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.

```
# Load the Pandas libraries with alias 'pd'
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
# Read data
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data(path="mnist.npz"

#Since I know the shape data, I could just write (60000, 28*28) but this is more general:
x_train = np.reshape(x_train, (x_train.shape[0], x_train.shape[1]*x_train.shape[2]))

train_dataframe = pd.DataFrame(x_train)
train_dataframe['label'] = y_train
```

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(x_train.shape)

print("\nColumn names: ")
print(train_dataframe.columns)

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

```
Data Dimensionality:
(60000, 784)

Column 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	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	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	
1	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	

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

```
from sklearn.utils import shuffle
from sklearn.model_selection import train_test_split
# Shuffle the training data

x_train, y_train = shuffle(x_train, y_train, random_state= 42)

# Split 80-20

print(x_train.shape)
x_train, x_validation, y_train, y_validation = train_test_split(x_train, y_train, test_siz


print("Train data shape:", x_train.shape, "Validation data shape:", x_validation.shape)
print("Train label shape:", y_train.shape, "Validation label shape:", y_validation.shape)
```

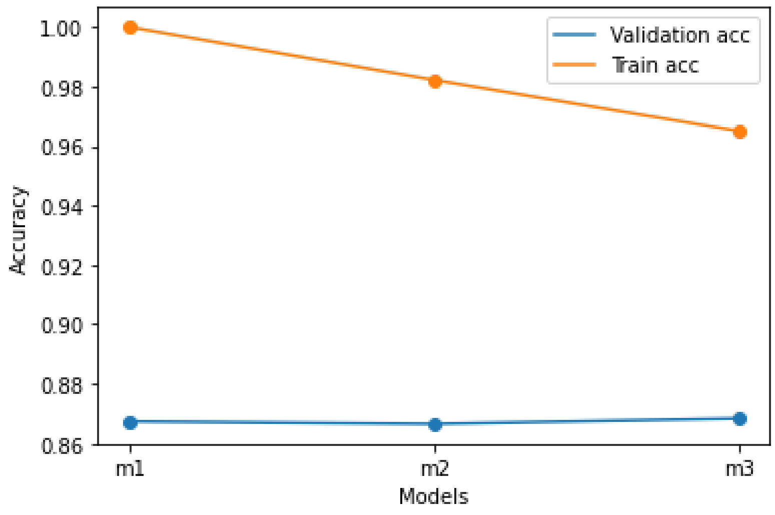
```
(60000, 784)
Train data shape: (48000, 784) Validation data shape: (12000, 784)
Train label shape: (48000,) Validation label shape: (12000,)
```

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.

```
from sklearn import tree
from sklearn.metrics import accuracy_score
from sklearn.metrics import mean_absolute_error, mean_squared_error
from math import sqrt
# Train decision tree classifiers


clf1 = tree.DecisionTreeClassifier(min_samples_split= 2)
clf2 = tree.DecisionTreeClassifier(min_samples_split= 5)
clf3 = tree.DecisionTreeClassifier(min_samples_split= 10)


clf1 = clf1.fit(x_train, y_train)
clf2 = clf2.fit(x_train, y_train)
clf3 = clf3.fit(x_train, y_train)


# Evaluate on validation set


##print(clf1.score(x_validation, y_validation))
##print(clf2.score(x_validation, y_validation))
##print(clf3.score(x_validation, y_validation))
##print(clf4.score(x_validation, y_validation))


y_pred1_train = clf1.predict(x_train)
train1_acc = accuracy_score(y_pred1_train, y_train)

## Evaluate model1 on validation set
y_pred1 = clf1.predict(x_validation)
val1 = accuracy_score(y_pred1, y_validation)
##

y_pred2_train = clf2.predict(x_train)
train2_acc = accuracy_score(y_pred2_train, y_train)

## Evaluate model2 on validation set
y_pred2 = clf2.predict(x_validation)
val2 = accuracy_score(y_pred2, y_validation)
##

y_pred3_train = clf3.predict(x_train)
train3_acc = accuracy_score(y_pred3_train, y_train)

## Evaluate model3 on validation set
y_pred3 = clf3.predict(x_validation)
val3 = accuracy_score(y_pred3, y_validation)
##

print('train1 accuracy: ' , train1_acc)
print('train2 accuracy: ' , train2_acc)
print('train3 accuracy: ' , train3_acc)
print('Validation1 accuracy: ' , val1)
print('Validation2 accuracy: ' , val2)
print('Validation3 accuracy: ' , val3)


# Plot errors

x_axis = ['m1', 'm2', 'm3']
val_acc = [val1, val2, val3]
train_acc = [train1_acc, train2_acc, train3_acc]

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()
```

```
train1 accuracy:  1.0
train2 accuracy:  0.9821666666666666
train3 accuracy:  0.9650208333333333
Validation1 accuracy:  0.8674166666666666
Validation2 accuracy:  0.8665833333333334
Validation3 accuracy:  0.8684166666666666
```



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.

```
from math import sqrt
from sklearn.metrics import mean_absolute_error, mean_squared_error
# Load test data
```

```
x_test = np.reshape(x_test, (x_test.shape[0], x_test.shape[1]*x_test.shape[2]))
print("x_test shape: ", x_test.shape)
print("y_test shape: ", y_test.shape)

# test prediction using a decision tree with all default parameters and 10 min-split value
y_pred = clf3.predict(x_test)

accuracy = accuracy_score(y_pred, y_test)
#print(clf1.score(x_test, y_test))


# Report your accuracy

print("Accuracy: ", accuracy)
```

```
    x_test shape:  (10000, 784)
    y_test shape:  (10000,)
    Accuracy:  0.8663
```

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.

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.

REPORT

Our task is classifying digits in MNIST dataset which is loaded from Keras by using decision tree classifiers in this assignment. **In the first part**, we loaded out data from Keras and our training data size is 60000 and test data size is 10000. When data is loaded from Keras, it came in the form of numpy arrays (contains 2 tuples). Thus, we reshaped the training data into dataFrame to be able to use pandas module. **In the second part**, we used .head() and .columns() functions on the dataFrame to have a visual understanding on our data. We did not do any additional pre-processing other than reshaping. I could not play with feautures because all of them was important since they are pixels. **In the third part**, we shuffled the training data and labels by using shuffle function of sklearn.utils. Then, we splitted our training data (60000) into 80 - 20 by using train_test_split funtion. At the end, 80% is our training set and %20 is our validation set. After all processes, I printed the exact shape of training and validation set. We also did the same operations for labels. **In the fourth part**, we trained three different models with min-samples 2, 5 and 10. After training we evaluated the each models on validation set to see their accuracies. Below we can see the resulted accuracies in a table:

Accuracy table for different min_samples_split

min_samples_split	Accuracy
2	0.8674166666666666
5	0.8665833333333334
10	**0.8684166666666666**

Since the highest validation accuracy is the third model whose min_samples_split = 10, I chose the third model to test it. **In the fifth part**, I found my third model's accuracy on the test data. Below we can see the resulted accuracy in a table:

Table

min_samples_split = 10	Accuracy
Model 3 = clf3	0.8663

When I plot the accuracy error table in the fourth part, I realized that when we increase the min_samples_split, it prevents overfitting by adjusting decision boundary. Also, I think that when we increase the min_samples_split, algorithm becomes faster.