Candace Edwards

ICS 635: Homework 5

Part 1: [Notebook Link]

Part 2: Q4: [Notebook Link]

```
from scipy.signal import convolve2d
import numpy as np
```

Question 1. Final Exam Practice Questions

1. B: the others are ReLu variations or Softplus activations
2. B: False - Training on too many epochs is leads to over fitting, and is computationally inefficient.
3. C: An MLP can also learn non-linear features for complex images
4. B: 2500 `see calculation in code below`
5. D : 996.0 x 996.0 x 50 `see calculation in code below`
6. D: All of the above, because each method has the potential to generalize the model / reduce overfitting.
7. B: Fast RNN does not use network pruning
8. B. False - Multi-Network systems can use more than two NNs.
9. D: All are benefits
10. C: Momentum in gradient descent is for optimization not generalization
11. B: Exploitation: Exploitatin is described as using known information to maximize reward [source]. Higher learning rate promotes exploitation to learn quickly, instead of gathering more information slowly (exploration)
12. A: Discount factor of 0 , no consideration to future rewards

```
#Q1.4
weight_count =  (5*50) + (50*30) +(30*20)+(20*10)
weight_count

     2550
```

```
#Q1.5
#Output height = (Input height + padding height top + padding height bottom - kernel height) / (stride height) + 1.
#depth = kernel count
output_dim = (1000 + 0 + 0 - 5) / (1) + 1
depth = 50
print(f'Dimensions: {output_dim} x {output_dim} x {depth}')

     Dimensions: 996.0 x 996.0 x 50
```

Question 2: Neural Network Output

- Answer: `p = 0.5, see code below`

```
def relu_activation(x):
  if x<0:
    return 0
  return x


# #relu_activation test #status: works as expected
# test_val = -1
# expected = 0
# print(f'Output:{relu_activation(test_val)} , Expected: {expected}')

# test_val = 8
# expected = 8

# print(f'Output:{relu_activation(test_val)} , Expected: {expected}')
```

```
def calc_inputs(input_1, weight_1, input_2, weight_2):
  return np.sum([(input_1 * weight_1),(input_2*weight_2)])


#calc_inputs() tests #status: works as expected
# x_1 = 5
# w_1_x_1 = -1
# w_2_x_1 = 0

# x_2 = 4
# w_3_x_2 = 1
# w_4_x_2=  2

# expected = -1
# print(f'Output:{calc_inputs(x_1,w_1_x_1,x_2,w_3_x_2)} , Expected: {expected}')


# expected = 8
# print(f'Output:{calc_inputs(x_1,w_2_x_1,x_2,w_4_x_2)} , Expected: {expected}')


def sigmoid(x):
  return 1/(1+np.exp(-x))



x_1 = 4
w_1_x_1 = 1
w_2_x_1 = -2

x_2 = -2
w_3_x_2 = 3
w_4_x_2=  -1

#HL: 1
reLu_in_1 = calc_inputs(x_1,w_1_x_1,x_2,w_3_x_2)
reLu_in_2 = calc_inputs(x_1,w_2_x_1,x_2,w_4_x_2)

#pass to ReLu, results become new inputs for next layer
x_1 = relu_activation(reLu_in_1)
w_1_x_1 = -2
w_2_x_1 = -2

x_2 = relu_activation(reLu_in_2)
w_3_x_2 = 1
w_4_x_2=  2


#HL2

reLu_in_1 = calc_inputs(x_1,w_1_x_1,x_2,w_3_x_2)
reLu_in_2 = calc_inputs(x_1,w_2_x_1,x_2,w_4_x_2)

x_1 = relu_activation(reLu_in_1)
pre_sig_w_1 = 5

x_2 = relu_activation(reLu_in_2)
pre_sig_w_2 = 4


#SIG
output= sigmoid(calc_inputs(x_1,pre_sig_w_1,x_2,pre_sig_w_2))
output
```

```
     0.5
```

Question 3: Convolutional Neural Network Output

- Answer: `p = 0.9784374743299705`, see code below`

```
#4x4 Input Image
input_image = np.array([[210,0,0,0],
                        [255,0,0,0],
                        [251,0,0,0],
                         [250,242,247,230]
                         ])
#3x3 Kernel
kernal_1 =np.array([[2,-2,0],
                     [1,0,1],
                     [0,0,1]])

#3X3 Kernal
kernal_2 = np.array([[0,1,0],
                     [-1,0,-1],
                     [-1,2,0]])


#convolutions with kernal 1 and 2
#source: https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.convolve2d.html
feature_map_1 = convolve2d(input_image,kernal_1,mode='valid')
feature_map_2 = convolve2d(input_image,kernal_2,mode='valid')


print(feature_map_1.shape)
print(feature_map_1)
```

```
(2, 2)
[[465   0]
 [516 -34]]
```

```
print(feature_map_2.shape)
print(feature_map_2)
```

```
(2, 2)
[[-255    0]
 [  -9  247]]
```

```
#2x2 max on a 2x2 feature map = max of feature map
max_pool_fmap_1 = np.max(feature_map_1)
max_pool_fmap_2 = np.max(feature_map_2)

print(max_pool_fmap_1, max_pool_fmap_2)
```

```
516 247
```

```
#max pool * weights
weights_1 =1
weights_2=-1
reLu_in_1 = calc_inputs(max_pool_fmap_1,weights_1,max_pool_fmap_2,weights_1)
reLu_in_2 = calc_inputs(max_pool_fmap_1,weights_2,max_pool_fmap_2,weights_1)

#ReLU activation
x_1 = relu_activation(reLu_in_1)
pre_sig_w_1 = 0.005

x_2 = relu_activation(reLu_in_2)
pre_sig_w_2 = -0.8

#SIG
output= sigmoid(calc_inputs(x_1,pre_sig_w_1,x_2,pre_sig_w_2))
output
```

```
0.9784374743299705
```

✓ 0s completed at 11:26 AM ● ×