National University of Computer and Emerging Sciences



Lab Manual 08

CL461-Artificial Intelligence Lab

|  |  |
| --- | --- |
| Course Instructor | Dr. Mubasher Baig |
| Lab Instructor (s) | Saad Ali  Mahmood Hussain |
| Section | A |
| Semester | Spring 2021 |

Department of Computer Science

FAST-NU, Lahore, Pakistan

Table of Contents

[Objectives](#_heading=h.30j0zll) **2**

[Task Distribution](#_heading=h.1fob9te) **2**

[**3. MNIST Handwritten Digits**](#_heading=h.rbmnkm7h0pyn) **3**

[3.1 Loading Dataset](#_heading=h.q2rx8o10h3f9) 3

[3.2 Features and Labels](#_heading=h.z71yz3tzok7d) 3

[3.3 Image Visualization](#_heading=h.kl1ox2aqhhov) 4

[3.4 Train-Test Splits](#_heading=h.xe9u4y24ye2z) 5

[3.5 Binary Image Classifier](#_heading=h.19auj0xbg2jn) 5

[3.5.1 Loading the Model](#_heading=h.e5nhpu3j1gja) 6

[3.5.2 Applying the model](#_heading=h.h423fzrjj4f1) 6

[3.5.3 Making Predictions](#_heading=h.5h35x4m6vdp4) **6**

[**4. Exercise**](#_heading=h.2pm68yx8nyl5) **6**

[4.1 Make a Multiple Digits Classifier (15)](#_heading=h.qc0mxkejay73) 6

[4.2 Deploy Your Model (10)](#_heading=h.idf7sndbsnzk) 7

[**5. Submission Instructions:**](#_heading=h.s1twrzd98wq) **7**

# Objectives

After performing this lab, students shall be able to understand binary and multiple image classifiers using MNIST handwritten digits dataset.

# Task Distribution

|  |  |
| --- | --- |
| **Total Time** | **170 Minutes** |
| Working with MNIST | 30 Minutes |
| Apply Machine Learning | 20 Minutes |
| Model Deployment | 20 Minutes |
| Exercise | 90 Minutes |
| Online Submission | 10 Minutes |
|  |  |

# 3. MNIST Handwritten Digits

The MNIST database is a large [database](https://en.wikipedia.org/wiki/Database) of handwritten digits that is commonly used for [training](https://en.wikipedia.org/wiki/Training_set) various image processing systems. The database is also widely used for training and testing in the field of machine learning. It was created by "re-mixing" the samples from NIST's original datasets.

The Dataset has 70000 handwritten digits images each of 28x28 pixels with labels on it. The dataset also contains the labels. From 70000 images, 60000 of them are used for training and 10000 are used for testing purposes.

An extended dataset similar to MNIST called EMNIST has been published in 2017, which contains 240,000 training images, and 40,000 testing images of handwritten digits and characters.

The dataset is used for supervised machine learning, deep learning and computer vision applications. Due to the popularity of the dataset, Python frameworks like Sklearn, Tensorflow, keras and Pytorch come with an installed MNIST dataset. We don't need to separately install or configure them.

The following section is dedicated towards importing and analyzing MNIST in Python Scikit-Learn framework.

## 3.1 Loading Dataset

To load the dataset, one has to import the Scikit-Learn library.

#import the dataset

from sklearn.datasets import fetch\_openml

#make a dataset object

mnist = fetch\_openml('mnist\_784')

## 3.2 Features and Labels

The dataset contains images in metrics form while labels in text form. These objects are stored in arrays and each array has a name. These names are called attributes. To access the images, one can use the ‘data’ attribute while to access labels one can incorporate the ‘target’ attribute in the object.

To access the images,

mnist['data']

To access the labels of each image, ‘target’ attribute can be used.

mnist['target']

At this stage, both features and labels are collected separately. These arrays will behave as an input output to our machine learning model .

x,y=mnist['data'],mnist['target']

## 3.3 Image Visualization

The images are in matrix format. These are certain libraries which can visualize these metrics in image format. Matplotlib is one of them.

In this section, we are going to display the array in image format.

First of all, we are required to import the libraries.

import matplotlib.pyplot as plt

import matplotlib

We can use indexing to select these images individually. By default, image arrays are single dimensional lists each containing 784 values. To realize them, matplotlib required a two-dimensional image. So we reshape the image into 28x28.

# Visualizing a single image

some\_digit=x[36001]

# Reshape the Image to Plot

some\_digit\_re=some\_digit.reshape(28,28)

Now show the image, Image is stored in RGB format.

plt.imshow(some\_digit\_re)

To display the image in binary format. one can use the cmap attribute.

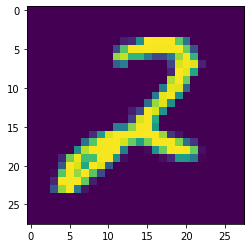
# In binary format

plt.imshow(some\_digit\_re,cmap=matplotlib.cm.binary)

#Turn off graph axis

plt.axis('off')

The output of RGB formatted image is:



## 3.4 Train-Test Splits

Before applying machine learning, data is required to split into training and testing modules. For this, we selected the first 60000 images as training sets and remaining 10000 as testing sets.

# train-test split

x\_train,x\_test=x[:60000], x[60000:]

y\_train, y\_test = y[:60000], y[60000:]

To avoid overfitting and underfitting issues, we shuffled the data using *numpy* library.

#shuffling the data

shuffle\_index=np.random.permutation(60000)

x\_train,y\_train= x\_train[shuffle\_index] , y\_train[shuffle\_index]

## 3.5 Building Binary Image Classifier

At this stage, we are only interested in binary classification. We selected only labels which contain only 2’s for both training and testing sets. The problem is to build a model which recognizes whether the given picture is in binary format or not.

#convert values from string to integer

y\_train=y\_train.astype(np.int8)

y\_test=y\_test.astype(np.int8)

# choose only labels containing 2's

y\_train\_2 = (y\_train==2)

y\_test\_2 = (y\_test==2)

### 3.5.1 Loading the Model

First step is to load the model, here we are choosing only the KNN classifier.

from sklearn.neighbors import KNeighborsClassifier

### 3.5.2 Applying the model

clf = KNeighborsClassifier()

Now apply the model, over training features and labels.

clf.fit(x\_train,y\_train\_2)

### 3.5.3 Making Predictions

To generate predictions from a model, one can use the predict function.

clf.predict([some\_digit])

# 4. Exercise

## 4.1 Make a Multiple Digits Classifier (15)

Use the same MNIST dataset to make a multiclass digits classifier, a classifier that can recognize any digit given by user (0-9). One can use any classification machine learning model we covered during previous labs.

## 4.2 Deploy Your Model (10)

You are required to deploy your model using Flask. (The UI should take an handwritten digit image from user, and should predict the digit )

# 5. Submission Instructions:

1. A data file is attached. For Practice Exercise, One has to use this file.
2. For Examples given in manual, no dataset is required because the dataset used is available in sklearn library.
3. To make the submission, Create a Jupyter Notebook File (lab8\_rollno.ipynb), create a .zip file along with the data file and submit on the Portal.