Open in Colab

ENGS 108 Fall 2020 Assignment 4

Due October 21, 2020 at 11:59PM on Canvas

Instructors: George Cybenko

TAs: Chase Yakaboski

Rules and Requirements

1. You are only allowed to use Python packages that are explicity imported in the assignment notebook or are standard (bultin) python libraries like random, os, sys, etc, (Standard Bultin Python libraries will have a Python.org documentation). For this assignment you may use:
    - numpy
    - pandas
    - scikit-learn
    - matplotlib
    - tensorflow
2. All code must be fit into the designated code or text blocks in the assignment notebook. They are indentified by a **TODO** qualifier.
3. For analytical questions that don't require code, type your answer cleanly in Markdown. For help, see the Google Colab Markdown Guide.

In [ ]:
```
''' Import Statements '''
import numpy as np
import pandas as pd
import tensorflow as tf
import matplotlib.pyplot as plt
import os
import copy
import pickle
import tqdm
```

Data Loading

In [ ]:
```
dataset_base_path = '/content/sample_data'

#TODO: Set your base datasets path. This is my base path, you will need to change to match yours.
dataset_github_path = '/content/drive/My Drive/git/ENGS_108_Fall_2020/datasets'
```

In [ ]:
```
#-- Load circles dataset, format is X, y where X is a 2 dimensional coordinate and y is the label.
with open(os.path.join(dataset_github_path, 'circles.pk'), 'rb') as f_:
  circles = pickle.load(f_)
```

Problem 1: Support Vector Machines

In this problem, you will be building a support vector machines to for both regression and classification tasks. >

> **Part 1** In this part we will be exploring the *circles* dataset. In this dataset you will have an X array of 2 dimensional samples of the form (x_1, x_2) and a y array of each samples associated label.
>
> > **(a)** Go through the circles dataset and create a scatterplot of the circles data using the y label of each samples color to designate their respective class.

In [ ]:
```
X, y = circles
#TODO: Your code goes here.
```

> > **(b)** Is this dataset linearly seperable? Explain why or why not?

TODO: Your answer should go here.

> > **(c)** Can you think of a transformation of the dataset that could make the dataset linearly seperable? If so, define what these transformation function(s) might look like, and if not explain why. *Hint: Think of a higher dimensional space.*

TODO: Your answer should go here.

> > **(d)** If you where able to find a transformation in (c), create a suitable graph showing the dataset is linearly seperable in this new feature space.

In [ ]:
```
#TODO: Your code goes here.
```

> **Part 2** What we accomplished in Part 1 is known as the kernel trick for SVMs. Now let's focus on how we can use this idea to accomplish non-linear classification on a real world dataset. In this next part and throughout the remainder of the assignment we will be using a food image dataset. These images are RGB images of many pixels.
>
> > **(a)** You have been given a number of code skeletons throughout the course all of which load and preprocess the data for you. In this excerise tho, we will be doing the data loading manually as it is an important skill to learn. Write some code that will walk through the *ExampleFoodImageDataset* directory structure and build a single large numpy array with all image features flattened into a large vector (Make sure to resize the image to something like (28, 28) or (32, 32), etc.) the first column being a integer id for the class. *Hint: You have been provided with a basic skeleton, study the operations of the code and finish the script.*

In [ ]:
```
from PIL import Image

#-- The dimensions of the resized image
RESIZE = (28,28)
#-- A map from integer ids to food categories (strings)
food_map = {}
#-- The data list that we will be filling in.
data = []
#-- The folder that the food images are in
folder = os.path.join(dataset_github_path, 'ExampleFoodImageDataset')

#-- Let's start our for loop (Just using tqdm to give us a pretty progress bar).
for idx, subfold in enumerate(tqdm.tqdm(os.listdir(folder), desc='Processing images', leave=False)):
  if os.path.isdir(os.path.join(folder, subfold)):
    #-- We have found image class folder so let's extract all example data
    map_[idx] = subfold
    for img_name in os.listdir(os.path.join(folder, subfold)):
      #-- Make sure the file is an image
      if img_name.endswith('.jpg'):
        #TODO: You do this part. Use the Image class from PIL to load the image and cast it into a np array.
        #TODO: Then make sure to resize the image (otherwise things will take awhile)

data = np.array(data)
```

> > **(b)** Split your dataset into training and testing sets with an 80/20 split. *Hint: Look at Sklearn's train_test_split function.* Then implement a SVM classifer and report your accuracy on the testing dataset.

In [ ]:
```
#TODO: Your code goes here.
```

> > **(c)** Choose a 2 hyperparameters to study and experiment with. Can you make an SVM that has better accuracy then just using the defaults?

In [ ]:
```
#TODO: Your code goes here.
```

Problem 2: Introduction to TensorFlow

In this problem, we will start working in tensorflow to build deep learning systems starting with fully connected neural networks. We will focus on using the food image dataset we built in the last problem. >

> > **(a)** Using the food image dataset we built in the last problem, build a tensorflow Data Dataset that is shuffled with a batch size of 10. *Hint: We did this in class.*

In [ ]:
```
#TODO: Your code goes here.
```

> > **(b)** Build a two layer fully connected neural network of any size with a ReLu activation function and a final softmax layer.

In [ ]:
```
#TODO: Your code goes here.
```

> > **(c)** Compile your model with an appropriate loss function and optimizer. Briefly describe your choices.

In [ ]:
```
#TODO: Your code goes here.
```

> > **(d)** Train your model on the food image training dataset. And report your accuracy on the testing dataset.

In [ ]:
```
#TODO: Your code goes here.
```

> > **(e)** Now try to tune this network by varying the number of layers, units, activations and see if you can outperform the network in part (d). Does your best model perform better or worse than the SVM in problem 1?

In [ ]:
```
#TODO: Your code goes here.
```

> > **(BONUS)** We lost a lot of information when we resized the images in part (a). What would happen if we didn't resize the images and we built fit the neural network with all this other information? Try it out! *Hint: Runtime will be much longer, both to create the image dataset without resizing and to train the model, so you might have to get the code working and then just let it run.*

In [ ]:
```
#TODO: Your code goes here.
```