

DFP - DAVIDON-FLETCHER-POWELL METHOD (Метод Дэвидона-Флетчера-Пауэлла)

Один из квазиньютоновских методов оптимизации(нахождение экстремума заданной функции), основанный на накоплении информации о кривизне целевой функции по наблюдениям за изменением градиента.

Основное отличие от ньютоновских методов - исключение явного формирования матрицы Гессе, заменяя её некоторым приближением.

Квазиньютоновские методы отличаются между собой лишь способом обновления матрицы Гессе.

Алгоритм схематически

Имеем функцию $f(x,y)$ - выпуклая, имеющая вторые производные.

1. Инициализируем начальную точку x_0 , выбираем необходимую точность $\epsilon > 0$, инициализируем симметрическую положительно определенную матрицу H_0 (обычно единичную), $k=j=0$
2. If $|grad(f(x_j))| < \epsilon \rightarrow stop$;
else $p_j = -H_j * grad(f(x_j))$ // Ищем точку в направлении которой будем производить поиск
3. Ищем такое $\alpha_j \geq 0$, чтобы $f(x_j + \alpha_j p_j) = \min(f(x_j + \alpha_j p_j))$ (Простая задача на нахождение экстремума функции одной переменной)
4. $x_{j+1} = x_j + \alpha_j p_j$
5. $s_j = x_{j+1} - x_j$
6. Вычисляем $grad(f(x_{j+1}))$
7. $y_j = grad(f(x_{j+1})) - grad(f(x_j))$
8. Находим приближение гессиана по формуле
$$H_{j+1} = H_j - (H_j y_j y_j^T H_j) / (y_j^T H_j y_j) + (s_j s_j^T) / (y_j^T s_j)$$
9. Возвращаемся на шаг №2

Пример исходного кода на языке C

```
int Fletcher_Powell_Davidon( double (*f)(double *),
                             void (*df)(double*, double*),
                             int (*Stopping_Rule)(double*, double*, int, int),
                             double *a, double *dfa, double
line_search_cutoff,
                             double line_search_cutoff_scale_factor,
                             double line_search_tolerance, int n )
{
    double *x;
    double *dx;
    double *dg;
    double *H;
    double *Hg;
    double *v;
    double p;
    int iteration = 0;
    int count_between_resets = n;
    int err = 0;

    x = (double *) malloc( n * sizeof(double) );
    v = (double *) malloc( n * sizeof(double) );
    dx = (double *) malloc( n * sizeof(double) );
    dg = (double *) malloc( n * sizeof(double) );
    Hg = (double *) malloc( n * sizeof(double) );
    H = (double *) malloc( ((n * (n + 1)) >> 1) * sizeof(double) );

    if (H == NULL) err = -1;
    else {
        df(a, dfa);
        do {
            iteration++;
            count_between_resets++;
            if (count_between_resets > n) {
                Identity_Matrix_ut( H, n );
                count_between_resets = 1;
            }
            Multiply_Sym_Matrix_by_Vector_ut(v, H, dfa, n);
            Minimize_Down_the_Line(f, a, f(a), &p, v, x, line_search_cutoff,
                                  line_search_cutoff_scale_factor,
line_search_tolerance, n);
            Subtract_Vectors(dx, x, a, n);
            Copy_Vector(a, x, n);
            Copy_Vector(x, dfa, n);
```

```

    df(a, dfa);
    Subtract_Vectors(dg, dfa, x, n);
    Update_H(H, dx, dg, Hg, n);
    err = Stopping_Rule(dx, dfa, iteration, n);
    } while ( !err );
}
free(Hg);
free(H);
free(dg);
free(dx);
free(v);
free(x);

return err;
}

static void Update_H(double *H, double* dx, double* dg, double* Hg, int
n)
{
    double *Hdg;
    double dxtdg;
    double dgtHdg;
    double *pH;
    int i, j;

    dxtdg = Inner_Product(dx, dg, n);
    Multiply_Sym_Matrix_by_Vector_ut(Hg, H, dg, n);
    dgtHdg = Inner_Product(dg, Hg, n);

    dxtdg = 1.0 / dxtdg;
    dgtHdg = 1.0 / dgtHdg;

    pH = H;
    for (i = 0; i < n; i++)
        for (j = i; j < n; j++)
            *pH++ += dxtdg * dx[i] * dx[j] - dgtHdg * Hg[i] * Hg[j];
}

```

Комментарии по коду

double (*f)(double *) //указатель на функцию f от n-переменных

void (*df)(double*, double*) //функция, вычисляющая градиент в точке a, и возвращающая результат в массив dfa

Вычисление гессиана вынесено в отдельную функцию.

Источник кода:

http://www.mymathlib.com/optimization/nonlinear/unconstrained/fletcher_powell_davidon.html

Выполнили: Богатырев Илья, Скаборт Константин, Степанов Денис, Амиров Аскар.
КМБО-03-14