***CS 6375***

***ASSIGNMENT 3 (Neural Networks)***

**Akhila Kancharana**

**(AXK180025)**

**Programming Part:**

**Dataset Links:**

1. Car Evaluation Dataset: <https://archive.ics.uci.edu/ml/datasets/Car+Evaluation>
2. Iris Dataset: <https://archive.ics.uci.edu/ml/datasets/Iris>
3. Adult Census Income Dataset: <https://archive.ics.uci.edu/ml/datasets/Census+Income>

**Pre-Processing:** The datasets are pre-processed by standardizing and scaling attributes.

Each dataset is preprocessed separately as the need for each dataset is different.

**Neural Net:**

Input Layers:1

Hidden Layers:2

Output Layers: 1

**Note:** number of nodes in each hidden layer can be changed.

The following parameters are changed accordingly to get better efficiency

* Learning rate
* Number of nodes in hidden layer
* Number of iterations

**Value Set-1:**

Number of iterations = 1000

Learning rate values -> 0.01, 0.05, 0.1,0.2

Number of nodes in hidden layer 1 = 4

Number of nodes in hidden layer 2 = 2

In Value Set-1 we analyze how the final output and error rate varies with change In learning rate.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Parameter ƞ** | **Dataset** | **Activation function** | **Training Error** | **Test Error** |
| 0.01 | Adult | Sigmoid | 0.06133809523427166 | 0.06128822044065387 |
|  |  | tanh | 0.8039414787798409 | 0.4547413793103448 |
|  |  | Relu | 0.12477204907161804 | 0.1231763925729443 |
|  | Car | Sigmoid | 0.09059549511322118 | 0.1648049849560462 |
|  |  | tanh | 1.2156804117236126 | 0.4388120629909528 |
|  |  | Relu | 0.3262128892107169 | 0.4985507246376812 |
|  | Iris | Sigmoid | 0.20942101300967889 | 0.20172987002786946 |
|  |  | tanh | 0.380734525670505 | 0.4379569506450124 |
|  |  | Relu | 0.8235294117647058 | 0.7931034482758621 |
| 0.05 | Adult | Sigmoid | 0.12477204905953515 | 0.12317639256136305 |
|  |  | tanh | 0.7705363063660478 | 0.6573275862068966 |
|  |  | Relu | 0.12477204907161804 | 0.1231763925729443 |
|  | Car | Sigmoid | 0.08773613789463233 | 0.16036045799015725 |
|  |  | tanh | 0.5561181592451064 | 1.23167906693983 |
|  |  | Relu | 0.3262128892107169 | 0.4985507246376812 |
|  | Iris | Sigmoid | 0.20645637921539503 | 0.1988399186137974 |
|  |  | tanh | 0.3707221047070063 | 0.39036940196066133 |
|  |  | Relu | 0.8235294117647058 | 0.7931034482758621 |
| 0.1 | Adult | Sigmoid | 0.12477204907161804 | 0.1231763925729443 |
|  |  | tanh | 0.8032783488063661 | 0.45971485411140584 |
|  |  | Relu | 0.12477204907161804 | 0.1231763925729443 |
|  | Car | Sigmoid | 0.17139841336305903 | 0.2929193865913792 |
|  |  | tanh | 0.5271542360608255 | 1.4855072463768109 |
|  |  | Relu | 0.3262128892107169 | 0.4985507246376812 |
|  | Iris | Sigmoid | 0.2061587017995282 | 0.19854861106424357 |
|  |  | tanh | 0.4277856766563455 | 0.33969682708394505 |
|  |  | Relu | 0.8235294117647058 | 0.7931034482758621 |
| 0.2 | Adult | Sigmoid | 0.12477204907161804 | 0.1231763925729443 |
|  |  | tanh | 0.8052677387267905 | 0.7936007957559682 |
|  |  | Relu | 0.12477204907161804 | 0.1231763925729443 |
|  | Car | Sigmoid | 0.17590295449724253 | 0.29761438282399794 |
|  |  | tanh | 1.2150615496017378 | 1.4855072463768115 |
|  |  | Relu | 0.3262128892107169 | 0.4985507246376812 |
|  | Iris | Sigmoid | 0.20601709450523228 | 0.1984099768946551 |
|  |  | tanh | 0.8235294117647058 | 0.7931034482758621 |
|  |  | Relu | 0.8235294117647058 | 0.7931034482758621 |

**Analysis**

We started with experimenting the different values of learning rate. Initially we started with relatively higher values of learning rate such as 0.1, 0.2 and then lower values such 0.05, 0.01

If the learning rate is low then the training error seems reliable but optimization can take lot of time. In conclusion, if training rate is very small, then the gradient descent can be low.

If learning rate is high, then the training may not converge or even diverge. Gradient descent can overshoot the minimum.

When learning rate was 0.01, the training error was approximately 0.06 which is way lower than when the learning rate is 0.2 or 0.1.

Hence, to conclude -> **learning rate should be relatively of smaller magnitude to achieve better training as it significantly impacts the training.**

Also, we have used different activation functions sigmoid, tanh, relu. To conclude about activation functions, we can say that there is nothing like best activation functions. One activation function may be better than other or worse than other. Selection of activation function does not affect what the neural network can learn, it just affects how many data points it needs. However, we observe that, relu works faster and gave better or similar results compared to sigmoid and tanh.

**Case 2:**

Here, we are setting number of iterations = 1000

Learning rate -> 0.05

Here we change the no of nodes in each hidden layer to check which one performs better.

**Case1:**

Number of nodes in hidden layer 1 = 6

Number of nodes in hidden layer 2= 3

**Case2:**

Number of nodes in hidden layer 1 = 3

Number of nodes in hidden layer 2 = 5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Parameter** | **Dataset** | **Activation function** | **Training Error** | **Test Error** |
| Number of | Adult | Sigmoid | 0.1244819297082228 | 0.1243368700265252 |
| Nodes in |  | tanh | 0.4620565318302387 | 0.7801724137931034 |
| hidden layer |  | Relu | 0.1244819297082228 | 0.1243368700265252 |
|  | Car | Sigmoid | 0.1007663069366666 | 0.10997473451664673 |
| h1 = 6 |  | tanh | 1.2192142533123917 | 0.4666666666666667 |
| h2 = 3 |  | Relu | 0.3562635771180304 | 0.3753623188405797 |
|  | Iris | Sigmoid | 0.15979928404232774 | 0.17256449483217806 |
|  |  | tanh | 0.3833312405513268 | 0.29623986685765336 |
|  |  | Relu | 0.7983193277310925 | 0.896551724137931 |
|  | Adult | Sigmoid | 0.1244819297082228 | 0.1243368700265252 |
|  |  | tanh | 0.7437209880636605 | 0.6435624204198227 |
|  |  | Relu | 0.1244819297082228 | 0.1243368700265252 |
| h1 = 3 | Car | Sigmoid | 0.1127500381854472 | 0.12205786850063367 |
| h2 = 5 |  | tanh | 0.4507603186097031 | 1.2782608695652007 |
|  |  | Relu | 0.3562635771180304 | 0.3753623188405797 |
|  | Iris | Sigmoid | 0.15978426080078748 | 0.17254565034579536 |
|  |  | tanh | 0.4369900597032704 | 0.4400404747812484 |
|  |  | Relu | 0.7983193277310925 | 0.896551724137931 |

**Analysis:**

We experimented with number of neurons in hidden layers such as h1=2, h2= 2 OR

h1 = 6, h2 = 3 OR h1=3, h2 = 5. Initially when we started increasing number of nodes in the hidden layer, we observed better model but after some threshold, we observed that test error was greater than training error (Case of overfitting)

**Case 3:**

Here we change the iterations parameter and check how the neural net performs.

Here, we are setting number of iterations = 500, 2000

Learning rate -> 0.05

Number of nodes in hidden layer 1 = 5

Number of nodes in hidden layer 2 = 8

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Parameter** | **Dataset** | **Activation function** | **Training Error** | **Test Error** |
|  | Adult | Sigmoid | 0.12392241379310345 | 0.126657824933687 |
|  |  | tanh | 0.45639920424403185 | 0.7987400530503979 |
|  |  | Relu | 0.12392241379310345 | 0.126657824933687 |
| Number of | Car | Sigmoid | 0.0993134076357164 | 0.11421166592862496 |
| iterations |  | tanh | 0.4503982621288921 | 0.4782608695652174 |
| 500 |  | Relu | 0.35372918175235335 | 0.3869565217391304 |
|  | Iris | Sigmoid | 0.16822834081891025 | 0.1725297426471153 |
|  |  | tanh | 0.5242574966588263 | 1.1744178782783512 |
|  |  | Relu | 0.8277310924369747 | 0.896551724137931 |
|  | Adult | Sigmoid | 0.12392241379310345 | 0.126657824933687 |
|  |  | tanh | 0.8012267904509284 | 0.4469496021220159 |
|  |  | Relu | 0.12392241379310345 | 0.126657824933687 |
|  | Car | Sigmoid | 0.09693355787782656 | 0.11387394359530799 |
| 2000 |  | tanh | 0.4518464880521361 | 0.47246376811594204 |
|  |  | Relu | 0.35372918175235335 | 0.3869565217391304 |
|  | Iris | Sigmoid | 0.16810098273549443 | 0.17243816465919587 |
|  |  | tanh | 0.5294117647058824 | 0.6379310344827587 |
|  |  | Relu | 0.8277310924369747 | 0.896551724137931 |

**Analysis**

When we increased the iterations, we observe that, test error was greater than training error (overfitting) because the model started memorizing values.