1　基本的な分析手法

機械学習についての概略の基本的な分析手法を学んでいく．まず，基本的な分析手法は４つに分類される．

1. 線形回帰分析
2. ロジスティック回帰分析
3. 因子分析
4. クラスター分析

今回は線形回帰分析について学んだ．線形回帰分析とは複数の変数における相関関係を直線モデルによって説明しようとする分析手法のことである．
下記の図が使用するデータである．

顧客番号	年齢 (x)	年間支出額 (y)
1	32	40,000
2	23	38,000
3	39	46,000
4	47	48,000

表 1: 顧客の年齢とその人が年間にかける化粧品代のデータ

図の作成及び計算は Python を用いた．

2　線形回帰分析

顧客の年齢と年間支出額がなんらかの関係があるのかを考えていく．関係を分かりやすくするため，表 1 のデータをグラフにすると図 1 のようになる．

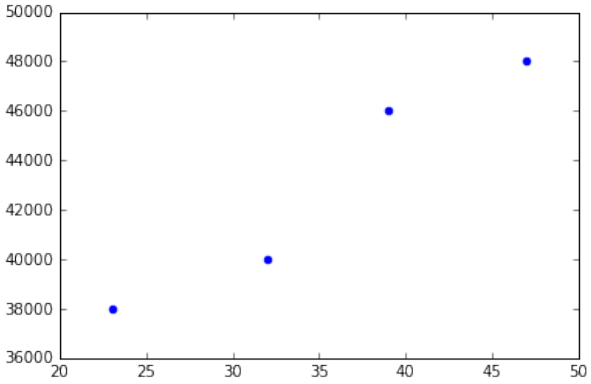

図 1: 表 1 の散布図

この際用いたソースコードが下記になる

ソースコード 1: 図 1 の作成に用いたもの

```
1  import numpy as np
2  import matplotlib.pyplot as plt
3
4  x = np.array([32, 23, 39, 47])
5  y = np.array([40000, 38000, 46000, 48000])
6
7  plt.scatter(x, y, color = 'blue')
8  plt.show()
```

図 1 でプロットされたデータに一本の近似した直線を引くことによって，年齢と年間支出額を予測する．直線のグラフは，$y = \beta_1 x + \beta_0$ で表される．このような変数 x と y の間の関係のとき，x と y は線形関係にあるという．

年齢 (x) が原因で，年間支出額 (y) が結果と考えると，y 方向に誤差が出ていると考えられる．そのため垂直方向の距離を誤差とする．この際，最小二乗法を使い，誤差を最小に近づける．

まず，グラフ $y = \beta_1 x + \beta_0$ の予想値を考える．顧客番号 2 は年齢 (x) が 23 で年間支出額が 38,000 円である．

年齢 (x) に 23 に代入すると，$y = 23\beta_1 + \beta_0$ になる．そこに実際の購入額である 38,000 円を y に代入すると，$38,000 = 23\beta_1 + \beta_0$ になる．さらに式を変形して $0 = 23\beta_1 + \beta_0 - 38,000$ にし，正と負の関係性を無視できるよう，二乗すると $0 = (23\beta_1 + \beta_0 - 38,000)^2$ になる．

これにより，それぞれの残差のみに注目できるようになった．同様に３つのデータについても，差の二乗を計算し，それらを合計すると，β_1 と β_0 の関係式になる．これを β_1 と β_0 を調整して最小に近づける．

以下が式になる．

$$(32\beta_1+\beta_0-40,000)^2+(23\beta_1+\beta_0-38,000)^2+(39\beta_1+\beta_0-46,000)^2+(47\beta_1+\beta_0+48,000)^2 \geqq 0$$

この式により，

$$\beta_1 \fallingdotseq 450.839 \quad \beta_0 \fallingdotseq 27107.913$$

が求まる．

よって，$y = \beta_1 x + \beta_0$ が

$$y \fallingdotseq 450.839x + 27107.913$$

となり，近似した 1 本の直線が求められた．

以下が図 1 に近似した 1 本の直線を加えたものである．

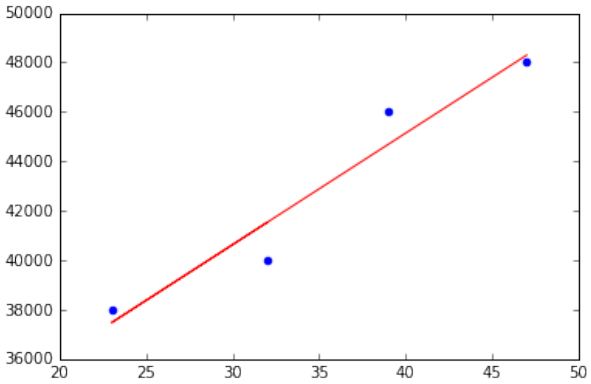

図 2: 近似した 1 本の直線を追加したもの

図２を作る際，用いたソースコードが下記になる．

ソースコード 2: 図２の作成に用いたもの

```
1  import numpy as np
2  import matplotlib.pyplot as plt
3
4  x = np.array( [32, 23, 39, 47] )
5  y = np.array( [40000, 38000, 46000, 48000] )
6
7  plt.scatter(x, y, color = 'blue')
8  xdata = np.vstack( [x, np.ones(len(x))] ).T
9  b1, b0 = np.linalg.lstsq(xdata, y)[0]
10
11 plt.scatter(x, y, color = 'blue')
12 plt.plot(x, (b1 * x + b0), color = 'red')
13 plt.show()
```

ソースコード 2 では．numpy.liang.lstsg 関数を使い，行列 xdata，y を代入し，$(x\beta_1 + \beta_0 - y)^2$ を最小にする β_1(b1)，β_0(b0) を求める．

xdata は numpy.liang.lstsg で x の値が扱えるように，全要素が 1 の配列を加えたものである．

$$xdata = \begin{pmatrix} 32 & 1 \\ 12 & 1 \\ 39 & 1 \\ 47 & 1 \end{pmatrix}$$

np.linalg.lstsq 関数に xdata, y を代入した時，出力される値が以下になる．

```
1  np.linalg.lstsq(xdata, y) = ( x の配列, 残差, 階数, 特異値の配列 )
2                            = ( array ( [ 450.83932854,  27107.91366906 ] ),
3                                array ( [ 4431654.67625901 ] ),
4                                2,
5                                array ( [ 72.71013252,  0.48644497] ) )
```

求めているのは，返り値の 1 つ目の要素なので [0] で一つ目を指定して b1,b0 に代入する．

最後に，$y = \beta_1 x + \beta_0$ に相当する $y = b1 * x + b0$ を $plt.plot(x, (b1 * x + b0))$ で描写する．

参考文献

[1] NumPy Reference
https://docs.scipy.org/doc/numpy/reference/index.html

[2] 解析情報グループ 最小二乗法
http://kaiseki-web.lhd.nifs.ac.jp/documents/Python/leastmean.htm