**\_\_\_\_\_**

PHYS 243 Final Project- Neural Networks

Dominique Acuna

August 23, 2019

Table of Contents

1.0 Task Summary 3

2.0 Neural Networks 3

2.1 Find simplest Neural networks 4

2.1.1 Different activations/Layers/Number of neurons 5

2.2 Build new attributes for new neural network 6

3.0 Recommendations/Comments 6

Appendix A- Python Code for Simple Neural Network 7

Appendix B- Python Code for New attributes Neural Network 8

# 1.0 Task Summary

The purpose of this project is to build a neural network model for a classification task on the data provided. To do this, different activations, number of layers, and number of neurons will be compared to see differences in performance. The one with the least complexity will be chosen.

Then, new attributes will be created to train the neural network. These new attributes are listed in below.

X 3 = X 12

X4 = X2

X5 = X1X2

The simplest Neural network will be found for the following set of inputs:

1. {X3,X4}

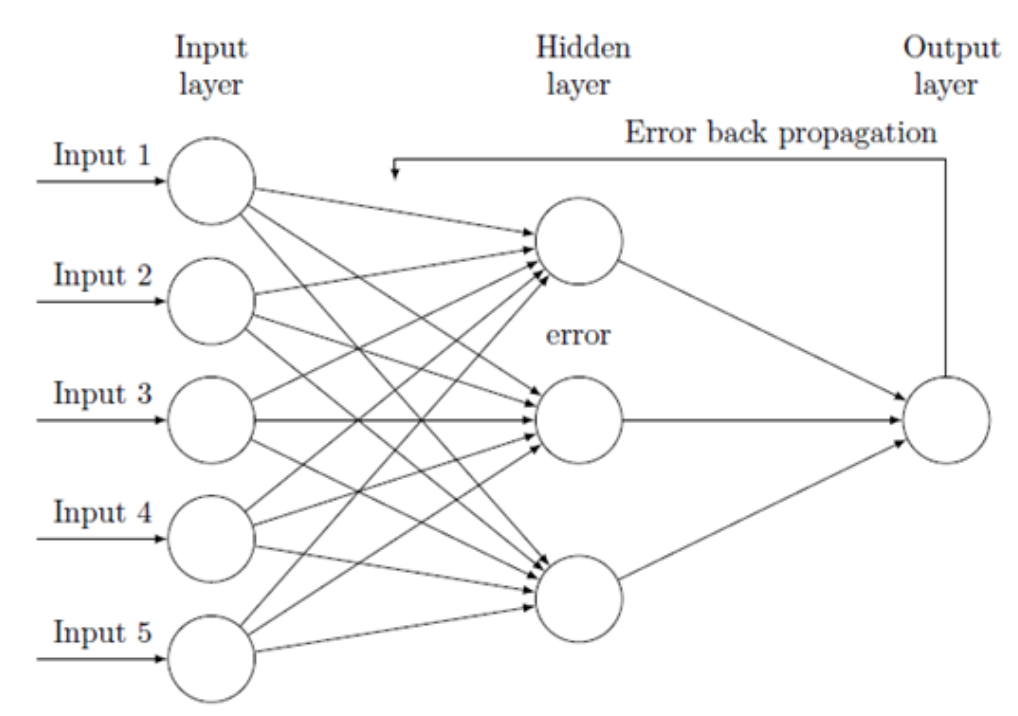
2. {X3,X5}

3. {X3,X4,X5}

4. {X1,X2,X3,X4,X5}

# 2.0 Neural Networks

A Neural network is used when a computer learns to perform some task by analyzing training examples. It is a type of machine learning which models itself after the human brain. This creates an artificial neural network that via an algorithm allows the computer to learn by incorporating new data.



**Figure 1. Neural Network Visual**

Figure 1 is a visual representation of the layers that will be used in our dataset.

## 2.1 Find simplest Neural networks

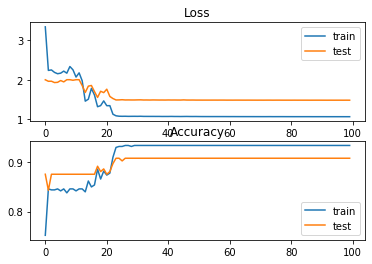
A neural network was constructed using the numpy and keras libraries. Numpy allows for use of loading our data files and keras allows for the modeling of the data through the sequential and dense applications. An input layer dimension of 2 was chosen because there were 2 features in the initial dataset. Fifty nodes in the hidden layer were used with 100 epochs.

After evaluating our model along with the accuracy, we can see that it resulted in 93% accuracy for the training dataset and about 91% for the test dataset. See Figure 2 for Python output.

**Macintosh HD:private:var:folders:s6:tl68phrn709c8hz6zrq6gpvm0000gn:T:TemporaryItems:Screen Shot 2019-08-22 at 9.44.56 PM.png**

**Figure 2. Accuracy Results**

Figure 3 shows the cross-entropy loss over epochs for the train and test dataset, and the bottom plot showing classification accuracy over epochs.

￼￼￼

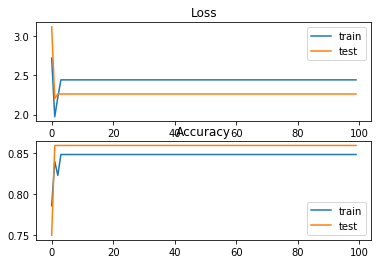
**Figure 3. Cross-Entropy Loss**

### 2.1.1 Different activations/Layers/Number of neurons

Using 350 to train and 100 nodes in the hidden layer, the results are the following:

Macintosh HD:private:var:folders:s6:tl68phrn709c8hz6zrq6gpvm0000gn:T:TemporaryItems:Screen Shot 2019-08-22 at 9.59.31 PM.png

**Figure 4. Accuracy for 350/100**

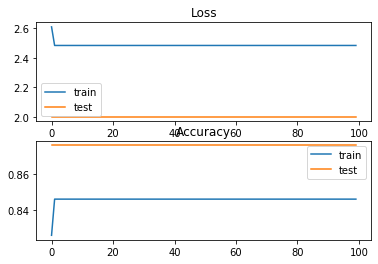
****

**Figure 5. Cross-Entropy Loss with 350/100**

Adding just more nodes to 200, we get the following results:

Macintosh HD:private:var:folders:s6:tl68phrn709c8hz6zrq6gpvm0000gn:T:TemporaryItems:Screen Shot 2019-08-22 at 10.09.40 PM.png

**Figure 4. Accuracy for increasing nodes to 200**

****

**Figure 5. Cross-Entropy for 200 Nodes**

Accuracy is decreasing as more nodes and amounts of records to train are increasing.

## 2.2 Build new attributes for new neural network

See Appendix B for Python Code

# 3.0 Recommendations/Comments

Overall, this is a dataset with many instances. After creating a neural network and going through many scenarios, one can see that 100 epochs with 50 nodes in the hidden layer worked best.

# Appendix A- Python Code for Simple Neural Network

See Github for better format

Note: r, g, b were converted to 0,1,2 respectively previous to codingfrom keras.layers import Dense
from keras.models import Sequential
from keras.optimizers import SGD
from keras.utils import to\_categorical
from matplotlib import pyplot
from numpy import loadtxt
dataset = loadtxt('/Users/dominiquecacuna/Desktop/UCR/PHYS 243/train\_set.txt', delimiter=',')
# split into input (X) and output (y) variables
X = dataset[:,0:2]
y = dataset[:,2]
# one hot encode output variable
y = to\_categorical(y)
# split into train and test
n\_train = 500
trainX, testX = X[:n\_train, :], X[n\_train:, :]
trainy, testy = y[:n\_train], y[n\_train:]
# define model
#50 nodes in the hidden layer and the rectified linear activation function, and an output layer that must be customized based on the selection of the loss function.
model = Sequential()
model.add(Dense(50, input\_dim=2, activation='relu', kernel\_initializer='he\_uniform'))
model.add(Dense(3, activation='softmax'))
# compile model
#fit using stochastic gradient descent with a sensible default learning rate of 0.01 and a momentum of 0.9
opt = SGD(lr=0.01, momentum=0.9)
model.compile(loss='categorical\_crossentropy', optimizer=opt, metrics=['accuracy'])
# fit model
#100 epochs on the training dataset and the test dataset will be used as a validation dataset, allowing us to evaluate both loss and classification accuracy on the train and test sets at the end of each training epoch and draw learning curves.
history = model.fit(trainX, trainy, validation\_data=(testX, testy), epochs=100, verbose=0)
# evaluate the model
\_, train\_acc = model.evaluate(trainX, trainy, verbose=0)
\_, test\_acc = model.evaluate(testX, testy, verbose=0)
print('Train: %.3f, Test: %.3f' % (train\_acc, test\_acc))
# plot loss during training
pyplot.subplot(211)
pyplot.title('Loss')
pyplot.plot(history.history['loss'], label='train')
pyplot.plot(history.history['val\_loss'], label='test')
pyplot.legend()
# plot accuracy during training
pyplot.subplot(212)
pyplot.title('Accuracy')
pyplot.plot(history.history['acc'], label='train')
pyplot.plot(history.history['val\_acc'], label='test')
pyplot.legend()
pyplot.show()

# Appendix B- Python Code for New attributes Neural Network

See Github for better format

#!/usr/bin/env python3
# -\*- coding: utf-8 -\*-
"""
Created on Thu Aug 22 21:18:14 2019
@author: dominiquecacuna
Find the simplest Neural network for the following set of inputs:
1. {X3,X4}
2. {X3,X5}
3. {X3,X4,X5}
4. {X1,X2,X3,X4,X5}
"""
from keras.layers import Dense
from keras.models import Sequential
from keras.optimizers import SGD
from keras.utils import to\_categorical
from matplotlib import pyplot
from numpy import loadtxt
import pandas
dataset = loadtxt('/Users/dominiquecacuna/Desktop/UCR/PHYS 243/train\_set.txt', delimiter=',')
# split into input (X) and output (y) variables
X1= dataset[:,0:]
X2= dataset[:,1:]
X3= X1\*X1
X4= X2\*X2
X5= X1\*X2
y = dataset[:,2]
# one hot encode output variable
y = to\_categorical(y)
# split into train and test
n\_train = 500
trainX, testX = X[:n\_train, :], X[n\_train:, :]
trainy, testy = y[:n\_train], y[n\_train:]
# define model
#50 nodes in the hidden layer and the rectified linear activation function, and an output layer that must be customized based on the selection of the loss function.
model = Sequential()
model.add(Dense(50, input\_dim=2, activation='relu', kernel\_initializer='he\_uniform'))
model.add(Dense(3, activation='softmax'))
# compile model
#fit using stochastic gradient descent with a sensible default learning rate of 0.01 and a momentum of 0.9
opt = SGD(lr=0.01, momentum=0.9)
model.compile(loss='categorical\_crossentropy', optimizer=opt, metrics=['accuracy'])
# fit model
#100 epochs on the training dataset and the test dataset will be used as a validation dataset, allowing us to evaluate both loss and classification accuracy on the train and test sets at the end of each training epoch and draw learning curves.
history = model.fit(trainX, trainy, validation\_data=(testX, testy), epochs=100, verbose=0)
# evaluate the model
\_, train\_acc = model.evaluate(trainX, trainy, verbose=0)
\_, test\_acc = model.evaluate(testX, testy, verbose=0)
print('Train: %.3f, Test: %.3f' % (train\_acc, test\_acc))
# plot loss during training
pyplot.subplot(211)
pyplot.title('Loss')
pyplot.plot(history.history['loss'], label='train')
pyplot.plot(history.history['val\_loss'], label='test')
pyplot.legend()
# plot accuracy during training
pyplot.subplot(212)
pyplot.title('Accuracy')
pyplot.plot(history.history['acc'], label='train')
pyplot.plot(history.history['val\_acc'], label='test')
pyplot.legend()
pyplot.show()