텐서플로우 플레이그라운드 핵심 개념 쉽게 배우기

텐서플로우 플레이그라운드는 복잡한 딥러닝 원리를 시각적으로 체험할 수 있는 최고의 도구입니다. 이 도구를 이해하기 위한 14가지 핵심 개념을 쉽고 재미있게 설명해 드릴게요!

1. 손실함수 (Loss Function)

⚡ **비유: AI의 성적표를 매기는 심판**

손실 함수는 인공지능이 **'얼마나 틀렸는지'**를 알려주는 점수판입니다. AI가 정답을 맞히면 점수가 낮아지고(손실이 적어짐), 틀리면 점수가 높아집니다(손실이 커짐). AI의 목표는 이 손실 함수 값이 가장 낮아지는 지점을 찾는 것입니다.

2. 손실함수와 딥러닝

손실 함수는 딥러닝 학습 과정 전체를 이끌어가는 나침반입니다. AI는 수많은 데이터로 예측을 시도하고, 손실 함수가 매긴 점수(손실)를 확인한 뒤, 이 점수를 낮추기 위해 신경망 내부의 가중치(Weights)를 조절하는 방식으로 학습합니다.

주요 손실 함수의 종류

AI가 풀고자 하는 문제(예측 문제인지, 분류 문제인지)의 목적에 따라 적절한 손실 함수를 선택해야 가장 효율적으로 학습할 수 있습니다.

1. **평균 제곱 오차 (Mean Squared Error, MSE)**
   - **사용 목적:** 회귀(Regression) 문제. (예: 주택 가격 예측, 온도 예측)
   - **설명:** 실제 값과 예측 값의 **차이를 제곱**하여 평균 낸 값입니다. 오차가 클수록 제곱으로 인해 손실이 훨씬 커지기 때문에, 모델이 큰 오류를 줄이는 데 집중하게 됩니다.
2. **이진 교차 엔트로피 (Binary Cross-Entropy)**
   - **사용 목적:** 이진 분류(Binary Classification) 문제. (예: 합격/불합격, 참/거짓)
   - **설명:** 결과가 0 또는 1로만 나오는 두 가지 선택지 중 하나를 예측할 때 사용합니다. 모델의 예측 확률 분포와 실제 정답의 확률 분포가 얼마나 다른지 측정하여 손실을 계산합니다.

3. **범주형 교차 엔트로피 (Categorical Cross-Entropy)**
    - **사용 목적:** 다중 클래스 분류(Multi-class Classification) 문제. (예: 사과/바나나/귤 중 하나 예측)
    - **설명:** 세 가지 이상의 여러 선택지 중 하나를 예측할 때 사용합니다. 이진 교차 엔트로피를 여러 개의 클래스로 확장한 형태입니다.

3. 학습률 (Learning Rate)

⚡ **비유: 산을 내려가는 등산가의 '발걸음 크기'**

경사하강법을 설명하기 전에! 학습률은 AI가 손실을 줄이기 위해 한 번에 얼마나 크게 가중치를 조절할지 결정하는 값입니다.

- **학습률이 크면:** 발걸음이 커서 빨리 내려가지만, 목표 지점을 휙 지나치거나(오버슈팅) 불안정해질 수 있습니다.
- **학습률이 작으면:** 발걸음이 작아 매우 느리지만, 목표 지점을 정확하게 찾을 가능성이 높습니다.

4. 경사하강법 (Gradient Descent)

⚡ **비유: 안개 속에서 가장 낮은 계곡을 찾는 등산가**

AI가 손실 함수라는 산봉우리의 가장 낮은 계곡(최소 손실 지점)을 찾는 방법입니다. 등산가(AI)는 눈을 감고(데이터를 전부 볼 수 없음) 현재 서 있는 위치에서 경사가 **가장 가파른 방향**으로, **학습률만큼의 크기**로 한 걸음씩 내려갑니다. 이 과정을 반복하여 최저점을 향해 이동하는 것이 경사하강법입니다.

5. 활성화 함수 (Activation Function)

⚡ **비유: 정보를 통과시키는 '신경 세포의 스위치'**

활성화 함수는 뉴런(노드)에서 들어온 입력 값의 합을 다음 뉴런으로 보낼지 말지, **'얼마나 보낼지'**를 결정하는 역할을 합니다. 가장 중요한 기능은 AI에 **비선형성(Non-linearity)**을 부여하여, 직선으로는 분류할 수 없는 복잡한 문제(예: XOR 문제)도 해결할 수 있게 만들어주는 것입니다.

6. 인공지능의 암흑기와 활성화 함수

과거 인공지능 연구가 정체되었던 '암흑기(AI Winter)'의 한 가지 원인은 단순한 활성화 함수(예: Step Function)의 한계 때문이었습니다. 복잡한 문제를 해결할 수 있는 **비선형 활성화 함수**의 등장(특히 ReLU)과 깊은 신경망(Deep Network)의 발전이 딥러닝의 부흥을 이끌었습니다.

7. 시그모이드 함수 (Sigmoid Function)

❗ 특징: 입력 값을 0과 1 사이의 값으로 압축합니다. (S자 모양)
문제점: 입력 값이 너무 크거나 작으면 기울기가 거의 0이 되어 학습이 멈추는 "기울기 소실(Vanishing Gradient)" 문제가 발생하기 쉽습니다.

8. 하이퍼볼릭 탄젠트 함수 (Hyperbolic Tangent Function, tanh)

❗ 특징: 시그모이드와 유사한 S자 모양이지만, 출력 값이 -1과 1 사이입니다.
장점: 출력의 중심이 0에 가깝기 때문에 시그모이드보다 학습이 빠르고 효율적입니다. 하지만 여전히 기울기 소실 문제의 가능성이 있습니다.

9. ReLU 함수 (Rectified Linear Unit)

❗ 특징: 입력이 0보다 작으면 0, 0보다 크면 입력 값을 그대로 출력합니다.
장점:

1. **계산이 매우 빠릅니다.** (단순한 조건문 형태)
2. 기울기 소실 문제를 완화합니다. (양수 영역에서는 기울기가 항상 1)
   단점: 입력이 음수이면 뉴런이 영구히 비활성화되는 "Dying ReLU" 문제가 발생할 수 있습니다.

10. Leaky ReLU 함수

❗ 특징: ReLU의 단점을 보완합니다. 입력이 0보다 작을 때도 아주 작은 값(예: $0.01x$)을 출력하여 뉴런을 완전히 죽지 않게 합니다.
장점: Dying ReLU 문제를 해결하여 학습 안정성을 높입니다.

11. 정규화 (Regularization)

⚡ **비유: AI의 과외 활동을 제한하는 '엄격한 선생님'**

정규화는 AI가 학습 데이터만 완벽하게 **'외워서'** (과적합, Overfitting) 실제 처음 보는 데이터에 대해서는 형편없는 성능을 보이는 것을 방지하는 기술입니다. 즉, AI가 너무 한쪽에 치우치지 않고 **일반적인 지식**을 배우도록 가중치(Weights)에 벌칙을 부과하는 것입니다.

12. L1 정규화 (Lasso Regularization)

L1은 가중치 값의 **절댓값의 합**에 벌칙을 부과합니다. 이 벌칙은 중요하지 않은 가중치들을 아예 0으로 만들어 버리는 경향이 있습니다. 덕분에 모델이 불필요한 특성(Feature)들을 스스로 제거하는 **'특성 선택(Feature Selection)'** 효과를 가집니다.

13. L2 정규화 (Ridge Regularization)

L2는 가중치 값의 **제곱의 합**에 벌칙을 부과합니다. 이 벌칙은 가중치들이 너무 커지는 것을 막아 모든 가중치가 작고 고르게 분포되도록 유도합니다. L2는 모델의 복잡도를 완화하고 과적합을 방지하는 데 가장 널리 사용됩니다.

14. 정규화 비율 (Regularization Rate)

⚡ **비유: 벌칙의 '강도'**

정규화 비율(또는 $₩lambda$)은 정규화 벌칙을 손실 함수에 얼마나 강하게 반영할지를 결정하는 값입니다.

- **비율이 높으면:** 벌칙이 강해져 가중치가 매우 작아지고, 과적합은 막을 수 있지만, 모델이 너무 단순해져서 **'과소적합(Underfitting)'**이 발생할 수 있습니다.
- 비율이 낮으면: 벌칙이 약해져 정규화 효과가 미미할 수 있습니다.
  이 값은 사용자가 직접 설정해야 하는 하이퍼파라미터 중 하나입니다.