|  |
| --- |
| [..\03.Introduction To Nerual Network](../03.Introduction To Nerual Network) |
| [音频\03-01.m4a](音频/03-01.m4a) |
| 关键词 |
|  |
| Linear fit 线性拟合 linear regression  简单介绍一下 |
| 5.Linear to Logistic Regression |
| 6.Logistic Regression Quiz |
| 8.Neural Networks |
| 9.Perceptron |
| 10.AND Perceptron Quiz      import pandas as pd  # TODO: Set weight1, weight2, and bias  weight1 = ~~0.0~~ 1  weight2 = ~~0.0~~ 1  bias = ~~0.0~~ 0.5  # DON'T CHANGE ANYTHING BELOW  # Inputs and outputs  test\_inputs = [(0, 0), (0, 1), (1, 0), (1, 1)]  correct\_outputs = [False, False, False, True]  outputs = []  # Generate and check output  for test\_input, correct\_output in zip(test\_inputs, correct\_outputs):  linear\_combination = weight1 \* test\_input[0] + weight2 \* test\_input[1] + bias  output = int(linear\_combination >= 0)  is\_correct\_string = 'Yes' if output == correct\_output else 'No'  outputs.append([test\_input[0], test\_input[1], linear\_combination, output, is\_correct\_string])  # Print output  num\_wrong = len([output[4] for output in outputs if output[4] == 'No'])  output\_frame = pd.DataFrame(outputs, columns=['Input 1', ' Input 2', ' Linear Combination', ' Activation Output', ' Is Correct'])  if not num\_wrong:  print('Nice! You got it all correct.\n')  else:  print('You got {} wrong. Keep trying!\n'.format(num\_wrong))  print(output\_frame.to\_string(index=False)) |
| 11.OR & NOT Perceptron Quiz  AND, OR, or NOT operations. Let's do one more, XOR  1）    import pandas as pd  # TODO: Set weight1, weight2, and bias  weight1 = ~~0.0~~ 1  weight2 = ~~0.0~~ 1  bias = ~~0.0~~ -0.5  # DON'T CHANGE ANYTHING BELOW  # Inputs and outputs  test\_inputs = [(0, 0), (0, 1), (1, 0), (1, 1)]  correct\_outputs = [False, False, False, True]  outputs = []  # Generate and check output  for test\_input, correct\_output in zip(test\_inputs, correct\_outputs):  linear\_combination = weight1 \* test\_input[0] + weight2 \* test\_input[1] + bias  output = int(linear\_combination >= 0)  is\_correct\_string = 'Yes' if output == correct\_output else 'No'  outputs.append([test\_input[0], test\_input[1], linear\_combination, output, is\_correct\_string])  # Print output  num\_wrong = len([output[4] for output in outputs if output[4] == 'No'])  output\_frame = pd.DataFrame(outputs, columns=['Input 1', ' Input 2', ' Linear Combination', ' Activation Output', ' Is Correct'])  if not num\_wrong:  print('Nice! You got it all correct.\n')  else:  print('You got {} wrong. Keep trying!\n'.format(num\_wrong))  print(output\_frame.to\_string(index=False))  2）    import pandas as pd  # TODO: Set weight1, weight2, and bias  weight1 = ~~0.0~~  -1  weight2 = 0.0  bias = ~~0.0~~  0.5  # DON'T CHANGE ANYTHING BELOW  # Inputs and outputs  test\_inputs = [(0, 0), (0, 1), (1, 0), (1, 1)]  correct\_outputs = [True, False, True, False]  outputs = []  # Generate and check output  for test\_input, correct\_output in zip(test\_inputs, correct\_outputs):  linear\_combination = weight1 \* test\_input[0] + weight2 \* test\_input[1] + bias  output = int(linear\_combination >= 0)  is\_correct\_string = 'Yes' if output == correct\_output else 'No'  outputs.append([test\_input[0], test\_input[1], linear\_combination, output, is\_correct\_string])  # Print output  num\_wrong = len([output[4] for output in outputs if output[4] == 'No'])  output\_frame = pd.DataFrame(outputs, columns=['Input 1', ' Input 2', ' Linear Combination', ' Activation Output', ' Is Correct'])  if not num\_wrong:  print('Nice! You got it all correct.\n')  else:  print('You got {} wrong. Keep trying!\n'.format(num\_wrong))  print(output\_frame.to\_string(index=False)) |
| 12.XOR Perceptron Quiz |
| 13.The Simplest Neural Network  1) step activation function    2)    3) f（h）可以是任何函数  比如    4）        5）    import numpy as np  def sigmoid(x):  # TODO: Implement sigmoid function  return 1/(1 + np.exp(-x))  inputs = np.array([0.7, -0.3])  weights = np.array([0.1, 0.8])  bias = -0.1  # TODO: Calculate the output  output = sigmoid(np.dot(weights, inputs) + bias)  print('Output:')  print(output)  6）weights与inputs相乘，用np.dot |
| 14.Gradient Descent  1) 误差的平方和      2）gradient descent        3) |
| 15.Gradient Descent The Math  看视频吧，全是干货。以下为偏导的推导过程。目的是要更新weights |
| 16.Gradient Descent The Code    import numpy as np  def sigmoid(x):  """  Calculate sigmoid  """  return 1/(1+np.exp(-x))  learnrate = 0.5  x = np.array([1, 2])  y = np.array(0.5)  # Initial weights  w = np.array([0.5, -0.5])  # Calculate one gradient descent step for each weight  # TODO: Calculate output of neural network  nn\_output = sigmoid(np.dot(x, w))  # TODO: Calculate error of neural network  error = y - nn\_output  # TODO: Calculate change in weights  del\_w = learnrate \* error \* nn\_output \* (1 - nn\_output) \* x  print('Neural Network output:')  print(nn\_output)  print('Amount of Error:')  print(error)  print('Change in Weights:')  print(del\_w) |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |