Pattern– HW#3

The main aim of the assignment is to make you familiar with the linear classifiers, specifically with the perceptron algorithm. Please solve the problems individually and cheating will be harshly punished. To generate the ***inputs*** you should read images from train folder with 128x128x3 form and convert the images into vector format. Resize all image from 128x128x3 to **1x49152**. The label of each image should be saved in a target vector, called **t**.

You should read test images and predict the class label for each one.

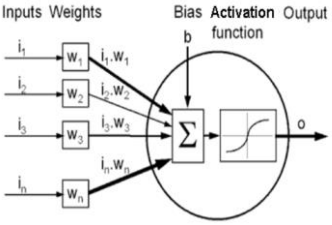

Fig. 1. General demonstration of perceptron algorithm [1].

Fig. 1 shows the general demonstration of perceptron algorithm. In the last head of activation, we will use the **SWISH activation function**.

We will use the gradient descent algorithm in case of training our perceptron algorithm.

`Feed Forward and Feed Backward in Perceptron`

Each training sample is a pair of the form $\langle \vec{x}, t \rangle$ where $\vec{x}$ is the vector of input values, and t is target output value. η is *learning rate*.

- Initialize each weight, w_i to some small value.
- Until the all samples classified correctly or iteration condition met. Do
  - Initialize each $\triangle w_i$ $(called\ gradient)$ to zero
  - For each training samples, Do
    - Input the instance $\vec{x}$ to the unit and compute the output o

      $g(\mathrm{x}) = Swish(\mathrm{x})$

      $g(\mathrm{y})' = dSwish(\mathrm{y})$

    - For each linear unit weight w_i Do

      $\triangle w_i = \eta (t - y) g'(y) x_i,$

      $(t - y)$ *referes to error*

      $g'(y)$ *refers to* derivative of *activation* function

      $g'(y) =$ *dSwish function*

  - For each linear unit weight w_i, Do

    $w_i = w_i + \triangle w_i$

[1] http://aass.oru.se/~lilien/ml/seminars/2007_02_01b-Janecek-Perceptron.pdf

```
def sigmoidp(x):
    sig = 1 / (1+np.exp(-x))
    return sig

def Swish(x):
    swish = x * sigmoidp(x)
    return swish

def dSwish(x):
    swish_der = (Swish(x)) + (sigmoidp(x)*(1-(Swish(x))))
    return swish_der
```

Feature	Description	Number of features
Mean	Average values of all pixels in a studied area	3
Entropy	Measure of energy behind all pixels	3
Standard deviation	Factor of variation for all pixels	3
Skewness	Measure of third moment of all pixels and refers to symmetric behavior of data	3
Moment	Refers to second moment of all pixels, called variance.	3
Mean FFT	Average all pixels after the Fourier transform	3
Percentile	Indicates the 50% percentile of all pixels	3
Median	Middle value of sorted pixels	3
Brightness Index	((R * R + G * G + B * B)/3)	1
Saturation Index	(R − B)/(R + B)	1
Hue Index	(2 * R − G − B)/(G − B)	1
Coloration Index	(R − G)/(R + G)	1
Redness Index	(R * R)/(B * G * G * G)	1
	Total number of features	29

Fig. 1. Features.

Step 1:

Implement a *trainPerceptron(inputs, t, weights, rho, iterNo):* function in PYTHON, in order to train the linear classifer, so called perceptron algorithm.

inputs **:** Feature vectors belonging to classes

- Assume that n is the image size in vector form as **1x49152**.
- You are expecting to extract statistical features (given in Fig.1) from this image vectors.
- Note that, for bias, you should add 1 to input vector.
- If the input vector size is n, then after adding bias, the input vector size will be n+1. Also the size of weights is n+1.
- Then the weights becomes **1x29**+1 → input becomes **1x30** array.
- Don't forget to shuffle your inputs before the training procedure. Recall from the class, we have touched about shuffle of training data. You can use sklearn library with following snippet code.

```
from sklearn.utils import shuffle
inputs, t = shuffle(inputs, t)
```

***t* :** labels of classes (encoded with 1 bits) .

For example: flamingo→ 0

pizza→ 1

https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html

weights: Initial weights for the linear discriminant function. You should initial weights random. The vector dimension of weights and input vector is same. Both of them are n+1.

***rho* :** learning rate is 0.0001

***iterNo*:** refers to number of iterations in perceptron is 1000

After the training procedure completed, you should save weights for testing purpose. In case of testing stage load the weights vector.

To save and load weights, you can use the numpy.

```
np.save('weights.npy',  weights) # save
weights  =  np.load('data.npy') # load
```

Once the

Step 2:

To implement your `testPerceptron(sample_test, weights)` function, you should return predicted value.

In this stage, you will test your perceptron algorithm

First of all, load weights which had been trained during the training stage.

Secondly, read a test image from test folder and send the feed forward process of perceptron algorithm.

Finally, the perceptron will give an output about label of test sample. Calculate the accuracy with sklearn accuracy_score function.

- You will graded over 100% if you have implemented the given train and test functions. Otherwise, you will be graded over 75%.
- Also, you have to add comment to your codes.
- In training part, you have to show the feed-forward process and feed-backward process by inserting comments.

Submit the Assignment

Send your py code and pdf as zip format.

Ex: No_Name_Surname_HW#.zip

Hint

You can look the implementation perceptron on notes of the course.