# 大作业(姓名：————，学号：————)

### 已布置作业更新（按照题目，更新后的代码，更新后代码的运行截图的顺序，给出每一道更新题目的更新）（直接更新到该题后面，按照project题目，代码，测试结果（运行结果截图！）的格式给出，当然，只有觉得需要更新的同学才需要完成这部分作业

Python basics部分每一道题的测试样例：

generating test cases for: SortEmbedded

[[68, 88, 80, 49, 11, 95, 53, 99, 54], [99, 15, 35]]

[[25], [10], [28, 5, 32]]

[[84, 5, 92, 60, 68, 8], [48, 66], [72, 81, 3, 95, 20, 99, 64, 54]]

[[31, 49, 48, 40, 39, 45, 19, 65, 57], [54, 75, 18], [14, 31, 18, 54, 69, 51, 92, 92, 84], [79, 1, 14, 58, 77, 4], [4, 85, 80, 11, 88, 61, 4]]

[[53], [72, 39, 87, 95, 99, 43], [96, 77, 34, 59, 43]]

[[56], [69, 63, 51, 69, 36, 24], 22, [38, 59, 40, 95, 52, 31, 67], [25, 56, 18, 46], [90, 46, 36, 17, 13, 87], [68, 93, 96], [11, 9, 90, 73, 31]]

[[19, 94, 93, 45, 58, 1], [55, 93, 67, 17, 6, 52, 91, 35, 68], [78, 85, 2, 15, 34, 21, 41], [94, 59, 29, 83, 83, 8, 95, 3, 64], [3, 88], [91, 94]]

[[31, 68, 29, 38, 83, 18], [26, 67]]

[[39, 23, 56, 58, 88, 52], 54, 52, [43, 62, 38, 81], [3, 63, 82, 17, 71, 54], [97, 48, 49], [54, 35], [12, 50, 30, 37], [27, 53, 37, 74]]

[[93, 69, 84, 72, 62, 94], [37, 43, 14], [83], [54, 94, 15, 39, 87, 38, 1, 60, 42], [77, 67, 14, 88], [69, 55], 88, [3, 88], [37, 36, 27, 20, 5, 40, 88, 75]]

generating test cases for: MatTranspose

[[0, 1, 2, 3, 4, 5, 6, 7, 8], [9, 10, 11, 12, 13, 14, 15, 16, 17], [18, 19, 20, 21, 22, 23, 24, 25, 26], [27, 28, 29, 30, 31, 32, 33, 34, 35], [36, 37, 38, 39, 40, 41, 42, 43, 44], [45, 46, 47, 48, 49, 50, 51, 52, 53], [54, 55, 56, 57, 58, 59, 60, 61, 62], [63, 64, 65, 66, 67, 68, 69, 70, 71], [72, 73, 74, 75, 76, 77, 78, 79, 80]]

[[0, 1, 2, 3, 4, 5, 6, 7], [8, 9, 10, 11, 12, 13, 14, 15], [16, 17, 18, 19, 20, 21, 22, 23], [24, 25, 26, 27, 28, 29, 30, 31]]

[[0], [1], [2], [3], [4], [5], [6], [7]]

[[0, 1, 2], [3, 4, 5], [6, 7, 8], [9, 10, 11], [12, 13, 14]]

[[0, 1, 2], [3, 4, 5], [6, 7, 8], [9, 10, 11], [12, 13, 14], [15, 16, 17]]

[[0, 1, 2], [3, 4, 5], [6, 7, 8]]

[[0], [1]]

[[0, 1, 2, 3, 4], [5, 6, 7, 8, 9], [10, 11, 12, 13, 14], [15, 16, 17, 18, 19], [20, 21, 22, 23, 24], [25, 26, 27, 28, 29], [30, 31, 32, 33, 34], [35, 36, 37, 38, 39], [40, 41, 42, 43, 44]]

[[0, 1, 2, 3, 4, 5, 6]]

[[0, 1, 2, 3, 4], [5, 6, 7, 8, 9], [10, 11, 12, 13, 14]]

generating test cases for: SumOfItems

[88, 11, '99', '99', 25, '61', 10, '3', 5, '14', 5, '68', 48]

['56', '81', 95, '64', 57, '31']

['39', '65', '54', '73', 31]

[51, 92, 79, 58, '4', '85']

['61', '49', 37, '39', 99, '96', '77', '43', '56', '54', '51', 24]

[22, 7, 59, 95, 67, 56, '90', 17, '83', 68, '21', '90', '13', 94]

[58, 74, '93', 6, 35, 79, '85', 34, '25', '29', 8, '64']

['18', '94', 54, '29', '18', '82']

['16', 12, '39', 56, '52', '54', 43, 17, 56, 43]

['81', 3, '82', '54', 48, '50', '35', '12', '37', 27, '37', 86, 69]

generating test cases for: FlattenList

[88, 11, [54, 67, 99, 15, 35], [16, 61, 59], [94, 87, 3, 28], 32, [84, 5, 92, 60, 68, 8, 50, 48], [56, 72, 81, 3, 95, 20, 99, 64, 54], [31, 49, 48, 40, 39, 45, 19, 65, 57], 75, 14, 18, 51]

[[54, 79, 1, 14, 58, 77], [4, 85, 80, 11, 88, 61, 4, 49, 53], [72, 39, 87, 95, 99, 43, 85, 96], [34, 59, 43, 23, 1, 56, 99], [63, 51, 69, 36, 24, 10, 49], [55, 7, 38, 59], 95, 67, 56, [90, 46, 36, 17, 13, 87, 83, 68], [11, 9, 90, 73, 31, 13, 19], [45, 58, 1, 74, 55, 93, 67], [91, 35, 68, 95, 79, 55]]

[15, [41, 25, 94, 59, 29, 83, 83], [64, 3, 88, 18, 91], 97, 31, 38, 18, [26, 67, 16, 12], [23, 56, 58, 88, 52, 49, 54, 18, 66], 45, [55, 56, 84, 43, 62, 38], 3, [17, 71, 54, 99], 49, [35, 4, 12, 50, 30, 37, 94, 68], 53, [93, 69, 84, 72, 62, 94, 37, 43], [33, 39, 41, 54, 94]]

[87, 60, [45, 20, 77, 67], [69, 55, 1, 88], 66, [40, 37, 36, 27, 20, 5, 40], 80, [14, 34, 16, 18, 89, 24, 60], [78, 48, 46], 32, 81, 61, [20, 21, 4, 75, 60, 12, 49, 7], [77, 59, 73, 1], [31, 83, 28, 71, 54, 40], 46, [47, 61, 81, 70, 81], 82, [77, 90, 6, 67, 65, 46]]

[[68, 80, 83, 85, 80, 14, 76, 50], 82, 39, [28, 22, 1, 74, 2, 37], [87, 27, 87], [4, 29, 36, 32, 20, 53, 42, 79, 51], 50, 78, 3, 62, [3, 56, 97, 64, 71, 26], [57, 59, 83, 77, 61, 18], 95, 67, [8, 63, 2, 86], 31]

[97, [24, 43, 8, 78, 48], 88, [86, 2, 95], 42, [90, 60, 10], [63, 43, 72, 50, 15], [48, 58, 27], [88, 11, 69, 40, 99, 1], [59, 42, 7, 38, 85, 81, 77], [38, 68, 48, 33, 33, 23], [87, 39, 2], 92, 45, 15, 96, [95, 15, 43, 93]]

[[30, 82, 51, 70, 8, 5, 88], 87, 53, 12, [10, 12, 31, 70, 66, 52], 38, [37, 40, 83, 4], 42, [15, 19, 19], [85, 74, 16, 78, 26, 48]]

[60, [46, 79, 61, 45, 9, 94, 8, 54, 37], [5, 13, 41, 35, 94], [13, 86, 24, 68, 61], 64, [92, 27, 28], 35, 59, 13, [8, 52, 83, 42, 30, 76]]

[10, [56, 3, 21, 47, 59, 8], 66, [79, 68, 9], 96, [83, 28, 12], 48, 20, 36, 47, [93, 14, 23, 88, 84, 76]]

[[50, 17, 45, 36, 59, 21, 97, 38, 82], 65, [61, 17, 72], 89, [43, 67, 60, 81], [54, 69, 34, 96, 86, 70], 93]

generating test cases for: StringToNumber

"9039"

"1350"

"18.646"

"45.273"

"2340"

"6769"

"30.742"

"5993"

"31.135"

"3883"

generating test cases for: StarTriangle

10

26

5

41

25

17

12

32

36

4

generating test cases for: SumOfMultiplesOf3And5

620

892

935

971

955

179

450

966

153

678

generating test cases for: FizzBuzz

18

34

13

49

33

25

20

40

44

12

generating test cases for: SumOfDigits

5740

7036

8139

179

450

4249

8006

5746

902

4476

generating test cases for: SumOfNumbersInString

8139 kkkkk vvvvvvv 5746 yyyyy qqqqqqqq 6330 eeeeeeeee eeeeeeee 2983 3867 ccccc cccccccc

qqqqqq sssssssss 7174 4469 rrrrrrr 588 vvvvvvvv 4209 ddddddd ppppppppp

ddddddddd 1897 3386 uuuuuuu 8017 gggggg 7524 8518 sssssssss 4961 3668 3610 uuuuuuu tttttttt 2726

6517 nnnnnn wwwwwwwww ddddddddd kkkkkkkkk mmmmm 1473 vvvvvvvvv

794 9705 962 5297 9330 iiiiiiiii sssssss rrrrrrr nnnnnn kkkkk 3239 5302 nnnnnnn cccccc

ggggggggg 7067 qqqqqqqq vvvvvvv 3470 4195 wwwwwwww wwwww vvvvv 7526 qqqqqq aaaaaaa 3861 ddddddd eeeeee 7981 7770

2472 9121 kkkkkkkkk aaaaaaa vvvvvv 138 2276 fffff 2213 1719 cccccc 9608 3198 ppppppp nnnnnnnnn 7413 yyyyyyy

4499 qqqqqq xxxxx rrrrrrrrr 760 lllllll 2026 mmmmmm aaaaaaaa oooooooo 2474 aaaaaaaa sssss

8925 ppppp mmmmmmmm 2854 2079 pppppppp 2479 1865 7563 ttttt 8165 aaaaaa wwwwwww dddddddd uuuuuuu ooooooo 9548 ddddddddd 1697

6299 4066 sssss 6261 2127 ccccccccc hhhhhh bbbbb qqqqqq ddddddddd ggggggg 3958 7360 llllll 2112 ppppp fffffff

generating test cases for: MaxOfThree

5641 6937 8040

80 351 4150

7907 5647 803

4377 9872 1341

7739 9724 5770

6231 2435 5788

1901 4994 3214

6902 5844 9221

2884 4232 3768

4860 3409 3

generating test cases for: DecToBin

38508

14247

12219

179

16404

16834

14360

10182

20633

13990

generating test cases for: SortList

[793, 836, 872, 856, 80, 945, 907, 351, 949, 867, 54, 579, 739] none

[753, 281, 656, 229, 876, 317] desc

[650, 94, 87, 387, 668, 877, 901, 416, 898, 142, 758, 724, 613, 5, 604, 956] none

[432, 706, 318, 783, 696, 968] asc

[890, 618, 479, 404, 995, 64, 182] desc

[370, 927, 945, 304, 680, 878, 807, 173, 275, 193, 953, 931, 438, 715, 274, 585, 526, 619, 31] asc

[947, 489, 348, 476, 980, 694, 847, 1, 14] asc

[132, 583, 644, 213, 720, 395, 216, 503, 496, 957, 644, 874, 177, 181, 812] none

[39, 215, 95, 739, 171, 853, 96, 627, 750, 632, 77, 802] none

[407, 257, 763, 696, 227, 916, 822, 197, 447, 819, 453, 420, 408, 766, 522] none

generating test cases for: ReverseDigits

38508

56188

79783

77755

65715

81940

63470

82370

14360

75718

generating test cases for: MergeSorted

[793, 793, 793, 793, 793, 793, 793, 793, 793, 793, 793, 793, 793] [80, 351, 856, 867, 872, 907, 945, 949]

[579, 579, 579, 579, 579, 579, 579, 579, 579, 579] [229, 281, 527, 656, 753, 803, 876]

[571, 571, 571, 571, 571, 571, 571, 571, 571, 571, 571, 571, 571, 571, 571, 571, 571] [87, 94, 387, 508, 650]

[877, 877, 877, 877, 877, 877, 877, 877, 877, 877, 877, 877, 877, 877, 877, 877] [5, 142, 416, 604, 613, 724, 758, 898, 956]

[136, 136, 136, 136, 136, 136, 136, 136] [318, 432, 696, 706, 783, 968]

[995, 945, 927, 890, 825, 618, 479, 404, 370, 337, 304, 284, 239, 182, 64, 3] [953, 931, 878, 807, 715, 585, 526, 438, 275, 274, 193, 173]

[980, 947, 847, 694, 489, 476, 461, 348, 186, 69, 54, 31, 18, 14, 1] [957, 874, 812, 720, 644, 644, 583, 503, 496, 395, 216, 213, 181, 177, 132]

[853, 840, 774, 739, 215, 171, 96, 95, 39] [939, 802, 750, 632, 315, 103, 77]

[916, 822, 819, 763, 696, 453, 447, 420, 257, 227, 197] [945, 871, 790, 766, 678, 629, 522, 497, 439, 410, 263, 123]

[921, 890, 820, 814, 735, 620, 579, 486, 440, 430, 296, 287, 274, 244, 90] [855, 622, 525, 467, 224, 221, 68, 17]

generating test cases for: RemoveDuplicate

[793, 793, 836, 856, 856, 856, 856, 856, 856, 856, 856, 80, 80, 80, 80, 907, 907, 907, 907, 907, 907, 907, 907, 907, 351, 351, 351, 351, 351, 351, 949, 949, 867, 867, 867, 867, 54, 579, 579, 579, 579, 579, 739, 739, 739]

[5, 5, 5, 5, 5, 5, 5, 604, 604, 604, 604, 604, 604, 604, 836, 836, 136, 136, 136, 136, 136, 136, 136, 136, 136, 50, 50, 50, 50, 50, 50, 50, 50, 50, 432, 432, 432, 706, 706, 318, 318, 318, 318, 318]

[825, 239, 239, 239, 239, 239, 239, 239, 239, 370, 927, 927, 927, 927, 927, 945, 945, 945, 945, 304, 304, 680, 680, 680, 680, 680, 680, 680, 680, 878, 878, 878, 807, 807, 807, 807, 807, 275, 275, 275, 275, 275, 275, 275, 275, 275, 193, 193, 193, 953, 953, 953, 953, 953, 953, 931, 931, 931, 438, 438, 438, 438, 715, 715, 715, 715, 715, 715, 715]

[496, 496, 496, 496, 496, 496, 957, 957, 644, 644, 644, 644, 874, 874, 177, 177, 177, 177, 177, 177, 177, 181, 812, 812, 812, 812, 812, 812, 37, 37, 37, 37, 37, 37, 840, 840, 840, 840, 840, 840, 840, 39, 39]

[822, 822, 822, 822, 822, 822, 822, 822, 822, 819, 819, 819, 819, 819, 453, 453, 453, 453, 453, 453, 453, 453, 453, 420, 420, 420, 420, 420, 420, 408, 408, 408, 408, 408, 408, 766, 766, 766, 766, 766]

[123, 123, 123, 123, 123, 123, 123, 123, 629, 629, 629, 629, 629, 629, 629, 187, 296, 296, 296, 296, 296, 620, 620, 620, 620, 620, 620, 620, 620, 620, 735, 735, 735, 287, 287, 287, 287, 287, 287, 890, 890, 579, 579, 579, 244, 244, 244, 244]

[777, 777, 777, 777, 777, 777, 777, 777, 474, 474, 474, 474, 474, 474, 713, 713, 909, 909, 909, 909, 909, 659, 659, 659, 350, 350, 888, 888, 888, 605, 605, 605, 605, 605, 605, 557, 557, 557, 557, 246, 58, 58, 58, 384, 769, 769, 769, 769, 586, 586, 586, 586, 586, 586, 586, 586]

[443, 443, 443, 443, 443, 797, 797, 797, 979, 979, 979, 979, 979, 723, 723, 648, 648, 648, 648, 648, 648, 479, 515, 576, 576, 576, 576, 576, 882, 882, 882, 882, 882, 882, 882, 882, 882, 131, 131, 216, 216, 216, 216, 216, 146, 146, 146, 146, 146, 146]

[791, 824, 88, 88, 88, 692, 692, 692, 692, 692, 692, 945, 945, 945, 945, 945, 945, 945, 755, 755, 755, 507, 507, 507, 507, 507, 507, 694, 694, 694, 694]

[958, 958, 806, 806, 806, 806, 806, 806, 806, 806, 806, 625, 721, 721, 721, 721, 721, 114, 114, 868, 868, 868, 259, 259, 259, 259, 259, 259, 319, 868, 868, 868, 868, 466, 466, 466, 529, 71, 71, 71, 71, 310, 310, 310, 310]

generating test cases for: CountHeadsAndLegs

388 1436

782 2382

522 1302

926 2740

886 2744

507 1220

392 1336

404 1488

620 1900

471 1498

generating test cases for: LeapYear

1908

2006

1948

1952

1968

1936

1924

2010

1916

1994

### 在lecture 5-1的ppt中，给出了神经网络训练时的权重更新公式： （输出层到隐含层）以及 和 （隐含层到输入层），不过按照ppt中的神经网络结果，在输入和隐含层都包含了一个bias结点，就是其输入固定为1的结点，这两个结点在训练时的更新公式是没有的，仿照ppt中的推导过程，推导这两个结点的权重更新公式（包含具体的过程）：（直接写在题后）

### 在neural.py文件（已经在大作业的伴随文件中给出）中补充class NeuralNetwork中\_\_init\_\_，TrainATime，RunATime，Train，Run，Save，Load这些成员函数，大部分代码可以参考lecture 5-1 ppt中的，不过ppt中是单独定义的函数，而这里要求这些函数是class NeuralNetwork的成员。同时在ppt中不包含bias结点的backpropagation更新，而这里在训练部分必须包含这些代码，并且这些代码，因为是独立实现的会成为这部分的得分重点。Load和Save只要了解了pickle的用法就非常简单。所有完成的代码都应该体现在最后提交的repo中的同名py文件中

### 针对随作业提供的字符数据集，（0-9，A-Z）不同的字符保存在不同的目录中，通过python程序把数据转换并存入两个二进制文件中。其中一个保存模式，另一个文件保存这些模式的标记（或分类信息）。模式文件的格式如下表：

|  |  |  |  |
| --- | --- | --- | --- |
| Offset | type | Value | Description |
| 0 | Byte\*4 | ‘IMAG’ | Magic number |
| 4 | 32 bit integer | ? | Number of patterns in the file |
| 8 | 32 bit integer | 100 | Rows of each pattern |
| 12 | 32 bit integer | 75 | Cols of each pattern |
| 16 | Unsigned byte | ?? | pixel |
| … | Unsigned byte | ?? | pixel |

从上面的表格可以看成，格式开始一个4字节的magic number以便标识该数据文件就是我们要的样本文件，后面是4字节的整数，保存文件中的样本数，接下来分别是模式的行数和列数，再后面是每一个模式的像素值，按从上到下，从左到右，一个模式接连一个模式的顺序保存在文件中。

标记文件的格式如下：

|  |  |  |  |
| --- | --- | --- | --- |
| offset | type | value | description |
| 0 | Byte\*4 | ‘LABL’ | Magic number |
| 4 | 32 bit integer | ? | Number of labels (should match patterns!) |
| 5 | Unsigned byte | ?? | Label |
| … | Unsigned byte | ?? | label |

对第一个类数字0，标记是0， 对于第二个类数字1， 标记是1， 对第一个字母类’A’，标记是10，其他依此类推。

在伴随的PatternIo.py文件中，完成其中给出的三个过程。它们分别实现把子目录中的模式图像读出来并保存在一个上面给出的二进制文件中；把二进制文件中的模式分为训练样本和测试样本，并单独保存；以及从一个二进制文件中读入其中的样本（比如训练样本，测试样本）。

### 按照75%的样本用来训练，25%的样本用来测试的方法，把样本分开，使用前一部分编写的代码，来读取这些样本（留意在样本中，并不是每一个类的样本数目都是一样多的！）。针对其中的数字样本，0-9. 使用训练样本的均值矢量作为每个模式的原型矢量，并使用最小距离分类器来对测试样本进行分类，给出分类结果，并计算正确率(关于最小距离分类器， 可以参考lecture 5-1中的相关内容)。这部分代码，统一写在一个名称为MinDistClassifier.py文件中。这一部分是大作业的一个重点，应该尽可能多的描述整个过程（如何实现的），结果原型矢量的图像，分类正确的例子和分类错误的例子等等。我们会根据完成情况，包括对过程的描述和结果来评定成绩（直接在该题后面给出报告）

### 使用前面完成的神经网络(neural.py中的class)，同样按照75%做训练样本，25%做测试样本，来训练这个神经网络，在这一步，只关心字母，不关心数字，即应该有A到Z共25类（留意其中的字母O因为和数字0无法区分，因此样本中并不包含这样一个类）。在网络收敛后，把网络保存到文件中。生成同样的网络，并读入训练好的系数（同样使用前面已经实现的class 中的方法），然后对测试样本进行测试，计算正确率。在这一部分，我们不规定特征的提取方法，包括最简单的直接使用每个像素作为一个特征，或者对样本下采样后一个采样点作为一个特征，或者计算统计矩后作为特征，还有PCA等等，都可以。把训练的过程py代码和测试过程中py代码分别写入两个过程（同一py文件， 该py文件命名为nn.py），可以通过python nn.py +参数的方法来运行上述两个过程。留意！这一部分应该是大作业的另一个重点，在这部分尽量多的描述清楚整个过程，对应的代码，以及结果。我们会根据完成情况来打分。（直接在该题后面给出报告）