CS412 - Machine Learning - 2020

Homework 1

100 pts

Goal

The goal of this homework is three-fold:

- Introduction to the machine learning experimental set up
- Gain experience with Decision tree approache
- Gain experience with the Scikit library

Dataset

MNIST is a collection of 28x28 grayscale images of digits (0-9); hence each pixel is a gray-level from 0-255.

Download the data from Keras. You must use a 20% of the training data for validation (no need for cross-validation as you have plenty of data) and **use the official test data (10,000 samples) only for testing.**

Task

Build a decision tree classifier with the scikit library function calls to classify digits in the MNIST dataset.

Software: You may find the necessary function references here:

http://scikit-learn.org/stable/supervised_learning.html

Submission:

Fill this notebook and submit this document with a link to #your Colab notebook (make sure to include the link obtained from the #share link on top right)

1) Initialize

- First make a copy of the notebook given to you as a starter.
- Make sure you choose Connect form upper right.

2) Load training dataset

- Read from Keras library.

```
# Read from Keras library
import keras
from keras.datasets import mnist

(x_train, y_train), (x_test, y_test) = mnist.load_data()


# Preprocessing: Reshape Testing and Training Data 2D to 1D
import numpy as np

x_train_size= x_train.shape
x_test_size= x_test.shape

x_train=np.reshape(x_train,(x_train_size[0], x_train_size[1]*x_train_size[2])).astype('flo
x_test=np.reshape(x_test,(x_test_size[0], x_test_size[1]*x_test_size[2])).astype('float32'

# Normalize
from sklearn.preprocessing import normalize
x_train = normalize(x_train)
x_test = normalize(x_test)

print('Train and Test Data shapes after Reshaping')
print('Train Data shape:' , x_train.shape)
print('Test Data shape:' , x_test.shape)

# Load the Pandas libraries with alias 'pd'
import pandas as pd


# Read data
train_dataframe = pd.DataFrame(x_train)
train_dataframe['label'] = y_train

labels = train_dataframe['label']
```

```
     Train and Test Data shapes after Reshaping
     Train Data shape: (60000, 784)
     Test Data shape: (10000, 784)
```

3) Understanding the dataset

There are alot of functions that can be used to know more about this dataset

- What is the shape of the training set (num of samples X number of attributes) ***[shape function can be used]***
- Display attribute names ***[columns function can be used]***
- Display the first 5 rows from training dataset ***[head or sample functions can be used]***

Note: Understanding the features, possibly removing some features etc. is an important part in building an ML system, but for this homework this is not really necessary as the features are homogeneous (pixels) and all necessary.

```
# print shape
print('Data Dimensionality: ')
print(train_dataframe.shape)

# Display attribute names
print('---------------------------')
print('Attribute Names: ')
print(train_dataframe.columns)

# print first 5 rows in your dataset
print('---------------------------')
print('Head of Data: ')
print(train_dataframe.head())
```

```
Data Dimensionality:
(60000, 785)
---------------------------
Attribute Names:
Index([      0,       1,       2,       3,       4,       5,       6,       7,
             8,       9,
       ...
           775,     776,     777,     778,     779,     780,     781,     782,
           783, 'label'],
      dtype='object', length=785)
---------------------------
Head of Data:
     0    1    2    3    4    5    6  ...  778  779  780  781  782  783  label
0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0      5
1  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0      0
2  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0      4
3  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0      1
4  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0      9

[5 rows x 785 columns]
```

4) Shuffle and Split TRAINING data as train (also called development) (80%) and validation (20%)

```
from sklearn.utils import shuffle

# Shuffle the training data

data = shuffle(train_dataframe)
```

```
from sklearn.model_selection import train_test_split

# Split 80-20
```

```
X_train, X_valid, Y_train, Y_valid = train_test_split(data, labels, test_size=0.2, random_

print("Train data shape:", X_train.shape, "Train label shape:", Y_train.shape)
print("Validation data shape:", X_valid.shape, "Validation label shape:", Y_valid.shape)
```

```
Train data shape: (48000, 785) Train label shape: (48000,)
Validation data shape: (12000, 785) Validation label shape: (12000,)
```

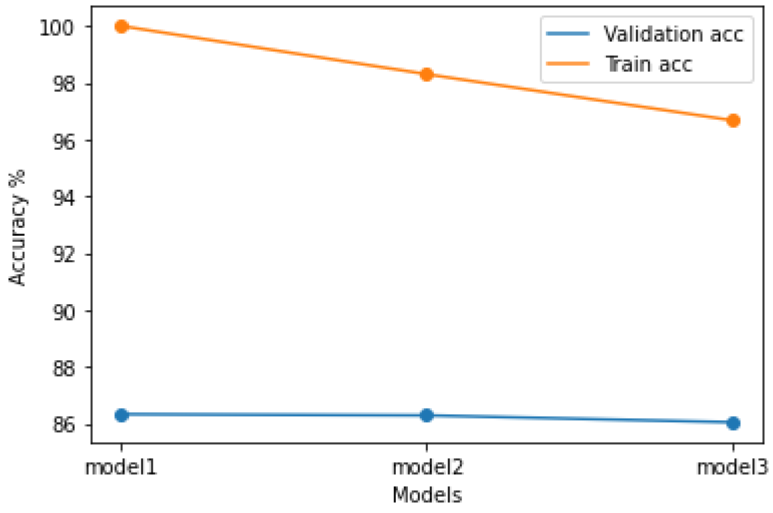
5) Train a decision tree classifier on development/train data and do model selection using the validation data

- Train 3 decision tree classifiers with different values of "min_samples_split" which is the minimum number of samples required to split an internal node: min_samples_split = [default = 2, 5, 10].
- Test the 3 models on validation set and choose the best one.
- Plot the train and validation set errors for those 3 settings - on one plot.

```
# We need to split training data to "attributes" and "label"
X_train_attributes = X_train.drop('label', axis=1)
X_train_label = X_train['label']

X_valid_attributes = X_valid.drop('label', axis=1)
X_valid_label = X_valid['label']


# Train decision tree classifiers
from sklearn.tree import DecisionTreeClassifier

dtree1 = DecisionTreeClassifier(min_samples_split = 2)
dtree2 = DecisionTreeClassifier(min_samples_split = 5)
dtree3 = DecisionTreeClassifier(min_samples_split = 10)

dtree1.fit(X_train_attributes, X_train_label)
dtree2.fit(X_train_attributes, X_train_label)
dtree3.fit(X_train_attributes, X_train_label)

predict_min_2_valid = dtree1.predict(X_valid_attributes)
predict_min_2_train = dtree1.predict(X_train_attributes)

predict_min_5_valid = dtree2.predict(X_valid_attributes)
predict_min_5_train = dtree2.predict(X_train_attributes)

predict_min_10_valid = dtree3.predict(X_valid_attributes)
predict_min_10_train = dtree3.predict(X_train_attributes)

dtree_arr = [dtree1, dtree2, dtree3]
```

```
# Evaluate on validation set


from sklearn.metrics import accuracy_score



valid_score_min_2 = (accuracy_score(X_valid_label, predict_min_2_valid) * 100)
print("For min_samples_split = 2, validation set accuracy = %.5f%%" % valid_score_min_2)
valid_score_min_5 = (accuracy_score(X_valid_label, predict_min_5_valid) * 100)
print("For min_samples_split = 5, validation set accuracy = %.5f%%" % valid_score_min_5)
valid_score_min_10 = (accuracy_score(X_valid_label, predict_min_10_valid) * 100)
print("For min_samples_split = 10, validation set accuracy= %.5f%%" % valid_score_min_10)
print("--------------------------------------------------------")
train_score_min_2 = (accuracy_score(X_train_label, predict_min_2_train) * 100)
print("For min_samples_split = 2, train set accuracy= %.5f%%" % train_score_min_2)
train_score_min_5 = (accuracy_score(X_train_label, predict_min_5_train) * 100)
print("For min_samples_split = 5, train set accuracy= %.5f%%" % train_score_min_5)
train_score_min_10 = (accuracy_score(X_train_label, predict_min_10_train) * 100)
print("For min_samples_split = 10, train set accuracy= %.5f%%" % train_score_min_10)
print("---------------------------------------------------------")



# Plot errors
import matplotlib.pyplot as plt

val_acc = [valid_score_min_2, valid_score_min_5, valid_score_min_10]
train_acc = [train_score_min_2, train_score_min_5, train_score_min_10]

print("Max validation set accuracy = %.5f%%" % (val_acc[np.argmax(val_acc)] ))
print("Chosen Decision Tree Number (1st,2nd or 3rd):", np.argmax(val_acc)+1)
print("--------------------------------------------------------")
x_axis = ['model1', 'model2', 'model3']
plt.scatter(x_axis, val_acc)
plt.scatter(x_axis, train_acc)
plt.plot(x_axis, val_acc, label = "Validation acc")
plt.plot(x_axis, train_acc, label = "Train acc")
plt.xlabel('Models')
plt.ylabel('Accuracy %')
plt.legend()
plt.show()
```

```
For min_samples_split = 2, validation set accuracy = 86.32500%
For min_samples_split = 5, validation set accuracy = 86.28333%
For min_samples_split = 10, validation set accuracy= 86.04167%
--------------------------------------------------------
For min_samples_split = 2, train set accuracy= 100.00000%
For min_samples_split = 5, train set accuracy= 98.30000%
For min_samples_split = 10, train set accuracy= 96.67500%
---------------------------------------------------------
Max validation set accuracy = 86.32500%
Chosen Decision Tree Number (1st,2nd or 3rd): 1
--------------------------------------------------------
```



7) Test your CHOSEN classifier on Test set

- Load test data
- Apply same pre-processing as training data (probably none)
- Predict the labels of testing data **using the best chosen SINGLE model out of the models that you have tried from step 6 (you have selected your model according to your validation results)** and report the accuracy.

```
# Load test data

test_dataframe = pd.DataFrame(x_test)
test_dataframe['label'] = y_test
```

```
# test prediction using a decision tree with all default parameters and ..... min-split va

X_test_attributes = test_dataframe.drop('label', axis=1)
X_test_label = test_dataframe['label']

min_samples_split_arr = [2,5,10]

print('I have obtained the best results:')
print('with the Decision Tree Classifier number= %d' % (np.argmax(val_acc)+1))
print('with parameters such that min_samples_split = %d' % (min_samples_split_arr[np.argma
predict_test = dtree_arr[np.argmax(val_acc)].predict(X_test_attributes)
```

```
# Report your accuracy

score_test = accuracy_score(X_test_label, predict_test)
print("and it gives classification accuracy = %.5f%% on the test data" % (score_test*100))

# Confusion Matrix

from sklearn.metrics import confusion_matrix

confusion = confusion_matrix(X_test_label, predict_test)
print('----------------------------------------------------')
print('Confusion Matrix: ')
confusion
```

```
I have obtained the best results:
with the Decision Tree Classifier number= 1
with parameters such that min_samples_split = 2
and it gives classification accuracy = 86.88000% on the test data
----------------------------------------------------
Confusion Matrix:
array([[ 915,    1,   13,    4,    9,   12,    6,    8,    4,    8],
       [   0, 1076,    5,    9,    1,    7,    7,    3,   24,    3],
       [  14,    6,  862,   41,   19,   14,   16,   26,   28,    6],
       [  15,    7,   30,  843,    6,   42,    4,   13,   32,   18],
       [   2,    2,   18,    5,  861,   15,   15,    6,   12,   46],
       [  12,    8,   15,   46,   14,  727,   19,    7,   22,   22],
       [  19,    4,   14,    4,   16,   25,  860,    1,   10,    5],
       [   2,    6,   23,   18,   12,    5,    1,  918,   14,   29],
       [  10,   12,   34,   48,   19,   30,   14,   12,  769,   26],
       [  11,    5,   13,   26,   38,   12,    0,   19,   28,  857]])
```

8) Notebook & Report

Notebook: We may just look at your notebook results; so make sure each cell is run and outputs are there.

Report: Write an at most 1/2 page summary of your approach to this problem at the end of your notebook; this should be like an abstract of a paper or the executive summary (you aim for clarity and passing on information, not going to details about known facts such as what dec. trees are or what MNIST is, assuming they are known to people in your research area).

Must include statements such as:

(Include the problem definition: 1-2 lines)

(Talk about train/val/test sets, size and how split.)

(Talk about any preprocessing you do.)

(Give the validation accuracies for different approaches, parameters **in a table** and state which one you selected)

(State what your test results are with the chosen method, parameters: e.g. "We have obtained the best results with the ….. classifier (parameters=....) , giving classification accuracy of …% on test data….""

(Comment on the speed of the algorithms and anything else that you deem important/interesting (e.g. confusion matrix)).

You will get full points from here as long as you have a good (enough) summary of your work, regardless of your best performance or what you have decided to talk about in the last few lines.

```
                    The Executive Summary
```

First of all, the problem is trying to train a model and implement MNIST dataset with python. In the end, the aim is to find the best Decision Tree classifier which gives the highest validation set accuracy when we try different min_samples_split parameters [2,5,10]. Then, I have predicted the testing data on the chosen Decision Tree so that I can obtain the possible highest accuracy on the test set.

I have started with reading data from Keras library. The image data is imported in 2 tuples such as 1 tuple for training images and 1 tuple for testing images. Then, I printed those tuple shapes to understand how much image we have (60,000 training images and 10,000 testing images) and how much our pixel values are (28x28). Then, I did preprocessing like reshaping of the dataset to use pandas DataFrame methods and because of the matrix multiplication (28x28=784) we must change the shape of the array such that x_train shape becomes (60000, 784) and x_test shape becomes (10000, 784). Later, by using pandas DataFrame methods such as head,columns etc I have analysed the data more detailly. For validation set, I shuffled training data before splitting into %80 train set, %20 validation set. It is shuffled to make the split random between two sets. After the split, the train set shape becomes (48000, 785) and validation set shape becomes (12000, 785).

For the training a decision tree classifier on development/train data part, first I need to split training data to "attributes" and "label". Then, I created 3 Decision Tree Classifiers with different parameters of min_samples_split such that = [2, 5, 10]. Later on, I trained those decision trees by giving train set attributes and labels. After the training is done, I predicted labels with both train and validation set attributes. By using predictions, I calculated accuracies and obtained 3 accuracy values for validation set and 3 accuracy values for the train set. However, because we select the model by looking to the validation set, I choose the Decision Tree with the highest accuracy among the validation accuracies, not by looking train set accuracies.

By checking the validation accuracies for different approaches:

For min_samples_split = 2, validation set accuracy = 86.32500%
For min_samples_split = 5, validation set accuracy = 86.28333%
For min_samples_split = 10, validation set accuracy= 86.04167%

As a result, I have obtained the best results with the first Decision Tree classifier (the highest validation accuracy) and (parameters-> min_samples_split = 2). In the end, when I have predicted the labels of testing data using the best chosen model (Decision Tree Number= 1), it gives classification accuracy on test data = 86.88000%

To discuss about the speed of algorithms, while I was training 3 different decision tree classifiers which have different min_samples_split values [2,5,10],I realized that when min_samples_split increases, training can be done more quickly. And it happened because when min_samples_split value is larger, tree nodes are created less and decision tree grows less. So, it improves the speed of algorithm/training. As a final discussion, Confussion matrix describes the performance of a classification model and in MNIST dataset, we can see accuracies of all digits from 0 to 9 as class by class. And, I printed confusion matrix to understand whether our model knew each class equally, or if it guessed some numbers well and/or failed others. For example, one of the highest confusion which our model did is that it predicted '3' instead of '8' for 48 times etc.
Confusion Matrix:
array([[915, 1, 13, 4, 9, 12, 6, 8, 4, 8], [0, 1076, 5, 9, 1, 7, 7, 3, 24, 3], [14, 6, 862, 41, 19, 14, 16, 26, 28, 6], [15, 7, 30, 843, 6, 42, 4, 13, 32, 18], [2, 2, 18, 5, 861, 15, 15, 6, 12, 46], [12, 8, 15, 46, 14, 727, 19, 7, 22, 22], [19, 4, 14, 4, 16, 25, 860, 1, 10, 5], [2, 6, 23, 18, 12, 5, 1, 918, 14, 29], [10, 12, 34, 48, 19, 30, 14, 12, 769, 26], [11, 5, 13, 26, 38, 12, 0, 19, 28, 857]])

9) Submission

Please submit your **"share link" INLINE in Sucourse submissions**. That is we should be able to click on the link and go there and run (and possibly also modify) your code.

For us to be able to modify, in case of errors etc, **you should get your "share link" as **share with anyone in edit mode**

Also submit your notebook as pdf as attachment, choose print and save as PDF, save with hw1-lastname-firstname.pdf to facilitate grading.

Questions?

You can and should ask all your Google Colab related questions under Forums and feel free to answer/share your answer regarding Colab.

You can also ask/answer about which functions to use and what libraries...

However you should **not ask** about the core parts, that is what is validation/test, which one shd. have higher performance, what are your scores etc.