09.04.03 Прикладная информатика

Профиль «Машинное обучение и анализ данных»

Дисциплина «Математические основы анализа данных»

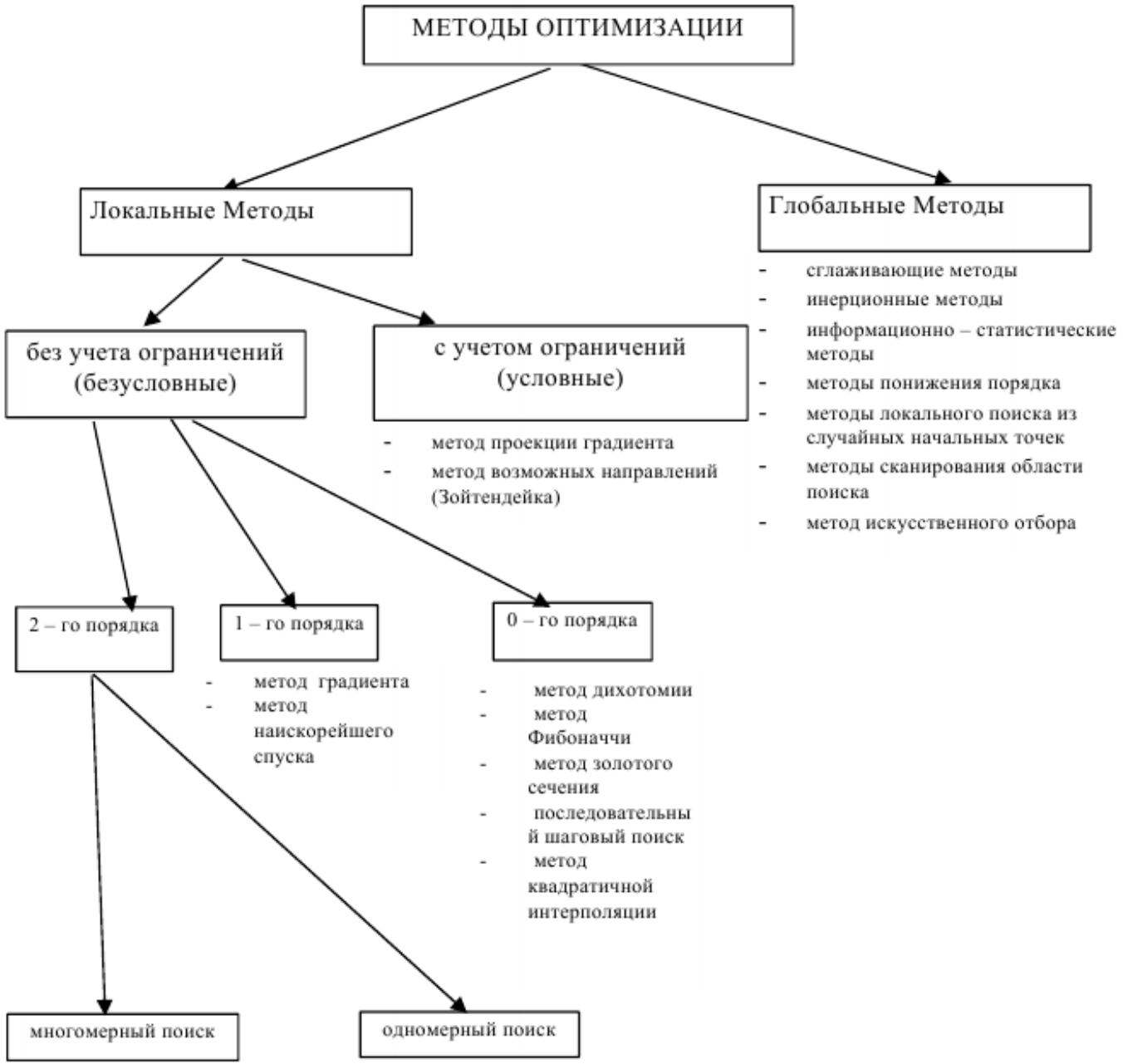
Лекция 8

Численные методы многомерной оптимизации

План лекции

- Классификации методов оптимизации
- Метод прямого поиска (метод Хука-Дживса)
- Метод деформируемого многогранника (метод Нелдера-Мида)
- Метод вращающихся координат (метод Розенброка)
- Градиентные методы

Классификация методов оптимизации



Метод прямого поиска (метод Хука-Дживса)

- Суть метода (на примере минимизации): Задаются некоторой начальной точкой x[0]. Изменяя компоненты вектора x[0], обследуют окрестность данной точки, в результате чего находят направление, в котором происходит уменьшение значений функции f(x).
- В выбранном направлении осуществляют спуск до тех пор, пока значение функции уменьшается.
- После того как в данном направлении не удается найти точку с меньшим значением функции, уменьшают величину шага спуска.
- Если последовательные дробления шага не приводят к уменьшению функции, от выбранного направления спуска отказываются и осуществляют новое обследование окрестности и т. д.

Алгоритм метода Хука-Дживса

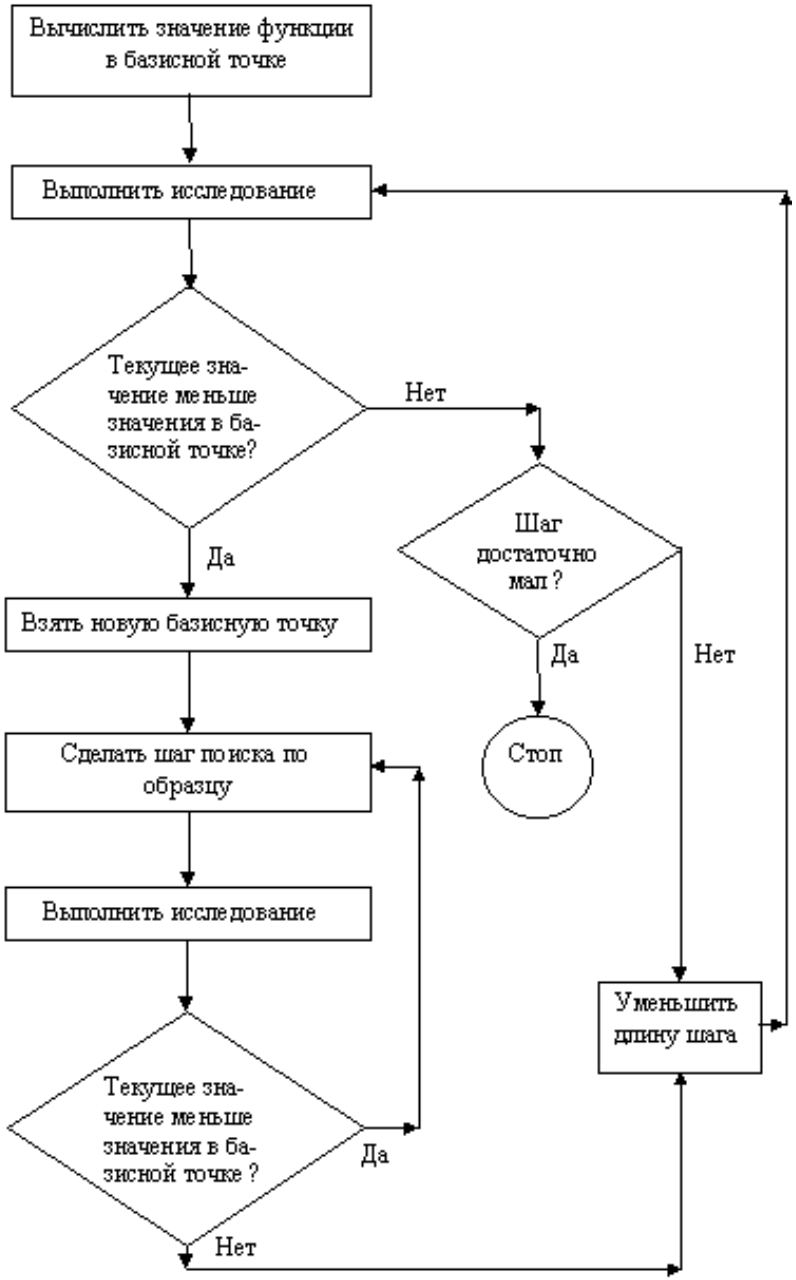
1. Задаются координаты $x_i[0]$, $i = 1, ..., n$, начальной (базисной) точки $x[0]$, и вектор изменения координат Δx.
2. Вычисляется значение функции $f(x[0])$ в базисной точке $x[0]$.
3. Каждая переменная по очереди изменяется прибавлением длины шага. Таким образом, вычисляем значение функции $f(x[0] + \Delta x_1)$, где Δx_1 –компонента вектора в направлении оси Ox.
4. Если это приводит к уменьшению значения функции, то $x[0]$ заменяется на $x[0] + \Delta x_1$. В противном случае вычисляется значение функции $f(x[0] - \Delta x_1)$, и если ее значение уменьшилось, то $x[0]$ заменяем на $x[0] - \Delta x_1$.

- Если ни один из проделанных шагов не приводит к уменьшению значения функции, то точка $x[0]$ остается неизменной и рассматриваются изменения в направлении оси Oy, т. е. находится значение функции $f(x[0] + \Delta x_2)$ и т. д.
- Когда будут рассмотрены все n переменные, мы будем иметь новую базисную точку $x[1]$.

- Если $x[1] = x[0]$, т. е. уменьшение функции не было достигнуто, то исследование повторяется вокруг той же базисной точки $x[0]$, **но с уменьшенной длиной шага**.
- На практике удовлетворительным является уменьшение шага (шагов) в десять раз от начальной длины.

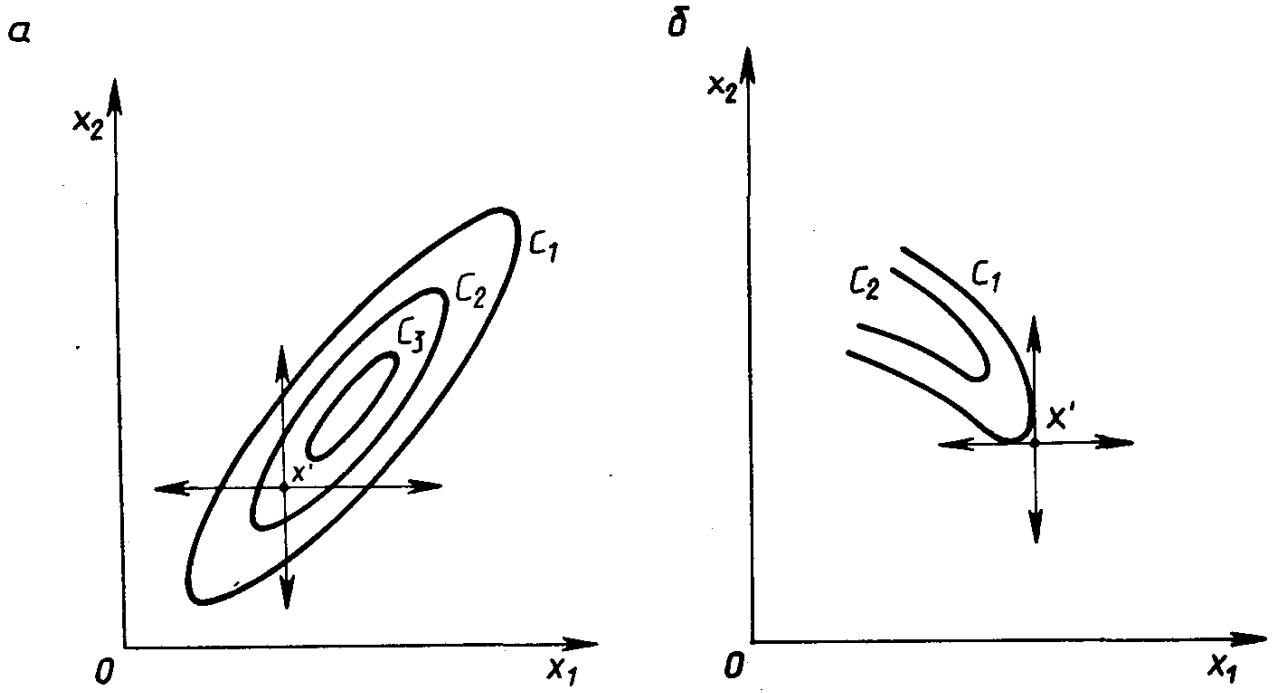
4. Если $x[1] \neq x[0]$, то производится поиск по образцу с пункта 3.

Блок-схема данного метода



Недостаток метода прямого поиска

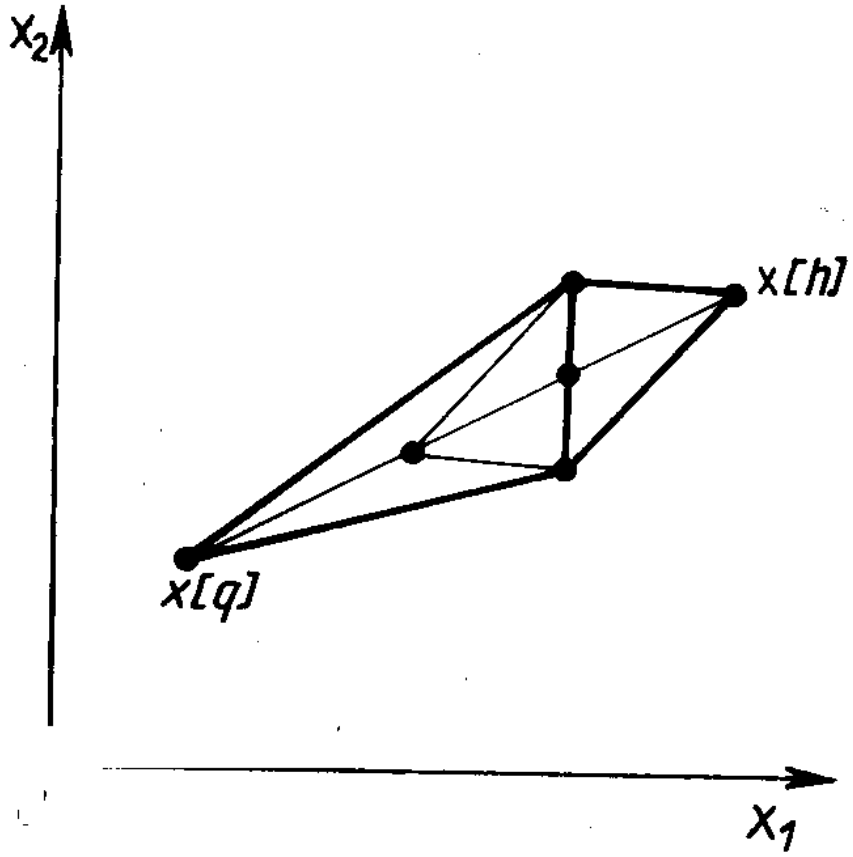
- В случае сильно вытянутых, изогнутых или обладающих острыми углами линий уровня целевой функции он может оказаться неспособным обеспечить продвижение к точке минимума.



Метод деформируемого многогранника (метод Нелдера-Мида)

- Для минимизации функции n переменных $f(x)$ в n-мерном пространстве строится многогранник, содержащий (n + 1) вершину.
- Каждая вершина соответствует некоторому вектору x.
- Вычисляются значения целевой функции $f(x)$ в каждой из вершин многогранника, определяются максимальное из этих значений и соответствующая ему вершина $x[h]$.
- Через эту вершину и центр тяжести остальных вершин проводится проецирующая прямая, на которой находится точка $x[q]$ с меньшим значением целевой функции, чем в вершине $x[h]$.

Геометрическая интерпретация метода деформируемого многогранника



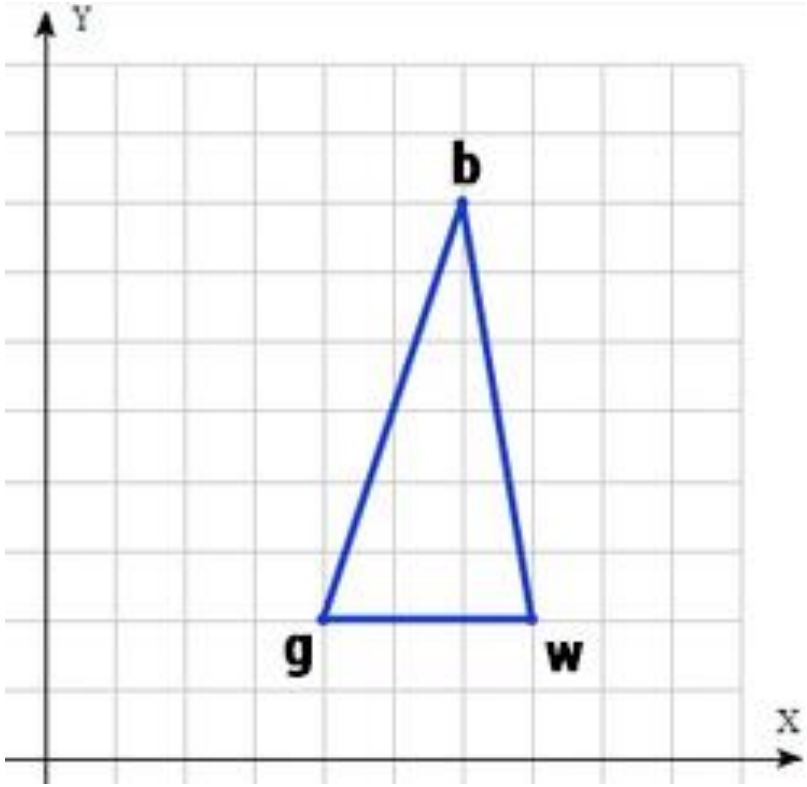
Алгоритм

1) Пусть $f(x,y)$ функция, которую необходимо оптимизировать.

На первом шаге выбираем три случайные точки и формируем симплекс (треугольник). Вычисляем значение функции в каждой точке: $f(V_1)$, $f(V_2)$, $f(V_3)$.

- Сортируем точки по значениям функции $f(x,y)$ в этих точках, таким образом получаем двойное неравенство: $f(V_2) \le f(V_1) \le f(V_3)$.

- Для удобства обозначим точки следующим образом:
  $b = V_2$, $g = V_1$, $w = V_3$, где b=best, g=good, w=worst — соответственно.



2) На следующем шаге находим середину отрезка, точками которого являются g и b.

Т.к. координаты середины отрезка равны полусумме координат его концов, получаем:

$$mid = \left(\frac{x_1 + x_2}{2}; \frac{y_1 + y_2}{2} \right)$$

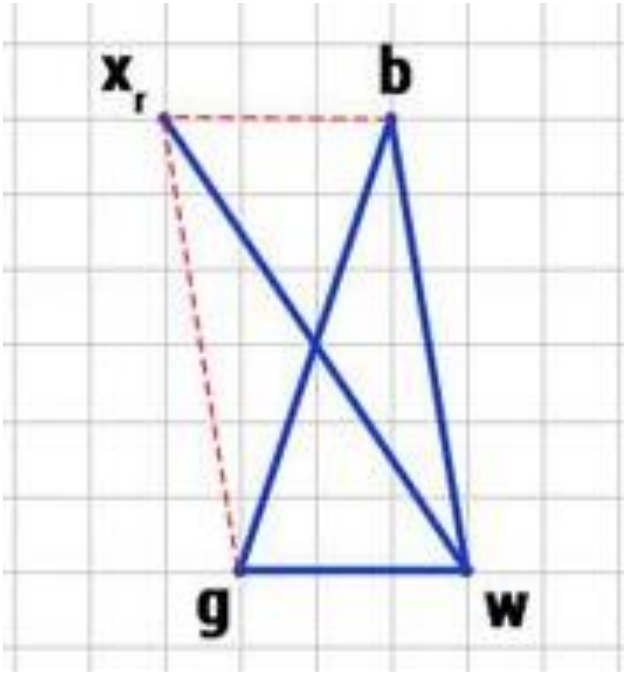
- В более общем виде можно записать так:

$$mid = \frac{1}{n} \sum_{i=1}^{n} x_i$$

3) Применяем операцию отражения: уаходим точку x_r следующим образом

$$x_r = mid + \alpha(mid - w)$$

- Т.е. фактически отражаем точку w относительно mid. В качестве коэффициента берут как правило 1.
- Проверяем нашу точку: если $f(x_r) < f(g)$, то это «хорошая» точка.
- А теперь попробуем расстояние увеличить в 2 раза, вдруг повезёт и найдём точку ещё лучше.

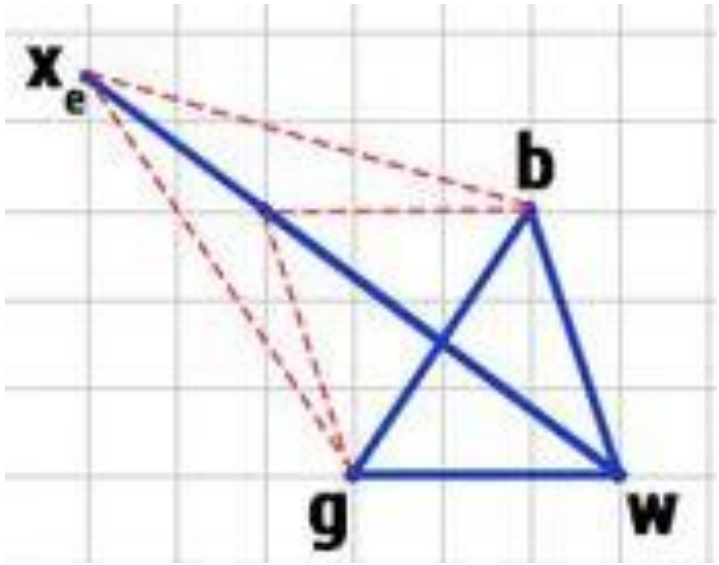


- 4) Применяем операцию растяжения: Находим точку x_e следующим образом:

$$x_e = mid + \gamma(x_r - mid)$$

- В качестве γ принимаем γ = 2, т.е. расстояние увеличиваем в 2 раза.
- Проверяем точку x_e : если $f(x_e) < f(b)$, то нам повезло и мы нашли точку лучше, чем есть на данный момент, если бы этого не произошло, мы бы остановились на точке x_r.

- Далее заменяем точку w на x_e, в итоге получаем:

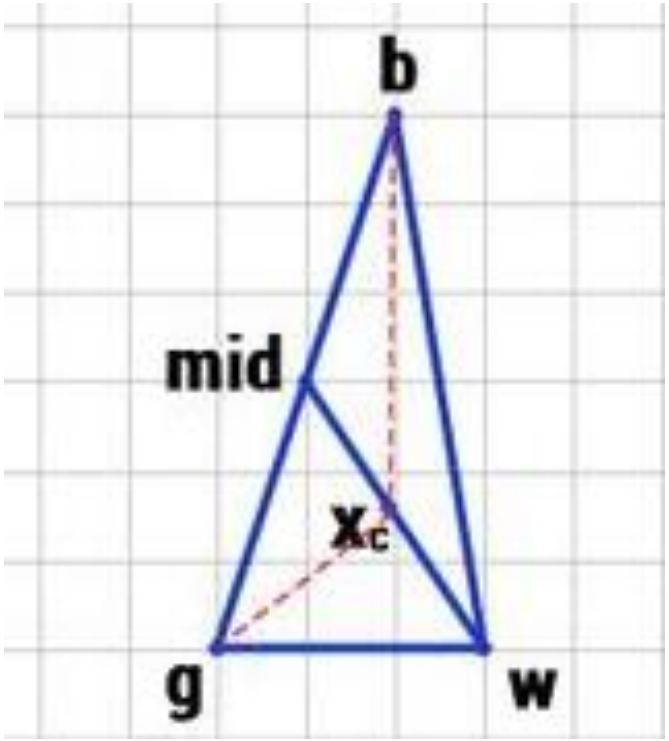


5) Если же нам не повезло и мы не нашли хороших точек, пробуем операцию сжатия, т.е. будем уменьшать отрезок и искать хорошие точки внутри треугольника.

- Пробуем найти хорошую точку x_c:

$$x_c = mid + \beta(w - mid)$$

- Коэффициент β принимаем равным 0.5, т.е. точка x_c на середине отрезка от w до mid.



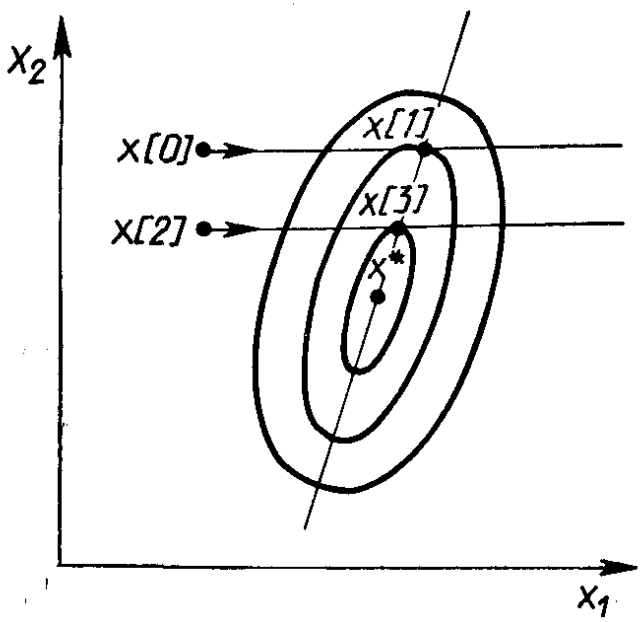
Алгоритм заканчивается, когда:

1)Было выполнено необходимое количество итераций.
2) Площадь симплекса достигла определенной величины.
3) Текущее лучшее решение достигло необходимой точности.

- Как и в большинстве эвристических методов, не существует идеального способа выбора инициализирующих точек.
- Можно брать случайные точки, находящиеся недалеко друг от друга для формирования симплекса.
- Реализация на языке программирования python: https://habr.com/ru/post/332092/

Метод вращающихся координат (метод Розенброка)

- Этот метод использует свойство квадратичной функции, заключающееся в том, что любая прямая, которая проходит через точку минимума функции x^*, пересекает под равными углами касательные к поверхностям равного уровня функции в точках пересечения.



- Из начальной точки x[0] осуществляют спуск в точку x[1] по направлениям, параллельным координатным осям.
- На следующей итерации одна из осей должна проходить в направлении y1 = x[1] — x[0], а другая – в направлении, перпендикулярном к y1.
- Спуск вдоль этих осей приводит в точку x[2] , что даёт возможность построить новый вектор x[2] — x[1] и на его базе новую систему направлений поиска.
- В общем случае данный метод эффективен при минимизации овражных функций, так как результирующее направление поиска стремится расположиться вдоль оси оврага.

Алгоритм метода вращающихся координат:

1. Обозначим через $p_1[k]$, ..., $p_n[k]$ направления координатных осей в некоторой точке $x[k]$ (на k-й итерации). Выполняют пробный шаг h_1 вдоль оси $p_1[k]$, т. е.

$$x[k_1] = x[k] + h_1 p_1[k].$$

- Если при этом $f(x[k_1]) < f(x[k])$, то шаг h умножают на величину $\alpha > 1$;
- Если $f(x[k_1]) > f(x[k])$, то на величину $(-\alpha)$, $0 < |\alpha| < 1$;

$$x[k] = x[k] + \alpha h_1 p_1[k].$$

- Полагая h_1 = a1 .получают x[kl] = x[k] + a1p1[k].

2. Из точки $x[k_1]$ выполняют шаг h_2 вдоль оси $p_2[k]$:

$$x[k_2] = \mathrm{x}[k] + a_1 p_1[k] + h_2 p_2[k].$$

- Повторяют операцию п. 1, т. е.

$$x[k_2] = \mathrm{x}[k] + a_1 p_1[k] + a_2 p_2[k].$$

- Эту процедуру выполняют для всех остальных координатных осей. На последнем шаге получают точку

$$x[k_n] = x[k+1] = x[k] + p_n[k] \ .$$

3. Выбирают новые оси координат $p_1[k+1]$, …, $p_n[k+1]$. В качестве первой оси принимается вектор $p_1[k+1] = x[k+\mathrm{l}]—x[k]$.

- Коэффициенты α подбираются эмпирически. Хорошие результаты дают значения $\alpha = -0{,}5$ при неудачных пробах ($f(x[k_i]) > f(x[k])$) и $\alpha = 3$ при удачных пробах ($f(x[k_i]) < f(x[k])$).
- В отличие от других методов нулевого порядка алгоритм Розенброка ориентирован на отыскание оптимальной точки в каждом направлении, а не просто на фиксированный сдвиг по всем направлениям.
- Величина шага в процессе поиска непрерывно изменяется в зависимости от рельефа поверхности уровня.

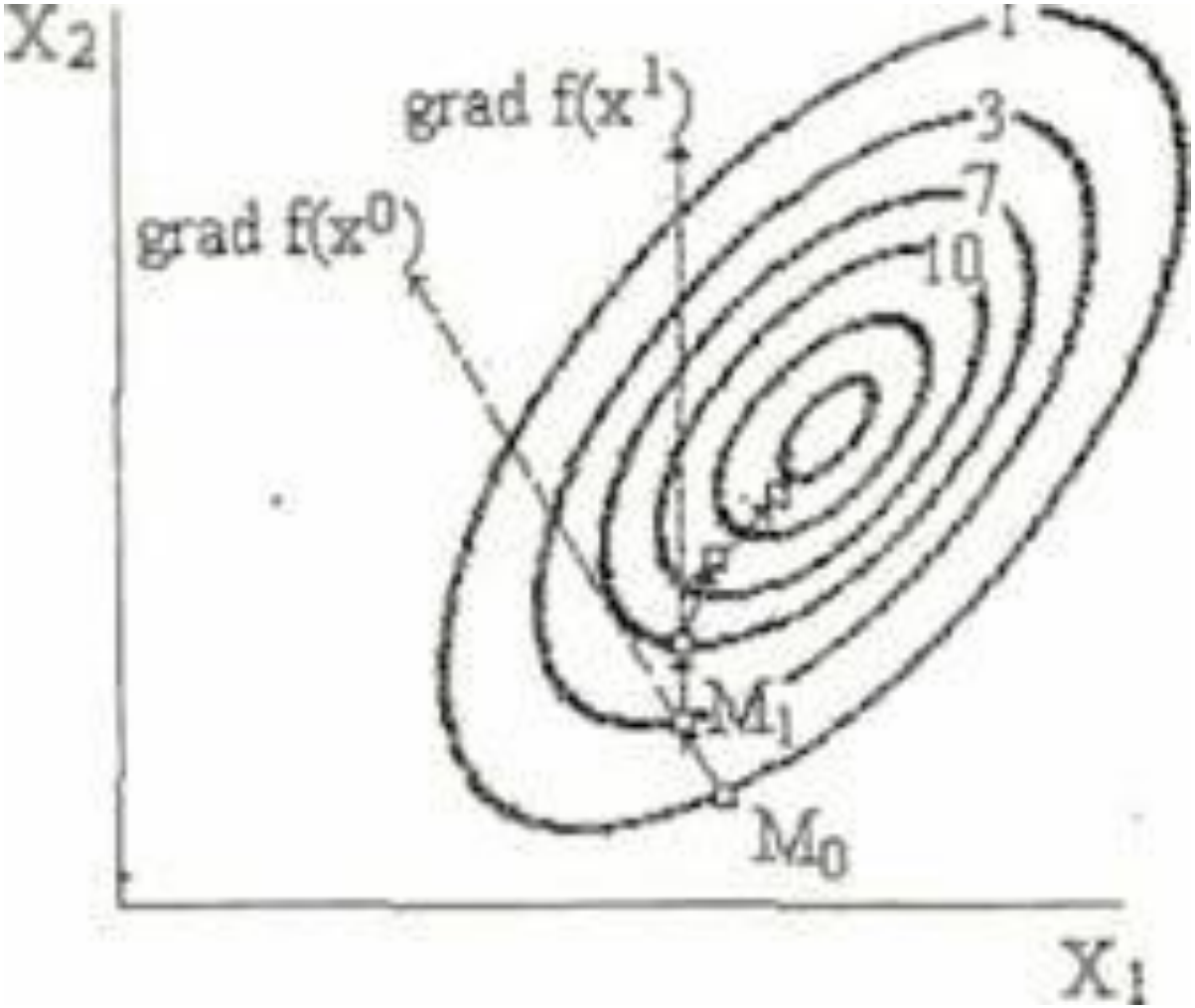
Градиентные методы

- Направление наибольшего возрастания функции характеризуется её градиентом.
- **Противоположное направление - это направление наиболее крутого убывания.**
- Если функция задана аналитически, то вычисление градиента не представляет принципиальных трудностей.
- Наряду с определением градиентного вектора основным вопросом, решаемым в методах градиента, является выбор шага движения по градиенту.
- Выбор величины шага в значительной степени зависит от вида поверхности.
- Если шаг слишком мал, это потребует продолжительных расчетов.
- Если наоборот размеры шага слишком велики, можно «проскочить» оптимум.

Метод градиента

- Пусть в начальный момент значения X_1 и X_2 соответствуют точке M_0. Цикл расчета начинается с серии пробных шагов.
- Сначала величине X_1 дается небольшое приращение > 0, причем в это время X_2 неизменно.
- Затем определяется полученное при этом приращение Δf, величины f, которое можно считать пропорциональным значению величины частной производной.

Иллюстрация метода градиента



- Далее производится приращение величины X_2. В это время X_1= const.
- Получаемое при этом приращение величины f является мерой другой частной производной.
- После нахождения составляющих градиента делается рабочие шаги в направлении вектора градиента, если стоит задача определения максимума и в противоположном направлении, если решается задача поиска минимума.

На (k+1)-ом шаге применяется формула:

$$X_i^{k+1} = X_i^k \pm h^k * S^k$$

- Здесь h^k – величина шага

$$S^k = \frac{\nabla f(X^k)}{\left\|\nabla f(X^k)\right\|}$$

- Норма градиента определяется как

$$\left\|\nabla f(X^k)\right\| = \sqrt{\left(\frac{\partial f}{\partial X_1}\right)^2 + \left(\frac{\partial f}{\partial X_2}\right)^2}$$

Пример: $f(X) = X_1^2 + 25\,X_2 \to min$

- Примем величину шага $h = 1, \Delta X_1 = \Delta X_2 = 0.05.$
- В качестве исходной точки поиска возьмём точку $X^0 = (2, 2)$

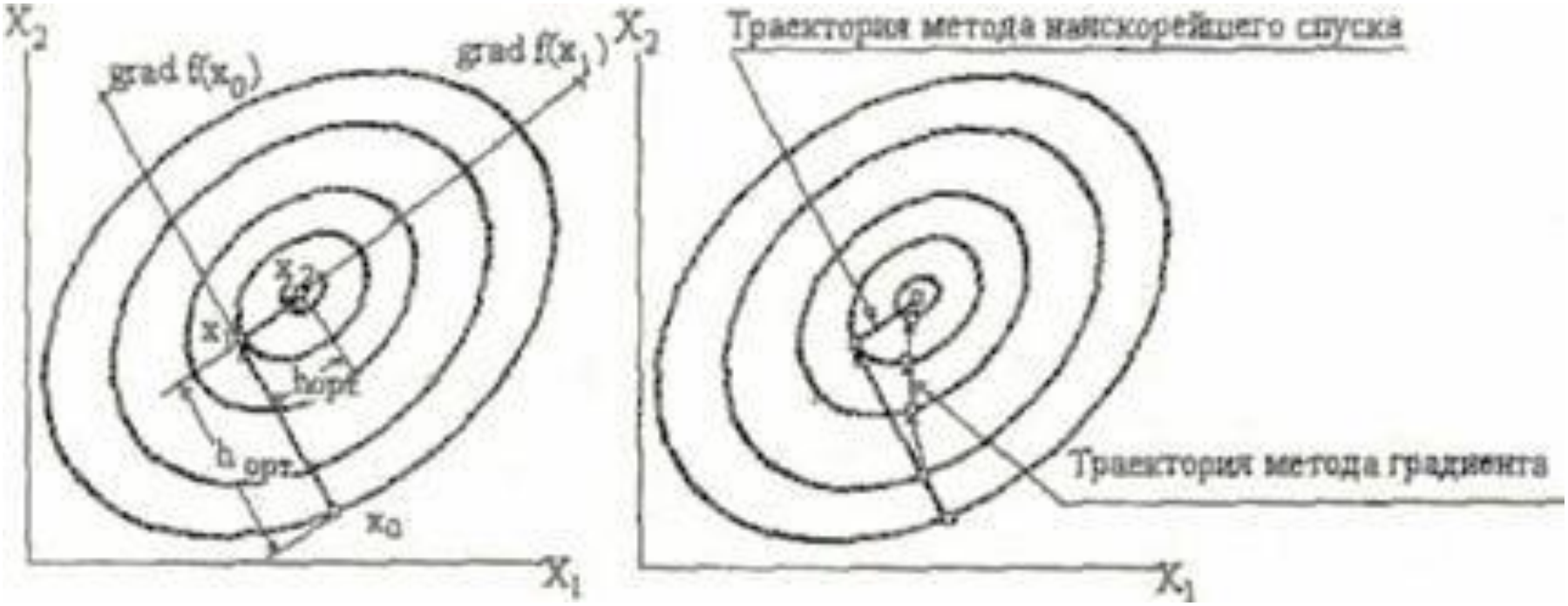
Шаг	X_1	X_2	$\nabla f(X_1^k)$	$\nabla f(X_2^k)$	$\|\nabla f(X^k)\|$	S_1^k	S_2^k	f
1	2	2	4.050	101.25	101.250	-0.040	-0.999	104.00
1	1.960	1.001	3.970	51.30	51.453	-0.077	-0.997	28.89
1	1.883	0.004	3.816	1.45	4.082	-0.035	-0.355	3.55
1	0.948	-0.351	1.94	-16.30	16.416	-0.119	0.993	3.98
Величина Δf>0,поэтому уменьшаем шаг вдвое.								
0.5	1.416	-0.174	2.882	-7.45	7.988	-.0180	0.466	2.76
0.5	1.236	0.292	2.552	15.85	16.049	-0.079	0.494	3.66
Величина Δf>0,поэтому уменьшаем шаг вдвое.								
0.25	1.326	0.059	2.702	4.20	4.994	-0.135	-0.21	1.84

Метод Коши (наискорейшего спуска или крутого восхождения)

- При использовании градиентного метода основной объем вычислении приходится на вычисление градиента целевой функции в каждой точке траектории спуска.
- Целесообразно уменьшить количество таких точек без ущерба для самого решения. Согласно этому методу Коши, после определения направления поиска оптимума в начальной точке, в этом направлении **делают не один шаг, а двигаются до тех пор пока происходит улучшение функции**, достигая таким образом, экстремума в некоторой точке.

- В этой точке вновь определяют направление поиска (с помощью градиента) и ищут новую точку оптимума целевой функции и т.д.
- В этом методе поиск происходит более крупными шагами, и градиент функции вычисляется в меньшем числе точек.
- **Метод наискорейшего спуска сводит многомерную задачу оптимизации к последовательности одномерных задач оптимизации, которые могут решаться, например, методом золотого сечения или половинного деления.**

Метод наискорейшего спуска



- Величину шага h можно определить из условия минимума $f(\mathrm{X}_k + h^k * S^k)$:

$$h^k = -\frac{\nabla^T f(X_k) S^k}{(S^k)^T * \nabla^2 f(X_k)}$$

Пример: $f(X) = X_1^2 + 25\,X_2 \rightarrow min$

- Начальные параметры:
- шаг $h = 1, \Delta X_1 = \Delta X_2 = 0.05$
- исходная точка $X^0 = (2, 2)$

Этап	Шаг h^k	X_1	X_2	$\frac{\partial f}{\partial X_1}$	$\frac{\partial f}{\partial X_2}$	f
0		2	2	4.050	101.25	104.00
1	2.003	1.92	-0.003	3.84	-0.15	3.19
2	1.85	0.07	0.07	0.14	3.5	0.13
3	0.07	0.07	-0.000			0.0049

Метод сопряжённых градиентов

- Квадратичная целевая функция n независимых переменных, имеющая минимум, может быть минимизирована за n шагов (или менее), если шаги предпринимаются в так называемых **сопряжённых направлениях**.
- Два вектора x и y называют A-сопряженными (или сопряженными по отношению к матрице A), если скалярное произведение x и Ay равно нулю.

$$(x, Ay) = 0$$

- В качестве матрицы берется

$$\left[\nabla^2 f(x_0)\right] = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} \\ \frac{\partial^2 f}{\partial x_2^2} & \frac{\partial^2 f}{\partial x_2 \partial x_1} \end{bmatrix}$$

$$\alpha_k = arg\min_{\alpha_k} F(x_{k-1} + \alpha_k p_k)$$

$$p_{k+1} = -F'(x_k) + \beta_k p_k$$

β_k можно вычислять по одной из трёх формул:

1. $\beta_k = -\frac{\langle F'(x_k), F'(x_k)\rangle}{\langle F'(x_{k-1}), F'(x_{k-1})\rangle}$ - *Метод Флетчера - Ривса (Fletcher–Reeves method)*
2. $\beta_k = \frac{\langle F'(x_k), F'(x_k) - F'(x_{k-1})\rangle}{\langle F'(x_{k-1}), F'(x_{k-1})\rangle}$ - *Метод Полака - Райбера (Polak–Ribi`ere method)*
3. $\beta_k = \frac{\langle F''(x_k)p_k, F'(x_k)\rangle}{\langle F''(x_{k-1})p_k, p_k\rangle}$

Справочная информация

- http://www.machinelearning.ru/wiki/index.php?title=Метод_сопряжённых_градиентов
- В машинном обучении применяется **градиентный бустинг**:
- https://neurohive.io/ru/osnovy-data-science/gradientyj-busting/