**AI504 3강 정리**

* 머신러닝: 인공지능에 포함된 개념, 특정한 일을 수행하기 위해 데이터를 사용해 통계적인 방법으로 기계를 학습시킴
* 딥러닝: 머신러닝에 포함된 개념, 일반적인 머신러닝보다 더 복잡한 일을 수행하기 위해 신경망과 빅데이터로 기계를 학습시킴
* 딥러닝을 사용하는 이유: Less Feature Engineering
* 머신러닝의 종류
  + Supervised Learning: 입력 X에서 출력 Y로의 함수를 학습
    - 예시: 이미지 분류, 번역, 이미지 설명
  + Unsupervised Learning: 데이터 X를 distribution/manifold하는 함수를 학습 (Y 없음)
    - 예시: 클러스터링(distribution only), 저차원 행렬 분해(distribution & manifold)
  + 위 둘 사이에서 생성모델과 self-supervised learning의 분류가 애매함
  + Reinforcement Learning: 환경 E와 행동 A가 주어졌을 때 long term reward R을 최대화 하는 함수를 학습
    - 바둑, 아타리(게임), 자율주행차
  + 이번 과목에서 할 것들은 전부 SL, UL임. RL은 없음.
* SL, UL, RL 모두 모델을 학습시켜야 함
  + 학습을 시킬 때는 목표가 필요함 -> 목표 함수
  + 예시: 개 vs 개 아님
    - 목표함수: 사진이 개인지 아닌지 구분
    - => loss 함수(true class와 predicted class의 차이)를 최소화 해야함
* Loss 함수가 가장 작아지도록 하는 것 = 최적화(Optimization)
  + => 함수의 최솟값 찾기 = 미분값이 0인 곳 찾기
  + 극값들을 구하게 되는데 그 중 가장 작은 것이 최솟값
  + 최솟값을 찾는 분석적 해법은 없음
  + 수치적 방법
    - 순서대로 값을 확인하여 만족할 수 있는 값 찾기
    - 경사하강법(Gradient Descent): 현재 값에서 현재 값의 기울기(미분값)에 비례한 정도의 양을 미분값이 작아지는 방향으로 이동한 값을 다음 값으로 선택함
    - SGD(Stochastic Gradient Descent): 경사하강법과 유사하지만 전체 데이터 대상이 아닌 일부의 데이터(예: 미니배치)만을 사용함
      * 사용하는 이유: 전체 데이터가 너무 큼. 극솟값을 피할 수도 있음.
* 언제 loss 함수가 충분히 작아졌다고 판단할 수 있는가? -> Evaluation
  + Evaluation에 Loss 함수의 값을 사용하는 대신 정확도와 같이 다른 평가지표 사용
* Train & Validation & Test
  + Train: 학습할 때 사용함
  + Validation: Evaluation에 사용하여 언제 학습을 그만둘지 결정함
  + Test: 학습과정에서 모델이 보지 못함. 최종 모델을 평가할 때 사용함
* N-fold Cross Validation
  + 데이터를 N개의 그룹으로 나누고 하나의 그룹을 Test Data로 나머지를 Train Data로 사용함
  + Test Data가 너무 쉬워서 평가가 잘못되는 것을 방지
  + 최근에는 잘 사용하지 않음
    - 데이터가 너무 많아서 굳이 할 필요가 없고 시간이 너무 오래 걸림
* 과적합 & 과소적합 (Overfitting & Underfitting)
  + 과소적합: 차수가 너무 낮아서 데이터에 안 맞음
  + 과적합: Train Data에 너무 맞추려고 하여(차수 높아짐) 보지 못한 실제 데이터에 맞지 않음
    - Train Data에서 아주 좋은 결과를 냈다고 해도 그것이 Test Data에서도 좋은 결과를 보일 것이라고 생각할 수 없다
  + 차원의 저주 (Curse of Dimensionality)
    - 과소적합일 경우에는 변수(Feature)를 추가함
    - 이 과정에서 불필요한 변수나 정보가 모델에 더해질 수 있음
* 정규화 (Regularization)
  + 모델이 튀는 것을 막아 과적합을 방지함
* 분류기(Classifier)
  + 로지스틱 회귀 (Logistic Regression): 확률에 log를 씌워 logit으로 변환
  + SVM (Support Vector Machine): 두 개의 분류 사이의 마진(가장 가까운 샘플들 사이의 거리)를 최대화
  + 결정 트리 (Decision Tree): 변수(Feature)를 토대로 트리를 만듦
* Ensembles
  + 여러 개의 분류기를 같이 사용하여 결과를 더 좋게 만듦
  + Bagging: 데이터/Feature의 Subset마다 다른 분류기를 사용하여 학습함
  + Boosting: k+1번째의 분류기가 k번째 분류기의 에러를 고침
* Clustering
  + K-means
    - Centroid: 클러스터의 중앙
    - 가장 가까운 centroid를 가진 클러스터로 각각의 샘플을 넣음
    - 새롭게 바뀐 클러스터의 centroid를 업데이트함
    - 위를 반복
  + Mixture of Gaussian
    - K-means의 일반화된 모델
    - 각 샘플은 각각의 클러스터에 속할 확률을 가짐
    - K-means처럼 각 클러스터의 평균(μ)과 표준편차(σ)를 업데이트 한 후 각 샘플의 확률을 업데이트 하고 이 과정을 반복함