Міністерство освіти і науки України НТУУ «Київський політехнічний інститут ім. Ігоря Сікорського»
Фізико-технічний інститут

# Розрахункова робота

## з методів оптимізації

## на тему: «Метод проекції градієнту»

Виконали:
студенти 4 курсу
ФТІ групи ФІ-52
Бурлака Марія
Єршов Степан
Кратт Ярослав
Овчарова Марина

Перевірив:
Данилов В. Я.

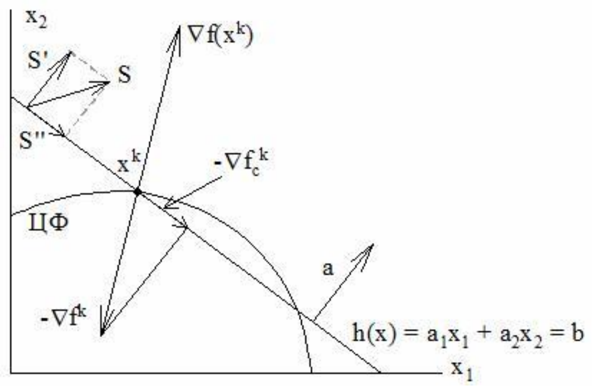
Київ
2018

# Метод проекції градієнту

Розглянемо задачу оптимізації при єдиному лінійному обмеження у вигляді рівності
$f(x) \to \min$, (1)

$$h(x) = \sum_{j=1}^{n} a_j x_j = a^T x = b$$
. (2)

В заданій точці $x^k$, в якій $\nabla f(x^k) \neq 0$, робиться спроба визначити напрям пошуку, яке б лежало на поверхні обмеження і було напрямом спуску. Такий напрям можна отримати геометрично ортогонально проектуючий вектор, протилежний $\nabla f(x^k)$ на поверхню обмеження (див. Мал.1).




Малюнок 1
Тут $\nabla f_c^k$ - проекція антиградієнта на поверхню обмежень, яка призводить в допустимі точки. Дійсно, для $\forall\alpha \geq 0$ точки, задані співвідношенням
$x = x^k - \alpha \nabla f_c^k$ (3)
задовольняють лінійному обмеження

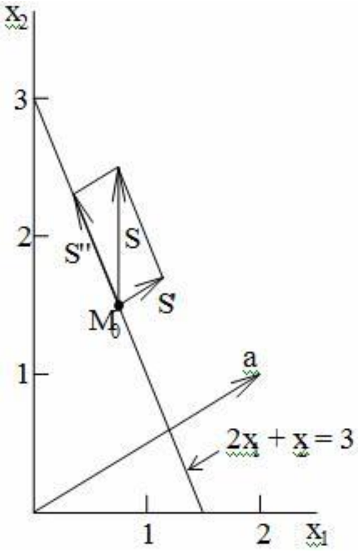
$$(a^T x = \underbrace{a^T x^k}_{=b} - \alpha \underbrace{a^T \nabla f_c^k}_{=0,\ т.к. вектор\ a \perp \nabla f_c})$$
.

Цей напрямок задає спуск, т. К. Кут між $\nabla f^k$ і $\nabla f_c^k$ більше 90 °. Процес ортогонального проектування полягає в розкладанні вектора на дві ортогональні компоненти: паралельну поверхні, заданої обмеженням, і перпендикулярну до неї. Паралельна компонента є шуканої проекцією градієнта.
Нехай вектор a - нормаль до поверхні обмеження. Відзначимо, що з виразу $a^T S = 0$ слід допустимість напрямки, що задається вектором S ( S паралельно поверхні).
Т. о. всі вектори, перпендикулярні до поверхні обмеження повинні бути паралельні до a. Отже, для будь-якого вектора S його компонента S ', перпендикулярна до поверхні обмеження, дорівнює значенню a, помноженому на константу.
Позначимо через S "компоненту S, паралельну поверхні обмеження. Тоді S "задовольняє співвідношенню
$a^T S'' = 0$. (4)
Тому будь-який вектор можна представити у вигляді векторної суми:
$S = S' + S''$, (5)
де $S' = \lambda a$, а $S''$ задовольняє рівняння $a^T S'' = 0$.
Знайдемо λ. Розглянемо скалярний твір a TS. В силу (5) і (4) маємо
$a^T S = a^T \lambda a + a^T S'' = \lambda a^T a$, (6)
звідки $\lambda = (a^T a)^{-1} a^T S$. (7)
З (5) знайдемо S "= SS '= S- λa. Підставами сюди (7), отримаємо
$S'' = Sa (a^T a)^{-1} a^T S = (I - a (a^T a)^{-1} a^T) S$, (8)

де I - одинична матриця, порядок якої узгоджений з S . Матриця P = I- a (a $^T$ a) -1, a $^T$ - проекційна матриця. Вона проектує вектор S на площину, що задається обмеженням h (x).
Відзначимо, що P є симетричної і позитивно полуопределенной. Симетричність P очевидна. Для доказу позитивної полуопределенности розглянемо твір y T Py для довільного y ≠ 0. Тоді

$$y^T Py = y^T [I - a(a^T a)^{-1} a^T ]y = y^T y - \frac{(y^T a)(a^T y)}{a^T a} = \frac{(y^T y)(a^T a) - (y^T a)^2}{a^T a}.$$

Малюнок 2




Використовуючи нерівність Шварца (y $^T$ a) 2 ≤ (y $^T$ y) (a $^T$ a), переконуємося, що чисельник невід'ємний.
властивості проекцій
Нехай S "= - P ∇ f ($x^k$) - напрямок спуску.
Якщо S "= 0, то точка x k відповідає неодмінним умовам Лагранжа. Вектор множників Лагранжа задається виразом
λ = ($AA^T$) -1 A ∇ f. (9)
Перше твердження випливає з того, що матриця P симетрична і позитивно піввизначена. Дійсно, маємо
$x^k$ +1 = $x^k$ -αP ∇ f ($x^k$); α≥0;
f ($x^k$ + 1 ) = f ($x^k$ - αP ∇ f ($x^k$)) = f ($x^k$) - α ∇ $f^T$ ($x^k$ + 1 ) P ∇ f ($x^k$)

Так як $\begin{cases} \nabla f^T P \nabla f \geq 0 \\ \alpha \geq 0 \end{cases}$, То f k +1 ≤f k , що й треба було довести.
Маємо S "= - P ∇ f, т. Е. S" - це проекція вектора ∇ f на поверхню обмежень. Тому, якщо S "= 0, значить градієнт ∇ f перпендикулярний поверхні обмежень.
Розглянемо друга властивість. Оскільки S "= - P ∇ f, то твір ∇ f $^T$ S" = - ∇ f $^T$ P ∇ f≤0.
Якщо S "= - P ∇ f = 0, то отже ∇ f∟ S", т. Е. ∇ f перпендикулярний поверхні обмежень. Тоді з формули слідує ∇ f = A $^T$ λ. (10) Оскільки рядками матриці A є вектори коефіцієнтів в лінійних обмеженнях, то (10) являє собою іншу форму запису необхідної умови Лагранжа, де $a_k$,
S=S'+S"=$A^T$λ+S"=0

$$\nabla f - \sum_{k=1}^{m} \lambda_k a_k = 0 \quad L = f(x) - \sum_{k=1}^{m} h_k(x) \lambda_k$$

- k-й вектор обмежень. Звідси ми бачимо, що $\lambda_k$ - це множники Лагранжа, т. Е. Множники функції L.
Необхідна умова екстремуму:

$$\frac{\partial L}{\partial x_i} = 0 \Rightarrow \nabla f - \sum \lambda_k \nabla h_k = 0$$
;

∇ $h_k$ = $a_k$ .
Неважко отримати вираз для λ: S = S '+ S ";

$$AS = AA^T \lambda + \underbrace{AS''}_{=0} \Rightarrow \lambda = (AA^T)^{-1} AS = (AA^T)^{-1} A \nabla f$$

тобто λ = ($AA^T$) $^{-1}$ A ∇ f, (12)
де (AA T ) -1 існує тільки в тому випадку, якщо значення $a_k$ лінійно незалежні. Якщо є лінійно залежні обмеження, їх слід виключити з розгляду.

**Основний алгоритм проекцій градієнта**

Обчислити матрицю $P = IA^T(AA^T)$ -1 A в припущенні, що вектори $a_k$ лінійно незалежні.
Задати ε> 0- похибка збіжності. Нехай знайдена допустима точка $x^k$.
**Крок 1**. Обчислити $S_k = -P \nabla f(x^k)$.

**Крок 2**. Якщо $| S_k | \le \varepsilon$, то обчислити λ за формулою (9) і закінчити обчислення. В іншому випадку - продовжити обчислення. Перехід на Крок 3.

**Крок 3.** Визначити максимальну довжину кроку:

$$\lambda_{\max} = \min \{\max \ (0, \frac{b_k - a_k^T x^k}{a_k^T S^k})$$

$$\text{или } \infty, \text{ если } a_k^T \cdot S^k = 0); \quad k = 1, \ldots, m\}.$$

**Крок 4.** Вирішити задачу одновимірного пошуку:
$f(x\ k + \alpha S_k) \to \min; 0 \le \alpha \le \alpha \max$ .

**Крок 5.** Покласти $x\ k+1 = x\ k + \alpha\ S\ k \to$ Крок 1.

**Зауваження.** Ми розглянули алгоритм, в якому використовуються лінійні рівності. Але його можна легко поширити на нерівності, використовуючи як додаткові змінні, так і активні обмеження, що краще, так як другий спосіб дозволяє зменшити розмірність.
При використанні другого способу з допомогою одержуваної по формулі (12) оцінки множників Лагранжа здійснюється почергове виключення обмежень з безлічі активних обмежень. Модифікація виглядає наступним чином. В заданій точці x k для визначення активного безлічі перевіряються обмеження у вигляді нерівностей: $a\ j\ T\ x \ge b\ j$ , j = 1..m.
Складається матриця обмежень з рядків, відповідних активним обмеженням. Обчислюється проекційний оператор P і проекція $S_k$ . Якщо $| S_k| \le \varepsilon$, то обчислюємо множники Лагранжа:
$\lambda = (AA^T)$ -1 A $\nabla f(x^k)$.
Якщо всі $\lambda\ i \ge 0$, то рішення знайдено. В іншому випадку обмеження з найбільшим по модулю множником Лагранжа виключається з безлічі активних обмежень, обчислюється заново P і робиться перехід на Крок 1.

**Завдання:** розв'язати задачу нелінійного програмування методом проекції градієнту.

**Умови:**

1. $f(x) = 7x_1^2 - 8x_1 + 5x_2^2 \ \to min, \quad x_1^2 + (x_2 - 4)^2 \le 4$
2. $f(x) = 7x_1^2 - 8x_1 + 5x_2^2 \ \to min, \quad x_1^2 + (x_2 - 4)^2 = 4$
3. $f(x) = 7x_1^2 - 8x_1 + 5x_2^2 \ \to min, \quad x_1^2 + (x_2 - 4)^2 \ge 4$

**Розв'язання:**

```
public class GradientProjectionMethod {
   double x1 = 2, x2 = 2, am = 1, bord = 4;

   public static double function(double x1, double x2) {
      return 7*x1*x1-8*x1+5*x2*x2;
   }

   public static double derivativeFuncX1(double x1, double x2) {
      return 14*x1-8;
   }

   public static double derivativeFuncX2(double x1, double x2) {
      return 10*x2;
   }

   public static double functionIter(double x1, double x2, double a) {
      return function(x1 - a*derivativeFuncX1(x1, x2), x2 - a*derivativeFuncX2(x1, x2));
   }

   public double functionBound() {
      return x1*x1 + (x2 - 4)*(x2 - 4);
   }

   public boolean lessOrEqual() {
      boolean b = false;
         if (functionBound() <= bord) {b = true;}
      return b;
   }

   public boolean equal() {
      boolean b = false;
      if (functionBound() == bord) {b = true;}
      return b;
   }

   public boolean moreOrEqual() {
      boolean b = false;
      if (functionBound() >= bord) {b = true;}
      return b;
   }

   public void proc(boolean ind) {
      double min = 100;
      while ((Math.abs(derivativeFuncX1(x1, x2)) > 0.01)&&(Math.abs(derivativeFuncX2(x1, x2)) > 0.01)) {
         for (double a = 0; a < 1; a = a + 0.01) {
            double s = functionIter(x1 ,x2, a);
            if (min > s) {
               min = s;
               am = a;
```

```
            }
        }
        //System.out.println("min function(a) = " + min + "  for a = " + am + ",  ");
        double temp = x1;

        if (ind == true) {
            x1 = x1 - am*derivativeFuncX1(x1, x2);
            x2 = x2 - am*derivativeFuncX2(temp, x2);
        } else {
            x1 = (x1 / (Math.sqrt(x1*x1 + x2*x2))*(x1 - am*derivativeFuncX1(x1, x2)));
            x2 = (x2 / (Math.sqrt(temp*temp + x2*x2))*(x2 - am*derivativeFuncX2(temp, x2)));
        }

        //System.out.println("x = (" + x1 + ", " + x2 + ")\n");
      }
      System.out.println("___________\nmin function(a) = " + min + "  for a = " + am + ",  x = (" + x1 + ", " + (int)x2 +
"")\n");
  }

  public static void main(String[] args) {
      GradientProjectionMethod lessOrEqual = new GradientProjectionMethod();
      GradientProjectionMethod equal = new GradientProjectionMethod();
      GradientProjectionMethod moreOrEqual = new GradientProjectionMethod();
      lessOrEqual.proc(lessOrEqual.lessOrEqual());
      equal.proc(equal.equal());
      moreOrEqual.proc(moreOrEqual.moreOrEqual());
  }
}
```

```
___________
min function(a) = -2.285713646854768  for a = 0.07,  x = (0.5713210979060069, 0)

___________
min function(a) = -2.285713646854768  for a = 0.07,  x = (0.5713210979060069, 0)

___________
min function(a) = -2.2856986021068804  for a = 0.08,  x = (0.5720704, 0)
```