Basic Machine Learning : Supervised Learning

1. 알고리즘(Algorithm)

- 문제를 해결하는 일련의 명령어
- 예전에는 문제가 주어졌을 때 해당 문제에 알맞은 알고리즘을 만드는 것이 일이였다.
- 그러나 머신러닝에서는 실제 데이터에서 문제를 특정 짓는 것부터 어렵다. 따라서 아래의 과정을 거친다.

1. 대략적인 문제의 정의가 있다.
2. 어떤 형태로든 데이터 훈련 샘플이 있다.
3. 머신러닝 모델이 이 문제를 해결하도록 훈련시킨다.

2. 지도학습(Supervised Learning)

Provided

- N개의 Input과 Output으로 구성된 훈련 샘플(Training Examples)
  $D = (x_1, y_1), (x_2, y_2), \cdots, (x_N, y_N)$
- Loss Function : 머신러닝 모델의 Output과, M(X) 실제 값들인 y를 비교하거나 평가할 지표
  $L(M(x), y) \geq 0$
- 모델을 평가할 검정 세트(Validation)와, 테스트 세트(Test) : 만든 모델이 기존에 보이지 않았던 데이터를 해결할 수 있는 가를 검정

Decide

Ⅰ. 가설집합(Hypothesis Sets) : 가설이란 이 문제를 풀기위한 알고리즘, 모델 아키텍처를 설정하는 과정
Ⅱ. 최적화 알고리즘(Optimization algorithm) : Loss를 낮출 수 있는 머신을 찾을 수 있는지에 대한 과정

모든 것이 결정되면, 지도 학습은 각 가설에 대하여 최적화 알고리즘을 사용해 제일 좋은 모델은 찾는다.

Given

- $D_{train} = (x_1, y_1), (x_2, y_2), \cdots, (x_N, y_N), D_{val}, D_{test}$
- $L(M(x), y) \geq 0$
- $H_1, H_2, \cdots, H_M$
- Optimization Algorithm

과정

1. [Training] 각 가설마다, Training Set을 사용해서 퍼포먼스가 제일 좋은 모델들을 찾는다.
   알고리즘 선택의 과정
   $\widehat{M}_m = \arg\min_{M \in H_m} \sum_{(x,y) \in D} L(M(x), y)$

2. [Model Selection] Validation Set을 사용해서 훈련된 모델들 중에 제일 좋은 모델을 선택한다.
   하이퍼 파라미터 선택의 과정
   $\widehat{M}_m = \arg\min_{M \in H_m} \sum_{(x,y) \in D_{val}} L(M(x), y)$

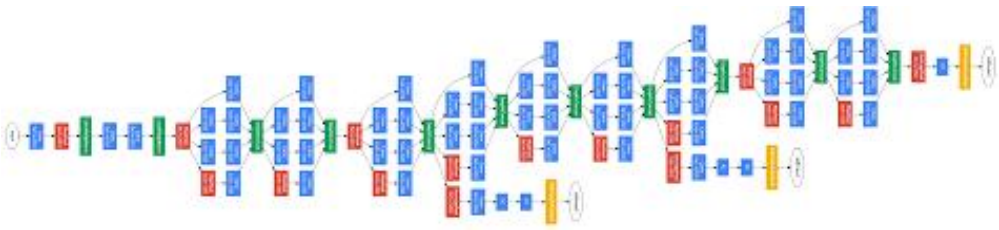
3. [Reporting] Test Set을 사용해서 제일 좋은 모델의 퍼포먼스를 측정한다.
   $R(\widehat{M}) \approx \frac{1}{|D_{test}|} \sum_{(x,y) \in D_{test}} L(\widehat{M}(x), y)$

3. 지도학습에서 가장 중요한 것

- 가설 집합(Hypothesis Set)
- 비용함수(Loss Function)
- 최적화 알고리즘(Optimization Algorithm)

4. 가설집합(Hypothesis Set)

- 가설 집합은 무수히 많다. 머신러닝 접근 방법, 모델 구조, 하이퍼 파라미터 등 요소를 하나씩 변화할 때마다 가설 하나가 세워지기 때문이다.
- 딥러닝에서 한정지어 얘기한다면, 네트워크 아키텍처를 선정하는 것이 중요하다. 네트워크 아키텍처가 정해지면, 하나의 가설집합이 된다.
- 각기 다른 가중치 매개변수 값에 따라 가설 집합에 속한 모델이 달라진다.



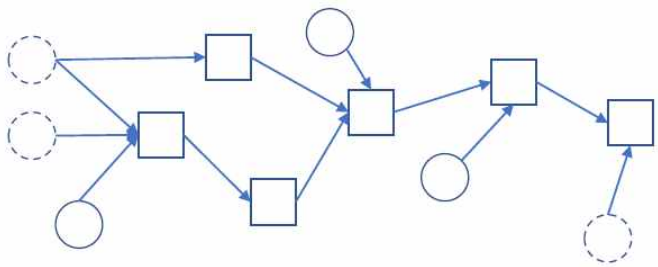
[네트워크 아키텍처 : GoogLeNet]

- Neural Network

  : Input data, 주어진 데이터, 값이 주어짐

  : parameters, 계속 변화시키는 값

  : computer nodes, 수시로 변하는 계산된 값

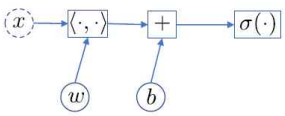


[Neural Network]

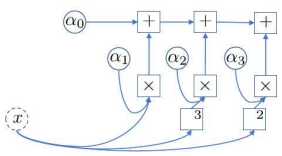
- Neural Network의 example

1. Logistic regression

$$p_\theta(y=1|x) = \sigma(w^\top x + b) = \frac{1}{1+\exp(-w^\top x - b)}$$



2. 3rd-order polynomial function $y = \alpha_0 + \alpha_1 x + \alpha_2 x^2 + \alpha_3 x^3$



- Neural Network는 Brain Science 이라기보다는 하나의 알고리즘을 계산 그래프로 표현하는 것이다.
- 지금은 계산 그래프의 방향이 순환이 아닌 일방향인 비순환 그래프(DAG)만 생각한다.
- 비순환 그래프 : Directed Acyclic Graph
- 각 노드들은 이미 Tensorflow, Pytorch 등 패키지에 잘 정의되어 있다. 분석가는 이 노드들을 어떻게 연결시킬 지에만 집중하면 된다.

5. 비용함수(Loss Function)

- 우리는 Loss Function이 최소가 되도록 Optimization을 해야 한다.
- 이미 많이 알려진 Loss Function이 있지만, 쓰기가 힘든 것들이 많다.
- 확률분포에 따라서 비용함수를 정의해야 한다.

6. 확률(Probability)

- 사건(Event) : 모든 가능한 사건의 집합
- 확률변수(Random Variable) : 사건 집합 안에 속하지만 정의되지 않은 어떤 값
- 확률(Probability) : 사건 집합에 속한 확률변수에게 어떤 값을 지정해주는 함수
- 확률의 특징 : 확률은 비음수(Non-negatives)이고, 모든 확률의 합은 1(Unit Volume)이다.
- 결합 확률(Joint probability) : 확률변수는 1개 이상이다.
- 조건부 확률(Conditional probability) : 어떤 확률변수의 특정 조건에서의 확률
- 한계확률(Marginal probability) : 두 개 이상의 확률변수를 고려할 대 어느 한 쪽만의 확률(주변 확률)

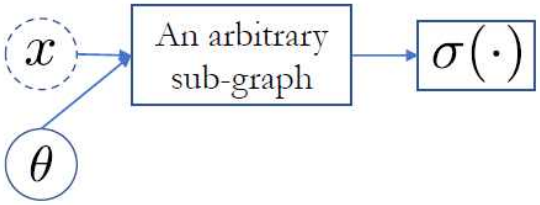
7. 지도학습에서의 비용함수

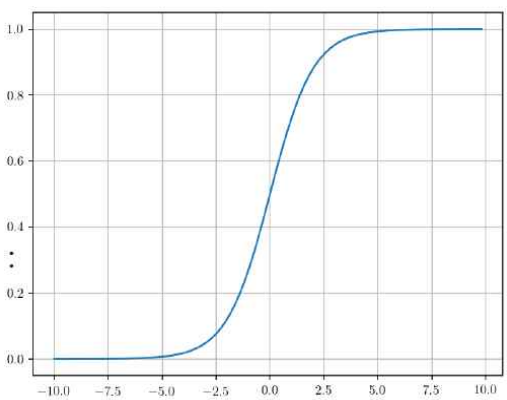
- 지도학습은 Input(x) 값을 넣었을 때 Output(y) 값을 산출하는 것이다. 조금 다르게 생각하면, Input(x) 값이 주어졌을 때 Output(y)의 값이 y' 일 확률을 구하는 것이다.
- 확률분포의 종류

  - 이진분류 : 베르누이(Bernoulli) 분포
  - 다중분류 : 카테고리(Categorical) 분포
  - 선형회귀 : 가우시안(Gaussian) 분포
  - 다항회귀 : 가우시안 믹스쳐(Mixture of Gaussian) 분포

- Bernoulli Distribution

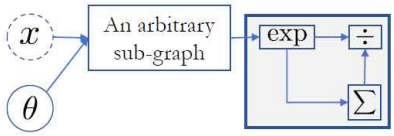
  - 확률 : $p(y|x) = \mu^{y}(1-\mu)^{1-y}, \quad where \ \ y \in (0,1)$
  - 추정된 Output : $\mu \in [0,1]$
  - sigmoid Function : Output을 0과 1사이로 매칭 시켜주는 함수, $\sigma(a) = \dfrac{1}{1+e^{-a}}$





- Categorical Distribution

  - Category : $C(\mu_1, \mu_2, \mu_3, \mu_4, \cdots, \mu_C)$
  - 확률 : $p(y=v|x) = \mu_v, \quad where \sum_{v=1}^{C} \mu_v = 1$
  - softmax Function : Output을 0과 1사이로 매칭 시켜주는 함수, $softmax(a) = \frac{1}{\sum_{v=1}^{C} \exp(a_v)} \exp(a)$



- Gaussian Distribution

  - 확률 : $p(y|x) = \frac{1}{Z} \exp(-\frac{1}{2}(y-\mu)^T(y-\mu))$
  - Fully characterized : $\mu \leq R^2$
  - Neural Network에서는 주어진 Input(x) 값을 Vector로 변환시키는 과정이 반드시 필요하다.

- 인공신경망 모델이 조건부 확률 분포를 출력하면, 이를 사용해서 비용함수를 정의할 수 있다.

  - 최대한 모델이 출력한 조건부 확률 분포가 훈련 샘플의 확률분포와 같게 만드는 것이다.
  - 즉 모든 훈련 샘플이 나올 확률을 최대화 하는 것이다.
  - $\arg\max_\theta \log p_\theta(D) = \arg max_\theta \sum_{n=1}^{N} \log p_\theta(y_n|x_n)$
  - 이렇게 함으로써 자동으로 비용함수를 정의할 수 있다. 이를 최대 우도 추정(MLE, Maximum Likelihood Estimation)이라고 한다.
  - Log를 사용한다, 최소화를 위해서 -1을 곱해준다
  - 최종적으로, 비용함수는 음의 로그확률(Negative Log-probabilities)의 합으로 결정된다.

  $L(\theta) = \sum_{n=1}^{N} l(M_\theta(x_n), y_n) = -\sum_{n=1}^{N} \log p_\theta(y_n|x_n)$

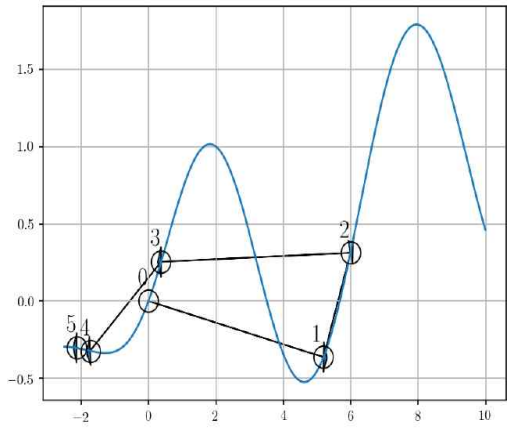
8. Optimization Methods

- Loss는 비순환 그래프(DAG)를 거쳐 계산된다.
- 가설이 무수히 많기 때문에 모든 것을 다 시도해보고 최적인 것을 고를 수가 없다.
- 따라서, 일단 아무 곳을 선택한 후에 Loss를 낮추는 방향으로 최적화를 진행한다.
- Local, Iterative Optimization: Random Guided Search

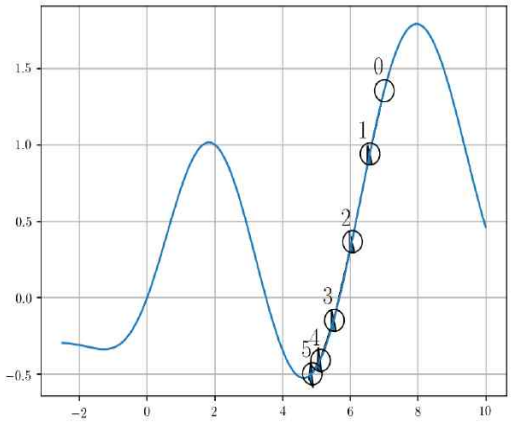
  - 장점 : 어떤 Loss Function을 사용해도 무관하다.
  - 단점 : 차원이 적을 때는 잘 되지만, 차원의 저주 때문에 커질수록 오래 걸린다.
    Sampling에 따라서 오래 걸린다.

- Gradient-based Optimization

  - 미분을 통해 최적화 할 방향을 정한다.
  - 장점 : Random Guided search에 비해서 탐색 영역은 작지만 확실한 방향을 정할 수 있다.
  - 단점 : 학습률(Learning rate)이 너무 크거나 작으면 최적의 값으로 못갈 수도 있다.



[Local, Iterative Optimization]



[Gradient-based Optimization]

9. 자동미분법(Automatic differentiation)

- 모델을 연결하는 비순환 그래프는 미분 가능한 함수들로 구성되어 있다. 따라서 미분의 연쇄법칙을 사용하여 자동으로 미분한다.
- 각 노드에서 Jacobian-vector product을 계산한다. DAG 역순으로 전파한다.
- $\frac{\vartheta(f \circ g)}{\vartheta x} = \frac{\vartheta f}{\vartheta x}\frac{\vartheta g}{\vartheta x}$
- Gradient를 손으로 직접 계산할 필요가 없어졌다.
- [Front-End]와 [Back-End]의 분리
  - [Front-End] : 비순환 그래프(DAG)를 만듬으로써 Neural Net 모델을 디자인하는데 집중한다.
  - [Back-End] : 디자인 된 비순환 그래프(DAG)는 타겟 컴퓨터 장치를 위한 효과적인 코드로 컴파일 된다. Framework 개발자들이 타겟에 맞춰 알맞게 구현하면 사용자들은 Back-end를 신경 쓰지 않고 쓸 수 있다.
- 쉽게 말해 네트워크 아키텍처를 구현할 때 back-end 부분(어떤 환경에서 계산을 수행할지 등, cpu, gpu, Mobile)을 신경 쓰지 않고, 디자인에만 신경 쓰면 된다.

10. Gradient-Based Optimization

- 자동미분법을 통해 계산하는 최적화 알고리즘을 역전파(Backpropagation)이라고 부르는데, 1986년에 소개 되었다. 이 알고리즘을 쓰면, Loss와 Gradient를 계산할 수 있다.
- Gradient descent, L-BFGS, Conjugate gradient 등 방법들이 많다.
- 그러나 이러한 방법들은 매개변수(Parameters)가 많아질수록 오래 걸린다.
- 그 이유는 훈련 샘플 전체의 Loss 각 샘플에 대한 Loss의 합으로 구해지며, 데이터가 많아질수록 계산 시간이 오래 걸리기 때문이다.
- 따라서 일반적으로 확률적 경사 하강법(Stochastic Gradient descent)을 사용한다.

11. Stochastic Gradient descent

- 가정 : “전체 Loss는 일부를 선택 후 나눠서 계산한 전체 Loss의 근사값(Approximate)과 같다.”
- M개의 훈련 샘플을 선택한다. 이를 미니 배치(Mini Batch)라고 한다.

  $D' = (x_1, y_1), \cdots, (x_{N'}, y_{N'})$
- 미니배치의 경사(Gradient)를 계산한다.
- 매개변수(parameter)를 갱신한다.

  $\theta \leftarrow \theta + \eta \nabla L(\theta; D')$
- 검증 세트로 구한 validation loss가 더 이상 진전이 없을 때까지 계산한다.

12. Early Stopping

- Optimization을 계속 진행하면 할수록 가설이 늘어난다는 단점이 있다.
- Validation의 Loss가 가장 낮은 곳에서 훈련을 중지한다.
- 과대적합(Overfitting)을 방지하기 위한 제일 좋은 방법이다.

13. 적응적 학습률(Adaptive Learning Rate)

- 확률적 경사 하강법은 학습률에 매우 민감하다. 이를 보완하기 위해서 Adam, Adadelta 등 다양한 알고리즘을 사용한다.
- 궁극적으로는 학습률도 하이퍼 파라미터로 보고 여러 가지를 고려해야 하지만, 빠른 결과를 위해서는 적응적 학습률을 사용하기도 한다.