ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO THỰC HÀNH

Môn: Ứng dụng Xử lý ảnh số & video số

Lab 02: THỰC HÀNH VỚI PYTORCH

Giảng viên hướng dẫn:
Lý Quốc Ngọc
Nguyễn Mạnh Hùng

Sinh viên thực hiện:

Nguyễn Quí Em
MSSV: 1712399

Ngày 30 tháng 06 năm 2020

Các thay đổi trong mạng

1. Ở file ffnn.py

- **Class FFNeuralNetwork:** Sửa đổi tham số input_size (mặc định: 3), hidden_size (mặc định: 10) và activation_func (hàm activation – mặc định là sigmoid) để người dùng có thể linh hoạt lựa chọn.

```
# Feed Forward Neural Network class
class FFNeuralNetwork(nn.Module):
    # initialization function
    def __init__(self, input_size = 3, hidden_size = 10, activation_func = 'sigmoid'):
```

- **Hàm activation:** Ngoài hàm sigmoid thì còn cài đặt thêm hàm tanh và đạo hàm tương ứng derivative_tanh.

```
# tanh activation
def tanh(s):
    return (torch.exp(s)-torch.exp(-s))/(torch.exp(s) + torch.exp(-s))

# derivative of tanh
def tanh_derivative(s):
    return 1-tanh(s)**2
```

- Tuỳ thuộc vào giá trị tham số activation_func được truyền vào khi tạo class FFNeutalNetwork mà lựa chọn gọi hàm activation theo yêu cầu.

```
# activation function using sigmoid
def activation(self, z):
    if self.z_activation_func =='sigmoid':
        self.z_activation = sigmoid(z)
    elif self.z_activation_func == 'tanh':
        self.z_activation = tanh(z)
    return self.z_activation

# derivative of activation function
def activation_derivative(self, z):
    if self.z_activation_func == 'sigmoid':
        self.z_activation_derivative = sigmoid_derivative(z)
    elif self.z_activation_func == 'tanh':
        self.z_activation_derivative = tanh_derivative(z)
    return self.z_activation_derivative
```

2. Ở file test_ffnn.py

- Sử dụng bộ dữ liệu doanh thu theo số tiền quảng cáo được lưu trong file "advertising.csv" kèm theo. Dữ liệu bao gồm 5 cột. Các cột "TV", "Radio", "Newspaper" thể hiện giá của các các hình thức quảng cáo qua TV, Radio, Newspaper. Cột "Sales" là giá trị doanh thu tương ứng.

TV	Radio	Newspaper	Sales
230.1	37.8	69.2	22.1
44.5	39.3	45.1	10.4
17.2	45.9	69.3	12
151.5	41.3	58.5	16.5

- Đọc file csv này, lấy 20 dòng đầu của 3 cột đầu tiên (TV, Radio, Newspaper) làm tập input X. Lấy 20 dòng đầu của cột cuối cùng (Sales) làm label y.

```
# sample input and output value for training
X = pd.read_csv('advertising.csv', usecols=[0,1,2]) # Read columns 'TV', 'Radio' and 'Newspaper' in dataset
X = X.values.tolist() # Convert dataframe to list
X = torch.tensor(X[:20], dtype=torch.float) # Convert to tensor pytorch, get 20 head rows

y = pd.read_csv('advertising.csv', usecols=[3]) # Read columns 'Sales' in dataset
y = y.values.tolist() # Convert dataframe to list
y = torch.tensor(y[:20], dtype=torch.float) # Convert to tensor pytorch, get 20 head rows
```

- Lấy kích thước input X, gọi class mạng mà truyền vào các tham số phù hợp. Lựa chọn số hidden units là 20, hàm activation sử dụng là hàm tanh.
Cho mạng học 1000 lần với learning rate là 0.1

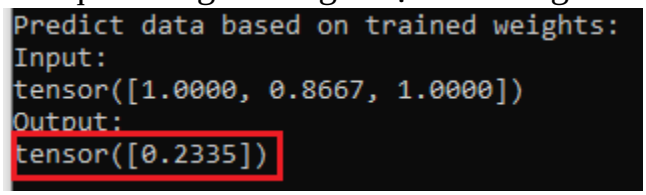
```
input_shape = X.shape[1] # Input shape
# create new object of implemented class
NN = ffnn.FFNeuralNetwork(input_size = input_shape, hidden_size = 20, activation_func = 'tanh')

# trains the NN 1,000 times
for i in range(1000):
    # print mean sum squared loss
    print("#" + str(i) + " Loss: " + str(torch.mean((y - NN(X)) ** 2).detach().item()))
    # training with learning rate = 0.1
    NN.train(X, y, 0.1)
```

- Sau khi load các trọng số của mô hình. Ta thử predict kết quả xem thế nào. Sử dụng 1 bộ giá trị gần giống trong file csv

TV	Radio	Newspaper	Sales
230.1	37.8	69.2	22.1
44.5	39.3	45.1	10.4
17.2	45.9	69.3	12
151.5	41.3	58.5	16.5

```
# sample input x for predicting
x_predict = torch.tensor(([45, 39, 45]), dtype=torch.float)
```

- Ta mong muốn kết quả dự đoán sẽ gần với giá trị Sales = 10.4 (như trong bộ dữ liệu). Dưới đây là kết quả predict. Kết quả này phải được nhân với giá trị lớn nhất trong bộ ba giá trị truyền vào thì mới ra giá trị Sales có thể so sánh (vì dữ liệu truyền vào đã được chuẩn hoá nên kết quả cần được xử lý lại).
  Ta có: 0.2335*45 = 10.5075
  Kết quả trên gần với giá trị 10.4 trong file dữ liệu => model khá tốt.



3. Nhận xét chung

Tuỳ vào các lần chạy thử nghiệm, thay đổi số units ở hidden layer, số lần train,…và quan sát loss mà ta có "kinh nghiệm" chọn ra các giá trị tham số mạng cho phù hợp.

Hết