Advanced Programming Techniques
Assignment 4

Welcome to TensorFlow

You can get more familiar with the TensorFlow documentation that you can find at https://www.tensorflow.org/api_docs/python/ or by looking at the tutorials at https://www.tensorflow.org/tutorials .

Note, we will be using TensorFlow 2.0 in Python 3 for Assignment 4.

There are noticeable differences between 2.0 and 1.XX. If you see sessions you should take this as reading an example from 1.XX. If you install TensorFlow locally, 2.0 is the version that is automatically pulled. (ex. pip install tensorflow)

To install packages on the university machines you will have to use a virtual environment. This is pretty simple to setup and work in. There is a file called setup.txt with the assignment materials and tutorials will address it.

Instead of a local install you can also use the online workbook environment of Google Collab. https://colab.research.google.com/
This environment lets us use both Python 3 and TensorFlow 2.0 from any location we desire that has internet access. Tutorials will also show how to use and code in this environment.

You will be expected to submit your .py files (and .ipynb files if you use Google Collab) to the D2L dropbox if you use this for your coding.

If you use Google Collab make your first block of code the following. This ensures the version that collab will deliver as a package is TensorFlow 2.0.

```
try:
  # %tensorflow_version only exists in Colab.
  %tensorflow_version 2.x
except Exception:
  pass
```

If you want to explore TensorFlow 1.XX code you find elsewhere in examples you can make this version number a 1.0 in a different notebook for that purpose.

For the parts of this assignment starter files will be provided for both regular local coding and for Google Collab work. Certain material such as interactive GUI files to make images will only work in a local environment.

The goal of this assignment is to explore three increasingly harder problems. The first is to improve the logistic regression accuracy of the MNIST digit image model we demonstrated in class. The second is to replace the MNIST image data with a harder set of letter representation data and make a model for it. The third is to create and develop a model for data loaded from a CSV file about Coronary Heart Disease and associated risk factors.

Part 1: MNIST Logistic Regression

It is pretty easy to get accuracy of ~90% on the MNIST dataset with a vanilla model as done in class. This performance is generally considered unacceptable. the dataset is basically solved and state of the art models reach accuracies above 99%. Improve the performance. You can use whatever loss functions, optimizers, epochs, and even models that you want, as long as your model is built in TensorFlow 2.0 and keras.

You will be provided the code of a starter model MNISTStarter.py and MNISTStarter.ipynb.

In a report, explain what you decided to do, instruction on how to run your code, and report your hyper-parameters/results. Submit to the dropbox your code in MNISTModel.py (and MNISTModel.ipynb) files when you are done.

You should be able to at least reach 99%+ on training data and 98%+ on test data.

Part 2: Logistic regression on a replacement for MNIST dataset

Machine learning community is a bit sick of seeing MNIST pop up everywhere so they created a similar dataset and literally named it notMNIST. Created by Yaroslav Bulatov, a research engineer previously at Google and now at OpenAI, notMNIST is designed to look like the classic MNIST dataset, but less 'clean' and extremely 'cute'. The images are still 28x28 and there are also 10 labels, representing letters 'A' to 'J'.

I have done the work of reducing this dataset down into something that is in the same format as the MNIST dataset. This file is available as notMNIST.npz.

You will have to upload the file to Google Collab using

```
from google.colab import files
uploaded = files.upload()
```

The starter code notMNISTStarter.py and notMNISTStarter.ipynb has the new data loading included in place of the keras MNIST loading. This loading is done using the numpy library.

Build a model for this notMNIST data like you did for part 1 and the MNIST data. This will be harder given the challenge of the letter pictures being much more diverse.

Once you think you have developed a good model. Add a line of code to save your model to a file called notMNIST.h5. (you can do the same with your MNIST.h5 model) Using this model you can now use the following files:

predict_test.py (predict_test.ipynb)	-> Give index of test image and see prediction
grabimage.py	-> Local program to create input .png images
predict.py (predict.ipynb)	-> Program to take input .png image and predict
interactive.py	-> Local program to take user input and predict

In the predict files you will have to change three lines to get them to work. First a line to load your model. Second, a line to get an array of percent confidence in each class for your image. Third, a line to decide the index of the highest prediction in the previous array. Report what lines you added to get these programs to work in your report.

Use these programs to create an image that you would classify as one of the classes, but your model gets very wrong. Then attempt to make changes to your model that lead to this input getting identify accurately without more than a percentage loss of accuracy to your developed model. Report this image you made. What your model's accuracy was when this image was inaccurately predicted. What changes you attempted. Then what your model predicted for your model after the changes, as well as your model's accuracy on the test data.

In a report, explain what you decided to do, instruction on how to run your code, and report your hyper-parameters/results. Submit to the dropbox your code in a notMNISTModel.py (and notMNISTModel.ipynb) files.

Part 3: Build a logistic regression model to predict if someone has coronary heart disease

You will be given a heart.csv file of data in csv format.

You should divide this file into heart_train.csv and heart_test.csv data. Generally, test data is only a portion of the size of the training data.

For example, in the MNIST data sets the test data is 1/6 of the size of the training data or (1/7 of the total data). For example, 60,000 training examples and 10,000 test examples.

Note, that good test data for fuzzy world data generally doesn't directly repeat any examples found in the training data.

For the file, the first row is the name of the observed variables. There are 10 variables:

1. sbp: Systolic blood pressure
2. tobacco: Cumulative tobacco consumption, in kg
3. ldl: Low-density lipoprotein cholesterol
4. adiposity: Adipose tissue concentration
5. famhist: Family history of heart disease (1=Present, 0=Absent)
6. typea: Score on test designed to measure type-A behavior

7. obesity: Obesity
8. alcohol: Current consumption of alcohol
9. age: Age of subject
10. **chd: Coronary heart disease at baseline; 1=Yes 0=No**

Each following row contains the information of one patient.
There are 462 samples in total.

We will be using the first 9 variables to predict the last variable. That is, your input will be 1-d tensor of 9 elements, and your label is binary. You should write the function to read in data yourself, and you should take care of dividing your data into train set and test set.

In terms of loading and processing csv data you will find the tutorial at https://www.tensorflow.org/tutorials/load_data/csv very helpful. You have 8 numeric pieces of data and 1 category.

One problem you will notice when building this model is overfitting due to the small amount of data samples for training. Examine the differences in your initial model attempts between higher accuracies of training data vs much lower accuracies in test data. Then make changes to your model to deal with overfitting and report the impact of your changes before and after. https://www.tensorflow.org/tutorials/keras/overfit_and_underfit will be of good use for this part.

In a report, explain what you decided to do, instruction on how to run your code, and report your hyper-parameters/results. Submit to the dropbox your code in a CHDModel.py (and CHDModel.ipynb) files downloaded when you are done. Also submit your heart_train.csv and heart_test.csv data.

Source of the data: http://statweb.stanford.edu/~tibs/ElemStatLearn/

Submit the following to the D2L Assignment 4 Dropbox:

1. An electronic copy of your report (in Word or PDF format). Be sure you include all supporting materials.
2. An electronic copy of your source code including code
3. Continue to use version control. (If you are in Google Collab you have two options. One is to download your code periodically to a directory for your repo and commit. The second is that you can connect Google Collab to GitHub and directly commit your changes there.) Submit an electronic copy of your change logs or a public link for your TA to see your version control logs at GitHub.com.

Advanced Programming Techniques
Assignment 4 Grading

Student:____

MNISTModel.py	4	______	
MNIST Report	4	______	
notMNISTModel.py	4	______	
notMNIST Load/Save/Predict	4	______	
notMNIST Image example improved	4	______	
notMNIST Report	4	______	
CHD train/test data	2	______	
CHDModel.py	6	______	
CHD examining overfitting	4	______	
CHD Report	4	______	
Version control	5	______	
Program Design Quality	5	______	
Total	**50**	______	_____**%**

Letter	**A+**	**A**	**A-**	**B+**	**B**	**B-**	**C+**	**C**	**C-**	**D+**	**D**	**F**
Min. Perc.	**95%**	**90%**	**85%**	**80%**	**75%**	**70%**	**65%**	**60%**	**55%**	**50%**	**45%**	**Below 45%**