Fundamentals of Deep Learning

Chapter 1. 신경망

* 1. 지능형 기계 만들기

뇌는 사람의 몸에서 가장 놀라운 기관. 뇌는 우리가 보고, 듣고, 냄새를 맡고, 맛보고, 만지는 모든 지각을 관장하며, 기억을 저장하고 감정을 경험하며 심지어 꿈을 꾸게 함. 어린아이의 뇌는 450g 정도에 불과하지만, 아주 강력한 슈퍼컴퓨터도 해결하지 못하는 복잡한 문제를 어떤 방법으로든 풀어낼 수 있음. 수십 년 동안 인간의 뇌와 유사한 기능을 지닌 지능형 기계의 탄생을 꿈꿔 옴. 그러나 이런 인공지능 기계를 만드는 것은 지금까지 씨름해 온 가장 복잡한 계산 문제들을 먼저 해결해야만 가능함. 이 문제들을 해결하기 위해서는 지난 10년 동안 크게 발전해 온 기술들을 사용해 지금까지와는 완전히 다른 컴퓨터 프로그래밍 방식을 개발해야만 함. 이것이 바로 인공지능 연구가 매우 활발한 Deep learning 분야임.

* 1. 기존 컴퓨터 프로그램의 한계

특정 문제들을 컴퓨터로 해결하기가 매우 어려운 이유가 무엇일까? 기존 컴퓨터 프로그램들은 **매우 빠르게 연산하기**와 **명시적**으로 명령어 리스트 수행하기 이 두가지에 아주 능하도록 설계됨.

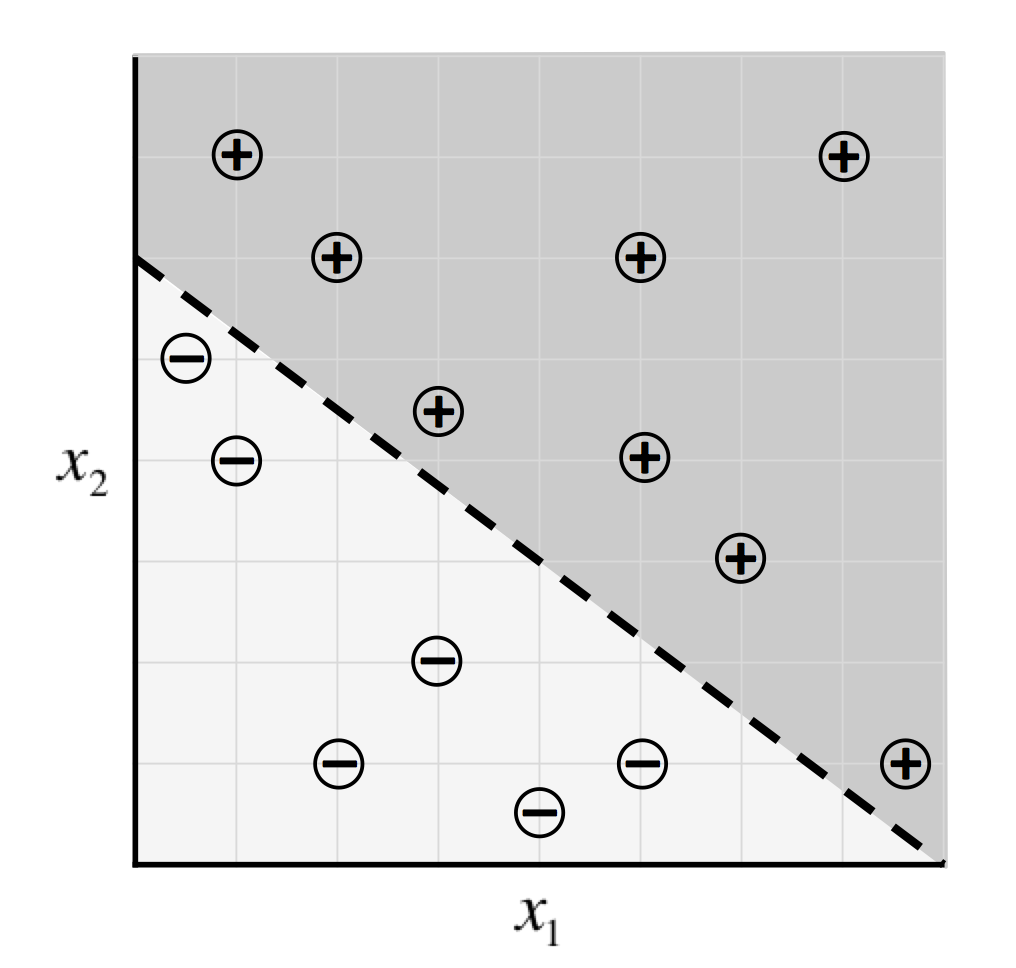
* 1. 머신러닝의 작동 원리

우리가 자라면서 학교에서 배운 많은 것은 기존 컴퓨터 프로그램과 공통점이 많음. 명령어들의 집합을 내제화해 숫자를 곱하고, 방정식을 풀고, 미분하는 것을 배움. 그러나 아주 어린 시절에 배운 것들, 가장 자연스럽게 받아들인 것들은 공식을 통해 배우는 것이 아니라 사례를 **경험**하면서 배우게 됨. 여러 경험을 통해 우리의 인식이 틀렸을 때 그것을 수정함으로써 어떤 것을 인지하는 방법을 배운다. 다시 말해, 뇌는 태어났을 때부터 어떻게 세상을 볼 수 있는지를 설명하는 모델을 제공함. 우리가 커가면서 그 모델은 감각적 입력을 받아들이고 경험한 것에 대한 예측을 만듬. 그 예측이 맞았다는 것을 부모님을 통해 확인받으면 우리의 모델은 강화됨.

Deep learning은 머신러닝이라는 인공지능의 더 일반적인 분야의 한 부분임. 머신러닝은 실제 사례를 통한 **경험**으로부터 배운다는 개념에 바탕을 두고 있음. 머신러닝에서는 문제를 풀기 위한 많은 규칙을 컴퓨터에 가르치는 것이 아니라 경험들을 **평가**할 수 있는 모델과 실수했을 때 모델을 수정하기 위한 작은 명령어 집합을 제공함. 그리고 **시간이 지나면서 조정된 모델**이 문제를 매우 정확하게 풀 수 있게 되리라고 기대함.

머신러닝 모델을 더 직관적으로 이해할 수 있는 간단한 예. 전날의 수면 시간과 공부 시간을 바탕을 시험 성적을 예측하고 싶음. 각 데이터 지점이 인 많은 데이터를 수집함. 여기서 수면 시간은 , 공부 시간은 ()이고, 성적이 평균보다 위인지 아래인지를 기록함. 목표는 다음과 같은 파라미터 벡터 를 가진 모델 를 학습시키는 것임.

다시 말해, 앞의 수식으로 설명되는 모델 에 대한 청사진을 추정함. 우리가 원하는 것은 입력 x가 주어지면 모델이 옳은 예측을 할 수 있게 하는 파라미터 벡터 를 학습하는 것임. 이 모델은 선형 퍼셉트론이라 불리며, 1950년대부터 사용됨. 다음과 같이 데이터를 가정해보자.



여기서 를 선택하면 모델은 모든 데이터 지점에 대해 정확한 예측을 하게 됨. 최적 파리미터 벡터 는 정확한 예측을 가능한 많이 할 수 있도록 분류기를 좌표 평면상에 배치함. 대부분의 경우 최적 를 위한 가능한 선택이 많이 있음. 다행히도 이 가능한 선택지들은 대부분 무시할 수 있을 만한 근소한 차이를 가지고 있어서 서로 매우 가까이에 위치함. 그렇지 않다면 의 선택 폭을 좁히기 위해 더 많은 데이터를 모아야 함

이 설정이 합리적인 것처럼 보이지만, 여전히 중요한 질문들이 남아 있음. 먼저, 어떻게 파라미터 벡터 에 대한 최적값을 도출할 것인가? 이 문제를 해결하려면 일반적으로 **최적화**라는 기술이 필요함. 최적화 도구는 오차가 최소화될 때까지 파라미터를 반복적으로 조정해 머신러닝 모델의 성능을 최대화하는 것은 목표로 함. 두 번째로, 이 특정 모델은 학습할 수 있는 관계가 상당히 제한돼 있음이 분명함. 데이터가 복잡한 형태일수록 이를 설명하기 위한 모델은 복잡해짐. 딥러닝 알고리즘들은 다른 머신러닝 알고리즘보다 훨씬 뛰어날 뿐 아니라 인간이 달성한 정확도와 거의 같다.

* 1. 뉴런

사람 뇌의 기본 단위는 뉴런임. 뉴런은 다른 뉴런들로부터 정보를 받아 고유한 방식으로 처리하고, 그 결과를 다른 세포에 전송하도록 최적화됨. 뉴런은 안테나와 비슷한 구조의 가지돌기를 통해 입력을 받음. 입력되는 연결이 얼마나 자주 사용되는지에 따라 동적으로 강해지거나 약해짐. 바로 이것이 뉴런의 출력에 대한 입력의 기여도를 결정하는 각 연결의 강도임. 입력은 각 연결의 강도만큼 가중된 후 신경 세포체에서 모두 더해짐. 이렇게 더해진 합은 세포의 축삭돌기를 따라 전파돼 다른 뉴런으로 보내는 새로운 신호로 변환됨.

이런 뉴런의 기능을 컴퓨터에서 표현할 수 있는 인공적 모델로 변환할 수 있음. 생물학적 뉴런과 마찬가지로, 인공 뉴런은 n개의 입력을 받고, 각각의 입력에 특정 가중치를 곱함. 이러한 가중치 입력은 신경 세포의 Logit을 생성하기 위해 앞에서처럼 합쳐짐. 많은 경우 logit은 상수인 bias를 포함함. 그 후 logit은 함수 f를 통해 전달돼 출력 y = f(z)를 생성한다. 이 출력은 다른 뉴런으로 전송될 수 있음.

벡터 형태로 뉴런의 기능을 다시 표현함으로써 인공 뉴런에 대한 수학적 논의를 끝낼 수 있음. 벡터 x = [x1 x2 ... xn]로 표현되는 입력과 w = [w1 w2 ... wn]로 표현되는 뉴런의 가중치로 재구성해보면 뉴런의 출력을 y = f(x\*w + b)로 다시 표현할 수 있음. 여기서 b는 bias임. 다시 말해, 입력과 가중치 벡터의 내적을 구하고 logit을 생성하기 위해 bias항을 더한 후 변환 함수를 적용함으로써 결과를 계산 할 수 있음.

* 1. 뉴런으로 선형 퍼셉트론 표현하기

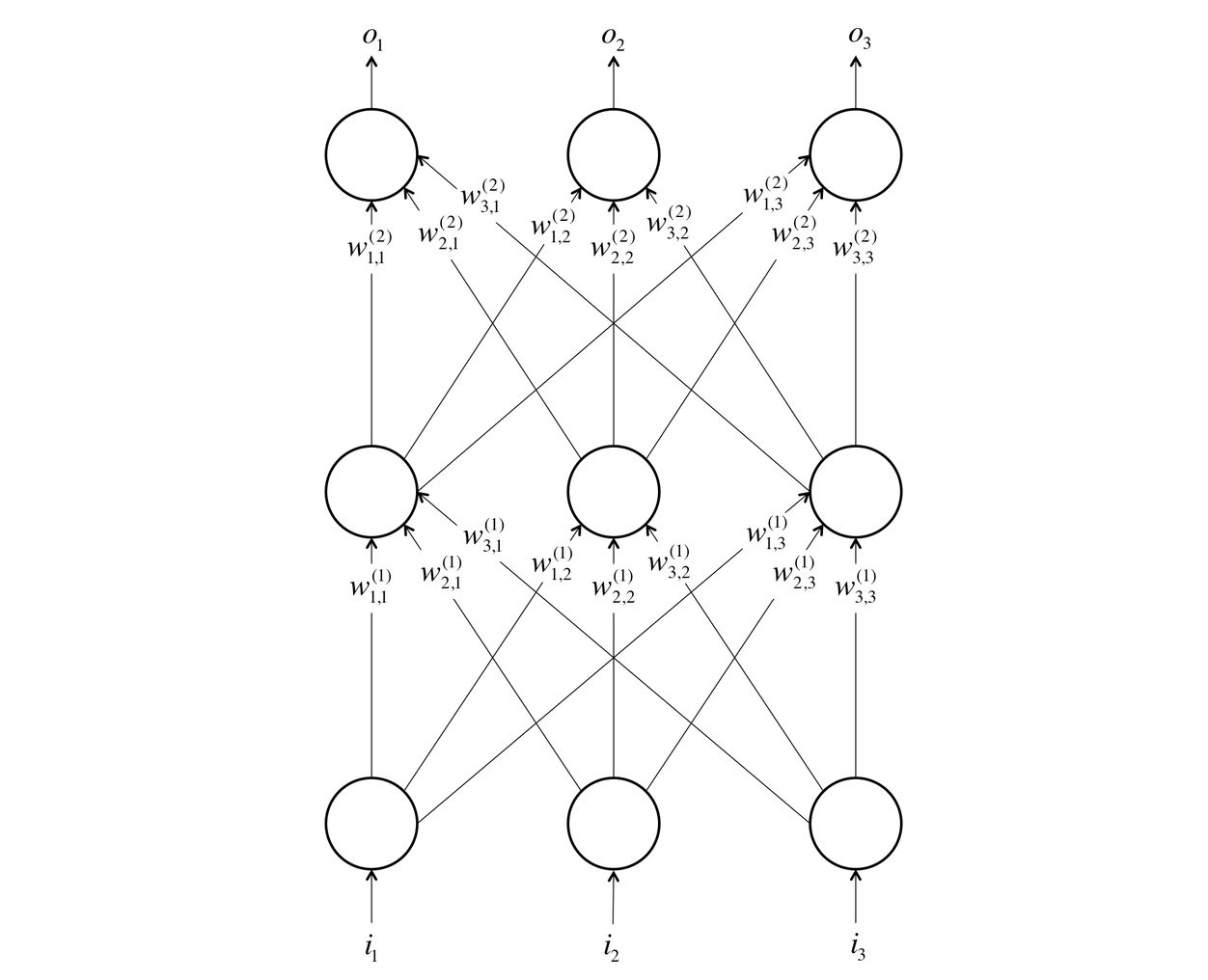
1.3에서 시험 성적과 공부 시간, 자는 시간 사이의 관계를 포착하기 위한 머신러닝 모델에 관해 이야기했음. 이 문제를 해결하고자 직교좌표 평면을 두 개로 나누는 선형 퍼셉트론 분류기를 만들었음.

1.3의 문제를 보면 우리의 선형 퍼셉트론과 뉴런 모델이 완벽하게 동등하다는 것을 쉽게 알 수 있음. 일반적으로 단일 뉴런은 선형 퍼셉트론보다 훨씬 표현력이 뛰어남. 다시 말해, 모든 선형 퍼셉트론은 하나의 뉴런으로 표현할 수 있지만, 단일 뉴런은 선형 퍼섭트론으로는 표현할 수 없는 모델은 표현할 수 있다.

1.6. 전방향 신경망

단일 뉴런이 선형 퍼셉트론보다 강력하다곤 하지만, 단일 뉴런도 복잡한 학습 문제를 해결할 만큼 충분한 표현력을 갖지는 못함. 뇌가 하나 이상의 뉴런으로 이루어진 이유가 있음. 예를 들어, 단일 뉴런으로는 손으로 쓴 숫자를 구분하는 것이 불가능함. 따라서 복잡한 문제를 해결하려면 머신러닝 모델을 발전시켜야 함.

사람 뇌의 뉴런은 층으로 구성됨. 사실 대뇌 피질은 6개 층으로 구성됨. 감각 입력이 개념적 이해로 변환될 때까지 정보는 한 층에서 다른 층으로 흐름. 예를 들어, 시각 피질이 가장 아래층은 눈으로부터 가공되지 않은 시각 정보를 받음. 이 정보는 각 층에서 처리되고 다음 층으로 전달됨. 이것은 고양이, 음료 캔 또는 비행기를 보고 있다는 결론을 내리는 최종 6번째 층까지 전달됨.



3개 층(입력층, 은닉층, 출력층)과 층당 3개의 뉴런이 있는 전방향 신경망의 간단한 예

이 개념을 빌려 인공 신경망을 만들 수 있음. 신경망은 뉴런을 입력 데이터와 학습 문제의 답에 해당하는 출력 노드에 연결해 만듬. 위 그림은 1943년에 매컬러와 피츠가 설명한 구조와 유사한 인공 신경망의 간단한 예임. 망의 가장 아래층이 입력 데이터를 가져오고, 뉴런의 최상위 층은 최종 답을 계산함. 뉴런의 중간층들은 은닉층이라고 함. 여기서 k번째 층의 j번째 뉴런과 k+1번째 층의 j번째 뉴런을 연결하는 가중치를 w라고 하면, 이 가중치들은 파라미터 벡터 를 구성함. 그리고 앞에서와 마찬가지로 신경망으로 문제를 해결하는 능력은 에 들어갈 최적값을 찾는 것에 달려 있다.

이 예에서 연결들은 아래층에서 위층으로만 이동한다는 점에 주목해야 함. 즉, 같은 층의 뉴런들 사이에는 연결이 없고 위층에서 아래층으로의 데이터 전송도 없음. 이런 신경망을 전방향 신경망이라고 함. 전방향 신경망이 가장 분석하기 쉬우므로 우리도 전방향 신경망부터 시작하기로 함. 2장에서는 이를 분석해보고 이보다 복잡한 연결들에 대해서는 뒤에 나올 장들에서 살표보겠음.

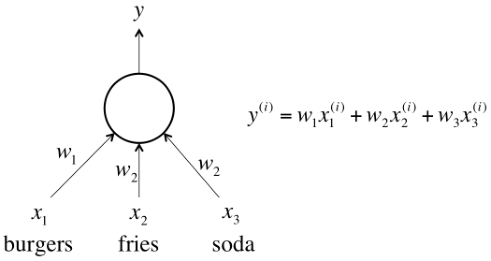
정방향 신경망에서 사용하는 층들의 주요 형태에 대해 논의하기 전에 잊어서는 안 될 다음의 몇 가지 중요한 사항을 살펴보자.

1. 앞에서 언급했듯이 첫 번째 층의 뉴런과 마지막 층의 뉴런 사이에 끼어 있는 뉴런의 층을 은닉층이라고 함. 이곳은 신경망이 문제를 해결하고자 할 때 대부분의 마법이 일어나는 곳임. 예전에는 유용한 특징은 찾는 데 많은 시간을 써야 했지만, 은닉층을 이 과정을 자동화함. 종종 은닉층에서 일어나는 활동을 들여다보면 신경망이 데이터에서 추출해 자동으로 학습해 온 특징들에 관한 많은 것을 발견할 수 있음.
2. 이 예에서는 모든 층이 같은 수의 뉴런을 가지고 있지만, 이런 구조가 꼭 필요하거나 권장되는 것은 아님. 대개 신경망은 원래 입력의 압축된 표현들을 배우도록 강제하기 위해 은닉층이 입력층보다 뉴런의 수가 적음. 예를 들어, 눈은 주변에서 원시 픽셀값을 얻지만, 뇌는 테두리와 등고선을 바탕으로 생각함. 이것은 뇌 안에 있는 생물학적 뉴런의 은닉층이 인식한 모든 것을 잘 표현하도록 우리에게 강요하기 때문임.
3. 모든 뉴런의 출력이 다음 층에 있는 모든 뉴런의 입력과 연결될 필요는 없음. 사실 어떤 뉴런이 다음 층의 어떤 뉴런과 연결될지를 결정하는 것은 경험으로부터 얻어지는 기술임. 이에 대해서는 신경망의 다양한 예를 다룰 때 자세히 설명하기로 함.
4. 입력과 출력은 벡터로 표현됨. 예를 들어 벡터로 표현된 이미지의 개별 픽셀 RBG값을 입력으로 하는 신경망을 상상해보면, 마지막 층은 문제에 대한 답을 나타내는 2개의 뉴런을 가지고 있을 것임. 이미지가 개를 포함하고 있다면 [1, 0], 고양이를 포함하고 있다면 [0, 1], 둘다 포함하고 있다면 [1, 1], 둘 다 포함하지 않는다면 [0, 0]이다.

또한, 뉴련의 재구성과 유사하게 신경망을 수학적으로 연속된 벡터와 행렬 연산으로 표현할 수 있음. 신경망에서 i번째 층의 입력을 벡터 x = [x1 x2 ... xn]라고 해보면, 우리는 뉴런을 통해 입력을 전파함으로써 생성되는 벡터 y = [y1 y2 ... yn]를 찾고 싶음. n\*m크기의 가중치 행렬 W와 크기 m인 바이어스 벡터로 구성한다면, 간단한 행렬 곱으로 이것을 표현할 수 있음. 이 행렬에서 각 열은 하나의 뉴런에 해당함. 여기서 열의 j번째 요소는 입력의 j번째 요소와 연결된 가중치에 해당함. 즉 다. 여기서 변환 함수는 벡터 요소별로 적용됨.

* 1. 선형 뉴런과 그 한계

대부분 뉴런의 유형은 로짓 z에 적용되는 함수 f에 의해 정의됨. 먼저 f(z) = az + b의 형태로 선형 함수를 사용하는 뉴런 층들을 살펴보자. 예를 들어, 패스트푸드점에서 식사 비용을 추정하려는 뉴런은 a = 1, b = 0인 선형 뉴런을 사용함. 즉, f(z) = z이고 각 항목의 가격과 동일한 가중치를 이용하면 밑의 그림의 선형 뉴런은 햄버거, 감자튀김, 탄산음료 3가지를 주문받고 조합된 가격을 출력함.

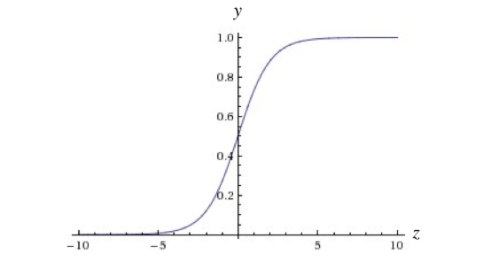


선형 뉴런은 계산하기 쉽지만 심각한 한계가 있음. 사실, 선형 뉴런으로만 구성되 전방향 신경망은 은닉층이 없는 신경망으로 표현될 수 있음. 앞서 살펴본 것처럼 은닉층이 입력 데이터로부터 중요한 특징을 배우므로 은닉층이 없는 것은 문제가 됨. 다시 말해, 복잡한 관계들을 학습하기 위해 일종의 비선형성을 채용한 뉴런을 사용해야 함.

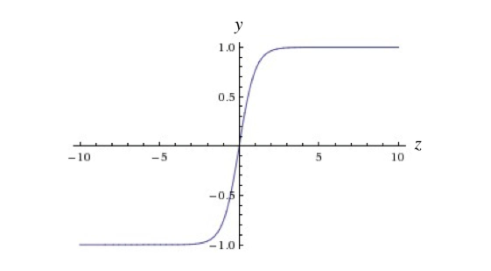
* 1. 시그모이드, tanh, ReLU 뉴런

뉴런 계산에서 실제로 사용하는 비선형성을 도입한 대표적 뉴런으로는 다음 3가지가 있음. 첫 번째는 시그모이드(Sigmoid) 뉴런으로, 이 뉴런이 사용하는 함수는 다음과 같음.

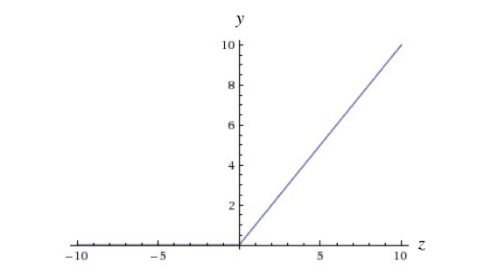
함수 형태를 미루어 보면 로짓이 아주 작을 때 로지스틱 뉴런의 결과가 0에 매우 가깝고, 로짓이 아주 클 때는 로지스틱 뉴런의 결과가 1에 가깝다는 것을 의미함. 이 뉴런은 밑의 그림처럼 S자 형태를 띰.



tanh 뉴런은 비슷한 종류의 S자 비선형성을 사용하지만, 결과의 범위가 0에서 1사이가 아니라 -1에서 1사임. 함수는 f(z) = tanh(z)임. 출력 y와 로짓 z 사이의 관계에 대한 결과는 밑의 그림에 설명되어 있음. S자 비선형성을 사용할 때 tanh 뉴런은 0이 중심이어서 시그모이드 뉴런보다 선호되는 경우가 많음.



다른 비선형성으로 ReLU(Restricted Linear Unit) 뉴런이 사용됨. 이것은 밑의 그림처럼 특징적인 하키 스틱 모양의 결과를 갖은 함수 f(z) = max(0, z)를 사용함.



* 1. 소프트맥스 출력층

출격 벡터가 상호 배타적인 레이블의 집합인 확률 분포가 되길 바라는 경우가 많음. 예를 들어, MNIST 데이터셋에서 손으로 쓴 숫자를 인식하는 신경망을 만들기를 원한다고 해보면, 각 레이블은 상호 배타적이지만, 100% 신회도로 숫자를 인식할 수 있을 것 같지는 않음. 확률 분포를 사용하는 것은 우리가 한 예측의 신뢰도가 어느 정도인지를 표현하는 좋은 방법임. 결과적으로 우리가 원하는 출력 벡터는 다음과 같은 형태가 되며, 여기서는 임.

이것은 소프트맥스(Sotfmax)층이라는 특별한 출력층을 사용해 완성함. 다른 층과 달리 소프트맥스층에서의 뉴런의 출력은 그 층의 모든 다른 뉴런의 출력에 의존하는데, 이는 모든 출력의 합이 1이 돼야 하기 때문이다. 를 i번째 소프트맥스 뉴런의 로짓이라고 하면, 이것의 출력을 다음과 같이 규정함으로써 정규화할 수 있다.

강한 예측은 출력 벡터에서 하나의 항이 1에 가깝지만, 나머지 항은 0에 가까울 것임. 약한 예측은 가능성이 거의 같은 여러 개의 가능한 레이블들을 가질 것임.