Санкт-Петербургский политехнический университет Петра Великого
Институт металлургии, машиностроения и транспорта
Кафедра «Мехатроника и роботостроение (при ЦНИИ РТК)»

КУРСОВАЯ РАБОТА

«Метод градиентного спуска»

по дисциплине «Программирование на языках высокого уровня»

Выполнила
Студентка гр.33335/2 А.Ю. Волкова

Руководитель М.С. Ананьевский

Санкт-Петербург

2018г.

Оглавление

1. Введение

1.1 Задача, которую решает алгоритм

Градиентный спуск — метод нахождения локального экстремума (минимума или максимума) функции с помощью движения вдоль градиента. Алгоритм данного метода позволяет решить задачу поиска минимума функции $f(x)\colon \mathbb{R}^n \to \mathbb{R}$,которая записывается в виде:

$$\underset{x}{argmin} f(x)$$

где функция $f(x)$ такова, что можно вычислить ее градиент.

1.2 Описание алгоритма

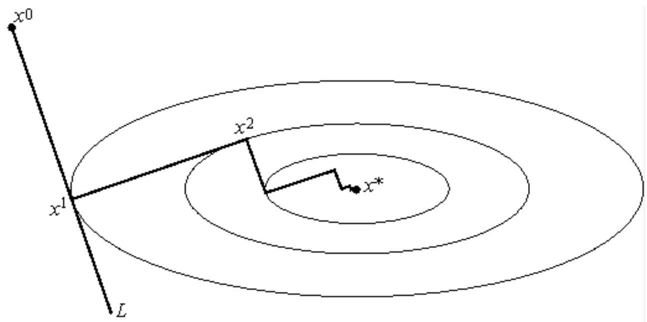
Основная идея:

Метод заключается в том, чтобы осуществлять оптимизацию в направлении наискорейшего спуска, а это направление задаётся антиградиентом $-\nabla f$:

$$x^{[k+1]} = x^{[k]} - \lambda^{[k]} \nabla f(x^{[k]})$$

где $\lambda^{[k]}$ выбирается наискорейшим спуском:

$$\lambda^{[k]} = \underset{\lambda}{\mathrm{argmin}}\, f(x^{[k]} - \lambda \nabla f(x^{[k]}))$$



Алгоритм:

Вход: функция $f\colon \mathbb{R}^n \to \mathbb{R}$

Выход: найденная точка оптимума x с приближенной точностью до ε

1. Задаем функцию f(x,y).
2. Находим частную производную функции f(x,y) по x.
3. Находим частную производную функции f(x,y) по y.
4. Задаем начальное приближение ε
5. Найдем значение нормы (длины) вектора по формуле:

6. Для метода наискорейшего спуска задаем функцию g как:

$$f\left(x-\lambda\cdot\frac{df(x,y)}{dx}, y-\lambda\cdot\frac{df(x,y)}{dy}\right)$$

7. Используем метод половинного деления для нахождения аргминимума **λ** в градиентном спуске. Данный метод следует использовать, если рассматриваемая функция непрерывна на заданном промежутке. Суть метода: пусть известно, что на *k*-м шаге последовательного поиска $x \in [a_k;b_k]$. На отрезке $[a_k, b_k]$ длиной l_k выберем две точки $x_{k1}=(a_k+b_k-\delta)/2$ и $x_{k2}=(a_k+b_k+\delta)/2$, где $\delta>0$ — некоторое достаточно малое число. Необходимо учитывать, что выполнение неравенства $l_{k+1}<\varepsilon$ (где l_{k+1} – длина отрезка на k+1 шаге), означающее достижение заданной точности нахождения точки, возможно лишь при условии выбора $2\delta\leq\varepsilon$.

8. Вычислим значения $f(x_{k1})$ и $f(x_{k2})$ функции f(x) в этих точках и выполним *процедуру исключения отрезка*.
   Если $f(x_{k1}) < f(x_{k2})$, то имеем $x \in [a_k;x_{k2}]$, а отрезок $[x_{k2},b_k]$ исключаем из дальнейшего рассмотрения.
   Наоборот, если $f(x_{k1}) \geq f(x_{k2})$, то $x \in [x_{k1};b_k]$, а отрезок $[a_k,x_{k1}]$ не рассматриваем. В результате получим новый отрезок $[a_{k+1};b_{k+1}] \subset [a_k;b_k]$.

9. Переходим к методу наискорейшего спуска, задав начальное приближение bx, by.

10. Находим $\lambda^{[k]}$ как минимум функции g.

11. Выполняем $x^{[k+1]}=x^{[k]}-\lambda^{[k]}\nabla f(x^{[k]})$

$$y^{[k+1]}=y^{[k]}-\lambda^{[k]}\nabla f(y^{[k]})$$

12. Проверяем условие остановки: следует прекратить вычисления, если:

$$norma(x[k+1]-x[k], y[k+1]-y[k]) < \varepsilon$$

2. Реализация алгоритма на языке С

```
#include <stdio.h>
#include <math.h>
#ifndef GRADIENTSP_H
#define GRADIENTSP_H
#define NMAX 10
//Собственно здесь записывается наша функция
double f(double x,double y)
{
   return (-4. * x + x*x - y - x * y + y * y);
}
//Это первая производная по dx
double f_dx(double x,double y)
{
   return (-4+2.*x-y);
}
//Это первая производная по dy
double f_dy(double x,double y)
{
   return (-1.-x+2.*y);
}

//двумерная норма
double norma(double x, double y)
{
   return sqrt(x*x+y*y);
}
//Это функция g в методе наискорейшего (градиентного) спуска
double g(double x, double y, double lambda)
{
   return f(x - lambda*f_dx(x,y), y - lambda*f_dy(x,y));
}
//Метод половинного деления для нахождения минимума в градиентном спуске
double Dihotomia(double a0, double b0, double epsilon, double x, double y)
{
   //Номер шага
   int k;
   //Отклонени от середины отрезка влево, вправо
```

```
    double lk, mk;
    //Величина, на которую мы отклонимся от середины отрезка
    double delta=0.5*epsilon;
    //Точка минимума
    double x_;
    //Отрезок локализации минимума
    double ak=a0, bk=b0;
    k=1;
    //Пока длина отрезка больше заданной точности
    double ff;
    do
    {
        //Берем середину
        lk=(ak+bk-delta)/2;
        mk=(ak+bk+delta)/2;
        k++;
        //Проверяем в какую часть попадает точка минимума слева от разбиения
или справа и выбираем соответствующую точку
        if(g(x,y,lk)<=g(x,y,mk))
        {
            //Теперь правая граница отрезка локализации равна mk
            bk=mk;
        }
        else
        {
            //Теперь левая граница отрезка локализации равна mk
            ak=lk;
        }
    } while ((bk-ak) >= epsilon);
    x_=(ak+bk)/2; //minimum point
    return x_;
}
// метод наискорейшего спуска
double GreatDescent(int bx, int by,double epsilon)
{
    double x[NMAX];
    double y[NMAX];
    double lambda[NMAX];
    int k;
```

```
    //Начальное приближение u[0]
    x[0]=bx;
    y[0]=by;
    printf("Results: (%f, %f) \n", x[0], y[0]);
    for (k=0; ; k++)
    {
        //Находим lambda_k как минимум функции g на отрезке -10000,100000
        lambda[k]=Dihotomia(-10000,100000,epsilon,x[k],y[k]);
        //Вычисляем u[k]
        x[k+1]=x[k]-lambda[k]*f_dx(x[k], y[k]);
        y[k+1]=y[k]-lambda[k]*f_dy(x[k], y[k]);
        if (k>1)
        {
            //Проверяем условие остановки
            if(norma(x[k+1]-x[k],y[k+1]-y[k])<epsilon)
            {
                break;
            }
        }
    }
    printf("\n Min: (epsilon = %f), f(x(k+1), y(k+1)) = (%f, %f)", epsilon, x[k+1],
y[k+1] );
    return f(x[k+1],y[k+1]);
}
#endif // GRADIENTSP_H


int main()
{
    double bx = 0;
    double by = 4;
    double epsilon = 0.001;
    printf("\n %f", GreatDescent(bx,by,epsilon));
    return 0;
}
```

3. Анализ алгоритма

Произведем оценку времени работы алгоритма.

Запустим программу. Время работы данного алгоритма 0,003674с

4. Применение

Алгоритм данного метода позволяет решить задачу поиска минимума функции $f(x): \mathbb{R}^n \to \mathbb{R}$,которая записывается в виде:

$$\underset{x}{argmin} f(x)$$

где функция $f(x)$ такова, что можно вычислить ее градиент.

Список литературы

1. Акулич И.Л. Математическое программирование в примерах и задачах: Учеб. пособие для студентов эконом. спец. вузов. — М.: Высш. шк., 1986.
2. Гилл Ф., Мюррей У., Райт М. Практическая оптимизация. Пер. с англ. — М.: Мир, 1985.
3. Коршунов Ю.М., Коршунов Ю.М. Математические основы кибернетики. — М.: Энергоатомиздат, 1972.