pa2_solution

April 8, 2020

[ ]:
```
""" Find the optimal weights for a sigmoid perceptron-like classifier
    Based on this tutorial:
    https://nbviewer.jupyter.org/github/Christof93/perceptron/blob/master/
 ↪perceptron_algorithm.ipynb """

import re
import codecs
import numpy as np


# Sigmoid model
def sigmoid(a, x):
    return 1/(1 + np.exp(-a*x, dtype=np.float128))


# Read input vectors
# WAR = 0
# PEACE = 1
def read_input():
    train_list = []
    find_counts = re.compile('.*?\t(.*)\t\\|\t(WAR|PEACE)', flags=re.UNICODE)
    f = codecs.open('pa2_input.txt', 'r', 'utf-8')
    for line in f:
        if 'Word' not in line:
            counts = re.search(find_counts, line)
            if counts:
                counts_arr = counts.group(1).split('\t')
                counts_arr = [int(i) for i in counts_arr]

                if counts.group(2) == 'WAR':
                    label = 0
                else:
                    label = 1

                current_row = (counts_arr, label)
                train_list.append(current_row)
```

```
    return train_list


# Initialize weights: length equals to the number of base words
def init_weights(dataset):
    dim = len(dataset[0][0])
    weights = [0] * dim
    return weights


# In this example we don't need too many iterations
def calc_weights(dataset, weights):
    for it in range(100):
        if it == 1 or it == 99:
            print("\nresults after", str(it + 1), "iterations:")
        for inp, desired_out in dataset:
            result = sigmoid(2, np.dot(inp, weights) - 0.5)
            if it == 1 or it == 99:
                print("true result:", desired_out, "output:", round(result, 3))
            error = desired_out - result
            if abs(error) > 0.0:
                for i, val in enumerate(inp):
                    # 0.2 is the learning rate which we can choose freely␣
 ↪between 0 and 1
                    weights[i] += val * error * 0.2
    return weights


# Write updated weights to the txt file
def write_weights(weights):
    f = codecs.open('weights_pa2.txt', 'w', 'utf-8')
    print("\nUpdated weights:")
    for w in weights:
        print(round(w, 3))
        f.write(str(round(w, 3)) + '\n')
    print('Weights written into weights_pa2.txt')


if __name__ == '__main__':
    training = read_input()
    initial_weights = init_weights(training)
    updated_weights = calc_weights(training, initial_weights)
    write_weights(updated_weights)
```