深層学習入門

田中 章詞（たなか あきのり） (github.com/AkinoriTanaka-phys)

(理化学研究所 iTHEMS/AIP, 慶應義塾大学 数理科学科)

> 今回の入門の詳細版へのリンク

今日お話しする内容

1. **Python入門**
2. **NumPy入門**
3. **Matplotlib入門**
4. **深層学習ライブラリ入門**

1. Python入門

深層学習のフレームワークの多くはプログラミング言語pythonから使います。そこで、まずはpythonの文法から説明します。

> 詳細版（Colaboratory notebook）：
> https://colab.research.google.com/github/AkinoriTanaka-phys/deeplearning_notes/blob/master/appendix/intro2python.ipynb

基本的な文法

更に詳しく知りたい方は**公式ドキュメントのチュートリアル**を読むといいのかもしれません。

あるいは、「python+やりたいこと」でインターネット検索して下さい。検索するとだいたいうまいやり方がヒットします。

> この方法でコードを拾ったときは、どこから拾ったかのURLも一緒にコメントなどで取っておくのが良いです。と言うのも、プログラミングのコードには**著作権が発生する場合がある**からです。

正確なことは**公式ドキュメントで検索**するのが間違い無くて良いかと思います。

文字列を表示する

> ### 値の出力の文法
>
> ```
> print(value)
> ```
>
> デフォルトでは最後に改行されます。

数値を表示：

```
print(365)
```

```
365
```

簡単な演算

演算

```
#addition
x + y
#subtraction
x - y
#product
x*y
#quotient
x/y
#floored quotient
x//y
#reminder
x%y
#power
x**y
```

計算順序がわからなくなった場合、半角括弧()で囲む。

コメント

読み手（未来の自分自身も含む）の理解を助けるコメントをつけたりします。

> **コメント**
>
> ```
> hogehoge # comments
> ```
>
> コメントは、なるべく英語で書くのが良いらしいです。

例えば以下のセルを実行すると（話の本筋とは関係のない）興味深い計算結果が得られますが、なぜそうなるのかコメントを読むとわかります：

```
1/0.9899 # Generating function of Fibonacci numbers 1/(1-x-x**2)  with x=1/100
```

```
1.0102030508132134
```

変数と型

数値などの値は変数に格納できます。

> **変数代入**
>
> ```
> x = value
> ```
>
> 左辺（x）に右辺（value）が代入されます。

計算は上から順に実行されます：

```
# 変数の使用例
x = 30
3*x
```

```
90
```

代入できる数値にも、様々な**型**があります。型について詳細を説明していると長くなりすぎるので、チートシート的なものを並べるだけに止めます。詳細は**公式ドキュメント**参照

- **数：int, float, complex**: `1` , `3.14` , `3+2j`
- **文字列：str**: `"this is it"`
- **シーケンス型：list, tuple** `[1,2,3]` , `(1, "two", 3.0)`
- **辞書型：dict** `{"short":290, "tall":330, "grande":370}`
- **論理型：bool** `True, False`
- **None オブジェクト** `None`

インデント(字下げ)

Pythonではインデント（字下げ）を使って処理のブロック（かたまり）を明記します。

> **インデント・ルール（とても重要）**
>
> ```
> block:
>    operation
>    ...
> ```
>
> インデントは入れ子の構造を持っていても構いません：
>
> > 例：ブロックAの中で、別のブロックBを定義
> >
> > ```
> > blockA:
> >    operationsA
> >    blockB:
> >        operationsB
> ```

for文の文法

```
for n in range(N):
    n-th operations
```

n=0,1,2,...,N-1と順にループします。

条件分岐の文法

```
if A:
    operationsA
elif B:
    operationsB
else:
    operationsC
```

elifやelseがなくても動きます。ifやelifにはブール代数の値（TrueかFalse）が入るようにします。

for文とif文を使えば素数の判定ができます：

```
N = 31

for n in range(2, N):
    if N%n==0:
        print(f"Its divisible by {n}")
        break  # stop the for loop here
    if n==N-1:
        print("prime")
```

```
prime
```

2. NumPy入門

実はpythonだけでは、計算速度が遅いため数値計算などには向いていません。数値計算したい場合はnumpyを使うのをオススメします。

> NumPy詳細版（Colaboratory notebook）：
> https://colab.research.google.com/github/AkinoriTanaka-phys/deeplearning_notes/blob/master/appendix/intro2np.ipynb

インポートコマンド

まず使うためには以下のようなコマンドを打ち込みます：

```
import numpy as np
```

コマンドは以下のように、npとコマンド名を.で区切ります：

> ```
> np.コマンド名
> ```

numpyは**配列=np.arrayを取り扱う**ライブラリーです。

pythonリストとnumpy配列

np.arrayはpythonのlist型と行き来できます

np.array ← リスト

```
x_np = np.array(x_list)
```

リスト ← np.array

```
x_list = x_np.tolist()
```

```
x_np = np.array([[2, 1], [5, 3]])
x_np
```

```
array([[2, 1],
       [5, 3]])
```

データの形状

np.arrayには**shape**という変数が設定されています。例：

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

形状を取得する：.shape

```
x_np.shape # -> (dimension0, dimension1, ...)
```

要素にアクセスする方法

x_npをshapeが(M, N, ...)であるようなnumpy配列とします

特定の要素にアクセス

```
x_np[m, n, ...]
```

まとめた要素にアクセス

```
x_np[m, n0:n1, ...]
```

n0からn1-1までの要素にアクセス。特に:だけだと、全要素にアクセスとなります。

Matplotlib入門

pythonで図を描く場合はだいたいこれを使います。

> Matplotlib詳細版（Colaboratory notebook）：
> https://colab.research.google.com/github/AkinoriTanaka-phys/deeplearning_notes/blob/master/appendix/intro2plt.ipynb

インポートコマンド

まず使うためには以下のようなコマンドを打ち込みます：

```
import matplotlib.pyplot as plt
%matplotlib inline
```

2行目は手元のノートブック環境で結果をブラウザなどの枠内に収めるためのコマンドでGoogle Colaboratoryでは不要です。

このライブラリはNumPyを使うことが想定されているので、NumPyもインポートしておきます。

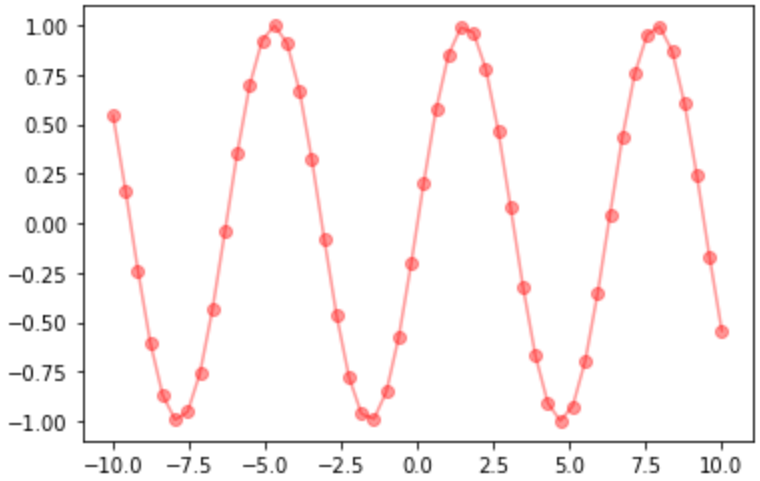
```
import numpy as np
```

グラフ

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

```
x = np.linspace(-10, 10)

plt.plot(x, np.sin(x), "o-", color="red", alpha=0.4)
plt.show()
```

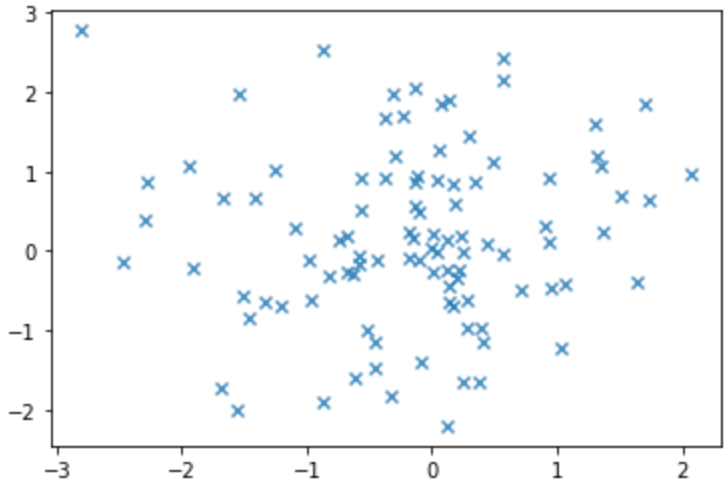


散布図

```
plt.scatter(x, y, options)
```

```
rg = np.random.default_rng() # 擬似乱数生成器
xy = rg.standard_normal(size=(100, 2))

plt.scatter(xy[:,0], xy[:,1], marker="x", alpha=0.8)
```



4. 深層学習ライブラリ入門

pythonでは様々な深層学習ライブラリを使うことができます。全てGPUでの並列処理が可能です。

ライブラリ名	TensorFlow	PyTorch	MXNet	CNTK	...
サポート	Google	Facebook	AWS	Microsoft	...

TensorFlow入門

ここまでの文法を踏まえて、実演してみます。

> TensorFlow（Colaboratory notebook）：
> https://colab.research.google.com/github/AkinoriTanaka-phys/deeplearning_notes/blob/master/appendix/intro2tf.ipynb

PyTorch入門

ここまでの文法を踏まえて、実演してみます。

> PyTorch（Colaboratory notebook）：
> https://colab.research.google.com/github/AkinoriTanaka-phys/deeplearning_notes/blob/master/appendix/intro2torch.ipynb