인공지능 > 머신 러닝 > 딥러닝

심볼릭 AI < - 옛날 지배적인 패러다임 (체스 게임) : 잘 정의한다면 논리적인 문제해결가능

머신러닝 : 명시적인 규칙이나 프로그램 없이 데이터로부터 학습하는 능력을 갖는 것

딥러닝 : 머신러닝 중 신경망을 사용하는 것

What is Machine learning?

전통적   
데이터 -> 일반 프로그래밍(규칙) - > 답 (어느정도 output을 알고있음)

데이터 / 답 -> 머신러닝 -> 규칙 (미리 답을 정의해놓고 그 룰을 만들어냄)

머신러닝의 특징

트레이닝 과정이 필요 -> 고성능 하드웨어 / 대량의 데이터  
통계와는 다르게 대용량의 복잡한 데이터셋을 다룸

딥러닝   
엔진니어링측면에서 효율화 개선 -> 경험으로 바탕으로 아이디어를 증명하는 경향이 있음

머신러닝이 어디에 유용한가?

1. 이미지 분류
2. 뇌종양 스캔(의학)
3. 챗봇
4. 보이스
5. 추천 / 예측

Reinforcement Learning: Robots, Games, 자율주행

<딥러닝> Represation Learning

데이터로부터 층(깊이) 심층학습으로 핵심적인 특징을 새롭게 표현하는 과정 (다단계 처리학습)

목표 : 가중치의 정확한 값을 찾는 것 -> 가중치 -> 층(데이터변환) - > 예측

입력

손실함수(목적함수) -> 테스트와 예측값간의 차이를 점수로 표현   
이렇게 얻은 손실 점수를 옵티마이저 해서 가중치를 업데이트 -> 성능향상  
이러한 과정을 트레이닝 루프(Training Loop)

ML 분류

* 지도 / 비지도 / 준지도
* 강화

ML 사례(수식)

* X->Y , 함수개념
* 지도 / 비지도학습 개념

ML 사례(코드)

* Keras 설치 및 실행
* 코드리뷰(딥러닝 예제)

• 지도학습(Supervised learning)  
- x : 데이터 (미리 정답을 구축) y : 정답  
- x -> y 맵핑Function을 찾는 것  
- Classification(분류), Regression(회긔)

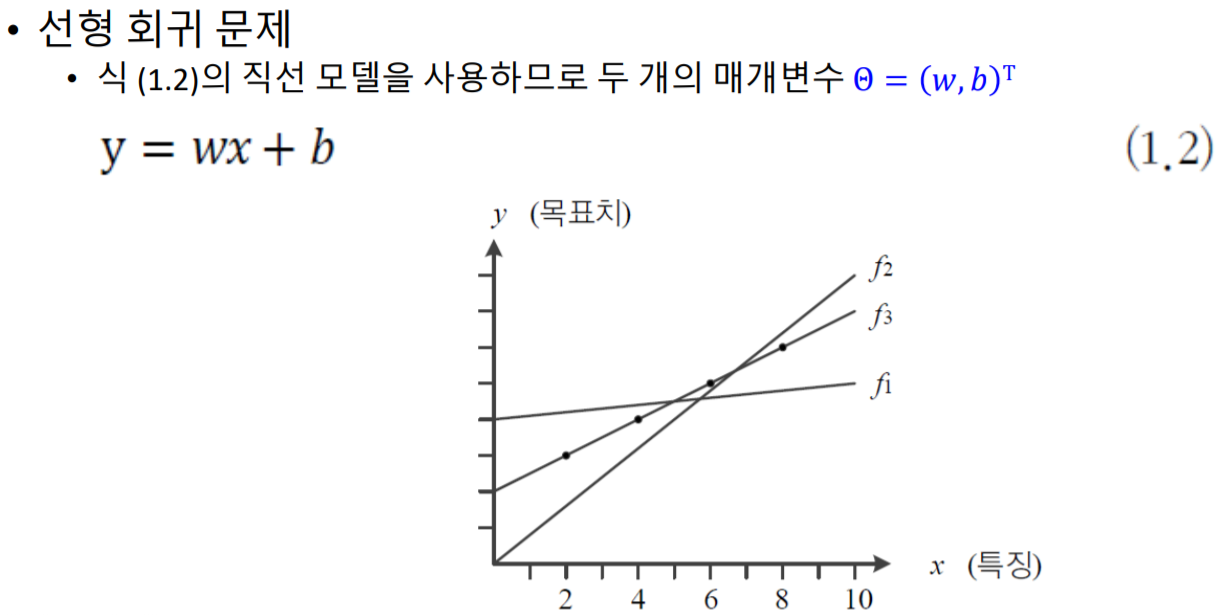
• 비지도학습(Unsupervised learning)  
- x : 데이터만 존재 (비슷한 정도만 제시, 목표값 없음)  
- 스스로 맞춤  
- 정답이 필요 없음   
- Clustering, Dimensionality reduction, Anomaly/Novelty detection

• 준지도학습(Semi-supervised learning)   
- 일부는 x와 Y를 모두 가지지만, 나머지는 X만 가진 상황  
- X의 수집은 쉽지만, Y는 수작업이 필요한 경우 유용함

• 강화학습(Reinforcement learning)  
 - 단계별 정책을 정함  
 - 묙표를 위한 단계 학습

다양한 기준에 따른 유형

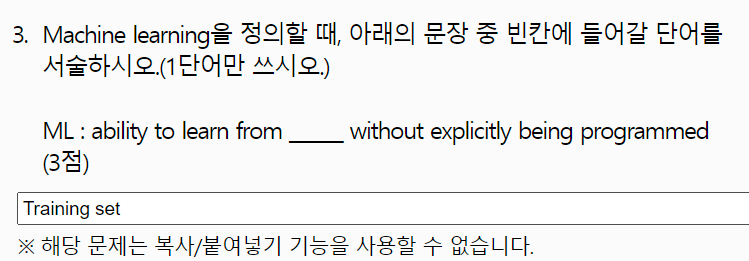
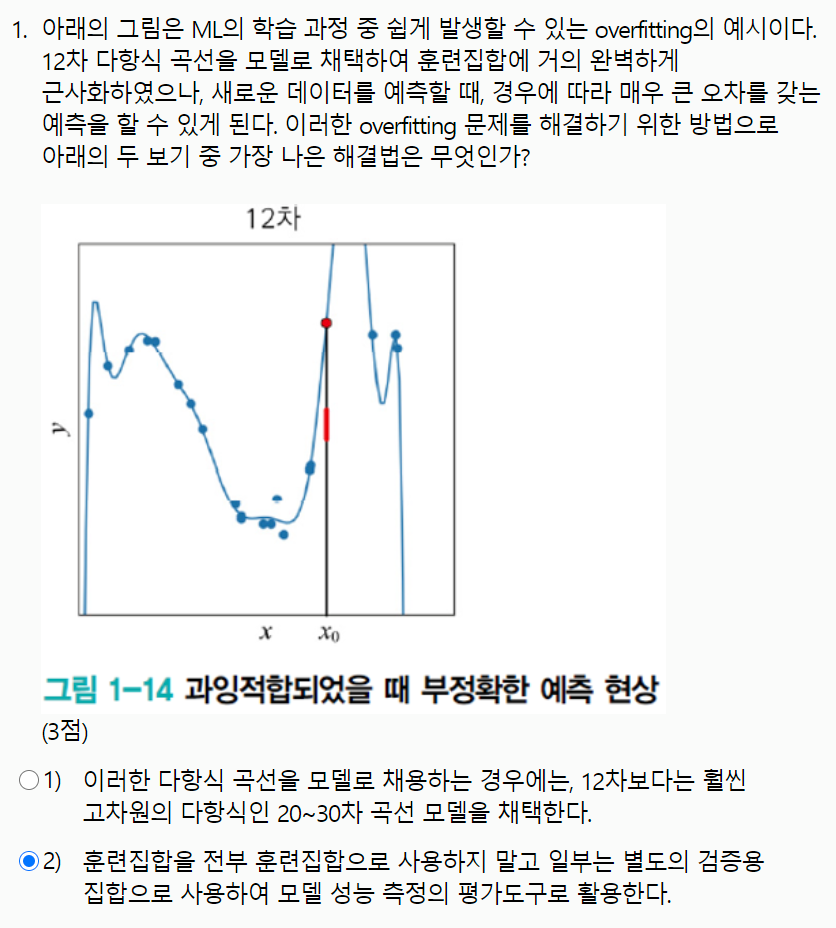
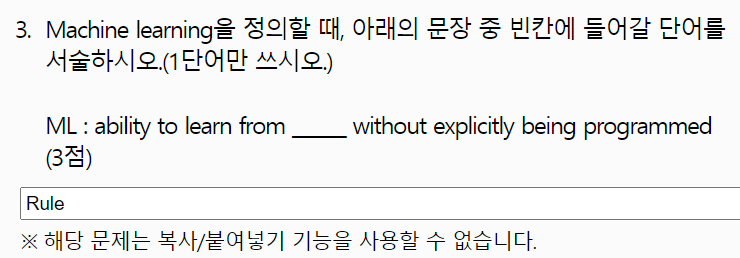
* 오프라인 학습과 온라인 학습(리얼 타임)
* 결정론적(deterministic)학습과 스토캐스틱(stochastic)학습  
  - 결정론적에서는 같은 데이터를 가지고 다시 학습하면 같은 예측기가 만들어짐  
  - 스토캐스틱 학습은 학습 과정에서 난수를 사용하므로 같은 데이터로 다시 학습하면 다른 예측기가 만들어짐.
* 분별(discriminative)모델과 생성(generative)모델  
  - 분별 모델은 분류 예측에만 관심. 즉 P(y|x)의 추정에 관심  
  - 생성 모델은 P(x) 또는 P(x|y)를 추정



기계학습에 베이스 정리  
예) Iris 데이터 분류 문제

정보 이론의 기본 원리 -> 확률이 작을수록 많은 정보

최적화 -> 주어진 데이터에서 최적의 해를 찾는 것이 (에러가 가장 낮은) 머신러닝의 핵심  
  
기계학습의 최적화  
-> 단지 훈련집합이 주어지고, 훈련집합에 따라 정해지는 목적함수의 최저점을 찾아야함  
 - 데이터로 미분하는 과정 필요 -> 오류 역전파 알고리즘  
 - 주로 스트캐스틱 경사 하강법 사용



Regression (회귀)

Y 절편인 b를 구하는 공식

B = y의 평균 –(x의 평균 x 기울기a)

Y(예측값) = ax + b

오차수정(잘못 그은 선 바로잡기)

* 일단 선을 그리고 -> 조금씩 수정해 나가기
* 언제까지?  
  - 오차가 최소가 될 때까지
* 오차?
* 선형그래프와 각 값과의 오차가 크다 <- 잘못됨

<오차>

평균 제곱 오차 (MSE)

평균 제곱 오차 = (예측값)-y(실제값))^2 의 평균

기울기 a와 오차와의 관계 : 적절한 기울기를 찾았을 때 오차가 최소화됨.

학습률:

* 어느 만큼 이동시킬지를 신중히 결정해야 하는데, 이떄) 이동 거리를 정해주는 것

경사하강법:

* 오차의 변화에 따라 이차 함수 그래프를 만들고 적절한 학습률을 설정해 미분 값이 0인 지점을 구하는 것 (Global minimum)
* Local minimum 으로 인해 기울기가 0이라서 오판할 때 있음.
* 매우 긴 구간 정체된 경우(Plateau) 오판할 때 있음

이를 위한 해결법

* Batch Gradient Descent  
  한 스탭의 배치로서 트레이닝 데이터셋을 통째로 가지고 있어야 한다.   
  트레이닝 셋이 큰 경우 느리다.
* Stochastic gradient Descent  
  배치보다 빠르다.   
  Stochastic->random   
  어느 시점에서 멈출 지 필요함  
  멈춤 시점이 최적의 코스트라는 보장은 할 수 없음. -> 러닝 레이스를 high -> low (점진적으로 줄이도록) 해결책 중 하나
* Mini-batch Gradient Descent  
  풀 배치를 계산하는 것(Batch)도 아니고 하나의 샘풀에 대해서만 계산하는 것(Stochastic)은 아니고 작은 단위의 배치로 계산  
  하드웨어가 받쳐주면 미니배치가 병렬작업으로 인해 더욱 빠르다.
* 1. 과대적합(Overfitting) :어려운 가짜 패턴을 포착해 정확한 예측을 할 수 없게 됨.
* 2. 과소적합(Underfitting) : 패턴을 포착하지 못해서 정확한 예측을 할 수 없게 됨.

https://blog.naver.com/hirit808/221643136657

일반화가 어렵다 -> 오버피팅’

희소한 모델(노이즈)/큰트렌드🡨 라소이용

Early stopping -> 트레이닝하다가 validation error가 최소화될 때 닥 스탑

규제화 정규화에 특화

Logistic Regression

50%확률로 넘나 안넘나 분류를 하는 것!(binary classfier)

Decision Tree는 작은 데이터셋 변화에 민감하다.

**SVM (Support Vector Machine)**

작고, 복작한 데이터 셋이라도 적절하게 분류 해준다.

Support vector : 두 그룹을 분리하는 거리(마진) 가장 길 때 데이터 샘플

가장 최적의 디지젼 바인더리를 optimal hyperplane (결졍 결계를 정하는 선현 함수 -> MAX)

서포트 벡트를 찾으면 optimal hyperplane을 찾을 수 있다.

특성 / 핵심

마진을 이용한 일반화 능력 향상 가능

Linear SVM

최대한 마진을 길게 해주는 것(wideset possible street) – large margin classifier

#만약 새로운 데이터를 입력했을 때 디시젼 바운더리에 영향을 줄까?  
 SVM은 다른 새로운 인스턴스가 추가되도 바뀌지 않는다.

Wx+b -> 기울기가 정해지면 b는 자동으로 정해짐

따라서 SVM목표는  
-> Margin이 가장 큰 hyperplane의 방향 w(기울기)를 찾는 것 (hard margin classification)

문제

1. 범위를 벗어나는 outlier에 민감

해결방법

1. 이를 위해 융통성있는 모델이 필요. 마진의 예외를 둘 필요가 있음 (soft margin classification). 단 일반화 능력에서 기존 하드 보다 떨어짐

Not linearly separable!

X2라는 Feature를추가 -> 1차함수에서 2차함수로 (데이터는 동일함)

앱실론은 넓이를 조절하는 파라미터이지만 새로운 데이터에 민감하지 않다(예측자체는 큰영향이 없다.)

변환함수(transformation) : 다른 차원으로 이동  
2차원 -> 2차원도 분리가능한 2차 공간으로 변환 가능

Kernel Trick의 필요성

* Kernel Trick: 원래 특정 공간 L(람다)에 정의된 두 특징 벡터 x와z에 대해 내적값인 변환함수가 존재하면k를 정의할 수 있고 이를 커널함수라 부른다.

1. 원 특정공간이 매우 고차원인 경우

Kernelized SVM

* linear SVM classifier를 풀기 위해 W 값을 최소화

Unsupervised Learning

* 세 가지 일반 과업

1. 군집화 : 유사한 샘플을 모아 같은 그룹으로 묶는 일
2. 밀도 추정(Density Est) : 데이터로부터 확률분포를 추정하는 일
3. 공간 변환(Transform) : 원래 특정 공간을 저차원 또는 고차워 공간으로 변환하는 일

표현학습

각레벨(다차원)의 특징을 쉽게 추출

-> Cursse of Dimensionality

데이터 차원이 너무 고차원일 때 극도로 문제를 해결하는게 어려워 질 수 있다.

고차원으로 갈수록 관련성이 떨어 질 수 도 있음(희소해짐) -> 데이터분석이 복잡해짐

-> Cursse of Dimensionality 해결법

모두 공간을 꽉채우기 위해서는 너무 많은 데이터가 필요됨  
그래서 차라서 차원을 줄인다.

Curse of Dimesionality 방법 2가지

1. Projection  
   언제나 좋은 것은 아님 -> 오히려 더 겹쳐는 경우(swiss roll 형태)
2. Manifold Learning

d보다 큰 n 차원에서의 일부로 맵핑하는 것. Locally resemble(지역적으로 닮은 모양)이 되는 d차원의 hyperplane을 만드는 것

* Manifold 한다는 것은 traninig instances가 그 manifold에 적절하게 맵핑 될때

차원 축소 알고리즘 : PCA (projection or manifold) : Principal Component Analysis

1. 가장 가까운 것에 맵핑 할 수 있는 hyperplane 을 찾는 것
2. 그 데이터를 방금 찾은 데이터에 project 하는 것

먼저 amount of variance가 최소(쵀대)가 되는 축을 찾는 것 -C1

그 상태에서 C1축과 직교하는 축을 찾는다.(남은 분산값이 최대가 되는) - C2

얼마나 많은 PC를 찾아야하는 가? 약 95% 차원

Nonlinear 차원축소(LLE : Locally Linear Embedding)

LLE는 다른 차원 축소기법으로써 manifold 기법에 기반한 알고리즘

단순한 projecting 이 아닌 지역적으로 이웃(거리가 가까운)것의 거리를 유지 -> 각 데이터의 연관성을 살릴 수 있는

t-SNE

비슷한 것은 가깝게 비슷하지 않은 것은 멀리 맵핑 하는 것  
시각화에서 효과적

Q&A

평균 Gradiend Descent(Batch) < 절충안(mini-batch) < 한 샘플당 (stochastic)

PCA

전처리 단계 : 데이터를 원점 중심으로 옮기는 전처리를 먼저 수행

가정 : 데이터들이 원점을 중심으로 분포되었다고 추정하는 변환 공간이 있다고 가정한다.

PCA 목적

* 손실을 최소화하면저 저차원으로 변환
* 주성분 분석은 변환된 훈련집합의 분산이 클수록 정보 손실이 적다고 판단

<주성분 분석의 학습 알고리즘>

공분산 행렬

고윳값과 고유 벡터를 구한다.

PCA최적화 문제  
비선형에서 선형벡터로 피팅하기 어려움 -> Kernel PCA

<Kernel PCA : 메모리 기반>

커널함수는 써서 새로운 샘플 데이터가 와도 Afit 을 구할 수 있다.

kPCA 값을 찾는 함수 GridSearchCV

Clustering  
: 클러스터의 묶어서 할당 (오브젝트의 집합)

Similarity and dissimilarity

유사도를 바탕으로 분류~

K-Means 분류(Clustering)