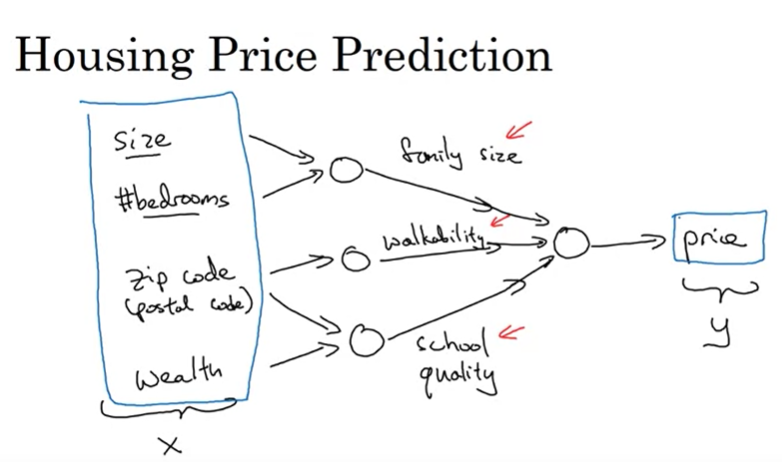
**Neural Networks and Deep Learning**



입력 값의 특성 : 집의 크기, 침실 수, 우편번호, 이웃집의 재산

신경망의 역할은 y값을 예측하는 것, 신경망 중간에 있는 레이어는 밀도가 높다.

[Supervised Learning]

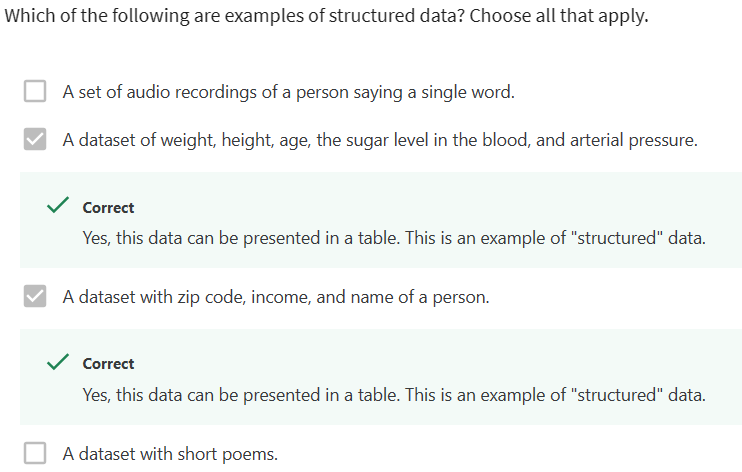
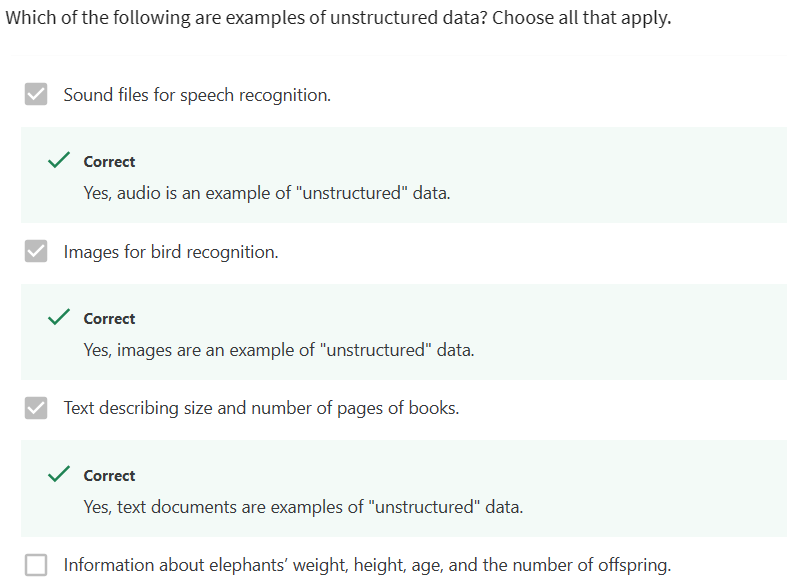
부동산 애플리케이션, 온라인 광고 ; 보편적인 표준 신경망 아키텍처 (Standard NN)

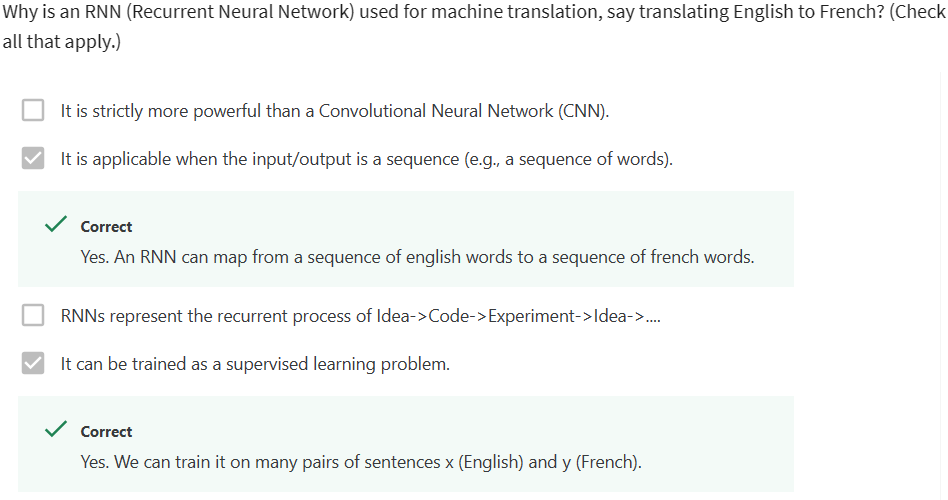
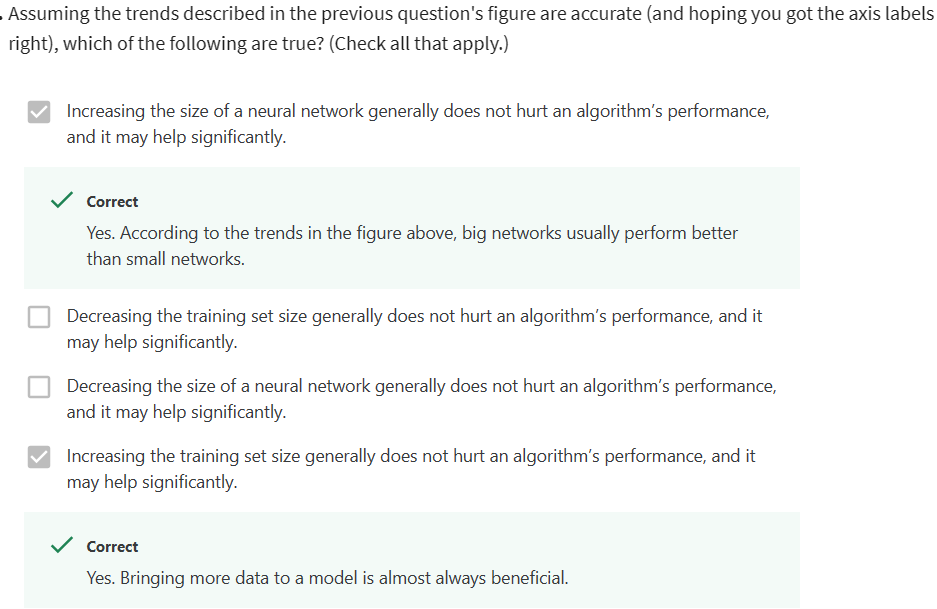
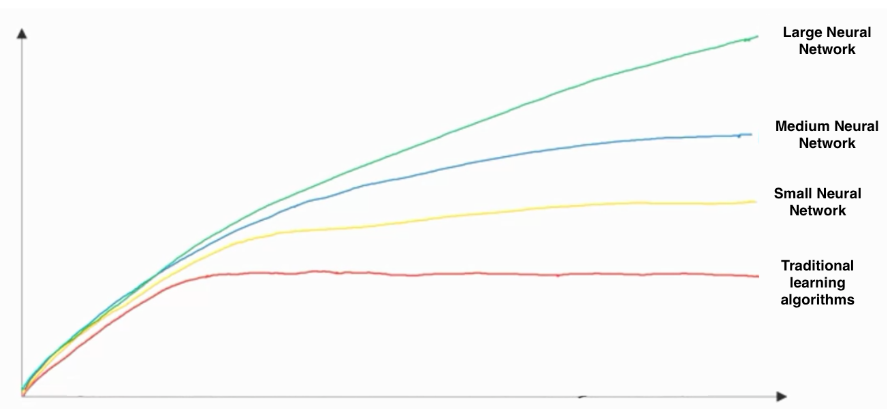
이미지 애플리케이션 ; CNN

언어, 오디오 ; 시간에 따라 재생되기 때문에 1차원 시계열 또는 1차원 시간 시퀀스 => RNN

자율주행 ; 하이브리드 신경망 아키텍처

[퀴즈]





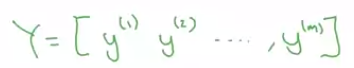
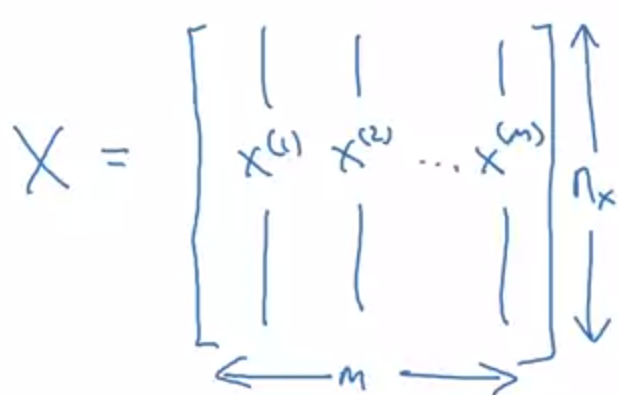
**# 2. Neural Network Basics**

[로지스틱 회귀] ; 이진 분류

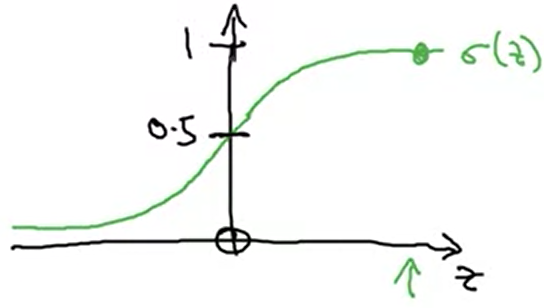
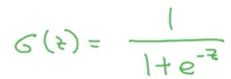
입력 이미지가 64 X 64 픽셀의 경우 빨/초/파란색 픽셀의 채도값에 해당하는 3개의 64 X 64 행렬 -> 픽셀들의 채도값을 특징 벡터로 바꾸기 위해 픽셀값 모두를 하나의 입력 특징 벡터 x에 펼쳐 -> 이 벡터 x의 전체 차원은 64 \* 64 \* 3 = 12288 (nx = 12288)

훈련 데이터 행렬 X는 nx X m 차원을 가짐 (m : 훈련데이터셋 수)

타겟 데이터 행렬 Y는 1 X m 차원을 가짐



시그모이드

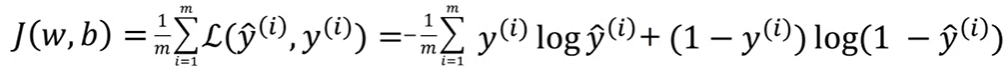
손실함수 : 실제 라벨이 y일 때 출력 ŷ이 얼마나 좋은지 측정하기 위해 정의해야 하는 함수



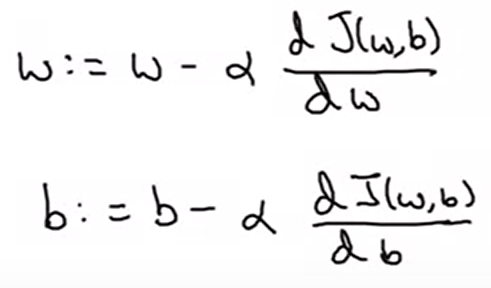
IF y = 1이면, 손실함수 = -log ŷ => 손실함수 값이 작으려면 log ŷ을 크게 하기를 원하고, 즉 ŷ도 크게 하기를 원한다. 하지만, ŷ는 시그모이드 함수 출력이기에 1이 될 수 없다. 가능한 한 크게 ŷ을 원하지만 1보다 클 수는 없다.

IF y = 0이면, 손실함수 = -log(1- ŷ) =>  ŷ을 작게 하기를 원한다.

비용함수

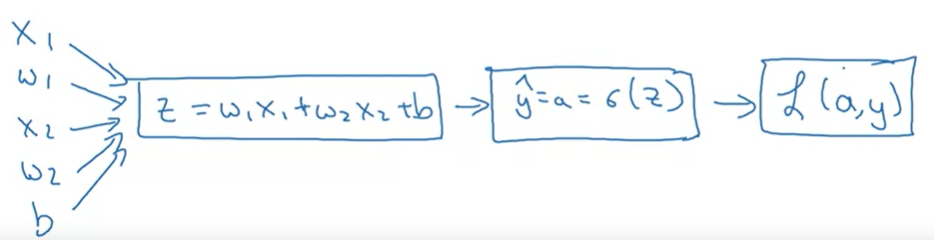


비용 함수에 대한 기울기 하강 ; 그레디언트 (경사하강법)

 알파 : 학습률

[로지스틱 회귀에 대한 기울기 하강]

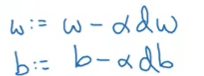
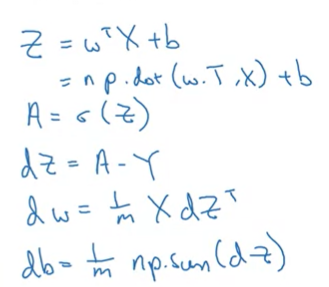
손실을 줄이기 위해 매개변수 w와 b를 수정



* 딥러닝 알고리즘을 구현할 때 코드에 for 루프가 명시되어 있으면 알고리즘 실행 효율성(속도)이 떨어진다. 점점 더 큰 데이터셋으로 이동하기에 명시적 for 루프를 사용하지 않고 알고리즘을 구현할 수 있다는 것은 매우 중요하며 훨씬 더 큰 데이터 셋으로 확장할 수 있다. => 벡터화 기법

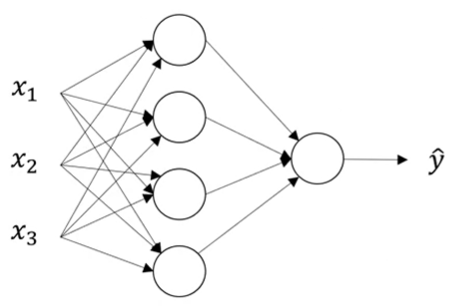
[코드] Vectorization

for문 대신 np.dot() 사용 / 로지스틱 회귀를 위한 기울기 경사 하강



[코드] 브로드캐스팅

[코드] numpy vector

[Neural Network Representation]  


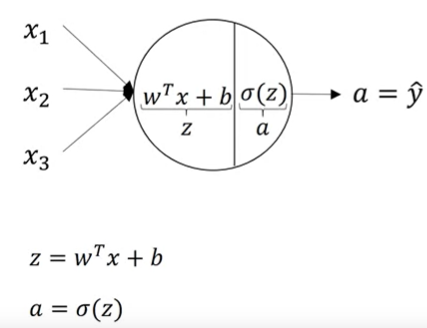
Input layer (a[0] = X)

Hidden layer (a[1])

Output layer (a[2] = y\_hat)

* 2 layer NN (input layer은 레이어라고 부르지 않음)

ex) w[1], b[1] -> layer1(Hidden layer)과 관련된 매개 변수  
w[1].shape : (4, 3) – 숨겨진 유닛과 layer의 4개 노드가 있으며, 3개의 입력 기능이 있다  
b[1].shape : (4, 1) 벡터  
w[2]. b[2] -> layer2(Output layer)과 관련된 매개 변수  
w[2].shape : (1, 4), b[2].shape : (1, 1) 벡터

   
a[l]\_i = sigmoid(z[l]\_i) l : layer 번호, i : 해당 레이어의 노드

z[1] = W[1] \* x + b[1] = W[1] \* a[0] + b[1]

a[1] = sigmoid(z[1]) => 활성화 함수

z[2] = W[2] \* a[1] + b[2]

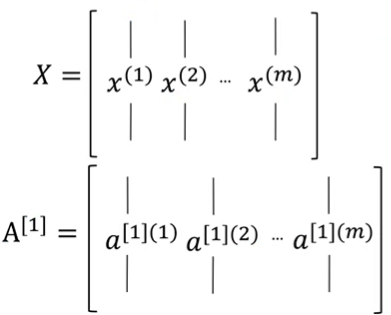
a[2] = sigmoid(z[2]) => 활성화 함수

---------------------------단일 훈련------------- X -> a[2] = y\_hat / 수직으로 쌓아올림

[m 훈련] ; 수평으로 쌓아올림 => 훈련 전체를 인덱싱

첫 번째 훈련 X(1) -> a[2](1) = y\_hat(1) | z[1](1) = W[1] \* X(1) + b[1]  
두 번째 훈련 X(2) -> a[2](2) = y\_hat(2) | z[1](2) = W[1] \* X(2) + b[1]

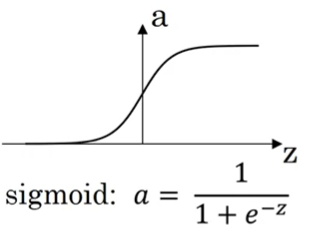
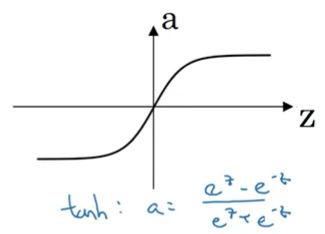
m 번째 훈련 X(m) -> a[2](m) = y\_hat(m) | z[1](m) = W[1] \* X(m) + b[1]  
 X(1), X(2) … X(m)은 수평으로 쌓이는 X 행렬



행렬을 수직으로 움직이면 숨겨진 단위 번호에 대한 인덱싱 (hidden units)  
행렬을 수평으로 움직이면 여러 훈련 예제 (training examples)

[활성화 함수 g()] ; 레이어마다 다를 수 있다

1. 시그모이드 2) 탄젠트 함수

성능 : 1 < 2 ; +1과 -1 사이의 값으로 숨겨진 층에서 나오는 활성화의 평균은 0에 더 가까움

데이터를 중심에 위치시키고, 탄젠트 함수를 사용해 데이터의 평균이 0이 되도록 할 수 있다. 이렇게 하면 다음 레이어에 대한 학습이 조금 더 쉬워진다.

시그모이드는 거의 사용하지 않지만, 출력 레이어에 대한 예외 존재  
; y가 0이거나 1이면 y\_hat은 -1 ~ +1 이 아니라 0~1 사이에 있는 출력하려는 숫자가 되는 것이 합리적이기에, 이진 분류를 사용할 땐 상위 레이어에 시그모이드 활성화 함수를 사용한다.  
출력이 0 1 값이고, 이진 분류를 사용하는 경우, 시그모이드는 출력층에 대해 자연스러운 선택

1과 2의 단점 : z가 매우 크거나 작으면 이 함수의 기울기의 도함수가 매우 작아지는데, 이는 기울기 하강 속도가 느려질 수 있다.

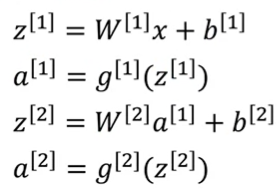
1. 정류된 선형 단위(Relu) ; 기본 선택 a = max(0, z)

단점 : z가 음수일 때 도함수 = 0

1. Leaky Relu : Relu의 단점 보완 a = max(0.01z, z)

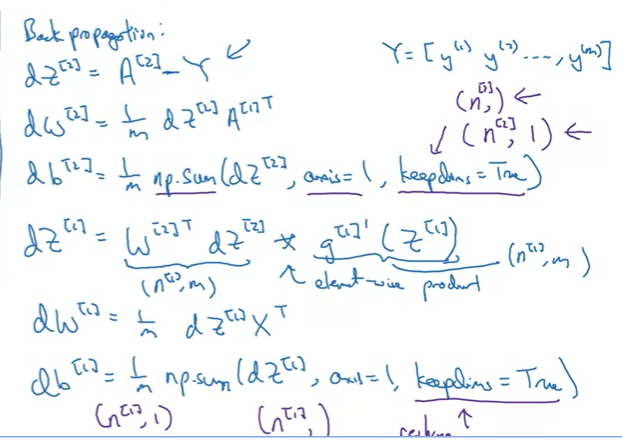
3과 4의 장점 : 활성화 함수의 도함수인 z의 많은 공간에 대해 활성화 함수의 기울기가 0과 매우 다르고, 빠른 학습을 한다.  
- 함수의 기울기가 0이 되어 학습 속도가 느려지는 효과가 적고, z 범위의 절반에 대해 값의 기울기가 0이라는 것을 알고 있다.

[비선형 활성화 함수]



IF 활성화 함수가 없다면, a[1] = z[1] => g(z) = z (선형 활성화 함수)  
a[1] = z[1] = W[1] \* X + b[1]  
a[2] = z[2] = W[2] \* a[1] + b[2] = W[2] \* (W[1] \* X + b[1]) + b[2]  
 = W[2] \* W[1] \* X + W[2] \* b[1] + b[2] = W’ \* X + b’  
신경망은 입력의 선형 함수 출력 => 선형의 숨겨진 층은 쓸모가 없다 => 표현력이 없다.  
이는 회귀 문제에 대한 머신러닝을 할 때 출력층에서 사용

[Gradient Descent or Neural Networks] 역전파



[Random Initialization]

매개변수에 대한 가중치 0으로 초기화한 후 기울기 하강하면 작동하지 않을 것이다.

가중치는 매우 작은 무작위 값으로 초기화하는 것을 선호 ex) 0.01

가중치가 너무 크면 활성화 값을 계산할 때, 기울기 하강이 매우 느릴 것이다.

w 값이 매우 크면, 훈련 초기에도 z의 값이 클 확률이 높다.   
-> 탄젠트나 시그모이드가 포화되어 학습 속도가 느려진다.

[다중 layer] 단일 -> m번 훈련 (1 -> m)

w[l].shape : (n[l], n[l-1]) / b[l].shape : (n[l], 1) / a[l] = z[l] .shape(n[l], 1)

함수들은 작지만 심층 신경망을 계산되며 작다는 것은 숨겨진 유닛의 수가 적다는 것을 의미  
그러나 얕은 네트워크로 동일한 함수를 계산하려고 하면, 숨겨진 레이어가 충분하지 않다면 계산하기 위해 숨겨진 유닛이 기하급수적으로 더 많이 필요할 수 있다.