CS 6375
Assignment 3:
Inductive Learning

Names of students in your group:
ADRITA DUTTA:axd172930
CHIRAG SHAHI:cxs180005

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.
References:
https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6

Part1- Refer to Document ML-Assignment-3-Part1.pdf

Part2:

Data Set used:IRIS

https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data

Activation	Learning Rate	Training Error	Test Error
sigmoid	0.05	22	8.54
sigmoid	0.10	22.67	8.522
tanh	0.05	35	14.499
tanh	0.10	35	14.499
relu	0.05	91	34
relu	0.10	91	34

Data Set used:CAR

https://archive.ics.uci.edu/ml/machine-learning-databases/car/car.data

Activation	Learning Rate	Training Error	Test Error
sigmoid	0.05	49.15	22.89
sigmoid	0.10	49	22
tanh	0.05	407.499	167.99
tanh	0.10	407	168
relu	0.05	196.5	91.5
relu	0.10	196	91

Report:

We tried the code on 2 different datasets and the following things were observed:

1)Car dataset:

-sigmoid activation function: the error decreases on test data than train data in case of both learning rates and the value of error is almost equal in them. Errors in general are lower than that of relu activation and tanh activation functions.

-tanh activation function: the error decreases on test data than train data in case of both learning rates and the value of error is almost equal in them. Errors in general are higher than that of relu activation and sigmoid activation functions.

-relu activation function: the error decreases on test data than train data in case of both learning rates and the value of error is almost equal in them. Errors in general are lower than that of tanh activation and higher than sigmoid activation functions.

2)Iris dataset:

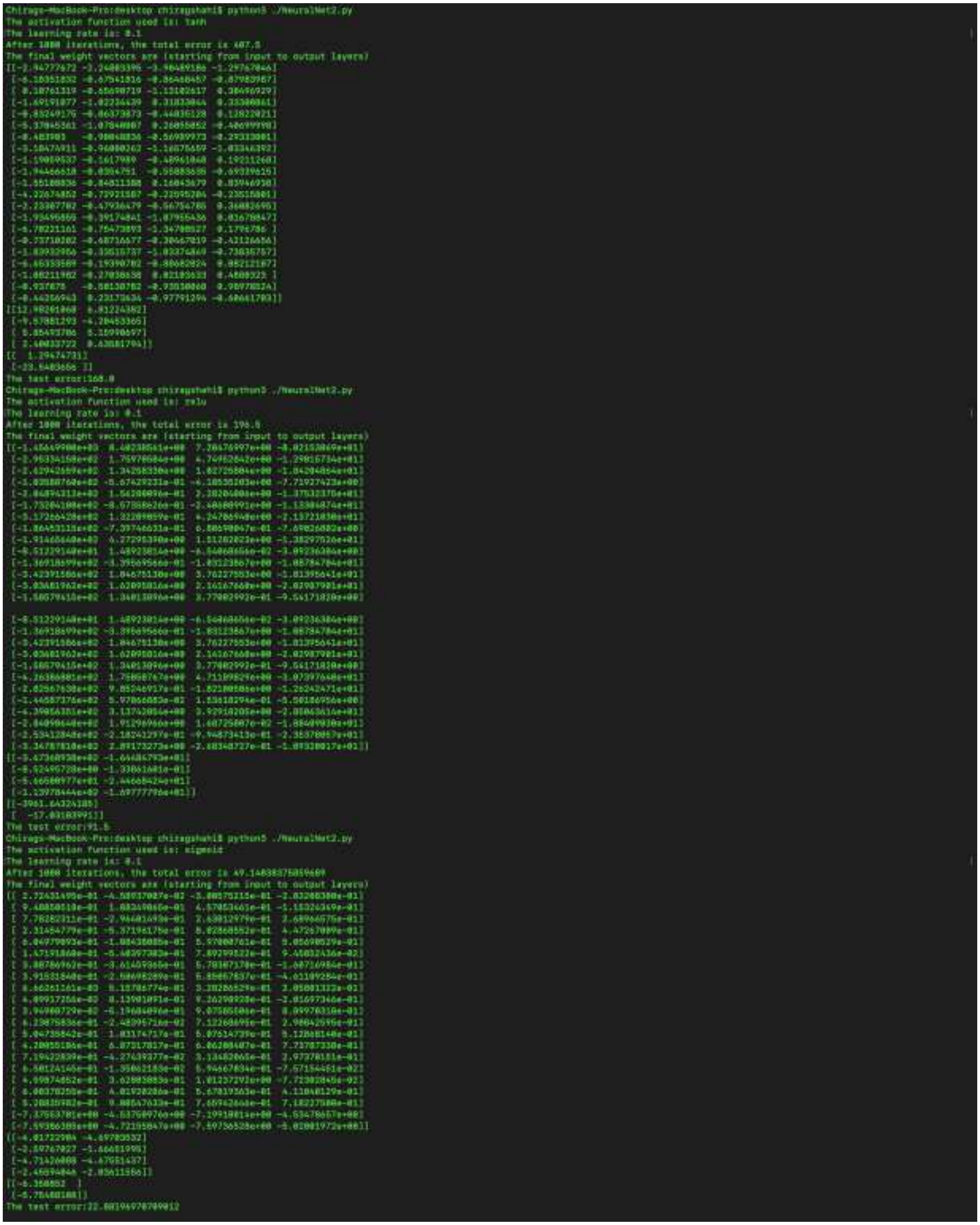
-sigmoid activation function: the error decreases on test data than train data in case of both learning rates and the value of error is almost equal in them. Errors in general are lower than that of relu activation and tanh activation functions.

-tanh activation function: the error decreases on test data than train data in case of both learning rates and the value of error is almost equal in them. Errors in general are lower than that of relu activation and higher than sigmoid activation functions

-relu activation function: the error decreases on test data than train data in case of both learning rates and the value of error is almost equal in them. Errors in general are higher than that of tanh activation and sigmoid activation functions.

In general it is seen that sigmoid activation is performing the best irrespective of learning rate and datasets. Within sigmoid learning rate 0.1(in car dataset) is giving a slightly lower error and learning rate 0.05(in iris dataset) is giving a slightly lower error.

Output:


Chirags-MacBook-Pro:desktop chiragshah1$ python3 ./NeuralNet2.py
The activation function used is: tanh
The learning rate is: 0.1
After 1000 iterations, the total error is 407.5
The final weight vectors are (starting from input to output layers)
The test error:168.0
Chirags-MacBook-Pro:desktop chiragshah1$ python3 ./NeuralNet2.py
The activation function used is: relu
The learning rate is: 0.1
After 1000 iterations, the total error is 196.5
The final weight vectors are (starting from input to output layers)
The test error:91.5
Chirags-MacBook-Pro:desktop chiragshah1$ python3 ./NeuralNet2.py
The activation function used is: sigmoid
The learning rate is: 0.1
The final weight vectors are (starting from input to output layers)

```
Chirags-MacBook-Pro:desktop chiragshah1$ python3 ./NeuralNet2.py
The activation function used is: tanh
The learning rate is: 0.05
After 1000 iterations, the total error is 407.49999920912325
The final weight vectors are (starting from input to output layers)
[[-1.12400702 -1.12821709 -2.08895708 -0.87322281]
 [ 0.89726367 -0.58301386 -0.91757497 -0.55918463]
 [-2.21098895 -0.34136942 -2.38311241  0.37122735]
 [-1.2841097  -0.88410764 -1.28446376  0.41827642]
 [-0.50483023 -0.60140932 -0.40201872  0.20330817]
 [-2.55334422 -0.91752883 -1.63871404 -0.16129136]
 [-0.26416488 -0.69021826 -1.18834872 -0.20051168]
 [-1.09284529 -0.84080976 -0.86875363 -0.90190091]
 [-0.79178507  0.24283509 -1.30632815  0.26935076]
 [-0.86958236  0.11258618 -0.27870999 -0.61003968]
 [-3.1782421A -0.5563926  -2.84944899  0.90614583]
 [-1.49776407 -0.48181366 -0.76350651  0.81182429]
 [ 1.11449092 -0.27798153 -0.34186224  0.50318161]
 [-1.36724463 -0.15086709 -0.67379383  0.11278186]
 [-1.20580056 -0.48397568 -1.47132556  0.50408172]
 [-1.27358455 -0.30238316 -3.6196189  -0.30235419]
 [-2.10360677 -0.11329422 -2.11552233 -0.44888268]
 [-0.88183862  0.07995259 -1.74189289  0.41880846]
 [-1.86257744  0.12239988 -1.87594646  0.60997612]
 [-0.25654005 -0.53284878 -2.11894177  0.9087119 ]
 [-0.47354248  0.27646331 -0.0993257  -0.59253284]]
[[ 6.39267226  4.91490797]
 [-5.01302009  0.07224014]
 [ 2.76674184  2.6240741 ]
 [ 1.38727983 -1.48273387]]
[[  0.64566764]
 [-11.56682873]]
The test error:167.99999967386577
Chirags-MacBook-Pro:desktop chiragshah1$ python3 ./NeuralNet2.py
The activation function used is: relu
The learning rate is: 0.05
After 1000 iterations, the total error is 196.5
The final weight vectors are (starting from input to output layers)
[[-1.33346675e+02 -4.30515137e+02 -1.78347360e+03 -8.54691588e+00]
 [-2.58993801e+01 -5.58653083e+01 -2.46219326e+02 -1.74719990e+00]
 [-2.33535580e+01 -8.12099629e+01 -2.18981437e+02 -1.45607906e+00]
 [-9.23140500e+00 -5.67534847e+01 -2.47954537e+02 -3.16815136e-01]
 [-1.78435878e+01 -6.11638183e+01 -2.62833287e+02 -1.07915635e+00]
 [-1.64879812e+01 -5.58099609e+01 -2.48114195e+02 -1.18007263e+00]
 [-1.86983292e+01 -5.83461628e+01 -2.49930526e+02 -2.28696647e+00]
 [-1.65983124e+01 -4.88125198e+01 -2.12728484e+02 -1.50291090e+00]
 [-1.71727821e+01 -6.27069130e+01 -2.64064363e+02 -1.05733992e+00]
 [-7.96946294e+00 -6.60291142e+01 -2.39936202e+02 -8.03220882e-01]
 [-1.23073283e+01 -5.59000752e+01 -2.45854138e+02 -1.21911867e-01]
 [-3.04026222e+01 -5.03594381e+01 -2.53578234e+02 -1.77627004e+00]
 [-2.70230783e+01 -5.30777078e+01 -2.34687001e+02 -1.01246652e+00]
 [-1.41532581e+01 -8.69395381e+01 -3.60834291e+02 -7.45376651e-01]

 [-7.96946294e+00 -6.60291142e+01 -2.39936302e+02 -8.03220882e-01]
 [-1.23073283e+01 -5.59000752e+01 -2.45854138e+02 -1.21911867e-01]
 [-3.04026222e+01 -5.03594381e+01 -2.53578234e+02 -1.77627004e+00]
 [-2.70230783e+01 -5.35777078e+01 -2.34687001e+02 -1.01246652e+00]
 [-1.41532581e+01 -8.69395381e+01 -3.60834291e+02 -7.45376651e-01]
 [-3.93076957e+01 -6.75646860e+01 -3.03071867e+02 -2.67939926e+00]
 [-2.46438221e+01 -6.91835958e+01 -3.11680018e+02 -1.38563226e+00]
 [-1.38321848e+01 -6.47603669e+01 -2.80467962e+02 -1.02347781e+00]
 [-3.96161642e+01 -7.42208278e+01 -3.24026761e+02 -2.43447644e+00]
 [-2.48642280e+01 -8.33844218e+01 -3.62267665e+02 -1.21568861e+00]
 [-2.63735843e+01 -6.58696675e+01 -3.00358417e+02 -1.84594871e+00]
 [-3.42660470e+01 -9.64698628e+01 -4.02757819e+02 -1.62012422e+00]]
[[-6.59327492e+01 -8.48791610e-01]
 [-1.69860769e+00 -3.07479909e+03]
 [-7.85767207e+00 -1.59206113e+04]
 [-1.90638914e+01 -3.80378454e-01]]
[[   -486.96209379]
 [-14211.8898179 2]]
The test error:91.5
Chirags-MacBook-Pro:desktop chiragshah1$ python3 ./NeuralNet2.py
The activation function used is: sigmoid
The learning rate is: 0.05
After 1000 iterations, the total error is 49.16969685190721
The final weight vectors are (starting from input to output layers)
[[ 0.16293681 -0.31331719 -0.08133377  0.31373702]
 [ 0.66178765 -0.28616768  0.35060909 -0.27641526]
 [ 0.47189669 -0.28638647  0.3981996   0.4731989 ]
 [ 0.4328774  -0.63041619  0.67555064  0.43684922]
 [ 0.52939509 -0.4504111   0.55950005  0.30482028]
 [ 0.1470185  -0.78637405  0.64437692 -0.07547168]
 [ 0.23779692 -0.48394638  0.47726277 -0.07649701]
 [ 0.10536992 -0.66074967  0.4613732  -0.6938857 ]
 [ 0.07635866  0.46760716  0.58095088  0.35527006]
 [-0.05426745  0.38059044  0.48081456 -0.35214628]
 [ 0.179343   -0.54106381  0.93588386  0.87256099]
 [ 0.34424715 -0.28220572  0.667463    0.2883332 ]
 [ 0.37128166 -0.01566253  0.66170899  0.60389917]
 [ 0.34274984  0.17798337  0.59871166  0.25396977]
 [ 0.61447616 -0.16086327  0.41013526  0.60260953]
 [ 0.46268829 -0.2077462   0.37761711 -0.02847389]
 [ 0.26963256  0.03440938  0.68379836 -0.36093184]
 [ 0.49660032  0.34915835  0.69206219  0.54794571]
 [ 0.45490633  0.45500962  0.8623769   0.70260685]
 [-4.31404235 -1.31409168 -5.9102096   1.42473016]
 [-4.57649861 -0.8570684  -6.11202358  0.32350636]]
[[-4.41030833 -4.29944638]
 [-0.86312587 -0.75912486]
 [-7.83227098 -5.63137528]
 [ 2.04449263 -0.28634746]]
[[-6.23381137]
 [-2.92680752]]
The test error:22.09004190752 3466
```

```
Chirags-MacBook-Pro:desktop chiragshahi$ python3 ./NeuralNet2.py
The activation function used is: tanh
The learning rate is: 0.05
After 1000 iterations, the total error is 35.000835521747845
The final weight vectors are (starting from input to output layers)
[[ 0.2467247  -0.96835682  0.86141751  0.38109465]
 [ 0.99464018 -0.65531852 -0.72375551  0.86483377]
 [ 0.39365314 -0.86799944  0.51120428  0.50730435]
 [ 0.846056    0.42304955 -0.75155087 -0.9603844 ]]
[[-3.4731573  -0.93295887]
 [ 2.01801613  0.70961244]
 [-2.44354133  0.11127508]
 [-1.84144994 -0.74121994]]
[[ 1.42086496]
 [-6.13895389]]
The test error:14.499384328441753
Chirags-MacBook-Pro:desktop chiragshahi$ python3 ./NeuralNet2.py
The activation function used is: relu
The learning rate is: 0.05
After 1000 iterations, the total error is 91.0
The final weight vectors are (starting from input to output layers)
[[ 0.24672023 -0.96835751  0.85887447  0.38179384]
 [ 0.9946457  -0.65531898 -0.7257285   0.86519093]
 [ 0.39363632 -0.86799965  0.51092611  0.50775238]
 [ 0.84604907  0.42304952 -0.75145808 -0.96023973]]
[[-0.94757803 -0.94338702]
 [-0.50757786  0.7200559 ]
 [ 0.07766213  0.10564396]
 [ 0.68406178 -0.75165337]]
[[-0.44163264]
 [ 0.17151854]]
The test error:34.0
Chirags-MacBook-Pro:desktop chiragshahi$ python3 ./NeuralNet2.py
The activation function used is: sigmoid
The learning rate is: 0.05
After 1000 iterations, the total error is 22.729642293437802
The final weight vectors are (starting from input to output layers)
[[ 0.53420614 -0.97119267 -0.25808061  0.52137422]
 [ 1.19217492 -0.65727128 -2.56629689  0.99369656]
 [ 0.47009117 -0.86877206  2.81386156  0.46507584]
 [ 0.85460571  0.42295242  0.55323906 -0.99522055]]
[[-3.96038714 -3.25401956]
 [-0.51119317  0.71666128]
 [ 7.32501764  7.41266891]
 [-2.32545959 -3.06051371]]
[[4.78138512]
 [4.89995224]]
The test error:8.547300029862438
```

```
Chirags-MacBook-Pro:desktop chiragshahi$ python3 ./NeuralNet2.py
The activation function used is: tanh
The learning rate is: 0.1
After 1000 iterations, the total error is 35.00000053981819
The final weight vectors are (starting from input to output layers)
[[ 0.24699168 -0.96835699  0.85221524  0.38049272]
 [ 0.99481088 -0.65531864 -0.73005402  0.8645644 ]
 [ 0.39374596 -0.86799948  0.50826243  0.50682812]
 [ 0.846073    0.42304955 -0.75213504 -0.96055309]]
[[-5.99873657 -0.85949923]
 [ 4.54361014  0.63616757]
 [-4.96474479  0.18937091]
 [-4.36696167 -0.6677678 ]]
[[  3.49510419]
 [-12.24293114]]
The test error:14.499999630726565
Chirags-MacBook-Pro:desktop chiragshahi$ python3 ./NeuralNet2.py
The activation function used is: relu
The learning rate is: 0.1
After 1000 iterations, the total error is 91.0
The final weight vectors are (starting from input to output layers)
[[ 0.24672023 -0.96835751  0.85887447  0.38179384]
 [ 0.9946457  -0.65531898 -0.7257285   0.86519093]
 [ 0.39363632 -0.86799965  0.51092611  0.50775238]
 [ 0.84604907  0.42304952 -0.75145808 -0.96023973]]
[[-0.94757803 -0.94338702]
 [-0.50757786  0.7200559 ]
 [ 0.07766213  0.10564396]
 [ 0.68406178 -0.75165337]]
[[-0.44163264]
 [ 0.17151854]]
The test error:34.0
Chirags-MacBook-Pro:desktop chiragshahi$ python3 ./NeuralNet2.py
The activation function used is: sigmoid
The learning rate is: 0.1
After 1000 iterations, the total error is 22.67460499549553
The final weight vectors are (starting from input to output layers)
[[ 0.5598197  -0.97160722 -0.28026902  0.54515281]
 [ 1.20968671 -0.65755627 -2.64762857  1.01825807]
 [ 0.4770879  -0.86888669  2.93269478  0.45132052]
 [ 0.8554333   0.4229374   0.62457542 -1.00439518]]
[[-4.36725979 -3.66261771]
 [-0.51161043  0.7162419 ]
 [ 8.20246842  8.29734944]
 [-2.73169625 -3.46850501]]
[[5.20138466]
 [5.35588772]]
The test error:8.522515561261335
```