**Background**

Our final assignment this semester has three main goals:

1. Implement neural networks as a powerful approach to supervised machine learning,
2. Practice using state-of-the-art software tools and programming paradigms for machine learning,
3. Investigate the impact of parameters to learning on neural network performance as evaluated on empirical data sets.

**Getting Started**

To begin this assignment, please follow this link:

https://classroom.github.com/g/O48-86aK

**Data Sets**

For this assignment, we will learn from the same pre-defined data sets that we began the semester with:

1. **monks1.csv**: A data set describing two classes of robots using all nominal attributes and a binary label. This data set has a simple rule set for determining the label: if head_shape = body_shape ∨ jacket_color = red, then yes, else no. Each of the attributes in the monks1 data set are nominal. Monks1 was one of the first machine learning challenge problems (http://www.mli.gmu.edu/papers/91-95/91-28.pdf). This data set comes from the UCI Machine Learning Repository: [http://archive.ics.uci.edu/ml/datasets/MONK%27s+Problems](http://archive.ics.uci.edu/ml/datasets/MONK%27s+Problems).

2. **iris.csv**: A data set describing observed measurements of different flowers belonging to three species of Iris. The four attributes are each continuous measurements, and the label is the species of flower. The Iris data set has a long history in machine learning research, dating back to the statistical (and biological) research of Ronald Fisher from the 1930s (for more info, see https://en.wikipedia.org/wiki/Iris_flower_data_set). This data set comes from Weka 3.8: [http://www.cs.waikato.ac.nz/ml/weka/](http://www.cs.waikato.ac.nz/ml/weka/) and is also on the UCI Machine Learning Repository: [https://archive.ics.uci.edu/ml/datasets/iris](https://archive.ics.uci.edu/ml/datasets/iris).

3. **mnist_100.csv**: A data set of optical character recognition of numeric digits from images. Each instance represents a different grayscale 28x28 pixel image of a handwritten numeric digit (from 0 through 9). The attributes are the intensity values of the 784 pixels. Each attribute is ordinal (treat them as continuous for the purpose of this assignment) and a nominal label. This version of MNIST contains 100 instances of each handwritten numeric digit, randomly sampled from the original training data for MNIST. The overall MNIST data set is one of the main benchmarks in machine learning:

The file format for each of these data sets is as follows:

- The first row contains a comma-separated list of the names of the label and attributes
- Each successive row represents a single instance
- The first entry (before the first comma) of each instance is the label to be learned, and all other entries (following the commas) are attribute values.
- Some attributes are strings (representing nominal values), some are integers, and others are real numbers. Each label is a string.

**Program**

Your assignment is to write a program called `nn` that behaves as follows:

1) It should take as input six parameters:
   a. The path to a file containing a data set (e.g., monks1.csv)
   b. The number of neurons to use in the hidden layer
   c. The learning rate $\eta$ to use during backpropagation
   d. The number of iterations to use during training
   e. The percentage of instances to use for a training set
   f. A random seed as an integer

   For example, if I wrote my program in Python 3, I might run

   ```
   python3 nn.py mnist_1000.csv 20 0.001 1000 0.75 12345
   ```

   which will create a neural network with 20 neurons in the hidden layer, train the network using a learning rate of $\eta = 0.001$ and 1000 iterations through monks1.csv with a random seed of 12345, where 75% of the data will be used for training (and the remaining 25% will be used for testing)

2) Next, the program should read in the data set as a set of instances, which should be split into training and test sets (using the random seed input to the program)

   a. Unlike our previous learning, instances are now represented by a pair of lists:
      i. A list of all attribute values
      ii. A list of label values

   b. For the attribute values in monks1.csv, you will need to convert each of the attributes into $m-1$ indicator variables using **one hot coding** (where the attribute originally took $m$ values) since each attribute is nominal. For example, the body_shape attribute takes $m = 3$ values (round, square, octagon), so we can create $m-1 = 2$ indicator variables:
\[body\_shape_{Round} = 1 \text{ if } body\_shape = \text{round}, \text{ else } 0\]
\[body\_shape_{Square} = 1 \text{ if } body\_shape = \text{square}, \text{ else } 0\.

**Note:** We will treat all attributes as continuous in iris.csv and mnist_100.csv, so you don’t have to do any pre-processing for these data sets.

c. There is now a list of label values since labels are discrete.

   i. For binary classification tasks (monks1.csv), the list will contain a single label value: 0 if “No”, 1 if “Yes”.

   ii. For multinomial classification tasks (iris.csv and mnist_100.csv), there is one value per possible label value. For example, in the MNIST data sets, the 0th-entry in the list will be a 1 if the label is “zero”, else it will be 0, the 1st-entry in the list will be a 1 if the label is “one”, else it will be 0, etc. To illustrate, a “seven” label will be represented by \([0, 0, 0, 0, 0, 0, 1, 0, 0]\). For iris.csv, you can pick any ordering of the label values as long as it is consistent for every attribute. *Note that this process is slightly different from one-hot coding as we don’t throw away one of the labels when there are three or more.*

3) You should create a neural network in Tensorflow that will be learned from the training data. The key parameters to the architecture of the neural network are based on your inputted parameters and the size of your data set:

   a. The number of attributes in the input layer is the length of each instance’s attribute list (which is the same for all instances)

   b. The number of neurons in a hidden layer will be inputted to the program as a parameter. Each hidden neuron should use \(tf.sigmoid\) as its activation function.

   c. The number of output neurons is the length of each instance’s label list

      i. For monks1.csv, there will be 1 output neuron that should use \(tf.sigmoid\) as its activation function

      ii. For iris.csv, there should be 3 output neurons that should use \(tf.nn.softmax\) as their activation function

      iii. For mnist_100.csv, there should be 10 output neurons that should use \(tf.nn.softmax\) as their activation function

4) You should use different cost/loss functions that the network tries to minimize depending on the number of labels:

   a. For binary classification in monks1.csv, use the sum of squared error

   \[\text{SSE}(X) = \sum_{j=1}^{n} (y^j - \hat{y}^j)^2\]

   The function \(tf.reduce\_sum\) will allow you to sum across all instances.
b. For multinomial classification in iris.csv and mnist_100.csv, use cross-entropy:

\[ CE(X) = - \sum_{j=1}^{n} \sum_{c_i \in C} y_{cj} \log \hat{y}_{ci} \]

which can be implemented with:

```python
cost = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(
    labels=y, logits=net_output))
```

5) For the implementation of Backpropagation, I would recommend using `tf.train.AdamOptimizer` (not just because of it’s awesome name, but because it is the state-of-the-art)

6) You should train your network using your inputted learning rate and for the inputted number of iterations. The iterations are simply a loop that calls Backpropagation a fixed number of times.

7) After training the network, calculate its confusion matrix on the test set. Then the confusion matrix should be output as a file with its name following the pattern:

```
results_<DataSet>_n_<Neurons>n_<LearningRate>r_<Iterations>i_<TrainingPercentage>p_<Seed>.csv
```

Please note that you are allowed to reuse your code from Homework 1 for generating random test/training sets, as well as for creating output files.

**Program Output**

The file format for your output file should be the same as in Homework 1. Please refer back to that assignment for more details.

**Programming Languages**

The primary programming language for Tensorflow is **Python**, so I would most recommend using Python to complete this assignment. However, there is also a library for using Tensorflow in **Java** which is increasingly improving, so for students who really want to use Java, that library might also work. However, I have no experience with it, so your mileage might vary.

**Questions**

Please use your program to answer these questions and record your answers in a README file:

1) Pick a single random seed, a single training set percentage, a single learning rate, and a single number of iterations (document each in your README). Pick five numbers to use for the number of hidden neurons (e.g., 2, 5, 10, 20, 50), then train and evaluate corresponding neural networks for each of the three data sets.

   a. What is the accuracy you observed on each data set for each number of neurons? Plot a line chart (using the tool of your choice: Excel, R, matplotlib in Python, etc.) of the accuracy on each data set as the number of neurons increased.
b. How did the accuracy change as the number of hidden neurons change? Why do you think this result occurred?
c. Calculate a 95% confidence interval for the best accuracy on each data set. How does this accuracy compare to the confidence intervals you calculated in HW1 for k-Nearest Neighbor? Did the neural network learn to outperform kNN?

2) Pick five different training set percentages. Use the number of neurons that gave the highest accuracy in Q1 and the same learning rate and same random seed. With 1000 for the number of iterations:
   a. Plot a line chart (using the tool of your choice: Excel, R, matplotlib in Python, etc.) of the accuracy on each data set as the training set size increased.
   b. Compare the accuracies within each data set – how did they change as the training percentage increased? Do we see the same trends across all three data sets? Why do you think this result occurred?

3) For the mnist_100.csv data set, use the three learning rates $\eta = 0.001, 0.01, 0.05$. Use the number of neurons that gave the highest accuracy in Q1, the training set percentage that gave the highest accuracy in Q2, and the same random seed used in both. Using 1000 for the number of iterations, track the accuracy on the training set and accuracy on the test set of the network trained with each learning rate every 50 iterations.
   a. For each learning rate, plot the training and test accuracy of the network as the number of iterations increased on a line chart (again using your favorite tool).
   b. Compare the training accuracy across the three learning rates. What trends do you observe in your line charts? What do you think this implies about choosing a learning rate?
   c. Compare the testing accuracy across the three learning rates. What trends do you observe in your line charts? What do you think this implies about choosing a learning rate?

**Bonus Question (5 points)**

Modify your program to be able to have multiple hidden layers, all with the same number of neurons. The number of hidden layers to create should be taken in as a seventh parameter on the command line (after the random seed).

Pick three different numbers of hidden layers. Repeat Question 1, except also vary the number of hidden layers based on the set of three that you picked (so that you have 15 different combinations of hidden layers and neurons per layer). How does changing the number of layers further impact the accuracy of the neural network as you also vary the number of hidden neurons per layer?
**README**

Within a README file, you should include:

1) Your answers to the questions above,
2) A short paragraph describing your experience during the assignment (what did you enjoy, what was difficult, etc.)
3) An estimation of how much time you spent on the assignment, and
4) An affirmation that you adhered to the honor code

Please remember to commit your solution code, results files, and README file to your repository on GitHub. You do not need to wait to commit your code until you are done with the assignment; it is good practice to do so not only after each coding session, but maybe after hitting important milestones or solving bugs during a coding session. *Make sure to document your code*, explaining how you implemented the different components of the assignment.

**Honor Code**

Each student is allowed to work with a partner to complete this assignment. Groups are also allowed to collaborate with one another to discuss the abstract design and processes of their implementations. For example, please feel free to discuss the process of creating a neural network in Tensorflow, how to create your instances, and how to track accuracies. However, sharing code (either electronically or looking at each other’s code) between groups is not permitted.