CS60050 - Machine Learning

Assignment 2

Song Popularity using Neural Networks

Bratin Mondal

21CS10016

Department of Computer Science and Engineering,
Indian Institute of Technology Kharagpur

1 Problem Statement

1.1 Objective:

This project aims to build neural network models for song popularity prediction based on a comprehensive set of music characteristics.

1.2 Model Specifications:

1. **ANN Specification 1**
    - No of hidden layers: 1
    - No. of neurons in hidden layer: 32
    - Activation function in the hidden layer: Sigmoid
    - 1 neuron in the output layer.
    - Activation function in the output layer: Linear
    - Optimization algorithm: Mini Batch Stochastic Gradient Descent (SGD)
    - Loss function: Mean Squared Error (MSE)
    - Learning rate: 0.01
    - No. of epochs = 200
2. **ANN Specification 2**
    - No of hidden layers: 2
    - No. of neurons in the 1st hidden layer: 64
    - No. of neurons in the 2nd hidden layer: 32
    - Activation function in both the hidden layers: ReLU
    - 1 neuron in the output layer.
    - Activation function in the output layer: Linear
    - Optimization algorithm: Mini Batch Stochastic Gradient Descent (SGD)
    - Loss function: Mean Squared Error (MSE)
    - Learning rate: 0.01
    - No. of epochs = 200

2 Modules

2.1 Preprocess

Purpose: Preprocesses the input dataset by dropping duplicates, normalizing the data, and splitting it into training and testing datasets.

2.2 DataLoader

Purpose: Loads the training and testing datasets and splits them into batches of size 32 for efficient training.

2.3 WeightInitialiser

Purpose: Initializes the weights of the dense layers in the neural network. It generates random weights within a specified range.

2.4 BiasInitialiser

Purpose: Initializes the biases of the dense layers in the neural network. It initializes biases for each layer to zero.

2.5 ForwardPass

Purpose: Performs the forward pass of the neural network. It calculates the outputs of each layer using the given input data and weights.

2.6 BackPropagation

Purpose: Performs the backward propagation of the neural network. It updates the weights and biases of the network based on the calculated gradients.

2.7 Training

Purpose: Trains the neural network by iterating over epochs. It utilizes forward pass and backpropagation to update model parameters and computes accuracy for each epoch.

2.8 Predict

Purpose: Predicts the labels for the training and testing datasets using the trained neural network model. It utilizes forward pass to obtain predictions.

3 Helper Functions

3.1 curAccu

Purpose: Calculates the accuracy of the model by comparing predicted labels with the ground truth labels.

3.2 BatchSplit

Purpose: Splits the dataset into batches of size 32 to facilitate mini-batch training.

3.3 gradSig

Purpose: Calculates the gradient of the sigmoid function. It's used in backpropagation for layers with sigmoid activation.

3.4 sig

Purpose: Calculates the sigmoid function. It's used as the activation function for the hidden layers.

3.5 reluDerivative

Purpose: Calculates the gradient of the ReLU function. It's used in backpropagation for layers with ReLU activation.

3.6 relu

Purpose: Calculates the ReLU function. It's used as the activation function for the hidden layers.

4 Model

4.1 Data Preprocessing

The input dataset is preprocessed by dropping duplicates and normalizing the data.

$$A_{ij}^{\text{processed}} = \frac{A_{ij} - \mu_j}{\sigma_j}$$

where A_{ij} is the j^{th} attribute of the i^{th} data point, μ_j is the mean of the j^{th} attribute, and σ_j is the standard deviation of the j^{th} attribute. The dataset is then split into training and testing datasets in the ratio 80:20.

4.2 Training

The model specifications are same as described in 1.2. The target label is scaled using the MinMaxScaler using the fomula:

$$y^{\text{processed}} = \frac{y}{100}$$

The neural network is trained using the training dataset. The weights and biases are initialized using the WeightInitialiser and BiasInitialiser modules. The training is performed using the Mini Batch Stochastic Gradient Descent (SGD) optimization algorithm. The loss function used is Mean Squared Error (MSE). The learning rate is set to 0.01 and the number of epochs is set to 200. The model is trained using the forward pass and backpropagation modules. The accuracy of the model is calculated after 10 epochs and the training and testing accuracies are plotted against the number of epochs.

4.3 Results

The training and testing accuracies for the two models are plotted against the number of epochs. The training and testing accuracies for the two models are as follows:

4.3.1 ANN Specification 1

- Training Accuracy: 1.901331769829969 %
- Testing Accuracy: 2.1105527638190953 %

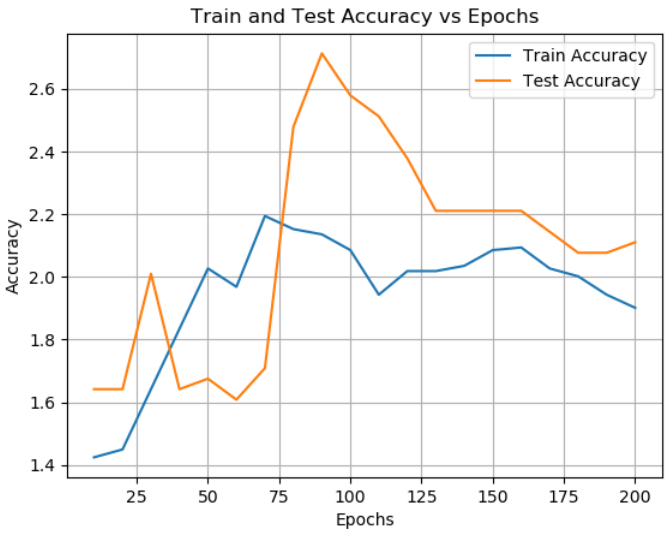

Figure 1: Training and Testing Accuracies for ANN Specification 1

4.3.2 ANN Specification 2

- Training Accuracy: 2.1105527638190953 %
- Testing Accuracy: 2.1105527638190953 %

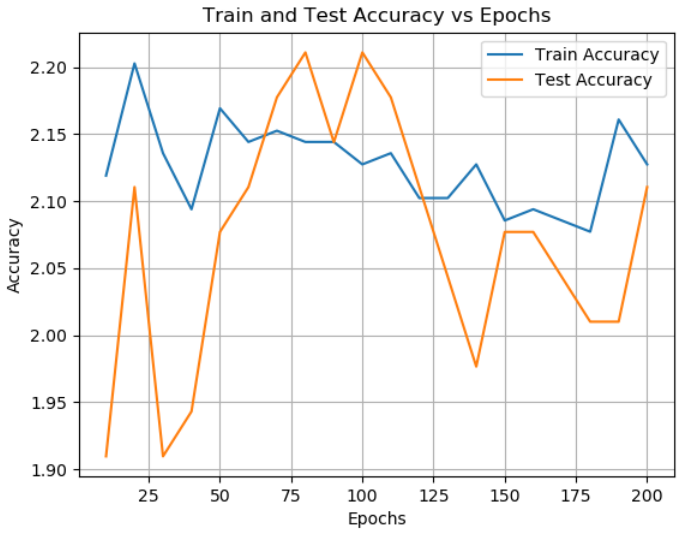

Figure 2: Training and Testing Accuracies for ANN Specification 2

5 Implementation using scikit-learn MLP

The same ANN specifications are implemented using the scikit-learn library. Two modeuls of the scikit-learn library are used: **MLPRegressor** and **MLR Classifier**.

5.1 MLPClassifier

5.1.1 ANN Specification 1

- Training Accuracy: 4.640254627690761 %
- Testing Accuracy: 2.4790619765494135 %

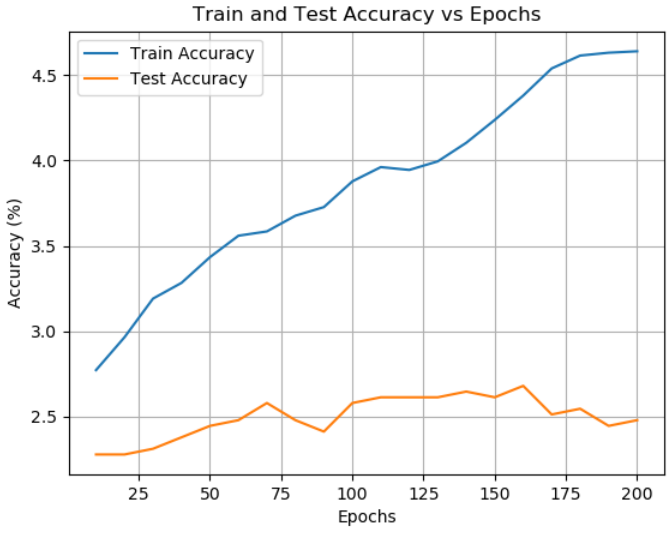

Figure 3: Training and Testing Accuracies for MLPClassifier Specification 1

5.1.2 ANN Specification 2

- Training Accuracy: 8.468045899991624 %
- Testing Accuracy: 1.2730318257956448 %

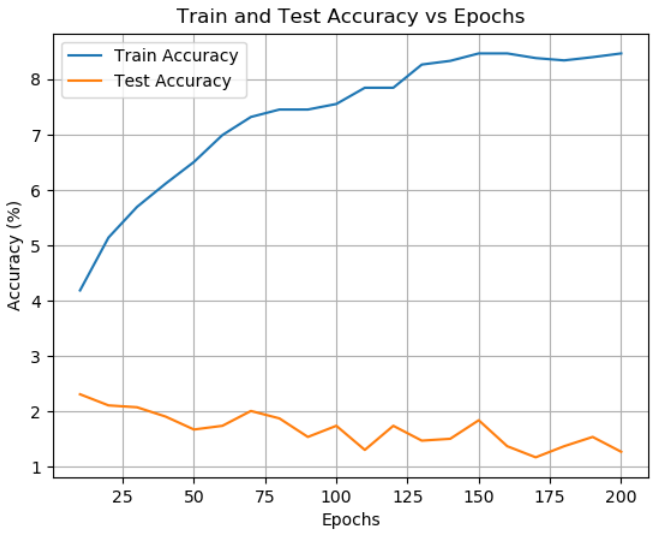

Figure 4: Training and Testing Accuracies for MLPClassifier Specification 2

5.2 MLPRegressor

5.2.1 ANN Specification 1

- Training Accuracy: 1.6249267107797973 %
- Testing Accuracy: 2.0435510887772192 %

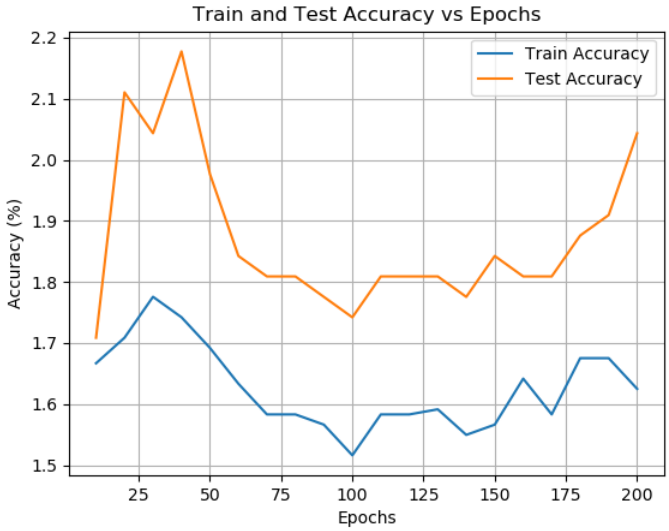

Figure 5: Training and Testing Accuracies for MLPRegressor Specification 1

5.2.2 ANN Specification 2

- Training Accuracy: 1.7840690175056537 %

- Testing Accuracy: 1.574539363484087 %

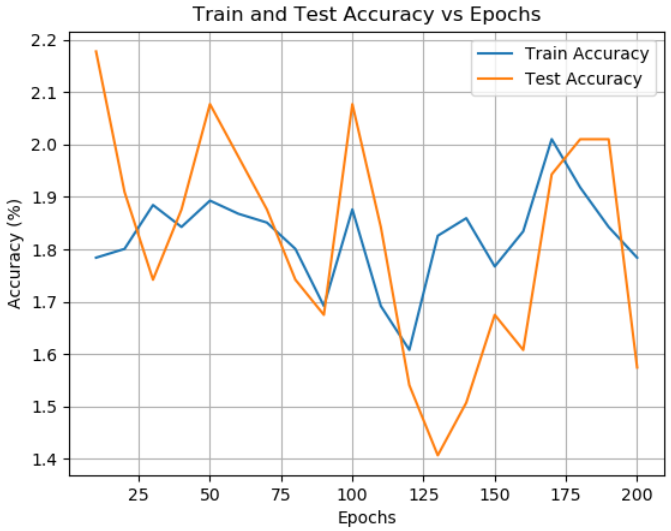

Figure 6: Training and Testing Accuracies for MLPRegressor Specification 2

6 Improving the Model

Using the scikit-learn library, we observed that MLPClassifier performs better than MLPRegressor on training set. We further use this idea to improve our model. Instead of predicting the popularity as a target, we create 101 classes for the popularity and use softmax activation function in the output layer and cross entropy loss function and choose the class with the highest probability as the predicted popularity. We use the same ANN specifications as described in 1.2 along with some minor changes.

1. **ANN Specification 1**
    - No of hidden layers: 1
    - No. of neurons in hidden layer: 32
    - Activation function in the hidden layer: Sigmoid
    - 101 neurons in the output layer.
    - Activation function in the output layer: Softmax
    - Optimization algorithm: Mini Batch Stochastic Gradient Descent (SGD)
    - Loss function: Categorical Cross Entropy
    - Learning rate: 0.01
    - No. of epochs = 200
2. **ANN Specification 2**
    - No of hidden layers: 2
    - No. of neurons in the 1st hidden layer: 64
    - No. of neurons in the 2nd hidden layer: 32
    - Activation function in both the hidden layers: ReLU
    - 101 neurons in the output layer.
    - Activation function in the output layer: Softmax

- Optimization algorithm: Mini Batch Stochastic Gradient Descent (SGD)
- Loss function: Categorical Cross Entropy
- Learning rate: 0.01
- No. of epochs = 200

6.1 Results

The training and testing accuracies for the two models are plotted against the number of epochs. The training and testing accuracies for the two models are as follows:

6.1.1 ANN Specification 1

- Training Accuracy: 3.769159896138705 %
- Testing Accuracy: 2.3450586264656614 %

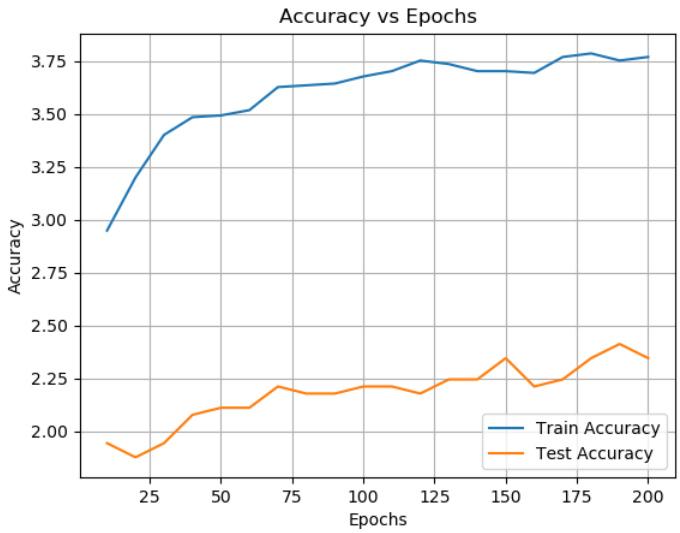

Figure 7: Training and Testing Accuracies for ANN Specification 1

6.1.2 ANN Specification 2

- Training Accuracy: 3.149342490995896 %
- Testing Accuracy: 2.4120603015075375 %

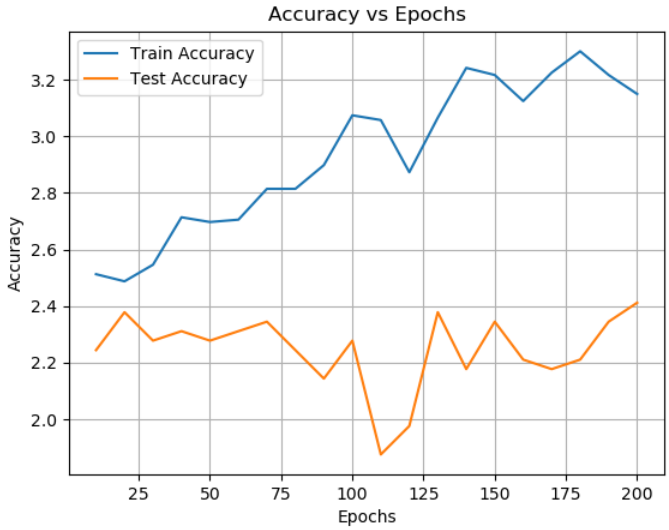

Figure 8: Training and Testing Accuracies for ANN Specification 2

7 Conclusion

We observe that the improved model performs better than the previous models. The training and testing accuracies are higher than the previous models. The improved model is able to predict the popularity of the songs with higher accuracy. We also note that since there are very less number of attributes, the model structure is not very complex and the target variable ranges over a large range of $1 - 100$, the model specifications could not acheive very high accuracy. We can further improve the model by adding more attributes, increasing the complexity of the model and changes in the loss function and activation functions.

A Appendix

A.1 Running the Code

Use the makefile to run the code.
For section 4 enter the following command:

```
make model
```

For section 5 enter the following command:

```
make test_scikit_learn
```

For section 6 enter the following command:

```
make mymodel
```

See *readme.md* for more details.