2022/4/22
第175回お試しアカウント付き並列プログラミング講習会
「スーパーコンピュータ超入門」

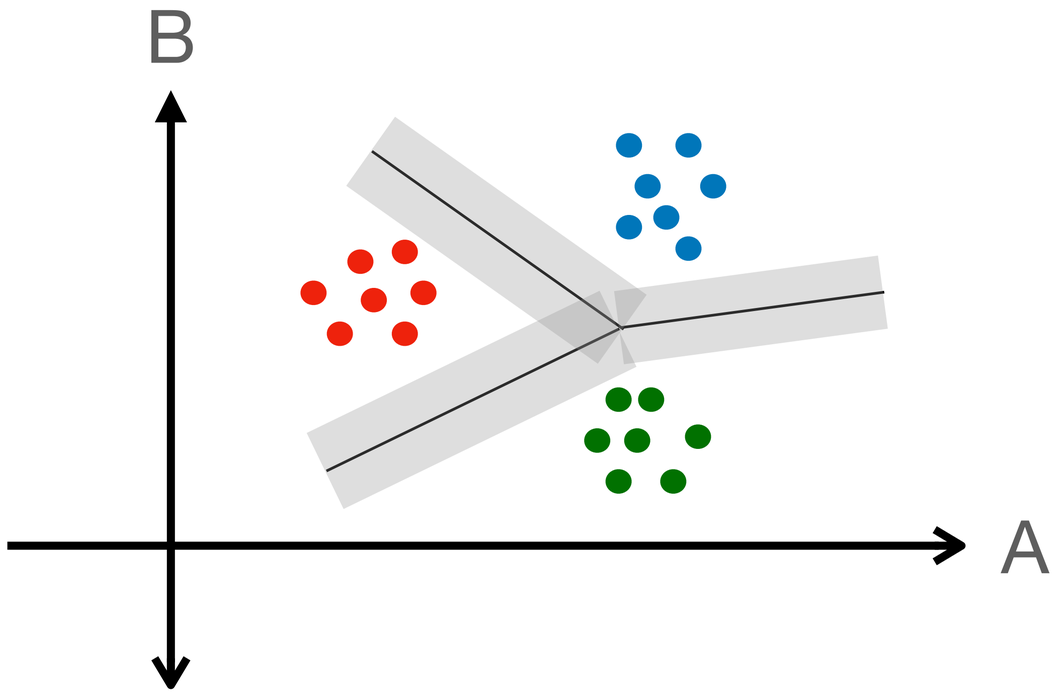
スパコンでの機械学習実行超入門

2022/4/19 v1.2

東京大学 THE UNIVERSITY OF TOKYO
東京大学情報基盤センター INFORMATION TECHNOLOGY CENTER, THE UNIVERSITY OF TOKYO

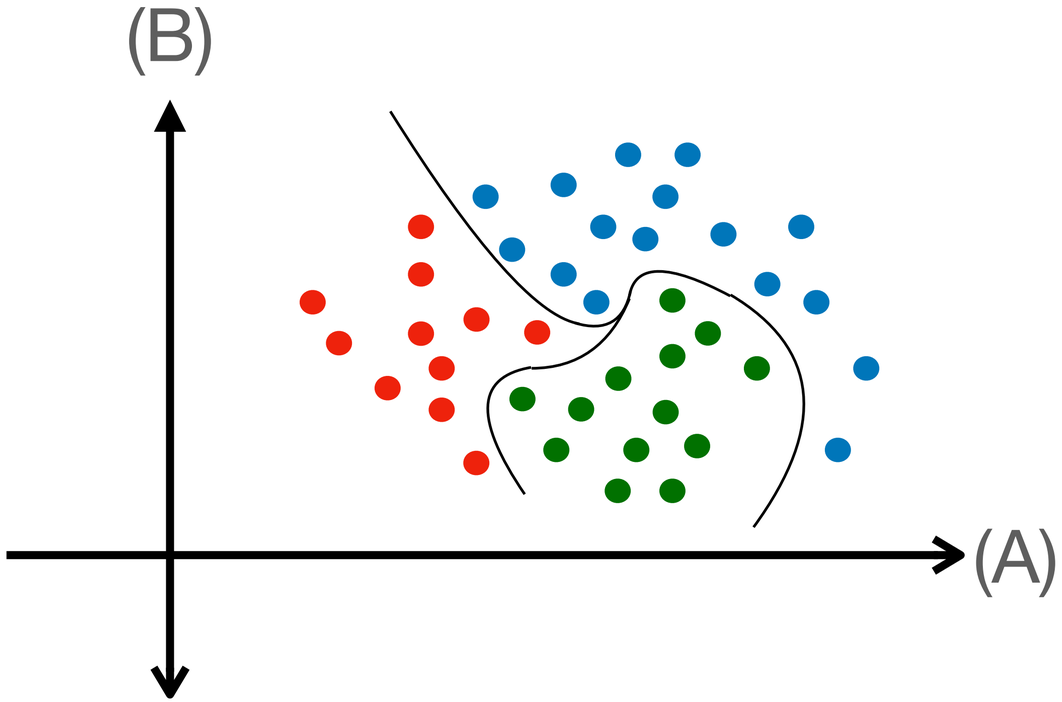
機械学習

分類問題の例



線形関係，多変量重回帰

→ 機械学習



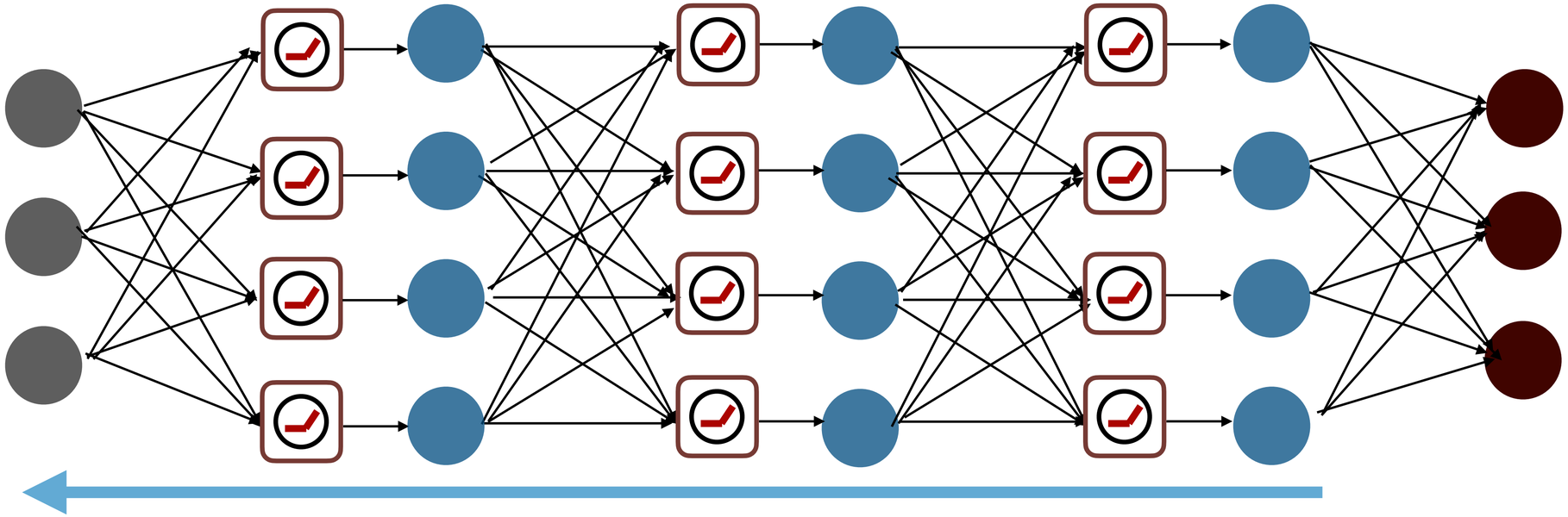
特徴量と分類対象の間に非線形関係

→ **深層学習**

深層学習とは

損失関数

誤差評価



入力層

‖

条件

出力層

‖

予測, 分類対象

誤差逆伝搬

損失関数を微分し誤差伝搬

各層の重みパラメーターを更新

→「知能」を獲得

$$y = \sum_j w_j x_j \qquad z = F(y)$$

w_j : 重み関数

活性化関数 = 非線形

ReLU, Tanh, Sigmoid,…

機械学習ツール

- 機械学習・深層学習のツール ＝ Python が大多数　（他 R, Octave など）
  - 理由？
    - シンタックス・文法の単純さ
    - numpy など行列計算のライブラリが成熟
    - 圧倒的多数が使ってるから（ソフトウェアスタック・エコシステム）

✓ もちろん10年後はどうなってるかはわかりません。。。

- フレームワークを利用
  - ニューラルネットワークの自在な設計ができる
  - 逆誤差伝搬用の高速な自動微分、典型的な最適化関数が提供されている
  - GPGPUが気軽に高速利用できる
  - 並列処理が気軽に利用できる　（スーパーコンピュータ利用の最大の理由！）

✓ **PyTorch**, TensorFlow, Keras, Caffe, (Chainer)

機械学習とGPU, スーパーコンピュータ

- 機械学習・深層学習では、ひたすら行列の積和演算を行う
  - ３Dグラフィックスとの処理の類似性
  - GPU はこの計算が得意，機械学習はGPU開発戦略の中心
  - 概してCPUの何十倍という速度で処理ができる

- 機械学習の大規模化、大量のGPU でないと学習できないモデルも

  例）画像　Google JFT-300M ＝ 30億画像
  - Transformer, MLP-Mixer など大規模向け学習モデルの発展
  - Selene@NVIDIA社，MN3@PFN　など民間企業も自らスパコン保有

参考：深層学習における並列化

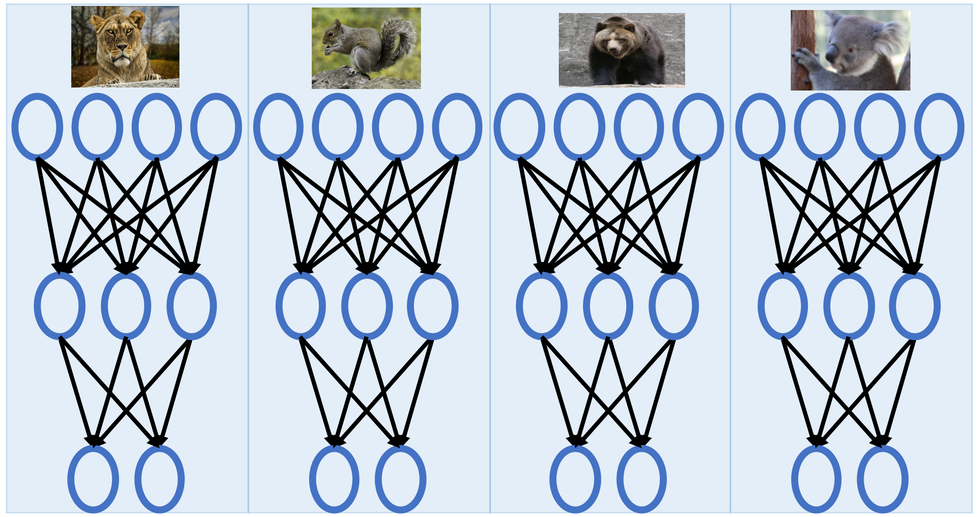
複数のプロセッサやGPUで処理するための２つの方法

データ並列学習

データを分割，ニューラルネットはコピー

1.異なるデータを処理して個々にモデルを更新

2.互いのパラメータを持ち寄って平均を取る

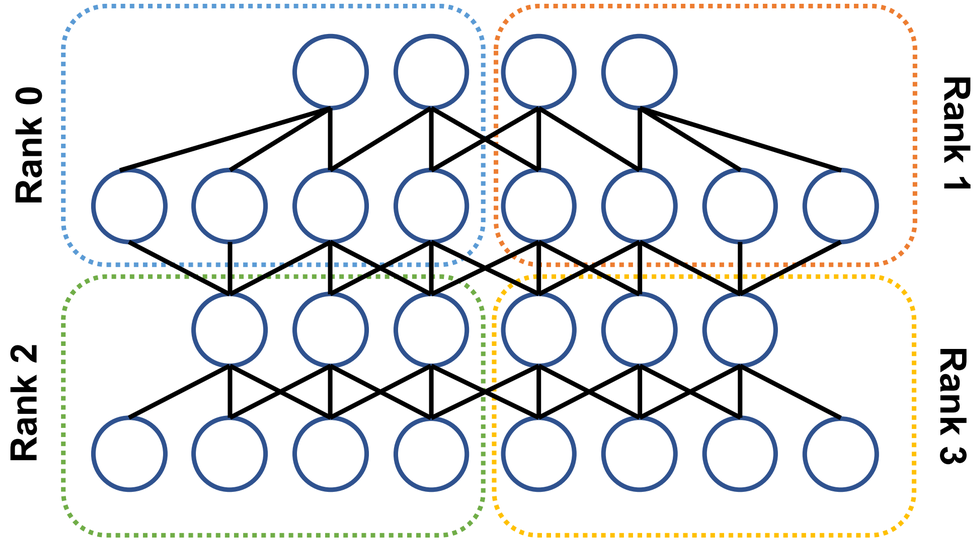


多くのフレームワークでそのまま利用可

モデル並列学習

モデルを分割して効率化

自動ツールは研究開発中（例: RaNNC @ NICT）



使用は技術的熟練が必要

実習：Python3 環境設定 (1/5)

最初に、/work に作業環境を用意します。

```
[tUVXYZ@obcx01 ~]$ cd /work/gt00/tUVXYZ/ Enter
[tUVXYZ@obcx01 tUVXYZ]$ mkdir deeplearning Enter
[tUVXYZ@obcx01 tUVXYZ]$ cd deeplearning Enter
```

機械学習を行う上で、現在使っている環境は重要です。
OBCX で使われている OS とデフォルトのPyhonのバージョンを確認してみます。

```
[tUVXYZ@obcx01 deeplearning]$ cat /proc/version Enter
Linux version 3.10.0-957.21.3.el7.x86_64 (mockbuild@x86-017.build.eng.bos.redhat.com)
(gcc version 4.8.5 20150623 (Red Hat 4.8.5-36) (GCC) ) #1 SMP Fri Jun 14 02:54:29 EDT 2019
[tUVXYZ@obcx01 deeplearning]$ python --version Enter
Python 2.7.5
[tUVXYZ@obcx01 deeplearning]$ python3 Enter
-bash: python3: command not found
```

OS は Red Hat Enterprise Linux 7，デフォルトは Python v2 であることがわかります。

実習：Python3 環境設定 (2/5)

★ PyTorch など最新のフレームワークでは、多くの場合 Python 3 が要求されます。
OBCXではデフォルトのPythonを使用せず、環境構築を進める必要があります。

<u>スパコンでの環境構築上の注意点</u>

- 個人のPCと異なり、一般ユーザーはroot 権限（管理者権限）を有しない
  - non-root でのアプリケーションの構築方法を習得
- 環境構築はインターネットに接続しているノードから (pip download)
  - OBCXではログインノードから行う必要
- 学習は計算ノードで実行する。
  - 計算ノードから参照可能な /work 領域でライブラリを構築する
- Python の各種ライブラリは、後方互換性のないアップデートがしばしば
  - pip, miniconda などを用いてバージョン整合性のある環境を構築
- 研究開発内容ごとに異なる環境を用意するために、しばしば仮想環境を使用
  - 本日の実習では、pip 仮想環境も選択可 (Option B)
  - スパコンでもDocker コンテナ を利用することが、多くの場合可能です

実習：Python3 環境設定 (3/5)

まずはPython３環境を module load します

```
[tUVXYZ@obcx01 deeplearning]$ module purge          Enter
[tUVXYZ@obcx01 deeplearning]$ module load python/3.7.3     Enter
[tUVXYZ@obcx01 deeplearning]$ python3 --version    Enter
Python 3.7.3
```

次にパッケージマネージャー (pip3) の使用環境を整えます。２つオプションがありますので、選択してください。

Option 1	pip3 user install を使用	← 初学者
Option 2	pip3 仮想環境を使用	← 経験者

実習：Python3 環境設定 (4/5)

Option 1 - pip3 の user install を使用

pip で--user オプションを付与したときのインストール先を指定します。
PYTHONUSERBASE 環境変数で指定すると、このディレクトリ以下にインストールされます。

```
[tUVXYZ@obcx01 deeplearning]$
                export PYTHONUSERBASE=/work/gt00/tUVXYZ/deeplearning/.local  Enter
[tUVXYZ@obcx01 deeplearning]$ echo $PYTHONUSERBASE    Enter
/work/gt00/t00570/deeplearning/.local
```

pip3（=python3 のパッケージマネージャー）を最新版にアップデートします。

```
[tUVXYZ@obcx01 deeplearning]$ pip3 install --upgrade pip --user    Enter
```

実習：Python3 環境設定 (5/5)

Option 2 - pip3 仮想環境を使用

仮想環境　＝　あるアプリケーションを使用するとき専用の通常と分離された環境

仮想環境の作成

```
[tUVXYZ@obcx01 deeplearning]$  python3 -m venv torch_env    Enter
```

仮想環境が格納された “torch_env” というディレクトリが作成されます。

仮想環境起動時に必要なライブラリの環境変数が設定されるようにします。

```
[tUVXYZ@obcx01 deeplearning]$ echo  export LD_LIBRARY_PATH=/work/opt/local/apps/python/
3.7.3/lib:$LD_LIBRARY_PATH >> ./torch_env/bin/activate    Enter        （途中改行せず、連続で記入）
```

ここから、仮想環境を起動します。仮想環境名が入ったプロンプトが返ってきます。

pip3 を最新版にアップデートします。

```
[tUVXYZ@obcx01 deeplearning]$ source ./torch_env/bin/activate    Enter
(torch_env) [tUVXYZ@obcx01 deeplearning]$  pip3 install --upgrade pip    Enter
```

Install PyTorch

HPで必要要件とインストール方法を確認

https://pytorch.org

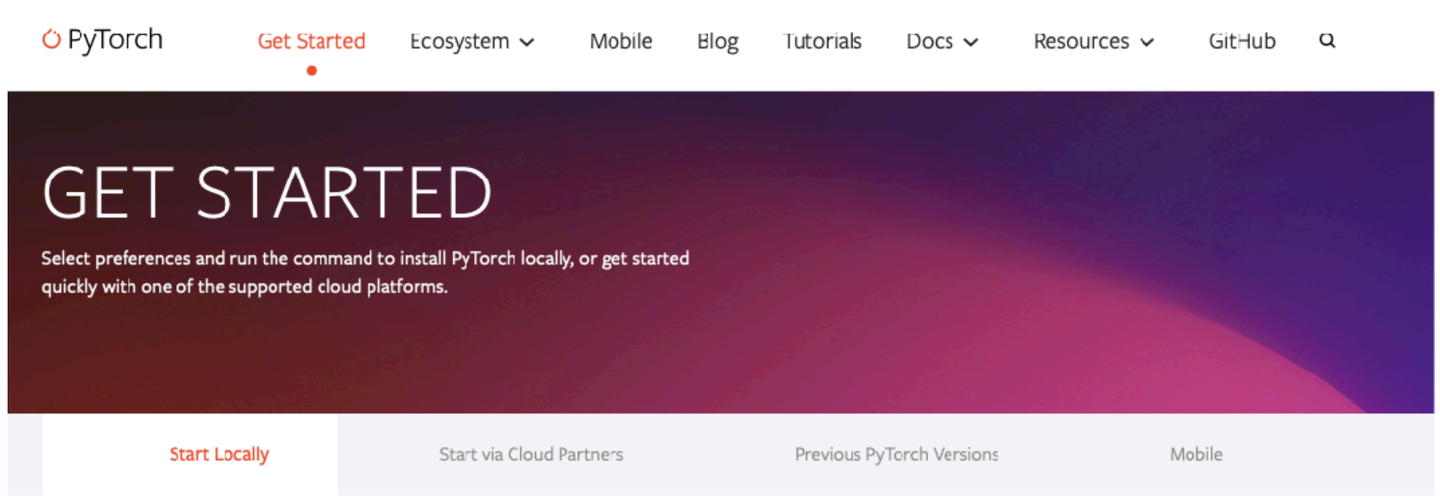
>> “Get Started”

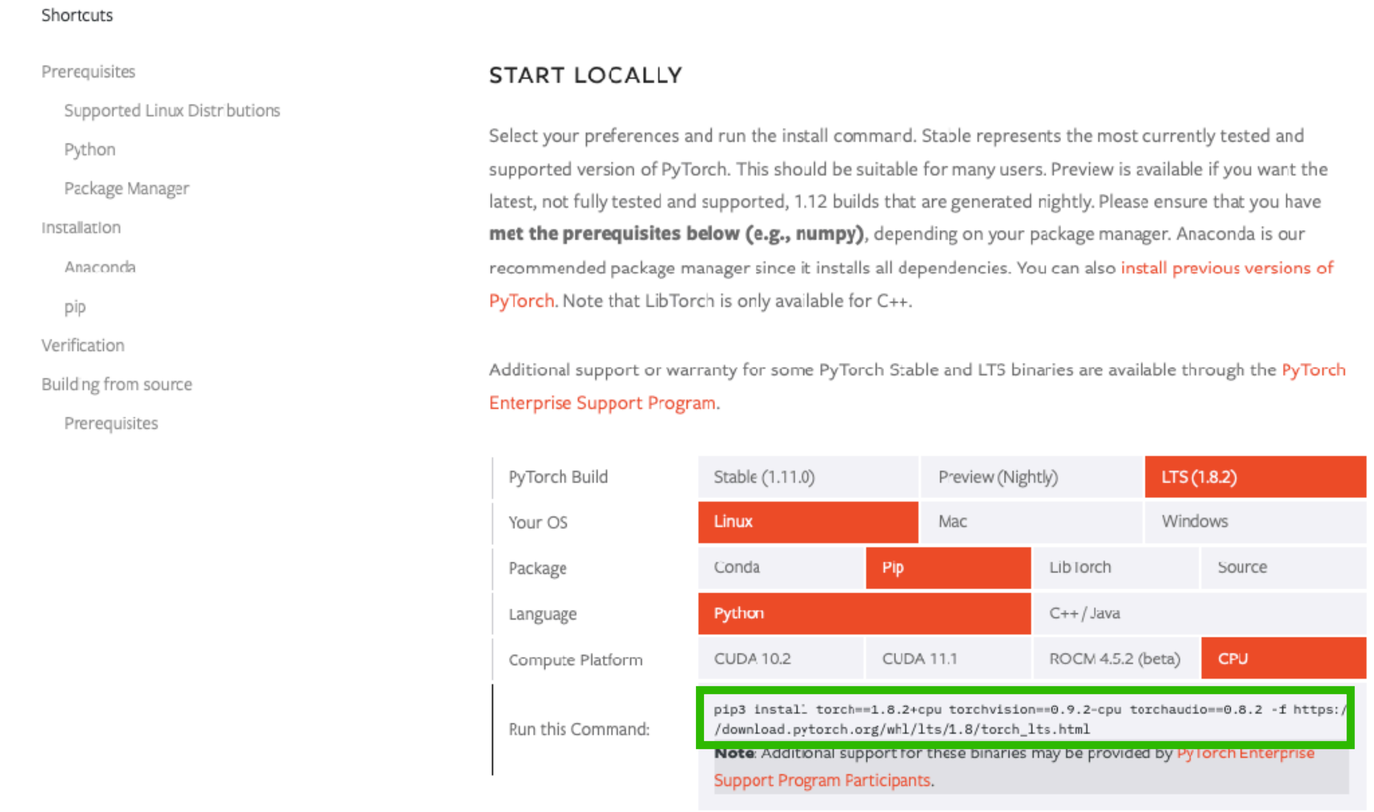
>> “Start Locally”

OBCX はGPUがないので

CPU版 PyTorch LTS (1.8.2)

を導入してみます。





PyTorchのインストール

PyTorch のホームページから得られたコマンドをそのまま入力します（改行なし）。

```
[tUVXYZ@obcx01 deeplearning]$  pip3 install torch==1.8.2+cpu torchvision==0.9.2+cpu
torchaudio==0.8.2 -f https://download.pytorch.org/whl/lts/1.8/torch_lts.html --user   Enter
```

（末尾の“--user”はOption 2では不要）

“pip3 list” でインストール済みパッケージを確認してみます。

“torch 1.8.2+cpu” が入っていればインストール成功です。

```
[tUVXYZ@obcx01 deeplearning]$  pip3 list  Enter
Package            Version
------------------ ---------
numpy              1.21.6
Pillow             9.1.0
pip                22.0.4
setuptools         40.8.0
torch              1.8.2+cpu
torchaudio         0.8.2
torchvision        0.9.2+cpu
typing_extensions 4.1.1
```

サンプルプログラムで機械学習を実行（1/4)

本日は，PyTorch 公式チュートリアルにあるFashionMNISTの画像分類を行います。

https://pytorch.org/tutorials/beginner/basics/quickstart_tutorial.html

1-1．データセット準備

OBCX の計算ノードではインターネットに接続不可

→ 事前にログインノードにてダウンロード

```
[tUVXYZ@obcx01 deeplearning]$ cp /work/gt00/share/z30122/torch_sample/download.py . Enter
[tUVXYZ@obcx01 deeplearning]$ python3 download.py Enter
```

download.py （抜粋）

```
training_data = torchvision.datasets.FashionMNIST(
    root="data",
    train=True,
    download=True,
    transform=torchvision.transforms.ToTensor()
)
```

ディレクトリ “data” に
データー式 (~30 MB)が
ダウンロードされます

サンプルプログラムで機械学習を実行（2/4)

1-2. ジョブ準備

フルのコードを単一の Python スクリプト (samp.py) として
として用意しました。手元にコピーしてください。

```
[tUVXYZ@obcx01 deeplearning]$ cp /work/gt00/share/z30122/torch_sample/samp.py .  Enter
```

次にジョブスクリプトを作成します（次頁）

サンプルプログラムで機械学習を実行 (3/4)

2. ジョブスクリプト作成

```
[tUVXYZ@obcx01 deeplearning]$ emacs job.sh    Enter
```

Option 1の方 (pip3 user install)

job.sh

```
#!/bin/bash
#PJM -L rscgrp=tutorial
#PJM -L node=1
#PJM -L elapse=0:15:00
#PJM -g gt00
#PJM -N fashionMNIST
#PJM -o result.txt
#PJM -j

module load python/3.7.3
export PYTHONUSERBASE=/work/gt00/tUVXYZ
    /deeplearning/.local    （改行なし連続）
python3 samp.py
```

Option 2の方 (pip3 仮想環境)

job.sh

```
#!/bin/bash
#PJM -L rscgrp=tutorial
#PJM -L node=1
#PJM -L elapse=0:15:00
#PJM -g gt00
#PJM -N fashionMNIST
#PJM -o result.txt
#PJM -j

source ./torch_env/bin/activate
python3 samp.py
```

サンプルプログラムで機械学習を実行（4/4)

3. 機械学習の実行と結果確認

ジョブを投入　[tUVXYZ@obcx01 deeplearning]$ pjsub job.sh　Enter

学習結果確認　[tUVXYZ@obcx01 deeplearning]$ less result.txt　Enter

以下のように学習の結果が表示されます。

```
Using cpu device
Shape of X [N, C, H, W]: torch.Size([64, 1, 28, 28])
Shape of y: torch.Size([64]) torch.int64
NeuralNetwork(
  (flatten): Flatten(start_dim=1, end_dim=-1)
  (linear_relu_stack): Sequential(
    (0): Linear(in_features=784, out_features=512, bias=True)
    (1): ReLU()
    (2): Linear(in_features=512, out_features=512, bias=True)
    (3): ReLU()
    (4): Linear(in_features=512, out_features=10, bias=True)
  )
)
```

```
Epoch 1
-------------------------------
loss: 2.300122  [    0/60000]
loss: 2.292114  [ 6400/60000]
loss: 2.272950  [12800/60000]
loss: 2.271642  [19200/60000]
loss: 2.254692  [25600/60000]
loss: 2.223023  [32000/60000]
loss: 2.236395  [38400/60000]
loss: 2.201323  [44800/60000]
loss: 2.201231  [51200/60000]
loss: 2.168899  [57600/60000]
Test Error:
 Accuracy: 42.9%, Avg loss: 2.164772
(中略)
```

```
Epoch 5
-------------------------------
loss: 1.343366  [    0/60000]
loss: 1.312548  [ 6400/60000]
loss: 1.152993  [12800/60000]
loss: 1.247405  [19200/60000]
loss: 1.125050  [25600/60000]
loss: 1.158031  [32000/60000]
loss: 1.172069  [38400/60000]
loss: 1.114118  [44800/60000]
loss: 1.147697  [51200/60000]
loss: 1.061013  [57600/60000]
Test Error:
 Accuracy: 65.2%, Avg loss: 1.084928
Done!
```

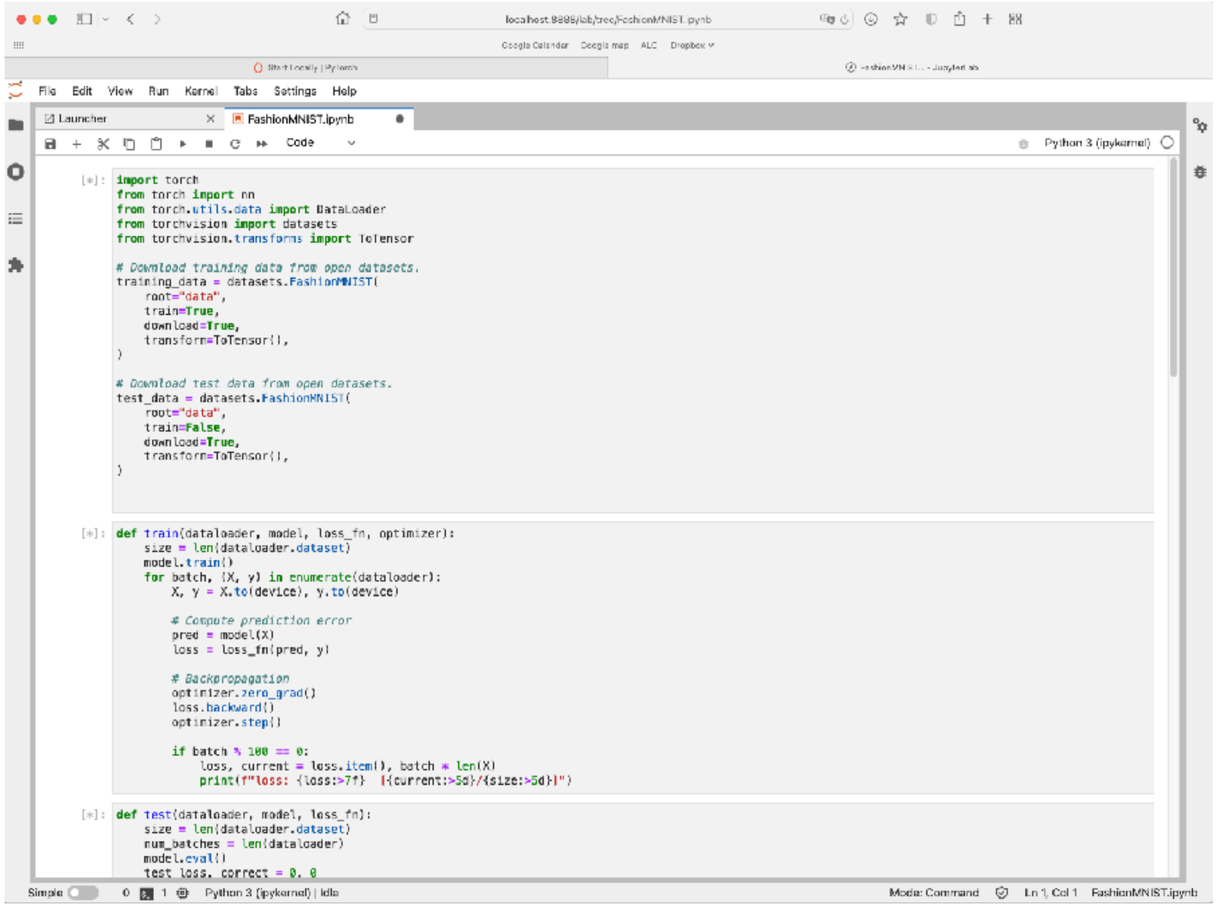
OBCX でご自身のJupyter Notebookを動かすには

```
[tUVXYZ@obcx01 deeplearning]$  jupyter nbconvert --to python  [FileName].ipynb
```

上記コマンドで出力された

[FileName].py

をジョブスクリプトで実行してください。



✳ Wisteria/BDEC-01では，JupyterHub環境が使用可能です。JupyterLabから直接ジョブを（計算ノードへ）投入できる仕組みがあります。