29th Sep. 2022

187th Parallel Programming Workshop with Trial Account

“Supercomputing for Beginners”

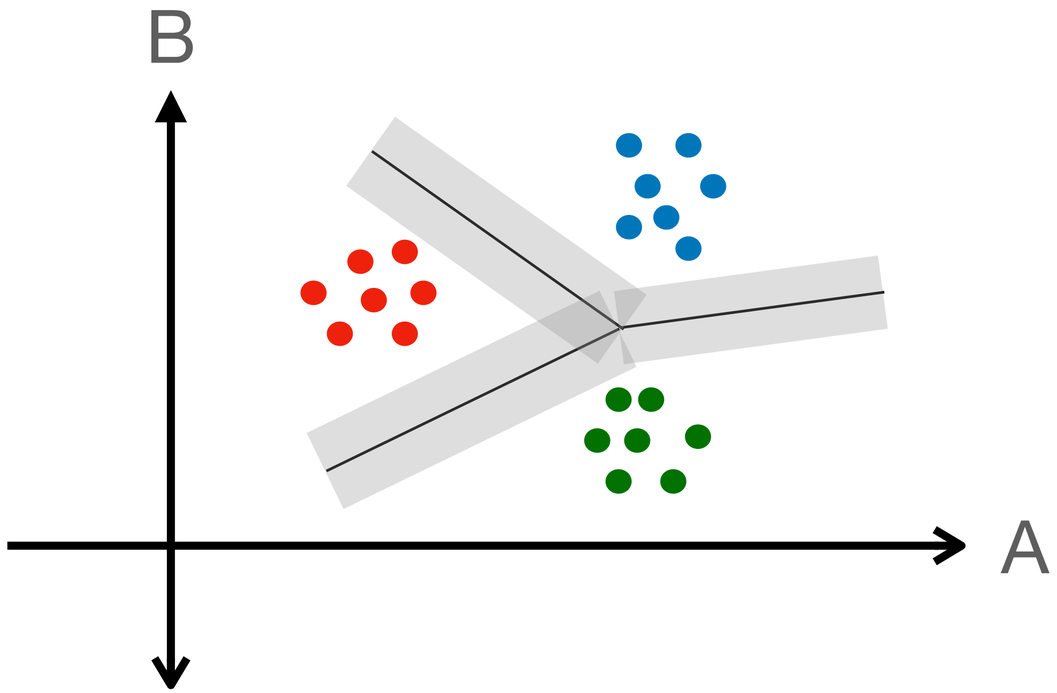
Running machine learning on supercomputers — for beginners

29th Sep 2022 v1.0

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

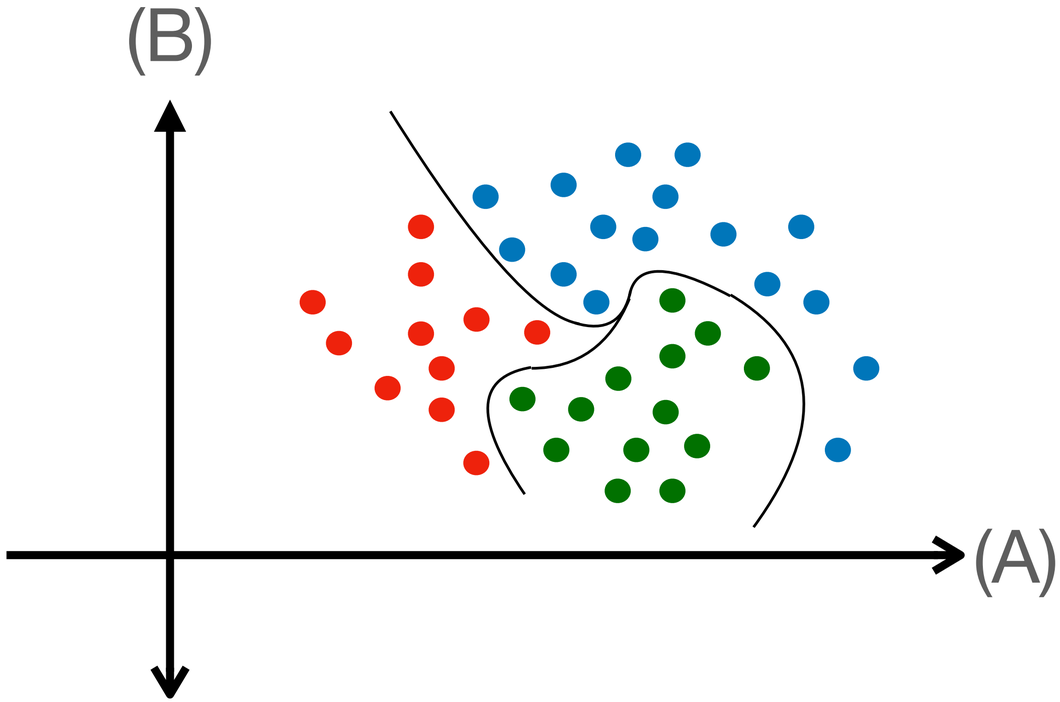
Machine learning

an example in classification tasks



linear relation, multiple regression

→ machine learning



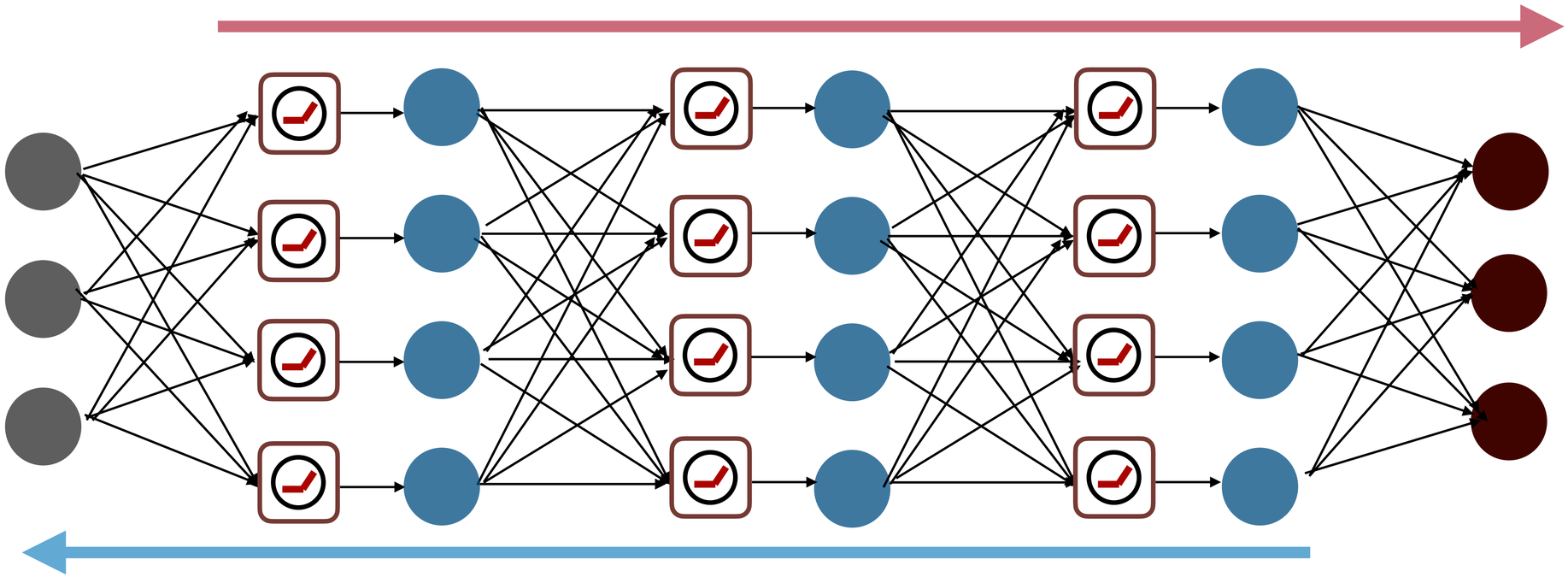
nonlinear relationship between the feature values and classes

→ **deep learning**

Deep learning

loss function

error estimate



input layer

output layer ‖ target

backprop

propagate error by differentiation

update weight in each layer

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

w_j : weight

activation function = nonlinear

ReLU, TanH, Sigmoid,…

Tools for machine learning

- majority = using Python (+ R, Octave, etc)
  - Why?
    - simple syntax and language
    - fast & convenient libraries for matrix operations, incl. numpy
    - many people use it :) — software stacks, ecosystems

      ✓ we don't know how it will change after 10 years, of course…
- specialized frameworks for machine learning are recommended
  - neural network can be easily designed
  - fast autograd for backprops, optimization tools
  - runs fast on GPGPUs
  - easy parallel learning (important for ML + HPC)

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

Machine learning and GPUs, supercomputers

- In ML/DL, most arithmetics are multiple-add matrix operations
  - affinity with 3D graphics
  - GPUs are good at such operations. Therefore, deep learning is one of the main target application in new GPU development.
  - typically tens of times faster than CPUs.

- Machine learning is becoming bigger and bigger, sometimes a massive amount of GPUs are necessary

  example）Google JFT-300M = 3 billion images
  - NN models for huge-scale learning, such as Transformer & MLP-Mixer
  - supercomputers in private sectors — Selene@NVIDIA, MN3@PFN, and etc.

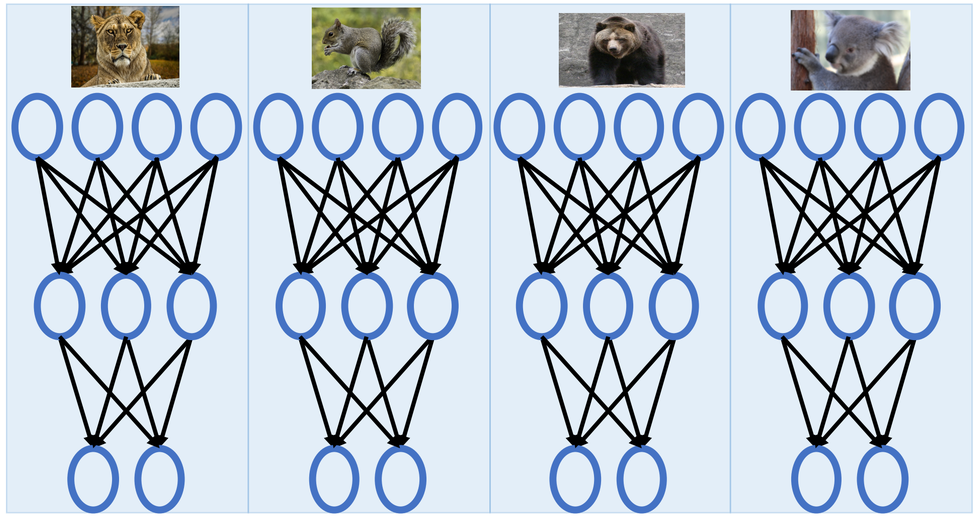
parallelization in deep learning

2 classes of methods for learning on multiple processors / GPUs.

Data parallel

Data divided, the network is copied over GPUs

1. update NN model with different data
2. take averages over weights



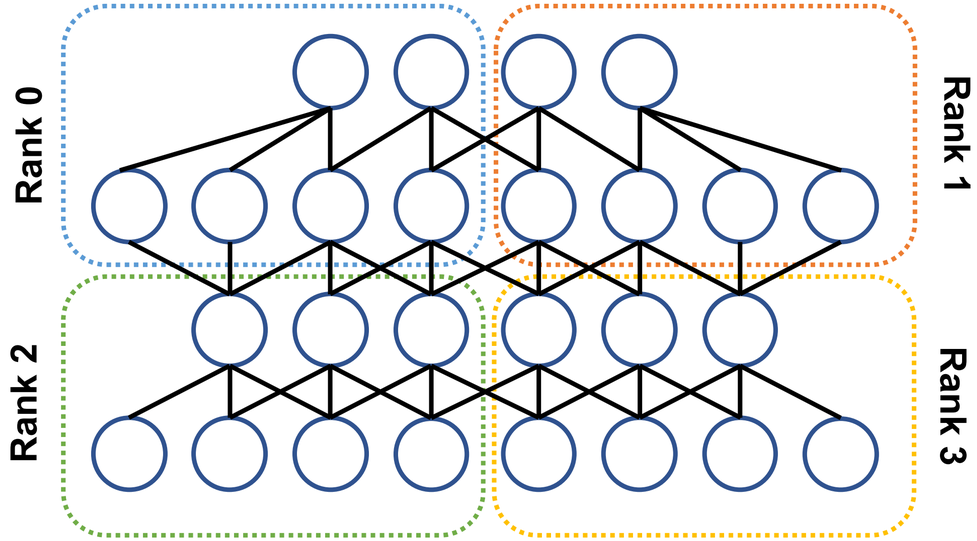
immediately available in many frameworks

Model parallel

divide model into different ranks

automation tools being developed

(ex: RaNNC@NICT)



expertise required

Exercise: python3 environment settings (1/5)

First, prepare your environment in /work directory.

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

It is important to know the current environment.
Let us confirm the operating system (on OBCX) and the version of python.

```
[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
```

As we are using is Red Hat Enterprise Linux 7, the default python is v2.

Exercise: python3 environment settings (2/5)

★ Many of the latest frameworks incl. PyTorch require Python 3.

Therefore, we cannot use default python on OBCX.

<u>Cautions for environment settings on supercomputers</u>

- Users do not have root (administrator) privileges, unlike on personal computers.
  - you need to work as non-root.
- The environment should be set up on the nodes that are connected to the internet.
  - on OBCX, therefore, we need to use the login node.
- We will run machine learning jobs on the compute nodes.
  - libraries should be built (set up) on /work on OBCX.
- Updates in python libraries often lack backward compatibility
  - we use pip, miniconda, and etc, in order to keep consistency.
- We may use virtual environments, if we do different machine learning with different tools.
  - you may practice today how to use pip virtualenv (Option B)
  - it is possible to use Docker containers even on supercomputers

Exercise: python3 environment settings (3/5)

First, load python3 environment on OBCX.

```
[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
```

Then we set up the package manager (pip3). You may choose one.

Option 1	pip3 user install	← for a beginner
Option 2	pip3 virtual env	← if you are familiar with python+ML

Exercise: python3 environment settings (4/5)

Option 1 - pip3 user install

set up the installation directory when --user option is used in pip3.

set it “PYTHONUSERBASE” environment variable

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

update pip3 (= package manager for python3) to the latest version.

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

Exercise: python3 environment settings (5/5)

Option 2 - pip3 virtual environment

virtual environment = separated environment intended for a single specific application

create a virtual environment

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

You will find a directory “torch_env”, wherein the virtual environment is set up.

We modify the script for activating the virtual environment as follows:

```
[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          (no linebreak !)
```

Then we launch the virtual environment. You will have the prompt with its name.
update pip3 (= package manager for python3) to the latest version.

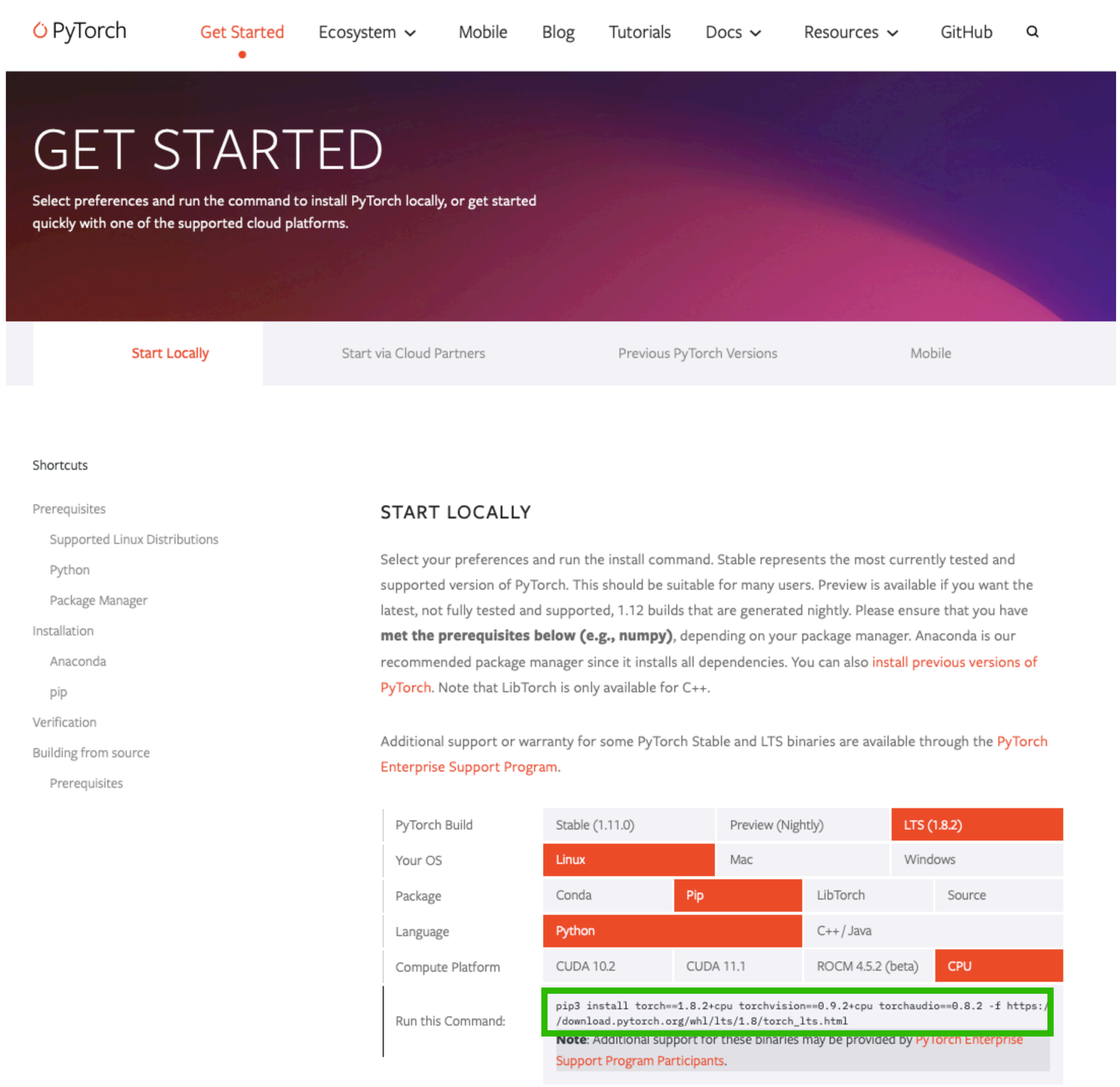
```
[tUVXYZ@obcx01 deeplearning]$ source ./torch_env/bin/activate    Enter
(torch_env) [tUVXYZ@obcx01 deeplearning]$  pip3 install --upgrade pip    Enter
```

Install PyTorch

Confirm which environments are needed and how to install on the website.

https://pytorch.org
>> “Get Started”
>> “Start Locally”

There are no GPUs on OBCX. We install PyTorch LTS (1.8.2) for CPUs.



Installing PyTorch

Type the command as it is (copy from the PyTorch web site ,with no linebreaks!)

```
[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
```

(“--user” is unnecessary for Option 2)

Confirm installed packages
by “pip3 list”

Installation successful if you find
“torch 1.8.2+cpu” in the list

```
[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
```

exercising machine learning with sample program (1/4)

Today, we will run image classification task of FashionMNIST on PyTorch tutorial
https://pytorch.org/tutorials/beginner/basics/quickstart_tutorial.html

1-1 preparation of the dataset

Compute nodes on OBCX are not connected to the internet
→ We prepare the dataset in advance

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

download.py (excerpt)

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

The dataset is downloaded
into the directory “data” (< 30 MB)

exercising machine learning with sample program (2/4)

1-2. preparation of the python script

We prepared a single Python script (in ful) as “samp.py”.
Copy it to your directory.

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

Then step forward to making your job scripts. (next page)

exercising machine learning with sample program (3/4)

2. write your job script

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

Option 1 (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   (no line break)
python3 samp.py
```

Option 2 (pip3 virtualenv)

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
```

sample code: samp.py

```python
import torch
from torch import nn
from torch.utils.data import DataLoader
from torchvision import datasets
from torchvision.transforms import ToTensor


# Download training data from open datasets.
training_data = datasets.FashionMNIST(
    root="data",
    train=True,
    download=False,
    transform=ToTensor(),
)

# Download test data from open datasets.
test_data = datasets.FashionMNIST(
    root="data",
    #root="/work/00/gt00/z30122/pytorch_tutorial/data",
    train=False,
    download=False,
    transform=ToTensor(),
)



def train(dataloader, model, loss_fn, optimizer):
    size = len(dataloader.dataset)
    model.train()
    for batch, (X, y) in enumerate(dataloader):
        X, y = X.to(device), y.to(device)

        # Compute prediction error
        pred = model(X)
        loss = loss_fn(pred, y)

        # Backpropagation
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

        if batch % 100 == 0:
            loss, current = loss.item(), batch * len(X)
            print(f"loss: {loss:>7f}  [{current:>5d}/{size:>5d}]")
```

```python
# Define model
class NeuralNetwork(nn.Module):
    def __init__(self):
   super(NeuralNetwork, self).__init__()
   self.flatten = nn.Flatten()
   self.linear_relu_stack = nn.Sequential(
            nn.Linear(28*28, 512),
            nn.ReLU(),
            nn.Linear(512, 512),
            nn.ReLU(),
            nn.Linear(512, 10)
   )

    def forward(self, x):
        x = self.flatten(x)
        logits = self.linear_relu_stack(x)
        return logits
```

```python
batch_size = 64

device = "cuda" if torch.cuda.is_available() else "cpu"
print(f"Using {device} device")

# Create data loaders.
train_dataloader = DataLoader(training_data, batch_size=batch_size)
test_dataloader = DataLoader(test_data, batch_size=batch_size)

for X, y in test_dataloader:
    print(f"Shape of X [N, C, H, W]: {X.shape}")
    print(f"Shape of y: {y.shape} {y.dtype}")
    break

model = NeuralNetwork().to(device)
print(model)

loss_fn = nn.CrossEntropyLoss()
optimizer = torch.optim.SGD(model.parameters(), lr=1e-3)

epochs = 5
for t in range(epochs):
    print(f"Epoch {t+1}\n-------------------------------")
    train(train_dataloader, model, loss_fn, optimizer)
    test(test_dataloader, model, loss_fn)
print("Done!")
```

exercising machine learning with sample program (4/4)

3. execute machine learning and confirm the result

submit the job

```
[tUVXYZ@obcx01 deeplearning]$ pjsub job.sh
```

Enter

confirm the result

```
[tUVXYZ@obcx01 deeplearning]$ less result.txt
```

Enter

It is successful if you get the following result.

```
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
...... ......
```

```
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!
```

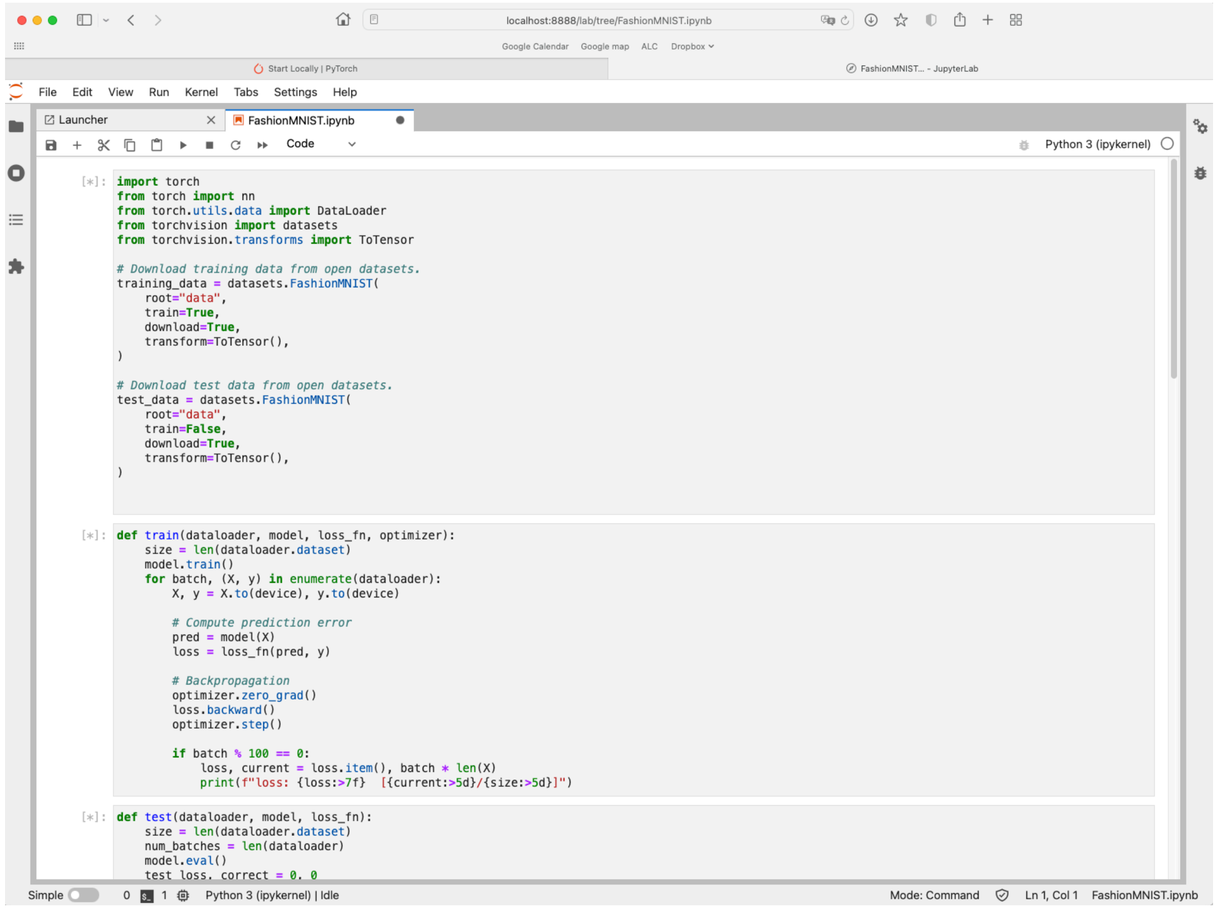
If you want to run your Jupyter Notebook on OBCX

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

The above command generates

[FileName].py

file. Run it in your job script.



- On Wisteria/BDEC-01, JupyterHub interface is available, where Python script jobs can directly be submitted (to compute nodes) from your Jupyter Notebook.