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

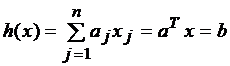
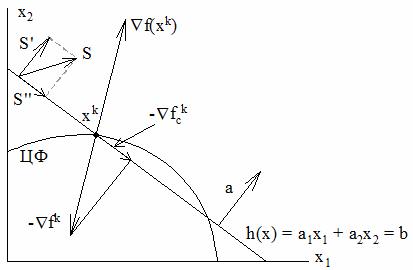
Овчарова Марина

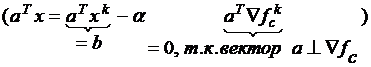
Перевірив:

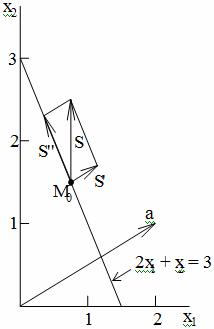
Данилов В.Я.

Київ   
2018

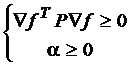
**Метод проекції градієнту**

Розглянемо задачу оптимізації при єдиному лінійному обмеження у вигляді рівності   
f(x) → min, (1)   
. (2)   
В заданій точці x k , в якій ▽ f (x k ) ≠ 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 ▽ f (xk) на поверхню обмеження (див. Мал.1).   
   
Малюнок 1

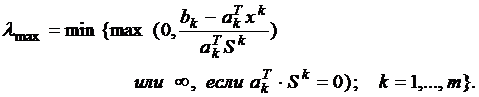
Тут ▽ fck - проекція антиградіента на поверхню обмежень, яка призводить в допустимі точки. Дійсно, для ∀α ≥ 0 точки, задані співвідношенням   
x = xk -α ▽fck  (3)   
задовольняють лінійному обмеження   
.   
Цей напрямок задає спуск, т. К. Кут між ▽ f k і ▽ fc k  більше 90 °. Процес ортогонального проектування 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 TS = 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 TS" = 0. (4)   
Тому будь-який вектор можна представити у вигляді векторної суми:   
S = S '+ S ", (5)   
де S' = λa, 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 λa + a T S "= λa T a, (6)   
звідки λ = (aT 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 полуопределенності розглянемо твір y T Py для довільного y ≠ 0. Тоді   
https://math.semestr.ru/optim/images/mpg-image009.gifhttps://math.semestr.ru/optim/images/mpg-image008.gif .   
   
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 (xk) - напрямок спуску.   
Якщо S "= 0, то точка x k відповідає неодмінним умовам Лагранжа. Вектор множників Лагранжа задається виразом   
λ = (AAT) -1 A ▽ f. (9)   
Перше твердження випливає з того, що матриця P симетрична і позитивно піввизначена. Дійсно, маємо   
xk +1 = xk -αP ▽ f (xk); α≥0;   
f (xk + 1 ) = f (xk - αP ▽ f (xk)) = f (xk) - α ▽ fT (xk + 1 ) P ▽ f (xk)   
Так як, 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 = AT λ. (10) Оскільки рядками матриці A є вектори коефіцієнтів в лінійних обмеженнях, то (10) являє собою іншу форму запису необхідної умови Лагранжа, де ak, S=S’+S”=ATλ+S”=0  
https://math.semestr.ru/optim/images/mpg-image012.gif https://math.semestr.ru/optim/images/mpg-image013.gif  
- k-й вектор обмежень. Звідси ми бачимо, що λk - це множники Лагранжа, т. Е. Множники функції L.  
Необхідна умова екстремуму:   
https://math.semestr.ru/optim/images/mpg-image014.gif;   
▽ hk = ak .   
Неважко отримати вираз для λ: S = S '+ S ";   
https://math.semestr.ru/optim/images/mpg-image015.gif   
тобто λ = (AAT) -1 A ▽ f, (12)   
де (AA T ) -1 існує тільки в тому випадку, якщо значення ak лінійно незалежні. Якщо є лінійно залежні обмеження, їх слід виключити з розгляду.

**Основний алгоритм проекцій градієнта**

Обчислити матрицю P = IAT(AAT) -1 A в припущенні, що вектори ak лінійно незалежні.   
Задати ε> 0- похибка збіжності. Нехай знайдена допустима точка xk.   
**Крок 1**. Обчислити Sk = -P ▽ f (xk).   
 **Крок 2**. Якщо | Sk | ≤ ε, то обчислити λ за формулою (9) і закінчити обчислення. В іншому випадку - продовжити обчислення. Перехід на Крок 3.   
 **Крок 3.** Визначити максимальну довжину кроку:   
   
 **Крок 4.** Вирішити задачу одновимірного пошуку:   
f (x k + αSk ) → min; 0≤α≤ α max .   
 **Крок 5.** Покласти x k +1= x k + α S k → 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≥b j , j = 1..m.   
Складається матриця обмежень з рядків, відповідних активним обмеженням. Обчислюється проекційний оператор P і проекція Sk . Якщо | Sk| ≤ ε, то обчислюємо множники Лагранжа:   
λ = (AAT) -1 A ▽ f (xk).   
Якщо всі λ i ≥ 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.  
2.3.

**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());  
 }  
}

