Big Data Analytics Programming

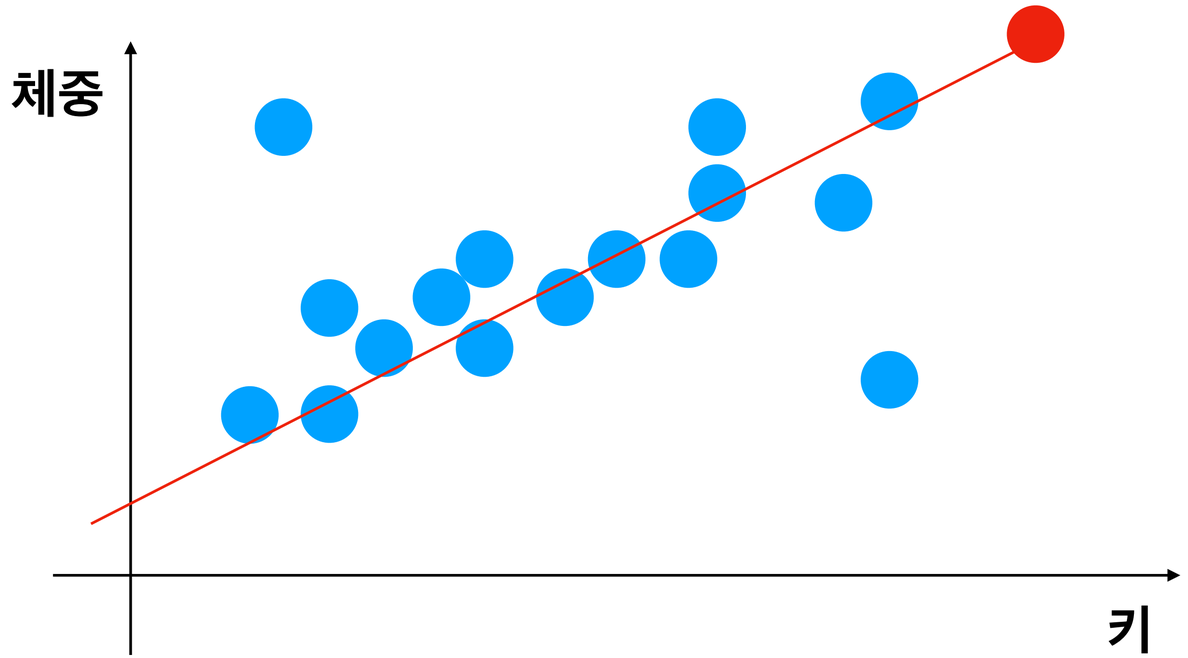
Week-07. Deep Learning

Jungwon Seo, 2020-Fall

Machine Learning

Linear Regression

- 선형회귀
  - 종속변수 y와 한 개 이상의 독립 변수 X와의 선형 상관관계를 모델링하는 회귀분석 기법

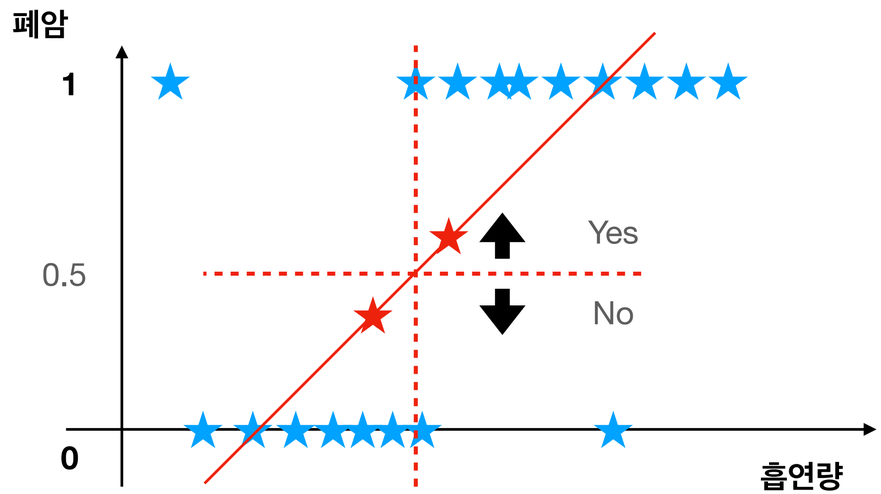


y(체중) = aX(키) + b

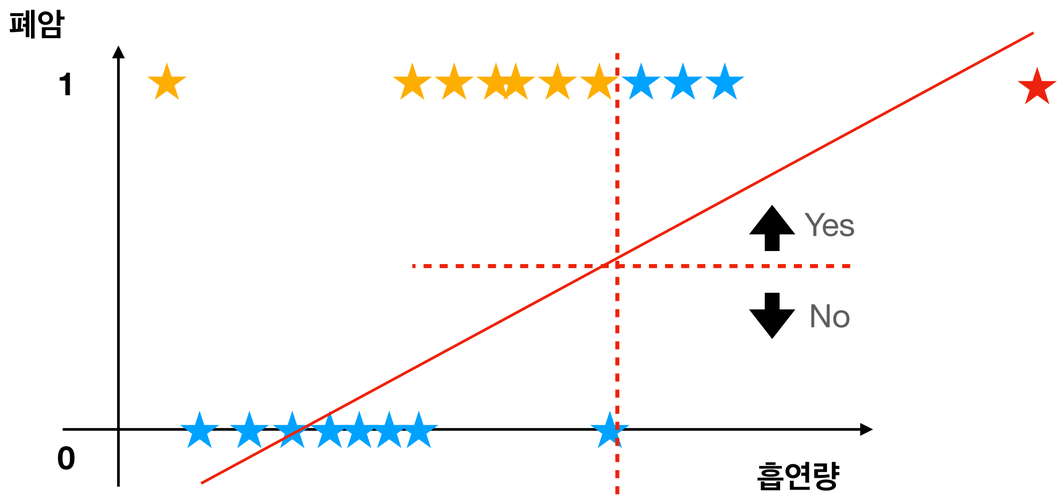
Machine Learning

Linear Regression for classification?

- 선형회귀로 분류를 하려고 했을 때의 문제점



$$y = ax + b$$



$$y = ax + b$$

Machine Learning

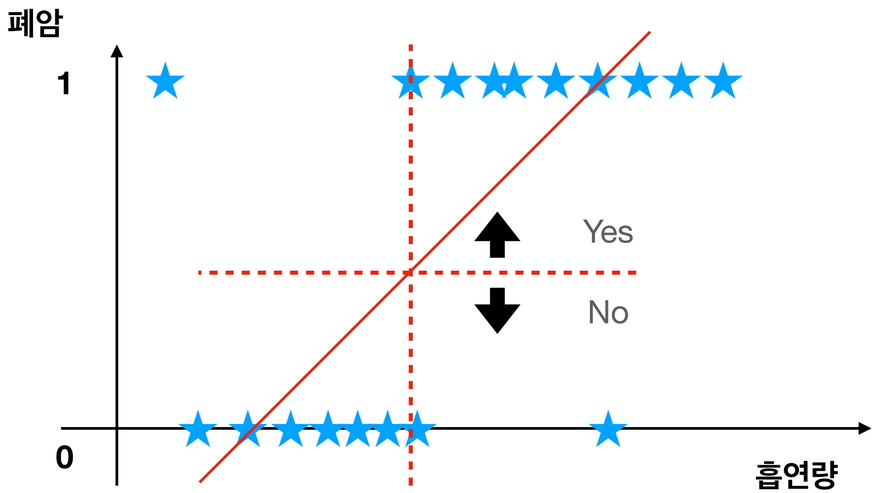
Linear Regression for classification?

- 선형회귀로 분류를 하려고 했을 때의 문제점
- 분류문제는 y과 0과 1사이에서 나와야 하는데, 선형회귀에서는 y값의 범위가 없다
  - $y = ax + b$
    - a= 0.5, b=0, x=10 => y=5
    - a= 0.5, b=0, x=100 => y=50
  - 그렇다면, x가 어떤 값을 갖든, 최종 결과값을 0과 1사이로 변환 시킬 수 있는 함수가 있지 않을까?
  - Sigmoid : $y = \frac{1}{1 + e^{-x}}$

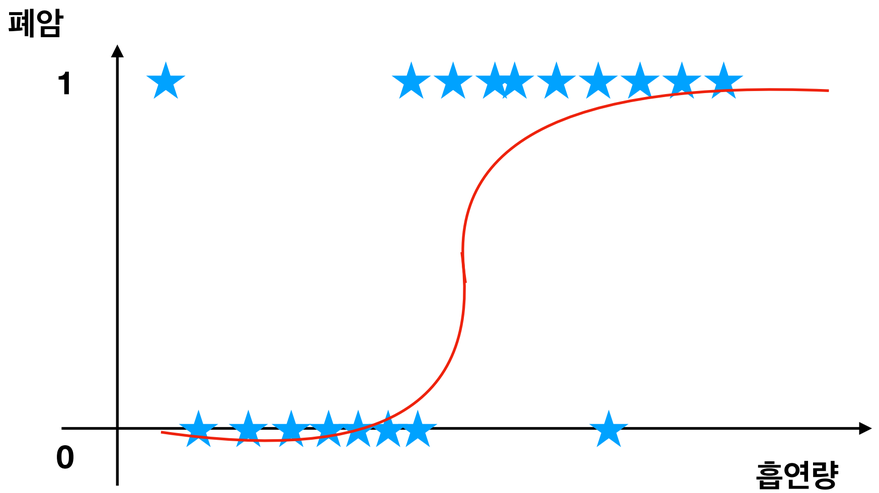
Machine Learning

Logistic Regression (Classification)

- 선을 다음과 같이 그리면 어떨까?



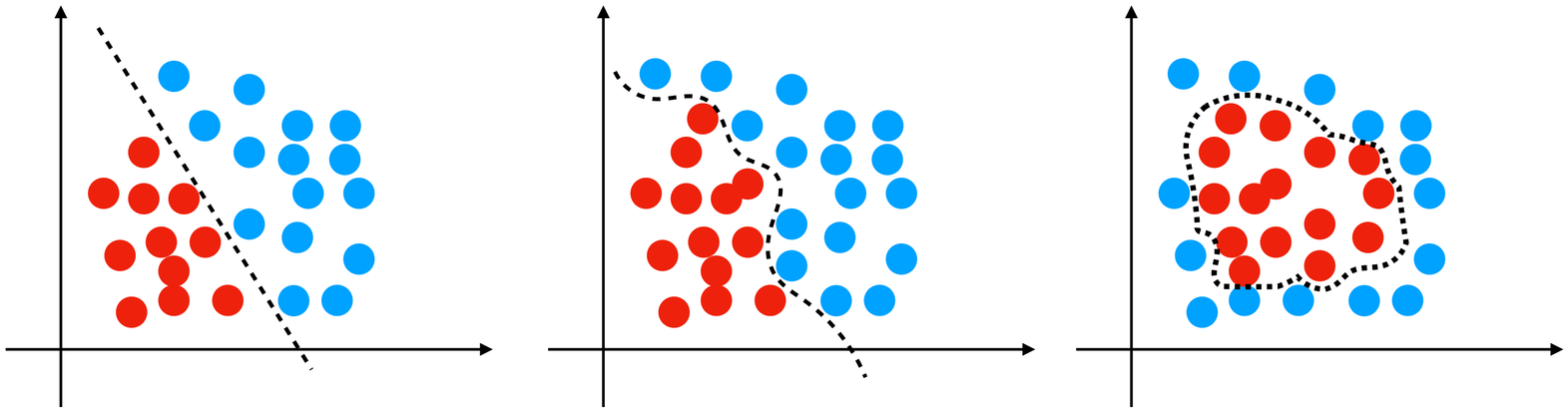
$$y = ax + b$$



$$y = \frac{1}{1 + e^{-\theta^T x}}$$

Machine Learning
Logistic Regression의 한계

- Non-linearity (비선형성)을 만들기 위해서는 많은 수의 변수 조합이 필요함
  - x1, x2 => x1, x2, x1^2, x2^2, x1x2, …
  - x1, x2, x3 => x1, x2, x3, x1x2, x1x3, x2x3, x1x2x3,… 폐암
  - 1024x1024픽셀 이미지 데이터의 경우?



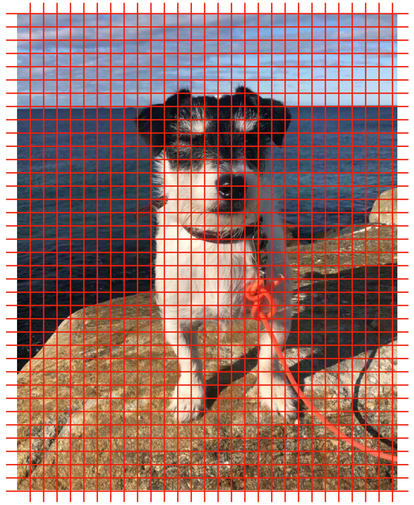
$y = a_1x_1 + a_2x_2 + b$

$y = a_1x_1 + a_2x_2 + a_3x_1x_2 + b$

$y = a_1x_1 + a_2x_2 + a_3x_1 \cdots$

Machine Learning
Dealing with Non-linearity

Dog



283x354

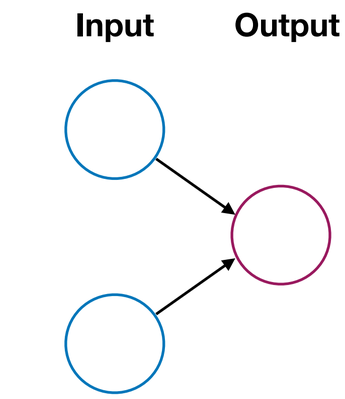
100,536 features!!!
RGB의 경우 x3

- 비선형성에 대응하기 위한 3가지 전략
  - Explicitly fixed mapping
    - 이전 예제
  - Implicitly fixed mapping
    - Kernel Method
      - SVM, Kernel Logistic regression
      - 데이터 분포를 추정
  - **Parameterized mapping**
    - **Multilayer feed-forward Neural Networks**

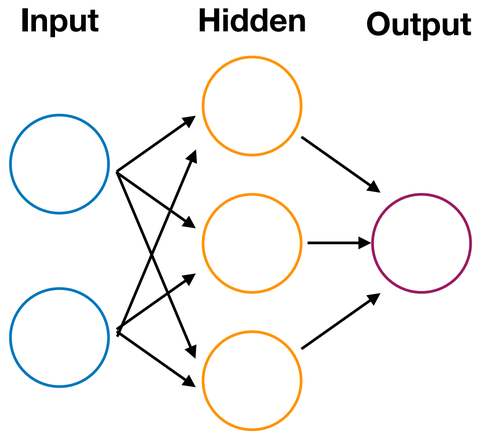
Artificial Neural Networks
인공신경망

- Non-linearity (비선형성)를 제공하기 위해, 변수의 조합이 아닌, 노드의 조합을 이용
  - 각각의 단일 노드 (hidden) 는 하나의 logit과 동일
  - 매 학습 당 독립변수에 곱해지는 파라미터 (weight)를 조정

- 모델이 학습되는 과정에서 값들이 레이어 간의 전파를 통해 이루어 진다고 해서 Feed Forward Neural Networks (FNNs)라고 도 불림



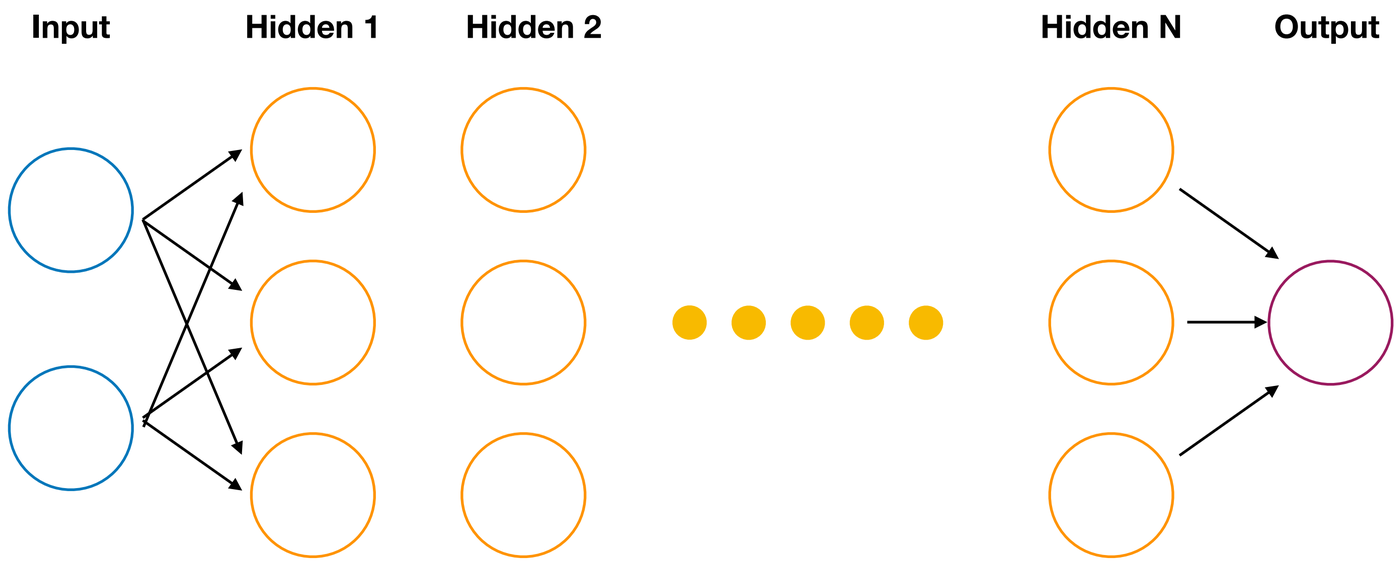
Logistic Regression



Artificial Neural Networks

Deep Neural Networks
깊은 신경망

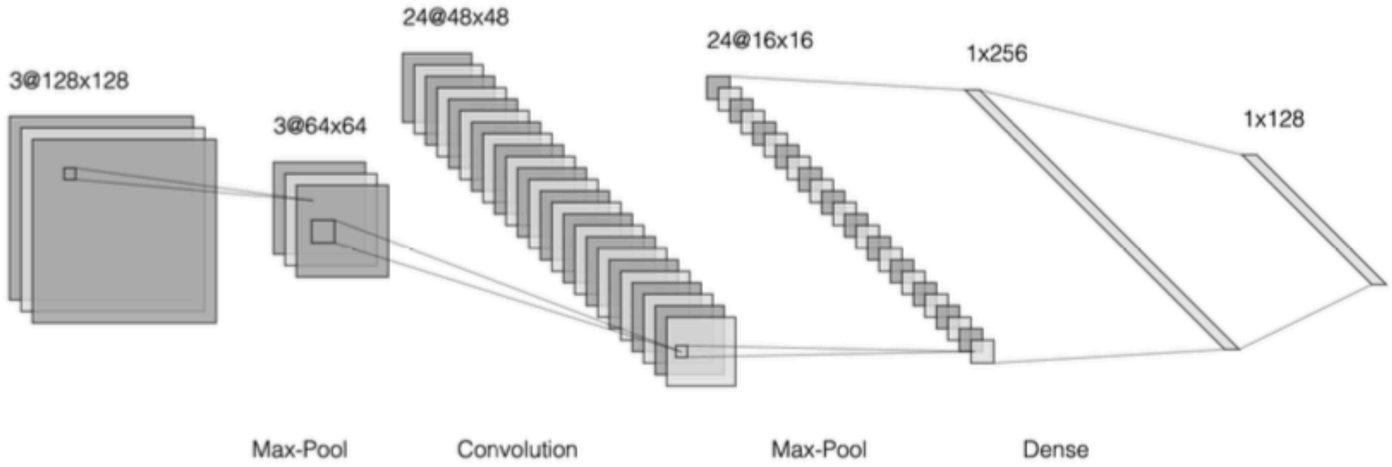
- 기존 인공 신경망에 더 많은 Hidden Layer의 수 를 추가해서 깊게 (Deep) 만든 신경망 모델
  - 처음 제시된 시점에 비해 (1960년대) 유명세를 얻기까지 시간이 걸림



[Deep Neural Networks]

Deep Learning
Convolutional Neural Networks (CNNs)

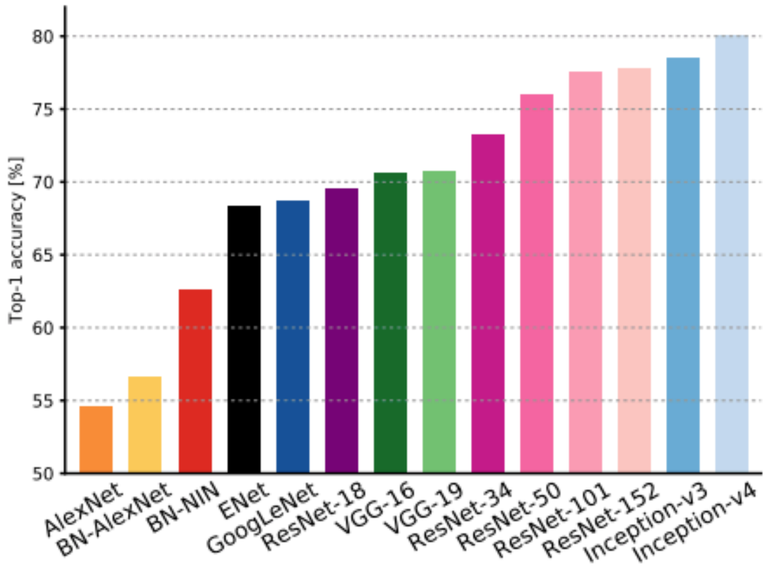
- 이미지 데이터에서 큰 효과를 보여준 모델
- 기존 FNNs의 한계를 보완함
  - 벡터화에 의한 이미지 형태 정보 손실 -> 이미지 원본 형태를 (행렬) 유지한채 학습
  - 벡터화에 의한 기하급수적 모델 파라미터 증가 -> Pooling 레이어로 축소된 이미지 처리

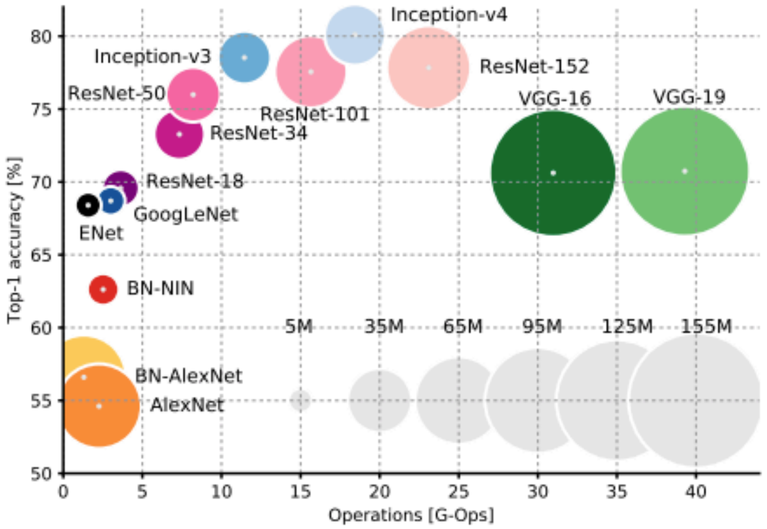


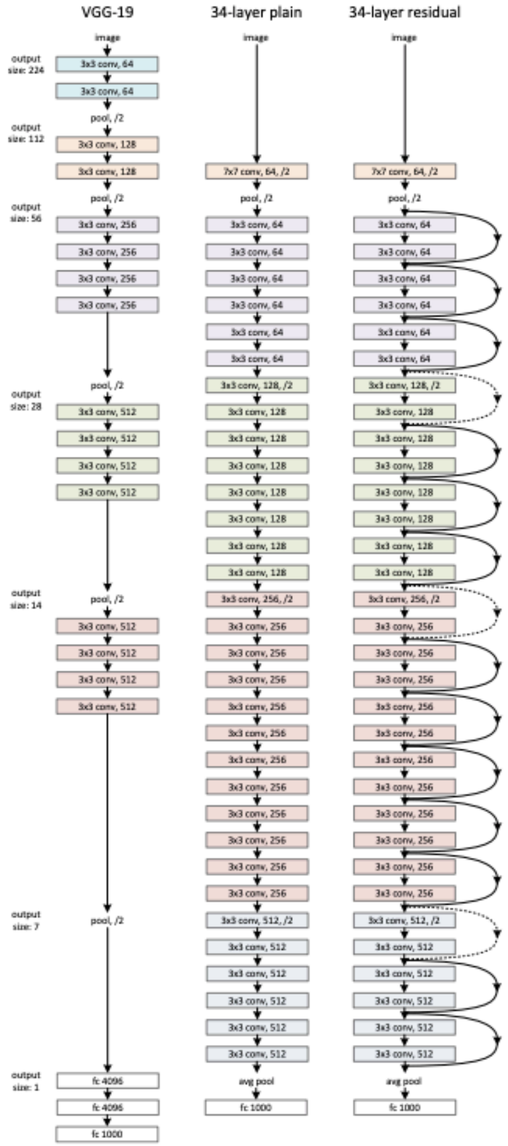
Deep Learning

Convolutional Neural Networks (CNNs)

- 여러가지 네트워크 모델들이 제안됨







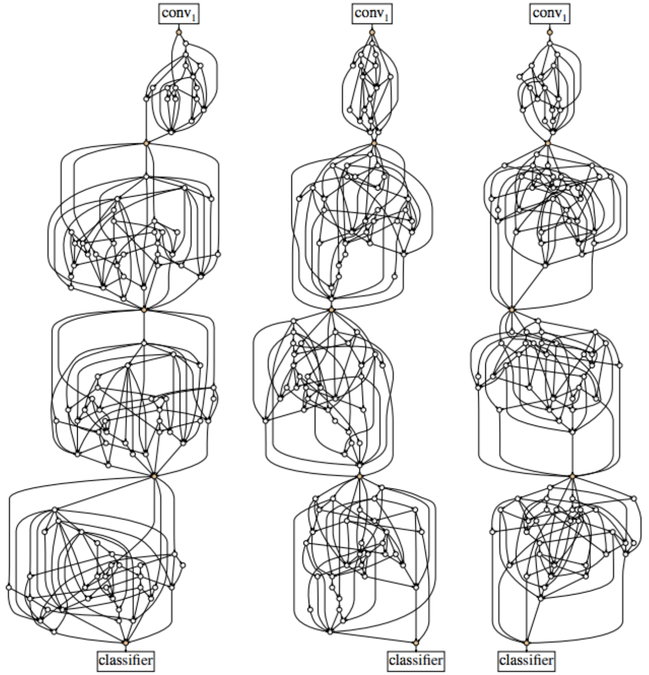
Canziani, Alfredo, Adam Paszke, and Eugenio Culurciello. "An analysis of deep neural network models for practical applications." *arXiv preprint arXiv:1605.07678* (2016).

He, Kaiming, et al. "Deep residual learning for image recognition." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.

Exploring Randomly Wired Neural Networks for Image Recognition

Saining Xie Alexander Kirillov Ross Girshick Kaiming He

Facebook AI Research (FAIR)

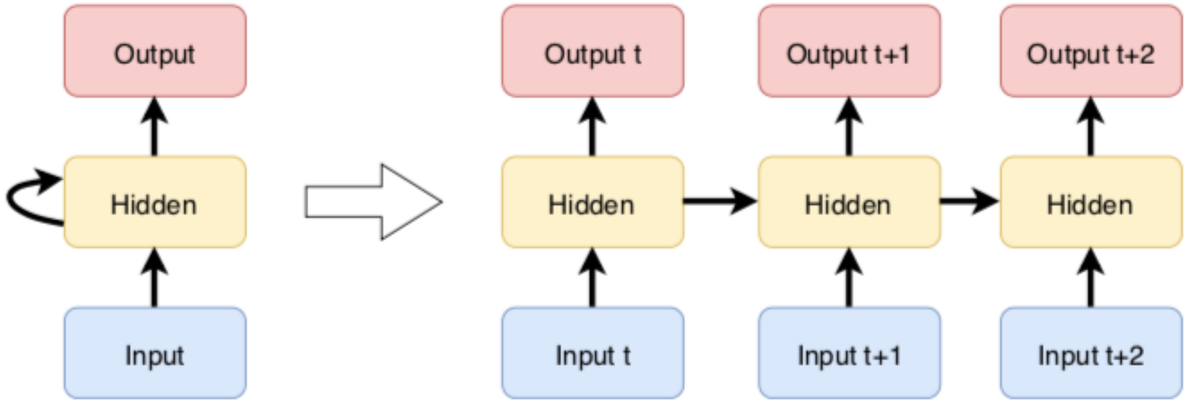


*** Source: Xie, Saining, et al. "Exploring** randomly **wired neural networks for image recognition." arXiv preprint arXiv:1904.01569 (2019).**

Deep Learning

Recurrent Neural Networks (RNNs)

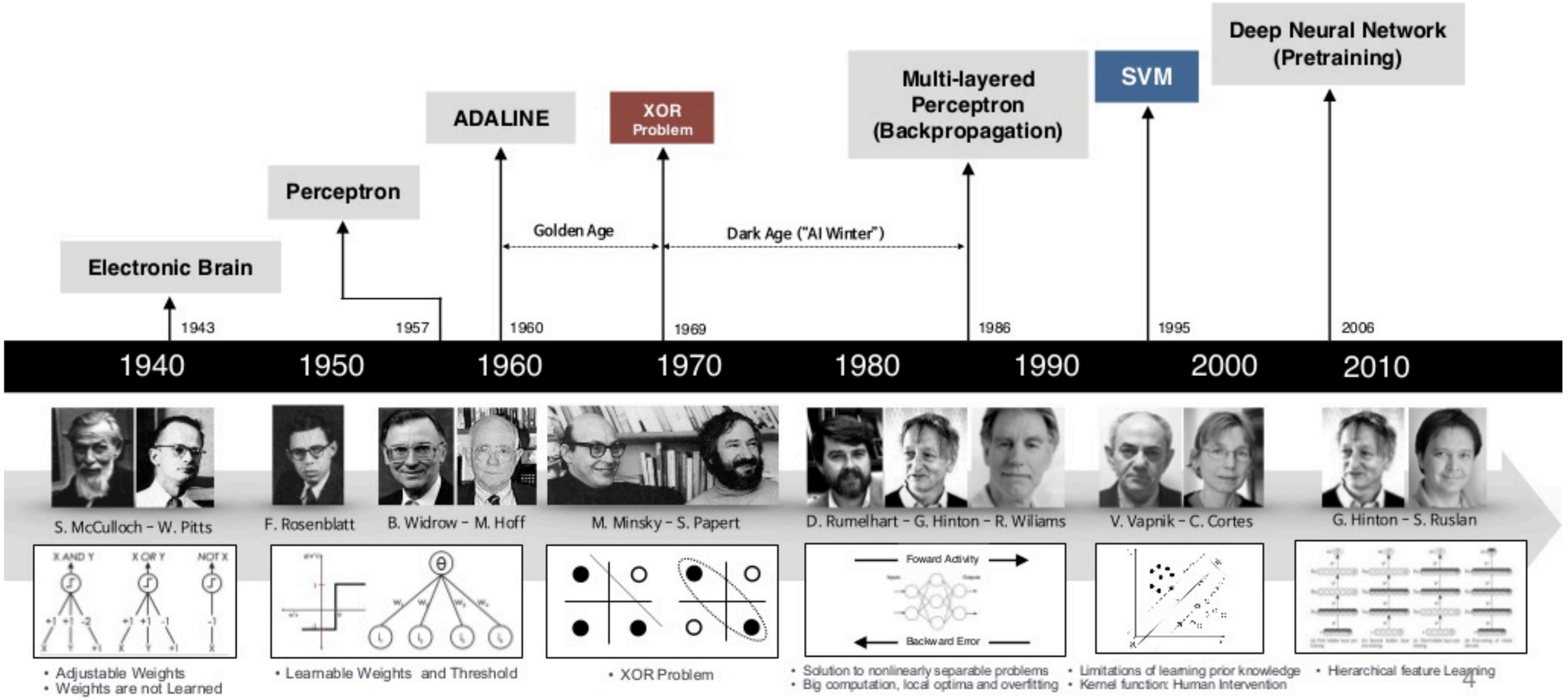
- 기존 Feed Forward Neural Networks (FNNs) 계열의 경우 시계열 또는 순서를 고려하지 못함
- 순서가 중요한 텍스트 데이터의 경우 RNN류의 모델을 쓰는게 적합
  - LSTM 또는 GRU



(a) 간소화된 표현 (b) 확장된 표현

Brief History of Neural Network

DEVIEW 2015



Naver D2, Sep 15, 2015, DEVIEW2015, https://www.slideshare.net/deview/251-implementing-deep-learning-using-cu-dnn

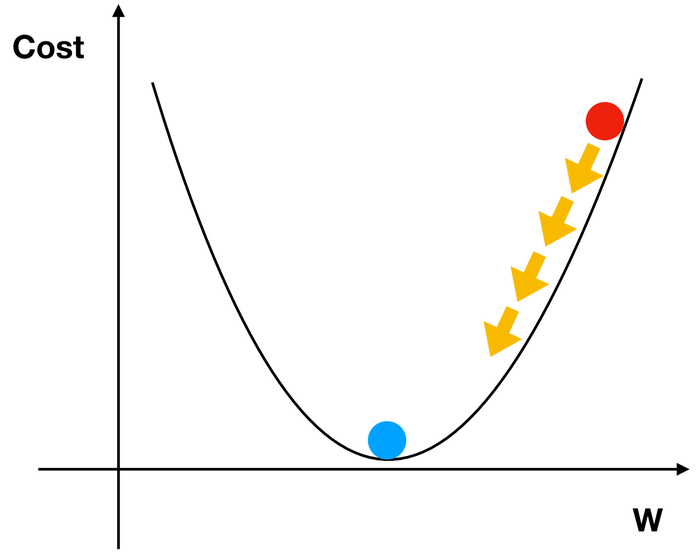
Deep Learning
이번엔 꺼지지 않는다!

- 훈련을 위한 데이터의 증가
- 하드웨어의 발전
  - 무어의 법칙
- 게임 산업의 발전에 의한 GPU 생산기술 발전
- 클라우드 컴퓨팅의 발전
- 기업의 투자
  - 인공지능에 대한 투자가 더 높은 수익을 가져올 수 있다는 것이 확인됨
  - 선순환 형성

Deep Neural Networks

How it works

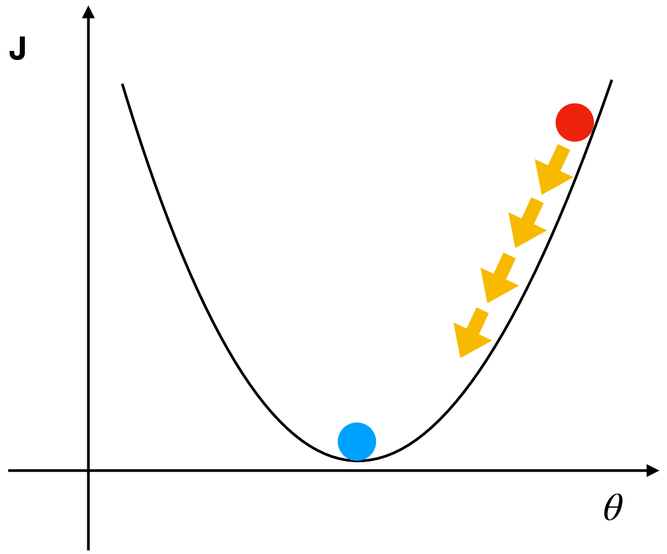
- 훈련을 한다는 것은?
- 다시 Linear Regression
  - $H(x) = \omega x + b$
  - $cost = \frac{1}{m}\sum_{i=1}^{m}(H(x^{(i)}) - y^{(i)})^2$
  - 학습 데이터가 주어졌을 때, cost가 최소가 되는, w와 b를 찾는 과정
  - 이때, 임의의 지점(빨간색)에서 시작해서, 기울기가 음수인 곳을 계속 찾아간다면 그 함수의 최소값에 도달하지 않을까?



Deep Neural Networks

How it works

- 미분을 이용한 경사하강법 (gradient descent)
  - $H(x) = \omega x + b,$
    $cost(\omega, b) = \frac{1}{m}\sum_{i=1}^{m}(H(x^{(i)}) - y^{(i)})^2$
  - $J(\theta_1, \theta_2) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^i) - y^i)^2$
    - $\theta_j := \theta_j - \alpha\frac{\partial}{\partial\theta_j}J(\theta_0, \theta_1)$
    - $\theta_j := \theta_j - \alpha\frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)x_j^{(i)}$



수렴할 때까지{

$\theta_j := \theta_j - \alpha\frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)x_j^{(i)}$

}

Deep Neural Networks

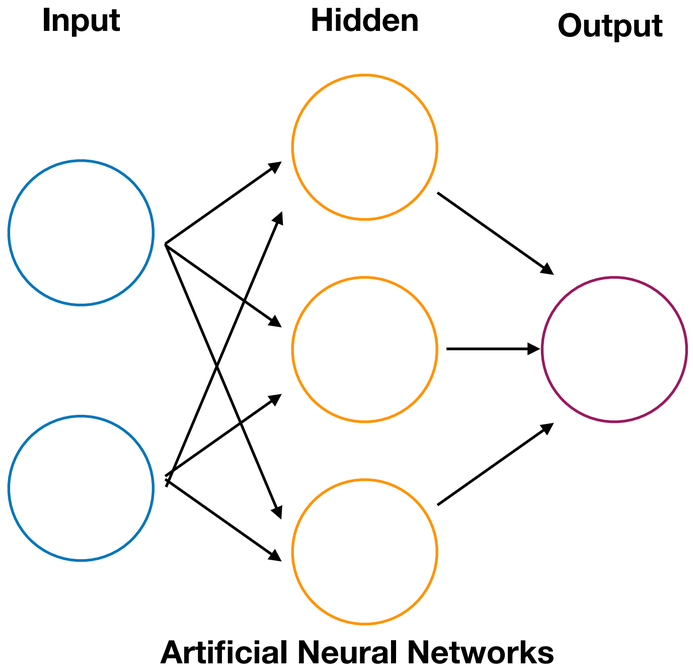
How it works

- Logistic Regression에서는
  - $H(x) = \frac{1}{1 + e^{-\theta^T x}},$
  - $J(\theta_1, \theta_2) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^i) - y^i)^2$
  - 많은 과정을 거친뒤!
  - $J(\theta) = -\frac{1}{m}[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^i) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)}))]$
    - $\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$

Deep Neural Networks

How it works

- Neural Networks에서는
  - $$J(\theta) = -\frac{1}{m}[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log h_\theta(x^i)_k + (1 - y_k^{(i)}) \log (1 - h_\theta(x^{(i)}))_k]$$
    - K= layer의 수
- Forward Propagation
  - 각 레이어의 아웃풋이 다음 레이어의 인풋
- Back Propagation
  - 각 레이어에서 미분을 통해 역으로 레이어 단위로 경사하강법 적용
  - 자세한 내용이 궁금하시다면!
    - https://www.youtube.com/watch?v=x_Eamf8MHwU&ab_channel=ArtificialIntelligence-AllinOne



Artificial Neural Networks

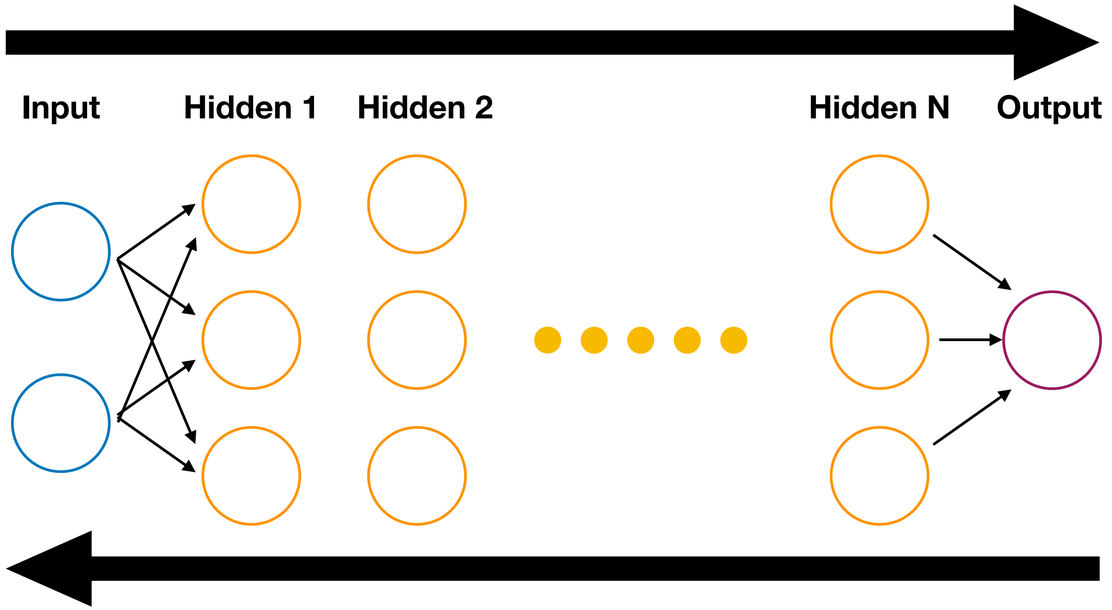
Deep Learning

Common Terms

- Epoch
- Batch
- Overfitting
- Activation function
- Loss function
- Optimizer
- Metric

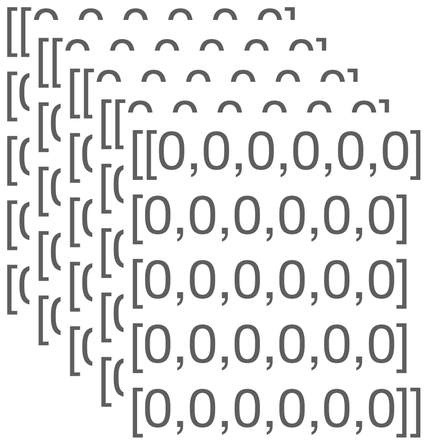
Deep Learning
Epoch

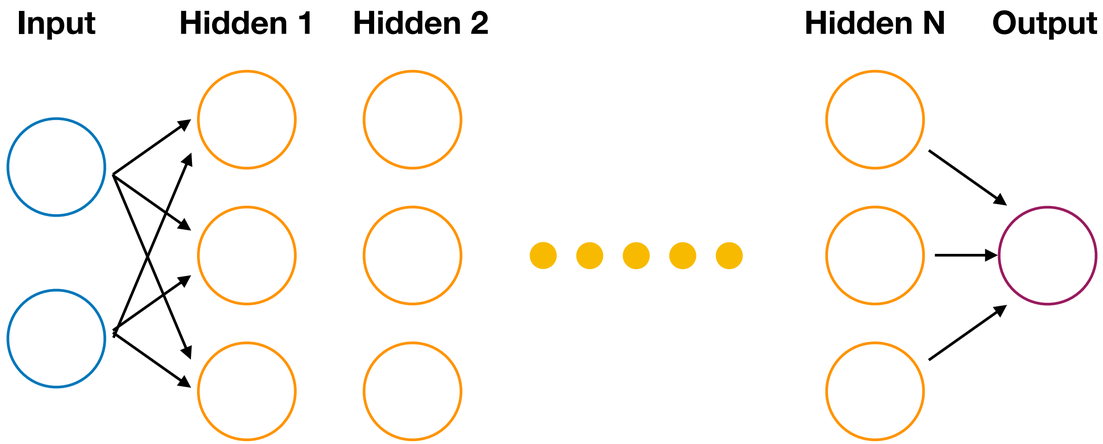
- Epoch
  - 전체 데이터를 몇번의 forward propagation과 backward propagation을 할 것인가
  - Low Epoch: Underfitting, High Epoch: Overfitting



Deep Learning
Batch

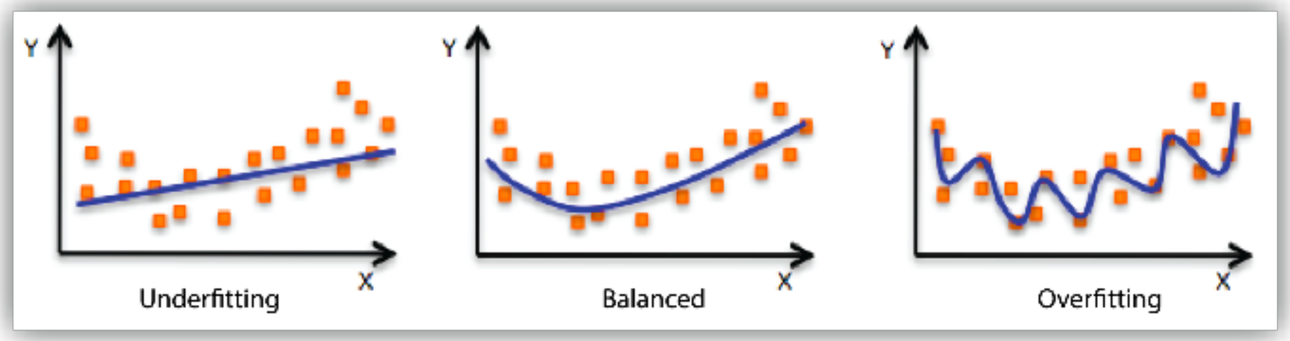
- Batch (Batch size)
  - 한 번의 Forward/Backword propagation에 몇개의 데이터셋을 이용할 것인가?
  - 총 1000개의 데이터셋을 128 batch size로 훈련 시키면, 128*7 + 104 = 8번에 나눠서 한 epoch을 구성
  - Low batch size: Longer training time, High batch size: More memory usage





Deep Learning
Overfitting

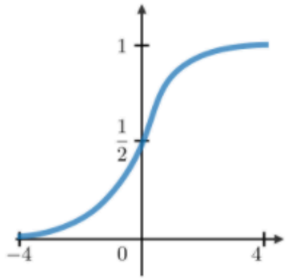
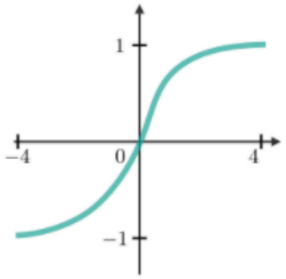
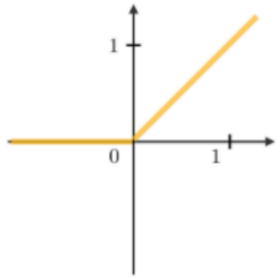
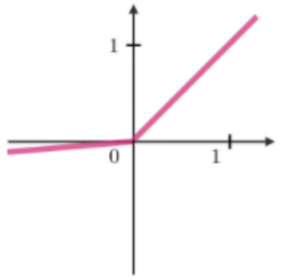
- Overfitting (과적합)
  - 훈련데이터셋에서 나오는 성능과 검증/테스트 데이터셋에서의 성능이 차이가 많이 나는 경우
    - 훈련 데이터셋에 Overfitting 된 상황!
  - 발생하는 이유
    - Validation 셋의 성능을 고려하지 않은 훈련
    - Feature의 개수에 비해 과도하게 많은 모델의 파라미터
    - 데이터 셋의 양에 비해 과도하게 높은 Epoch
  - Overfitting에 대응하는 방법
    - 가중치 규제
    - 드랍아웃 레이어 추가



https://docs.aws.amazon.com/machine-learning/latest/dg/model-fit-underfitting-vs-overfitting.html

Deep Learning
Activation Function

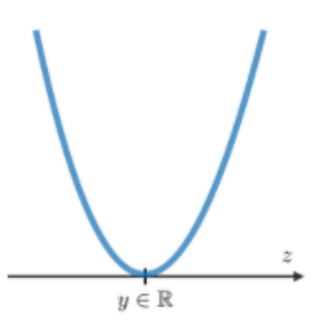
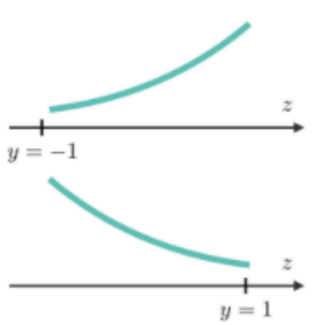
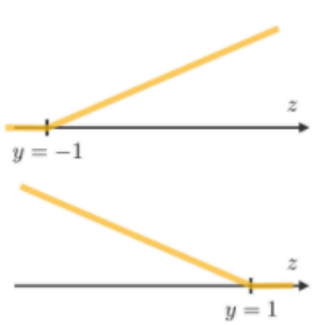
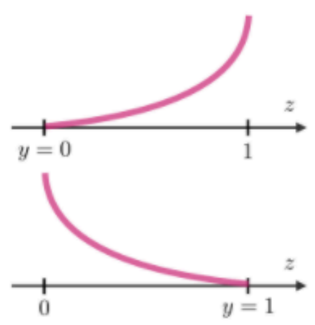
- Activation Function (활성함수)
  - 활성화 함수는 은닉 유닛의 끝에 사용되어 모델에 비선형 복잡성을 도입
  - 일반적으로, Hidden Layer에는 ReLU를 활용하고
  - Output에서는 Task에 따라서
    - Binary Classification: **sigmoid**
    - Multiclass Classification: **softmax**
    - Regression: **Linear**

Sigmoid	Tanh	ReLU	Leaky ReLU
$g(z) = \frac{1}{1+e^{-z}}$	$g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$	$g(z) = \max(0, z)$	$g(z) = \max(\epsilon z, z)$ with $\epsilon \ll 1$
			

https://stanford.edu/~shervine/teaching/cs-229/cheatsheet-deep-learning

Deep Learning
Loss Function

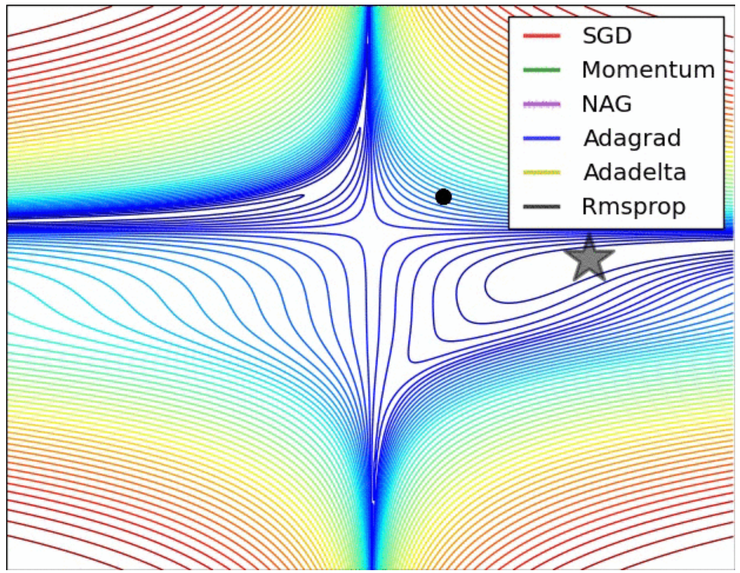
- Loss Function (손실함수)
  - 모델을 훈련을 시키기 위한 지표, 낮을 수록 데이터에 잘 적합되었다는 의미
  - **Cost function**: sum of loss functions + (something else)
  - Task나 Model에 따라 다름
    - Regression: Squared error
    - Classification: Cross Entropy

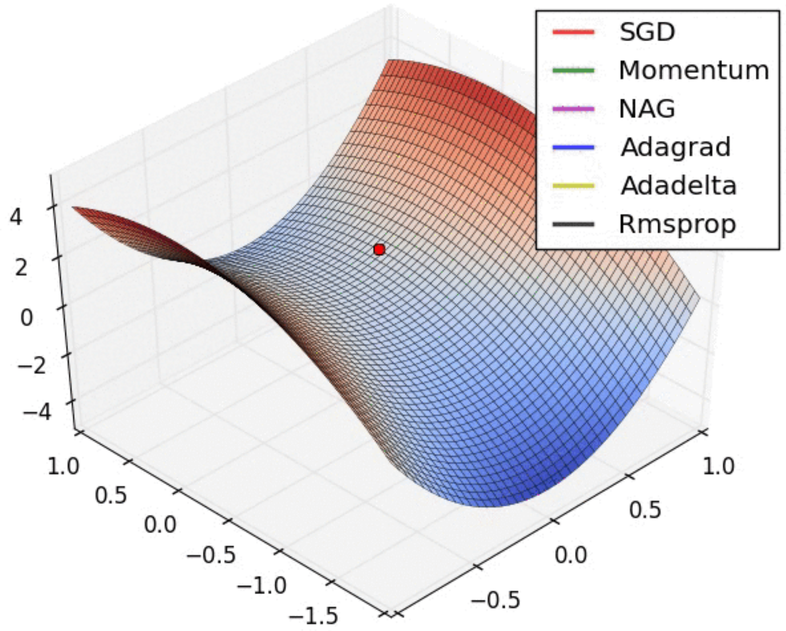
Least squared error	Logistic loss	Hinge loss	Cross-entropy
$\frac{1}{2}(y-z)^2$	$\log(1+\exp(-yz))$	$\max(0,1-yz)$	$-\left[y\log(z)+(1-y)\log(1-z)\right]$
			
Linear regression	Logistic regression	SVM	Neural Network

https://stanford.edu/~shervine/teaching/cs-229/cheatsheet-supervised-learning

Deep Learning
Optimizer

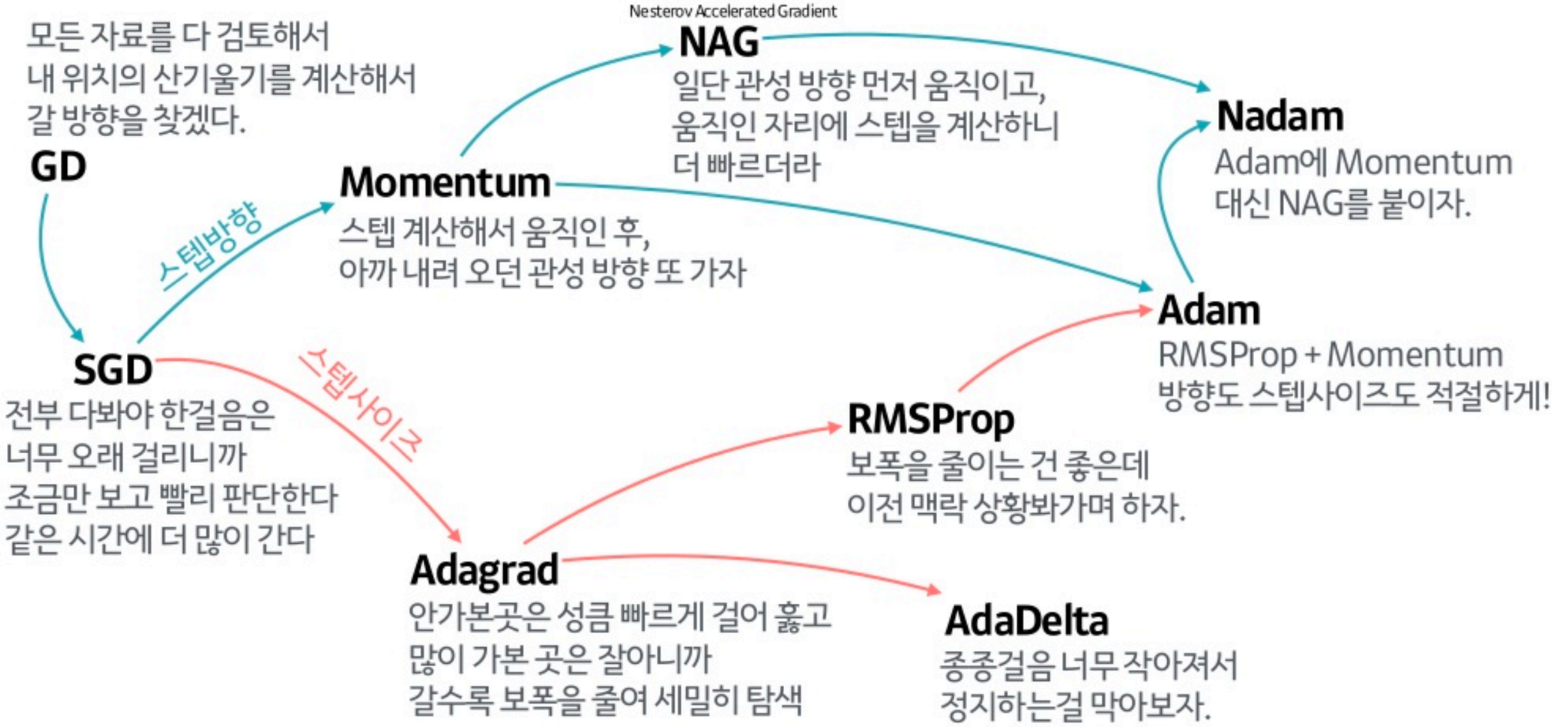
- Optimizer
  - 모델의 학습과정에서 경사하강법(mini batch) 사용시 Parameter를 업데이트 하는 방식
  - 어떻게 최적의 파라미터(cost함수 값이 가장 낮은)를 찾을 것인가!





https://ruder.io/optimizing-gradient-descent/

산 내려오는 작은 오솔길 잘찾기(Optimizer)의 발달 계보



https://www.slideshare.net/yongho/ss-79607172

Deep Learning

Metric

- Metric: 모델의 성능을 평가하는 지표들
  - Task에 따라 다른 metric을 사용
  - Classification: Accuracy, Precision, Recall, F1-score
  - Regression: MSE, MAE, RMSE
- Loss와 metric의 차이?
  - Loss는 모델을 최적화 여부를 판단하기 위한 지표
  - Metric은 모델의 성능을 평가하기 위한 지표
  - 일반적으로 둘이 강한 상관관계를 갖음 (Regression의 경우 동일)
  - 결국 모델은 Accuracy와 같은 것들을 고려하지 않고 본인이 Cost를 낮추는 방향이 맞는 방향이라고 생각하는 방식으로 훈련을 함

References

- Géron, Aurélien. Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems. O'Reilly Media, 2019.

E.O.D