신경망

학부연구생 이현재

퍼셉트론

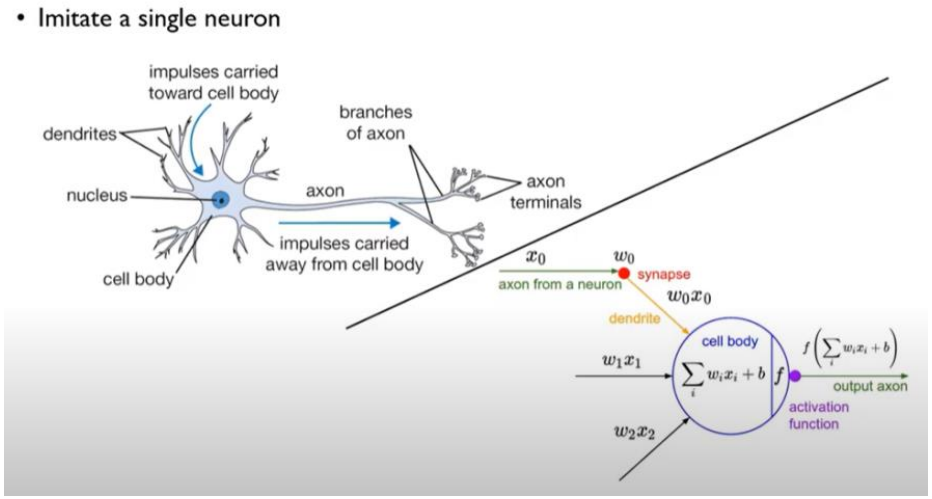
- 사람이 뇌에서 정보를 처리하는 활동을 묘사하는 기계를 만들고 싶다는 아이디어 기반
- 다수의 신호(input)을 입력 받아서 하나의 신호(output)를 출력
  - 이런 과정이 마치 뉴런이 전기 신호를 내보내 정보를 전달하는 것과 비슷
  - 실제 뉴런이 신호를 전달하는 역할을 퍼셉트론은 가중치(weight)가 그 역할을 수행
  - 각각의 입력 신호(input)에 부여되어 입력 신호와의 계산을 하고 신호의 총합이 정해진 임계값(θ; theta,세타)을 넘었을 때 1, 넘지 못하면 0을 출력

Output

$$w_1x_1 + w_2x_2 + \ldots + w_nx_n > \theta \Rightarrow 1$$

$$w_1x_1 + w_2x_2 + \ldots + w_nx_n \leq \theta \Rightarrow 0$$

  - w는 가중치, x는 입력이며, 이 둘의 선형 결합과 임계값을 비교하여 output 출력.(더 정확하게 설명하면 선형 결합 뒤에 활성화 함수를 통과하는데 이는 나중에 설명)



퍼셉트론

f는 계속 작아져야한다.

- 퍼셉트론의 학습 방법
  - 가중치는 임의로(Random) 초기화
  - 가중치에 대한 선형 결합의 스칼라 y^를 구한 뒤에 실제 y값과 비교한 뒤에 손실 함수를 계산
  - 손실 함수를 낮추는 방향으로 경사 하강법을 적용하여 최적의 가중치 찾기
- 경사하강법
  - 현재 위치 x_{old}에서 내려 가야할 방향과 보폭을 구해야 함
  - 우리가 만들고 있는 모델을 f, 방향을 f', 보폭(=학습률)을 α(0< α<1)으로 하면, 다음 위치 x_{new}는 이렇게 정의

$$x_{new} = x_{old} - \alpha f'(x_{old})$$

이것은 우리가 이전에 공부한 경사하강법과 똑같음

  - 테일러 급수를 활용한 경사하강법 정의

동영상에서는 Δx의 크기가 작기 때문에 제곱하면 0에 가깝고 그래서 그 뒤에는 다 0으로 처리한다고 하였음

$$f(x+\Delta x) = f(x) + \frac{f'(x)}{1!}\Delta x + \frac{f''(x)}{2!}\Delta x^2 + \dots$$

이 식을 적용하기 위해서는 Δx의 크기가 매우 작아야한다고 함

다른 풀이

$$f(x_{new}) = f\left[x_{old} - \alpha f'(x)\right]$$

$$= f(x_{old}) + \frac{f'(x_{old})}{1!}\left[-\alpha f'(x_{old})\right] + \frac{f''(x_{old})}{2!}\left[-\alpha f'(x_{old})\right]^2 + \dots$$

$$\cong f(x_{old}) - \alpha\left[f'(x_{old})\right]^2 < f(x_{old})$$

결국 이 식에 의해서 계속 오차를 줄여나가면 언제 멈출지 고민했었는데, f'가 0이 되면 결국 xnew = xnew가 되기 때문에 멈췄다고 이해하였습니다.

2번째 식을 3번째 식으로 근사해서 표현 가능(근사할 수 있는 이유는 따로 이미지 첨부)

세 번째 식을 확인하면 $f'(x_{old})$는 항상 양수이기 때문에 $f(x_{new}) < f(x_{old})$가 성립. 이는 경사하강법을 통해 오차(손실함수)를 계속해서 줄일 수 있다는 것을의미

Prof. Hyerim Bae (hrbae@pusan.ac.kr)

퍼셉트론

- 가중치와 편향 도입
  - 2페이지의 퍼셉트론 식에서 세타(θ)를 –b(편향)로 치환하여 좌변으로 넘기면 다음과 같이 정리 가능(입력, 가중치가 각각 2개로 가정)

    b + w1x1 + w2x2 <0 => 0

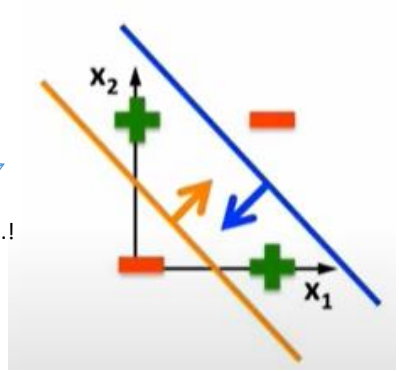
    b + w1x1 + w2x2 >=0 => 1
  - 편향은 θ(theta)로 학습 데이터(Input)이 가중치와 계산되어 넘어야 하는 임계점으로 이 값이 높으면 높을수록 그만큼 분류의 기준이 엄격하다는 것을 의미

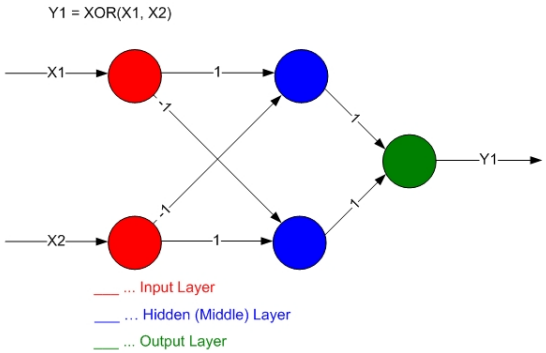
    => 그래서 편향이 높을수록 모델이 간단해지는 경향이 있으며(변수가 적고 일반화가 된 경우) 오히려 과소적합이 발생

    => 반대로 편향이 낮을수록 임계점이 낮아 데이터의 허용 범위가 넓어지는 만큼 학습데이터에만 잘 들어맞는 모델이 만들어질 수 있으며 모델이 더욱 복잡해진다 = 과적합

    (이 부분은 제가 선형 회귀에 다뤘던 분산-편향 트레이드오프와 밀접한 관계가 있습니다.)
- 퍼셉트론의 한계점
  - 퍼셉트론은 모든 학습 데이터를 정확히 분류시킬 때까지 학습이 진행되기 때문에 학습 데이터가 선형적으로 분리될 수 있을 때 적합한 알고리즘
  - 한계 : 직선 하나로 나눈 영역만 표현할 수 있어 XOR과 같은 데이터 형태는 분류가 불가능

저는 이 설명이 신박(?)했습니다..!





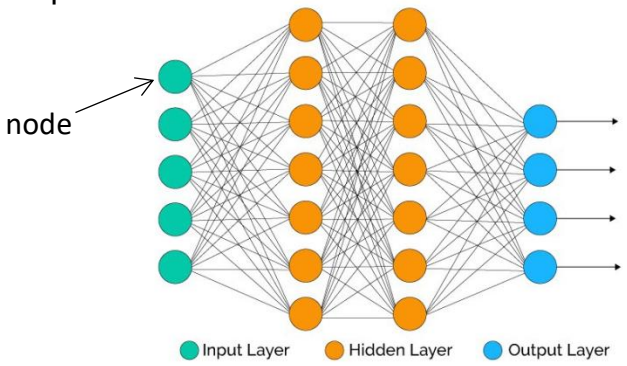
이를 극복하기 위해, 왼쪽 그림처럼 왼쪽 하단을 구분하는 퍼셉트론 하나, 오른쪽 상단을 구분하는 퍼셉트론을 만들어 이 둘을 적절하게 조합해 사이 공간을 분류하는 모델을 생성

=> 다층 퍼셉트론(MLP)

Prof. Hyerim Bae (hrbae@pusan.ac.kr)

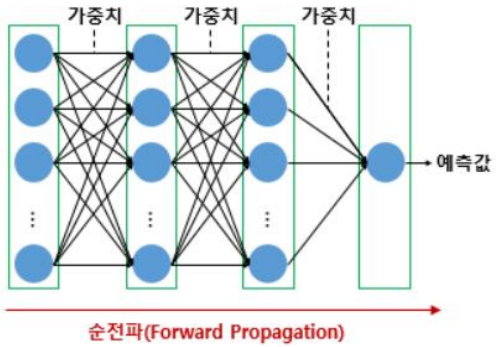
다층 퍼셉트론 (Multi Layer Perceptron , MLP)

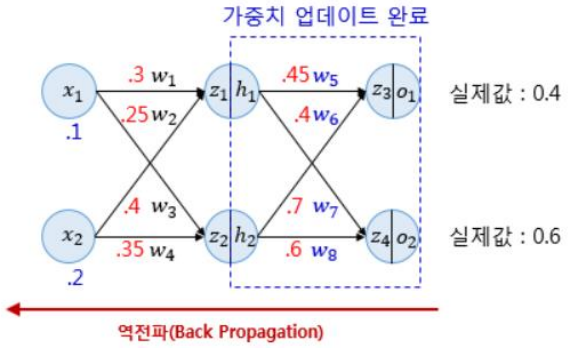
- 단층 퍼셉트론이 해결하지 못했던 비선형 분류를 여러 개를 조합해서 해결
- 구조



- 입력층 / 은닉층 / 출력층으로 구성
- 은닉층이 2개 이상인 신경망을 심층 신경망(DNN)이라고 함
- 은닉층(hidden layer)에서 다음 은닉층으로 전달할 때, 비선형함수인 활성화 함수를 통과후 전달! 활성화 함수가 비선형 함수여야 하는 이유는 추후 설명
- 은닉층이 많을수록 더욱 복잡한 모델
- 각 층은 여러 개의 node, fully connected(모델에 따라 아닐수도 있고, drop out을 쓸 수도 있음)
- 깊이 : 신경망의 층 수, 넓이 : 각 층에 존재하는 node 수

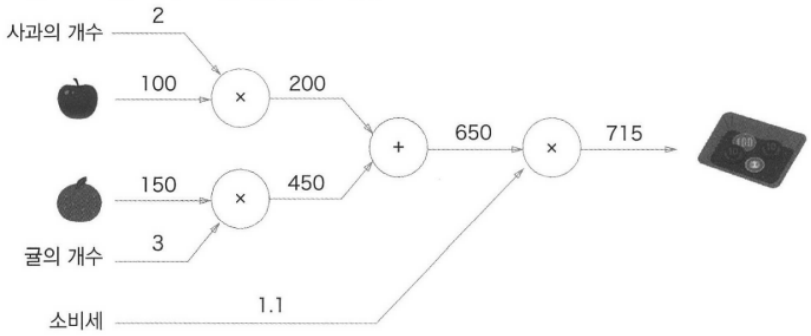
- 학습 방법
  - Forward Propagation(순전파) : 입력 x로부터 정보를 순방향으로 진행하여 예측값 y햇을 생성하여 비용함수 계산
  - Backpropagation(역전파) : 계산된 비용함수를 이용해 역방향으로 정보를 전달하여 가중치 매개변수의 기울기를 효과적으로 계산(경사하강법 사용을 위하여)





역전파 (Backpropagation)

- 가중치 매개변수의 기울기를 효율적으로 계산하는 방법
- 계산 그래프
  - 복수의 노드(node)와 에지(edge)로 표현
  - 계산 과정을 노드와 화살표로 표현. 노드는 원으로 표기하고 원 안에 연산 내용을 적음. 또, 계산 결과를 화살표 위에 적어 각 노드의 계산 결과가 왼쪽에서 오른쪽으로 전해짐

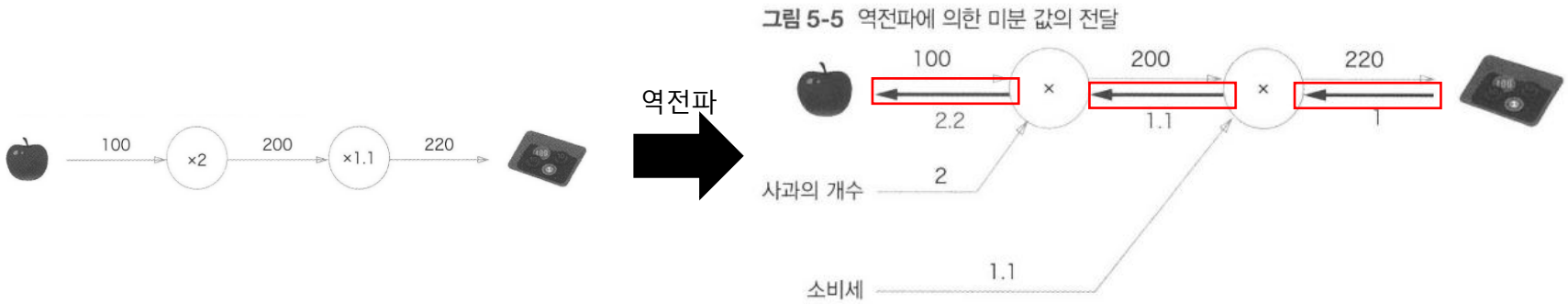


계산 그래프의 흐름
1. 계산 그래프를 구성
2. 그래프에서 계산을 왼쪽에서 오른쪽으로 진행

이 2번째 흐름이 '순전파',
그리고 이 '순전파'의 반대가 **'역전파'**

  - 계산 그래프의 특징은 '국소적 계산'을 전파함으로써 최종 결과를 얻는다는 점. 국소적 계산은 결국 전체에서 어떤 일이 벌어지든 상관없이 자신과 관계된 정보만으로 결과를 출력 => 이를 이용하여 역전파 알고리즘
  - '국소적 계산'의 예를 들면 위의 계산 그래프에서 최종 지출은 이전의 값 * 소비세이다. 그 전의 숫자가 어떻게 계산 되었는지와는 상관없이, 단지 2개의 수만 곱하면 된다. 즉, 각 노드는 자신과 관련된 계산 외에는 신경 쓸 게 없음.
  - 계산 그래프를 사용하는 가장 큰 이유는 역전파를 통해 '미분'을 효율적으로 계산하기 위함

역전파 (Backpropagation)



- 이 예에서 역전파는 오른쪽에서 왼쪽으로 '1 → 1.1 → 2.2'순으로 미분값을 전달
  - 이 결과로부터 '사과 가격에 대한 지불 금액의 미분' 값은 2.2라는 것을 알 수 있고, 사과가 1원 오르면 최종 금액이 2.2원 오른다는 뜻

- 계산 그래프의 순전파는 계산 결과를 왼쪽에서 오른쪽으로 전달
- 역전파는 **'국소적인 미분'**을 순방향과는 반대인 오른쪽에서 왼쪽으로 전달 => '국소적 미분'을 전달하는 원리가

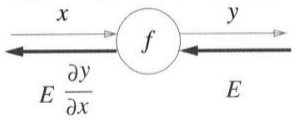
연쇄 법칙

Prof. Hyerim Bae (hrbae@pusan.ac.kr)

연쇄 법칙 (Chain Rule)

- 계산 그래프의 역전파

그림 5-6 계산 그래프의 역전파 : 순방향과는 반대 방향으로 국소적 미분을 곱한다.



- 역전파의 계산 절차는 그림과 같이 신호 E에 노드의 국소적 미분($\frac{\partial y}{\partial x}$)을 곱한 후 노드로 전달하는 것
- 국소적 미분이란 순전파 때의 y = f(x) 계산의 미분을 구한다는 것, 이는 x에 대한 y의 미분을 구하는 것 (영어로 하면 local gradient)
- 이 국소적인 미분을 상류에서 전달된 값(이 예제에서는 신호 E)에 곱해 앞쪽(오른쪽에서 왼쪽) 노드로 전달

- 연쇄 법칙
  - 역전파가 가능한 원리
  - 합성함수 : 여러 함수로 구성된 함수 ex) $z = t^2$
  - 합성 함수의 미분은 합성 함수를 구성하는 각 함수의 미분의 곱으로 나타낼 수 있기 때문에 연쇄법칙이 가능
  - $\frac{\partial z}{\partial t}$(z의 t에 대한 미분)은 $\frac{\partial z}{\partial t}$(z의 t에 대한 미분)과 $\frac{\partial t}{\partial x}$(t의 x에 대한 미분)의 곱으로 나타낼 수 있다.

$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial t}\frac{\partial t}{\partial x} = \frac{\partial z}{\cancel{\partial t}}\frac{\cancel{\partial t}}{\partial x}$$

예) $z = t^2$, $t = x + y$

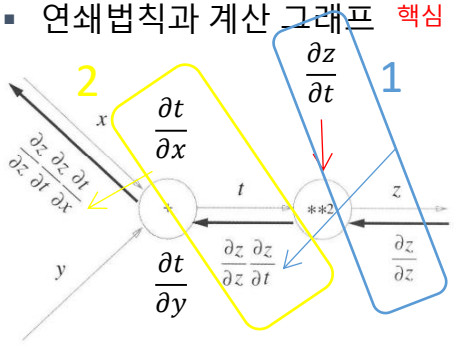
$$\frac{\partial z}{\partial t} = 2t, \quad \frac{\partial t}{\partial x} = 1 \Rightarrow \frac{\partial z}{\partial x} = \frac{\partial z}{\partial t}\frac{\partial t}{\partial x} = 2t \cdot 1 = 2(x + y)$$

Prof. Hyerim Bae (hrbae@pusan.ac.kr)

연쇄 법칙 (Chain Rule)

Input(혹은 그 전 노드의 gradient)을 global gradient, 현재 노드의 미분을 local gradient라고 한다.
즉, 역전파에 의해 다음 노드의 기울기는 global gradient * local graident가 된다.

- 연쇄법칙과 계산 그래프 핵심



첫 번째의 역전파를 확인하면, 입력은 $\frac{\partial z}{\partial z}$ 이며, 이에 국소적인 미분 $\frac{\partial z}{\partial t}$를 곱하고 다음 노드로 넘김

결국 가장 중요한 것은 맨 왼쪽 역전파인데 , 역전파에 의해서 다음이 성립
즉, 역전파가 하는 일은 연쇄 법칙의 원리와 같다.

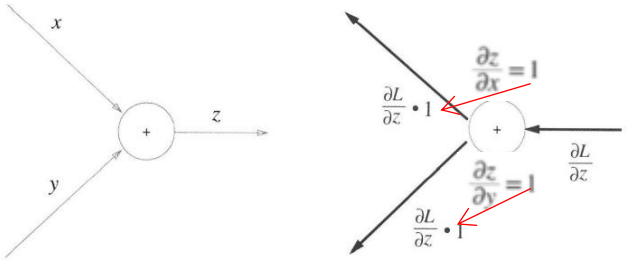
$$\frac{\partial z}{\partial z}\frac{\partial z}{\partial t}\frac{\partial t}{\partial x} = \frac{\partial z}{\partial t}\frac{\partial t}{\partial x} = \frac{\partial z}{\partial x}$$

- 덧셈 노드의 역전파

z = x + y

$$\frac{\partial z}{\partial x} = 1$$

$$\frac{\partial z}{\partial y} = 1$$



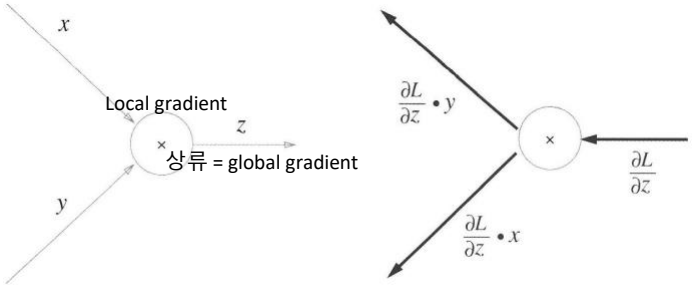
덧셈 노드의 역전파는 결국 1을 곱하기 때문에
양갈래 길로 나뉘어도 둘이 같다.

- 곱셈 노드의 역전파

z = xy

$$\frac{\partial z}{\partial x} = y$$

$$\frac{\partial z}{\partial y} = x$$



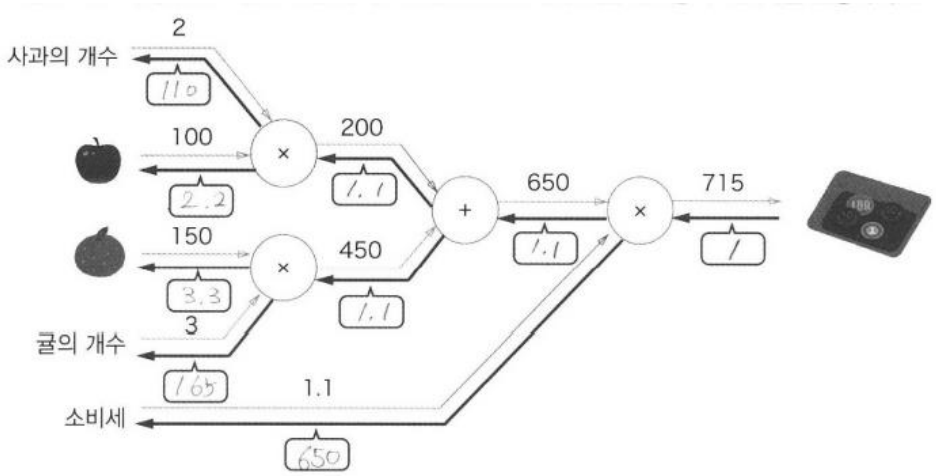
곱셈 노드의 역전파는 상류의 값에 순전파 때의
입력 신호들을 '서로 바꾼 값'을 곱해서 전달

'서로 바꾼 값'이란 순전파 때 x였다면 역전파에
서는 y, 순전파 때 y였다면 역전파에서는 x로 바
꾸는 것을 의미한다.

Prof. Hyerim Bae (hrbae@pusan.ac.kr)

연쇄 법칙 (Chain Rule)

- 예



처음 시작은 1!
그리고 앞에서 배운 개념을 적용하면
충분히 이해하실 수 있습니다!!

간단하게 요약하면,
덧셈 노드의 역전파일 때는 그대로,
뺄셈 노드의 역전파일 때는 서로를 바꿔서 곱해주면 됩니다.

- 사용하는 이유 : 다음 노드의 미분(기울기) 값을 구하기 위해서 그 전의 계산된 값만을 활용한다는 점에서 계산량을 줄일 수 있다. 마치 Dynamic programming처럼 k+1 값을 구하기 위해 k값을 이용하는 것인데 이를 통해서 계산량을 줄여 효율적으로 미분값을 구할 수 있다.

(그리고 구한 미분값을 경사하강법을 통해 수정)

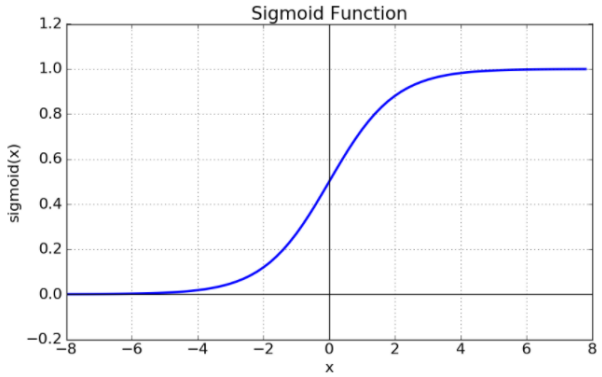
Prof. Hyerim Bae (hrbae@pusan.ac.kr)

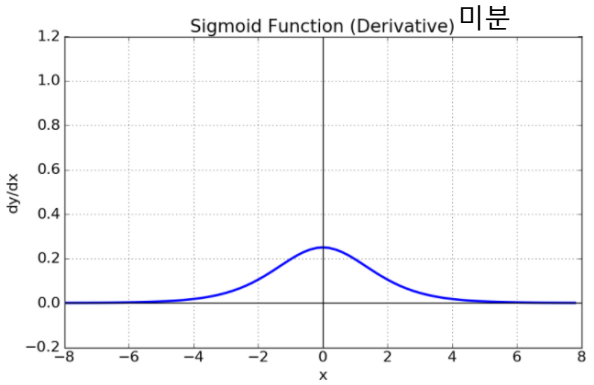
활성화 함수

- 비선형함수를 활성화 함수로 사용해야 하는 이유?
  - 만약에 선형함수를 이용하면 신경망의 층을 깊게 하는 의미가 없어지기 때문
  - 선형 함수의 문제는 층을 아무리 깊게 해도 '은닉층이 없는 네트워크'로도 똑같은 기능을 할 수 있다는 점
  - 예를 들면, 선형 함수인 h(x) = cx를 활성화 함수로 사용한 3층 네트워크를 만든다 했을 때, 이를 식으로 나타내면 y(x) = h(h(h(x)))가 된다. 이 계산은 y(x) = c * c * c * x처럼 선형 함수를 3번 작용하지만 결국 이것은 y = ax와 똑같다. c^3 = a로 표현하면 은닉층이 없는 네트워크로 표현 가능
  - 그래서 층을 쌓는 혜택을 얻고 싶다면 활성화 함수로는 반드시 비선형 함수를 사용해야 한다.
- 시그모이드 함수

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

$$\sigma'(x) = \sigma(x)(1-\sigma(x))$$





  - 특징으로 함수값이 (0, 1)로 제한, 중간 값은 ½ , 매우 큰 값을 가지면 함수값은 거의 1, 매우 작은 값을 가지면 거의 0
  - 단점
    - Gradient Vanishing : x의 크거나 작으면 미분값이 0이다. 미분 값이 0이 되는 경우가 있기 때문에 Gradient Vanishing이라고 한다. 그리고 미분 최대값이 0.25이기 때문에 계속 반복하다 보면 0에 수렴할 수 있는 문제점이 있다.
    - 함수의 중심값이 0이 아니다.(not zero-centered) : 이유는 따로 이미지로 표기
    - Exp 연산의 비용이 높음

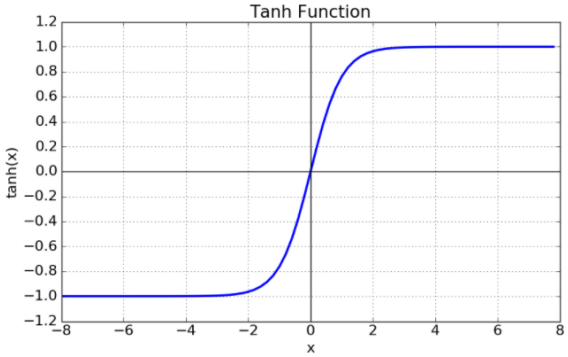
Prof. Hyerim Bae (hrbae@pusan.ac.kr)

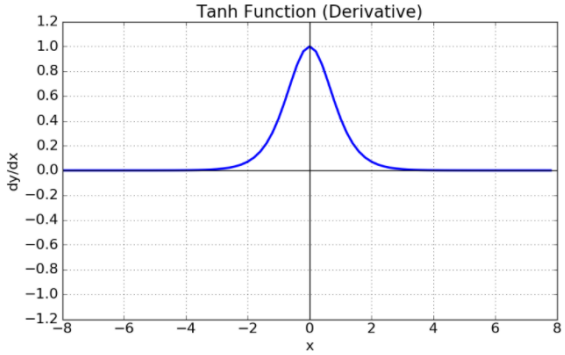
활성화 함수

- Tanh 함수 **(Hyperbolic tangent function)**

$$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

$$tanh'(x) = 1 - tanh^2(x)$$





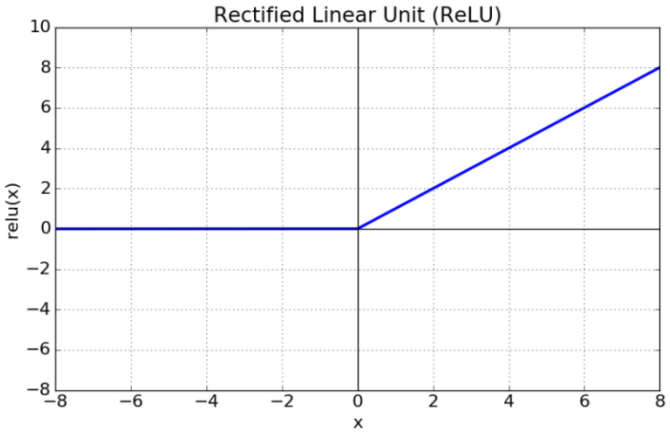
  - Tanh 함수는 함수의 중심값을 0으로 옮겨 시그모이드의 최적화 과정이 느려지는 문제를 해결
  - 하지만 여전히 일정값 이상 커지거나 작아질 때, 미분값이 소실되는 gradient vanishing은 여전히 발생

- 이러한 gradient vanishing의 문제 때문에 출력층으로 시그모이드 혹은 tanh 함수를 사용하지 않는다. 즉, 층이 깊어질수록 0에 수렴할 가능성이 높기 때문에 사용하면 안된다.
- 시그모이드 함수는 값을 0~1로 만들어주기 때문에 가공된 데이터를 다음 은닉층으로 보내줘도 극단적인 형태가 아니고 신경망이 데이터를 섬세하게 분류할 수 있도록 도와준다.

활성화 함수

- Relu 함수

$$f(x) = max(0, x)$$



- 최근 가장 많이 사용되는 활성화 함수
- Input이 0보다 크면 그 input 그대로 반환하고, 0보다 작으면 0을 반환하는 함수
- Sigmoid, tanh 함수보다 학습이 빠르고 연산 비용이 크지 않고 구현이 매우 간단(exp 사용을 하지 않기 때문에)
- Gradient Vanishing 문제가 발생하지 않는다 => 출력값의 범위가 넓고, 양수인 경우 자기 자신을 그대로 반환하기 때문에
  (직관적으로 0보다 작은 것은 다 버리고 0보다 큰 것에 대해서는 기울기가 1이기 때문에 미분값을 유지)
- 한계점
  - 가중치가 업데이트 되는 과정에서 가중치의 합(선형결합)이 음수가 된다면 Relu는 0을 반환하기 때문에 해당 뉴런은 그 이후로 아무것도 변하지 않는(학습하지 않는) 현상이 발생할 수 있다. => 이러한 현상을 Dying Relu라고 함
  - Relu 함수는 Gradient Vanishing 방지를 위해 사용하는 활성화 함수이기 때문에 은닉층에서 사용하는 것을 추천
  - Relu의 출력값은 0 혹은 양수, Relu의 미분 값은 0 또는 1이다. 이로 인해, 가중치 업데이트시 지그재그로 최적의 가중치를 찾아가는 지그재그 현상은 여전히 있다.
  - Relu는 0에서 미분 불가능(0에 걸릴 확률이 적으니 무시하고 사용한다고 한다.)

Prof. Hyerim Bae (hrbae@pusan.ac.kr)

활성화 함수

- Leakly Relu
  - Relu의 뉴런이 죽는(Dying Relu) 문제점을 해결하기 위해 나온 함수

$$f(x) = max(0.01x, x)$$

  - 음수의 x 값에 대해 미분값이 0이 되지 않는다는 점을 제외하면 Relu와 같은 특성을 지닌다.
- PRelu

$$f(x) = max(\alpha x, x)$$

  - Leakly Relu와 거의 유사하지만 새로운 파라미터 a를 추가하여 x가 음수인 기울기를 학습할 수 있게 하였다.
- Exponential Linear Unit(ELU)
  - 가장 최근에 나온 함수

$$f(x) = x \quad if \quad x > 0$$

$$f(x) = \alpha(e^x - 1) \quad if \quad x \leq 0$$

  - Relu의 모든 장점 포함, Dying Relu 문제 해결, 출력값이 거의 zero-centered에 가깝다, 일반적인 Relu와 달리 exp 계산을 처리하기 때문에 비용이 높다.
- Maxout 함수
  - Relu가 가지는 모든 장점을 가졌으며, Dying Relu문제 또한 해결했지만 계산이 복잡하다는 단점

$$f(x) = max(w_1^T x + b_1, w_2^T x + b_2)$$

Prof. Hyerim Bae (hrbae@pusan.ac.kr)

퍼셉트론 관련 모델 코드 구현

https://github.com/LeeYunseol/Lab_study/blob/main/%EC%8B%A0%EA%B2%BD%EB%A7%9D/%EB%B3%B4%EC%8A%A4%ED%84%B4%EC%A3%BC%ED%83%9D%EB%8D%B0%EC%9D%B4%ED%84%B0_MLP.ipynb

참고 자료

- 밑바닥부터 시작하는 딥러닝