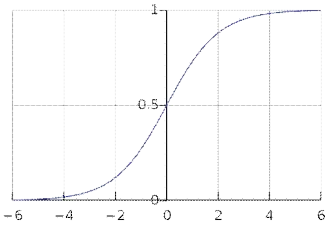
Activation Function

- 정의 : 이전 층의 결과값을 변환하여 다른 층의 뉴런으로 신호를 전달
  -> 활성 함수를 사용하면 입력값에 대한 출력값이 비선형적으로 나오기에
  선형분류기를 비선형분류기로 만들 수 있음(비선형 문제를 해결하는데 중요한 역할)
- 종류 : Sigmoid , tanh(쌍곡선 함수), ReLU, softmax, Leaky ReLU, ELU, Maxout
- 특징

(1) Sigmoid : $sigmoid(x) = \frac{1}{1+e^{-x}} \rightarrow \frac{d}{\partial x} sigmoid(x) = \frac{1}{1+e^{-x}}(1-\frac{1}{1+e^{-x}})$

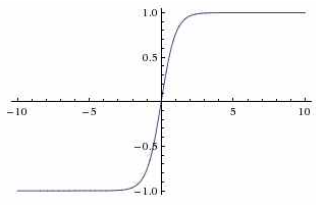
- 출력값이 항상 0과 1 사이 -> 확률을 나타낼 때 적합
- 이진 분류 : 주어진 입력값이 특정 클래스에 속할 확률을 계산
- S자 곡선 : 미분 가능 형태 -> 최적화가 용이
- 범위 : 0 ~ 1 / 중앙값 : 0.5 / 미분 최댓값 : 0.25



- 단점
  1. 기울기 소실
  -> 미분 함수에 대해 값이 일정 이상 커지면 미분값 소실
  -> 입력값이 매우 크거나 작다면 기울기가 0에 가까워짐
  -> 가중치 업데이트 속도가 늦어짐
  2. 포화 상태
  -> 활성화 함수의 출력이 1이나 0에 매우 가까워짐
  -> 가중치와 편향치를 업데이트하기 어려워져 신경망의 학습 능력 제한됨

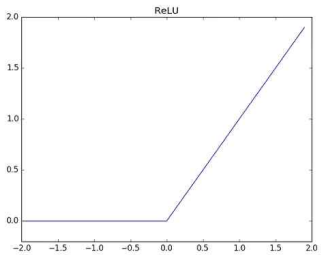
(2) tanh : $\tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \rightarrow \frac{d}{dx}\tanh(x) = 1-\tanh(x)^2$

- 입력신호를 (-1, 1) 사이의 값으로 정규화
+ 정규화 : 데이터를 특정 범위로 변환하여 범위를 일치시킴
- 범위 : -1 ~ 1 / 중앙값 : 0 / 미분 최댓값 : 1
- 단점 : 기울기 소실 (시그모이드 함수보다 기울기 소실은 적음)
+ 시그모이드 함수에서 발생하는 편향 이동은 발생하지 않음

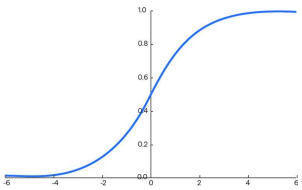


(3) ReLU : $ReLU(x) = \max(0, x)$

- 양수면 입력값을 출력하고 아니면 0을 출력
- 미분값(기울기)가 0이나 1이기 때문에 기울기 소실 없음
- 계산 효율이 6배 정도 좋음
- 단점 : zero-centerd가 아니고 음수 영역에서 포화 상태
+ non-zero-centerd : 활성함수의 결과값이 0이 아님
-> zig-zag update 발생



(4) Softmax : $softmax(x)_i = \frac{e^{x_i}}{\sum_{j=1}^{k} e^{x_i}}$

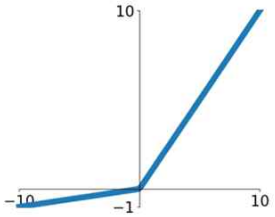


- 입력받은 값의 출력을 0 ~ 1 사이의 값으로 모두 정규화
- 출력 값들의 총합 = 1
- 미분 가능하고 알고리즘을 사용하여 신경망을 훈련하는데 중요
- 입력 값이 증가하면 해당 확률도 증가

  -> 출력층에서 이중 분류면 sigmoid, 다중 분류면 softmax 사용

+ sigmoid & softmax : 수학적으로는 같은 함수로 softmax가 sigmoid가 일반화된 형태

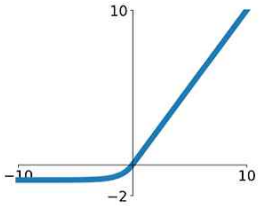
(5) Leaky ReLU : $\text{LeakyReLU}_a(x) = \max(ax, x)$



- ReLU의 변형으로 입력값이 음수일 때도 완전이 0이 아님

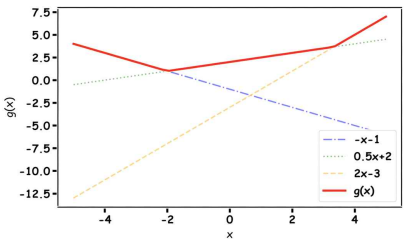
  -> 비활성화였던 뉴런의 정보를 유지하고 활용 가능
- 입력이 양수이든 음수이든 기울기가 소멸되지 않음
- 계산 쉬움
- zero-centered에 가까움

(6) ELU(Exponential Linear Units) : $\begin{cases} x & x \geq 0 \\ a(e^x - 1) & x < 0 \end{cases}$



- 음의 값에서 기울기를 가짐 -> 기울기 소실 문제 해결
- 아주 큰 음숫값에서 함숫값이 커지지 않아 노이즈에 덜 민감
- 단점 : 계산이 느림(훈련하는 동안은 수렴 속도가 빨라서 상쇄 / 테스트 시에는 느림)

(7) Maxout : $\max(w\frac{T}{1}x + b_1, w\frac{T}{2}x + b_2)$

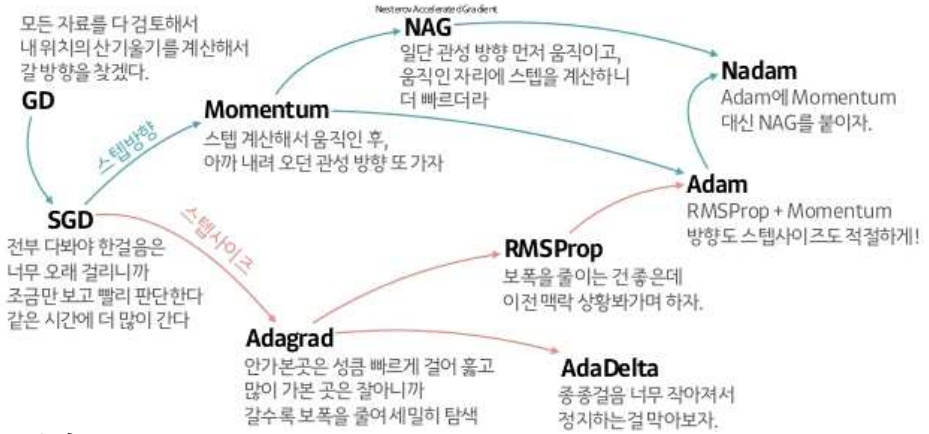


- 입력을 받아들이는 특정한 기본 형식을 미리 정의 x
- ReLU와 Leaky ReLU의 좀 더 일반화된 형태
- 포화상태가 되지 않고 기울기가 있음
- 단점 : 계산량이 복잡 -> 뉴런 당 파라미터수가 2배가 됨

뉴런별로 선형 함수를 여러 개 학습 후 최댓값을 취함

최적화 함수

- 최적화 : 목적함수를 최대한, 혹은 최소화하는 파라미터 조합을 찾는 과정
- 종류 : Gradient Descent, Momuntum, RMSProp, Adam etc...

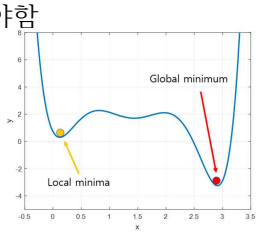


- 특징

(1) 경사 하강법

- 정의 : 함수의 값이 낮아지는 방향으로 독립 변수들의 값을 변형시키면서 함수가 최솟값을 갖도록 하게 하는 독립변수의 값을 찾는 것
- 최적의 학습 패턴을 위해 자신의 파라미터를 검증 -> 검증 과정에서 손실 함수의 값이 가장 낮은 파라미터를 발견했다면 해당 파라미터가 최적 파라미터
- 기울기가 음수면 파라미터를 증가시켜 최솟값을 찾고, 기울기가 양수면 파라미터를 감소시켜 최솟값을 찾음
- 문제점

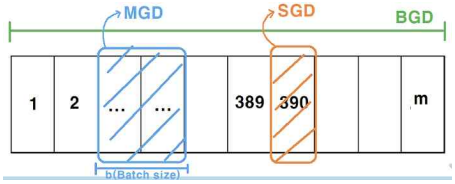
    1. step size 선정 : 과도하게 설정 시 손실 함수의 최솟값을 계산하기 어렵고, 작게 설정 시 소요되는 시간이 오래 걸림 -> 적절하게 설정 해야함
    2. Local minima(지역 극솟값)
    - 최적의 파라미터를 찾기 위해 최저점을 찾아야 하지만, 경사 하강법은 알고리즘이 시작되는 파라미터 위치가 랜덤이라 지역 극솟값에 빠질 수 있음



- 종류

    1. 배치 경사하강법 : 전체 데이터 샘플 기울기 계산
    2. 확률적 경사하강법 : 전체 훈련 데이터 세트를 무작위화 한 다음 하나의 데이터에 대해서만 손실함수에 대한 기울기를 계산 후 업데이트
    3. 미니배치 경사하강법 : 하나의 미니 배치에 속해 있는 데이터들에 대해 기울기 구한 후 평균 기울기를 통해 매개변수 업데이트

    -> BDG : 전체 / SGD : 하나 / MGD : 설정한 사이즈



(2) 모멘텀 : $v_t = \gamma v_{t-1} + \eta \nabla f(x_{t-1})$

- 경사하강법의 한계점을 개선함
- 외부에서 힘을 받지 않으면 정지해 있거나 운동 상태를 유지하려는 성질
  ->경사 하강법으로 이동할 때 관성을 부여하는 최적화 기법

(3) AdaGrad(적응적 기울기) : $g_t = g_{t-1} + (\nabla f(x_{t-1}))^2 / x_t = x_{t-1} - \frac{\eta}{\sqrt{g_t + \epsilon}} \cdot \nabla f(x_{t-1})$

- 특성을 고려하여 학습을 효율적으로 도움 -> 큰 기울기를 가져 학습이 많이 된 변수는 학습률을 감소, 학습이 적게된 변수는 학습률을 높힘
- 단점 : 학습이 오래 진행되어 학습률이 0에 가까워지면 학습이 진행 안됨

(4) RMSProp : $g_t = \gamma g_{t-1} + (1-\gamma)(\nabla f(x_{t-1}))^2 / x_t = x_{t-1} - \frac{\eta}{\sqrt{g_t + \epsilon}} \cdot \nabla f(x_{t-1})$

- AdaGrad의 단점인 학습률이 0으로 수렴하여 학습이 더 이상 진행 안되는 한계 보완
- 변수별로 학습률을 조절하되 기울기를 지수이동평균을 활용하여 업데이트
- 변수마다 적절한 학습률을 적응해 효율적 학습을 진행가능 / 학습을 더 오래 할 수 있음

(5) Adam

- 경사하강법 알고리즘을 기반으로 하고 모멘텀과 학습률 감소와 같은 개선된 기능을 추가함
- 기울기의 지수이동평균을 사용하여 학습률 조절

K-Fold

- 정의 : 집합을 체계적으로 바꿔가며 모든 데이터에 대해 모형의 성과를 측정하는 검증 방식
  -> K겹 : 전체 데이터셋을 K개의 부분집합으로 나눔
  ex) 1번 fold, 2번 fold, 3번 fold, ... k번 fold
  -> 교차 : fold들을 차례대로 교차해 테스트 데이터로 사용
  -> 검증 : 모델이 데이터를 잘 학습했는가 확인
- 모델의 성능 : 총 k개 나온 결과물들의 평균값
- 장점 : 모든 데이터가 한번씩 테스트 데이터로 사용되어 데이터를 효율적으로 사용가능
- 단점 : k값이 크면 연산 비용 증가