기계학습의기초및응용 HW2 보고서

과목: 기계학습의기초및응용
교수명: 이규행 교수님
학과: 모바일시스템공학과
학번: 32224020
이름: 전은지

Index

I. Introduction

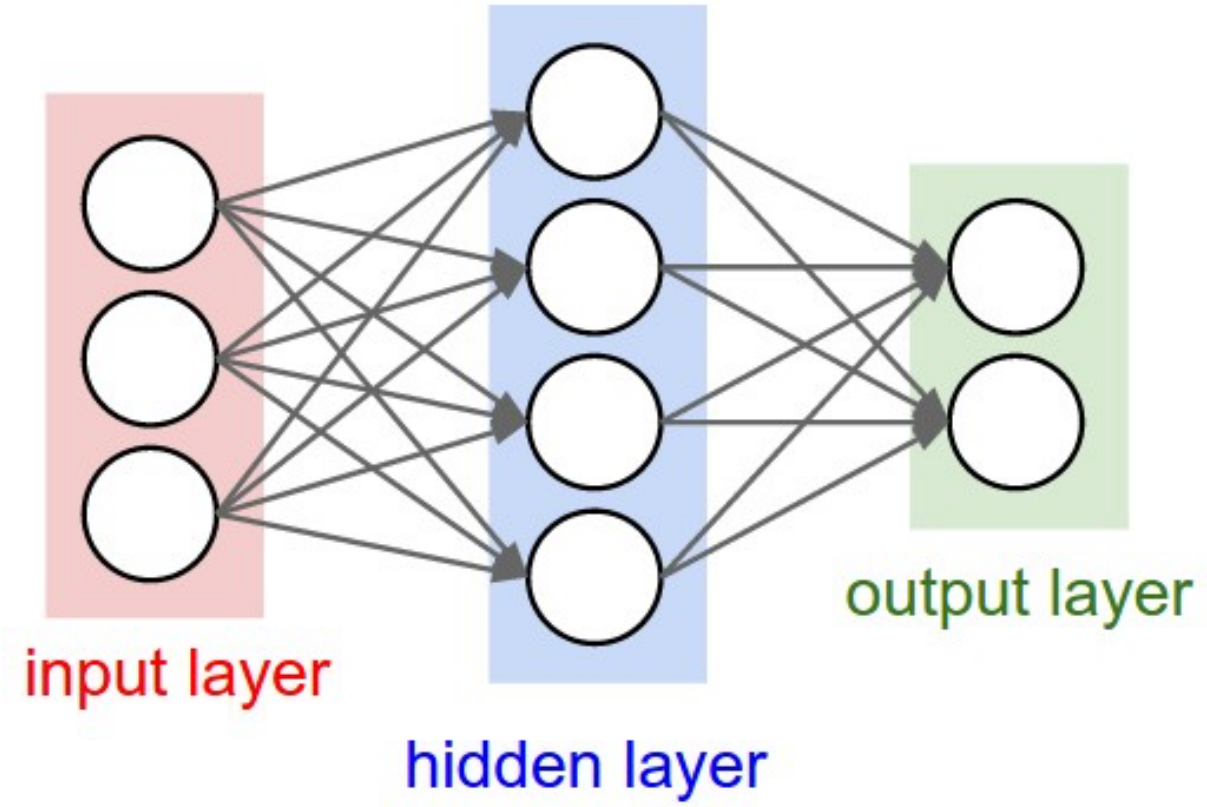
이번 과제의 목표는 주어진 데이터를 설명할 수 있는 함수를 근사화 하기 위한 신경망을 설계하는 것이다. 데이터 집합에는 두 개의 값 집합이 있다. 주어진 파일의 첫번째 열을 가리키는 x와 두번째 열 y(y=f(x))이다. 자신만의 모델을 설계하고 학습한 후 최적의 파라미터 세트를 찾아야한다. 또한, 모델과 가중치를 모두 보여야한다.

II. Design

1. Main Idea

1) MLP 신경망

MLP란 여러 개의 퍼셉트론 뉴런을 어려 층으로 쌓은 다층 신경망 구조로 입력층과 출력층 사이에 하나 이상의 은닉층을 가지고 있는 신경망이다.



단층 퍼셉트론은 AND연산에 대하여만 학습이 가능하기 때문에 비선형적으로 분리되는 데이터에 대해서는 제대로 된 학습이 불가능하다는 한계가 존재한다. 이를 극복하기 위해 입력층과 출력층 사이에 하나 이상의 중간층을 두어 XOR연산과 같은 비선형 데이터에 대한 학습이 가능한 다층 퍼셉트론이 고안되었다.

2) 활성화 함수

다층 퍼셉트론은 비선형 데이터에 대한 학습이 가능하도록 활성화 함수를 사용한다. Sigmoid, tanh, ReLU 등의 활성화 함수가 존재하고 MLP에서는 ReLU가 기본적으로 사용

된다.

3) 입력층
데이터셋이 입력되는 층인 입력층은 데이터셋의 특성에 따라 입력층의 노드 수가 결정된다. 단순히 값만 들어오는 층이기 때문에 어떠한 연산도 이루어지지 않기 때문에 MLP의 층 수를 계산할 때 입력층은 포함되지 않는다.

4) 은닉층
은닉층은 입력층과 출력층 사이에 있는 층으로 계산의 결과를 사용자가 볼 수 없기 때문에 은닉층이라 불린다. 이 층에서 입력된 데이터를 가중치와 편향을 이용하여 계산한다.

5) 출력층
가장 마지막에 위치한 층으로 은닉층에서 연산을 마친 값이 출력된다.

6) 순전파
입력층에서 입력된 신호가 은닉층의 연산을 거쳐 출력층에서 값을 내보내는 과정이다. 입력층에서 전달받은 데이터를 가중치 및 편향과 연산한 후 활성함수를 통과시켜 예측값을 출력한다. 이후 예측값과 실제값의 차이를 계산하는 것까지 순전파의 과정이다.

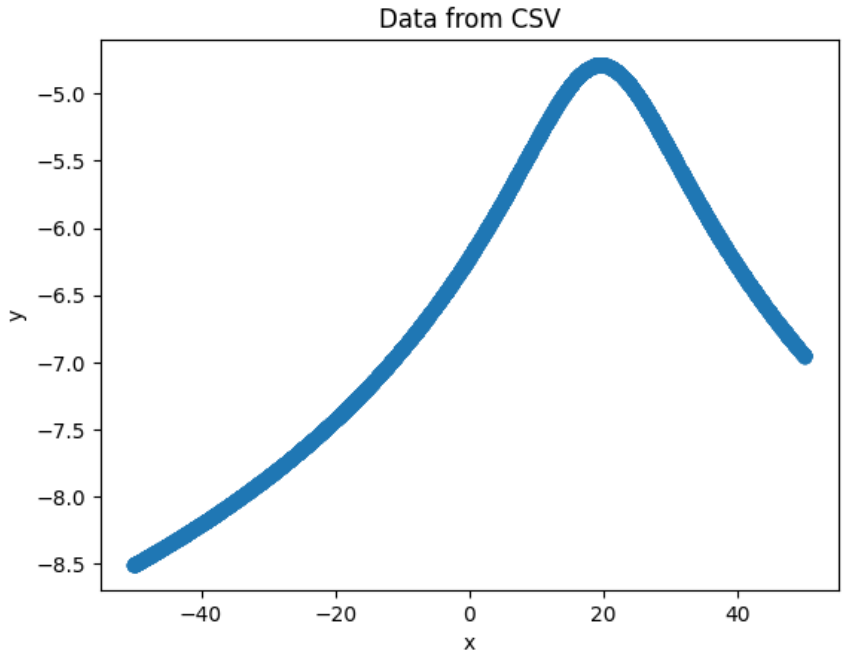
7) 역전파
역전파는 매 반복마다 손실 정보를 출력층에서 입력층까지 전달하여 순전파를 통해 계산된 오류를 기반으로 가중치를 업데이트하는 과정이다. 가중치를 업데이트하는 과정에서 경사하강법이 사용된다. 최종적으로 손실을 줄여 최적의 학습 결과를 찾아내는 것이 목표이다.

III. Evaluation Results

1. Evaluation setup

1) 데이터 플롯 출력 및 정규화

데이터의 전체적인 범위를 파악하기 위해 플롯을 출력한다.



데이터 정규화 코드를 추가하여 데이터의 스케일을 조절하였다.

```
# 데이터 정규화
x = (data[:, 0] - np.mean(data[:, 0])) / np.std(data[:, 0])
y = (data[:, 1] - np.mean(data[:, 1])) / np.std(data[:, 1])
```

2) 파라미터 설정 & 가중치 초기화

은닉층의 개수를 3개로 설정, 학습률을 0.0001로 설정하고 총 1000번 반복하도록 하였다. 또한, 가중치와 편향을 초기화하여 모델이 학습을 시작할 때 적절한 초기 조건을 제공하였다.

```
# 파라미터 셋팅
input_size = 1
hidden_size = 3
output_size = 1
learning_rate = 0.0001
epochs = 1000

# 가중치 초기화
np.random.seed(42)
w1 = np.random.randn(input_size, hidden_size) * np.sqrt(2 / input_size)
b1 = np.zeros((1, hidden_size))
w2 = np.random.randn(hidden_size, output_size) * np.sqrt(2 / hidden_size)
b2 = np.zeros((1, output_size))
```

3) 훈련

하이퍼볼릭 탄젠트를 활성화 함수로 이용하여 순전파 과정을 통해 예측값을 계산하였다. 이후 역전파 과정을 통해 각 층의 가중치와 편향에 대한 오차를 계산하고 학습률을 사용하여 가중치와 편향을 업데이트하였다.

```
# 훈련
for epoch in range(epochs):

    # 순전파
    z1 = np.dot(x.reshape(-1, input_size), w1) + b1
    a1 = np.tanh(z1)
    z2 = np.dot(a1, w2) + b2
    y_pred = z2

    # 손실 계산
    loss = 0.5 * np.mean((y_pred - y.reshape(-1, output_size))**2)

    # 역전파
    grad_y_pred = y_pred - y.reshape(-1, output_size)
    grad_w2 = np.dot(a1.T, grad_y_pred)
    grad_b2 = np.sum(grad_y_pred, axis=0, keepdims=True)
    grad_a1 = np.dot(grad_y_pred, w2.T)
    grad_z1 = grad_a1 * (1 - np.tanh(z1)**2)
    grad_w1 = np.dot(x.reshape(-1, input_size).T, grad_z1)
    grad_b1 = np.sum(grad_z1, axis=0, keepdims=True)

    # 경사 하강
    w1 -= learning_rate * grad_w1
    b1 -= learning_rate * grad_b1
    w2 -= learning_rate * grad_w2
    b2 -= learning_rate * grad_b2
```

2. Training / Validation error

처음 훈련을 진행할 때 학습률을 0.001로 설정하였는데 다음과 같은 오류가 발생하였다.

```
Epoch 200, Loss: nan
Epoch 300, Loss: nan
Epoch 400, Loss: nan
Epoch 500, Loss: nan
Epoch 600, Loss: nan
Epoch 700, Loss: nan
Epoch 800, Loss: nan
Epoch 900, Loss: nan
```

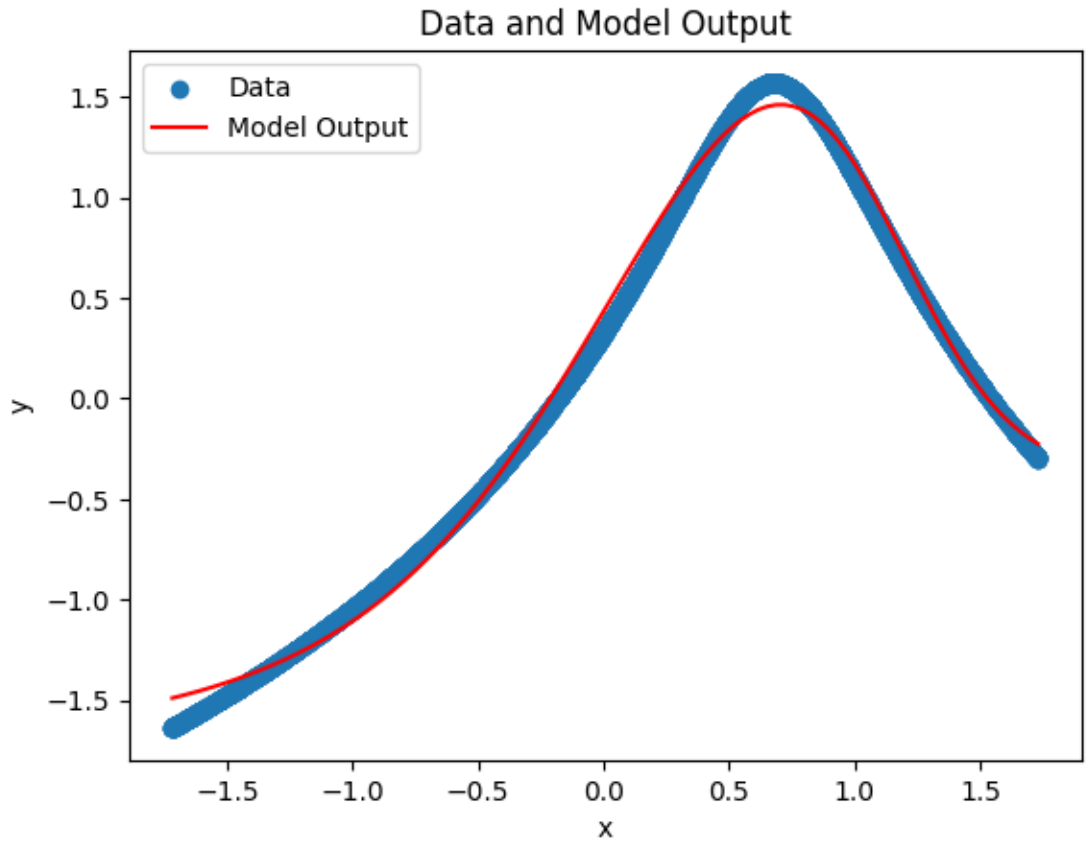
200번째 반복부터 손실이 계산되지 않았고 결국 가중치와 편향이 업데이트 되지 않아 nan값이 뜨면서 그래프를 그릴 수 없었다. 따라서 학습률을 좀 더 낮춰 0.0001로 설정하였고 손실을 계산할 수 있었다.
또한, 활성화 함수를 선택할 때도 많은 고민을 하였다. 보편적으로 사용되는 ReLU 함수는 0 이하의 입력값은 다음 층에 전달하지 않는다. 주어진 데이터셋에는 음수의 입력값이 존재하기 때문에 ReLU를 선택하기 어렵다고 판단하여 Tanh를 사용하여 훈련을 진행하였다.

3. Final Model and Parameters

가중치 및 편향

```
Final model:
w1: [[ 0.97062636 -0.36885383  1.94535502]]
b1: [[-0.17187393 -0.01149758 -2.24269954]]
w2: [[ 2.24462031 -0.3772196  -1.83958649]]
b2: [[-0.9871131]]
```

4. Set of snapshots



IV. Conclusion

이번 과제에서는 주어진 데이터셋을 설명할 수 있는 함수를 만들기 위한 인공 신경망 모델을 설계하고 훈련시켰다. 최종적으로 디자인한 모델을 학습 시켰을 때 다음과 같이 손실이 줄어들면서 0에 가까워지는 것을 확인할 수 있었다.

```
Epoch 0, Loss: 0.1905533493457871
Epoch 100, Loss: 0.032291018593904 27
Epoch 200, Loss: 0.010565102987295183
Epoch 300, Loss: 0.005676032189013471
Epoch 400, Loss: 0.004088290262135013
Epoch 500, Loss: 0.0033052440791543456
Epoch 600, Loss: 0.0027867728591336265
Epoch 700, Loss: 0.002407613801481434
Epoch 800, Loss: 0.0025371399828276937
Epoch 900, Loss: 0.0022511296027744395
```

또한, 시각화 결과 모델이 학습 데이터를 잘 반영하고 있는 것을 확인할 수 있다. 그러나 결과값에 대한 의문을 지울 수는 없었다. Tanh의 경우 출력값의 범위가 [-1, 1]인데 학습된 모델의 아웃풋의 범위가 [-1.5, 1.5]로 나왔다. ReLU를 사용했을 때도 마찬가지로 출력값의 범위가 동일했으나 모델의 플롯이 Tanh가 좀 더 데이터셋과 가까웠기 때문에 Tanh를 사용하였다. 향후 더 복잡한 데이터셋에 대한 실험을 진행한다면 모델에 맞는 활성화 함수 설정과 범위 설정이 필요함을 깨닫게 되었다.