Севастопольский государственный университет
Институт информационных технологий

Лекция
ИТЕРАТИВНЫЕ АЛГОРИТМЫ ОПТИМИЗАЦИИ

Бондарев Владимир Николаевич

АЛГОРИТМЫ ОПТИМИЗАЦИИ ЦЕЛЕВОЙ ФУНКЦИИ

План

1. Итеративные алгоритмы оптимизации
2. Алгоритм наискорейшего спуска (SDA)
   - SDA с фиксированной скоростью обучения
   - SDA с минимизацией вдоль направления
3. Метод Ньютона
4. Метод сопряженных градиентов

Итеративные алгоритмы оптимизации

Цель оптимизации заключается в поиске вектора $\mathbf{x}$, который минимизирует целевую функцию $F(\mathbf{x})$.

Будем использовать для этого алгоритмы последовательного приближения – **итеративные алгоритмы**. Поиск начинается с начального значения $\mathbf{x}_0$ и на каждом шаге

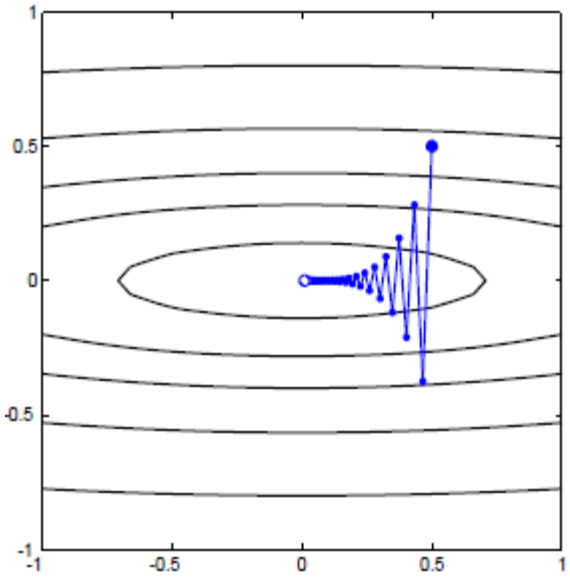
$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k,$$

или

$$\Delta \mathbf{x}_k = (\mathbf{x}_{k+1} - \mathbf{x}_k) = \alpha_k \mathbf{p}_k,$$

где $\mathbf{p}_k$ – вектор, определяющий направление поиска; α_k - скорость обучения, определяющая длину шага.

Алгоритмы отличаются выбором вектора направления поиска $\mathbf{p}_k$ и способами вычисления значений скорости обучения



Алгоритм наискорейшего спуска (steepest descent algorithm – SDA)

При поиске минимума должно выполняться условие : $F(\mathbf{x}_{k+1}) < F(\mathbf{x}_k)$. (1)

Рассмотрим ряд Тейлора для ЦФ в районе т. $\mathbf{x_{k+1}}$

$$F(\mathbf{x}_{k+1}) = F(\mathbf{x}_k + \Delta\mathbf{x}_k) \approx F(\mathbf{x}_k) + \mathbf{g}_k^T \Delta\mathbf{x}_k, \quad \mathbf{g}_k \equiv \nabla F(\mathbf{x})\big|_{\mathbf{x}=\mathbf{x}_k}. \quad (2)$$

где $\mathbf{g}_k$ - градиент .

Чтобы выполнялось условие (1), второй член (2) должен быть отрицательным:

$$\mathbf{g}_k^T \Delta\mathbf{x}_k = \alpha_k \mathbf{g}_k^T \mathbf{p}_k < 0 \qquad \text{или} \qquad \mathbf{g}_k^T \mathbf{p}_k < 0 .$$

Любой вектор $\mathbf{p}_k$, удовлетворяющий этому условию, соответствует **направлению спуска. Направление наискорейшего спуска :**

$$\mathbf{p}_k = -\mathbf{g}_k .$$

Соответственно процедура **SDA** определяется выражением:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \mathbf{g}_k .$$

Есть **два подхода определения** α_k :

1) использовать фиксированное значение, например $\alpha_k = 0.01$ или переменное, но предопределенное, например $\alpha_k = 1/k$.

2) минимизировать на каждом шаге $F(\mathbf{x})$ по отношению к α_k вдоль $\mathbf{p}_k$

Отметим, что направление наискорейшего спуска ортогонально линиям равных контуров целевой функции.

Устойчивость SDA с фиксированной скоростью обучения

Пусть целевая функция является **квадратичной**. Тогда $\nabla F(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{d}$.
Подставив это значение в SDA, получим

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha \mathbf{g}_k = \mathbf{x}_k - \alpha(\mathbf{A}\mathbf{x}_k + \mathbf{d}) \quad \text{или} \quad \mathbf{x}_{k+1} = [\mathbf{I} - \alpha\mathbf{A}]\mathbf{x}_k - \alpha\mathbf{d}.$$

Это уравнение линейной динамической системы, которая будет устойчивой, если собственные значения матрицы $[\mathbf{I} - \alpha\mathbf{A}]$ меньше 1.
Пусть $\{\lambda_1, \lambda_2, \dots, \lambda_n\}$ и $\{\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_n\}$ собственные значения и собственные векторы **A**.
Тогда

$$[\mathbf{I} - \alpha\mathbf{A}]\mathbf{z}_i = \mathbf{z}_i - \alpha\mathbf{A}\mathbf{z}_i = \mathbf{z}_i - \alpha\lambda_i\mathbf{z}_i = (1 - \alpha\lambda_i)\mathbf{z}_i.$$

Т.е собственные векторы матрицы $[\mathbf{I} - \alpha\mathbf{A}]$ совпадают с собственными векторами **A,** а собственные значения равны $(1 - \alpha\lambda_i)$. Тогда **условие устойчивости SDA** запишется в виде $|(1 - \alpha\lambda_i)| < 1$.
Если квадратичная $F(\mathbf{x})$ имеет сильный минимум, то все собственные значения должны быть положительными. Тогда условие устойчивости

$$\alpha < \frac{2}{\lambda_i}.$$

Поскольку оно должно выполняться для всех собственных чисел, то, очевидно,

$$\alpha < \frac{2}{\lambda_{max}}.$$

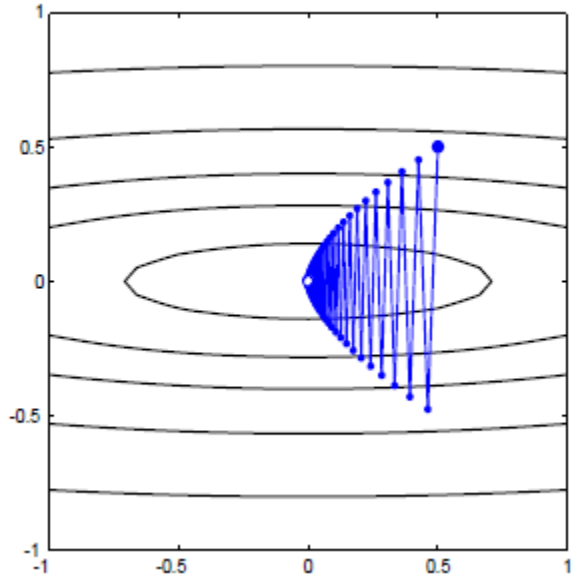
Т.е. скорость обучения обратно пропорциональна кривизне квадратичной функции.

Пример SDA

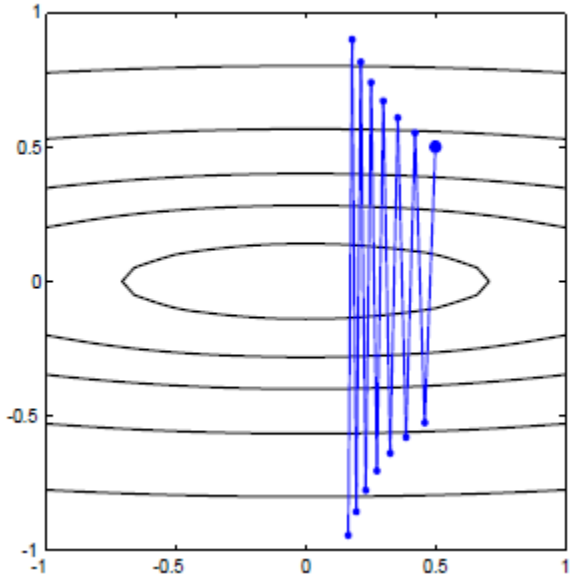
$$F(\mathbf{x}) = x_1^2 + 25x_2^2 \qquad \mathbf{A} = \begin{bmatrix} 2 & 0 \\ 0 & 50 \end{bmatrix} \qquad \left\{(\lambda_1 = 2), \left(\mathbf{z}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}\right)\right\}, \left\{(\lambda_2 = 50), \left(\mathbf{z}_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}\right)\right\}.$$

$$\alpha < \frac{2}{\lambda_{max}} = \frac{2}{50} = 0.04. \qquad \mathbf{x}_0 = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}. \qquad \nabla F(\mathbf{x}) = \begin{bmatrix} \frac{\partial}{\partial x_1} F(\mathbf{x}) \\ \frac{\partial}{\partial x_2} F(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} 2x_1 \\ 50x_2 \end{bmatrix}. \qquad \mathbf{g}_0 = \nabla F(\mathbf{x})\Big|_{\mathbf{x} = \mathbf{x}_0} = \begin{bmatrix} 1 \\ 25 \end{bmatrix}.$$

$\alpha = 0.039$



$\alpha = 0.041$



SDA с оптимизацией целевой функции по скорости обучения

В этом случае на каждом шаге алгоритма выполняется поиск α_k которое **минимизирует** $F(\mathbf{x}_k + \alpha_k \mathbf{p}_k)$ **вдоль направления** $\mathbf{p}_k$.
Для произвольных $F(\mathbf{x})$ требуется выполнять линейный поиск, но для квадратичной $F(\mathbf{x})$ решение можно найти аналитически.
Можно показать, используя ряд Тейлора и $\Delta \mathbf{x}_k = \alpha_k \mathbf{p}_k$, что производная $F(\mathbf{x})$ по скорости обучения

$$\frac{d}{d\alpha_k} F(\mathbf{x}_k + \alpha_k \mathbf{p}_k) = \nabla F(\mathbf{x})^T \Big|_{\mathbf{x}=\mathbf{x}_k} \mathbf{p}_k + \alpha_k \mathbf{p}_k^T \nabla^2 F(\mathbf{x}) \Big|_{\mathbf{x}=\mathbf{x}_k} \mathbf{p}_k .$$

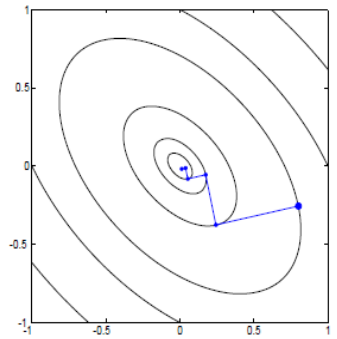
Приравняв эту производную нулю, получим оптимальное α_k

$$\alpha_k = - \frac{\nabla F(\mathbf{x})^T \Big|_{\mathbf{x}=\mathbf{x}_k} \mathbf{p}_k}{\mathbf{p}_k^T \nabla^2 F(\mathbf{x}) \Big|_{\mathbf{x}=\mathbf{x}_k} \mathbf{p}_k} = - \frac{\mathbf{g}_k^T \mathbf{p}_k}{\mathbf{p}_k^T \mathbf{A}_k \mathbf{p}_k} .$$

Для квадратичной функции гессиан $\mathbf{A}$ не зависит от k.
Последовательные шаги алгоритма выполняются вдоль взаимно ортогональных направлений, т.к.

$$\frac{d}{d\alpha_k} F(\mathbf{x}_{k+1}) = \nabla F(\mathbf{x})^T \Big|_{\mathbf{x}=\mathbf{x}_{k+1}} \frac{d}{d\alpha_k} [\mathbf{x}_k + \alpha_k \mathbf{p}_k] = \nabla F(\mathbf{x})^T \Big|_{\mathbf{x}=\mathbf{x}_{k+1}} \mathbf{p}_k = \mathbf{g}_{k+1}^T \mathbf{p}_k = 0$$

Следовательно в т. минимума по α_k градиент ортогонален предыдущему направлению поиска.



Метод Ньютона

Алгоритм SDA основан на использовании первых производных. Метод Ньютона основан на поиске стационарной точки квадратичной аппроксимации ЦФ $F(\mathbf{x})$:

$$F(\mathbf{x}_{k+1}) = F(\mathbf{x}_k + \Delta\mathbf{x}_k) \approx F(\mathbf{x}_k) + \mathbf{g}_k^T \Delta\mathbf{x}_k + \frac{1}{2}\Delta\mathbf{x}_k^T \mathbf{A}_k \Delta\mathbf{x}_k . \quad (1)$$

Найдем градиент (1) по отношению к $\Delta\mathbf{x}_k$ и приравняем его нулю

$$\mathbf{g}_k + \mathbf{A}_k \Delta\mathbf{x}_k = \mathbf{0} .$$

Отсюда оптимальный шаг равен $\Delta\mathbf{x}_k = -\mathbf{A}_k^{-1}\mathbf{g}_k$.

Метод Ньютона

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \mathbf{A}_k^{-1}\mathbf{g}_k .$$

Пример:

$$F(\mathbf{x}) = x_1^2 + 25x_2^2 \qquad \mathbf{A} = \begin{bmatrix} 2 & 0 \\ 0 & 50 \end{bmatrix} \qquad \mathbf{x}_0 = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} .$$

На первом шаге метода Ньютона получаем:

$$\mathbf{x}_1 = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} - \begin{bmatrix} 2 & 0 \\ 0 & 50 \end{bmatrix}^{-1} \begin{bmatrix} 1 \\ 25 \end{bmatrix} = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} - \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} .$$

Метод всегда находит минимум квадратичной функции за один шаг. В общем случае метод Ньютона не сходится за один шаг и не гарантирует схождение. Это зависит от вида $F(\mathbf{x})$ и начальных условий поиска.