2024/9/26　AI R&D Center

学習班ガイダンス

やりたいことを、できることに。
湘南工科大学
SHONAN INSTITUTE OF TECHNOLOGY

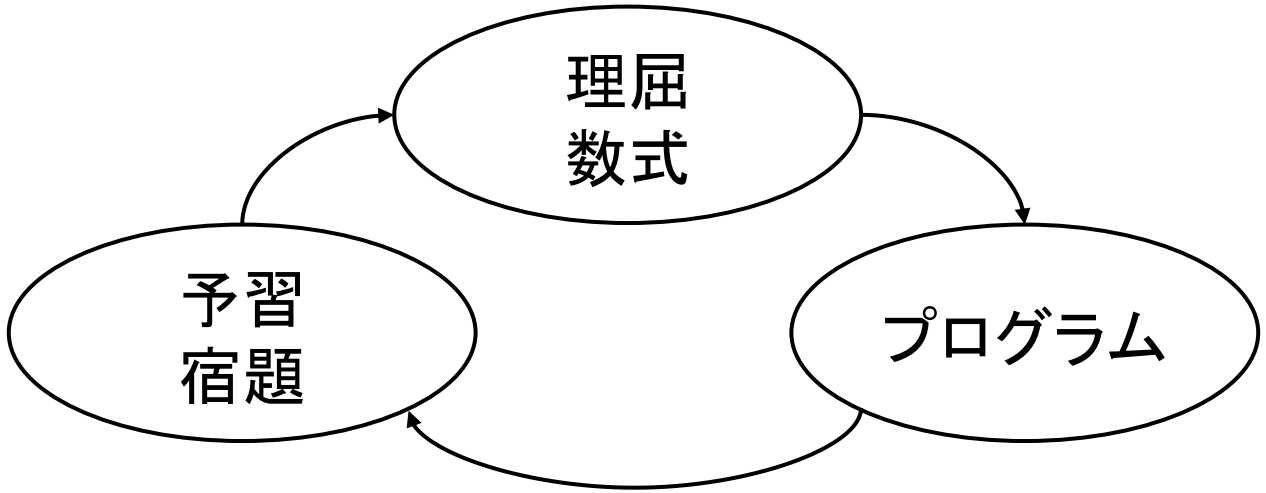
AI R&D CENTER

目的

自分自身のオリジナルのAIを作るための基礎を身につける

基礎を理解していると，

- 他人の発表が理解できる
- 目標が変わっても対応できる
- 質問対応の自信になる



全員の共通認識にするため，再度基礎の内容から実施する

実施内容

宿題の解説+理屈の説明+プログラムの作成+宿題

日付	内容	日付	内容
9/26	ガイダンス	11/21	セマンティックセグメンテーション 画像生成
10/3	単回帰，線形回帰	11/28	画像のAttention
10/10	重回帰，ロジスティック回帰	12/5	再帰的ニューラルネットワーク
10/17	多項式回帰，過学習，正則化	12/12	文章分類，文章生成
10/24	2層のニューラルネットワーク	12/19	Transformer，BERT
10/31	多層のニューラルネットワーク	1/9	方策勾配法
11/7	最適化アルゴリズム	1/16	ポスター発表の練習
11/14	畳み込みニューラルネットワーク， 画像分類	1/23	ポスター発表

日程：木曜日のどこか（おそらく1限）

参加：任意

対象：誰でも

参加要件：人の倍努力して，人の倍成長したい人

課題解決の時間外も活動できる人

真に楽しいと感じるためには，努力という山登りが必要

至らない点が多いと思いますが，相互に成長できればと思います

学習班　第1回
単回帰，線形回帰

やりたいことを、できることに。
湘南工科大学
SHONAN INSTITUTE OF TECHNOLOGY
AI R&D CENTER

人工知能の区分

人工知能

アルゴリズム，エキスパートシステム

機械学習

教師あり学習

線形回帰，多項式回帰，ロジスティック回帰，物体検出，決定木，XGBoost

ニューラルネットワーク

畳み込みニューラルネットワーク，セマンティックセグメンテーション，GAN，
拡散モデル，Transformer，BERT，GPT

再帰的ニューラルネットワーク，文章生成
Autoencoder，対照学習

DQN，方策勾配法，DDPG

Kmeans，PCA，GMM，NMF

教師なし学習

Q学習，SARSA，TD法

強化学習

教師あり学習

ある入力から答えを予測する学習法

ex) 身長から体重を予測する

気温・湿度から明日の天気を予測する

AIに与える情報を説明変数，AIに予測させる情報を目的変数と呼ぶ

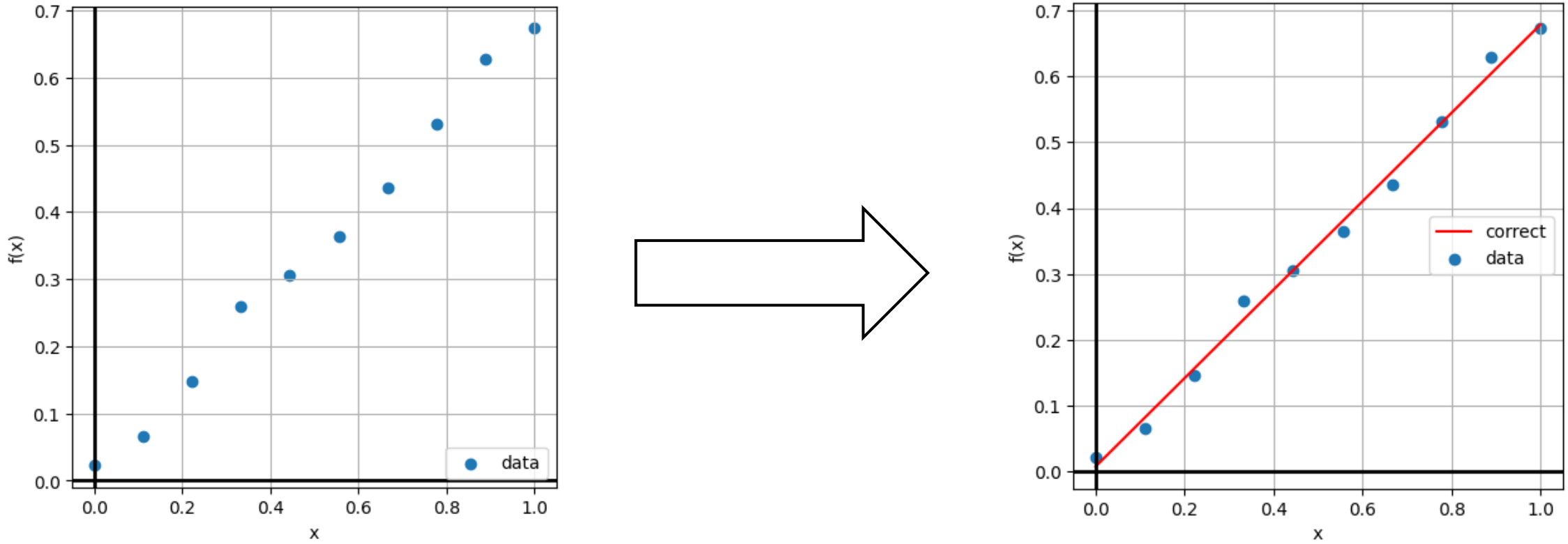
説明変数が1つの場合単回帰，複数の場合重回帰と呼ぶ

問題の呼称	予測させる情報
回帰問題	連続値
ロジスティック回帰	確率
分類問題	離散値

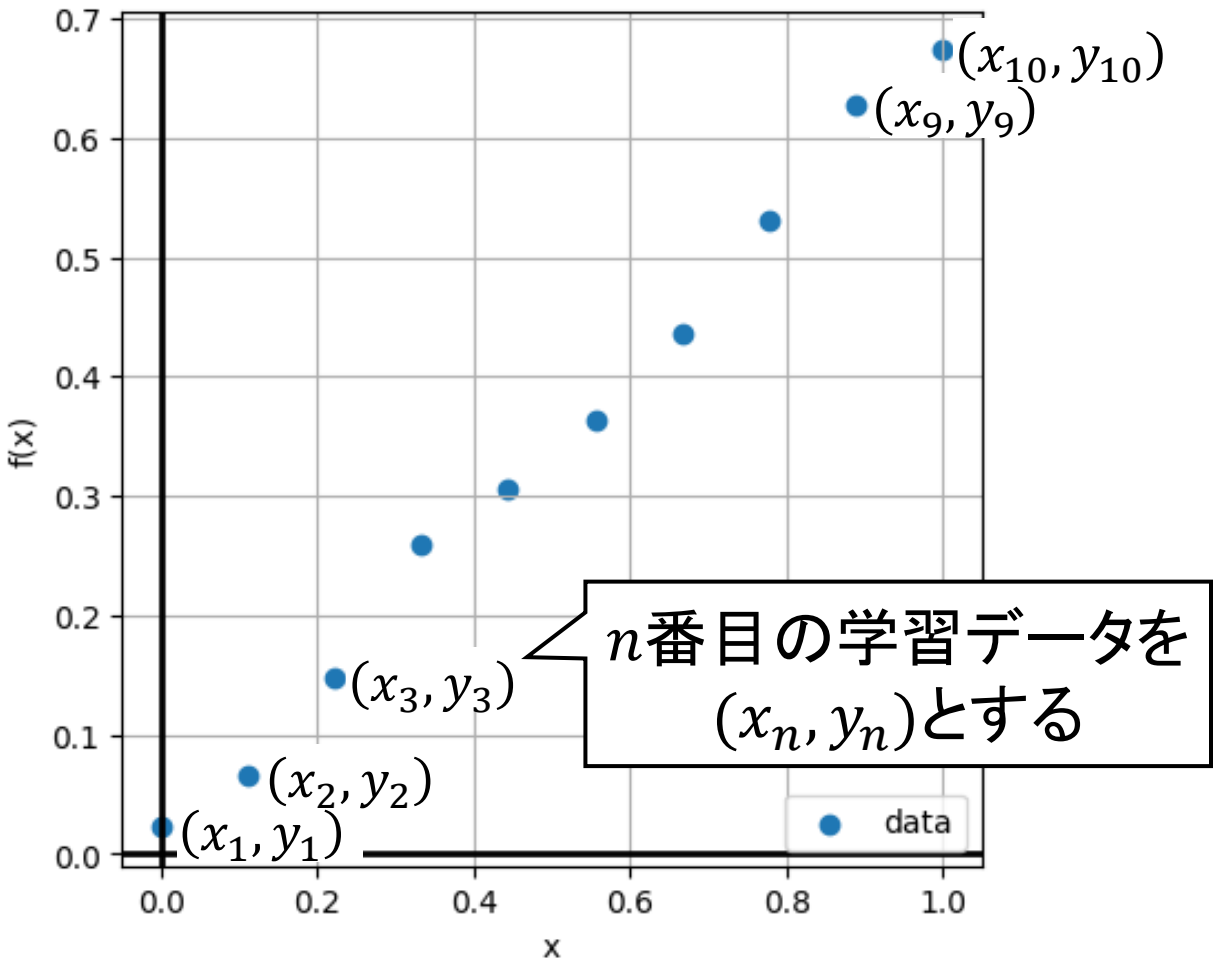
線形回帰

データを表す直線を得る

ex) 身長から体重を予測する



x座標からy座標を予測する問題設定を考える



学習データを10個の点とする

n	x	y
1	0.0	0.022
2	0.111	0.065
3	0.222	0.147
…	…	…
10	1.0	0.673

学習データの生成

```
# データの個数
NUM_DATA = 10

# 正解の直線の傾きと切片を乱数で決定
a = np.random.uniform(-1, 1)
b = np.random.uniform(0, 1)

# 0~1の数字をNUM_DATA個の等差数列にする
x = np.linspace(0, 1, NUM_DATA)
# 乱数を生成する
noise = np.random.uniform(-0.03, 0.03, size = (NUM_DATA, ))
y = a * x + b + noise

print(f"a : {a}, b : {b}")
for i, (X, Y) in enumerate(zip(x, y)):
    print(f"{i + 1}番目のデータ, x : {X}, y : {Y}")
```

```
np.random.uniform(low, high, size = None)
```

low以上，highより小さい実数の乱数を生成

sizeで生成する乱数の形状を指定する

指定しない(size=None)場合，スカラの乱数が生成される

```
np.linspace(start, stop, num)
```

startから始まりstopで終わるnum個の要素を持つ等差数列(numpy.ndarray)を生成

```
# ex)
print(np.linspace(0, 1, 9))
# => array([0. , 0.125, 0.25 , 0.375, 0.5 , 0.625, 0.75 , 0.875, 1. ])
```

numpy.ndarray

n次元配列（ベクトル，行列，テンソル）を実現するnumpyのデータ型

```
np.array([[1, 2, 3],
          [4, 5, 6]])
```

Pythonの配列をnumpy.ndarrayに変換する

Pythonの配列と異なり，

- 全ての要素のデータ型が同じである必要がある
- 各次元の要素数が同じである必要がある

```
# ex)
print(np.array([[1, 2, 3, 4, 5],
                [6, 7, 8, 9, 10]]))
# OK!
```

```
# ex)
print(np.array([[1, 2, 3, 4, 5],
                [6, 7, 8]]))
# NG...
```

numpy.ndarray

強力なキャスト機能を有する

```
a = np.array([1, 2, 3])
print(a)
# => [1 2 3]
b = a + 1
print(b)
# => [2 3 4]
c = a * 2
print(c)
# => [2 4 6]
d = a * 2 + 1
print(d)
# => [3 5 7]
```

```
a = np.array([[1, 2, 3],
              [4, 5, 6]])
print(a)
# => [[1 2 3]
#     [4 5 6]]
b = a + np.array([[1],
                  [2]])
print(b)
# => [[2 3 4]
#     [6 7 8]]
c = a * np.array([7, 8, 9])
print(c)
# => [[ 7 16 27]
#     [28 40 54]]
```

10個の学習データが生成された

```
a : -0.687287293295421 9, b : 0.456873367653432 1
1番目のデータ, x : 0.0, y : 0.456873367653432 1
2番目のデータ, x : 0.111111111111111, y : 0.380508112842829 7
3番目のデータ, x : 0.222222222222222, y : 0.304142858032227 3
4番目のデータ, x : 0.333333333333333, y : 0.227777603221624 82
5番目のデータ, x : 0.444444444444444, y : 0.151412348411022 4
6番目のデータ, x : 0.555555555555556, y : 0.075047093600419 89
7番目のデータ, x : 0.666666666666666, y : -0.001318161210182467
8番目のデータ, x : 0.777777777777777, y : -0.077683416020784 88
9番目のデータ, x : 0.888888888888888, y : -0.154048670831387 3
10番目のデータ, x : 1.0, y : -0.230413925641989 81
```

生成したデータを散布図にする

```
fig = plt.figure(figsize = (5, 5))
ax = fig.add_subplot()
# 散布図を描画
ax.scatter(x, y, label = "data")
ax.axhline(0, color = 'black', linewidth = 2)
ax.axvline(0, color = 'black', linewidth = 2)
ax.set_xlabel("x")
ax.set_ylabel("f(x)")
ax.legend()
ax.grid()
plt.show()
```

```
fig = plt.figure(figsize = (width, height))
```

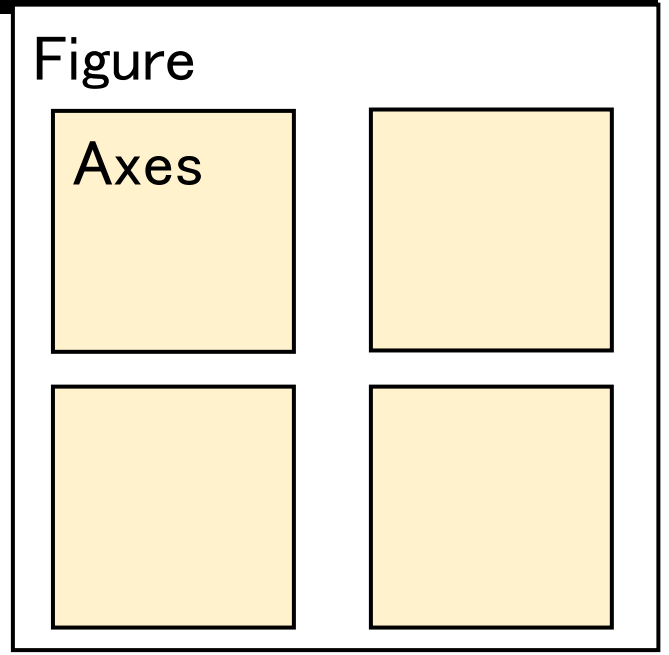
縦幅をwidth, 横幅をheightとして描画領域(土台)を作成

数字によって解像度が変わる

```
ax = fig.add_subplot()
```

描画領域に座標軸(画用紙)を追加する

Figureが土台,
Axesが画用紙のイメージ



土台に直接描画しても良いが,

画用紙に描画する方が便が良い

```
ax.scatter(x, y, label = "data")
```

配列x，yの各要素をx，y座標として散布図を描画する

labelオプションは凡例を指定する際に指定する

```
ax.axhline(0, color = 'black', linewidth = 2)
ax.axvline(0, color = 'black', linewidth = 2)
```

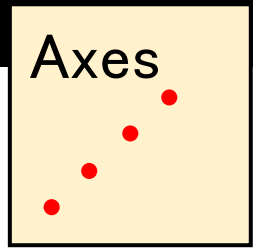
x=1，y=1の直線を描画する

colorオプションで色，linewidthオプションで線の太さを指定する

```
ax.set_xlabel("x")
ax.set_ylabel("f(x)")
```

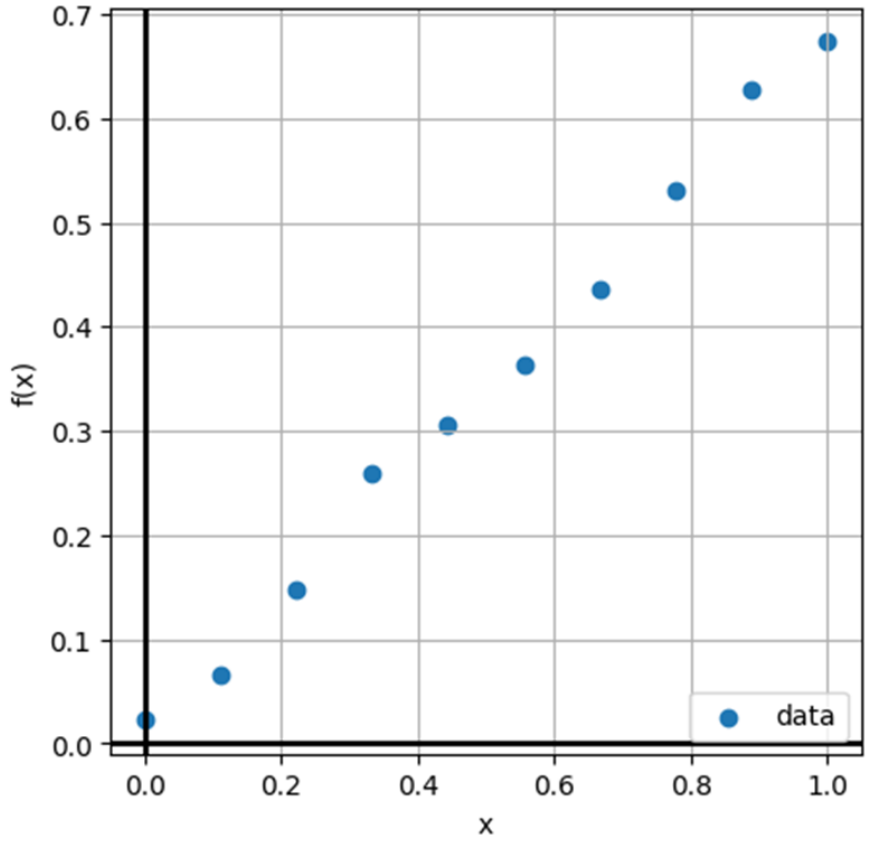
横軸，縦軸の軸ラベルを指定する

Axesに要素を描画する



```
ax.legend()
ax.grid()
```

凡例，グリッド線を描画する

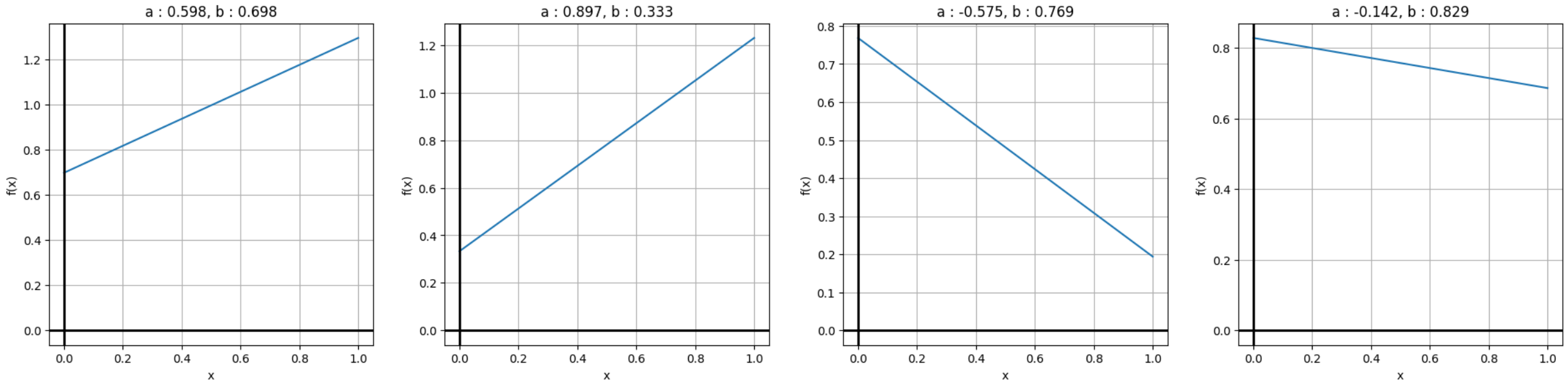


線形回帰モデル

$$f(x) = ax + b$$

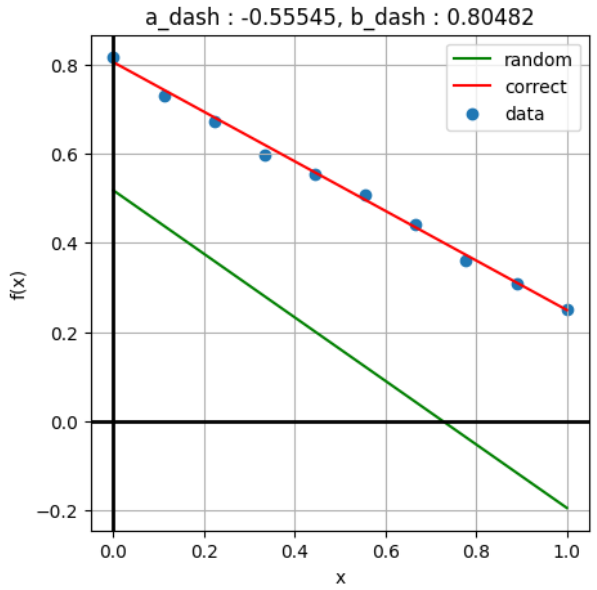
※ a, b : 学習によって決定するパラメータ

線形回帰モデルはa, bの値によって，あらゆる直線になれる

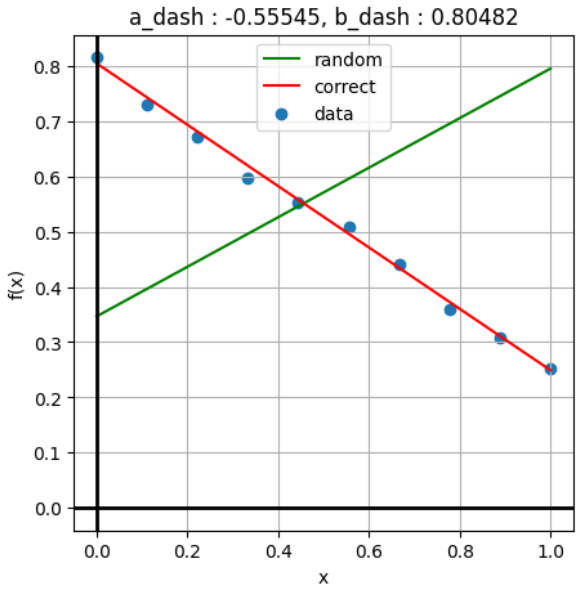


ランダムな直線を描画

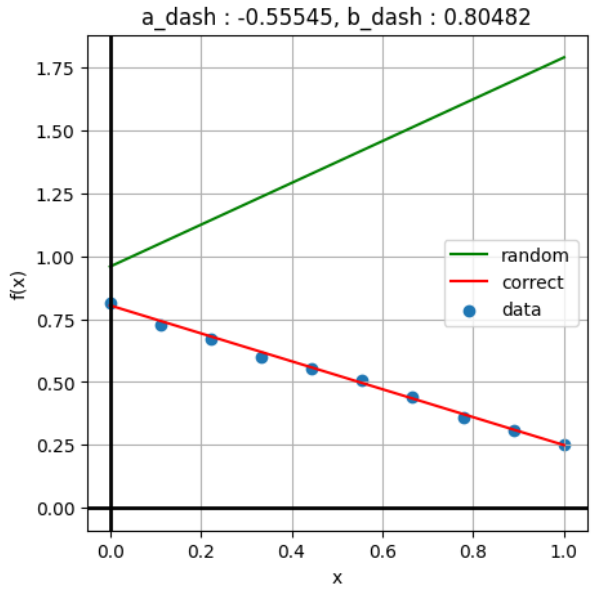
```
# 傾き，切片を乱数で決定
a_dash = np.random.uniform(-1, 1)
b_dash = np.random.uniform(0, 1)
y_dash = a_dash * x + b_dash

fig = plt.figure(figsize = (5, 5))
ax = fig.add_subplot()
# ランダムな直線を描画
ax.plot(x, y_dash, label = "random", color = "green")
# 正解の直線を描画
ax.plot(x, a * x + b, label = "correct", color = "red")
# 散布図を描画
ax.scatter(x, y, label = "data")
以下略
```



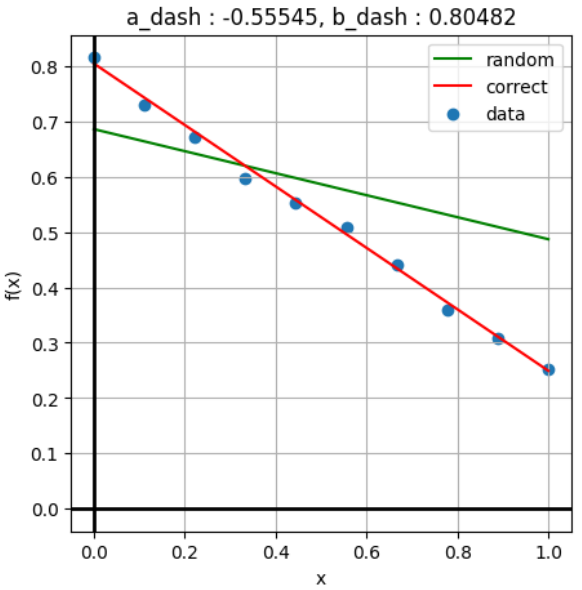
(A)



(B)



(C)



(D)

どの直線が最もデータを表せているか？

直感的には(D)？

平均二乗誤差

モデルがどの程度データに適しているかの指標

$$E = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{2}(f(x_n) - t_n)^2$$

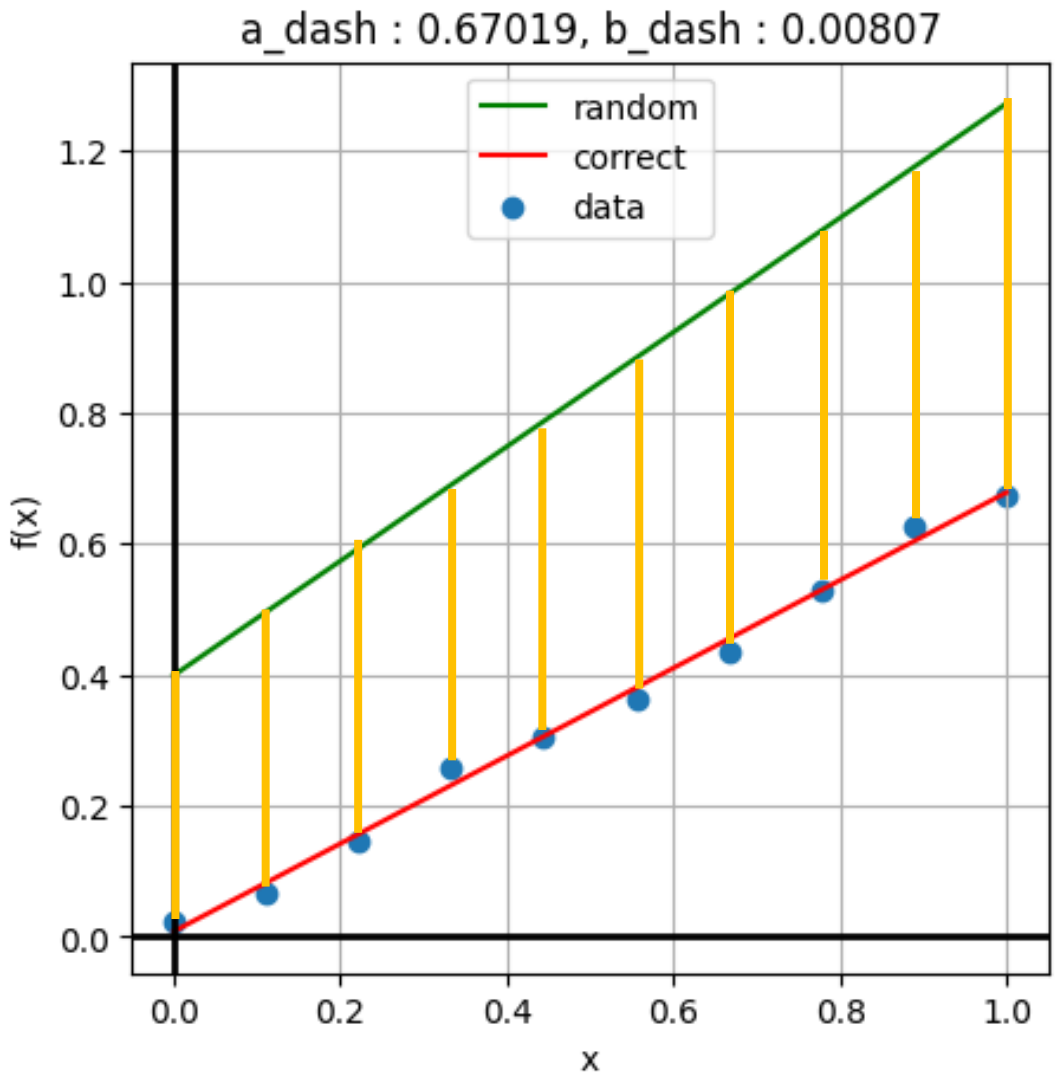
※ N：データの総数，x_n：n番目の入力，t_n：n番目の入力に対する教師信号

$f(x)$：xを入力した際のモデルの出力

Mean Squared Error(MSE)，二乗和誤差とも呼ばれる

$$E = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{2}(f(x_n) - t_n)^2$$

正解とモデルの出力の距離の総和



ランダムな直線に対し平均二乗誤差を計算する

```
# 平均二乗誤差を計算する関数
def mean_squared_error(y_true : np.ndarray, y_pred : np.ndarray) ->
np.float64:
    return np.average( (1 / 2) * np.square( y_true - y_pred ) )

# 傾き，切片を乱数で決定
a_dash = np.random.uniform(-1, 1)
b_dash = np.random.uniform(0, 1)
y_dash = a_dash * x + b_dash

# 平均二乗誤差
loss = mean_squared_error(y, y_dash)
print(f"平均二乗誤差 : {loss}")

中略

ax.set_title(f"MSE : {loss:.5f}")
plt.show()
```

```
def mean_squared_error(y_true: np.ndarray, y_pred: np.ndarray) ->
np.float64:
    return np.average( (1 / 2) * np.square( y_true - y_pred ) )
```

平均二乗誤差を計算する関数

y_true, y_predにはnumpy.ndarray型，戻り値はnp.float64型を想定

```
ax.set_title(label)
```

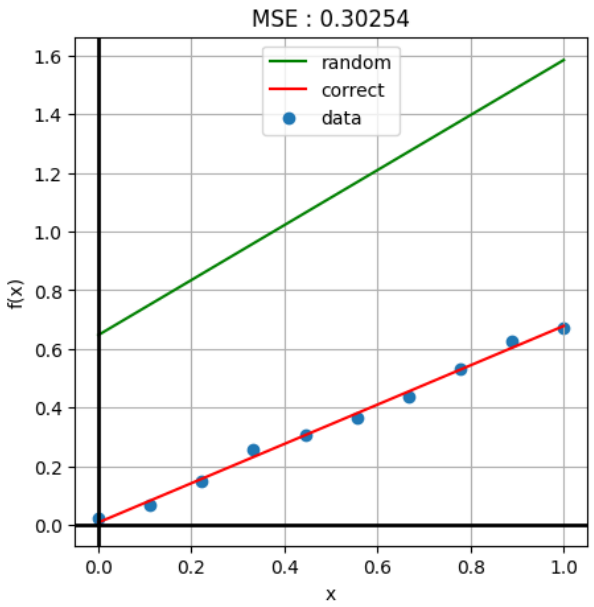
座標軸にタイトルを指定する

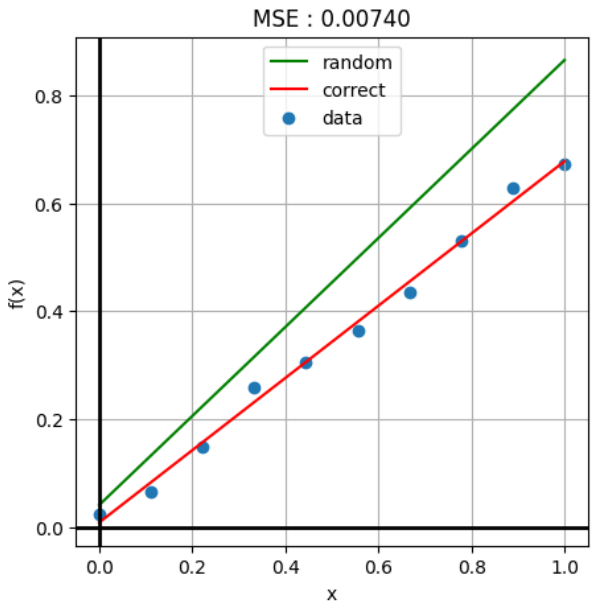
{}の外側は通常の文字列

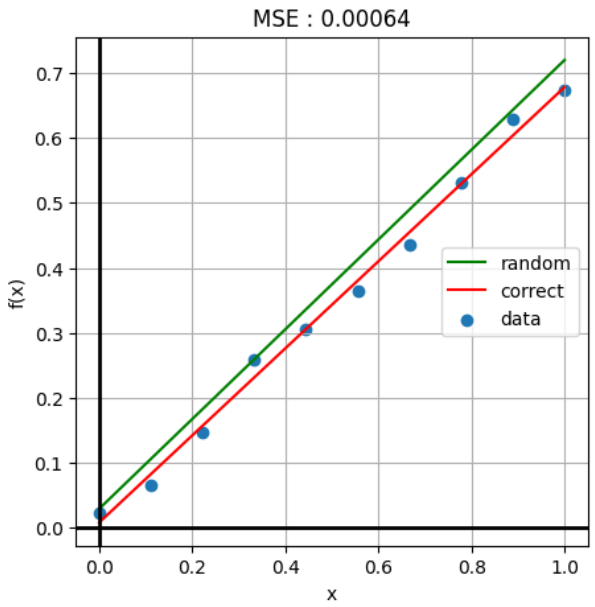
```
f"{a:.5f}"
```

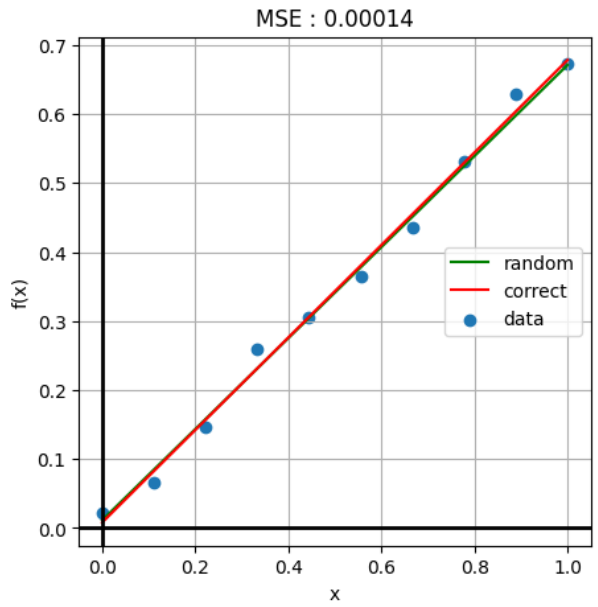
f”{var:format}”はフォーマットストリング

上記の場合，変数aを小数点以下5桁の文字列に変換する









正解の直線に近いほど平均二乗誤差の値が小さいことがわかる

どのように平均二乗誤差の値が小さいパラメータを得るか？

プログラマ的回答例？

```
while True:
    # 傾き，切片を乱数で決定
    a_dash = np.random.uniform(-1, 1)
    b_dash = np.random.uniform(0, 1)
    y_dash = a_dash * x + b_dash

    # 平均二乗誤差
    loss = mean_squared_error(y, y_dash)
    if loss < 0.0001: break
print(f"平均二乗誤差 : {loss}")
```

平均二乗誤差が規定値より小さくなるまで乱数を生成し続ける

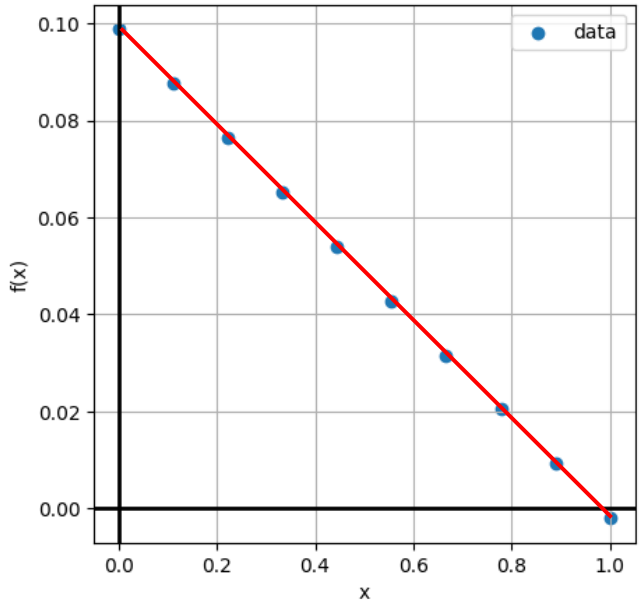
→ 上記の条件の場合，1分以上条件を満たすパラメータは得られなかった

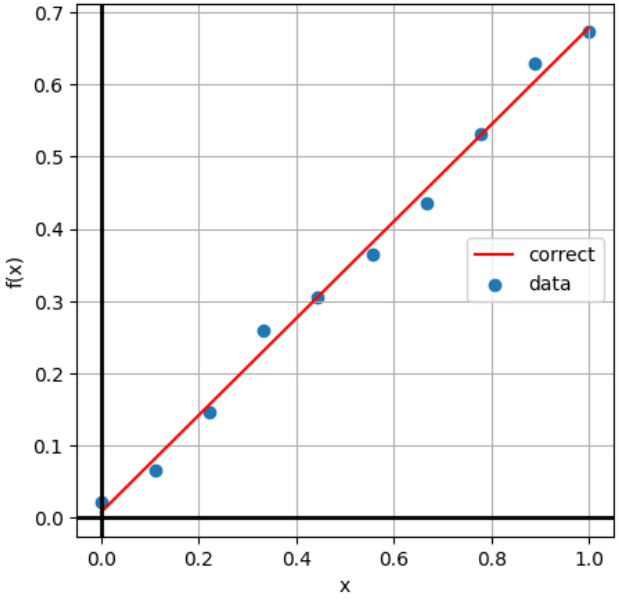
直線が全部の点を通っている場合，平均二乗誤差は0になる

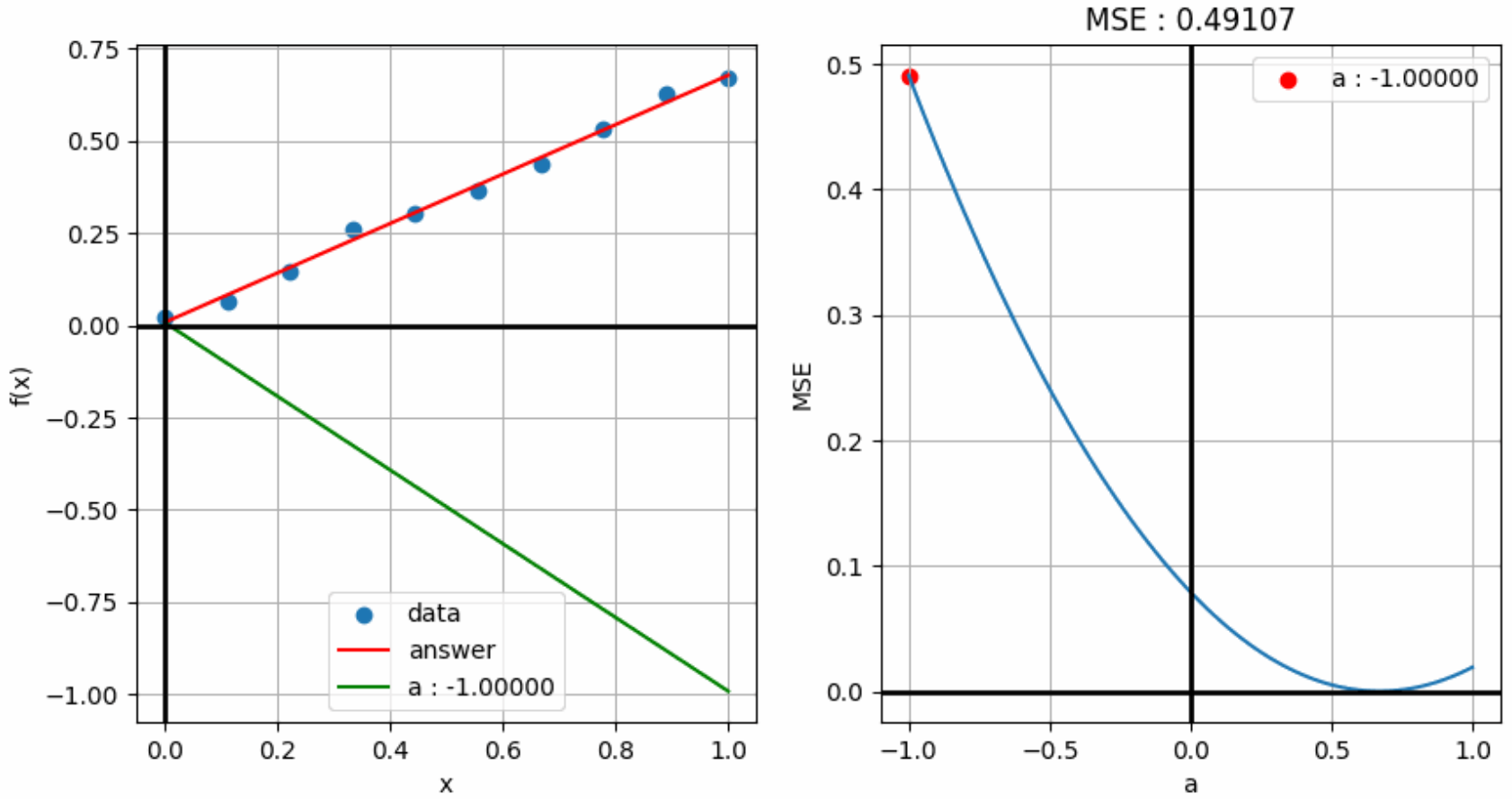
→ $E = 0$としてパラメータを求める

点が同一直線上にある場合，この考え方は正しいが，

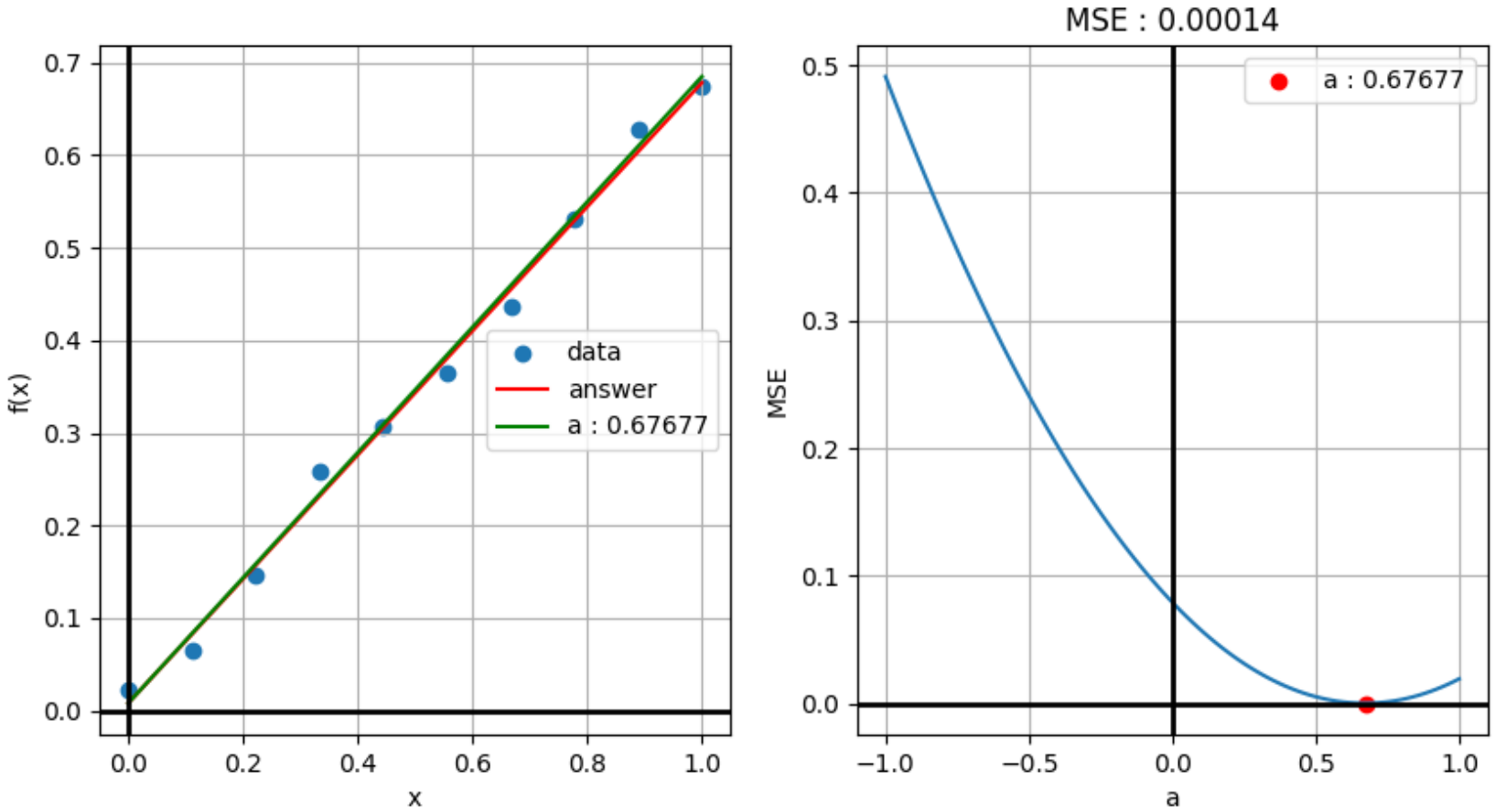
点が同一直線上にない場合，この考え方は成立しない





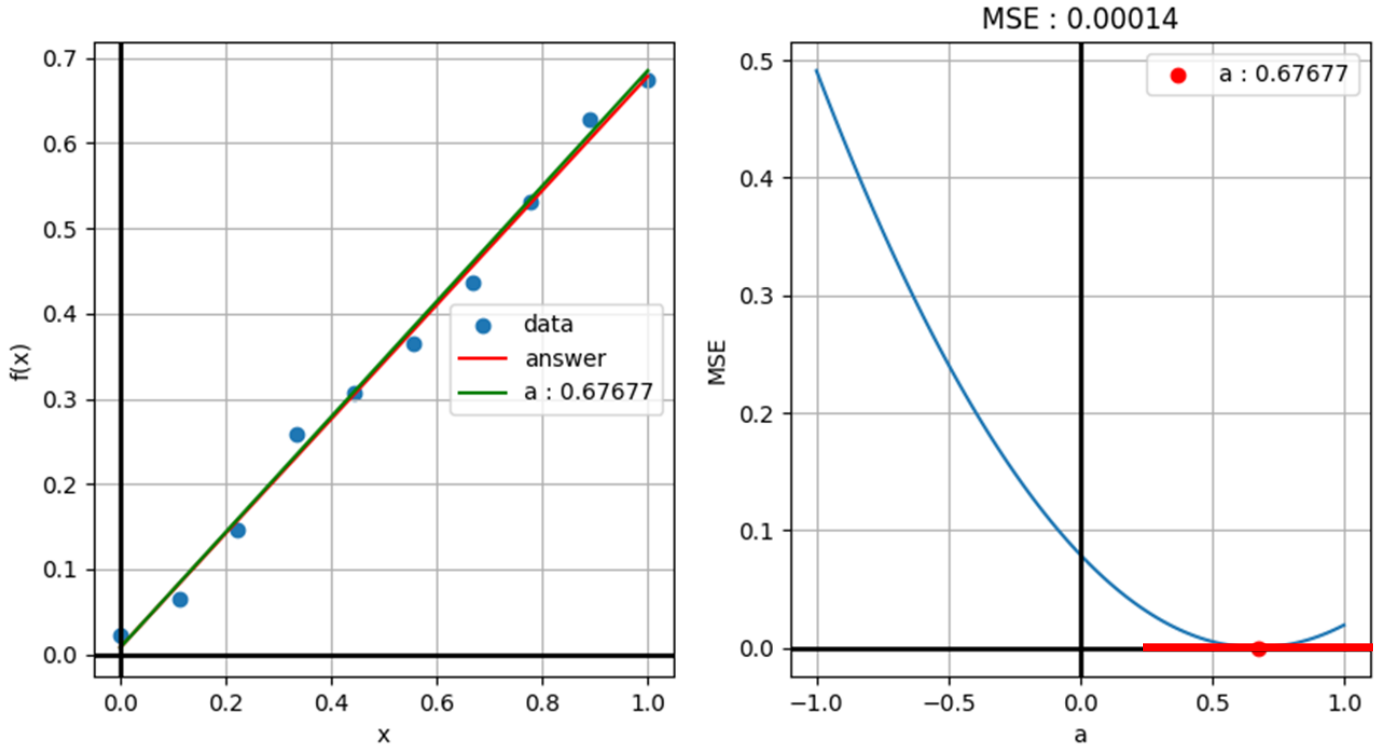


最小の平均二乗誤差は0.00013($a = 0.67676$)だった



最小の平均二乗誤差は0.00014($a = 0.67677$)だった

平均二乗誤差が最小の場合，その傾きが0になっている



最小二乗法

平均二乗誤差の傾きが0になるパラメータを求める

→ $\frac{\partial E}{\partial a} = 0,\ \ \frac{\partial E}{\partial b} = 0$を満たす$a, b$を求める

$$E(a,b) = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{2}(f(x_n) - t_n)^2$$

パラメータが入力の関数であると考える

$\frac{\partial E}{\partial a}, \frac{\partial E}{\partial b}$は偏微分

多変数関数を1つの変数で微分する場合に用いる

微分する変数以外を定数であると考える

$$E(a,b) = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{2}(f(x_n) - t_n)^2,\ f(x) = ax + b$$

$$\frac{\partial E}{\partial a} = \frac{\partial}{\partial a}\left\{\frac{1}{N}\sum_{n=1}^{N}\frac{1}{2}(f(x_n) - t_n)^2\right\}$$

$$= \frac{1}{N}\sum_{n=1}^{N}\frac{\partial}{\partial a}\left\{\frac{1}{2}(f(x_n) - t_n)^2\right\}$$

※ 和の微分

$$= \frac{1}{N}\sum_{n=1}^{N}\left[\frac{\partial}{\partial f}\left\{\frac{1}{2}(f(x_n) - t_n)^2\right\}\right]\left\{\frac{\partial}{\partial a}(ax_n + b)\right\}$$

※ 連鎖律

$$= \frac{1}{N}\sum_{n=1}^{N}(f(x_n) - t_n)\,x_n$$

$$E(a,b) = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{2}(f(x_n) - t_n)^2,\ f(x) = ax + b$$

$$\frac{\partial E}{\partial b} = \frac{\partial}{\partial b}\left\{\frac{1}{N}\sum_{n=1}^{N}\frac{1}{2}(f(x_n) - t_n)^2\right\}$$

$$= \frac{1}{N}\sum_{n=1}^{N}\frac{\partial}{\partial b}\left\{\frac{1}{2}(f(x_n) - t_n)^2\right\}$$

※ 和の微分

$$= \frac{1}{N}\sum_{n=1}^{N}\left[\frac{\partial}{\partial f}\left\{\frac{1}{2}(f(x_n) - t_n)^2\right\}\right]\left\{\frac{\partial}{\partial b}(ax_n + b)\right\}$$

※ 連鎖律

$$= \frac{1}{N}\sum_{n=1}^{N} f(x_n) - t_n$$

$\frac{\partial E}{\partial a} = \frac{1}{N}\sum_{n=1}^{N}(f(x_n) - t_n)\,x_n$, $\frac{\partial E}{\partial b} = \frac{1}{N}\sum_{n=1}^{N} f(x_n) - t_n$が得られる

$f(x) = ax + b$, $\frac{\partial E}{\partial a} = \frac{1}{N}\sum_{n=1}^{N}(f(x_n) - t_n)\,x_n$, $\frac{\partial E}{\partial b} = \frac{1}{N}\sum_{n=1}^{N} f(x_n) - t_n$

$\frac{\partial E}{\partial a} = 0$, $\frac{\partial E}{\partial b} = 0$を満たす$a, b$を求めるのが目的であるため,

$$0 = \frac{1}{N}\sum_{n=1}^{N}(f(x_n) - t_n)\,x_n \qquad \cdots①$$

$$0 = \frac{1}{N}\sum_{n=1}^{N}(f(x_n) - t_n) \qquad \cdots②$$

①, ②, $f(x) = ax + b$より,

$$0 = \frac{1}{N}\sum_{n=1}^{N}(ax_n + b - t_n)\,x_n \qquad \cdots③$$

$$0 = \frac{1}{N}\sum_{n=1}^{N} ax_n + b - t_n \qquad \cdots④$$

④より,

$$0 = \frac{1}{N}\sum_{n=1}^{N} ax_n + b - t_n \qquad \cdots④$$

$$= a\frac{1}{N}\sum_{n=1}^{N} x_n + \frac{1}{N}\sum_{n=1}^{N} b - \frac{1}{N}\sum_{n=1}^{N} t_n$$

$$b = \frac{1}{N}\sum_{n=1}^{N} t_n - a\frac{1}{N}\sum_{n=1}^{N} x_n$$

ここで，$\bar{x} = \frac{1}{N}\sum_{n=1}^{N} x_n$とすると，

※ $\bar{x} = \frac{1}{N}\sum_{n=1}^{N} x_n$は入力の平均

$$b = \bar{t} - a\bar{x} \qquad \cdots⑤$$

⑤, ③より,

$$b = \bar{t} - a\bar{x} \qquad \cdots ⑤$$

$$0 = \frac{1}{N}\sum_{n=1}^{N}(ax_n + b - t_n)\,x_n \qquad \cdots ③$$

$$= \frac{1}{N}\sum_{n=1}^{N}(ax_n + \bar{t} - a\bar{x} - t_n)\,x_n$$

$$= a\frac{1}{N}\sum_{n=1}^{N}x_n^2 + \bar{t}\frac{1}{N}\sum_{n=1}^{N}x_n - a\bar{x}\frac{1}{N}\sum_{n=1}^{N}x_n - \frac{1}{N}\sum_{n=1}^{N}t_n x_n$$

$$= a\left(\overline{x^2} - \bar{x}^2\right) + \bar{t}\bar{x} - \overline{tx}$$

$$\overline{tx} - \bar{t}\bar{x} = a\left(\overline{x^2} - \bar{x}^2\right)$$

$$a = \frac{\overline{tx} - \bar{t}\bar{x}}{\overline{x^2} - \bar{x}^2} \qquad \cdots ⑥$$

$$a = \frac{\overline{tx} - \bar{t}\bar{x}}{\overline{x^2} - \bar{x}^2}$$

分子は入力と教師信号の共分散，分母は入力の分散となる

入力の分散

$$\begin{aligned}\sigma_x^2 &= \frac{1}{N}\sum_{n=1}^{N}(\bar{x} - x_n)^2 \\ &= \frac{1}{N}\sum_{n=1}^{N}\bar{x}^2 - 2\bar{x}x_n + x_n^2 \\ &= \bar{x}^2 - 2\bar{x}^2 + \overline{x^2} \\ &= \overline{x^2} - \bar{x}^2\end{aligned}$$

入力と教師信号の共分散

$$\begin{aligned}\sigma_{tx} &= \frac{1}{N}\sum_{n=1}^{N}(\bar{x} - x_n)(\bar{t} - t_n) \\ &= \frac{1}{N}\sum_{n=1}^{N}\bar{t}\bar{x} - \bar{x}t_n - \bar{t}x_n + t_n x_n \\ &= \bar{t}\bar{x} - 2\bar{t}\bar{x} + \overline{tx} \\ &= \overline{tx} - \bar{t}\bar{x}\end{aligned}$$

$$a = \frac{\sigma_{tx}}{\sigma_x^2}, b = \bar{t} - a\bar{x}$$

入力の平均と分散，教師信号の平均，入力と教師信号の共分散によって，
線形回帰モデルの最適なパラメータが得られる

最適なパラメータを計算する

```
var_x = np.var(x)
cov_xy = np.cov(x, y, bias = True)[0, 1]
avg_x = np.average(x)
avg_y = np.average(y)

a_dash = cov_xy / var_x
b_dash = avg_y - a_dash * avg_x

print(f"本来のa : {a}, 導き出したa : {a_dash}")
print(f"本来のb : {b}, 導き出したb : {b_dash}")
```

```
本来のa : 0.670188872328175, 導き出したa : 0.670835474309117
本来のb : 0.008073570554582243, 導き出したb : 0.007846655248160583
```

学習データにノイズが含まれるので必ず同じ数字にはならないが，

かなり実際の値に近い値が得られている

```
np.var(a)
```

np.ndarrayのaの分散を計算する

```
np.cov(m, y, bias = True)
```

np.ndarrayのm，yの分散共分散行列*1を計算する

bias = Trueによって標本分散*2で計算される

分散共分散行列*1

対角成分が分散で，非対角成分が共分散である行列

$$\Sigma = \begin{bmatrix} \sigma_m^2 & \sigma_{my} \\ \sigma_{ym} & \sigma_y^2 \end{bmatrix}$$

標本分散*2

標本分散 : $\sigma_x^2 = \frac{1}{N}\sum_{n=1}^{N}(\bar{x} - x_n)^2$

不偏分散 : $\sigma_x^2 = \frac{1}{N-1}\sum_{n=1}^{N}(\bar{x} - x_n)^2$

np.covはデフォルトでは不偏分散を計算するので注意！

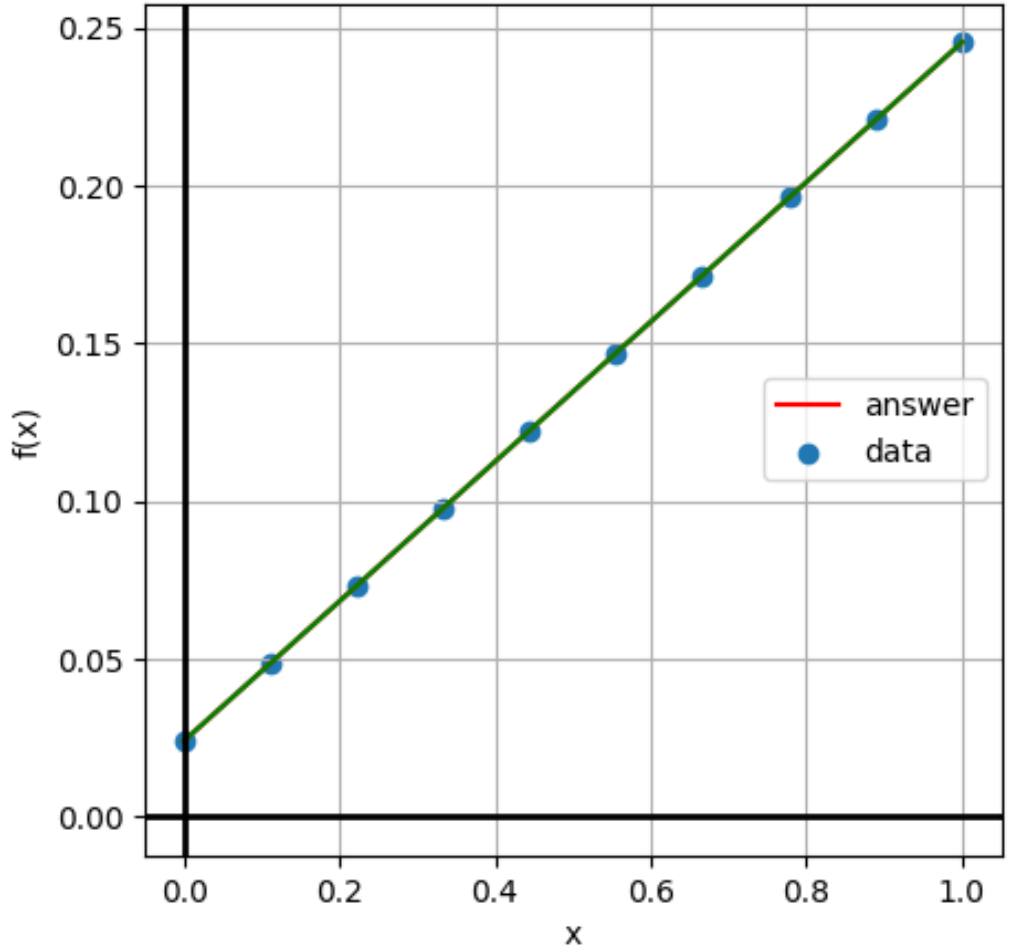
不偏分散はサンプルの分散ではなく母集団の分散を計算する考え方

なんで$N-1$で割ると母集団の分散になるのかは謎…

得られたパラメータから直線を描画

```
y_dash = a_dash * x + b_dash

fig = plt.figure(figsize = (5, 5))
ax = fig.add_subplot()
# 正解の直線を描画
ax.plot(x, y, label = "answer", color = "red")
#パラメータから直線を描画
ax.plot(x, y_dash, color = "green")

以下略
```

正解の直線と同様の直線が得られている



まとめ

- 教師あり学習は入力から教師信号を予測するモデル
- 線形回帰モデルはデータを表す直線を得る
- 線形回帰モデルのパラメータは平均二乗誤差の傾きが0になる時，最適となる
- 線形回帰モデルの最適なパラメータは，入力と教師信号の平均，分散，
  共分散によって求められる

宿題1-1

下記の身長・体重のデータの平均，分散，共分散をnp.average，np.var，np.covを用いず，定義式に則りプログラムで計算する

データ番号	身長	体重
1	170cm	60kg
2	175cm	70kg
3	168cm	60kg
4	180cm	75kg
5	172cm	68kg
6	165cm	58kg
7	178cm	73kg
8	160cm	55kg
9	182cm	78kg
10	167cm	63kg

宿題1-2

下記の身長・体重データの散布図を描画する

データ番号	身長	体重
1	170cm	60kg
2	175cm	70kg
3	168cm	60kg
4	180cm	75kg
5	172cm	68kg
6	165cm	58kg
7	178cm	73kg
8	160cm	55kg
9	182cm	78kg
10	167cm	63kg

宿題1-3

下記の身長・体重データに適した線形回帰モデルのパラメータを求める.
また，求めたデータとパラメータを用いて散布図と回帰直線を描画する

データ番号	身長	体重
1	170cm	60kg
2	175cm	70kg
3	168cm	60kg
4	180cm	75kg
5	172cm	68kg
6	165cm	58kg
7	178cm	73kg
8	160cm	55kg
9	182cm	78kg
10	167cm	63kg