[XOR 게이트 훈련과정 요약]

본격적으로 층별 모델을 심층적으로 파악하기 앞서 간단히 전체적인 흐름에 대해 설명드리겠습니다. 먼저, 가장 기본적인 AND 연산, OR 연산은 보시는 바와 같이 퍼셉트론으로 구현 즉, 0과 1의 영역을 구분해내는 것이 가능합니다. 그러나 XOR 연산의 경우 입력값 x_1, x_2가 서로 다를 때 결괏값이 1이 되는 연산이므로 앞의 AND, OR 연산과 같이 0과 1의 영역을 선형적으로 분리하는 것이 불가능합니다. 그렇기에 데이터로 XOR 논리연산자의 진리표를 제공하여 수많은 횟수의 학습을 진행시켜도 제대로 작동하는 가설식을 찾을 수 없습니다. 이를 해결하기 위해 만들어낸 방법이 퍼셉트론을 여러 개 사용하는 것입니다. x_1, x_2의 입력이 동시에 두 퍼셉트론으로 들어가며, 첫 번째 층에서 출력된 값이 다시 새로운 퍼셉트론 즉, 두 번째 층의 2번 퍼셉트론으로 입력되어 최종 출력을 얻는 방식으로 개선됩니다. 이러한 작동과정 중에서는 XOR 게이트 자체가 비선형 문제이므로 이를 gorufgkcxie 기 위해 비선형성을 가지는 은닉층이 최소한 하나 이상 필요하게 됩니다. 그리고 이런 각 은닉층은 비선형 활성화 함수로 입력 공간이 변환되어 력 데이터의 비선형 특성을 학습할 수 있게 됩니다. 그러면 앞서 단일 퍼셉트론으로는 구현할 수 없었던 것이 다층으로 연결되면서 왜 갑자기 가능해졌고 다층으로 연결된다는 것은 무엇을 의미하는 것일까요? 이는 가설식이 구성되는 차원이 넓어졌음을 의미합니다. 한 층을 거쳐 출력된 값이 다른 층의 입력값으로 다시 들어가는 것은 수식 안으로 수식이 다시 들어가는 재귀적 포함과정입니다. 이렇게 되면 가설식에는 차원이 늘어나는 효과가 생기게 되고 퍼셉트론으로 답을 내는 공간이 3차원 이상으로 확장됩니다. 이처럼 퍼셉트론을 둘 이상의 층으로 연결하는 구조를 다층 퍼셉트론이라 부릅니다. 퍼셉트론이 다층으로 구성된다는 것은 학습을 위해 설정된 가중치의 구성 체계가 보다 입체화되며 복잡도가 늘어난다는 것을 의미합니다. 다만, 은닉층의 개수가 많아질 수록 신경망은 더 복잡한 함수를 학습하게 되므로 과적합의 위험성이 증가할 수 있습니다. 과적합에 대해서는 이후 덧붙이겠습니다. 이렇게 모델 학습 과정 중에 증가된 차원과 자유도는 데이터의 습득에 큰 영향을 주게 되고 이것이 딥러닝의 시작이 됩니다.

이어서 게이트의 전체적인 훈련과정에 대해 소개해드리겠습니다.

첫번째 과정은 PyTorch의 nn.Module을 상속하여 XOR 클래스 즉, 신경망을 정의합니다. 이 클래스는 MLP를 구현하며, nn.Linear를 사용하여 선형 레이어를 정의합니다. 이 때 출력된 값을 활성화 함수 nn.Sigmoid를 사용하여 [0, 1] 범위로 변환합니다.

두번째로 순전파를 정의합니다. forward 메서드는 입력 데이터를 받아 신경망을 통해 전달하여 출력값을 반환합니다. nn.Sequential을 사용하여 메서드를 각 레이어에 차례대로 적용하고, 이렇게 통과된 데이터들은 변환되어 최종 출력값이 생성됩니다.

마지막으로 모델링에 관련된 작업을 수행합니다. 훈련된 XOR 모델을 확인하기 위해 먼저 XOR 클래스의 인스턴스를 생성하고, 주어진 데이터와 설정된 에폭 수로 모델을 훈련합니다. 이후 훈련된 모델을 사용하여 입력 데이터에 대한 예측을 생성하고, 이를 기반으로 예측된 클래스를 계산합니다. 마지막으로 named_parameters 메서드를 사용하여 훈련된 모델의 파라미터를 확인합니다. 이를 통해 각 레이어의 가중치(w)와 편향(bias) 값을 살펴볼 수 있습니다.

[과적합 설명]

먼저 가운데 그려진 그래프부터 살펴보겠습니다. 이 그래프를 그린 모델은 전체 변수 범위에서 데이터의 모양을 그럴듯하게 근사하고 있음을 알 수 있습니다. 물론 학습 데이터에 대해 100%의 적합도를 보여주지는 않지만 새로운 데이터가 들어오더라도 어느 정도 잘 맞을 것으로 예측할 수 있습니다.

왼쪽에 있는 선형 모델은 변수가 0 인쪽의 데이터 몇 개에 대해서는 비교적 잘 근사하지만, 나머지 데이터는 회귀 그래프에서 떨어져 있습니다. 또한 변수가 큰 부분의 데이터는 우하향하고 있는데 회귀

그래프는 그대로 우상향하고 있는 것을 보면 제대로 된 회귀는 아닙니다. 오른쪽의 경우 학습 데이터와 생성된 모델의 오차를 구해보면 0 에 가까울 것입니다. 즉, 회귀 그래프가 모든 점을 지나고 있는 것을 볼 수 있습니다. 그렇다면 둘 중 어떤 모델이 더 좋은 지를 가늠하기 위해 시험 데이터를 하나 추가하여 모델과의 오차를 비교해보겠습니다. 시험 데이터는 아래 그림에서 빨간색 점으로 나타납니다.

추가된 시험 데이터는 학습 데이터셋의 경향을 크게 벗어나지 않는 것처럼 보이지만 왼쪽의 경우, 오차가 상당히 많이 나는 것을 볼 수 있습니다. 가운데 모델을 보겠습니다. 가운데 모델은 오차가 크지 않습니다. 학습 및 시험 데이터 둘 다 비슷하게 근사하는 것을 볼 수 있습니다. 그러나 앞서 학습 데이터를 100%에 가깝게 근사했던 오른쪽 모델은 오차가 없었음에도 시험 데이터와는 큰 오차를 보이는 것을 알 수 있습니다.

위 이미지에서 왼쪽 그래프와 같이 너무 단순한 모델을 생성하여 학습 데이터와 잘 맞지 않을 때 모델이 **과소적합(Underfitting)**되었다고 합니다. 이럴 때는 모델의 복잡도를 늘려줌으로써 문제를 해결해야 합니다. 반대로 오른쪽처럼 너무 복잡한 모델을 생성하는 바람에 학습 데이터에는 굉장히 잘 맞지만 새로운 데이터에는 잘 맞지 않는 현상을 **과적합(Overfitting, 과대적합)**이라고 합니다.

위에서 살펴본 바와 같이 왼쪽 그래프와 오른쪽 그래프는 시험 데이터와의 오차가 발생하는 이유가 다릅니다. 왼쪽은 훈련 데이터도 제대로 근사를 못할 만큼 모델이 단순한 것이 문제였습니다. 이런 경우에 발생하는 오차를 **편향(Bias)**이라고 합니다. 그에 비해 오른쪽은 훈련 데이터에 대해서는 근사를 매우 잘하지만 새로운 데이터가 들어왔을 때 제대로 근사하지 못한다는 문제가 있었습니다. 이런 경우에 발생하는 오차는 **분산(Variance)**이라고 합니다.

[기본설명]

학습률(lr)은 가중치 및 편향 업데이트의 속도를 결정하므로 0.001~0.1 사이로 숫자를 넣어보며 최적의 값을 찾게 됩니다. 다만, 학습률이 올라가면 속도는 빨라지지만 최적화 과정 자체가 불안정해질 수 있기 때문에 무조건 높은 학습률이라고 좋은 것은 아닙니다.

에폭은 데이터셋 전체를 한 번 학습하는 단위로 적은 수의 에폭으로 시작하여 학습이 제대로 수행될 때까지의 에폭수를 점차 늘려나가는 것이 좋은 방법입니다. 에폭이 너무 높을 시 히든레이어와 마찬가지로 과적합이 발생할 수 있기 때문입니다.

[HiddenLayer_2]

첫 번째로 살펴볼 모델은 2개의 은닉층과 1개의 출력층으로 구성되어 있으며 입력층과 은닉층은 각각 2개의 노드로, 출력층은 2개의 입력 노드와 한 개의 출력 노드로 이루어져 있습니다. 학습을 위해 설정한 파라미터에서 에포크는 이만, 학습률은 0.2로 설정하였습니다. 학습 결과, 손실값은 점점 감소하여 최종 손실은 피피티의 마지막 값으로(0.0008782774675637484) 나타났습니다. 출력값이 0.5보다 크면 1, 그렇지 않으면 0으로 설정하였으며 결괏값이 0, 1, 1, 0으로 출력된 것을 보아 xor 학습이 잘 이루어졌음을 확인할 수 있었습니다. 학습된 모델의 가중치와 편향은 다음과 같습니다. 이후 학습된 가중치와 편향을 대입하여 최종 출력값 또한 xor로 잘 나타난 것을 확인할 수 있었습니다. 결괏값 그래프를 위해 다음과 같은 코드에 가중치와 편향을 대입하여 계산한 후 이처럼 그래프가 잘 나타난 것을 볼 수 있었습니다.

[HiddenLayer_3]

히든레이어 3개인 경우에는 앞선 레이어와 구조는 동일하지만 은닉층이 3개 존재합니다. 또한 활성화 함수와 손실함수도 앞선 레이어와 동일하게 설정하였습니다. 최적화 알고리즘으로는 에포크는 만 번, 학습률은 0.01, Adam optimizer을 사용하여 진행하였습니다. 최종 손실 값은 매우 낮은 값(2.077104181807954e-05)으로 수렴하였고 또한 결괏값이 xor 값으로 나온 것을 확인할 수 있

었습니다. 학습된 가중치와 편향은 이렇게 출력되었습니다. 가중치와 편향을 이용하여 최종 출력값도 xor로 나온 것을 확인할 수 있었습니다. 다음으로는 그래프에 대해 보여드리도록 하겠습니다. 입력변수, 가중치 변수, 편향 변수를 심볼릭 변수로 정의한 채 다음과 같이 각층의 은닉층에 시그모이드 함수를 사용하여 은닉층을 계산한 뒤 3D로 그래프를 출력하였습니다. 순서대로 은닉층 1, 2, 3번입니다. 최종적인 그래프 코드는 이렇게 사용하였으며 다음과 같은 그래프가 출력되었습니다.

[HiddenLayer_4]

히든레이어 4는 아래와 같은 코드를 사용하여 4개의 은닉층을 거치도록 하였습니다. 학습 파라미터로 30000번의 에폭, 학습률 0.01, reset='xavier'를 설정하여 손실값으로 다음과 같은 값이 출력되었습니다. 학습된 가중치와 편향은 이렇게 나왔습니다. 가중치와 편향을 사용한 최종 출력값 또한 xor로 잘 학습되었음을 알 수 있었습니다. 이후 앞선 레이어와 동일하게 변수 설정을 해준 뒤 각 은닉층의 그래프를 출력하였습니다. 순서대로 은닉층 1 ,2, 3, 4입니다. 최종적인 결괏값 그래프에 대한 코드는 다음과 같습니다. 최종 그래프는 이렇게 출력되었습니다.

[HiddenLayer_5]

마지막으로 앞서 설명한 2, 3, 4개의 은닉층과 구조는 동일하지만, 은닉층이 총 5개로 구성된 모델에 대해 설명해 드리겠습니다. 학습파라미터에 에폭은 10000번, 학습률은 0.01로 설정하였습니다. 손실값은 다음과 같이 출력되었습니다. 결괏값이 xor로 잘 학습된 것을 알 수 있었으며 학습된 가중치와 편향은 다음과 같습니다. 가중치와 편향을 이용한 최종 출력값은 다음과 같습니다. 최종적인 그래프는 다음과 같이 출력되었습니다.

[insight]

앞선 발표에서 나타난 결과를 보시면 레이어의 개수와 구성을 변경함으로써 학습된 파라미터 값이 다양하게 변화함을 알 수 있습니다. 또한 학습 중 여기 보이시는 그래프와 같이 그래프의 모양이 맞지만, 축이 0~1 사잇값이 아닌 그래프가 출력되기도 하였습니다. 이러한 문제를 해결하면서 앞선 학습이 xor이기 때문에 결괏값 축이 0~1 사이로 나와야 한다는 점을 다시 한번 깨닫게 되었습니다. 또한 레이어 4, 5에서는 그래프 모양이 앞선 2, 3과 다르게 출력되었습니다. 점 그래프를 통해 xor 결괏값이 알맞게 출력된 것은 확인할 수 있었습니다. 하지만 그래프 모양은 앞선 레이어2, 3과 다르게 출력되는 점에 대해서 모델의 학습 과정과 파라미터 설정의 영향을 확인할 수 있었습니다. 학습 과정에서는 앞선 과적합에 대한 문제와 시그모이드 함수 은닉층의 깊이가 깊어질수록 기울기 소실 문제점이 있다는 것을 알게 되었습니다. 기울기 소실은 역전파 과정에서 입력층이 출력층과 멀어지면서 기울기가 작아져 값이 0에 가까워지는 현상입니다. 이를 통해 모델 학습이 제대로 이루어지지 않을 수도 있다는 것을 알게 되었습니다. 또한 파라미터 설정에서는 학습률이 너무 높거나, 에폭 수가 작거나, 가중치 초기화 방법이 적절하지 않거나 등 다양한 이유가 존재함을 알게 되었습니다.

[기여도]

정후리(20220896): 25%

이가현(20220967): 25%

손재윤(20220625): 25%

손민정(20220817): 25%