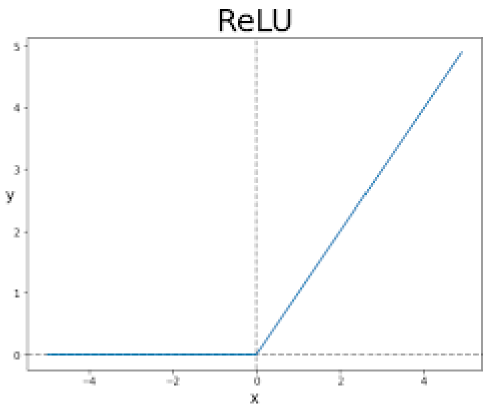
1. 딥러닝 소개

1. 신경망

- **딥러닝**: 신경망을 학습시키는 것
- **신경망**
    - < house 크기에 따른 가격 함수를 이용하여 설명>
      X: 주택의 크기(=신경망의 입력)
      원인노드(신경망의 하나의 뉴런) : 주택의 크기를 입력으로 받아서, 선형함수 계산 + 결과값과 0 중 큰 값을 주택 가격으로 예측
      Y: 주택의 가격(=신경망의 출력)

- 0으로 유지되다가 직선으로 올라가는 형식의 함수 : **ReLU 함수(Rectified Linear Unit)**



2. 지도학습

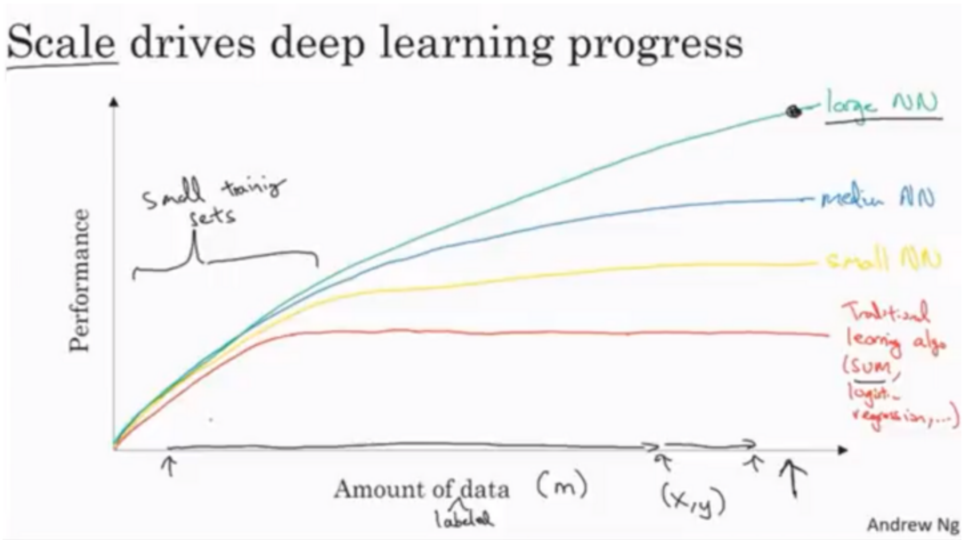
- **지도학습** : 머신러닝의 한 종류, 정답이 주어져 있는 데이터를 사용하여 컴퓨터를 학습시키는 방법
    - 입력 X와 출력 Y에 매핑되는 함수 학습하고자 함.
    - **사용 예시**
        - Input: Ad, user info / Output: Click on ad -> 클릭할 만한 광고를 보여줌 : **Standard NN(표준 신경망 사용)**
        - 음성 파일을 입력으로 넣고, 텍스트 대본으로 출력 가능 / 기계 번역 -> **RNN(순환 신경망)**

- **자율 주행**(입력: 이미지, 레이더에서 얻은 정보, 출력: 도로 위 차들의 위치 정보) -> **CNN(합성곱 신경망), Hybrid 신경망 사용**
- 보통 이미지 데이터는 **합성곱 신경망**에서 사용됨.
- **순환 신경망** : 1차원 시퀀스 데이터에 강함

- **구조적 데이터** : 기본적으로 데이터베이스로 표현된 데이터
- **비구조적 데이터**: 음성 파일 or 인식하고자 하는 이미지, 텍스트 데이터(구조적 데이터보다 작업하기 어렵)
- 딥러닝 덕분에 컴퓨터가 비구조적 데이터를 인식가능

3. 딥러닝은 왜 잘되고 있는가

- 더 큰 신경망을 사용할수록 훈련할 데이터 양이 많아질 때의 성능 좋아짐
- 훈련할 데이터 양이 적을 때: 알고리즘의 상대적 순위가 잘 정의되지 X(의미 없음)



- 초창기 딥러닝의 문제: **데이터와 계산의 규모**
- 신경망의 활성화 함수
  : **시그모이드 함수 -> ReLU 함수로의 변화** : 경사 하강법이라는 알고리즘 탄생
  - 머신러닝에서 시그모이드 함수 사용 시의 문제점 : 함수의 경사가 거의 0인 곳에서의 학습이 매우 느려짐
  - 이전과 달리 빠른 실험 결과를 얻을 수 있어서, **아이디어(Idea) 생산 > 코드(Code) 구현 > 실험(Experiment)결과**의 시간이 단축.

2. 신경망과 로지스틱 회귀

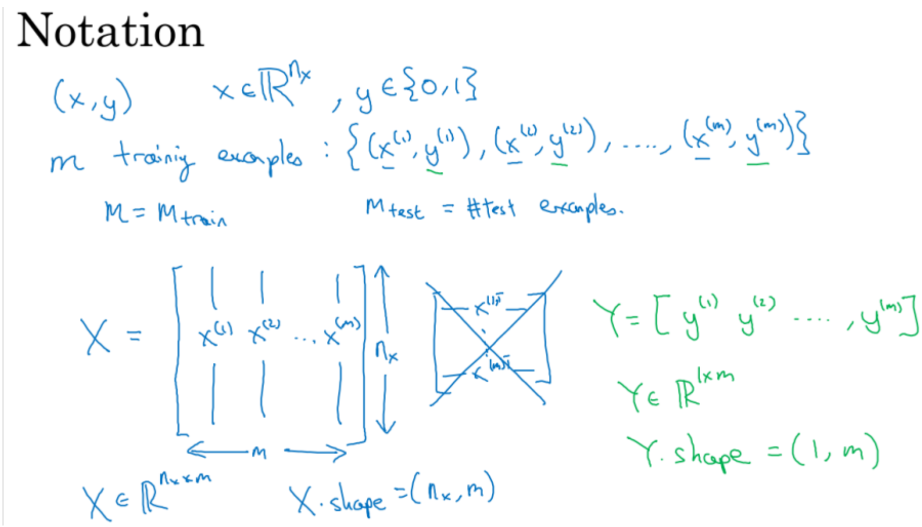
1. 이진 분류(Binary Classification)

- **로지스틱 회귀** : 이진 분류를 위한 알고리즘
- **이진 분류**
    - 그렇다 / 아니다 2개로 분류하는 것. 이때 결과가 ‘그렇다’ 이면 1로 표현하고 ‘아니다’이면 0으로 표현
      ex) 고양이다(1) / 고양이가 아니다.(0)
    - 입력하는 이미지 크기: 64 * 64 픽셀 + *Red/Greem/Blue(채도)*

    → 3개의 64x64 행렬

    - 픽셀 하나하나에 적혀있는 채도 값들을 합쳐서 하나의 **차원 특징 벡터 (feature Vetor) X**
    - 차원 특징 벡터 X: **열 벡터(행렬)**
    - X의 전체 차원(성분) 수 Nx: **64*64*3 = 12888** 됩니다.
    - 이진 분류의 목표 : x에 대한 레이블 y가 0인지, 1인지를 예측할 수 있는 분류기를 학습시키는 것.
- **Notation**
    - 하나의 훈련 샘플을 (X, Y)로 표현:
        - x는 차원 특징 벡터이고, 레이블 y는 0과 1 중 하나의 값
    - 훈련 세트(훈련할 사진 데이터들의 집합)를 m
        - 훈련 세트: { (X1 , Y1) , (X2 , Y2) ~~~~~ (Xm-1 , Ym-1) , (Xm , Ym) }
        - m training example: m은 훈련 샘플의 개수
    - X: n_x * m 행렬, Y: 1m 행렬
        - X1 , X2 ~~~ , Xm(m은 훈련 세트의 수)은 각각 행렬의 열을 의미

      → 행렬 X m 개의 열이 존재하며, Nx 개의 행이 존재



2. 로지스틱 회귀

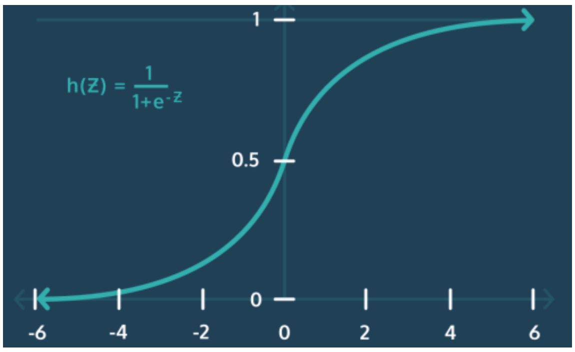
- X: 입력 특성 / y: 주어진 입력특성 X에 해당하는 실제 값 / y-hat: y의 예측값 = **P(y=1 | x)**의미
  - ex) X가 고양이 사진이라면, y의 예측값은 그 사진이 고양이 사진일 확률을 의미
- 선형회귀: 이진 분류를 위한 좋은 알고리즘 X

→ y의 예측값은 확률이므로 **0과 1사이**여야 하는데, w^T + b는 1보다 훨씬 크거나 음수일 수 있음

→ 시그모이드함수를 통해 0과 1사이의 값으로 변환

따라서 로지스틱 회귀를 위한 $\hat{y} = \sigma(W^T X + b)$ 로 구하게 됩니다.

참고) 시그모이드 함수 $sigma(z) = \frac{1}{1+e^{-z}}$



- 로지스틱 회귀의 매개 변수: W , b
  - W는 x와 같은 차원 벡터, b는 실수값

3. 로지스틱 회귀의 비용 함수

- 매개변수 w, b 학습하려면, 비용함수 정의 필요
- i번째 훈련 샘플의 y 예측값 : w^T * x^(i) + b에 시그모이드 함수 적용한 값
- Loss function(손실 함수): 알고리즘이 출력한 y의 예측값과 참값 y의 제곱오차의 반

$$L(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^2$$

→ 잘 사용하지 X(지역 최소값에 빠질 수 있기 때문)

- 로지스틱 회귀에서 사용하는 손실 함수: **L(yhat, y) = -(y*log(yhat) + (1-y)*log(1-yhat))**

$$L(\hat{y}, y) = -(y \log \hat{y} + (1-y) \log (1-\hat{y}))$$

  - **y=1**] L(y의 예측값, y) = -log(y예측값) -> log(y예측값)이 최대한 커져야 함.

  → yhat 커져야 함 → 하지만, yhat은 시그모이드 함수의 값이므로 1보다 클 수 없음 = 1에 수렴

  - **y=0**] L(y의 예측값, y) = -log(1-yhat) -> log(1-yhat) 최대한 커져야 함

  → yhat 작아져야 함 → yhat은 0과 1 사이 = 0에 수렴하도록

- 손실 함수는 훈련 샘플 하나에 관하여 정의됨.
- 비용 함수: 훈련 세트 **전체에** 대해 얼마나 잘 추측되었는지 측정해주는 함수

$$J(w, b) = -\frac{1}{m}\Sigma_{i=1}^{i=m}(y^{(i)} \log \hat{y}^{(i)} + (1-y^{(i)}) \log (1-\hat{y}^{(i)}))$$

- 로지스틱 회귀 모델을 학습하는 것 = **손실함수 J를 최소화해주는 매개 변수 w, b 찾는 것**

4. 경사 하강법

: 비용 함수 J(w,b)를 가장 작게 만드는 매개변수 w, b를 훈련 세트에 학습시키는 방법

- 비용 함수는 볼록한 형태여야 함.

→ 볼록하지 않은 함수를 쓰게 되면, 경사하강법을 통해 최적의 파라미터를 찾을 수 없음.

- 보통 초기값은 **0으로** 시작
- 경사하강법은 **가장 가파른(steepest) 방향**(가장 빨리 내려올 수 있는 방향) 선택
  = 즉 함수의 기울기를 따라서 최적의 값으로 갱신해나감.

- $w : w - \alpha \frac{dJ(w,b)}{dw}$
- $b : b - \alpha \frac{dJ(w,b)}{db}$
- α : 학습률이라고 하며, 얼만큼의 스텝으로 나아갈 것인지 정합니다.
- $\frac{dJ(w)}{dw}$: 도함수라고 하며, 미분을 통해 구한 값 입니다. dw 라고 표기하기도 합니다.

- **dw >0** : 파라미터 w 는 기존의 w 값 보다 **작은 방향**으로 업데이트 될 것이고
- **dw <0** : 파라미터 w 는 기본의 w 값 보다 **큰 방향**으로 업데이트 될 것

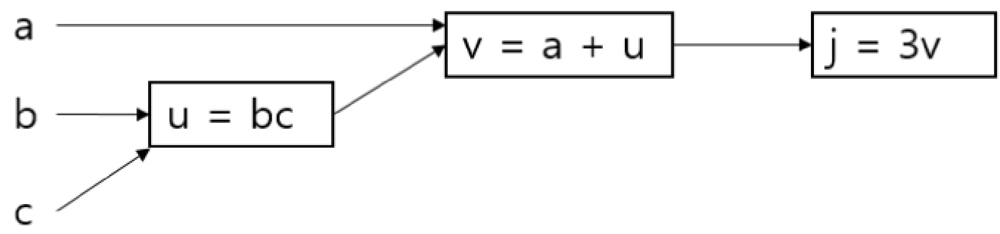
5. 미분

1. 함수의 도함수 = 함수의 기울기
2. 함수의 기울기는 위치에 따라 다른 값을 가질 수 있다.

6. 계산 그래프

- 정방향 패스, 정방향 전파 : 신경망의 출력값 계산
- 역방향 패스, 역방향 전파 : 경사, 도함수 계산
- EX) J(a,b,c) = 3(a + bc)

1. b*c 계산 → u = b * c
2. a + bc 계산 → v = a + u
3. J = 3*v



- **미분의 연쇄법칙**
  : a -> v -> J(a는 v에 영향, v는 J에 영향)일 때, **dJ/da = dJ/dv * dv/da**
- 최종변수를 Final output var, 미분하려고 하는 변수를 var 라고 정의

→ dFinal output var / dvar = dvar

7. 로지스틱 회귀의 경사하강법

- dL/da 를 변수 da로 표기 = -(y/a)+((1-y)/(1-a))
  - L(a,y) = (y log(a) + (1-y) log (1 - a))
  - L(a,y)를 a에 대해 미분 → -(y/a+(1-y)/(1-a))da
  - dL/da = -(y/a+(1-y)/(1-a))
- dz = dL/dz=a-y=dL/da*<u>da/dz</u>
- dL/dw1 = dw1 = x1 dz
- dw2 = x2 dz
- db=dz

8. m개 샘플의 경사하강법

- **Cost function J (비용 함수)의 값**: m 개의 손실 함수의 총합을 구하여, m으로 나누어 평균을 구하기
- 역방향 계산을 통해 도함수(Derivatives) 값의 평균 구하

Logistic regression on m examples

$J=0;\ dw_1=0;\ dw_2=0;\ db=0$

$\rightarrow$ For $i=1$ to m

$$z^{(i)}=w^Tx^{(i)}+b$$

$$a^{(i)}=\sigma(z^{(i)})$$

$$J \mathrel{+}= -\left[y^{(i)}\log a^{(i)}+(1-y^{(i)})\log(1-a^{(i)})\right]$$

$$dz^{(i)}=a^{(i)}-y^{(i)}$$

$$\left\{\begin{array}{l} dw_1 \mathrel{+}= x_1^{(i)}dz^{(i)} \\ dw_2 \mathrel{+}= x_2^{(i)}dz^{(i)} \\ db \mathrel{+}= dz^{(i)} \end{array}\right. \quad n=2$$

dw_3 $\vdots$ dw_n

$J \mathrel{/}= m \leftarrow$

$dw_1 \mathrel{/}= m;\ dw_2 \mathrel{/}= m;\ db \mathrel{/}= m. \leftarrow$

$$dw_1=\frac{\partial J}{\partial w_1}$$

$$w_1 := w_1-\alpha\, dw_1$$

$$w_2 := w_2-\alpha\, dw_2$$

$$b := b-\alpha\, db$$

Vectorization

Andrew Ng