제3장: 세가지 기본 신경망I: MLP

Soyoung Park

Pusan National University
Department of Statistics

soyoung@pusan.ac.kr

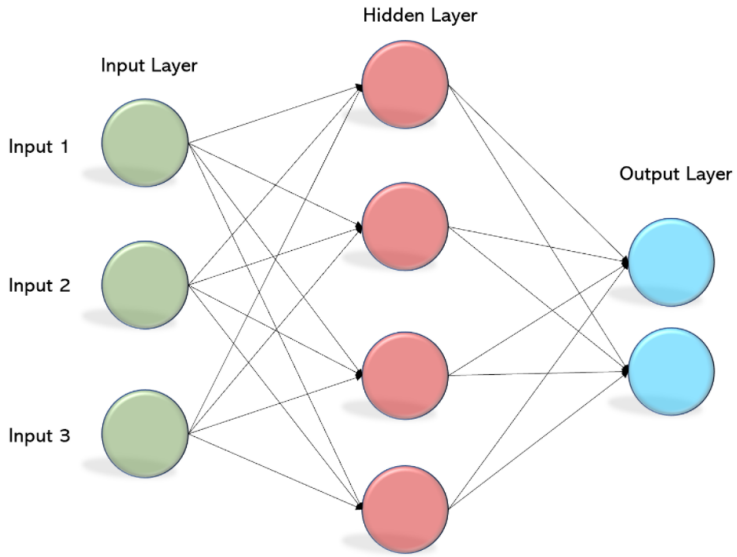
세가지 핵심 신경망과 데이터타입

세가지 핵심 신경망

- MLP (multiplayer perceptrons)
- CNN (convolutional neural networks)
- RNN (recurrent neural networks)

- MLP - 하나의 표본에 1D텐서의 특성변수
-
- CNN - 하나의 표본에 2D텐서 또는 3D텐서의 특성변수
-
- RNN - 하나의 표본에 1D텐서의 특성변수 형태로 시간스텝에 따라 반복적으로 제공
-

MLP(multilayer perceptrons)



- 1D텐서인 특성변수가 입력되면, 이를 선형결합하고, 이 선형결합에 **활성함수(activation function)**을 활용하여 비선형변환한다.
- 이 값은 0D텐서(스칼라)이며, 은닉층의 첫번째 출력값이 됨.

[1] https://becominghuman.ai/multi-layer-perceptron-mlp-models-on-real-world-banking-data-f6dd3d7e998f

MLP(multilayer perceptrons)

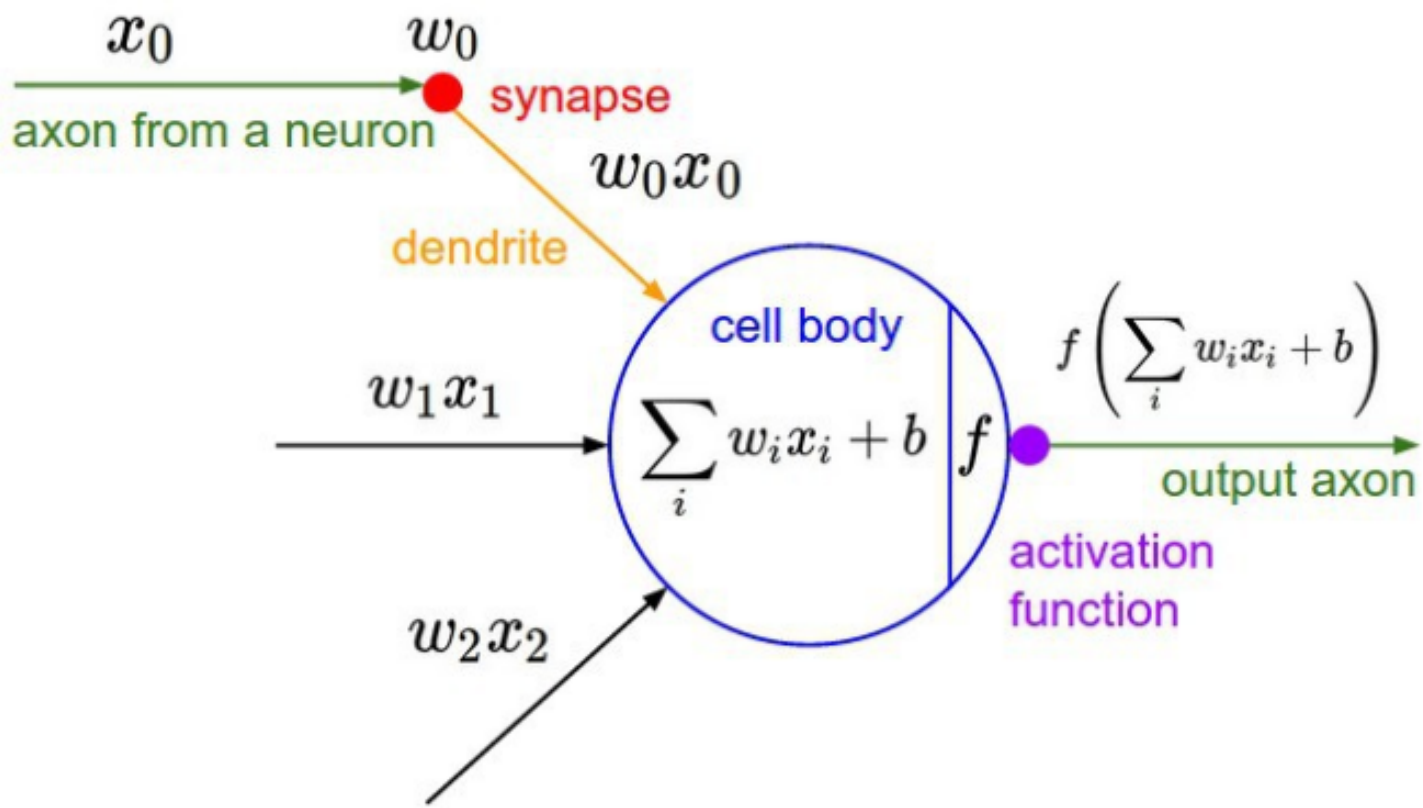
- 또 다른 선형결합과 활성함수에 의한 비선형 변환을 통해 은닉층의 두번째 출력값을 만듬
-
- 이 n_1개의 출력은 크기가 n_1이 1D텐서가 되고, 뒤따르는 다음 은닉층의 입력이 됨
- 직전 은닉층에서 했던 동일한 작업을 통해 크기가 n_2인 1D텐서를 출력하고, 다시 뒤이은 은닉층의 입력이 됨
- 미리 설정한 은닉층 숫자만큼 앞의 작업을 반복
- 최종적으로 출력층을 통해 출력하며,

활성함수(activation function)

-
-
- 퍼셉트론은 뉴런처럼 입력 받은 수치들을 계산하고, 출력하기 전에 활성화 함수를 거쳐 출력에 변화를 줌

[1] https://blog.naver.com/jaeyoon_95/222300238922

활성함수(activation function)



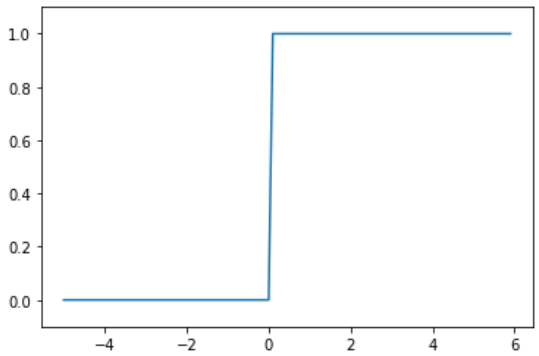
[1] https://blog.naver.com/ollehw/221520228073

활성함수(activation function)

계단함수

- 활성함수 중 가장 간단한 함수
-
-

$$h(x) = \begin{cases} 0\ (x \leq 0\ ) \\ 1\ (x > 0) \end{cases}$$



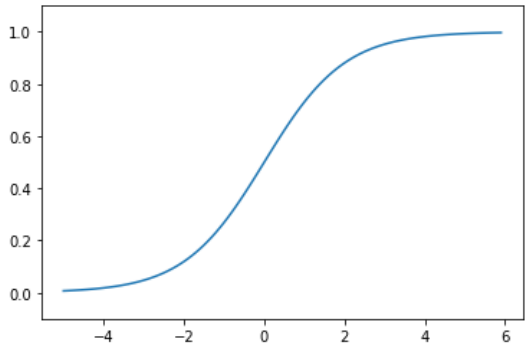
계단 함수

활성함수(activation function)

Sigmoid(Logistic)

- 목적변수가 범주형일 때 0-1사이의 값을 출력
-

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



시그모이드 함수

활성함수(activation function)

Sigmoid(Logistic)

- 계단함수의 단점을 보완하고자 등장함, S자 곡선을 갖음
-
- 시그모이드 함수는 vanishing gradient라는 문제를 발생
- 또한 데이터의 중심이 0이 아니기 때문에, 함수로 들어오는 데이터가 항상 양수인 경우, gradient는 모두 양수 또는 음수가 됨.
- 따라서 gradient를 업데이트 할 때, 지그재그로 변동하는 문제점이 발생하여 학습이 느려지고, 효율성이 감소

Vanishing Gradient

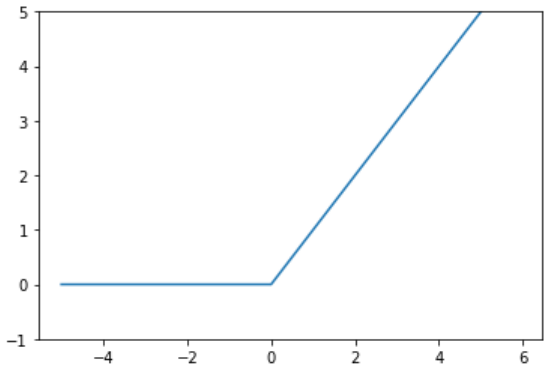
- 딥러닝은 기울기 조정해 가며, 적절한 파라미터 값을 찾아 다니는데, 앞서 말한것 처럼 은닉층이 많게 되면, 1 미만의 값들이 계속 곱해지기 때문에, 결국 기울기가 0이 되어버리는 기울기 소멸 문제가 발생
- 학습이 진행되면서 각 파라미터에 대한 가중치의 미분값(경사)가 매우 작아져 0에 가깝게 되는 현상
- 즉, 입력 값이 무한대로 커진다고 하더라도, 모델의 계층이 많을수록 gradient값이 0에 수렴하는 경향 (시그 모이드의 모든 값은 1보다 작은 수를 가지기 때문에, 계층이 많을수록 0.xxx끼리 계속 곱해지게되고, 결국에는 0에 수렴)
- 신경망 학습 시간이 오래 걸리거나 학습 진행 중에 수렴하는 결과를 야기

활성함수(activation function)

ReLU(Rectified linear unit)

-
- 입력값이 0보다 작으면 0, 양의값을 가지면 출력은 입력 그대로 출력

$$h(x) = \begin{cases} x & (x > 0) \\ 0 & (x \leq 0) \end{cases}$$



렐루 함수

활성함수(activation function)

ReLU(Rectified linear unit)

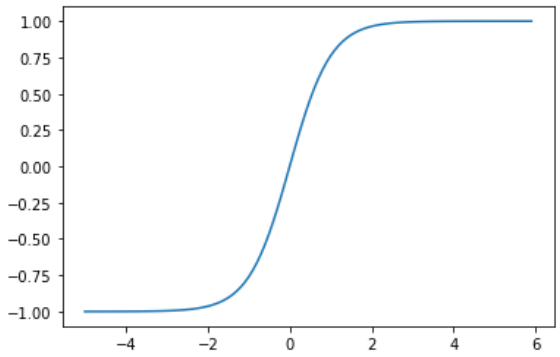
- 0과 1사이의 값이 아니므로, 0 이하의 정보는 과감히 무시함
- Sigmoid함수에서 발생하는
-
- exp()를 사용하지 않기 때문에, 연산 속도 또한 엄청 빠름
-

활성함수(activation function)

하이퍼볼릭 탄젠트 함수(tanh)

-
- -1 - 1 사이의 값을 출력
- 데이터의 평균이 0.5가 아닌 0 $\rightarrow$ 데이터의 중심이 0이 아니어서 발생하는 문제 해결
- vanishing gradient문제

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



하이퍼볼릭 탄젠트 함수

활성함수(activation function)

Maxout 함수

- Maxout 함수는 앞서 나온 ReLU의 단점을 보완하고, 장점은 그대로 가져온 함수
- 연결된 두 개의 뉴런 값 중 큰 값을 사용하는 함수, 실제로 성능도 제일 좋은 경향
- 하지만 이 함수는 계산 량이 복잡하다는 단점
- 하나의 뉴런당 파라미터 수가 두배가 되기 때문 (아래 수식에서 ω_1^T, ω_2^T 모두 파라미터)

$$f(x) = max(\omega_1^T x + b_1, \omega_2^T x + b_2) \quad (1)$$

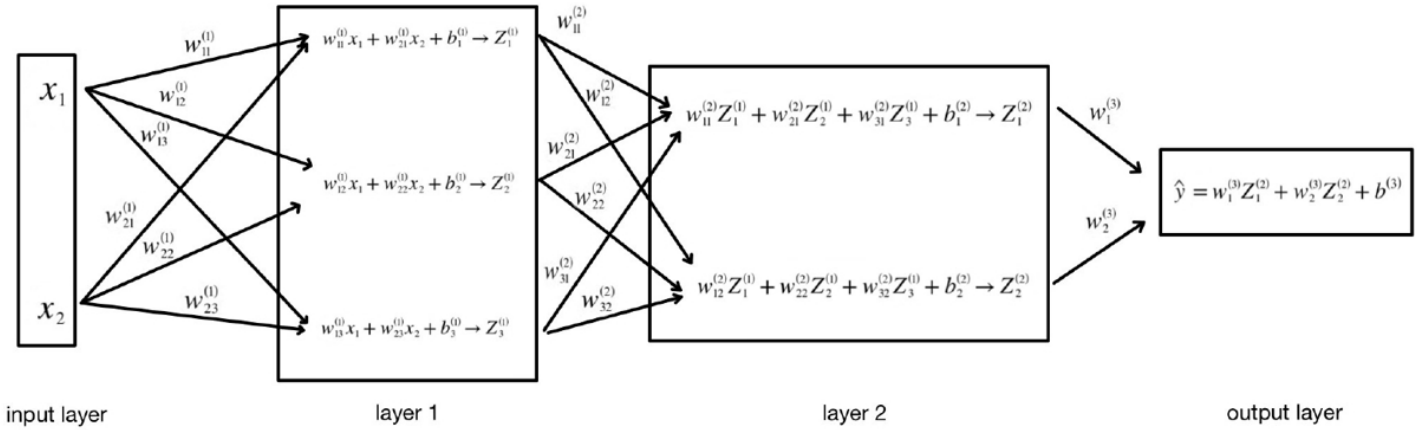
활성함수(activation function)

Softmax 함수

- 조금 특별한 활성화 함수
- 그래프는 존재하지 않고,
-
-

MLP 아키텍쳐

- 2개의 특성변수 x_1과 x_2가 입력층을 구성
- 3개의 노드를 가지는 첫번째 은닉층
- 2개의 노드를 가진 두번째 은닉층
- 1개의 노드로 구성된 출력층



$$Z_i^{(1)} = \sigma\left(w_{1i}^{(1)}x_1 + w_{2i}^{(1)}x_2 + b_i^{(1)}\right) \qquad i = 1, 2, 3$$

$$Z_i^{(2)} = \sigma\left(w_{1i}^{(2)}Z_1^{(1)} + w_{2i}^{(2)}Z_2^{(1)} + w_{3i}^{(2)}Z_3^{(1)} + b_i^{(2)}\right) \qquad i = 1, 2$$

그림 2–5 MLP 아키텍처(입력층 → 2개의 은닉층 → 출력층)

MLP 아키텍쳐

Flatten

Nodes in deep learning architecture

Nodes in deep learning architecture

MLP 아키텍쳐

- 입력부터 출력까지
- 선형결합에 사용된 b_i^k는 bias
- 추정할 모수는 ω_{ij}^k와 b_i^k
- 입력층 $\rightarrow$ 첫번째 은닉층: $2 \times 3 + 3 = 9$
- 첫번째 은닉층 $\rightarrow$ 두번째 은닉층: $3 \times 2 + 2 = 8$
- 두번째 은닉층 $\rightarrow$ 출력층: $2 \times 1 + 1 = 3$, 총 20개

MLP 아키텍쳐

- 배치 또는 모든 표본을 MLP 아키텍쳐에 입력
- 출력변수 y_i에 대응되는 예측치 $\hat{y}_i$를 출력하고 손실함수 계산
- 이 손실함수를 최소로 하는 모수 ω_{ij}^k와 b_i^k를 추정 (최적화때 다시 back)
- 모수 ω_{ij}^k와 b_i^k는 표본에 의존하지 않는다. (표본간 독립성 가정 때문)
  - 모수가 표본에 의존하지 않는다는 것은 결국 다르게 말하면 각 표본이 특성변수의 선형결합에 동일한 기여를 한다는 것
  - 결론적으로 표본이 같은 분포에서 나온 서로간에 독립(iid)임을 의미하며, 이는 역시 모형의 일반화에 있어 매우 중요한 요소

[1] https://blog.naver.com/jejero97/222303819808

MLP 아키텍쳐

- MLP에서 특성변수가 1D텐서가 아니더라도, MLP적 연결이 가능하다
  - ex) 컬러이미지인 3D텐서자료를 1D텐서로 전환이 가능함
  - 28×28픽셀 RGB채널 데이터는 (28,28,3)인 3D텐서 채널별로 28×28 행렬로 해석하고, 두번째 행부터 마지막행까지 차례로 첫번째 행의 옆에 붙이면 $28 \times 28 = 784$개의 특성변수가 만들어지고,
  - 각 채널을 다시 하나의 행으로 옆에 붙이면 $784 \times 3 = 2352$개의 1D 특성변수로 변환 가능하여 MLP에 적용 가능함

MLP 아키텍쳐

- 입력 특성변수가 2352개이므로 은닉층의 노드수가 62개로 하면,
- (딥러닝에서 은닉층의 노드수는 2배수 (power of 2)로 하는데, 이는 GPU의 구조에 따라 2배수가 연산에 효율적이기 때문)
- 모수의 수가 $2352 \times 64 + 64 = 150,592$가 된다.
  - 첫번째 문제: 은닉층의 숫자가 늘어나면 모수의 숫자가 기하급수적으로 늘어나, 모수의 추정이 불가능하거나, 최소한 모수추정 수렴속도가 매우 느려진다.
  - 그렇다면 노드의 수를 줄여 해결할수 있을까? 노드의 수를 8로 줄이면 2352개의 특성변수가 8개의 노드로 압축되게 되며 이때 병목현상 (bottle-neck)이 발생한다. 딥러닝에서 노드의 병목현상은 모수 추정의 가장 큰 방해요소!
  - 두번째 문제: 모수의 수가 지나치가 많아져 딥러닝에서 가장 기피하는 과대적합문제 발생

MLP 아키텍쳐

- MLP 아키텍쳐의 가장 큰 약점: 모든 가능한 노드의 연결로 지나치게 많은 모수
  - 해결방법1: 입력이 표본 하나당 2D텐서일때, MLP는 각 행을 독립적인 표본으로 받아 처리
  - 모수의 수는 증가하지 않지만, 2D텐서가 주는 위치정보의 손실이 있음
  - 해결방법2: 2D텐서, 3D텐서 자료의 특성을 그대로 반영하면서 모수를 획기적으로 줄이며, 정보 손실을 최소화하고, 예측성능을 높일수 있는 CNN, RNN을 사용!