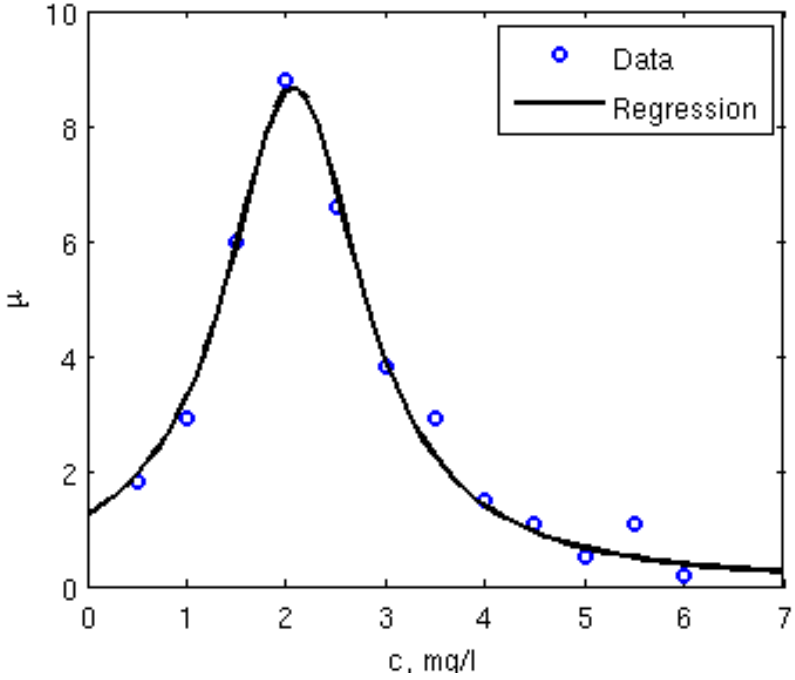
Лекция 9. Нелинейная регрессия



Краткое содержание

1. Нелинейная регрессия: постановка задачи
2. Методы Гаусса-Ньютона, Левенберга-Марквардта, доверительных областей
3. Функции в MATLAB/GNU Octave
4. Практическая реализация нелинейной регрессии в MATLAB/GNU Octave

Виды регрессии

1. Линейная регрессия

$$y = b_0 + b_1 x_1 + \cdots + b_n x_n$$
$$X\beta = y;\ \hat{\beta} = (X^\top X)^{-1} X^\top y$$

2. Нелинейная регрессия, поддающаяся линеаризации

$$y = b_0 + b_1 f_1(x_1, \dots, x_n) + \cdots + b_n f_n(x_1, \dots, x_n)$$

<u>Примеры:</u> $k = k_0 \exp\left(-\frac{E_a}{RT}\right) \Rightarrow \ln k = \ln k_0 - \frac{E_a}{RT}$; $c_p(T) = a + bT + cT^2$;

3. Нелинейная регрессия, не поддающаяся линеаризации

$$y = f(\beta_1, \dots, \beta_m, x_1, \dots, x_n) = f(\vec{\beta}, \vec{x})$$

<u>Пример:</u> $y = \frac{1}{1+\exp(-\beta_1 - \beta_2 x)}$

Для решения таких задач используют численные методы: Ньютона, Гаусса-Ньютона, градиентного спуска, Левенберга-Марквардта

Численные методы и МНК

Нелинейная система уравнений

$$\begin{cases} \varphi(\beta_1, \dots, \beta_k, x_{11}, x_{21}, \dots x_{m1}) = y_1 \\ \varphi(\beta_1, \dots, \beta_k, x_{12}, x_{22}, \dots, x_{m2}) = y_2 \\ \dots \\ \varphi(\beta_1, \dots, \beta_k, x_{1n}, x_{2n}, \dots, x_{mn}) = y_n \end{cases}$$

Как правило, полученную систему уравнений нельзя решить аналитическими методами

Нужно использовать численные методы

Минимизируемая сумма квадратов

$$F(\vec{\beta}) = \sum_i (\hat{y}_i - y_i)^2 = \sum_i \left(\varphi(\vec{\beta}, \vec{x}_i) - y_i\right)^2 = \sum_i f_i^2$$

Система уравнений для поиска минимума (ср. с линейным МНК)

$$\begin{cases} \dfrac{\partial F(\vec{\beta})}{\partial \beta_1} = 0 \\ \dots \\ \dfrac{\partial F(\vec{\beta})}{\partial \beta_m} = 0 \end{cases}$$

Рассматриваемые численные методы

1. Метод Ньютона
2. Метод Гаусса-Ньютона
3. Метод Левенберга-Марквардта
4. Метод доверительных областей с ломаным шагом

Метод Ньютона

Одномерный случай (нелинейное уравнения)

$$f(x) \approx f(x_0) + f'(x_0)(x - x_0) = 0$$

$$x = x_0 - \frac{f(x_0)}{f'(x_0)}$$

Многомерный случай (система нелинейных уравнений)

$$F(\vec{\beta}) \approx F(\vec{\beta}_0) + \sum_i \frac{\partial F(\vec{\beta}_0)}{\partial \beta_i}\left(\beta_i - \beta_i^{(0)}\right) = 0 \Rightarrow F(\vec{\beta}_0) + \nabla F(\vec{\beta}_0)\vec{p} = 0$$

Особенности метода Ньютона

1. Быстро сходится (квадратичная сходимость)
2. В случае нескольких корней может быть найден любой из них (зависит от начального приближения)
3. Скорость сходимости (и сама сходимость) может зависеть от начального приближения

Метод Ньютона: практическая реализация

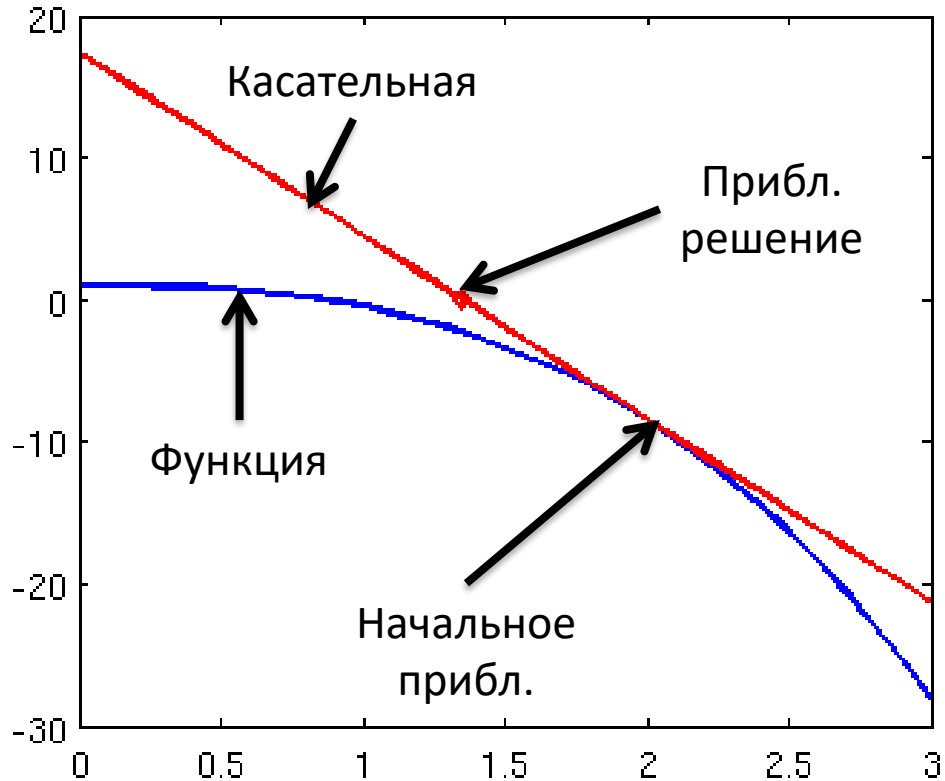
Уравнение

$$f(x) = \cos x - x^3 = 0$$

$$f'^{(x)} = -\sin x - 3x^2 = 0$$

Программа

```
format long;
delta = 1e-15;
x0 = 0.5;
x = x0 + 2*delta;
while abs(x-x0) >= delta
    x0 = x;
    f = cos(x0)-x0^3;
    df = -sin(x0)-3*x0^2;
    x = x0 - f/df;
    disp(x);
end
```




```
2.000000000000000
1.348055393079852
1.001262924310922
0.880591138294078
0.865691456412747
0.865474078978736
0.865474033101617
0.865474033101614
```

МНК и метод Гаусса-Ньютона

Система уравнений и метод Ньютона

$$\begin{cases} \dfrac{\partial F(\vec{\beta})}{\partial \beta_1} = 0 \\ \cdots \\ \dfrac{\partial F(\vec{\beta})}{\partial \beta_m} = 0 \end{cases} \Rightarrow \frac{\partial F(\vec{\beta})}{\partial \beta_i} \approx \frac{\partial F(\vec{\beta}^{(0)})}{\partial \beta_i} + \sum_j \frac{\partial F(\vec{\beta}^{(0)})}{\partial \beta_i \partial \beta_j}\left(\beta_j - \beta_j^{(0)}\right) = 0$$

Матричная запись и метод Гаусса-Ньютона

$$\nabla F(\vec{\beta}^{(0)}) + H(\vec{\beta}^{(0)})\vec{p} = 0 \Rightarrow J^\top(\vec{\beta}^{(0)}) f(\vec{\beta}^{(0)}) + J^\top(\vec{\beta}^{(0)}) J(\vec{\beta}^{(0)})\vec{p} = 0$$

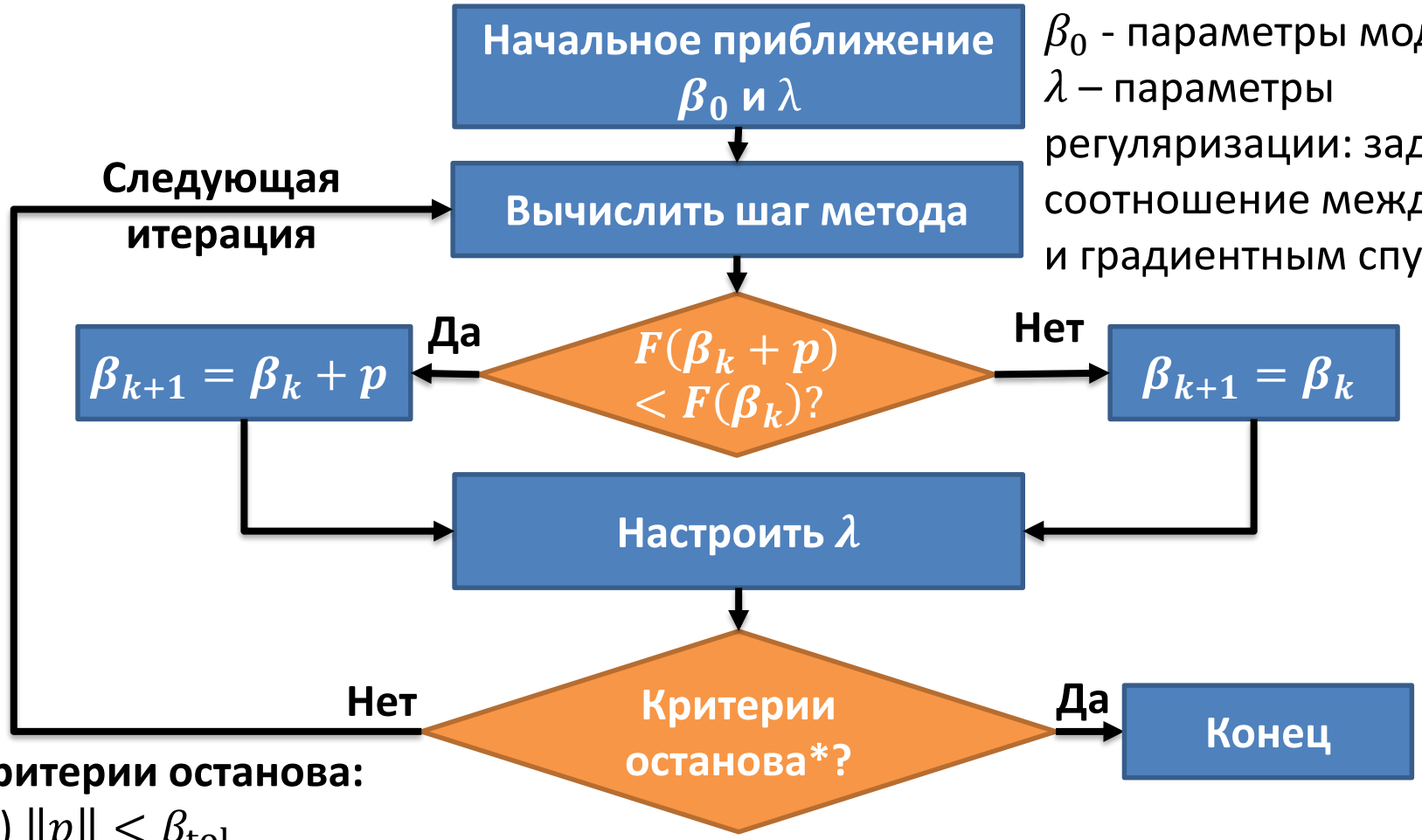
Градиент (∇F), матрицы Якоби (J) и Гессе (H)

$$(\nabla F)_i = \frac{\partial F}{\partial \beta_i} = \frac{\partial}{\partial \beta_i}\left(\sum_j f_j^2\right) = 2\sum_j f_j \frac{\partial f_j}{\partial \beta_i} = 2J^\top f; J_{ij} = \frac{\partial f_i}{\partial \beta_j} = \frac{\partial \varphi(\vec{\beta}, \vec{x_i})}{\partial \beta_j}$$

$$H_{ij} = \frac{\partial F(\beta)}{\partial \beta_i \partial \beta_j} = \frac{\partial}{\partial \beta_j}\left[2\sum_k f_k \frac{\partial f_k}{\partial \beta_i}\right] = 2\sum_k \left[\frac{\partial f_k}{\partial \beta_i}\frac{\partial f_k}{\partial \beta_j} + \underbrace{f_k \frac{\partial f_k}{\partial \beta_i \partial \beta_j}}_{\approx 0}\right] \approx 2\sum_k \frac{\partial f_k}{\partial \beta_i}\frac{\partial f_k}{\partial \beta_j} = 2J^\top J$$

Демпфирование метода Гаусса-Ньютона

В некоторых случаях метод Гаусса-Ньютона не сходится
Необходимо сочетать его с методом градиентного спуска



β_0 - параметры модели
λ – параметры регуляризации: задают соотношение между Г.-Н. и градиентным спуском

Критерии останова:

(а) $\|p\| < \beta_{\text{tol}}$

(б) $F(\beta_k) - F(\beta_{k+1}) < F_{\text{tol}}$

(в) $\lambda < \lambda_{\min}$ или $\lambda_{\max} < \lambda$?

Метод Левенберга-Марквардта

$$(J^\top J + \lambda I)\vec{p} = -J^\top f = -0.5\nabla F \text{(вариант 1)}$$
$$(J^\top J + \lambda \mathrm{diag}[J^\top J])\,\vec{p} = -J^\top f = -0.5\nabla F \text{(вариант 2)}$$

Является итерационным. При итерациях варьируется параметр λ:

1. Задать $\beta^{(0)}$, $\lambda_0 > 0$, $\nu > 1$ и $k = 1$. Задать $r = -1$.
2. Решить уравнение для $\beta^{(k-1)}$ и $\lambda_k = \lambda_{k-1}\nu^r$, получив шаг p_k
3. Если значение F не уменьшилось, то $r = r + 1$. Иначе $\beta^{(k)} = \beta^{(k-1)} + p_k$, $r = -1$, $k = k + 1$.
4. Если критерии останова не выполнены, то перейти к шагу 2

Сходится медленнее метода Гаусса-Ньютона, но менее требователен к начальному приближению. Широко применяется на практике.

Является комбинацией методов Гаусса-Ньютона и градиентного спуска

$\vec{p} = -\xi\nabla F\left(\vec{\beta}^{(0)}\right)$ (градиентный спуск)

$2J^\top J\vec{p} = -\nabla F$ (метод Гаусса-Ньютона)

Обозначения

$F = RSS(\beta) = \sum_i f_i^2$

$p_k = \beta^{(k)} - \beta^{(k-1)}$

$\lambda > 0$ – параметр регуляризации

$\nu > 1$ (обычно $\nu \approx 10$) – задаёт скорость варьирования λ

Метод доверительных окрестностей с ломаным шагом (Trust-region dogleg method)

Разложим функцию $\boldsymbol{F(\beta)}$ в ряд Тейлора в окрестностях точки $\boldsymbol{\beta}$

$$F(\beta + p) \approx L(p)$$
$$= F(\beta) + (\nabla F)^\top p + 0.5 p^\top H p$$
$$L(p) = F(\beta) + 2(J^\top f)^\top p + p^\top (J^\top J) p$$

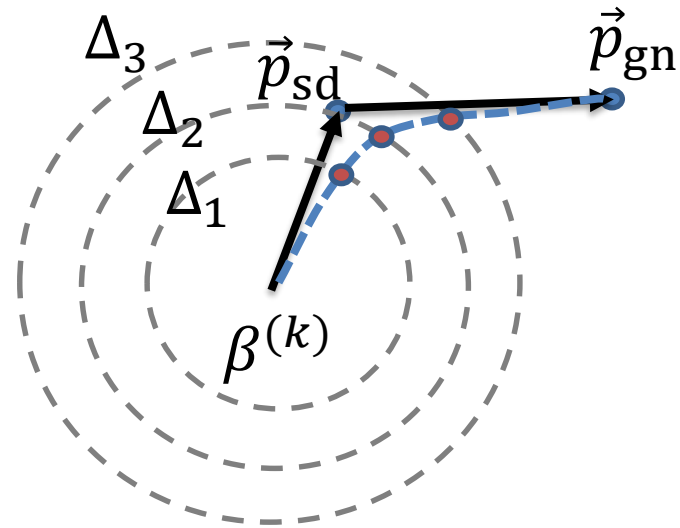
Глобальный минимум функции $L(p)$:

$$\frac{\partial L(p)}{\partial p_i} = 0 \Rightarrow \nabla F + Hp = 0$$

Для сходимости ищут минимум $L(p)$ внутри доверительной окрестности Δ:

$$\min_{\|p\|<\Delta} L(p)$$

Радиус Δ варьируют в зависимости от результатов предыдущего шага



Зависимость шага $\vec{p}$ от Δ

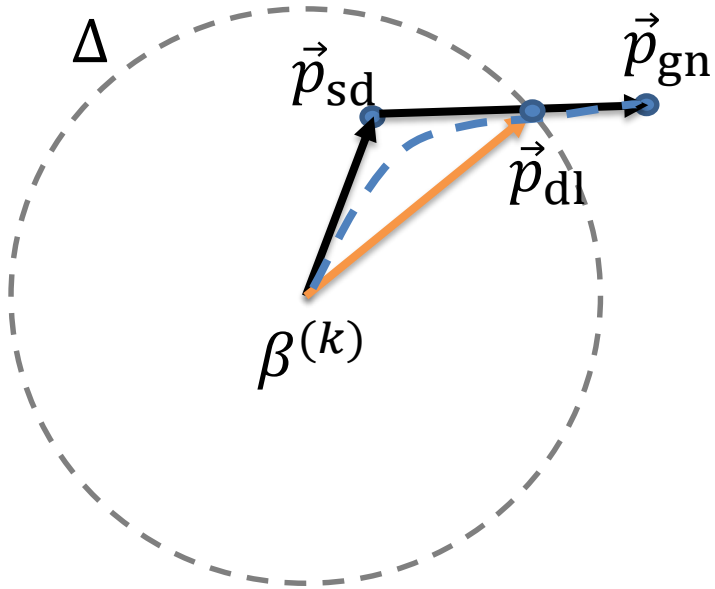
Доверительные окрестности

Ломаный шаг (приближение)

Точное нахождение шага с учётом границы окрестности сложно! Нужны приближения!

МДО с ломаным шагом (Trust-region dogleg method)

Ломаный шаг метода $\vec{p}_{\mathrm{dl}}$



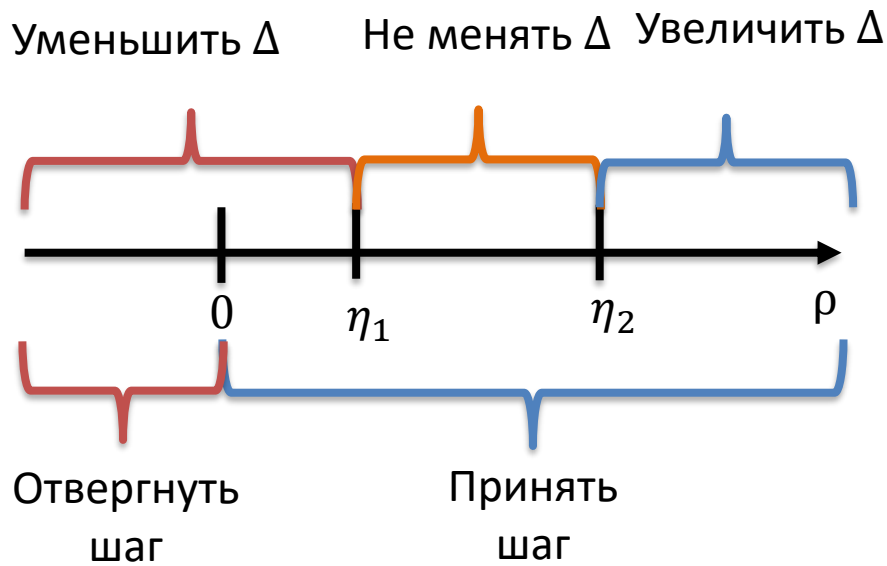
Если шаг не пересекает границы окрестности, то $p_{\mathrm{dl}} = p_{\mathrm{gn}}$

Свойства ломаного шага при положительно определённой H:

1. Значение $L(p)$ монотонно убывает при движении от $\beta^{(k)}$ до $\vec{p}_{\mathrm{gn}}$
2. Если пересекает границы окрестности, то в одной точке

Оценка качества шага метода $\vec{p}_{\mathrm{dl}}$
На каждой итерации

$$\rho = \frac{L(0) - L(p)}{F(\beta) - F(\beta + p)}$$



p_{gn} - шаг метода Гаусса-Ньютона
p_{sd} - шаг метода наискорейшего градиентного спуска

О функциях в GNU Octave и MATLAB

Функция – фрагмент программного кода, который можно вызывать из других частей программы. Имеет входные и выходные параметры (аргументы), а также свою область видимости для переменных.

Именованные функции

Хранятся в файлах. Синтаксис:

```
% Справочная информация
function [o1,…,om]=funcname(i1,…,in)
% Тело функции
end
```

Пример:

```
% SQREQ_ROOTS  Finds the roots
% of square equation
function [x1,x2] = sqreq_roots(a,b,c)
D = b.^2 - 4*a.*c;
x1 = (-b + sqrt(D)) ./ (2*a);
x2 = (-b - sqrt(D)) ./ (2*a);
end
```

Анонимные функции

Хранятся в переменных. Синтаксис:

```
func = @(arg1,…,argn) expression
```

Примеры:

```
sqr = @(x) x.^2
len = @(x,y) sqrt(x.^2 + y.^2)
```

См. также: вложенные функции (nested function), MEX-файлы, объектно-ориентированное программирование

Нелинейная регрессия: практическая реализация

Шаг 1. Задание аппроксимирующей функции

$$y = \beta_1 + \beta_2 \exp(-\beta_3 x)$$

Шаг 2. Задание якобиана

$$\begin{cases} \dfrac{\partial y_i}{\partial \beta_1} = 1 \\ \dfrac{\partial y_i}{\partial \beta_2} = \exp(-\beta_3 x_i) \\ \dfrac{\partial y_i}{\partial \beta_3} = -x\beta_2 \exp(-\beta_3 x_i) \end{cases}$$

Шаг 3. Запись на языке Octave

```
b0 = [10 10 10];
[b, res, J] = ...
  lsqfit_lm(X,Y,@func, b0);
[db, b_lb, b_ub, sb] = ...
  lsqfit_ci(b, res, J);

function [F, J] = func(b, x)
F = b(1) + b(2)*exp(-b(3)*x);
if nargout == 2
  df_db1 = ones(size(x));
  df_db2 = exp(-b(3)*x);
  df_db3 = -b(2)*exp(-
b(3)*x).*x;
  J = [df_db1 df_db2 df_db3];
end
```

Шаг 4. Подбор начального приближение и визуализация результатов

Нелинейная регрессия: доверительные интервалы

Линейная регрессия

$$s^2_{\hat{\boldsymbol{\beta}}} = \hat{\sigma}^2 \mathrm{diag}[(X^\top X)^{-1}]$$

$$X = \begin{pmatrix} 1 & x_1^{(1)} & \cdots & x_m^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(n)} & \cdots & x_m^{(n)} \end{pmatrix}$$

Нелинейная регрессия

$$s^2_{\hat{\boldsymbol{\beta}}} = \hat{\sigma}^2 \mathrm{diag}[(J^\top J)^{-1}]$$

diag – главная диагональ матрицы

Матрица Якоби J и ошибка регрессии $\hat{\sigma}^{\mathbf{2}}$ рассчитываются в точке $\hat{\boldsymbol{\beta}}$

Исходная система уравнений

$$\begin{cases} \varphi(\beta, x_1) = y_1 \\ \cdots \\ \varphi(\beta, x_n) = y_n \end{cases}$$

Результат линеаризации в векторной форме

$$J(\hat{\beta})(\beta - \hat{\beta}) = y - \hat{y} = e$$

Разложение в ряд Тейлора

$$\varphi(\beta, x) = \varphi(\hat{\beta}, x) + \sum_i \frac{\partial \varphi(\hat{\beta}, x)}{\partial \hat{\beta}_i} (\beta_i - \hat{\beta}_i)$$

$J(\hat{\beta})$ - аналог матрицы X в линейной регрессии

Рекомендуемая литература

[1] *Levenberg K.* A method for the solution of certain non-linear problems in least squares // Quarterly of Applied Mathematics. 1944. V. 2. N 2. P.164-168. https://doi.org/10.1090/qam/10666

[2] *Marquardt D. W.* An Algorithm for Least-Squares Estimation of Nonlinear Parameters //, J. Soc. Indust. Appl. Math. 1963. V. 11. N 2. P. 431-441. http://dx.doi.org/10.1137/0111030

[3] *Lourakis M. I. A., Argyros A. A.* Is Levenberg-Marquardt the Most Efficient Optimization Algorithm for Implementing Bundle Adjustment? 2005. ICCV'05 Proceedings of the Tenth IEEE International Conference on Computer Vision. V. 2. Beijing, China. P.1526-1531. http://dx.doi.org/10.1109/ICCV.2005.128.

[4] *Nocedal J., Wright, S. J.* Numerical Optimization. 2006. Springer New York. http://dx.doi.org/10.1007/978-0-387-40065-5, ISBN 978-0-387-30303-1.

[5] *Гилл Ф., Мюррей У., Райт М.* Практическая оптимизация. Пер. с англ. Москва, «Мир», 1985. 509 с.

МДО с ломаным шагом: некоторые формулы

$$F(\beta + p) \approx L(p) = F(\beta) + \sum_i \frac{\partial F(\beta)}{\partial \beta_i} p_i + \frac{1}{2} \sum_i \sum_j \frac{\partial^2 F(\beta)}{\partial \beta_i \partial \beta_j} p_i p_j$$

$$L(p) = F(\beta) + (\nabla F)^\top p + 0.5 p^\top H p = F(\beta) + 2(J^\top f)^\top p + p^\top (J^\top J) p$$

Минимум функции L (ищется **методом Гаусса-Ньютона**):

$$\frac{\partial L(p)}{\partial p_i} = \frac{\partial F(\beta)}{\partial \beta_i} + \sum_j \frac{\partial^2 F(\beta)}{\partial \beta_i \partial \beta_j} p_j = 0 \Rightarrow \nabla F + Hp = 0 \Rightarrow p_{gn} = -(J^\top J)^{-1} J^\top f$$

Матрица Гессе $\boldsymbol{H}$ – положительно определённая

$$H = 2J^\top J$$

$$p^\top H p = 2(Jp)^\top Jp = 2\|Jp\|^2 \geq 0$$

Метод наискорейшего спуска

$$p_{sd} = -\xi \nabla F(\beta) = -2\xi g;\ g = J^\top f$$

$$\frac{\partial L(\xi)}{\partial \xi} = -4\|\mathrm{g}\|^2 + 4\xi g^\top H g = 0$$

$$\xi = \frac{\|g\|^2}{g^\top H g} = \frac{\|g\|^2}{2\|Jg\|^2}$$