CS 6375

ASSIGNMENT 02 – Part II

Names of students in your group:

Dishant Khanna (2021438172)

Ekansh Goyal (2021474594)

Number of free late days used:  0

Note: You are allowed a **total** of 4 free late days for the **entire semester**. You can use at most 2 for each assignment. After that, there will be a penalty of 10% for each late day.

Please list clearly all the sources/references that you have used in this assignment.

**Project Report**

**Title:** Implementation of Neural Network.

**Assumptions made:**

The user will provide valid paths for training, validation & testing datasets.

**Accomplishments:**

·        Successfully created the Neural Network with 2 hidden layers besides input and output layer.

·        Successfully implemented different activation functions of neural network e.g. tanh and relu.

·        Successfully calculated the accuracy of training and testing datasets.

·        Successfully pre-processed the data, which includes converting non-numerical attributes e.g. categorical attributes to numerical values, standardizing, and scaling the attributes.

·        Successfully corrected the weights.

**RESULTS**

* For Datasets 01(iris): error on training data is \_\_\_. After \_\_\_ iterations and learning rate of \_\_ the error reduced to \_\_\_.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Function Used | Max Iterations | Learning Rate | Training Error | Testing Error |
| Sigmoid | 10000 | 0.05 | 0.50 | 0.15 |
| Tanh | 10000 | 0.05 | 23.624 | 6.624 |
| Relu | 10000 | 0.05 | 25.625 | 5.125 |

* For Datasets 02(wine): error on training data is \_\_\_. After \_\_\_ iterations and learning rate of \_\_ the error reduced to \_\_\_.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Function Used | Max Iterations | Learning Rate | Training Error | Testing Error |
| Sigmoid | 10000 | 0.005 | 1.292 | 0.451 |
| Tanh | 10000 | 0.005 | 0.788 | 0.738 |
| Relu | 10000 | 0.005 | 21.824 | 5.979 |

* For Datasets 03(cars): error on training data is \_\_\_. After \_\_\_ iterations and learning rate of \_\_ the error reduced to \_\_\_.

NOTE: This dataset contains Non-Numerical attributes, which have been taken care by data pre-processing.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Function Used | Max Iterations | Learning Rate | Training Error | Testing Error |
| Sigmoid | 1000 | 0.007 | 50.696 | 13.811 |
| Tanh | 1000 | 0.007 | 221.99 | 52.666 |
| Relu | 1000 | 0.007 | 240.666 | 64.333 |

**Best results :**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* IRIS \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

SIGMOID ACTIVATION FUNCTION

After 10000 iterations, the total error is 0.5053148684103779

The final weight vectors are (starting from input to output layers)

[[-2.2643578 -0.16755632 1.93125533 -0.2961835 ]

[ 3.79370481 -1.5562071 -2.2901324 1.95310896]

[ 1.02333214 -0.90861198 -0.08883524 -0.85608526]

[ 0.96989945 1.01583473 0.0064889 -1.34584844]]

[[-3.76398355 0.83176217]

[ 1.36183629 2.31976173]

[ 1.01513085 3.32659475]

[ 8.44545285 -3.39691266]]

[[-8.3287886]

[ 7.9593744]]

Test Output error: 0.15138451600929725

**Sigmoid Activation function out performs every other activation function as seen from the results.**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Wine \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

SIGMOID ACTIVATION FUNCTION

After 10000 iterations, the total error is 1.2928654791788285

The final weight vectors are (starting from input to output layers)

[[-0.44432925 0.05794698 -1.21124174 -3.69025999]

[-0.36675336 -0.45314212 1.13545473 0.44553379]

[-0.28629994 -0.02756575 -0.4718404 -1.24559636]

[-0.38385641 0.10018492 -0.2210971 0.63779295]

[ 0.12128657 -0.05743642 -0.16548284 -0.95150123]

[-0.3063777 -1.08021581 0.03863601 1.68584188]

[ 0.61416647 0.32439058 0.59044881 0.07203996]

[ 0.74210454 -0.04232407 -0.50573837 -0.06145845]

[-0.61965952 -0.87754018 -0.77999675 -1.46337005]

[ 0.77809878 -0.24269799 -0.60621482 0.69362257]

[-0.80300804 -1.07961205 1.34194009 0.59698812]

[ 0.51319821 0.542917 -0.77374307 0.46424531]

[ 0.64393459 0.01516956 0.2238608 -0.28198644]]

[[-0.7623834 0.8995487 ]

[-0.96989184 1.80930809]

[ 0.18685129 -0.55132749]

[ 2.14255422 -2.50619871]]

[[ 3.03098839]

[-3.14072319]]

Test Output error: 0.45103357428344

**Sigmoid Activation function out performs every other activation function as seen from the results.**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Cars \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

SIGMOID ACTIVATION FUNCTION

After 1000 iterations, the total error is 50.6901485418464

The final weight vectors are (starting from input to output layers)

[[ 0.66210441 -0.10313464 -0.05786094 -0.17967419]

[ 0.87881746 -0.31914481 0.25642912 -0.35042409]

[ 0.81295843 -0.06520471 -0.26842439 0.65159515]

[-0.59004565 -1.41872404 2.86416231 2.62896154]

[ 0.34367875 -0.60836216 0.14061435 -0.44613735]

[-0.22603615 -0.42482328 0.07965035 -0.50176075]]

[[ 0.41685904 1.02102028]

[-0.08301269 1.44032762]

[ 0.45367236 -1.9906327 ]

[ 0.28096154 -2.229913 ]]

[[-0.67175692]

[ 3.27330921]]

Test Output error: 13.811408770258447

**Sigmoid Activation function out performs every other activation function as seen from the results.**

**Learning:**

* Learnt programming in python
* Learnt the implementation of the Back-Propagation Algorithm in accomplishment of Neural Network.
* Analysed different data pre-processing methods and implemented several of them.
* The pre-processing strategy we used is based on converting non-numerical attributes e.g. categorical attribute to numerical values, standardization, and scaling the attributes.

**Pre-Processing Strategy Used:**

* We have used “LabelEncoder” to transform non-numerical labels to numerical labels. It encodes the labels with the value between 0 and n\_classes-1.
* We have used “SimpleImputer” for filling out the missing values in the dataset with the mean of the values in that column.
* We have used “MinMaxScaler” for standardization and scaling the attributes in the given range on the dataset. We have converted each value in the dataset within the range 0-1

**Note:** Screenshots and complete output of a run of the program are present in the folder ‘output’ for reference.

**References:**

1. <https://pandas.pydata.org/>
2. <https://docs.python.org/3/>
3. <https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html>
4. <https://scikit-learn.org/stable/modules/generated/sklearn.impute.SimpleImputer.html>
5. <https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html>