INF 552
Sudeeptha Mouni Ganji (ID: 2942771049)
Vanessa Tan (ID: 4233243951)

Programming Assignment 5 Report

PART 1: IMPLEMENTATION

Back Propagation Algorithm:

Data Structures:

- **Lists:** We have used lists to store the gesture names from the training data and test data as they maintain the order of the input. we have separated the down gesture and the remaining gestures into two separate lists named, files_one and files_zero respectively.
- **Numpy.ndarray:** We have read the gestures and stored them along with their labels into a numpy.ndarray for efficiency and ease of usage. We have also divided the training data and stored them into numpy.ndarray for easy data manipulation.
- **Class object:** We have stored the neural network as a class object. We have passed the number of input layers, hidden layers and perceptrons as class variables.

Code-level optimizations:

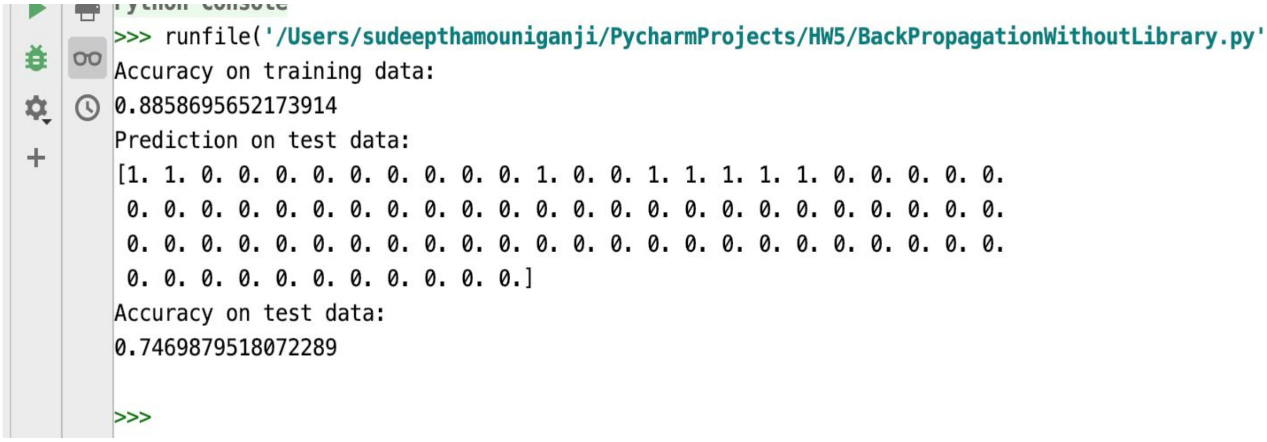
- We created our own class to encapsulate the neural network as the existing basic data structures will not be suitable for storing a neural network.
- We have reshaped our images and stored them into lists and appended labels to the image list for quicker search and execution of operations.
- We have randomly split our training data and shuffled them to reduce chances of bias.

Challenges:

- Our foremost challenge was in deciding which initial values to give the various parameters and weights.
- We had some difficulty in trying to improve the accuracy of the data. We made several code changes to get an optimal balance between avoiding overfitting of data and underfitting of data.

Output:

- The accuracy on my training data is usually around 88% while my accuracy on downgesture_test.list (test data) usually ranges from 70-78 %. This could be because of initializing random weights at the beginning as well as slight overfitting of data due to small size of training data and test data.



```
>>> runfile('/Users/sudeepthamouniganji/PycharmProjects/HW5/BackPropagationWithoutLibrary.py'
Accuracy on training data:
0.8858695652173914
Prediction on test data:
[1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 1. 1. 1. 1. 1. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Accuracy on test data:
0.7469879518072289

>>>
```

PART 2: SOFTWARE FAMILIARIZATION

Back Propagation Algorithm

Libraries

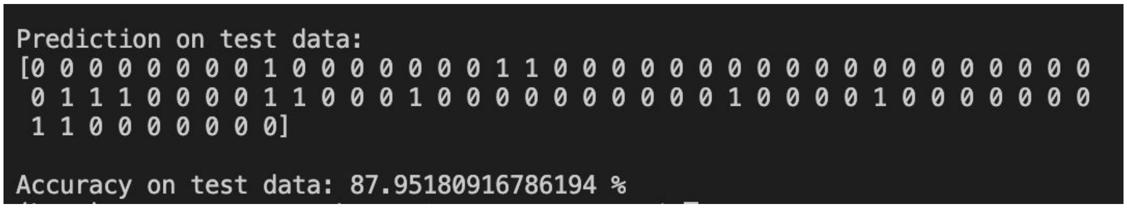
- **Keras**: We used the Sequential model to build the neural network and added in Dense layers to build the 960-100-1 network layout. We used the SGD (stochastic gradient descent) optimizer and mean squared error loss function for our model as well.
- **Pandas**: We used the pandas data frame structure to read in the training and test data and added in a column for the data labels.
- **Numpy**: We converted the pandas data frames into numpy arrays for the Keras library to take in the training and test data.
- **Skimage**: This is a scikit-image image processing library that we used to convert the pmg images into numerical data for our network model.

Implementation Comparison

- We parsed through the training and test data in similar ways for the back propagation implementations with and without a library since we had to prepare the data for our neural networks. We had to convert the image data into a 2D array of 960x184 integers. We also got the labels for the training and test data by checking if "down" was included in the image file name. For the library implementation, we used the stochastic gradient descent optimizer with the sigmoid activation function. We fit the model with the training data and then used it to evaluate and predict the output on the test data.

Result Comparison

- Prediction on test data:
  [0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 1 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0]



```
Prediction on test data:
[0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 1 1 1 0 0 0 0 1 1 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0
 1 1 0 0 0 0 0 0 0]

Accuracy on test data: 87.95180916786194 %
```

- Accuray on test data: 87.95 %

Possible Improvements

- While implementing our algorithm, we came across some overfitting of data. This could be reduced by an increase in the data size.
- Increasing the number of hidden layers and perceptrons and making necessary code changes accordingly could help to improve the accuracy of the program especially when working with larger data sets.

PART 3: APPLICATIONS

- **Sonar target recognition:** A two layer back propagation network can be trained to classify reflected sonar signals of rocks and metal cylinders at the bottom of Chesapeake Bay. The input data could be taken in based on Fourier transform of raw time signal.

- **Complex trait prediction in cattle:** Artificial neural networks are capable of capturing relationships between single nucleotide polymorphisms and phenotypic values without a genetic model. Genomic covariate structures for Holstein-Friesian and German Fleckvieh cattle were used as network input to assess their ability to predict milk traits using large scale single nucleotide polymorphism data.
- **Image classification**: A neural network model can be trained to classify cats and dogs from a set of images. The image training data is passed in as input along with the correct classifications to the network.

Group Members and Contributions

- Our group members for this programming assignment were Sudeeptha Mouni Ganji (ID: 2942771049) and Vanessa Tan (ID: 4233243951).
- Vanessa and Sudeeptha worked together to research and understand the algorithms.
- After implementation, Sudeeptha and Vanessa worked together to write the report.

Programming:

Algorithm Implementation : Sudeeptha Mouni Ganji, Vanessa Tan
Library Implementation: Sudeeptha Mouni Ganji, Vanessa Tan

Report Writing:

Part 1: ● Back propagation algorithm	Vanessa Tan Sudeeptha Mouni Ganji
Part 2: ● Back Propagation algorithm	Vanessa Tan Sudeeptha Mouni Ganji
Part 3: ● Back Propagation algorithm	Sudeeptha Mouni Ganji Vanessa Tan