Hewlett Packard Enterprise

SMOKE TEST

Optimized PyTorch

Date Prepared: Aug 2020

Hewlett Packard Enterprise

Document Information

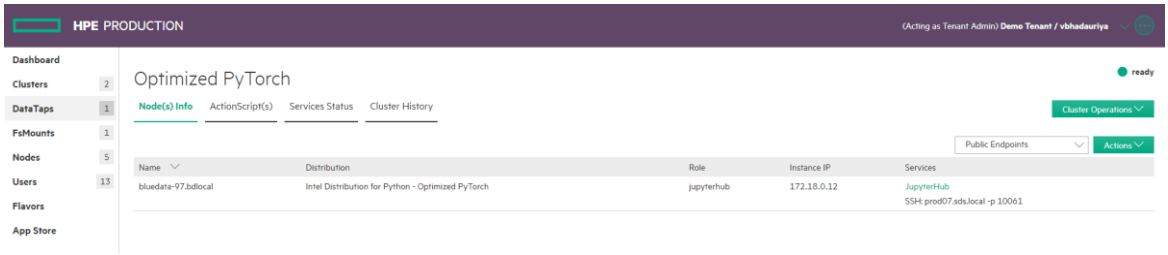
Project Name	**EPIC Accelerator Deployment & Integration Services**		
Project Owner		Document Version No	0.2
Quality Review Method			
Prepared By	Vivek	Preparation Date	Aug 2020
Reviewed By		Review Date	

Table of Contents

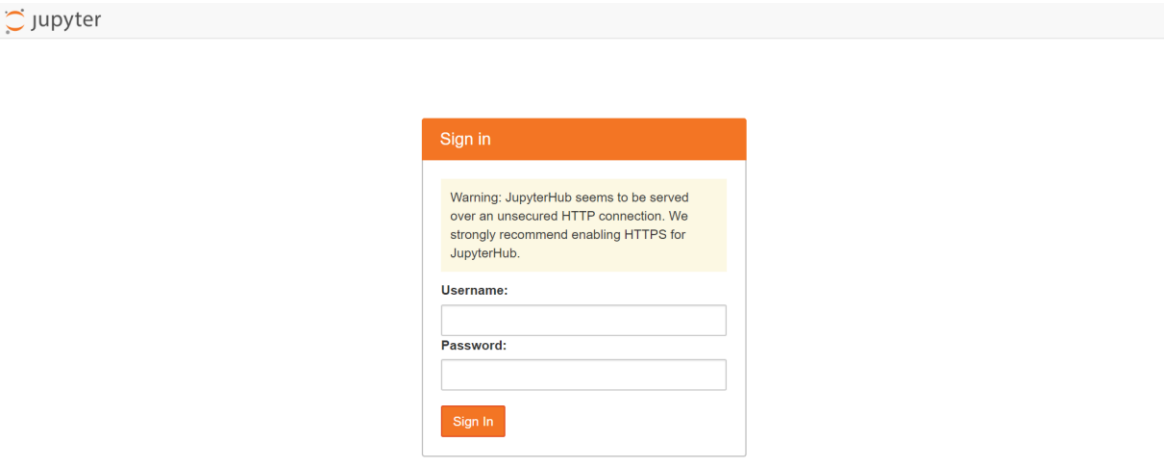
1 LOGIN TO JUPYTER HUB WEB UI

Login to the JupyterHub Web UI, using the following steps.

1. From the Cluster page under Services, click on **JupyterHub**



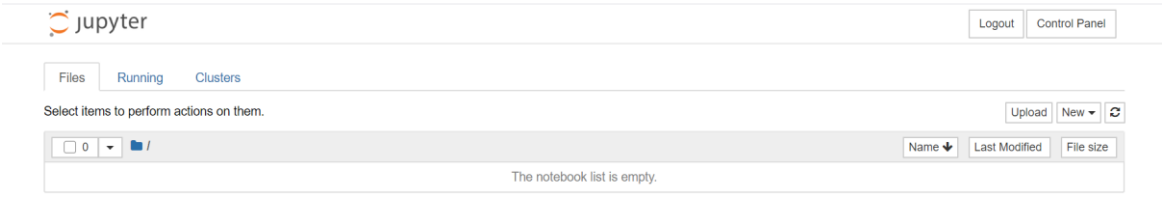
2. It will ask for username and password, insert you AD username and password and login to Jupyter Hub.



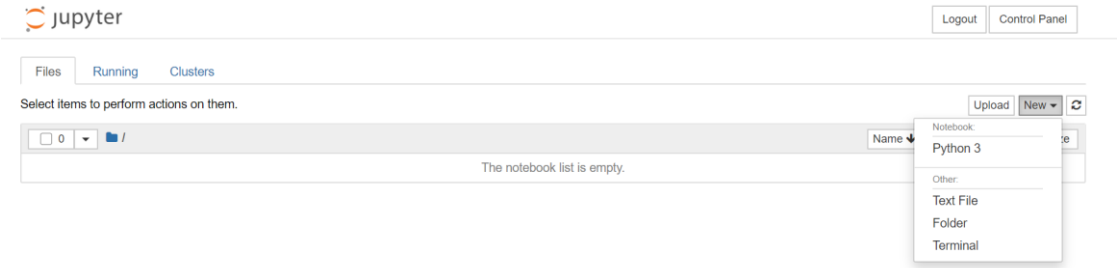
2 CREATE NOTEBOOK

To create a new notebook, in Jupyter Hub use the following steps:

1. Once you logged in to Jupyter Hub with your AD account, you will get the dashboard like below.



2. Click on New drop-down button, to create a new notebook and click on python 3.



3. In Python3 notebook, add the following code in the cell.

```
import torch
from torch.autograd import Variable
import torch.nn.functional as F
import torch.utils.data as Data


import matplotlib.pyplot as plt
%matplotlib inline


import numpy as np
import imageio


torch.manual_seed(1)    # reproducible
```

```
x = torch.unsqueeze(torch.linspace(-10, 10, 1000), dim=1)  # x data (tensor), shape=(100, 1)
y = torch.sin(x) + 0.2*torch.rand(x.size())                # noisy y data (tensor), shape=(100, 1)

# torch can only train on Variable, so convert them to Variable
x, y = Variable(x), Variable(y)
plt.figure(figsize=(10,4))
plt.scatter(x.data.numpy(), y.data.numpy(), color = "blue")
plt.title('Regression Analysis')
plt.xlabel('Independent varible')
plt.ylabel('Dependent varible')
plt.savefig('curve_2.png')
plt.show()

# another way to define a network
net = torch.nn.Sequential(
torch.nn.Linear(1, 200),
torch.nn.LeakyReLU(),
torch.nn.Linear(200, 100),
torch.nn.LeakyReLU(),
torch.nn.Linear(100, 1),
)

optimizer = torch.optim.Adam(net.parameters(), lr=0.01)
loss_func = torch.nn.MSELoss()  # this is for regression mean squared loss

BATCH_SIZE = 64
EPOCH = 200

torch_dataset = Data.TensorDataset(x, y)
```

```
loader = Data.DataLoader(
dataset=torch_dataset,
batch_size=BATCH_SIZE,
shuffle=True, num_workers=2,)

my_images = []
fig, ax = plt.subplots(figsize=(16,10))

# start training
for epoch in range(EPOCH):
for step, (batch_x, batch_y) in enumerate(loader): # for each training step

b_x = Variable(batch_x)
b_y = Variable(batch_y)

prediction = net(b_x)     # input x and predict based on x

loss = loss_func(prediction, b_y)     # must be (1. nn output, 2. target)

optimizer.zero_grad()   # clear gradients for next train
loss.backward()         # backpropagation, compute gradients
optimizer.step()        # apply gradients
if step == 1:
# plot and show learning process
plt.cla()
ax.set_title('Regression Analysis - model 3 Batches', fontsize=35)
ax.set_xlabel('Independent variable', fontsize=24)
ax.set_ylabel('Dependent variable', fontsize=24)
```

```
ax.set_xlim(-11.0, 13.0)
ax.set_ylim(-1.1, 1.2)
ax.scatter(b_x.data.numpy(), b_y.data.numpy(), color = "blue", alpha=0.2)
ax.scatter(b_x.data.numpy(), prediction.data.numpy(), color='green', alpha=0.5)
ax.text(8.8, -0.8, 'Epoch = %d' % epoch,
fontdict={'size': 24, 'color':  'red'})
ax.text(8.8, -0.95, 'Loss = %.4f' % loss.data.numpy(),
fontdict={'size': 24, 'color':  'red'})

# Used to return the plot as an image array
# (https://ndres.me/post/matplotlib-animated-gifs-easily/)
fig.canvas.draw()       # draw the canvas, cache the renderer
image = np.frombuffer(fig.canvas.tostring_rgb(), dtype='uint8')
image  = image.reshape(fig.canvas.get_width_height()[::-1] + (3,))
my_images.append(image)
# save images as a gif
imageio.mimsave('./curve_2_model_3_batch.gif', my_images, fps=12)
fig, ax = plt.subplots(figsize=(16,10))
plt.cla()
ax.set_title('Regression Analysis - model 3, Batches', fontsize=35)
ax.set_xlabel('Independent variable', fontsize=24)
ax.set_ylabel('Dependent variable', fontsize=24)
ax.set_xlim(-11.0, 13.0)
ax.set_ylim(-1.1, 1.2)
ax.scatter(x.data.numpy(), y.data.numpy(), color = "blue", alpha=0.2)
prediction = net(x)     # input x and predict based on x
ax.scatter(x.data.numpy(), prediction.data.numpy(), color='green', alpha=0.5)
plt.savefig('curve_2_model_3_batches.png')
plt.show()
```

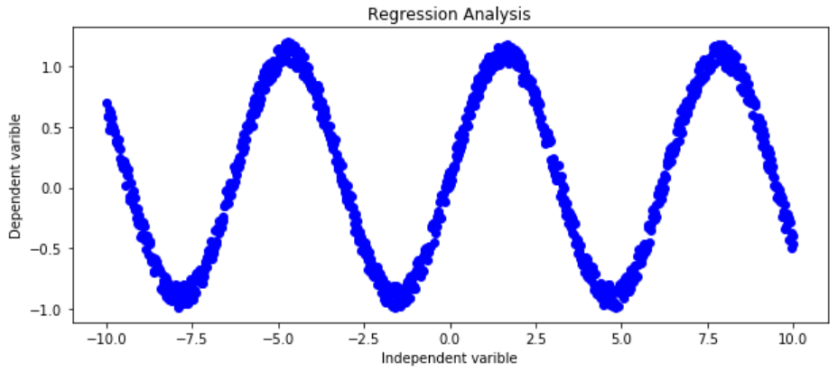
3 REGRESSION WITH NEURAL NETWORK

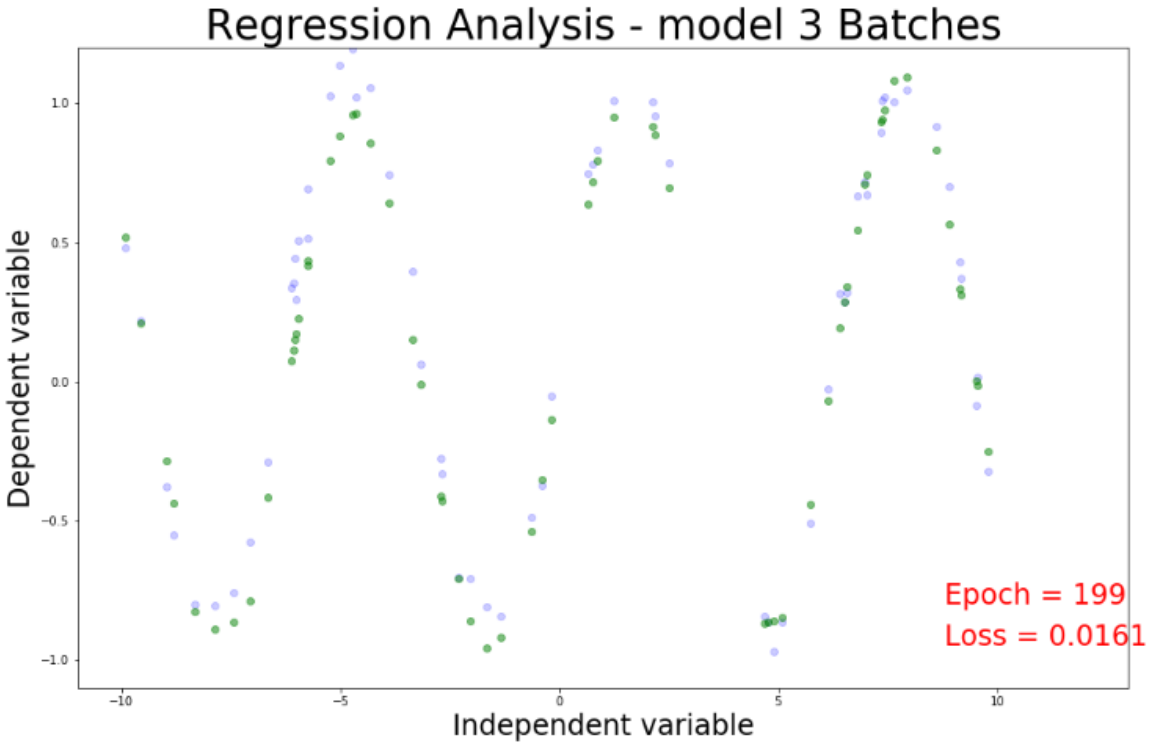
```
import torch
from torch.autograd import Variable
import torch.nn.functional as F
import torch.utils.data as Data

import matplotlib.pyplot as plt
%matplotlib inline

import numpy as np
import imageio

torch.manual_seed(1)    # reproducible

x = torch.unsqueeze(torch.linspace(-10, 10, 1000), dim=1)  # x data (tensor), shape=(100, 1)
y = torch.sin(x) + 0.2*torch.rand(x.size())                 # noisy y data (tensor), shape=(100, 1)

# torch can only train on Variable, so convert them to Variable
x, y = Variable(x), Variable(y)
plt.figure(figsize=(10,4))
plt.scatter(x.data.numpy(), y.data.numpy(), color = "blue")
plt.title('Regression Analysis')
plt.xlabel('Independent varible')
plt.ylabel('Dependent varible')
plt.savefig('curve_2.png')
plt.show()

# another way to define a network
net = torch.nn.Sequential(
        torch.nn.Linear(1, 200),
        torch.nn.LeakyReLU(),
        torch.nn.Linear(200, 100),
        torch.nn.LeakyReLU(),
        torch.nn.Linear(100, 1),
    )
```

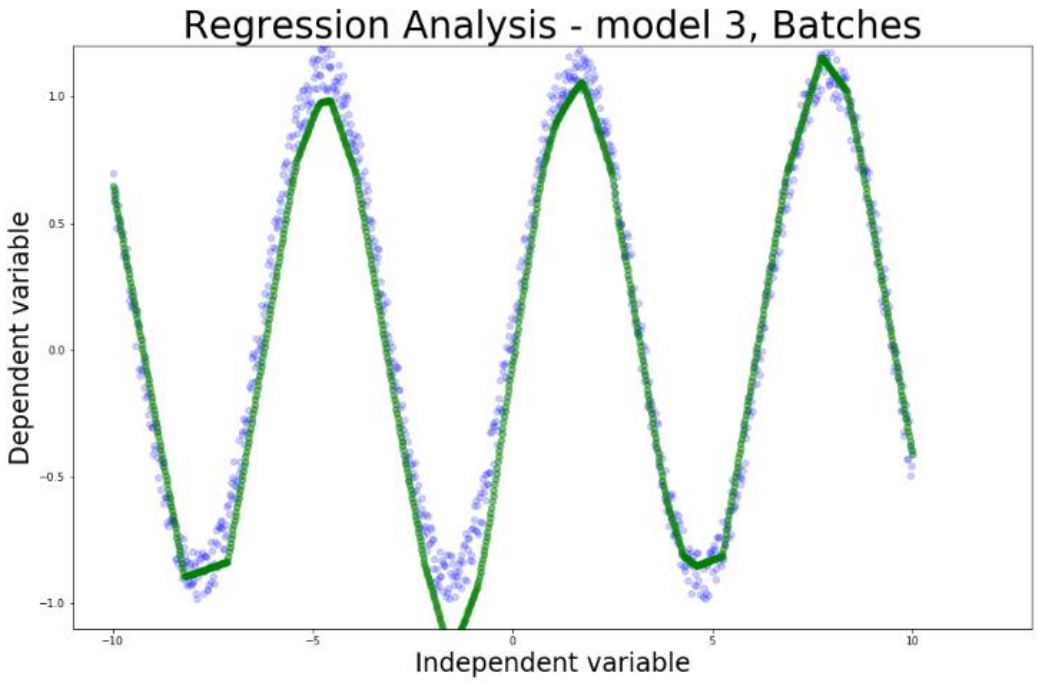
```
optimizer = torch.optim.Adam(net.parameters(), lr=0.01)
loss_func = torch.nn.MSELoss()  # this is for regression mean squared loss

BATCH_SIZE = 64
EPOCH = 200

torch_dataset = Data.TensorDataset(x, y)

loader = Data.DataLoader(
    dataset=torch_dataset,
    batch_size=BATCH_SIZE,
    shuffle=True, num_workers=2,)

my_images = []
fig, ax = plt.subplots(figsize=(16,10))

# start training
for epoch in range(EPOCH):
    for step, (batch_x, batch_y) in enumerate(loader): # for each training step

        b_x = Variable(batch_x)
        b_y = Variable(batch_y)

        prediction = net(b_x)     # input x and predict based on x

        loss = loss_func(prediction, b_y)     # must be (1. nn output, 2. target)

        optimizer.zero_grad()   # clear gradients for next train
        loss.backward()         # backpropagation, compute gradients
        optimizer.step()        # apply gradients

        if step == 1:
            # plot and show learning process
            plt.cla()
```

```
            ax.set_title('Regression Analysis - model 3 Batches', fontsize=35)
            ax.set_xlabel('Independent variable', fontsize=24)
            ax.set_ylabel('Dependent variable', fontsize=24)
            ax.set_xlim(-11.0, 13.0)
            ax.set_ylim(-1.1, 1.2)
            ax.scatter(b_x.data.numpy(), b_y.data.numpy(), color = "blue", alpha=0.2)
            ax.scatter(b_x.data.numpy(), prediction.data.numpy(), color='green', alpha=0.5)
            ax.text(8.8, -0.8, 'Epoch = %d' % epoch,
                    fontdict={'size': 24, 'color':  'red'})
            ax.text(8.8, -0.95, 'Loss = %.4f' % loss.data.numpy(),
                    fontdict={'size': 24, 'color':  'red'})

            # Used to return the plot as an image array
            # (https://ndres.me/post/matplotlib-animated-gifs-easily/)
            fig.canvas.draw()       # draw the canvas, cache the renderer
            image = np.frombuffer(fig.canvas.tostring_rgb(), dtype='uint8')
            image  = image.reshape(fig.canvas.get_width_height()[::-1] + (3,))

            my_images.append(image)



# save images as a gif
imageio.mimsave('./curve_2_model_3_batch.gif', my_images, fps=12)


fig, ax = plt.subplots(figsize=(16,10))
plt.cla()
ax.set_title('Regression Analysis - model 3, Batches', fontsize=35)
ax.set_xlabel('Independent variable', fontsize=24)
ax.set_ylabel('Dependent variable', fontsize=24)
ax.set_xlim(-11.0, 13.0)
ax.set_ylim(-1.1, 1.2)
```

```
ax.scatter(x.data.numpy(), y.data.numpy(), color = "blue", alpha=0.2)
prediction = net(x)     # input x and predict based on x
ax.scatter(x.data.numpy(), prediction.data.numpy(), color='green', alpha=0.5)
plt.savefig('curve_2_model_3_batches.png')
plt.show()
```




Regression Analysis - model 3 Batches
Dependent variable
Independent variable
Epoch = 199
Loss = 0.0161


Regression Analysis - model 3, Batches
Dependent variable
Independent variable