



PART 2

다층 퍼셉트론(MLP)

1장. 다층 퍼셉트론의 기본 구조

딥러닝 & 강화학습 담당
이재화 강사





이 장에서 다룰 내용

1. 다층 퍼셉트론 신경망 구조와 은닉 계층
2. 비선형 활성화 함수와 ReLU 함수
3. 민스키의 XOR 문제와 비선형 활성화 함수의 필요성
4. 다층 퍼셉트론 신경망을 지원하는 함수 구현
5. Part 1에서 다뤘던 세 문제를 다층 퍼셉트론으로 풀어보기



4.1 다층 퍼셉트론을 위한 재정의 함수 탐색



1. 파라미터 초기화 함수

```
def init_model():  
    global weight, bias, input_cnt, output_cnt  
    weight = np.random.normal(RND_MEAN, RND_STD, [input_cnt, output_cnt])  
    bias = np.zeros([output_cnt])
```

2. 신경망 연산 함수

```
def forward_neuralnet(x):  
    global weight, bias  
    output = np.matmul(x, weight) + bias  
    return output, x
```

3. 역전파 연산 함수

```
def backprop_neuralnet(G_output, x):  
    global weight, bias  
    g_output_w = x.transpose()  
  
    G_w = np.matmul(g_output_w, G_output)  
    G_b = np.sum(G_output, axis=0)  
  
    weight -= LEARNING_RATE * G_w  
    bias -= LEARNING_RATE * G_b
```

“단층 퍼셉트론에서
다층 퍼셉트론으로!”



4.2 은닉 계층을 위한 파라미터 생성 함수 정의



```
def init_model_hidden1():  
  
    global pm_output, pm_hidden, input_cnt, output_cnt, hidden_cnt  
  
    pm_hidden = alloc_param_pair([input_cnt, hidden_cnt])  
  
    pm_output = alloc_param_pair([hidden_cnt, output_cnt])  
  
  
def alloc_param_pair(shape):  
  
    weight = np.random.normal(RND_MEAN, RND_STD, shape)  
  
    bias = np.zeros(shape[-1])  
  
    return {'w':weight, 'b':bias}
```



4.3 은닉 계층을 위한 신경망 연산 함수 정의



```
def forward_neuralnet_hidden1(x):  
  
    global pm_output, pm_hidden  
  
    hidden = relu(np.matmul(x, pm_hidden['w']) + pm_hidden['b'])  
  
    output = np.matmul(hidden, pm_output['w']) + pm_output['b']  
  
    return output, [x, hidden]  
  
def relu(x):  
  
    return np.maximum(x, 0)
```



4.4 은닉 계층을 위한 역전파 함수 정의(1/3)



```
def backprop_neuralnet_hidden1(G_output, aux):  
    global pm_output, pm_hidden  
  
    x, hidden = aux  
  
    g_output_w_out = hidden.transpose()  
    G_w_out = np.matmul(g_output_w_out, G_output)  
    G_b_out = np.sum(G_output, axis=0)  
  
    g_output_hidden = pm_output['w'].transpose()  
    G_hidden = np.matmul(G_output, g_output_hidden)  
  
    pm_output['w'] -= LEARNING_RATE * G_w_out  
    pm_output['b'] -= LEARNING_RATE * G_b_out  
  
    G_hidden = G_hidden * relu_derv(hidden)  
  
    g_hidden_w_hid = x.transpose()  
    G_w_hid = np.matmul(g_hidden_w_hid, G_hidden)  
    G_b_hid = np.sum(G_hidden, axis=0)  
  
    pm_hidden['w'] -= LEARNING_RATE * G_w_hid  
    pm_hidden['b'] -= LEARNING_RATE * G_b_hid
```



4.4 은닉 계층을 위한 역전파 함수 정의(2/3)



```
def backprop_neuralnet_hidden1(G_output, aux):
    global pm_output, pm_hidden

    x, hidden = aux

    g_output_w_out = hidden.transpose()
    G_w_out = np.matmul(g_output_w_out, G_output)
    G_b_out = np.sum(G_output, axis=0)

    g_output_hidden = pm_output['w'].transpose()
    G_hidden = np.matmul(G_output, g_output_hidden)

    pm_output['w'] -= LEARNING_RATE * G_w_out
    pm_output['b'] -= LEARNING_RATE * G_b_out

    G_hidden = G_hidden * relu_derv(hidden)

    g_hidden_w_hid = x.transpose()
    G_w_hid = np.matmul(g_hidden_w_hid, G_hidden)
    G_b_hid = np.sum(G_hidden, axis=0)

    pm_hidden['w'] -= LEARNING_RATE * G_w_hid
    pm_hidden['b'] -= LEARNING_RATE * G_b_hid
```



4.4 은닉 계층을 위한 역전파 함수 정의(3/3)

```
def relu_derv(y):  
    return np.sign(y)
```



4.5 가변적 은닉 계층을 위한 파라미터 생성 함수 정의



```
def init_model_hiddens():  
  
    global pm_output, pm_hiddens, input_cnt, output_cnt, hidden_config  
  
    pm_hiddens = []  
  
    prev_cnt = input_cnt  
  
    for hidden_cnt in hidden_config:  
  
        pm_hiddens.append(alloc_param_pair([prev_cnt, hidden_cnt]))  
  
        prev_cnt = hidden_cnt  
  
    pm_output = alloc_param_pair([prev_cnt, output_cnt])
```



4.6 가변적 은닉 계층을 위한 신경망 연산 함수 정의



```
def forward_neuralnet_hiddens(x):  
  
    global pm_output, pm_hiddens  
  
    hidden = x  
  
    hiddens = [x]  
  
    for pm_hidden in pm_hiddens:  
  
        hidden = relu(np.matmul(hidden, pm_hidden['w']) + pm_hidden['b'])  
  
        hiddens.append(hidden)  
  
    output = np.matmul(hidden, pm_output['w']) + pm_output['b']
```



4.7 가변적 은닉 계층을 위한 역전파 함수 정의(1/2)



```
def backprop_neuralnet_hiddens(G_output, aux):
    global pm_output, pm_hiddens

    hiddens = aux

    g_output_w_out = hiddens[-1].transpose()
    G_w_out = np.matmul(g_output_w_out, G_output)
    G_b_out = np.sum(G_output, axis=0)

    g_output_hidden = pm_output['w'].transpose()
    G_hidden = np.matmul(G_output, g_output_hidden)

    pm_output['w'] -= LEARNING_RATE * G_w_out
    pm_output['b'] -= LEARNING_RATE * G_b_out

    for n in reversed(range(len(pm_hiddens))):
        ...
```



4.7 가변적 은닉 계층을 위한 역전파 함수 정의(1/2)



```
def backprop_neuralnet_hiddens(G_output, aux):  
    ...  
    for n in reversed(range(len(pm_hiddens))):  
        G_hidden = G_hidden * relu_derv(hiddens[n+1])  
  
        g_hidden_w_hid = hiddens[n].transpose()  
        G_w_hid = np.matmul(g_hidden_w_hid, G_hidden)  
        G_b_hid = np.sum(G_hidden, axis=0)  
  
        g_hidden_hidden = pm_hiddens[n]['w'].transpose()  
        G_hidden = np.matmul(G_hidden, g_hidden_hidden)  
  
        pm_hiddens[n]['w'] -= LEARNING_RATE * G_w_hid  
        pm_hiddens[n]['b'] -= LEARNING_RATE * G_b_hid
```



4.8 실행 보조 함수 정의_1



```
global hidden_config
```

```
def init_model():  
    if hidden_config is not None:  
        print('은닉 계층 {}개를 갖는 다층 퍼셉트론이 작동되었습니다.'. \  
              format(len(hidden_config)))  
        init_model_hiddens()
```

```
def forward_neuralnet(x):  
    if hidden_config is not None:  
        return forward_neuralnet_hiddens(x)
```

```
def backprop_neuralnet(G_output, hiddens):  
    if hidden_config is not None:  
        backprop_neuralnet_hiddens(G_output, hiddens)
```



4.9 실행 보조 함수 정의_2



```
def set_hidden(info):  
  
    global hidden_cnt, hidden_config  
  
    if isinstance(info, int):  
  
        hidden_cnt = info  
  
        hidden_config = None  
  
    else:  
  
        hidden_config = info
```



5.1 Part 1에서 다뤘던 세 문제를 다층 퍼셉트론으로 풀어보기



```
%run ../AI_CODE/AI_abalone.ipynb
```

```
%run ../AI_CODE/mlp.ipynb
```

```
set_hidden([10,6,2])
```

```
abalone_exec()
```

```
%run ../AI_CODE/AI_pulsar.ipynb
```

```
%run ../AI_CODE/mlp.ipynb
```

```
set_hidden([24,12])
```

```
pulsar_exec()
```

```
%run ../AI_CODE/AI_steel.ipynb
```

```
%run ../AI_CODE/mlp.ipynb
```

```
set_hidden([36,12,6])
```

```
steel_exec()
```

