

14. Activation Function 학습률 사용하는 이유? Softmax, Sigmoid 학습의 차이는?

- 활성화 학습은 인공신경망에서 각 노드의 출력값을 결정하는 핵심, 신경망이 비연형적인 패턴을 학습할 수 있도록 하는 핵심요소이다. 모든 층에서 선형 학습만을 사용한다면 아무리 층을 많이 쌓아도 전체적으로 하나의 선형 학습이다. 동일한 결과가 나오게 되었고, 비연형성을 부여하기 위해 활성화 학습이 필요하다.
- 출력층에서는 문제의 유형에 따라 적절한 활성화 학습이 선택된다.
- 이전 복수 문제에서는 시그모이드(Sigmoid) 학습이 사용된다. 이 학습은 각 출력값을 0과 1 사이의 학습률로 변환하여, 결과적으로 학습값은 특별 축제로 수렴할 확률로 험해진다. 시그모이드 학습은 각 뉴런의 출력값을 확장으로 계산하여 양의 차별 분류와 같이 여러 클래스에 동시에 속할 수 있는 문제에서도 사용될 수 있다.
- 반면, 다중 클래스 복수 문제에서는 소프트맥스(Softmax) 학습이 사용된다. 소프트맥스는 출력 벡터 전체를 고려하여 각 클래스의 출력값의 전체 합이 1이 되도록 정규화된 학습률로 학습률을 설정한다. 이를 통해 모델은 가능성이 높은 하나의 클래스를 선택할 수 있게 된다.
- 따라서, 시그모이드는 클래스 간 독립적인 학습 계산에 적합하며, 소프트맥스는 서로 배타적인 클래스 간 선택을 고려하는 문제에 적합하다. 모인 설계 시에는 이러한 복수 문제의 유형을 고려하여 두 학습 중 알맞은 것을 선택해야 한다.

15. Forward propagation, Backward propagation 이란?

- 순전파(Forward propagation)는 인공신경망에서 입력 데이터를 기반으로 예측값을 계산하는 과정이다. 이 과정은 입력층부터 시작해 각 은닉층을 지나 출력층까지 예측값을 계산하는 과정이다. 이러한 계산은 입력층부터 시작해 각 은닉층을 거쳐 출력층까지 한 방향으로 전개된다. 각 층에서는 이전 층의 출력값에 가중치(weight)를 곱하고 편향(bias)을 더한 후, 활성화 함수(activation function)를 적용해 다음 층의 입력값을 생성한다. 계산식은 다음과 같다.
- $$z^{(l)} = W^{(l)} a^{(l-1)} + b^{(l)}, \quad a^{(l)} = f(z^{(l)})$$
- $a^{(l)}$ 은 입력벡터, $W^{(l)}$ 은 l 번째 층의 가중치 행렬, $b^{(l)}$ 는 편향 벡터, f 는 활성화 함수이다.
- 순전파의 목적은 현재 신경망이 입력에 대해 어떤 예측값을 출력하는지를 알아내는 것이다. 예측값과 실제값의 차이를 바탕으로 손실(loss)을 계산하는데 쓰인다. 이 차이를 주로 모델 학습 단계에서 손실을 계산할 때나, 학습 이후 예측을 수행할 때 사용된다.
- 반면, 역전파(Backpropagation)은 순전파로 계산된 예측값과 실제값의 차이를 이용해 신경망의 가중치를 학습 방향으로 조절하기 위한 기울기(gradients)를 계산하는 과정이다. 이 기울기는 출력층에서 출력층 방향으로 역방향으로 전파되며, chain rule를 기반으로 각 층의 가중치와 편향이 손실에 얼마나 영향을 주는지를 미분을 통해 계산한다. 이때 계산된 기울기는 경사하강법 알고리듬에 따라 가중치를 점진적 수렴하는데 사용된다. 즉, 오차를 줄이기 위해 신경망이 어떻게 학습할지를 결정하는 핵심 기법이다. 역전파는 학습 과정 중에만 사용되며, 예측 시에는 수행되지 않는다. 역전파를 계산하는 방법은 아래와 같다.
- 다음 가중치(Weight) = 현재 Weight - learning rate * 미분값

16. 손실함수란 무엇인가? 가장 많이 사용하는 손실함수는?

- 손실함수는 인공지능 모델의 학습할 때, 예측값과 실제 정답 사이의 차이를 수치적으로 평가하는 핵심이다. 이 학습은 모델이 얼마나 잘못 예측했는지를 정량화하여, 하나의 손실과 같은 손실함수를 계산하는데 사용된다. 이렇게 계산된 손실 값은 학습 과정에서 모델의 성능에 따라 변화하는 기준이 되며, 손실 값을 최소화하는 방향으로 모델이 점점 더 정답에 가까운 출력을 내도록 학습을 진행하게 된다. 손실함수는 문제의 유형에 따라 적절한 형식을 선택해야 하며, 모델의 성능과 학습 과정에 큰 영향을 미친다.
- MSE(Mean Squared Error)는 회귀 문제에서 주로 사용되는 손실함수이다. 이 학습은 예측값과 실제값의 차이를 제곱하여 평균을 내는 방식으로, 결과를 큐드를 확장한 손실함수로 계산되는 특징이 있다. 예전에 큰 화제에 빠졌던 학습방법이다.
 - Cross-Entropy Loss는 분류 문제에서 모델 사용되는 손실함수로, 모델이 예측한 학습 분포와 실제 정답 사이의 차이를 기반으로 손실을 계산한다. 예측이 정답에서 일어날 수 있는 모든 경우에 걸쳐 확률을 유지하는 정확한 학습 예측을 유도하는데 효과적이다. 이 손실함수는 이런 분포를 다중 클래스 분포 모두 활용될 수 있으며, 복수 문제의 기본적인 손실함수로 자리잡고 있다.
 - Binary Cross-Entropy는 Cross-Entropy의 변형으로, 이런 분류 문제에 특화된 손실함수이다. 정답이 0 또는 1로 주어지는 문제에서, 모델은 각 샘플에 대해 해당 클래스의 학습률을 출력하고, 이 학습률과 실제 정답 사이의 차이를 손실로 계산된다. 학습률에 가까울수록 손실은 적고, 멀수록 손실은 커지며 학습률 기반 예측이 매우 중요하게 표시된다. 출력층에서 주로 sigmoid 학습률을 사용된다.
 - Categorical Cross-Entropy는 클래스가 세개 이상일 때 사용하는 손실함수로, 각 클래스에 대한 학습률을 예측하고 정답 클래스의 학습률 차이를 기반으로 손실을 계산한다. 정답은 일반적으로 원-핫 인코딩된 형태로 저장되며, 모델은 각 클래스에 대해 학습률을 출력한다. 이 학습은 클래스 간의 학습률 차이를 고려하여 손실을 계산하였으므로, 다중 클래스 분류 문제에 적합하다. 출력층에서는 소프트맥스(Softmax) 학습률을 함께 사용하는 것이 일반적이다.

17. 유틸리티어란 무엇인가? 유틸리티어와 손실함수의 차이점은?

- 유틸리티어는 인공신경망 모델의 학습 과정에서 가중치(파라미터)를 초기화하여 손실함수의 값을 최소화하는 역할을 하는 알고리즘이다. 유틸리티어는 모델의 정답에 가까운 출력을 낼 수 있도록 가중치를 초기화하는 방식과 전략을 담당하는 핵심 모듈이다. 손실함수의 기울기(미분값)를 계산한 뒤, 그 힘을 바탕으로 가중치가 어느 방향으로 얼마나 이동해야 손실이 줄어들 수 있을지 계산한다. 기본적인 경사하강법은 학습률이 너무 크면 발산하거나, 너무 작으면 수렴이 매우 느려진다. 이러한 문제를 해결하기 위해 등장한 것이 디파인 흐름의 유틸리티어이다.
- 손실함수는 모델이 예측한 값과 실제 정답 사이의 차이를 수치화한 값을 계산하는 평가 학습이다. 이 같은 모델의 학습 결과를 예측하고 있는지를 정량적으로 표현해주며, 모델의 성능이 얼마나 나쁜지를 알려주는 기준 역할을 한다.
- 손실함수는 크게는 예측이 정답과 크게 차이난다는 의미이고, 반대로 손실함수가 작다는 것은 모델이 정답에 가까운 예측을 하고 있다는 것을 의미한다.
- 유틸리티어는 이 손실값을 줄이기 위해 모델의 가중치를 조정하는 알고리즘이다. 즉, 손실함수는 학습 솔루션을 '최적화'하는 기준이고, 유틸리티어는 그 최적화 결과를 바탕으로 모델을 개선해 나가는 구현이다.

18. 경사하강법 (Gradient Descent)의 의미 및 종류

- : 경사하강법은 미신너닝 및 딥러닝 모델이 학습하는 과정에서 손실함수의 경사를 최적화하기 위해 사용하는 대표적인 최적화 알고리즘이다. 모델의 예측은 결과와 실제 정답 사이의 차를 줄이기 위해 손실함수의 기울기(그라디언트)을 계산한 뒤 그 기울기의 반대 방향으로 가중치와 편향을 조정해 이를 반복적으로 업데이트하는 방식으로 작동한다. 이 과정을 통해 모델은 점차 더 나은 예측을 하도록 학습해 나간다. 경사하강법은 학습 데이터의 사용 방식에 따라 세 가지 종류로 나뉜다.
- 1) 배치 경사하강법 (Batch Gradient Descent)은 전체 학습 데이터를 한 번에 사용해 손실함수의 평균 기울기를 계산한 후, 이를 바탕으로 가중치를 업데이트 한다. 이 방식은 계산이 안정하고 수렴速度가 일정하다는 장점이 있지만, 데이터 양에 맞춰 연산량이 매우 크며 학습 속도가 느려는 단점이 있다.
 - 2) 확률적 경사하강법은 학습 데이터 중 하나의 샘플만을 사용해 매번 가중치를 즉시 업데이트하는 방식이다. 연산 속도가 매우 빠르고 유기적인 학습에 적합하다는 장점이 있으나, 개별 샘플에 기반한 업데이트로 인해 손실값이 불안정하게 변동되며, 최적값 근처에서 수렴이 어려울 수 있다.
 - 3) 미니배치 경사하강법은 전체 데이터셋을 일정 크기의 작은 묶음(미니배치)으로 나누어, 각 미니배치를 기준으로 기울기를 계산하고 가중치를 업데이트 한다. 이 방식은 배치 학습의 안정성과 확률적 학습의 빠른 학습 속도를 혼합해 적용한 것이다.

19. 교차 검증과 K-Fold 교차검증의 의미와 차이

- : 교차검증은 예산제모델의 성능을 보다 안정적으로 신뢰할 수 있게 평가하기 위한 방법이다. 일반적으로 데이터는 훈련 세트와 검증 세트로 한 번 나누어 평가할 때, 그 분할 방식에 따라 평가 결과가 달라질 수 있어 모델의 일반화 성능을 제대로 판단하기 어렵다. 교차검증은 아래와 같은 단계를 통해 해결하기 위해, 데이터를 여러 개의 부분(Fold)으로 나누고, 각 부분을 번갈아가며 검증 세트로 사용하면서 나머지 데이터를 훈련에 사용하는 작업을 여러 번 반복한다. 이를 통해 모델이 다양한 데이터 분할에 대해서도 일관된 성능을 보여줄 수 있으며, 모델의 학습 능력과 일반화 능력을 높이는 데에 도움이 된다.
- K-Fold 교차 검증은 교차 검증의 대표적인 방법 중 하나로, 전체 데이터를 K개의 동일한 크기로 나눈다. 이때 나뉜 각각의 조각을 Fold(폴드)라고 부른다. 그 후, 각 폴드를 한 번씩 평가용(검증용) 데이터로 사용하고 나머지 폴드들은 학습용 데이터로 사용하여, 모델을 꽂기 전 학습하고 평가한다. 이렇게 하면 모든 데이터가 한 번씩은 검증에 사용되었고, 데이터 사용의 불균형성이 낮고, 평가 결과에 특정 데이터가 대체로 영향을 미치는지를 알 수 있다. 교차 검증은 평가 방식을 통틀어 이는 일반적인 개념이고, K-Fold 교차 검증은 그 중 하나의 구체적인 구현 방법이라는 차이가 있다.

20. 하이퍼파라미터 튜닝 어떤 무엇인가?

- : 하이퍼파라미터 튜닝은 미신너닝 또는 딥러닝 모델의 성능을 최적화하고, 학습률이나 과적합 등이 안정적인 학습을 목표하기 위해 시점이 각각 설정해야 하는 하이퍼파라미터의 값을 조정하는 과정이다. 하이퍼파라미터에는 학습률, 배치 크기, 습 수, 뉴런 수, 정규화 계수, 학습률 학습률 모델 내부에서 자동으로 학습되지 않는 값들이 포함된다. 하이퍼파라미터는 모델의 학습 방식과 복잡도에 적합적인 영향을 미치며, 잘못 설정될 경우 모델의 학습률이 높아지거나 저하될 수 있다. 학습률이 너무 높으면 손실함수가 수렴하지 않고 발산할 수 있으며, 너무 낮으면 학습 속도가 지나치게 느려지는 등의 학습 불안정성이 발생된다. 하이퍼파라미터 튜닝을 통해 모델은 더욱 빠르고 정확한 학습을 할 수 있으며, 테스트 데이터에 대해서도 일반화 성능이 향상된다.

21. CNN의 활성화 (Convolution)의 역할

- CNN(활성화 신경망)에서 활성화는 네트워크 초기 계층에 위치하여, 주로 이미지나 영상, 시계열 데이터와 같은 고차원 입력에서 유의미한 특징(feature)을 추출하는 핵심 역할을 수행한다. 이 계층은 주로 네트워크의 가장 앞부분, 즉 입력을 처음 받을 때 사용된다.
- 활성화 학습은 필터를 적용해 데이터 위에 윤곽선을 그리며 적용된다. 이 과정에서 필터와 입력의 양의 영역에 대해 내적 연산(dot product)을 수행하고, 그 결과를 축소해 모자 특징맵(feature map)을 생성한다. 하이퍼파라미터는 특징의 필터는 통장한 득점(예: 수평선, 수직선, 가로자리, 절감 등)을 감지하여, 여러 개의 필터를 사용하면 다양한 특징을 동시에 추출할 수 있다. 이 과정을 통해 모델은 원본 이미지의 공간적 구조와 중요한 정보를 유의하게 분별할 수 있다. 또, 같은 필터를 반복해서 사용하면서 학습률 파라미터 수가 줄어들어 계산의 효율화이다. 또한 활성화는 전연결층(fully connected layer)과 달리 가치를 공유하고, 학습해야 할 파라미터 수가 줄어들어 계산 효율성이 높다.

22. CNN의 폴링(Pooling Layer)의 역할

- : CNN에서 폴링(Pooling Layer)은 주로 활성화층 뒤에 위치하여, 활성화를 통해 추출된 특징 맵(feature map)의 크기를 줄여주는 역할을 한다. 이 과정은 차원축소(dimensionality reduction)라고 하며, 연산량을 줄이고 모델의 복잡도를 낮추는데 기여한다.
- 폴링은 일반적으로 작은 영역(예: 2x2) 내에서 최대값(Max Pooling)이나 평균값(Average Pooling)을 선택하여 하나의 대표값으로 치환하는 방식으로 수행된다. 이로 인해 데이터의 크기는 차이가지만 줄여보는 유효할 수 있다. 이때 폴링 리셉트는 통해 계산량이 줄어들고, 모델은 입력 이미지의 위치나 모양이 걸림 변화에도 학습 정보를 안정적으로 추출할 수 있는 특성, 즉 위치 불변성(translational invariance)을 갖게 된다. 또한 파라미터 수가 줄어들어 학습률을 줄이는 데에도 도움이 되다. 계산적으로 폴링은 CNN의 연산 효율성을 높이고, 더 일반화된 특징 표현을 학습할 수 있도록 풍기는 중요한 계층이다.

23. CNN의 Dense Layer (완전연결층)의 역할

- : Dense Layer(완전연결층)은 활성화층과 폴링층을 거쳐 추출된 고수준의 특징들을 기반으로 최종 예측 결과를 생성하는 계층이다. 이 계층은 주로 CNN의 마지막 단계에 위치하여, 블록나 흐미와 같은 출력 결과를 도출하는데 사용된다.
- 완전연결층에서는 각 뉴런이 어떤 층의 모든 출입값과 연결되어 있으며, 이 연결을 통해 입력값에 가중치를 곱한 후 모두 더하고, 학습한 학습률을 적용된다. 이러한 과정을 통해 입력된 특징 정보가 종합적으로 처리되어, 예측해야 할 클래스의 확률 분포나 흐미값 등 최종 출력이 계산된다. Dense layer는 특징 맵을 일차원 벡터로 평탄화(Flatten)한 후, 이 점을 기반으로 학습률을 수행함으로써, 차원간 특징 간의 비슷한적인 연관을 학습할 수 있게 된다. 또한 네트워크가 학습한 파라미터를 고려하여 최종 판정을 내는데 중요한 역할을 한다. 마지막 단계연결은 CNN에서 추출된 다양한 특징을 종합적으로 결합하여 최종 출입값을 생성함으로써 블록나 흐미 계층을 수행하는 역할을 맡고 있다.

24. CNN의 Stride와 Filter의 역할 및 필터 가중치 경량化工

: stride는 필터가 입력 데이터 위를 이동하는 간격을 의미한다. stride 값이 1인 필터가 한 번씩 이동하고, 값이 2 이상일 때 필터가 더 넓은 간격으로 이동하여 특징맵의 크기가 작아진다. stride를 크게 설정하면 출력 크기가 줄어들어 계산량이 감소하고, 끝장 전자를 더 압축해 표현할 수 있다. Filter는 입력 데이터에서 특징을 추출하기 위해 사용하는 작은 헤드 형태의 가중치이다. 필터는 이미지 내에서 엣지, 코너, 텍스처 등과 같은 특정 패턴을 감지하는 역할을 하며, 이 과정을 통해 특징 맵이 생성된다. 다양한 필터를 사용하면 서로 다른 시각적 특성을 포착할 수 있다.

필터의 가중치 경량化工는 학습을 통해 이루어진다. 초기에는 필터 가중치를 무작위로 설정하여, 이후 학습 과정에서 디그리 알고리를 통해 손실함수의 기울기를 계산하고, 경사기법을 사용하여 가중치를 비례적으로 업데이트한다. 이 과정을 통해 필터는 입력 데이터에서 중요한 특징을 정성 더 효과적으로 추출할 수 있게 된다.

25. RNN을 사용하는 이유와 한계점

: RNN(Recurrent Neural Network)은 시계열 데이터, 자연어 처리, 음성 인식 등 데이터의 순서와 유래에 중요한 문제를 해결하기 위해 사용된다. RNN은 일반적인 신경망과 달리 이전 시점의 출력을 현재 시점의 입력과 함께 고려하여 차이점을 찾거나, 시점에 따른 데이터 간의 연속적인 의존성을 학습할 수 있다. 이러한 특성 덕분에 문장의 문맥을 파악하거나 주석의 관계를 기반으로 미래를 예측하는 등 시간의 순서에 의존하는 문제에 효과적이다.

하지만 RNN에는 몇 가지 구조적 한계가 존재한다. 대표적으로 기울기 소실 문제가 있다. 이는 시퀀스(순서대로 나열된 데이터)가 길어질수록 연결과 각자에서 기울기가 모험적으로 차이가 생기면서 초기 기울기의 영향력이 사라지는 현상으로, 초기 의존 관계(long-term dependency)를 학습하는데 어려움을 초래한다. 또한, RNN은 시점별 계산에 순차적으로 이루어지기 때문에 병렬 처리가 어렵고 학습 속도가 느려는 단점도 있다. 이로 인해 긴 문장이나 긴 시퀀스를 처리하는 데 비효율적일 수 있다.

26. LSTM을 사용하는 이유와 한계점

: LSTM(Long Short-term Memory)은 순환 신경망(RNN)의 구조적 한계를 보완하기 위해 고안된 모델이다. 일반적인 RNN은 긴 시퀀스를 처리할 때 각각 정보를 효율적으로 유지하지 못하는 경향이 있는 경향을 보여온다. 이를 위해 LSTM은 세 가지 게이트(임의 게이트, 삭제 게이트, 출력 게이트)를 통해 충분한 정보를 유지하고 복제된 정보는 차단하는 특성을 확보할 수 있다. 이 구조를 통해 자연어 처리, 음성 인식, 시계열 예측 등에서 기존 RNN보다 훨씬 더 성능을 발휘한다.

LSTM의 이름은 장기 기억(Long-term Memory)과 단기 기억(Short-term Memory)을 동시에 처리하는 구조적 특성에서 유래하였다. 단기 기억은 바로 이전의 입력처럼 바로전 정보를 기억하고, 장기 기억은 문맥이나 주제처럼 장기적인 유의사항에 대한 정보를 포함한다.

LSTM은 이 두 가지 정보를 분리하여 효과적으로 학습할 수 있도록 설계되었다.

하지만 LSTM은 다음과 같은 한계점을 지녔다. 첫째, 내부 구조가 복잡하여 학습 속도가 느려다. 이는 다수의 게이트를 혼란화 시킨다. 두 번째, 모델의 파라미터 수가 많아 메모리 사용량이 크며, 하드웨어의 가능성을 초래한다. 셋째, 병렬 처리가 어렵기 때문에 하드웨어 자원의 저학용한 환경에서는 학습이 비효율적일 수 있다. LSTM은 시계열 의존성이 중요한 시퀀스 데이터를 처리하는 데 유용하지만 구조적 복잡성과 예산 자원 노동에 따른 고려가 필요하며, 경쟁에 따라 GRU나 Transformer 같은 대체 모델의 활용성이 적절할 수 있다.

27. GRU를 사용하는 이유와 차별점

: GRU(Gated Recurrent Unit)는 순환신경망(RNN)의 한 변형으로, LSTM의 구조적 복잡성을 줄이면서도 유사한 성능을 달성할 수 있도록 고안된 모델이다. GRU는 LSTM보다 간단한 구조를 가지고 있으며, 계산량이 적고 학습 속도가 빠르다는 장점을 지닌다. 이 때문에 모델의 경량화가 필요하거나 자원이 저학용한 환경에서 효과적으로 사용된다.

LSTM이 임의 게이트, 삭제 게이트, 출력 게이트의 세 가지 게이트를 사용하는 데 반해, GRU는 임의 게이트와 삭제 게이트 두 가지만을 사용한다. 이러한 구조적 단순화는 계산 효율을 높이고 학습 시간을 단축하는데 기여한다. 특히 GRU는 내부적으로 셀 상태와 은닉 상태를 병렬화해 않고, 하나의 통합된 은닉 상태만을 사용한다. 이로 인해 메모리 사용량이 줄어들고 구현이 간편해지는 효과가 있다.