250715

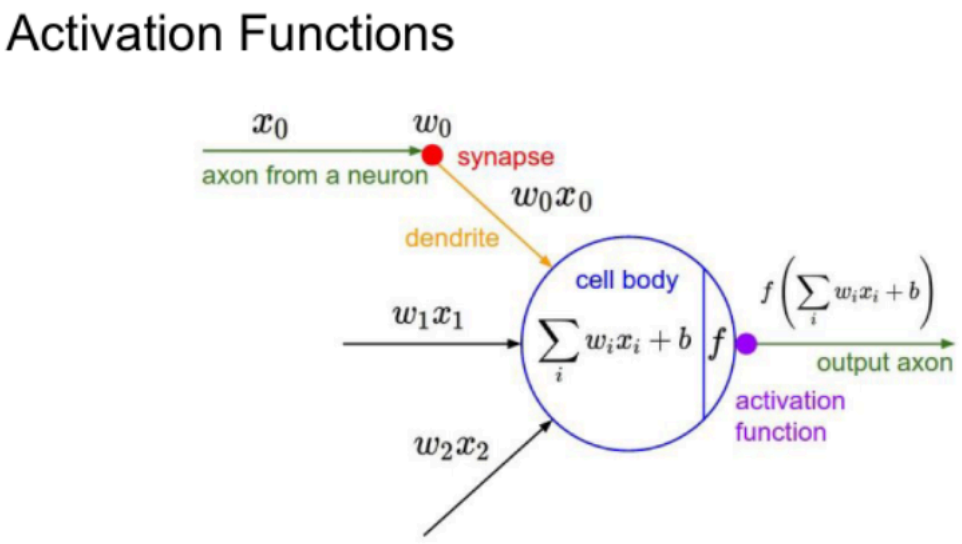
2주차-1

주피터에 코드 누적

OX_Classification

- MLP로 분류 진행하기
- OX_Dataset 불러오기
- OX_image 전처리
- MLP로 진행시 K-fold적용 , 미적용 각 각 정확도 산출

- acitvation function
  - 딥러닝 네트워크에서 노드에 입력된 값들을 비선형 함수에 통과시킨 후 다음 레이어로 전달하는데, 이 때 사용하는 함수 → 입력 신호의 총합을 출력 신호로 변환하는 함수

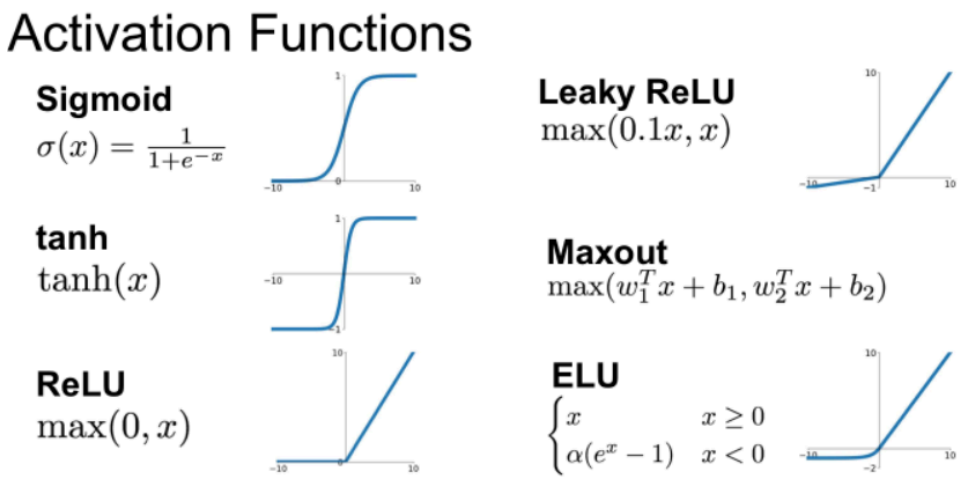


[출처 cs231n] Activation Function

  - 목적

- **인공 신경망에서 각 노드의 입력 신호 총합을 출력 신호로 변환하여, 네트워크의 비선형성을 만들고 복잡한 문제 해결 능력을 향상시키는 것**
- 입력 데이터를 적절히 처리하여 다음 레이어로 전달할지 여부를 결정하고, 이를 통해 신경망이 다양한 패턴을 학습하고 예측할 수 있도록 도움 → **신경망의 학습 능력을 향상시키고, 다양한 비선형 문제를 해결할 수 있도록 하는 핵심적인 역할**

◦ 종류 및 특징



[출처 cs231n] Sigmoid부터 LU 패밀리까지 한 번 제대로 배워보도록 하겠습니다

◦ Sigmoid

- 특징
  - 출력 값을 0에서 1로 변경 (Squashes number to range [0, 1])
  - 가장 많이 사용되었던 활성화 함수
- 단점
  - Saturation (포화상태)
  - Sigmoid outputs are not zero-centered

◦ tanh

- 특징
  - 출력 값은 -1에서 1로 압축시킴
  - zero-centerd (Nice!, sigmoid가 가졌던 두 번째 문제점을 해결)
- 단점
  - 여전히 gradient가 죽는 구간이 있음 (양수/음수 구간 모두 존재)

- ReLU
  - 특징
    - 양의 값에서는 Saturated 되지 않음
    - 계산 효율이 뛰어납니다. sigmid/tanh보다 훨씬 빠름 (6배 정도)
    - 생물학적 타당성도 가장 높은 activation function
  - 단점
    - zero-centerd가 아니라는 문제가 다시 발생 (non-zero centered)
    - 또한 음수 영역에서 saturated 되는 문제가 다시 발생

▼ 그 외

- Leaky ReLU (ReLU 이후에 조금 수정된 버전)
  - 특징
    - ReLU와 유사하지만 negative regime(음의 영역)에서 더 이상 0이 아님
    - saturated 되지 않음
    - 여전히 계산이 효율적이며 빠름
    - 더 이상 Dead ReLU 현상이 없게 됨
- PReLU (prametric rectifier)
  - egative space에 기울기가 있다는 점에서 Leaky ReLU와 유사한 것을 알 수 있음
  - 다만 여기에서는 기울기가 alpha라는 파라미터로 결정 → alpha를 딱 정해놓는 것이 아니라 backprop으로 학습시키는 파라미터로 만든 것
- ELU (Exponential Linear Units)
  - LU 패밀리 (ReLU, LeakyReLU, PReLU...) 하지만 ELU는 zero-mean에 가까운 출력 값
  - 앞선 LU패밀리가 zero-mean 출력 값을 갖지 못하는 것에 비해 상당한 이점
  - negative에서 "기울기"를 가지는 것 대신에 또다시 "Saturated"되는 문제

        - ReLU와 Leaky ReLU의 중간 정도라고 봄 → ELU는 Leaky ReLU처럼 zero-mean의 출력을 내지만 Saturation관점에서는 ReLU의 특성도 가지고 있음

- softmax
    - 입력받은 값을 출력으로 0~1사이의 값으로 모두 정규화하며, 출력 값들의 총합은 항상 1이 되는 특성을 가진 함수
    - 분류하고 싶은 클래수의 수만큼 출력으로 구성
    - **출력층**에서 사용되는 함수
    - **다중 클래스 분류 모델**을 만들 때 사용 → 결과를 확률로 해석할 수 있게 변환해주는 함수로 높은 확률을 가지는 class로 분류 (결과값을 정규화시키는 것으로도 생각)

$$p_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_j}}$$

$$j = 1, 2, \ldots, K$$

    - 특징
        - **확률 분포 생성**

          출력 값은 0과 1 사이의 실수이며, 모든 출력 값의 합은 1 → 따라서 각 클래스에 대한 확률로 해석
        - **입력 값의 상대적인 크기 반영**

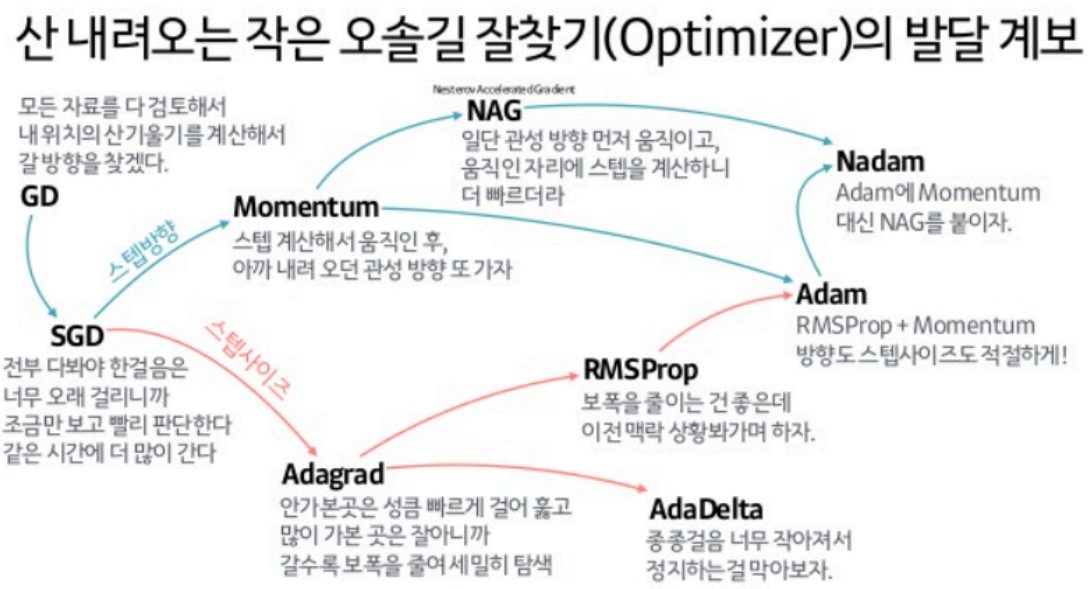
          입력 값 중 큰 값에 대해 더 큰 출력을 생성하여, 해당 클래스가 선택될 가능성을 높임
        - **미분 가능**

          역전파 알고리즘을 사용하여 학습에 활용
        - **다중 클래스 분류 문제에 적합**

          여러 개의 클래스 중에서 하나의 클래스를 선택하는 문제에 유용하게 사용
        - **안정적인 학습**

          출력 값의 범위가 0과 1 사이로 제한되어 안정적인 학습이 가능

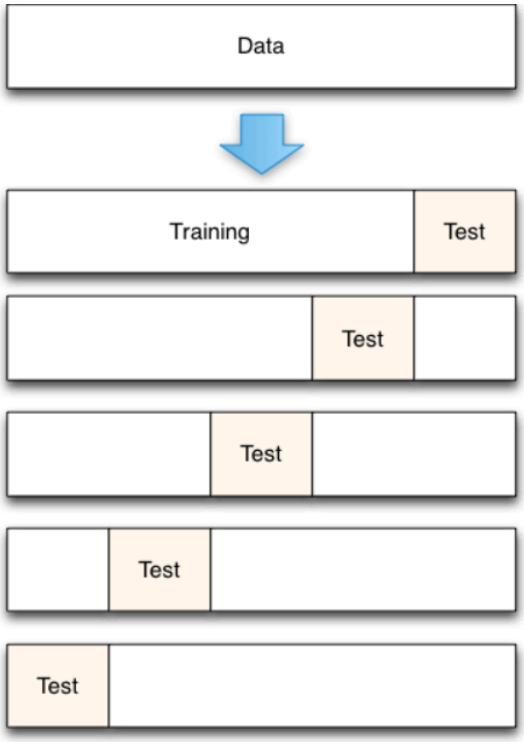
- 최적화 함수
  - **손실 함수의 결과를 최소화하는 데 사용되는 함수**
  - 목적
    - 주로 손실 함수의 값을 최소화하는 것
  - 종류, 특징



    - **GD (경사하강법, (Batch) Gradient Descent)**
      - **전체 데이터를 사용**하여 함수의 기울기를 구하고 기울기의 반대 방향으로 이동시켜 최소값에 이르를때까지 반복하는 기법
      - 전체 데이터에 대해서 loss 및 기울기를 구하여 이동 → 많은 연산량을 요구, 안정적
    - **SGD (확률적경사하강법, Stochastic Gradient Descent)**
      - **매 스텝마다 랜덤 샘플링된 일부 데이터만 사용**하여 loss 및 기울기를 구하여 이동 → 일부 데이터(Mini-batch)만을 사용하기에 계산속도가 GD에 비해서 빠름, 불안정
    - **Momentum**
      - 이동하는 방향을 결정할 때
      - 이전 스텝의 업데이트 방향을 기억하여, 관성을 가진 것처럼 움직여서 지역 최저점(local minima)에 갇히는 것을 방지

        - **AdaGrad (Adaptive Gradient)**
        - 손실 함수 곡면의 변화에 따라 적응적으로 학습률을 정하는 알고리즘
            - 각 파라미터에 대한 학습률을 다르게 조정하여, **학습이 진행됨에 따라 학습률을 감소**시키는 방식 → 희소한 데이터에 효과적
        - **RMSProp (Root Mean Sqaure Propagation)**
            - 기존 AdaGrad가 학습이 진행됨에 따라서 **학습률이 감소하다 0으로 수렴하여 더이상 학습되지 않는 문제점을 개선**하기 위해서 등장한 기법
            - **지수 이동 평균**(Exponential Moving Average, EMA)을 활용하여 기울기를 누적 → **최근의 기울기는 많이 반영**하고 먼 과거의 기울기는 조금만 반영
        - **Adam (Adaptive Moment Estimation)**
            - **Momentum**과 **RMSProp**의 **장점을 결합**한 알고리즘
            - 각 파라미터에 대한 **진행하던 속도에 관성도 주고, 최근 경로의 곡면의 변화량에 따른 학습률을 적응적으로 조정** → 현재 딥러닝에서 가장 널리 사용되는 최적화 알고리즘 중 하나

- k-fold (K-Fold Cross Validation)
    - 데이터셋을 K개의 폴드(fold)로 나눈 후, 매번 다른 폴드를 검증 데이터로 사용하여 모델을 평가
    - k-fold 원리



k-Fold 교차 검증 과정

K=5인 경우, 데이터셋을 5개의 폴드로 나눈 후, 각 폴드를 한 번씩 검증 데이터로 사용하며

나머지 4개의 폴드를 훈련 데이터로 사용

이 과정을 통해 최적의 성능을 내는 모델을 찾은 후, 마지막으로 테스트 데이터로 평가를 진행