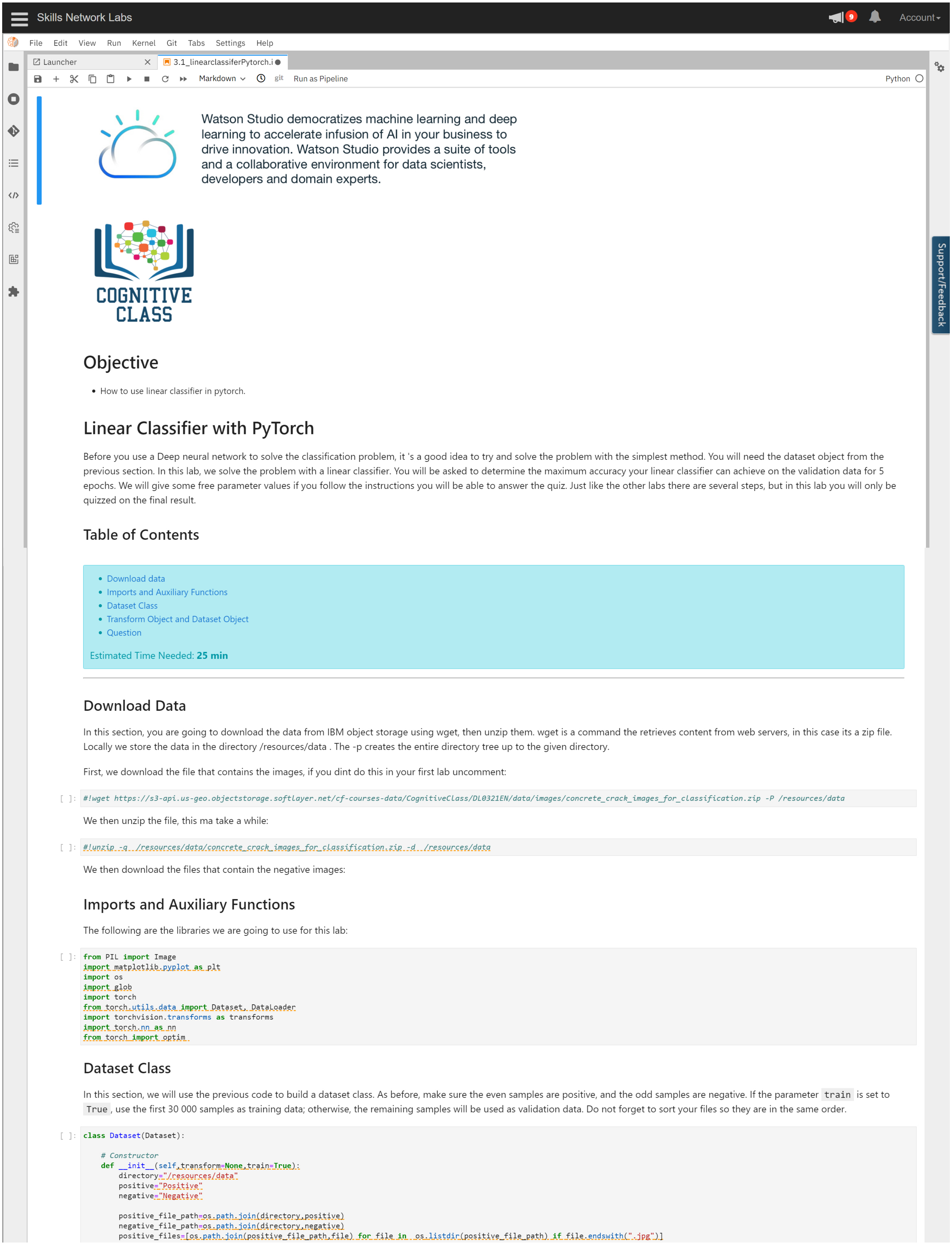


Watson Studio democratizes machine learning and deep learning to accelerate infusion of AI in your business to drive innovation. Watson Studio provides a suite of tools and a collaborative environment for data scientists, developers and domain experts.

Objective

- How to use linear classifier in pytorch.

Linear Classifier with PyTorch

Before you use a Deep neural network to solve the classification problem, it 's a good idea to try and solve the problem with the simplest method. You will need the dataset object from the previous section. In this lab, we solve the problem with a linear classifier. You will be asked to determine the maximum accuracy your linear classifier can achieve on the validation data for 5 epochs. We will give some free parameter values if you follow the instructions you will be able to answer the quiz. Just like the other labs there are several steps, but in this lab you will only be quizzed on the final result.

Table of Contents

Estimated Time Needed: **25 min**

Download Data

In this section, you are going to download the data from IBM object storage using wget, then unzip them. wget is a command the retrieves content from web servers, in this case its a zip file. Locally we store the data in the directory /resources/data . The -p creates the entire directory tree up to the given directory.

First, we download the file that contains the images, if you dint do this in your first lab uncomment:

```
[ ]: #!wget https://s3-api.us-geo.objectstorage.softlayer.net/cf-courses-data/CognitiveClass/DL0321EN/data/images/concrete_crack_images_for_classification.zip -P /resources/data
```

We then unzip the file, this ma take a while:

```
[ ]: #!unzip -q  /resources/data/concrete_crack_images_for_classification.zip -d  /resources/data
```

We then download the files that contain the negative images:

Imports and Auxiliary Functions

The following are the libraries we are going to use for this lab:

```
[ ]: from PIL import Image
     import matplotlib.pyplot as plt
     import os
     import glob
     import torch
     from torch.utils.data import Dataset, DataLoader
     import torchvision.transforms as transforms
     import torch.nn as nn
     from torch import optim
```

Dataset Class

In this section, we will use the previous code to build a dataset class. As before, make sure the even samples are positive, and the odd samples are negative. If the parameter `train` is set to `True`, use the first 30 000 samples as training data; otherwise, the remaining samples will be used as validation data. Do not forget to sort your files so they are in the same order.

```
[ ]: class Dataset(Dataset):

         # Constructor
         def __init__(self,transform=None,train=True):
             directory="/resources/data"
             positive="Positive"
             negative="Negative"

             positive_file_path=os.path.join(directory,positive)
             negative_file_path=os.path.join(directory,negative)
             positive_files=[os.path.join(positive_file_path,file) for file in  os.listdir(positive_file_path) if file.endswith(".jpg")]
```

```
        positive_files.sort()
        negative_files=[os.path.join(negative_file_path,file) for file in  os.listdir(negative_file_path) if file.endswith(".jpg")]
        negative_files.sort()
        number_of_samples=len(positive_files)+len(negative_files)
        self.all_files=[None]*number_of_samples
        self.all_files[::2]=positive_files
        self.all_files[1::2]=negative_files 
        # The transform is goint to be used on image
        self.transform = transform
        #torch.LongTensor
        self.Y=torch.zeros([number_of_samples]).type(torch.LongTensor)
        self.Y[::2]=1
        self.Y[1::2]=0

        if train:
            self.all_files=self.all_files[0:30000]
            self.Y=self.Y[0:30000]
            self.len=len(self.all_files)
        else:
            self.all_files=self.all_files[30000:]
            self.Y=self.Y[30000:]
            self.len=len(self.all_files)    

    # Get the length
    def __len__(self):
        return self.len

    # Getter
    def __getitem__(self, idx):


        image=Image.open(self.all_files[idx])
        y=self.Y[idx]


        # If there is any transform method, apply it onto the image
        if self.transform:
            image = self.transform(image)

        return image, y
```

Transform Object and Dataset Object

Create a transform object, that uses the `Compose` function. First use the transform `ToTensor()` and followed by `Normalize(mean, std)`. The value for `mean` and `std` are provided for you.

[ ]:

```
mean = [0.485, 0.456, 0.406]
std = [0.229, 0.224, 0.225]
# transforms.ToTensor()
#transforms.Normalize(mean, std)
#transforms.Compose([])

transform =transforms.Compose([ transforms.ToTensor(), transforms.Normalize(mean, std)])
```

Create object for the training data `dataset_train` and validation `dataset_val`. Use the transform object to convert the images to tensors using the transform object:

[ ]:

```
dataset_train=Dataset(transform=transform,train=True)
dataset_val=Dataset(transform=transform,train=False)
```

We can find the shape of the image:

[ ]:

```
dataset_train[0][0].shape
```

We see that it's a color image with three channels:

[ ]:

```
size_of_image=3*227*227
size_of_image
```

Question

Create a custom module for Softmax for two classes,called model. The input size should be the `size_of_image`, you should record the maximum accuracy achieved on the validation data for the different epochs. For example if the 5 epochs the accuracy was 0.5, 0.2, 0.64,0.77, 0.66 you would select 0.77.

Train the model with the following free parameter values:

Parameter Values

- learning rate:0.1
- momentum term:0.1
- batch size training:1000
- Loss function:Cross Entropy Loss
- epochs:5
- set: torch.manual_seed(0)

[ ]:

```
torch.manual_seed(0)
```

Custom Module:

[ ]:

Model Object:

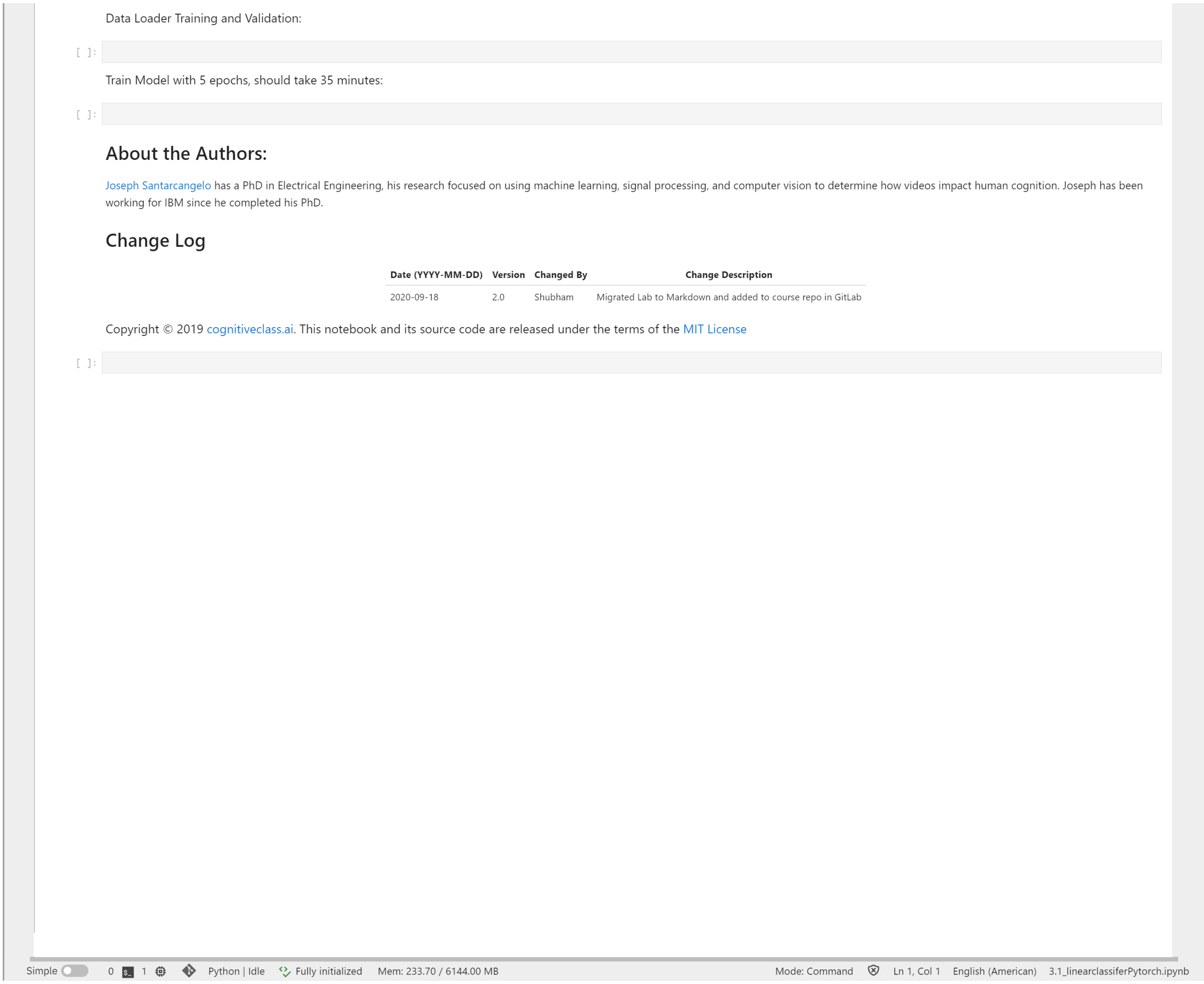
[ ]:

Optimizer:

[ ]:

Criterion:

[ ]:



Data Loader Training and Validation:

```
[ ]:
```

Train Model with 5 epochs, should take 35 minutes:

```
[ ]:
```

About the Authors:

Joseph Santarcangelo has a PhD in Electrical Engineering, his research focused on using machine learning, signal processing, and computer vision to determine how videos impact human cognition. Joseph has been working for IBM since he completed his PhD.

Change Log

Date (YYYY-MM-DD)	Version	Changed By	Change Description
2020-09-18	2.0	Shubham	Migrated Lab to Markdown and added to course repo in GitLab

```
[ ]:
```