Анализ данных

Хашин С.И.
http://math.ivanovo.ac.ru/dalgebra/Khashin/index.html

Ивановский университет

.

Линейная регрессия

.

Иваново-2024

План

Линейные модели

Ответ ищем в виде:

$$a(x) = w_0 + \sum_{i=1}^{k} w_i x_i.$$

Параметрами модели являются веса или коэффициенты w_j .
Вес w_0 также называется свободным коэффициентом или сдвигом (bias).
Формулу можно записать через скалярное произведение:

$$a(x) = w_0 + (w, x)$$

или в виде:

$$a(x) = (w', x')$$

где $w' = (w_0, \ldots, w_k)$, $x' = (1, x_1, \ldots, x_k)$.

Измерение ошибки

MSE (Mean Square Error):

$$MSE(a, X) = \frac{1}{n} \sum (a(x_i) - y_i)^2.$$

RMSE (Mean Square Error):

$$RMSE(a, X) = \sqrt{\frac{1}{n} \sum (a(x_i) - y_i)^2}.$$

MAE (Mean Absolute Error)

$$MAE(a, X) = \frac{1}{n} \sum |a(x_i) - y_i| .$$

Измерение ошибки

MAPE (mean absolute percentage error):

$$MAPE(a, X) = \frac{1}{n} \sum \frac{|a(x_i) - y_i|}{y_i}.$$

$$???(a, X) = \frac{1}{n} \sum \left(\frac{a(x_i) - y_i}{y_i} \right)^2.$$

Линейная регрессия, dim=0

Имеется n студентов, возраст каждого известен: $(x_1, \ldots, x_n)$.
Найти w_0 — средний возраст студентов группы.

- Среднее арифметическое;
- Среднее геометрическое;
- Середина между максимумом и минимумом;
- Медиана;
- И др..

Линейная регрессия, dim=0

Рассмотрим квадратичную погрешность:

$$S(w_0) = \sum_{i=1}^{n} (w_0 - x_i)^2.$$

Тогда

$$\frac{\partial S}{\partial w_0} = 2 \sum_{i=1}^{n} (w_0 - x_i) = 2(nw_0 - \sum x_i),$$

отсюда находим: $w_0 = (\sum x_i)/N$.
Пусть $n = 20$ и $(x_i) = (20, 20, \ldots, 20, 2003)$. Тогда
$w_0 = (\sum x_i)/N = 119.15$.

Линейная регрессия, dim=0

Пусть $n = 20$ и $(x_i) = (20, 20, \ldots, 20, 2003)$.
Будем минимизировать величину

$$S_1(a_0) = \sum_{i=1}^{n} |w_0 - x_i|.$$

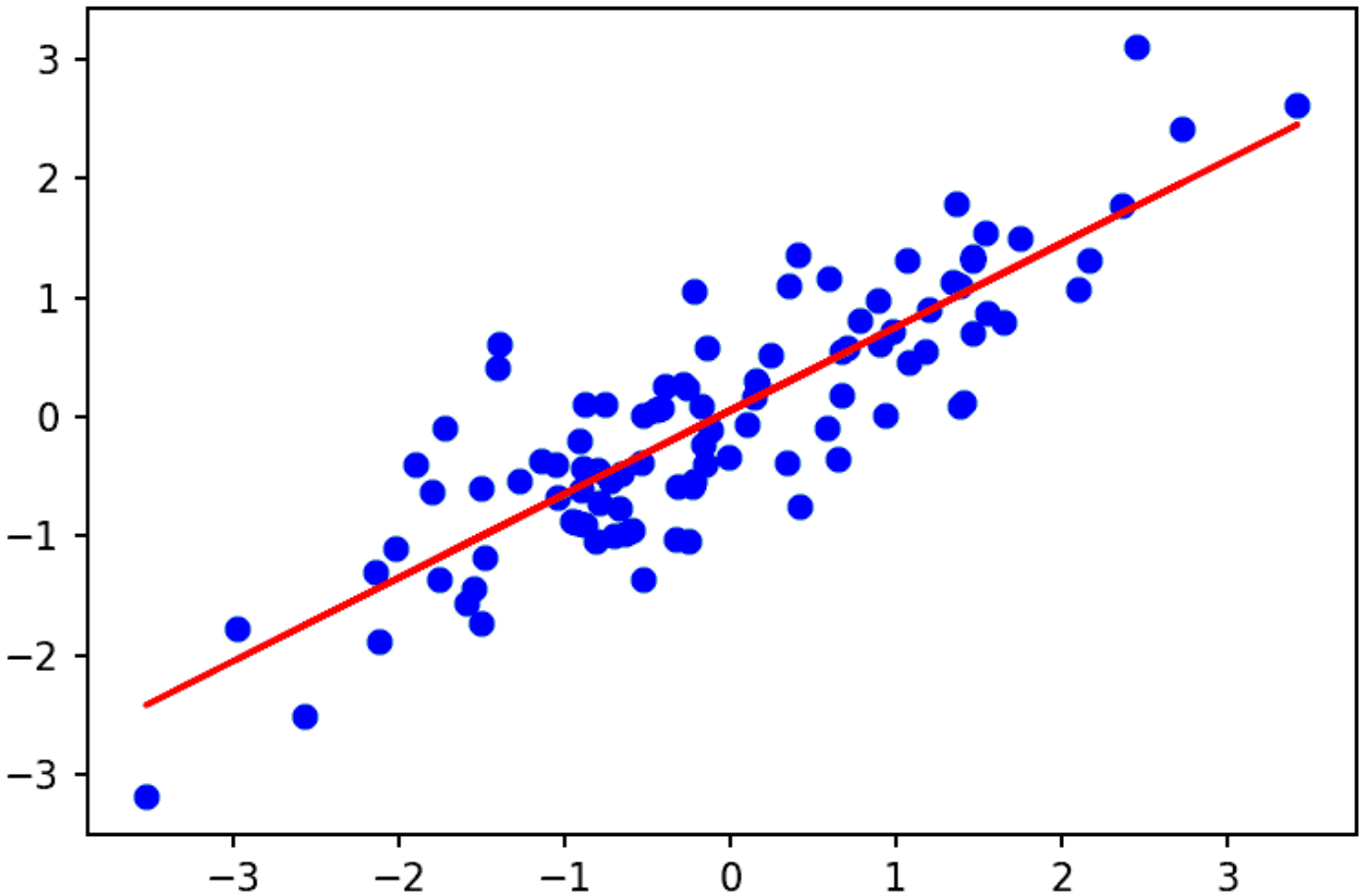
В нашем примере будем считать, что $20 \leq a_0 \leq 2003$, тогда

$$S_1(w_0) = 24(w_0 - 20) + (2003 - w_0) = 23w_0 + 1523.$$

Минимум погрешности достигается при наименьшем $w_0 = 20$.

Линейная регрессия, dim=1

Пусть даны n значений функции $y = f(x)$ в отдельных точках: $f(x_1) = y_1, \ldots, f(x_n) = y_n$. Мы хотим аппроксимировать функцию линейной: $f(x) \approx w_0 + w_1 x$.



Линейная регрессия, dim=1

Пусть даны n значений функции $y = f(x)$ в отдельных точках: $f(x_1) = y_1, \ldots, f(x_n) = y_n$. Мы хотим аппроксимировать функцию линейной: $f(x) \approx w_0 + w_1 x$. Выпишем квадратичную погрешность:

$$S(w_0, w_1) = (w_0 + w_1 x_1 - y_1)^2 + \cdots + (w_0 + w_1 x_k - y_k)^2.$$

$$\frac{1}{2}\frac{\partial S}{\partial w_0} = \sum_{i=1}^{n}(w_0 + w_1 x_i - y_i) = nw_0 + w_1 \sum_i x_{i1} - \sum_i y_i.$$

$$\frac{1}{2}\frac{\partial S}{\partial w_1} = \sum_{i=1}^{n} x_i(w_0 + w_1 x_i - y_i) = w_0 \sum_i x_i + w_1 \sum_i x_i^2 - \sum_i y_i x_i.$$

Линейная регрессия, dim=1

Введем обозначения:

$$a_{00} = n, \qquad a_{01} = a_{10} = \sum_{i=1}^{n} x_i, \qquad a_{11} = \sum_{i=1}^{n} x_i^2,$$

$$b_0 = \sum_{i=1}^{n} y_i, \qquad b_1 = \sum_{i=1}^{n} y_i x_i.$$

Тогда систему уравнений $\partial S/\partial w_j = 0$ можно записать так:

$$\begin{aligned} a_{00} w_0 + a_{01} w_1 &= b_0, \\ a_{10} w_0 + a_{11} w_1 &= b_1. \end{aligned}$$

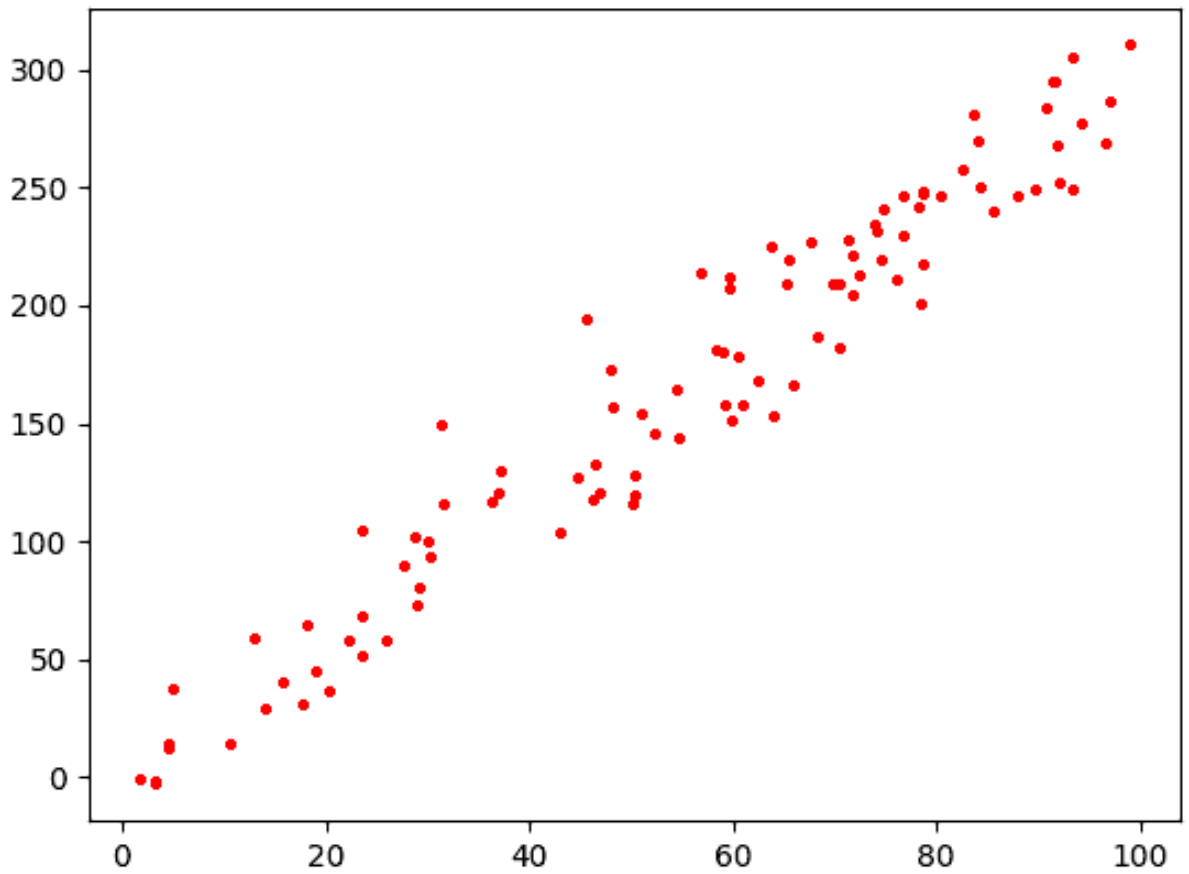
Отсюда:

$$w_0 = (b_0 a_{11} - b_1 a_{10})/D, \;\; w_1 = (b_1 a_{00} - b_0 a_{10})/D,$$

где $D = a_{00} a_{11} - a_{10}^2$.

Линейная регрессия, dim=1, Питон

Данные в файле lin_regr1.csv:



Линейная регрессия, dim=1, Питон

Данные в файле lin_regr1.csv (100 строк):

```
x,y
  59.7356, 212.2550
  58.3187, 181.3538
  61.0353, 158.2258
  94.2480, 277.5547
 ...
```

Линейная регрессия, dim=1, Питон

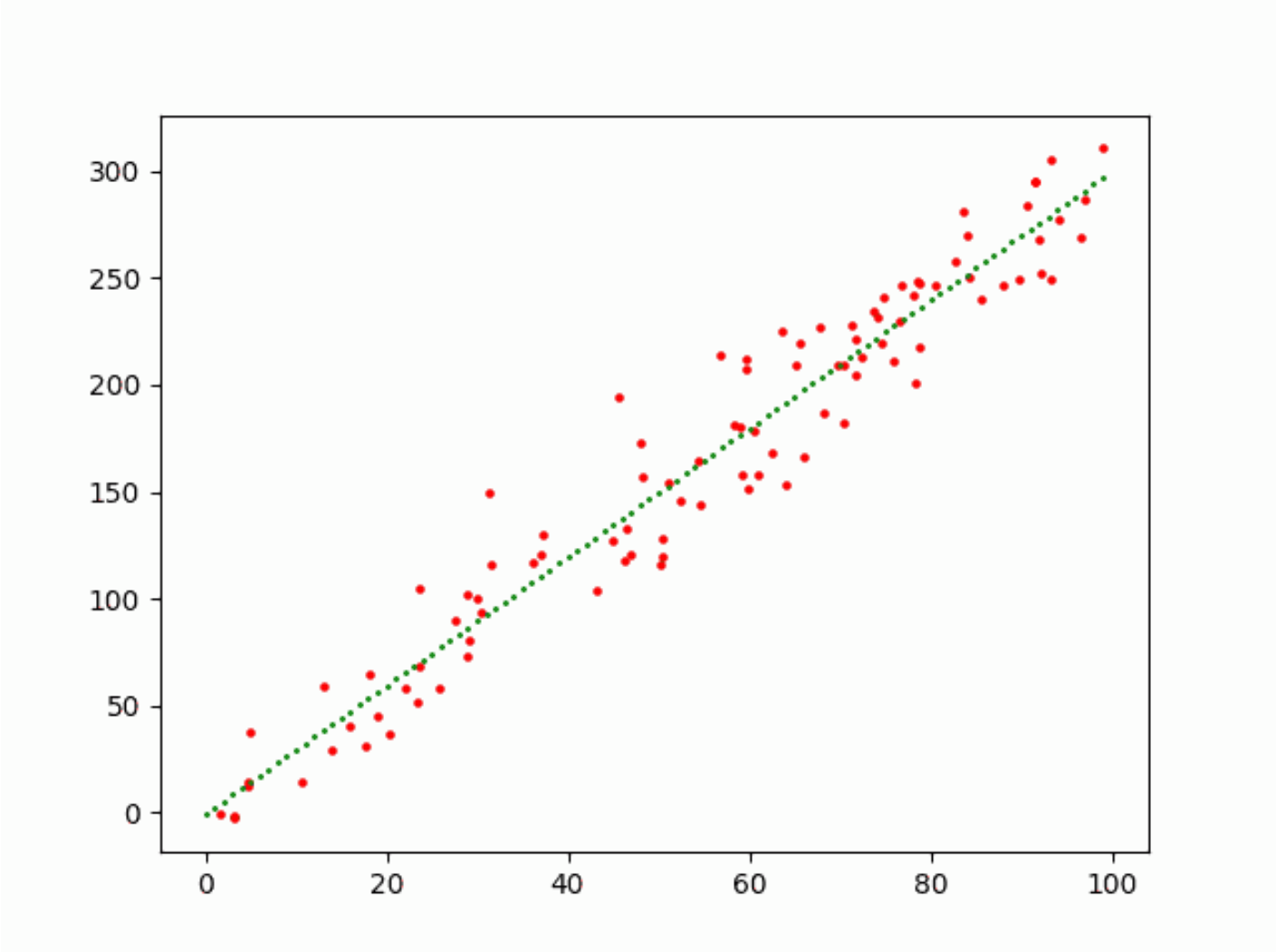
```
csv = np.loadtxt('lin_regr1.csv', skiprows=1,
      delimiter=',')
x, y = csv[:,0],csv[:,1]
a00, a10, a11 =  len(x), np.sum(x), np.sum(x*x)
b0, b1 = np.sum(y), np.sum(x*y)
De = a00*a11 - a10**2
w0 = (b0*a11-b1*a10)/De
w1 = (b1*a00-b0*a10)/De
print(f'w0={w0:8.5f}, w1={w1:8.5f}')
```

Ответ:

```
w0=-0.79924, w1= 3.00686
```

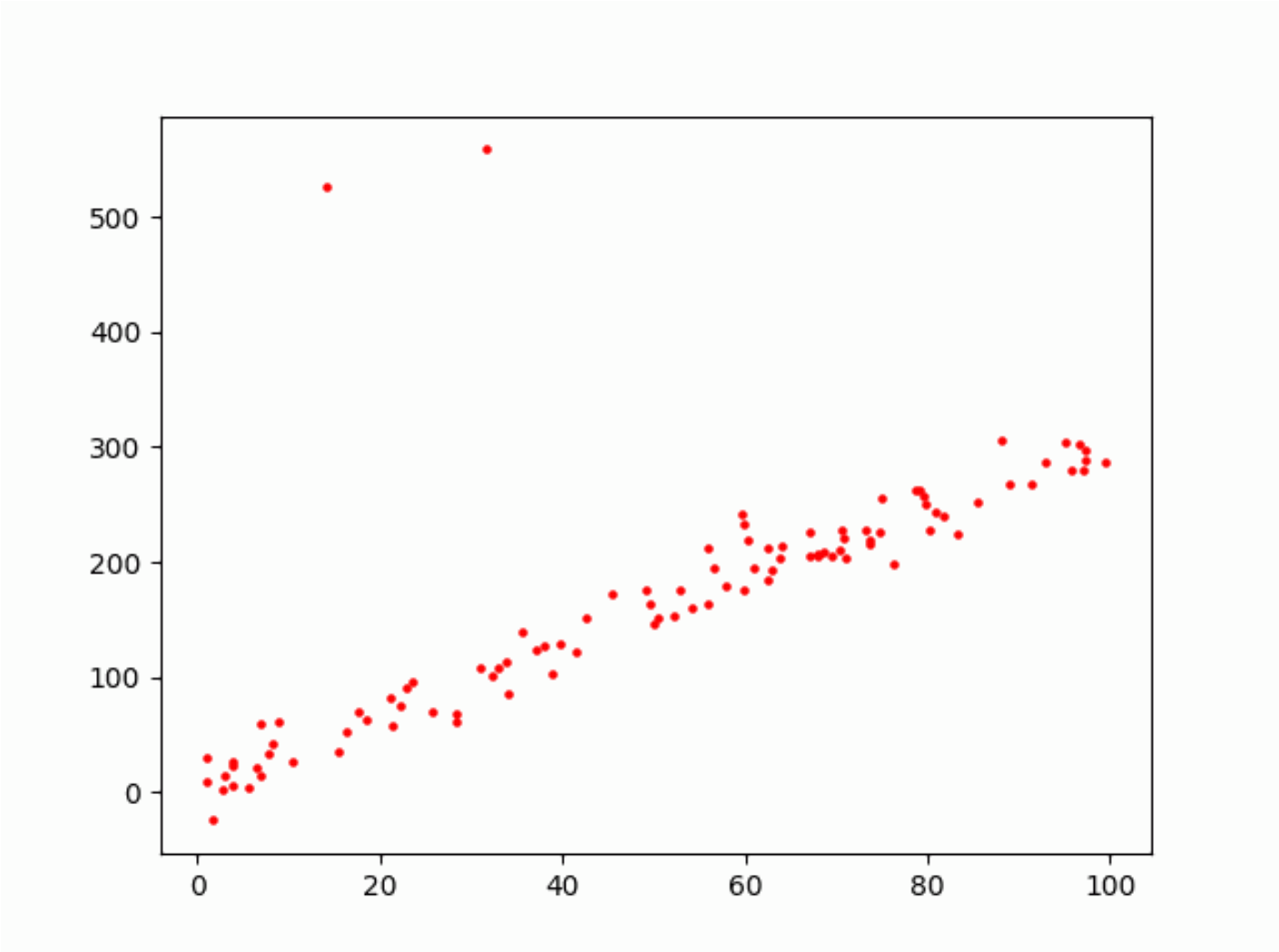
Линейная регрессия, dim=1, Питон

w0=-0.79924, w1= 3.00686



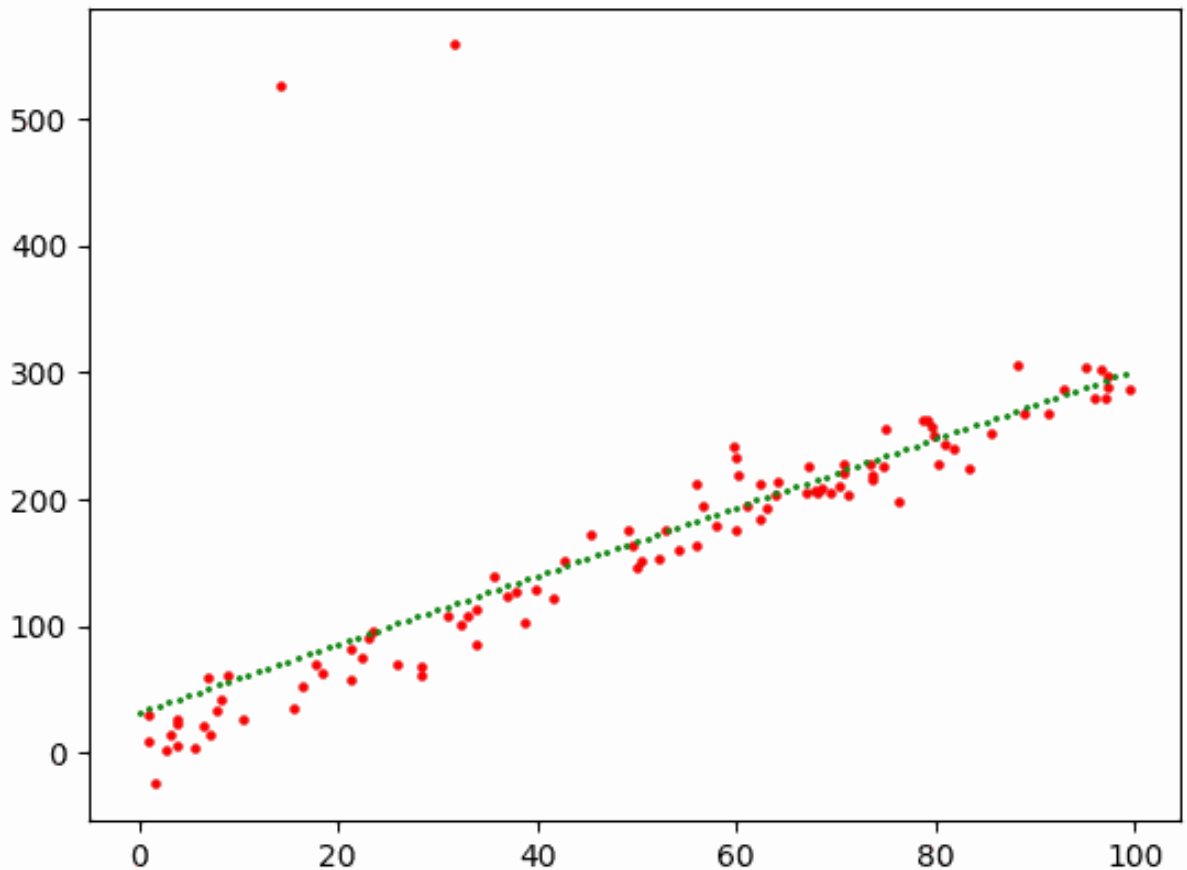
Испорченные данные

Данные в файле lin_regr1a.csv:



Испорченные данные

Данные в файле lin_regr1a.csv:
w0=31.74447, w1= 2.70062



Среднеквадратичное отклонение

Вычисления σ:

```
def sigm(x):
    k = len(x)
    s1 = np.sum(x)
    s2 = np.sum(x**2)
    return np.sqrt(s2/k - (s1/k)**2)
```

На самом всё просто, функция есть готовая:

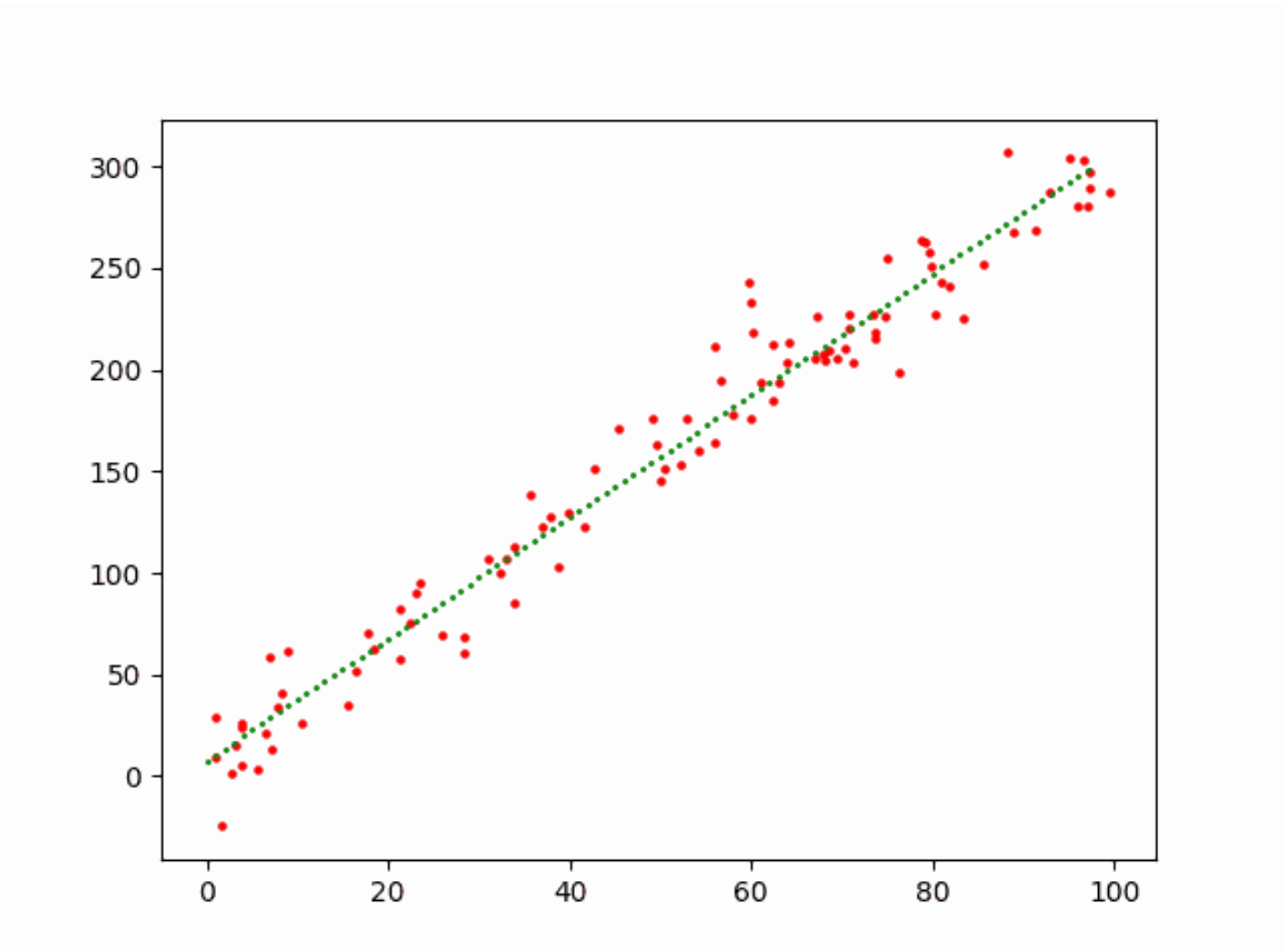
```
sigma =  x.std() # Стандартное отклонение
```

Линейная регрессия, чистка

```
w0,w1 = lin_reg1(x,y)
print(f'w0={w0:8.5f}, w1={w1:8.5f}')
err = y - (w0+w1*x)      # погрешность
sigma = err.std()
print(f'sigma={sigma:8.3f}')
a = np.where(abs(err)<3*sigma)[0]
print(a)
x2,y2 = x[a],y[a]
w0,w1 = lin_reg1(x2,y2)
print(f'w0={w0:8.5f}, w1={w1:8.5f}')
plt.scatter(x2, y2, s=5, color='red')
xx = np.arange(len(x2))
yy = w0+xx*w1
plt.scatter(xx, yy, s=1, color='green')
```

Чистка

w0=31.74447, w1= 2.70062
sigma= 66.863
w0= 7.73021, w1= 2.99563



Линейная регрессия

Мы хотим аппроксимировать действительную функцию $f(x)$, заданную в отдельных точках в d-мерного пространства:

$$f(x_i) = y_i, \ \ i = 1, \dots n,$$

где $x_i \in \mathbb{R}^d, y_i \in \mathbb{R}$.
Попытаемся аппроксимировать её линейной функцией:

$$a(x) = a(x_1, \dots, x_d) = w_0 + w_1 x_1 + \dots w_d x_d.$$

Линейная регрессия

Квадрат среднеквадратичной погрешности:

$$S = \sum_{i=1}^{n} (w_0 + w_1 x_{i1} + \dots w_d x_{id} - y_i)^2.$$

$$\frac{1}{2}\frac{\partial S}{\partial w_0} = \sum_{i=1}^{n} (w_0 + w_1 x_{i1} + \dots w_d x_{id} - y_i)$$

$$= n w_0 + w_1 \sum_i x_{i1} + \dots + w_d \sum_i x_{id} - \sum_i y_i.$$

При $j > 0$:

$$\frac{1}{2}\frac{\partial S}{\partial w_j} = \sum_{i=1}^{n} x_{ij} (w_0 + w_1 x_{i1} + \dots w_d x_{id} - y_i)$$

$$= w_0 \sum_i x_{ij} + w_1 \sum_i x_{i1} x_{ij} + \dots + w_d \sum_i x_{id} x_{ij} - \sum_i y_i x_{ij}.$$

Линейная регрессия

Введем обозначения:

$$a_{00} = n,$$

$$a_{0j} = a_{j0} = \sum_{i=1}^{n} x_{ij},\ 0 < j \leq d,$$

$$a_{ji} = a_{ij} = \sum_{i=1}^{n} x_{ij}x_{ij},\ 0 < i, j \leq d,$$

$$b_0 = \sum_{i=1}^{n} y_i,$$

$$b_j = \sum_{i=1}^{n} y_i x_{ij},\ 0 < j \leq d.$$

Линейная регрессия

Тогда систему уравнений $\partial S/\partial w_j = 0$ можно записать так:

$$\begin{aligned} a_{00}w_0 + a_{01}w_1 + \ldots a_{0d}w_d &= b_0, \\ a_{10}w_0 + a_{11}w_1 + \ldots a_{1d}w_d &= b_1, \\ &\ldots \\ a_{d0}w_0 + a_{d1}w_1 + \ldots a_{dd}w_d &= b_d, \end{aligned}$$

или, в матричном виде $Aw = B$, где $A = (a_{ij})$, $B = (b_j)$.

Линейная регрессия

Запишем исходные данные в матричном виде:

$$X = \begin{pmatrix} x_{11} & \dots & x_{1d}, \\ & \dots & \\ x_{n1} & \dots & x_{nd}, \end{pmatrix}, Y = \begin{pmatrix} y_1 \\ \dots \\ y_n \end{pmatrix}.$$

Дополним матрицу X единичным столбцом:

$$\tilde{X} = \begin{pmatrix} 1 & x_{11} & \dots & x_{1d}, \\ & & \dots & \\ 1 & x_{n1} & \dots & x_{nd}, \end{pmatrix}.$$

Линейная регрессия

Тогда

$$A = \tilde{X}^t \cdot X = \begin{pmatrix} 1 & \dots & 1, \\ x_{11} & \dots & x_{n1}, \\ & \dots & \\ x_{1d} & \dots & x_{dn}, \end{pmatrix} \begin{pmatrix} 1 & x_{11} & \dots & x_{1d}, \\ & & \dots & \\ 1 & x_{n1} & \dots & x_{nd}, \end{pmatrix},$$

$$B = \tilde{X}^t \cdot Y = \begin{pmatrix} 1 & \dots & 1, \\ x_{11} & \dots & x_{n1}, \\ & \dots & \\ x_{1d} & \dots & x_{dn}, \end{pmatrix} \begin{pmatrix} y_1 \\ \dots \\ y_n \end{pmatrix},$$

lin_regr4.csv

```
y, x0, x1, x2, x3
  89.8445,   8.3165,   1.3775,   8.7254,   8.4110
  82.3274,   8.9453,   4.2634,   6.3009,   4.9156
  66.5088,   3.9030,   2.0224,   4.2700,   5.2507
  46.9930,   4.7485,   5.2998,   0.1964,   4.3302
 109.6883,   9.7637,   5.7978,   6.9418,   8.8990
  80.3742,   1.8671,   9.9772,   3.9436,   7.2508
  ...
```

Линейная регрессия, dim=4

```
csv = np.loadtxt('lin_regr4.csv', skiprows=1,
      delimiter=',')
y = csv[:,0].copy()
x = csv
x[:,0] = 1
print('y: ', y.shape)
print('x: ', x.shape)
A = x.transpose().dot(x)
print(A, '=A')
b = x.transpose().dot(y)
print(b, '=b')
w = np.linalg.solve(A,b)
print(w, '=w')
```

Линейная регрессия, dim=4

```
def lin_reg(x,y): # x уже содержит столбец из 1
    #x = np.hstack((np.ones(len(x)).reshape((-1,1)), x))
    A = x.transpose().dot(x)
    b = x.transpose().dot(y)
    return np.linalg.solve(A,b)

w = lin_reg(x,y);  print(w, '=w')
err = y - x.dot(w)      # погрешность
sigma = err.std()
print(f'sigma={sigma:8.3f}')
a = np.where(abs(err)<3*sigma)[0]
x2,y2 = x[a],y[a]
w2 = lin_reg(x2,y2);  print(w2, '=w2')
```

Регуляризация

Иногда при решении системы уравнений
`np.linalg.solve(AtA,Atb)`
матрица AtA оказывается вырождена ($det(AtA) = 0$ или «почти вырождена», то есть $abs(det(AtA))$ очень мал. Тогда решение вообще не будет найдено, или будет плохое.
В этом случае применяем «регуляризацию Тихонова», а именно, к квадратной матрице AtA прибавим единичную матрицу с малым коэффициентом
$eps = 1e-6 * abs(AtA).max()$.

```
eps = 1e-8*abs(AtA).max()
AtA += eps*np.eye(len(AtA))
```