2일차

p.65) Machine learning

For an output Y and input X=(X1, X2, X3, ... X_p), the relation can be generally presented by $Y = f(x) + \varepsilon$.

Classification : Y가 class이거나 type일 경우 사용

p.67) Parameter Tuning : Training errors and test errors are different. -> 이를 통해, 모델이 점점 복잡해지거나, 단순해진다.

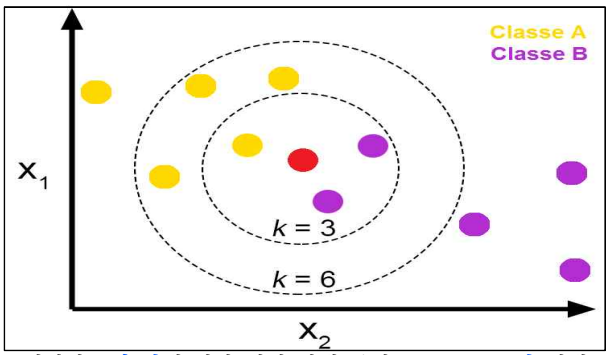
- Finding the best parameter and model through the validation approach.

Model - 1. K-nearest Neighbor(KNN)

$\hat{Y} = \frac{1}{K}\sum Y_k$. (Determine the outcome of a sample using the k nearest samples of known outcomes.)

- 단순하지만, 비선형적인 모델이 만들 수 있다.(Totally non-parametric, simple but powerful.)

K is tuning parameter controlling the model flexibility. -> 예) 12개의 샘플 중 K개의 가까운 것만 뽑는다.



그림처럼 K의 값에 따라 범위 안에 들어오는 sample값 선택.

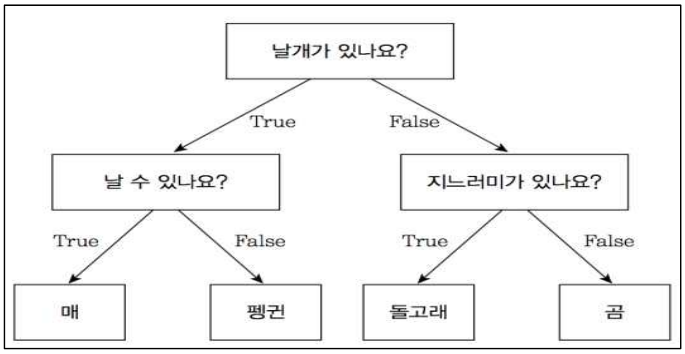
2. Naive Bayesian : classification에만 적용가능.

$$\Pr[Y|X_1, X_2, X_3, \ldots X_p] = \frac{\Pr[X_1, X_2, \ldots X_p|Y]\Pr[Y]}{\Pr[X_1, X_2, \ldots, X_p]}$$

In general, $\Pr[X_1, X_2, X_3, \ldots X_p|Y]$ is very difficult to estimate because it is a joint distribution in a p-dimensional space.

When X is continuous, we assume Gaussian distribution.

3. CART(Decision Tree) : The goal of the partitioning is to the subregions $R_1, R_2, R_3, \ldots R_j$ that minimizes the loss. (처음 자를 때는 어떤 기준으로 자를지 각각 다르다.)

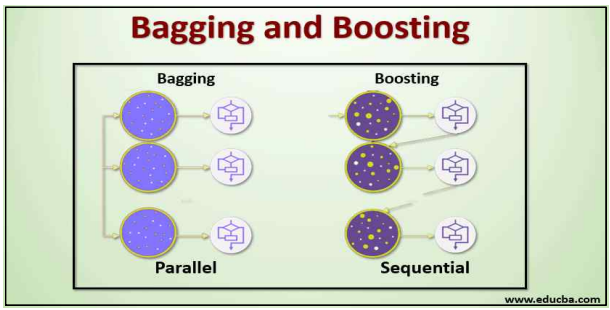


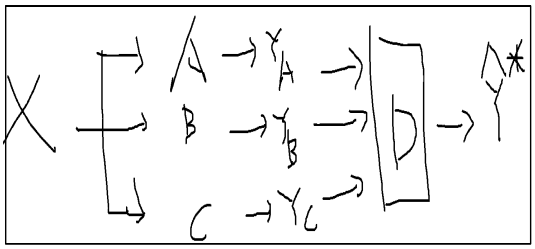
Cons(단점) : low accuracy(정확도가 낮다) -> 앙상블

Pros(장점) : high interpretability(해석이 용이하다), mimicking human decision, visualization, non-parametric

The tree size(# of leaf nodes) is the tuning parameter.

4. Bagging : bootstrap aggregating의 줄임말로 통계적 분류와 회귀 분석에서 사용되는 기계 학습 알고리즘의 안정성과 정확도를 향상시키기 위해 고안된 일종의 앙상블 학습법의 메타 알고리즘이다.





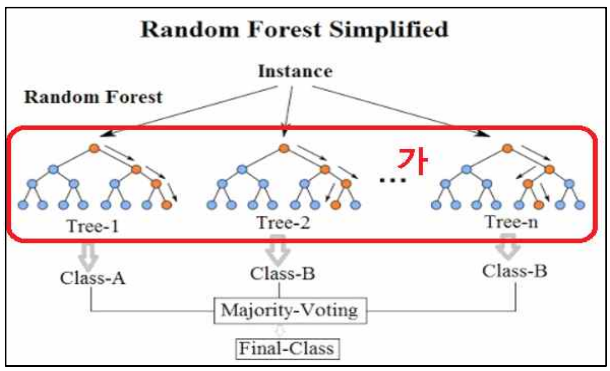
A, B, C 각각 다른 방법 사용

앙상블 대표적인 방법 : bagging, boosting

$\hat{Y}^*$을 구할 때 평균값을 통해 구한다. $\hat{Y}^* = \frac{1}{B}\sum_{i=1}^{B}\hat{f}_i(X)$

- Bagging : training model over multiple bootstrapped sets, and aggregating into the final model
- A bootstrapped set is generated by resampled with replacement from the training set.
- Aggregation can be done by averaging for regression, or by major voting for classification.

5. Random Forest : From a training data set, generate a set of tree predictors and make the final decision by aggregating the predictions of multiple tree predictors.



- (가) : Building the models : for each subsample, a decision tree is constructed based on a random se of m feature, the results fall into leaves

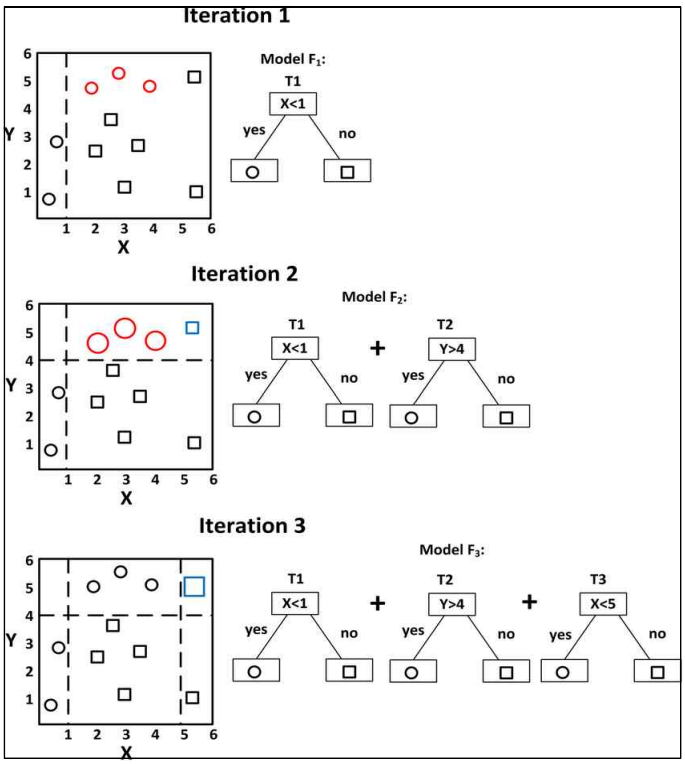
순서 : Bootstrap sampling - Building the models - Bootstrap aggregating

6. Gradient Boosting

- Boosting is a general method to improve prediction accuracy, but here we are focusing on tree-based gradient

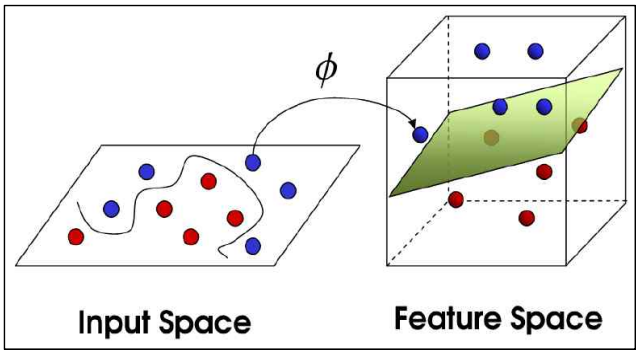
boosting.

- Gradient boosting learns sequentially and slowly using many simple trees, while bagging learns everything at once and combines many full trees.

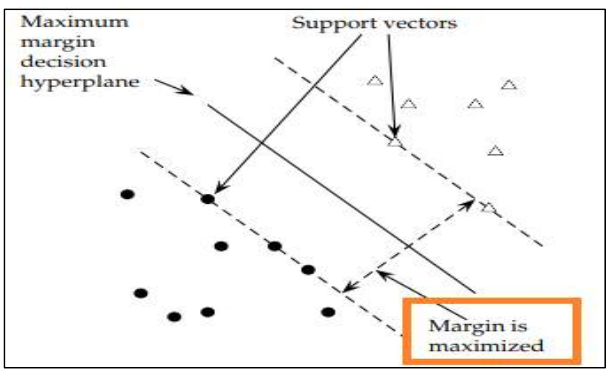


7. SVM : Maximal margin + Soft Margin + Kernel

- Soft margin : allowing samples within the margin(margin안에 sample이 들어올 수 있고, 그 정도, 양을 최소 기준으로 마련)
- Kernel : transforming data so that a hyper plane can be easily found. (고차원으로 변경)



- maximal margin classifier : classifying the class by the hyper plane with the maximal margin.



기준 선에서 가까운 것들이 구분이 잘 안 되는 점인데, 이 둘 점 사이의 최대 거리가 maximal margin classifier

NN : Learning model mimicking brain neurons.

- Input variables are connected to hidden variables, and hidden variables generate outputs.
- Each connection is non-linear connection : sigmoid or tanh is often used.(각각의 연결은 비선형 관계이고, 이를 표현하기 위해서, "tanh" or "non-linear connection"으로 표현 가능.
- The connection mimics the chemical transfer between neurons.

DNN : deep learning(Deep Neural Network)

- It can handle more complex problems by providing more complex models.

Unsupervised learning : clustering, dimension reduction(비지도 학습)

- Clustering : a very broad set of techniques for finding subgroups or clusters in a data set.
- Grouping samples similar together.
- We need to define what is similar or dissimilar : distance or dissimilarity.

K-Means Clustering의 일반화 버전 : Gaussian Mixture

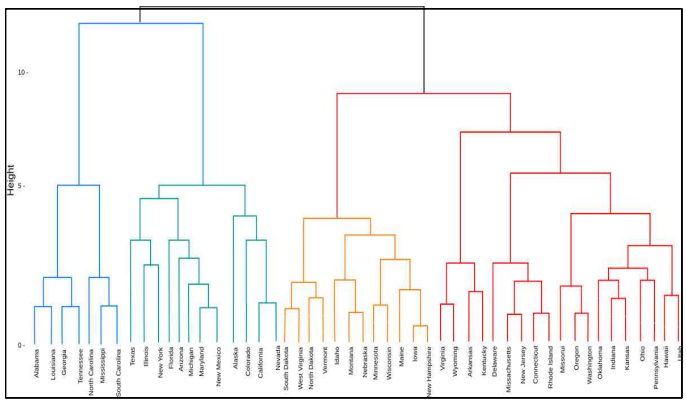
- Modeling with several Gaussian distributions(가우시안 분포), and finding their distributions.

기존의 2가지 방법은 어떤 거리를 특정하는 식의 평평한 경우 분포를 알아내는 방법

Hierarchical Clustering

- 계층을 만들면서 올라가는 것.(비슷한 것 끼리 묶어서)

-making a hierarchical structure among clusters which is presented as a upside-down tree or [1]dendrogram.



- Agglomerative : bottom-up build, it is built starting from the leaves and combining clusters up to the trunk.
- options : distance metric and linkage.

예) 30개의 sample 중에 비슷한 것 하나를 하나의 group으로 다른 sample과 결합하면, 총 sample은 29개가 된다. 이런 방법을 반복하여 적정 수준의 group을 만드는 방법.

1) 덴드로그램은 클리스터링의 결과를 시각화하기 위한 대표적인 그래프입니다.

인공지능 기초/응용 SW programming 강좌 : deep learning with pytorch를 중심으로 - 이재구(국민대 교수)

outline
1. introduction
2. neural networks과 deep learning의 기초
3. pytorch 기초 실습
4. deep learning 응용
5. deep learning 최신 경향

- Nvidia CEO : Software is eating the world, but AI is going to eat software : Jensen Huang predicts that health care and autos are going to be transformed by artificial intelligence.

AI : Artificial intelligence
- the simulation oof human intelligence processes by machines(computer systems)

ex) self-correction, knowledge ...

AI in production(생활 속 AI)
- Robotics, Real-time object recognition, Autonomous driving, Recommender systems

(서비스나 마케팅적으로 AI라는 용어 사용하기도 한다.)

A brief timeline of historical events in AI
Electronic Brain - Perceptron - Adaline - Xor Problem - Multi-layered Perceptron(Back Propagation) - SVM - Deep Neural Network(Pretraining)

understanding and imitating the brain
1950) one neuron : perceptron
1980) several neuron : multi layer perceptron
2010) many neuron : deep neural networks

Deep learning - ex) MLPs(Multi layer perceptron) < Machine Learning - ex) Logistic regression < AI

classic machine learning : input -> **hand designed feature(사람 개입)** -> mapping from features -> output

Representation learning : 1) input -> features -> mapping from features -> output
2) input -> simple features -> additional layers of more abstract features -> mapping from features -> output

performance : Deep Neural Networks > Medium Neural Networks > Shallow Neural Networks > Traditional Machine Learning

General-Purpose computing on Graphics Processing Units)
- General-Purpose computing on Graphics Processing Units의 머릿글자로, 직역하면 'GPU의 범용 연산'.
- CPU가 맡았던 연산을 GPU에도 사용해 연산 속도를 향상 시키는 기술. 흔히 '하드웨어 가속'이라고 하면 GPU를 가리키는 경우가 많다.

Background for deep learning
- Definition of machine learning : Tom Hichell(1998) = A program is said to learn from experience 'E' with respect to any task 'T' and some measure of performance 'P' if their performance in 'T' measured by 'P', improvement experience 'E'.

E(Experience)*T(Task)=P(performance)

- 강화학습, 지도학습, 비지도 학습
비지도 학습 : unsupervised learning
give(x) just dat, **no label**

Goal : learn some underlying hidden structure of the data

ex) clustering, dimesionality reduction, feature learning, density estimation

강화학습 : **Reinforcement learning** : given Problems involving an **agent** interacting with an **environment** which provides numeric **reward** signals.

ex) learn how to take actions in order to maximize reward
- robotics

background for deep learning
components of learning
- metaphor : credit approval(신용평가)

formalization
x : input - customer application
y : output - approve or deny

f=P(y|x) : target distribution - ideal credit approval formula
ex) $P(y|x)$: UNKNOWN Target distribution(target function f plus noise)

$(x_1,y_1),(x_2,y_2),....,(x_N,y_N)$: data - historical records
$g:x\rightarrow y$: hypothesis - formula to be used

Freom linear algebra
- feature vector : toy example(image -> vector)

vector는 공간과 관련이 있다.

inner product : 곱, 합의 성질

effect of representation learning in algebraic viewpoint
- representation matter : example of different coordinates : cartesian coordinates(데카르트 좌표계), polar coordinates (극좌표계)

- not only the mapping from representation(input) to output but also the representation itself.

- can find **good features** more **rapidly than human** -> better performance

representation learning : automatically discover the representations needed for feature detection or classification from raw data.

deep learning(or representation learning)
- deep neural network with multiple levels of linear/non-linear operations

1) hierarchy of representations with increasing level of abstraction

2) each stage : a kind of trainable feature transform

learning representation -> data-driven features.

deep neural network = universal approximator.
(layer가 많을수록 추상화가 될 확률이 높아진다.)

ML vs DL
ML : input -> feature extraction -> classification -> output(사람 개입)
Deep Learning : input -> feature extraction + classification -> output(car, not car)

Neural networks
- inner product = linear function(내적 자체가 선형)
- activation function = non - linear function

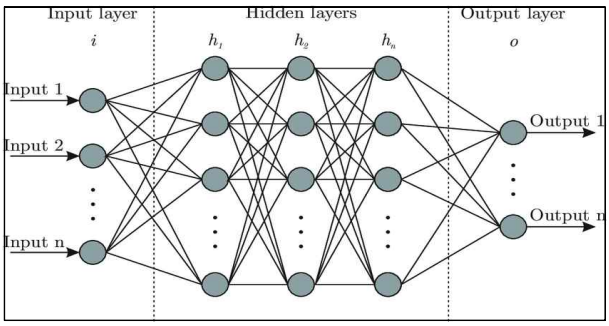
activation function 자주 사용 : sigmoid(logistic), rectifier(ReLU), sigmoid(tanh)

f(x,W) = Wx+b (선형일 경우, 유사도 판정, 판단)
-> find parameter (W, b) for correct answer!, b: bias
extension to deep neural network = multiple levels of linear + non linear operations.

the perceptron implements h(x)=sign(w^Tx)

XOR problem

Neural networks - structure



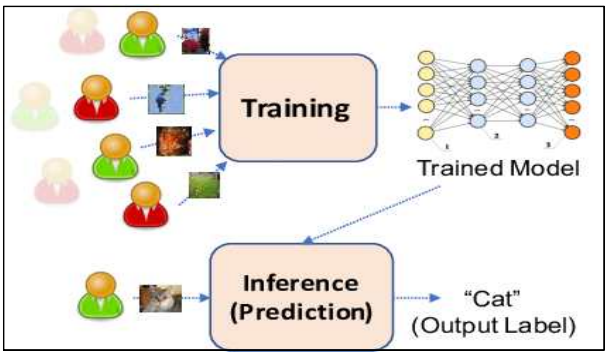
input, output은 상황에 따라 달라지는 구조

신경망에 존재하는 파라미터, 가중치는 초기에 랜덤 초기화가 된다.
데이터를 입력하여 학습을 통해, 점점 조정해나간다.(tunning을 통해 최적화를 해나간다.)
가중치(w) : 비선형 값

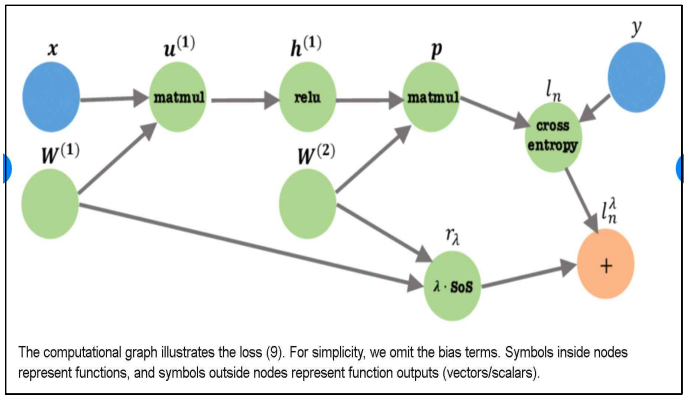
- training : 모의고사, inference : 수능(predicting label for unknown data)

수많은 컴퓨터를 이용해야 하는 딥러닝 분산처리의 요구사항을 도출하기 위해 딥러닝 기술을 분석하면 크 게 트레이닝(training)과 인퍼런스(inference)로 나뉜다.

트레이닝(training)은 **입력 데이터(data)**를 통해 모델을 **학습**하는 과정이고 **인퍼런스(inference)**는 **학습된 모델**로 인식 등의 **서비스**를 수행하는 과정이다.



- 위의 inference를 보면 고양이 사진을 input으로 넣어서, 고양이인지 아닌지를 구분하는 서비스, 동작, application을 수행하고 있다.
- 그러나 **training**은 inference에 쓰일 model을 input과 output을 통해 model이 향후 **좋은 성과**를 낼 수 있도록(만들 수 있도록) 모델을 학습시킨다.



Computational graph : 연쇄적 동작 수행
Forward pass <-> Backward pass(오류 역전파)

Backpropagation :

iterative optimization : 1) instead of analytically setting
2) **gradient descent is a very general algorithm**
- gradient = the **derivative** of vector functions
- direction of greatest increase of a function
- repeatably update weights, iterate to next step

$$w(t+1) = w(t) - \eta \nabla E_{\in}(w(t))$$

compute activations : forward propagation
compute derivatives : back propagation
w가 바뀌면 Loss가 바뀐다.

chain rule : 합성 곱

gradient descent
- full batch gradient descent -> 한 번 하려고 데이터 전체를 계산해야하기에 효율적이지 않다.
- stochastic gradient descent : 효율적인 계산을 위해서 사용. 속도가 빠르다. 치우침 발생 가능. random값 한 개로 측정.
- mini-batch gradient descent : random값 여러 개로, batch를 여러 개를 통해 계산한다.

고차원 일수록, 안장 점을 피하는 것이 문제.

성능평가 : layer수에 따라 달라지는 것(layer수가 많을수록 성능 good), Regularization λ값에 따라서(값이 작을수록 성능 good)
-> 주의사항 : over fitting

regularization(정규화) 전략
1. bagging
2. dropout
3. batch, layer, instance normalization

self-supervised learning : 일반적으로 Supervised Learning(지도학습)이 높은 성능의 모델을 만드는 것이 유리하지만, 수많은 데이터에 label을 전부 달아야 한다는 점에서 **데이터 셋 모으기가 어려우며 따라서 활용하는 방법도 제한적일 수밖에 없다**.

이와 같은 문제를 해결하고자 나온 방법이

아주 일부분만 label이 존재하는 데이터셋을 사용하는 Semi-Supervisd Learning(준지도 학습)

label이 전혀 없는 데이터 셋을 사용하는 Unsupervised Learning(비지도 학습)

이고, 최근 주목받는 연구 방법이 **Self-Supervised Learning(자기지도 학습)**이다.

보통 Self-Supervised Learning을 연구할 때, 다음과 같은 과정을 따른다:

1. **Pretext task**(연구자가 직접 만든 task)를 정의한다.
2. Label이 없는 데이터 셋을 사용하여 1의 Pretext task를 목표로 모델을 학습시킨다. 이때, 데이터 자체의 정보를 적당히 변형/사용하여 (label은 아니지만) 이를 supervision(지도)으로 삼는다.

3. 2에서 학습시킨 모델을 Downstream task에 가져와 weight는 freeze시킨 채로 transfer learning을 수행한다(2에서 학습한 모델의 성능만을 보기 위해).

4. 그래서 처음에는 label이 없는 상태에서 직접 supervision을 만들어 학습한 뒤, transfer learning 단계에서는 label이 있는 ImageNet 등에서 Supervised Learning을 수행하여 2에서 학습시킨 모델의 성능(feature를 얼마나 잘 뽑아냈는지 등)을 평가하는 방식이다.

- training - exploration of hypothesis(parameters) space