フィッティング練習

次に、y(x)=ax+b をフィッティングしてみましょう。

```
import numpy as np
import math
```

x = np.linspace(-math.pi, math.pi, 20)

```
print(x)
```

```
[-3.14159265 -2.81089869 -2.48020473 -2.14951076 -1.8188168  -1.48812284
 -1.15742887 -0.82673491 -0.49604095 -0.16534698  0.16534698  0.49604095
  0.82673491  1.15742887  1.48812284  1.8188168   2.14951076  2.48020473
  2.81089869  3.14159265]
```

linspace に注意してください。linespace ではありません。これは PyTorch チュートリアルの一例の一部コードです。これらの小数はあまり直感的ではないかもしれません。

```
x = np.linspace(0, 100, 20)
```

```
import numpy as np
import math

x = np.linspace(0, 100, 20)
y = np.linspace(0, 100, 20)

print(x)
print(y)
```

このようにして 2 組のデータが得られます。では、これをどのようにグラフで表現するのでしょうか。

しかし、驚くべきことに x と y は同じです。

```
x = np.random.rand(2)
print(x)
```

このコードは、NumPy ライブラリを使用して 2 つのランダムな数値を生成し、それらを表示するものです。具体的には、`np.random.rand(2)` が 0 から 1 の範囲で 2 つのランダムな浮動小数点数を生成し、`print(x)` でその結果を出力します。

```
[0.06094295 0.89674607]
```

(注：このコードブロック内の数値はそのまま保持されます。翻訳の必要はありません。)

続けて修正します。

```
x = np.random.rand(2)*100
print(x)
```

このコードは、NumPy ライブラリを使用して、0 から 100 の範囲で 2 つのランダムな浮動小数点数を生成し、それを出力します。コード自体は翻訳の対象ではありませんが、説明を日本語で提供します。

```
[39.6136151  66.15534011]
```

(注：この数値は座標や特定のデータを示している可能性がありますが、翻訳の対象ではないため、そのまま記載しています。)

続けて修正します。

```
import numpy as np
import math
import matplotlib.pyplot as plt

x = np.random.rand(10)*100
y = np.random.rand(10)*100

plt.plot(x, y)
plt.show()
```

```
[20.1240488  59.69327146 58.05432614  3.14092909 82.86411091 43.23010476
 88.09796699 94.42222486 58.45253048 51.98479507]
[58.7129098   1.6457994  49.34115933 71.13738592 53.09736099 15.4485691
 45.12200319 20.46080549 67.48555147 91.10864978]
```

見ての通り、(20.1,58.7) から (59.7,1.6)、そして (58,49.3) へと続いています。この図は一見乱雑に見えますが、実は規則性があります。それは一筆書きで描かれているのです。

```
import numpy as np
import math
import matplotlib.pyplot as plt

x = np.random.rand(2)*100
y = np.random.rand(2)*100

print(x)
print(y)

plt.plot(x, y)
plt.show()
```

x と y のスケールが常に変化していることに気づきました。そのため、一見同じように見える 2 本の直線も、実際には異なります。では、y(x) = ax + b の a と b をどのように求めるのでしょうか。ここで、この直線上の 2 点がわかっているとします。紙の上で計算して解くだけで十分です。2 つの方程式を引き算して b を消去し、a を求めます。その後、a を 1 つの方程式に代入して b を求めます。

しかし、推測の方法を使うことはできるでしょうか。二分法を使ってみましょう。試してみます。

```
import numpy as np
import math
import matplotlib.pyplot as plt

x = np.random.rand(2)*100
y = np.random.rand(2)*100

a_max = 1000
a_min = -1000
b_max = 1000
b_min = -1000

def cal_d(da, b):
    y0 = x[0] * da + b
```

```
    y1 = x[1] * da + b
    d = abs(y0 - y[0]) + abs(y1 - y[1])
    return d

def cal_db(a, db):
    y0 = x[0] * a + db
    y1 = x[1] * a + db
    d = abs(y0 - y[0]) + abs(y1 - y[1])
    return d

def avg_a():
    return (a_max + a_min) / 2

def avg_b():
    return (b_max + b_min) / 2

for i in range(100):
    a = avg_a()
    b = avg_b()
    max_d = cal_d(a_max, b)
    min_d = cal_d(a_min, b)
    if max_d < min_d:
        a_min = a
    else:
        a_max = a

    a = avg_a()
    max_db = cal_db(a, b_max)
    min_db = cal_db(a, b_min)
    if max_db < min_db:
        b_min = b
    else:
        b_max = b

print(x)
print(y)
print('a = ', avg_a())
```

```
print('b = ', avg_b())
print(avg_a() * x[0] + avg_b())
print(avg_a() * x[1] + avg_b())
```

実行してみましょう。

```
[42.78912791 98.69284173]
[68.95535212 80.89946202]
a =  11.71875
b =  -953.125
-451.68990725289063
203.4317390671779
```

このコードブロックは数値データの出力を示しています。具体的には、2つの座標点 [42.78912791, 98.69284173] と [68.95535212, 80.89946202]、そして変数 a と b の値、さらに2つの計算結果 -451.68990725289063 と 203.4317390671779 が表示されています。これらの値は、何らかの計算や処理の結果として出力されたものと考えられます。

結果は大きく異なりました。

問題を単純化しましょう。y(x) = ax と仮定します。x と y の組が与えられたとき、a を求めることを考えます。直接計算することもできますが、ここでは推測してみましょう。

```
import numpy as np
import math
import matplotlib.pyplot as plt
from numpy.random import rand, randint

x = randint(100)
y = randint(100)

a_max=1000
a_min=-1000

def cal_d(da):
    y0 = x * da
    return abs(y0 - y)

def avg_a():
    return (a_max + a_min) / 2
```

```
for i in range(1000):
    a = avg_a()
    max_d = cal_d(a_max)
    min_d = cal_d(a_min)
    if max_d < min_d:
        a_min = a
    else:
        a_max = a

print(x)
print(y)
print(avg_a())
print(avg_a()*x)
```

結果は喜ばしいものでした。予想が当たりました。

```
96
61
0.6354166666666667
61.00000000000001
```

(注：数値データのため、翻訳の必要はありません。)

しかし、通常は `for i in range(15):` と書いて、15 回の反復で十分正確になります。なぜでしょうか。私たちの `x` と `y` はどちらも 0 から 100 の範囲内にあることに注意してください。したがって、`a` の値も 0 から 100 の範囲内にあります。`x=1,y=99` と `x=99,y=1` の場合を考えてみましょう。したがって、`a_min` と `a_max` の初期値を最適化することができます。`1/99` は `0.01` であることに注意してください。したがって、`0.01` の精度まで計算するには、2^n ≈ 10000 となる n を求めます。log2(10000)=13.28 です。つまり、14 回程度に設定すれば十分です。