h) = \nabla_f(x) + H_f(x)h = 0$$

$$h = -H_f(x)^{-1}\nabla_f(x)$$

Oznacza to, że optymalne przesunięcie z punktu x_k to wektor $-\mathrm{H}_f^{-1}(x_k)\nabla_f(x_k)$. Zwróćmy uwagę, że h minimalizuje wyrażenie przybliżenia wielomianu drugiego stopnia.

Zauważmy, że dla wielomianów drugiego stopnia (przyjmijmy na chwilę przypadek jednowymiarowy $f : \mathrm{R} \to \mathrm{R}$, metoda Newtona zbiega w jednej iteracji!

$$f(x) = ax^2 + bx + c?$$

$$x^{(2)} = x^{(1)} - \frac{2x^{(1)}a + b}{2a} = x^{(1)} - x^{(1)} - \frac{b}{2a} = -\frac{b}{2a}$$

Metoda Newtona przybliża wykres funkcji f wielomianem drugiego stopnia. Dla zadanego przybliżenia znajdowane jest ekstremum, do którego przechodzimy – przyjmujemy podejście iteracyjne.

Do wykorzystania metody Newtona potrzebna jest znajomość macierzy Hesianu w dowolnym punkcie f. W tym celu stosujemy przybliżenie numeryczne, którego osiągane jest przy pomocy poniższej funkcji:

```
num_hessian <- function(f, x, h=10^-3){
  #' Numerical hessian
  #'
  #' @description Numerical (central) hessian calculation
  #'
  #' @param f function that is subject to be calculate hessian
  #' @param x point at with the gradient is calculated (must be compliant with
  #' f function).
  #' @param h finite difference (numerical perturbation scalar).
  #'
  #' @return hessian matrix of f.
```

```
12
13    n <-length(x)
14    H <- matrix(NA, nrow=n, ncol=n)
15    e <- diag(n)
16    for(i in 1:n){
17      for(j in 1:n){
18        if(i>j){
19          H[i,j] <- H[j,i]
20        }else{
21          H[i,j] <- (f(x+e[i,]*h+e[,j]*h)-f(x-e[i,]*h+e[,j]*h)
22                     -f(x+e[i,]*h-e[,j]*h)+f(x-e[i,]*h-e[,j]*h)) / (4*h^2)
23        }
24      }
25    }
26    return(H)
27  }
```

Listing 1: Implementacja algorytmu spadku gradientowego

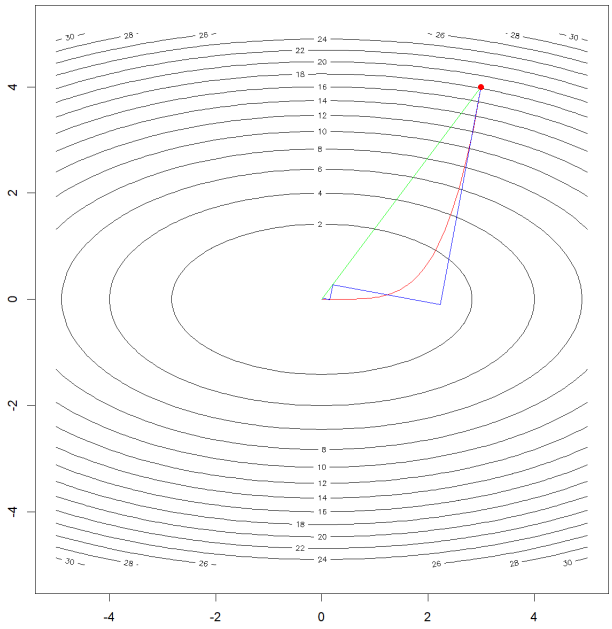
Poniżej przedstawiono przykładową implementację metody Newtona.

```
1  source('num_grad.R')
2  source('num_hessian.R')
3
4  newton_descent <- function(f, x, max_iter=100){
5    #' Newton Descent solver
6    #'
7    #' @description Newton Descent Solver (for R^n -> R functions)
8    #'
9    #' @param f objective function that is subject to be optimize.
10   #' @param x initial solution that is scalar/vector (must be compliant with
11   #' f function).
12   #' @param max_iter maximum number of iterations used in the solver.
13   #'
14   #' @return list with the optimization results.
15
16   start_t <- Sys.time()
17   n<-length(x)
18   out <- list(x_opt=x,
19               f_opt=f(x),
20               x_hist=matrix(NA, nrow=max_iter, ncol=n),
21               f_hist=rep(NA, max_iter),
22               a=rep(NA, max_iter),
23               t_eval=NA)
24
25   # Assign first solution as a x0 imputed to the function
26   out$x_hist[1,] <- x
27   out$f_hist[1] <- f(x)
28   out$a[1] <- NA
29
30   xk <- x
31   for(k in 2:max_iter){
32     G <- num_grad(f, xk)
33     H <- num_hessian(f, xk)
34     if(abs(det(H)) < 10^-3) H <- H+diag(n)*10^-3
35
36     xk <- xk - solve(H)%*%G
37     if(f(xk) < f(x)){
38       x <- xk
39     }
40     out$x_hist[k,] <- xk
41     out$f_hist[k] <- f(xk)
42   }
43
```

```
  # Assign optimal solution to output
  out$x_opt <- x
  out$f_opt <- f(x)
  out$t_eval <- Sys.time() - start_t
  return(out)
}
```

Listing 2: Kod wykorzystany do porównania działania algorytmu GD

Poniżej przedstawiono porównanie metod: Spadek Gradientowy, Najszybszy Spadek, oraz Spadek Newtona.

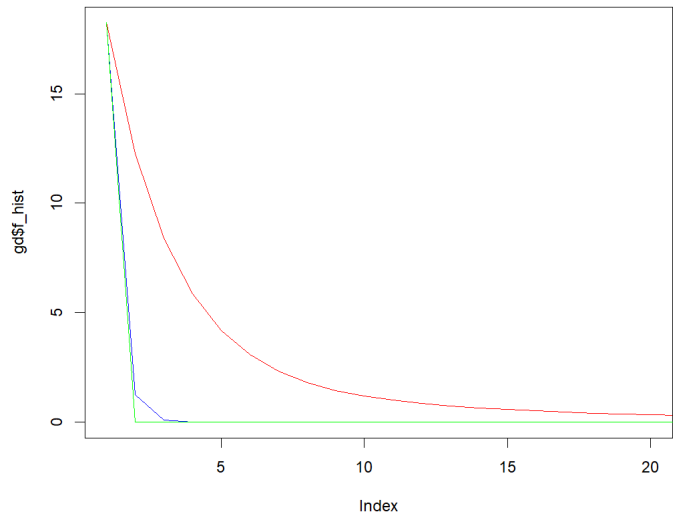


Rysunek 2.1: Trajektoria wektora x w kolejnych iteracjach algorytmów GD, SD, ND

Poniżej porównano wartości funkcji celu, dla poszczególnych solwerów optymalizacyjnych.

Bibliografia

[KW19] M. Kochenderfer and T. Wheeler, "Algorithms for optimization, "*Mit Press*, 2019.



Rysunek 2.2: Wartość funkcji celu w kolejnych iteracjach algorytmów GD, SD i ND SD