

DATA MINING 101

Somewhere in between Statistics and AI

Contents

1. What is Data Mining?

A. 정의 및 접근법

B. 프로세스 이해하기: CRISP-DM

2. What is Modeling?

A. 정의

B. 지도학습 (Supervised Learning)

1) 편향-분산 트레이드오프 (Bias-Variance Tradeoff)

3. How to Avoid Overfitting?

A. 교차 검증 (Cross Validation)

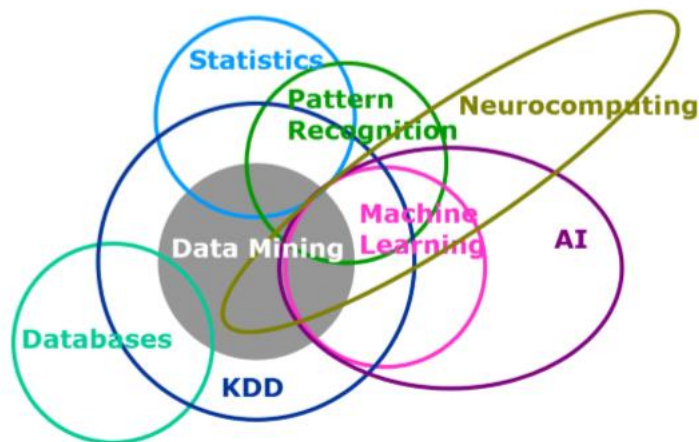
B. 차원의 저주 (Curse of Dimensionality)

1. What is Data Mining?

A. 정의 및 접근법

‘인공지능’, ‘빅데이터’와 같은 키워드가 대두됨에 따라 관련 기술을 배우고자 하는 수요 역시 증가하는 추세이다. 빠르게 변화하는 시대의 사회 흐름을 읽고자 하는 노력은 중요한데, 보다 본질적으로 이러한 기술들을 이해하고 활용하기 위해서는 이들 기술의 **주요한 개념을 이해하는 것은 물론이고, 이들이 왜 등장하게 되었는지에 대해 살펴봐야** 한다. 즉, 이미 구현된 기능/기술들을 단순히 자신의 상황에 끼워 맞춰 사용하는 관습은 지양하고, 통계학적인 관점에서 이들이 어떠한 의미를 지니는지 고민해보는 노력이 필히 수반되어야 하는 것이다.

이처럼 이론적으로 탄탄한 베이스를 구축하는 것과 더불어 우리는 데이터 마이닝 관련 기술들을 어떻게 실제 상황에 적용할 수 있는지 적극적으로 고민해보아야 한다. **이는 데이터 마이닝이 여러 학문의 교집합에 자리하고 있다는 점을 인지하는 것에서 출발한다.** 아래 그림 ([그림 1])에서 볼 수 있듯 데이터 마이닝은 인공지능 (Artificial Intelligence), 패턴 인식 (Pattern Recognition), 통계학, 데이터베이스와 같은 학문들의 개념을 포함한다.



[그림 1]

머신 러닝 (Machine Learning, 기계 학습)과 딥러닝 (Deep Learning)은 AI 분야 주요 개념들로, 인간의 학습 과정을 모방하고자 하는 노력에서 그 뿌리를 찾을 수 있다. '약한 인공지능 (Weak AI)', '강한 인공지능 (Strong AI, 또는 Artificial General Intelligence)'와 같은 용어들이 이러한 맥락에서 사용된다 ([그림 2] 참고). 일반적인 기계학습 문제 (a well-posed learning problem)는 다음과 같이 정의될 수 있다:

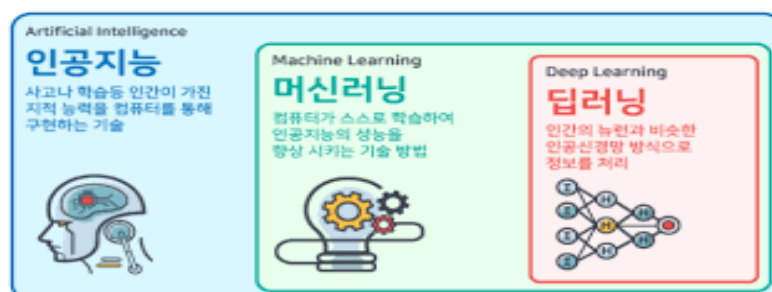
컴퓨터가 경험 (E, Experience), 즉 **데이터를 주 재료로 하여 학습**의 과정 (learn, 또는 **train**)을 거쳐 특정한 기능 (T, Task)을 달성한다. 성능 평가 지표 (P, Performance measure)를 통해 주어진 기능을 얼마나 잘 수행하는지 정량적으로 검토할 수 있으며, 이들 지표 P는 데이터 E를 통해 개선될 수 있다.

여기서 '학습'이라는 단어가 중요한데, 기계학습은 일반적인 컴퓨팅 과정과는 달리 사람이 직접 규칙 또는 패턴을 수동적으로 입력하지 않고 컴퓨터가 그 과정을 수행한다는 점에 있어 유의미하다고 할 수 있다. 모델에 데이터를 넣어주고, 수행하고자 하는 대략적인 범주의 task (분류, 예측 등)만 입력해주면 컴퓨터가 알아서 뽀로롱~하고 이를 수행해준다.

딥러닝 (Deep Learning)은 인공지능 분야에서 가장 작은 부분집합을 차지하는데, 인공 신경망을 구축하여 업무를 수행하는 모델을 의미한다. (더 자세한 내용은 P-SAT 딥러닝팀의 교안을 살펴보자!)

약한 인공지능(Weak AI)	강한 인공지능(Strong AI)
특정 문제의 해결	사람처럼 사고
지능이 있는 것처럼 프로그래밍되는 것	지능을 가질 수 있도록 프로그래밍 되는것
인간 두뇌의 특정한 일부 기능 모사하여 특정 목적에 유용한 제한된 지능	인간두뇌의 대체가능한 수준으로 다목적 과제 수행 가능한 범용적 지능
데이터 패턴의 해독	빅데이터 기반의 분석 및 자체적 딥러닝
프로그래밍 기반의 로봇 제어	인간과의 게임(바둑 등 수준) 수행

[그림 2]

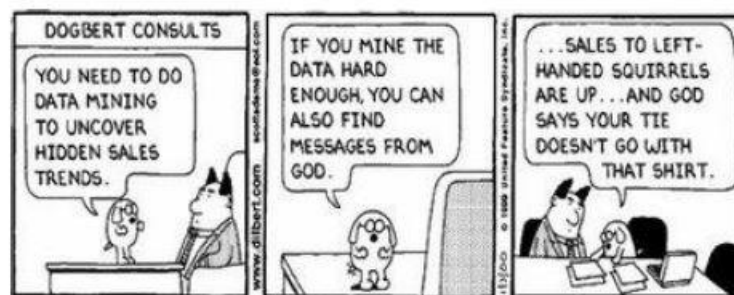


[그림 3]

패턴 인식 (Pattern Recognition)에 대해서도 가볍게 살펴보자. 자세한 내용은 3주차 때 다룰 것이니 지금은 그냥 속~! 패턴 인식, 또는 패턴 발견 (pattern discovery)는 방대한 양의 데이터가 저장되어 있는 데이터베이스에서 유용한 규칙을 찾는 것을 의미한다. '이마트에서 우유를 구입하는 사람들이 자주 같이 사는 상품은 무엇일까?' 와 같은 질문이 바로 패턴 인식, 연관 법칙 (rule association) 발견 과정에 포함된다.

앞서 가볍게 언급한 분류, 예측과 같은 task들을 수행한다는 점에 있어서 데이터 마이닝과 AI는 공통점을 보이지만 분석의 목적에서 그 차이점을 찾을 수 있다. AI는 이러한 task들을 '수행 (implement)'하는 데에 중점을 둔다면 데이터 마이닝은 이에 더하여 주요한 '인사이트를 채굴 (mine)'하는 것을 목표로 한다. 이러한 목표를 달성하기 위한 수단으로서, AI 기술을 활용한다고 이해하면 쉽겠다. 뻔히 알고 있는 단순한 사실을 열거하는 것이 아니라 인사이트를 주는 정보라 볼릴 수 있는, 의미 있는 패턴을 알아내는 방법으로는 무엇이 있을까?

본페로니 법칙 (Bonferroni's Principle)은 방대한 데이터로부터 유용한 패턴을 찾아내고자 하는 집착이 과할 때 일어날 수 있는 위험한 상황에 대해 서술한다. 만약 데이터에서 발견되는 관심 개체수가 우리가 기대하는 (expected) 무작위 통계 현상에 의해 관측된 개체 수보다 훨씬 많다면, 이 개체들이 유의미한 통계 현상일 확률은 굉장히 낮다.

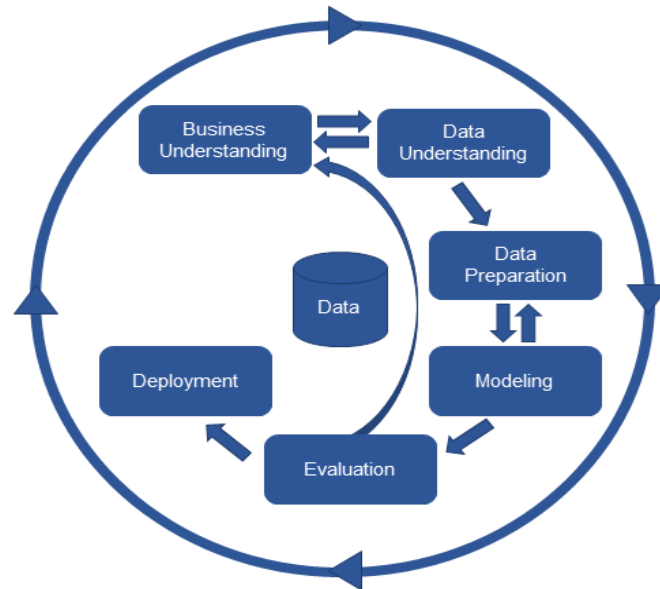


[그림 4]

우리가 시행하고자 하는 분석이 유의한 시도인지에 대해 알아보기 위해서는 통계학적인 관점에서 추정 (estimation) 및 검정 (testing) 과정을 거쳐야 하는데, 이와 같이 데이터 자체의 형태, 또는 분포에 의거하여 통계적 현상을 설명하는 방법론이 궁금하다면 P-SAT의 다른 팀 교안들을 참고해보자. 데이터 마이닝 과정이 '지식의 발굴'을 목표로 하며, 타 근접 학문들의 개념을 모두 포괄한다는 점을 인지한 상태에서 우리가 배울 다채로운 방법론들을 하나하나 익혀보자.

B. 프로세스 이해하기: CRISP-DM

‘데이터 마이닝 프로세스’의 정석이라 꼽히는 프레임워크, CRISP-DM을 통해 전반적인 프로세스를 이해해보자. SAS사에서 개발한 분석 프로세스인 SEMMA는 통계학적인 관점에서 서술된 반면, **CRISP-DM은 보다 비즈니스 관점에서 작성**되었기 때문에 실무에서 그 활용도가 더 높을 것이라 생각된다.



[그림 5]

CRISP-DM은 Cross-Industry Standard Process for Data Mining의 약자로, 이름에서부터 볼 수 있듯이 적용하고자 하는 분야가 무엇이든지 간에 상관없이 통용되는, 그야말로 ‘스탠다드’한 프레임워크이다. 예시와 함께 각 단계의 내용을 파악해보자.

1. Business Understanding – 비즈니스 문제 이해

CRISP-DM은 비즈니스 관점에서 작성되었기 때문에 가장 먼저 비즈니스를 운영하는 데 있어 기존의 데이터를 활용해 어떠한 시도를 새롭게 해볼 수 있을지에 대해 고민하는 단계에서 출발할 수 있다. IBM이 해당 프레임워크에 대해 출간한 공식 문서를 참고하면, ‘현재의 비즈니스 상황과 관련한 배경 지식을 쌓는 것’ (‘Start gathering background information about the current business situation’)이 첫번째 task로 등장한다. 이와 맥락을 같이 하는 내용으로 ‘데이터 마이닝 과정의 성공 여부를 정의 내릴 수 있는 기준을 세우는 것’ (‘Agree upon criteria used to determine data mining success from a business perspective’) 역시 주요 task로 기술되어 있다.

예를 들어 스타벅스에서 새롭게 출시한 음료를 홍보하기 위해 이 음료를 기존의 케익류들과 함께 세트에 묶어 판매하는 전략을 실시한다고 해보자. '세트로 제품을 팔았을 때 단품으로 팔았을 때보다 몇 퍼센트포인트 이상의 매출 증대가 관찰되었을 때 "성공적인 전략"이라 할 수 있을지', '케익 중에서도 어떠한 종류의 케익과 판매되었을 때 더욱 눈에 띄는 매출 증대를 관찰할 수 있는지' 와 같은 물음들에 대해 조직은 확실한 기준을 세워야 하며, 이에 대해 구성원들 간 합의를 이끌어내야 한다.

P-SAT 내에서의 주제분석, 그리고 공모전 출품을 위한 데이터 분석 과정의 관점에서도 이해해 봐야 한다 (왜냐하면 우리가 진짜 해볼 테니까). 이 단계가 가장 많은 창의력을 요구하곤 하는데, 단순히 특정 task를 수행하는 모델의 예측력/정확도를 높이는 것이 목표가 아니라 구성원들이 각자 관심 있어 하는 데이터를 선정하여 이들 데이터로부터 유의미한 인사이트를 발굴하는 것이 목표라면, 문제 상황을 명확히 정의하는 단계에서부터 애를 먹을 확률이 매우 높다... 그렇기 때문에, 흩어져 있는 데이터들을 한 군데로 모아 큰 그림부터 그려보고 작은 크기의, 실행 가능한 task들로 문제 상황을 쪼개 세부적으로 정의하는 것이 무엇보다 중요하다.

2. Data Understanding – 데이터 이해

흔히 EDA (Exploratory Data Analysis)라 불리는 작업들을 수행하는 단계이다. 이름에서도 볼 수 있듯 데이터에 대해 '탐색적인' 분석을 수행하는 것인데, 시각화를 통해 데이터에 대한 직관적인 이해를 달성하는 것이 목표라 할 수 있다. 주어진 데이터의 변수들이 지닌 의미, 변수 간의 관계를 파악하는 단계이다. 파이썬에서 `describe()` 함수와 같은 명령들을 통해 변수들이 분포한 값의 범위를 파악하거나, `median`, `min/max`와 같은 내장 함수를 사용, 또는 `boxplot`을 통해 이상치를 파악 (outlier detection), 결측치의 유무를 파악하는 과정이 모두 포함된다.

3. Data Preparation – 데이터 준비

바로 그 악명 높은 데이터 전처리의 과정이다. 웹 크롤링 (web crawling)을 통해 직접 데이터를 수집했거나, 데이터 간 기술된 형식이 일정하지 않을 때 데이터 전처리가 필수적으로 이루어져야 한다. 수동적이고 반복적인 과정처럼 여겨질 수 있지만, 모델의 성능을 개선하는 데에는 데이터의 질 (quality)도 주요한 역할을 차지하므로 가능하다면, 도메인 지식을 염두에 두고 작업을 수행하는 것이 좋다.

4. Analysis & Data Modeling: 데이터 분석과 모델링

앞서 언급한 머신러닝/딥러닝 기법들을 적용하는 단계이다. 추천, 예측, 해석 등 어떠한 분석을 실시하고 싶은지에 따라 각기 다른 방향으로 전개된다. 분석자가 어느 정도 수동으로 처리해줄 부분이 있기도 하지만 파이썬의 경우 Scikit-Learn (sklearn, 사이킷런) 라이브러리에 많은 분류기들이 폭넓게 구현되어 있으니 상황에 맞춰 적절히 사용하면 된다.

5. Evaluation: 분석 모델의 평가

4단계에서 어떤 분석을 수행했는지에 따라 ‘모델링이 잘 되었는지’ 평가하는 방법이 여러 갈래로 나뉘는데, 만약 예측해야 할 라벨 (target variable, 또는 y)이 범주형 (categorical) 데이터인 경우 misclassification rate을, 라벨이 연속형 (continuous) 데이터인 경우 RMSE를 지표로 사용할 수 있다.

6. Deployment: 분석 결과의 적용

데이터 마이닝 과정을 통해 산출된 결과를 서비스, 제품 개발 등 실제 비즈니스 상황에 적용해보는 것을 의미한다. P-SAT 주제분석 또는 기타 공모전에 참여한다면 유의미한 결론을 적절히 정리하여 문서화/발표하는 것을 포함할 수 있겠다.

2. What is Modeling?

A. 정의

데이터 마이닝 과정에서 모델링은 어떠한 역할을 차지하는가? 분석의 대상이 되는, 다시 말해 우리 분석의 주 재료가 되는 원천은 그 양이 매우 방대한데 (‘빅데이터’), 그렇기 때문에 이들 데이터로부터 수동적으로 정보를 발굴해내기란 불가능에 근접한다. 이러한 문제상황을 짧은 시간 내에 효과적으로 해결하기 위해서는 컴퓨터의 자동화 기술에 기대는 것이 현명하다. 이 때, 우리는 머신러닝의 모델링 개념을 도입한다. 컴퓨터가 학습 (train) 할 수 있는 데이터 (experience)를 넣어주고 수행하고자 하는 task를 적절히 명시해주었을 때, **컴퓨터가 알아서 규칙을 발견하고 데이터의 경향성을 설명해주었으면 하는 것이 우리의 목표**이기 때문이다.

다음의 예시를 함께 보자.

Bedrooms	Sq. feet	Neighborhood	Sale price
3	2000	Normaltown	\$250,000
2	800	Hipsterton	\$300,000
2	850	Normaltown	\$150,000
1	550	Normaltown	\$78,000
4	2000	Skid Row	\$150,000

[training data]

Bedrooms	Sq. feet	Neighborhood	Sale price
3	2000	Hipsterton	???

[test data]

[그림 6]

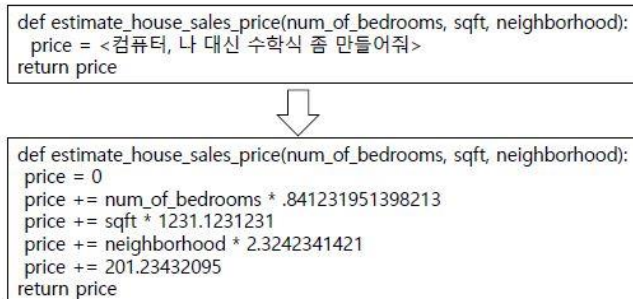
좌측의 표는 'training data'라 명시되어 있고, 우측의 표는 'test data'라 적혀 있다. 좌측의 주어진, 이미 답을 알고 있는 데이터를 바탕으로 우측의 새로운, 답을 구하려고 하는 문제가 주어졌을 때 이 문제의 답을 찾으려 한다. 우리는 특정 주택 내 방의 개수, 면적, 주택이 속한 동네에 대한 정보를 이미 알고 있고, 이러한 정보들이 주어졌을 때 이들 주택의 가격이 어떻게 책정되는지가 궁금하다. 부동산 일을 오래 해온 전문가라면 어떠한 요소들이 갖춰졌을 때 주택이 잘 팔리는지, 또 어떤 범위 내에서 가격이 책정되는지에 대해 알고 있을 것이다. 이들 전문가의 도움을 받아 새로운 주택의 가격을 알고자 한다면 다음과 같이 코드를 작성할 수 있다.

```
def estimate_house_sales_price(num_of_bedrooms, sqft, neighborhood):
    price = 0
    price_per_sqft = 200
    if neighborhood == "hipsterton":
        price_per_sqft = 400
    elif neighborhood == "skid row":
        price_per_sqft = 100

    price = price_per_sqft * sqft
    if num_of_bedrooms == 0:
        price = price - 20000
    else:
        price = price + (num_of_bedrooms * 1000)
    return price
```

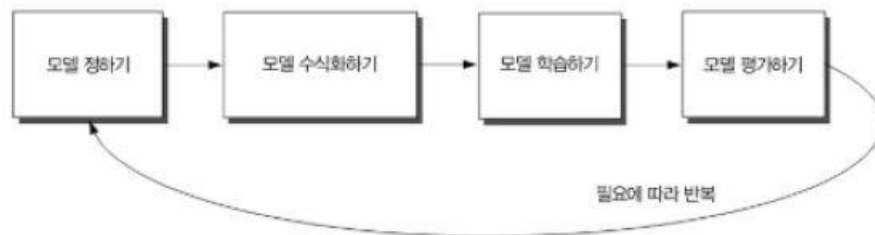
[그림 7]

if문을 통해 조건을 제약하고, 이들 조건을 만족했을 때 특정한 값을 지니도록 코드를 설계할 수 있다. 하지만 데이터의 양이 어마어마하게 많다면? 거기에 더해, 이들 데이터가 분포해 있는 범위가 매우 넓어 위와 같이 조건문을 작성할 수 없다면? 이 경우 우리는 컴퓨터의 계산 능력에 기대는 것이 최선이다. 머신 러닝은 다음과 같은 방식의 워크플로우를 따른다.



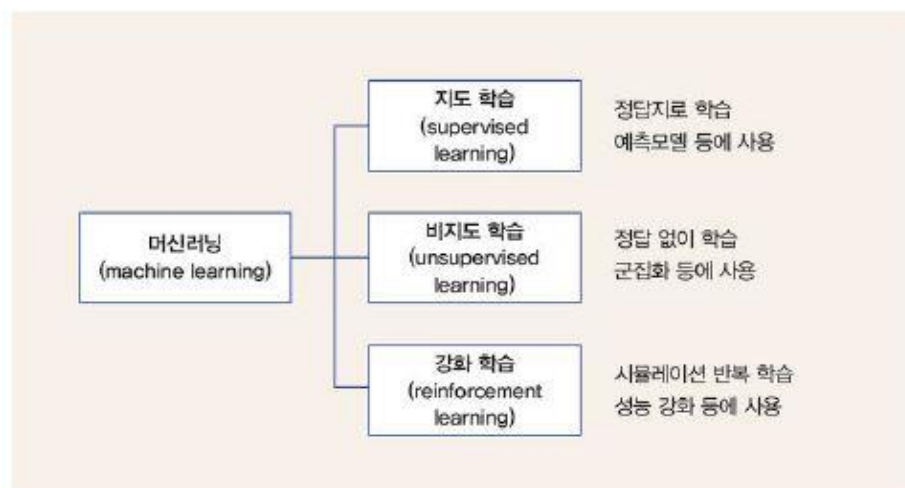
[그림 8]

'컴퓨터, 나 대신 수학적 좀 만들어줘' 라는 구문에서 볼 수 있듯 수학적식을 만들고, 패턴을 찾는 것은 결국 컴퓨터가 해내는 일인 것이다. 머신 러닝 이론에서는 '수학적식', 또는 이들 수학적식의 모음을 모델이라 일컫는다. 인공지능 분야에서 컴퓨터가 학습하는 과정은 인간의 학습 과정을 모방한다고 했다 (아니, 적어도 인간의 학습 과정과 최대한 가까워지려 한다). 그렇기 때문에 인간인 우리도 새로운 내용을 학습할 때 시행 착오를 겪듯, 기계 역시 끊임없는 시행 착오를 겪는다. 첫 시도에 '짜잔~!'하고 컴퓨터가 내놓은 모델이 반드시 가장 좋은 성능을 내는 최적의 모델은 아니다. 다양한 경우의 수를 포괄하는 우수한 질의 데이터를 학습 과정에서 활용하거나, 모델의 하이퍼 파라미터 (hyperparameter)를 조정하는 등의 과정이 필히 동원되곤 한다. [그림 9]가 이러한 머신 러닝 모델링의 과정을 한눈에 보여준다.



[그림 9]

머신 러닝에서 학습의 목적은 크게 예측 (prediction)과 추론 (inference), 그리고 데이터의 경향성을 파악하는 묘사 (description)로 나눌 수 있는데, 이를 기준 삼아 한 번 더 학습의 종류를 나누면 지도학습 (supervised learning)과 비지도학습 (unsupervised learning)으로 구분할 수 있다. 지도 학습은 쉽게 말해 데이터마다 라벨, 즉 y 값이 달려 있어서 분류/예측 문제를 수행한 후 답을 확인해볼 수 있는 과정이다. 위의 부동산 가격 예측 문제가 여기에 속한다. 비지도학습은 이와는 다르게 각 데이터마다 라벨이 달려 있지 않아 높은 정확도의 답을 맞히는 것과 같은 task를 수행하는 것이 아니라, 데이터의 구조를 묘사하고 해석하는 데 더 초점을 두게 된다. 그렇기 때문에 분석자의 주관이 필히 개입된다. 알파고와 같은 알고리즘은 강화학습 (reinforcement learning)의 산물로, 본 교안에서는 다루지 않는다. 비지도학습과 관련한 내용은 3주차에 본격적으로 다룰 예정이다. 머신 러닝 알고리즘 대부분이 지도학습과 관련되어 있을 뿐만 아니라, 데이터 분석 과정 전반을 이해하는 데에 지도학습의 프레임워크를 따르므로 본 교안에서는 우선 지도학습을 자세히 설명하고자 한다. 머신 러닝 알고리즘의 큰 갈래 ([그림 10])는 아래 첨부되어 있다.



[그림 10]

B. 지도학습

지도학습은 예측과 추론 작업을 수행하는데 동원된다. '무엇을 바탕으로 무엇을 예측/추론하고자 하는가?'와 같은 물음이 자연스럽게 떠오를 것이다. 여기서 잠깐 핵심 용어들을 짚고 넘어가자. 위 물음의 첫번째 '무엇'은 이미 주어진 데이터들의 값이다. 주택 가격 예측 예시에서 해당 주택이 보유하고 있는 방의 개수, 속한 동네, 면적과 같은 요소들 각각의 값이다. 두 번째 '무엇'은 test data에서 새로운 주택이 얼마에 팔릴 지, 바로 그 가격이다. 즉, 가격이라는 종속 변수를 결정하는 독립 변수로서 방의 개수, 속한 동네, 면적과 같은 요소들이 활용된 것이다. 중, 고등학교 때 배운 함수의 개념을 도입해본다면 첫번째 '무엇'은 x 에, 알아내고자 하는 가격인 두 번째 '무엇'은 y 에 대응된다. 이들 독립 변수는 'feature'라는 용어로도 통용되곤 하니 잊지 말자. 종속 변수 y 는 'target', 또는 'target variable'이라고도 일컬어진다.

'예측'과 '추론'이라는 단어가 잘 와 닿지 않을 수 있다. 지도학습에서 예측하고자 하는 값의 형태가 범주형이나, 연속형이냐에 따라 다시 문제가 분류와 회귀로 나뉜다. 부동산 가격 예시에서 우리는 연속형의 숫자를 그 값으로 예측하고자 하기에 회귀 문제라 가정하고 문제를 해결하였다. [그림 11]이 지금까지 설명한 내용을 깔끔하게 정리해준다.



[그림 11]

1) 편향-분산 트레이드오프 (Bias-Variance Tradeoff)

부동산 가격 예측 문제를 다시 한번 생각해보자. 우리는 여러 가지 제약 조건들을 모두 고려하여 조건문을 수동으로 작성할 수 없기 때문에 컴퓨터로 하여금 자동으로 수학적식을 만들어주도록 했다. 한편, 컴퓨터가 만들어준 수학적식은 현실의 모든 데이터를 완전히 반영하지 않는다. 아니, 전부 반영할 수 없다. 표본 조사의 개념에 기초해보자면, 우리는 여러 제약 조건 하에 실험을 설계하고 수행하기 때문에 전체 모집단에서 일부 표본을 추출하여 그에 대해서만 조사를 수행한다. 그렇기 때문에 이 수학적식의 실제 모습은 알 수 없고, 다만 우리는 주어진 상황에서 모집단의 경향성을 가장 유사하게 반영하는 수학적식을 설계할 수밖에 없다.

다음의 형식을 눈여겨보자.

when matrix X such as

$$X = \begin{pmatrix} x_{11} & \cdots & x_{1p} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{np} \end{pmatrix}.$$

is given as input, Y is defined as the following

$$Y = f(X) + \epsilon$$

where Y is the response vector, X is the set of p predictors given as columns,

and ϵ being the random error term

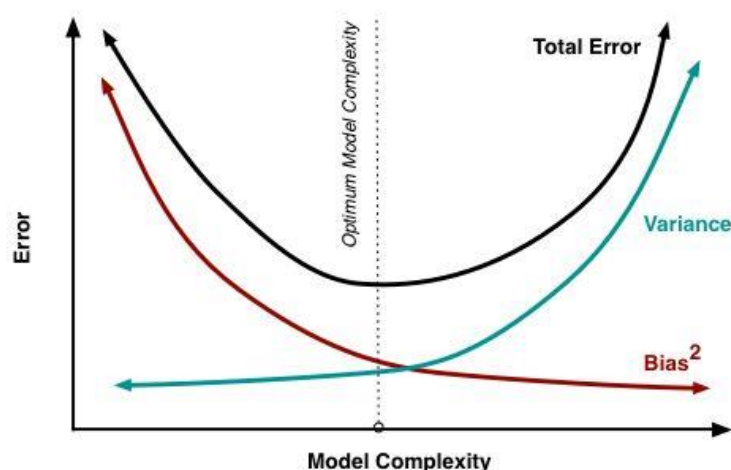
위의 형식에서 볼 수 있듯, 우리의 수학식은 f 로써 표현된다. 주의해야 할 점은 우리가 구한 이 f 는 실제 현상의 추정량 (estimator)이기 때문에 \hat{f} 라고 적어주는 것이 정확하다. 이와 같이 수학식을 만들어주고 새로운 데이터에 대해 예측 값을 산출해냈다고 가정해보자. 이 추정치 (estimate) y 는 우리가 세워준 수학식, 그러니까 모델이 얼마나 실제 상황과 유사하게 정의되었는지를 말해준다. 예측된 y 와, 실제 y 값의 절대적 오차가 작으면 작을수록 모델이 적절히 설계되었음을 말해준다.

모집단에서 표본이 추출될 때마다 모델은 조금씩 다르게 설계될 텐데, 우리는 평균적으로 우리가 구한 예측치가 실제 값과의 차이가 작길 기대한다. 회귀 문제로 제약한다면, 이러한 모델의 성능을 평가하는 지표 (performance metric) 로서 MSE (Mean Squared Error)를 사용할 수 있다. MSE 는 우리가 구한 예측치 y 와 실제 y 간의 차이의 제곱을 평균 낸 형태이다. 따라서 모델은 MSE 를 최대한 줄이는 방향으로 설계되어야 한다. 다음의 수식을 살펴보자.

$$\begin{aligned} E[(y - \hat{f})^2] &= E[y^2 + \hat{f}^2 - 2y\hat{f}] \\ &= E[y^2] + E[\hat{f}^2] - E[2y\hat{f}] \\ &= \text{Var}[y] + E[y]^2 + \text{Var}[\hat{f}] + E[\hat{f}]^2 - 2fE[\hat{f}] \\ &= \text{Var}[y] + \text{Var}[\hat{f}] + (f^2 - 2fE[\hat{f}] + E[\hat{f}]^2) \\ &= \text{Var}[y] + \text{Var}[\hat{f}] + (f - E[\hat{f}])^2 \\ &= \sigma^2 + \text{Var}[\hat{f}] + \text{Bias}[\hat{f}]^2 \\ &= \text{irreducible error} + \text{reducible error} \end{aligned}$$

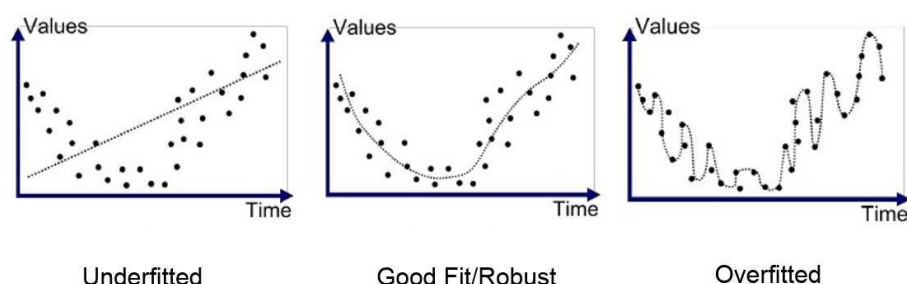
MSE 는 Bias 의 제곱, Variance, 그리고 irreducible error 인 σ^2 로 표현된다. 표본 추출로 인해 자연스럽게 발생하는 σ^2 를 차치하고 reducible error 에만 집중해보자. Bias, 즉 편향은 추정된 f 와 \hat{f} 의 차이이다. $Var(\hat{f})$, 모델의 분산은 매번 다른 표본 추출의 상황마다 각 모델이 추정되었을 때, 이 모델들이 얼마나 다양한 형태를 띠는지를 의미한다. 우리는 모델을 적절히 설계하여 reducible error 인 Bias와 Variance를 최소화하고자 한다. 하지만, 이들 조건을 동시에 우리가 원하는 수준으로 줄이기란 어려운 일이다. **하나를 줄이면 다른 하나는 그 값이 커지는 트레이드 오프 (trade-off) 상황을 필연적으로 맞이하기 때문이다.**

아래의 그림을 보자.



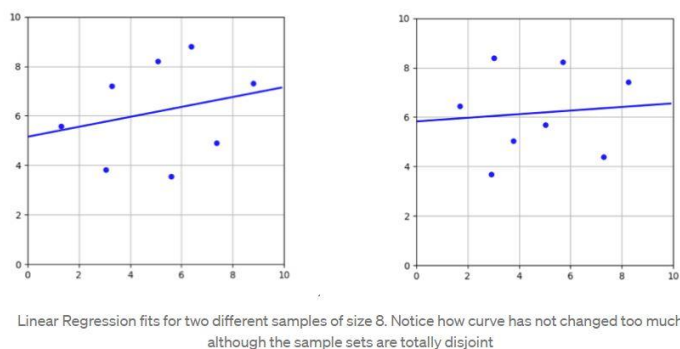
[그림 12]

여기서 모델의 복잡도 (model complexity)에 대한 이야기를 하지 않을 수 없다. 데이터 마이닝 과정에서 우리가 모델을 설계하는 근본적인 이유는 주어진 데이터에 대한 이해도를 높이고 현상 전반에 대해 추상화를 달성하고자 위함이다. 하지만 한편으로는 너무 추상적이고 원론적인 분석은 지양하고 싶을 테다 (언더피팅 지양). 우리는 주어진 데이터의 특수한 경향성을 모델을 통해 표현하고 싶은데, **바로 이러한 딜레마 상황이 편향-분산 트레이드오프의 개념과 맞닿아 있다.**



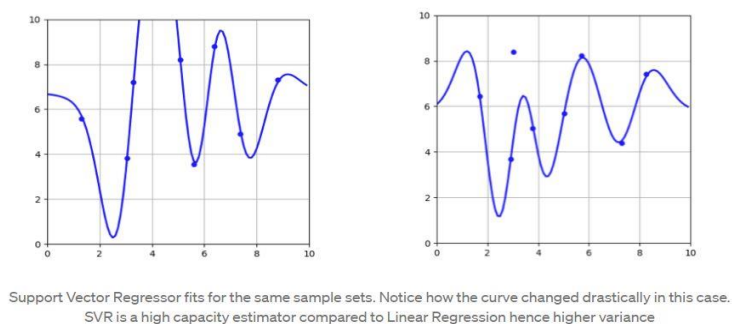
[그림 13]

단순선형회귀 (simple linear regression)와 같은 복잡하지 않은 모델은 이해하기에도, 해석하기에도 좋다. 지나치게 많은 변수를 사용하지 않는 이 모델은 데이터의 경향성 전반을 설명하는 데에 적합하다. 그렇기 때문에 각기 다른 데이터를 바탕으로 모델을 설계했을 때에도 그 변동성이 크지 않다. 즉, 분산은 작게 나타난다. 서로 다른 샘플링 데이터를 선형회귀모델에 적합 시킨다면 다음과 같이 두 개의 그래프가 그려질 수 있겠다. 확인할 수 있듯, 두 그래프는 비슷한 모습을 띈다.



하지만, 이처럼 단순한 모델이 늘 적절하게 사용될 수 있는 것은 아니다. 새로운 데이터에 대해 예측과 같은 task를 수행했을 때 이러한 모델은 틀린 답을 내놓을 확률이 높다. 다시 말해, 편향은 크게 나타난다.

조금 더 복잡한 모델을 설계해볼까? 위 [그림 13]의 가장 오른쪽, 'overfitted'한 그래프를 살펴보자. 각 데이터의 지점을 하나하나 연결 지은 모습이다. 모든 관측치에 대해 예측 오류가 없도록 설계한 이러한 모델은 샘플링된 데이터에 따라 모델의 모양이 크게 달라진다. 다시 말해, 우리에게 주어진 데이터에 대해서는 충분히 그 기능을 수행할 수 있으나 일반화된 해석력을 확보하지는 못하므로 분산은 크게 나타난다. 그렇다면 편향은? 완전히 적합한 모델은 실제의 상황을 매우 정확하게 묘사하는 경향성을 띠므로 편향은 작게 나타난다. 그렇기에 아래 두 개의 그래프는 매우 다른 형태를 띠고 있는데, 이를 통해 '모델 간 분산이 크다'라는 표현의 의미를 직관적으로 캐치할 수 있다.



가장 이상적인 모델은 [그림 13]의 가운데에 위치한 모델이다. 이러한 모델을 우리는 'good fit', 또는 'robust'한 모델, 강건한 모델이라 일컬으며 [그림 12]의 그래프에서 편향과 분산 모두 적절히 작은 값으로 표현되는 상태에서 우리는 최적의 모델을 얻을 수 있다.

3. How to Avoid Overfitting?

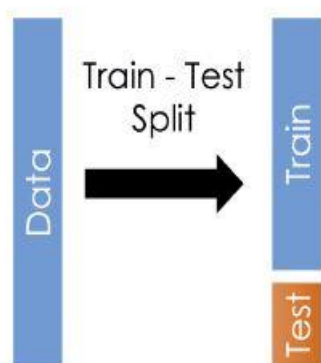
앞 장에서 모델의 복잡도와 함께, 흔히 일어날 수 있는 딜레마 상황인 편향-분산 트레이드오프에 대해서 살펴보았다. Underfitting된 모델도 문제가 있기에 촉각을 곤두세워야 하지만, 우리는 주어진 상황에서 우리의 모델이 overfitting되었는지 파악할 수 있어야 한다. **우리가 얻은 모델이 새로운 데이터에 대해 적합 시켰을 때에도 충분히 사용 가능해야 하기 때문에, 학습 데이터에 대해서만 좋은 성능을 보이는 모델은 적절하지 않다.** 마치 중간고사를 준비하기 위해 교재에서 제시된 예제를 풀었을 때는 높은 정확도를 보이지만, 실제 시험을 치렀을 때에는 형편없는 점수를 받는 상황을 지양해야 하는 것이다.

학습 과정 자체에서 오버피팅을 미리 예방하기 위해서 여러 가지 조치를 취할 수 있는데, 본격적으로 이들 개념을 설명하기에 앞서 어떻게 모델의 오버피팅을 감지하고 모델의 성능을 정확히 평가할 수 있는지에 대해 이야기해보려 한다.

A. 교차 검증 (Cross Validation)

앞서 제시한 중간고사 대비 예시를 생각해보자. 시험을 치기 전 미리 자신의 실력을 가늠할 수 있는 방법은 없을가? 실제 시험 문제와 매우 유사한 문제들 (예: 족보)을 풀어보고 이를 통해 자신의 점수를 예상해보는 것이 하나의 방법이 될 수 있겠다. 모델링, 그리고 이 과정에 사용되는 데이터셋에 대한 관점에서 보았을 때 족보 문제는 하나의 '검증 데이터 (validation data)'로서 그 역할을 수행할 수 있는 것이다.

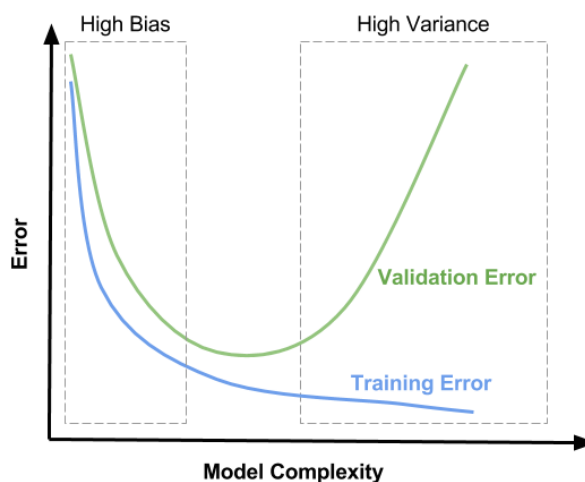
실제 수학적, 그리고 실제 y 값에 대한 정보를 알지 못하는 우리는 모델링을 통해 값을 추정해낼 수밖에 없다고 언급한 바 있다. **바로 이 검증 데이터를 통해 확인한 정확도를 가지고 테스트 데이터의 정확도를 추정하는 것이다.** 그렇다면 이러한 검증 데이터는 어떻게 구성하는 게 좋을가? 흔히 모델링 공모전에서는 참가자에게 학습 데이터와, 제출을 위한 테스트 데이터를 제공한다. 고로 참가자는, 주어진 학습 데이터의 일부를 검증 데이터로 삼아 이를 통해 자신이 설계한 모델의 정확도를 확인해야 한다. **이렇게 학습 데이터를 또 다시 학습 데이터와, 검증 데이터로 나누는 과정을 'train-test split'이라 일컫는다.** 이 과정을 통해 생성된 검증 데이터는 흔히 '테스트 데이터'라고도 불리므로, 라벨값이 주어지지 않은 테스트 데이터와 혼동하지 않도록 주의하자. 아래 그림을 통해 이 과정을 다시 확인해보자.



[그림 14]

train-test split 과정에서, 데이터의 몇 퍼센트를 학습 데이터로 삼을지 (몇 퍼센트를 테스트 데이터로 삼을지) 그 비율을 적절히 선정하는 것이 중요하다. 학습 데이터의 양이 너무 적을 경우 언더피팅이 일어나 편향이 큰 값으로 산출되고, 반대로 학습 데이터의 양이 많게 될 경우 오버피팅이 일어날 확률이 높다. 통상적으로는 학습 데이터:검증 데이터의 비율을 7:3 또는 8:2로 두어 언더피팅과 오버피팅 상황 모두를 적절히 피해가도록 한다.

이러한 방식으로 모델링을 진행한다면, 우리는 검증 데이터에 대한 error가 차츰 감소하는 현상을 기대할 수 있다. 하지만 모델의 복잡도가 클 경우 검증 데이터에 대한 error는 다시 증가하여 나쁜 성능을 보인다. 이와 같이 모델의 복잡도를 선정하는 과정과, 검증 데이터를 뒀으로써 모델의 오버피팅 여부를 확인하는 과정은 서로 맞물려 있다.



[그림 15]

train-test split을 통해 단일한 검증 데이터셋을 생성해내는 방법은 테스트 데이터의 error를 추정하는데 가장 최적의 방법이라 할 수 없다. 해당 검증 데이터셋이 전체 데이터의 경향성을 두루 포함하고 있음을 보장할 수 없으며 최악의 경우 이상치들의 집합으로만 구성되어, 왜곡된 모델을 설계하는 데 일조할 수 있기 때문이다. 이러한 문제점을 보완하기 위한 방법으로 k-fold 교차검증 (k-fold cross validation)이 제시된다.

CV는 위의 train-test split 과정을 반복적으로 수행하여 학습-검증 데이터의 조합을 여러 개 생성하는 방법으로, 각 조합 별 성능 평가 점수를 평균 산출하여 최종 테스트 데이터의 성능을 추정한다. 학습-검증 데이터의 조합을 생성하는 방식에는 여러 가지가 있다. 이 중 k-fold 교차 검증 방법을 주로 사용한다.

1) Leave-One-Out Cross Validation (LOOCV)

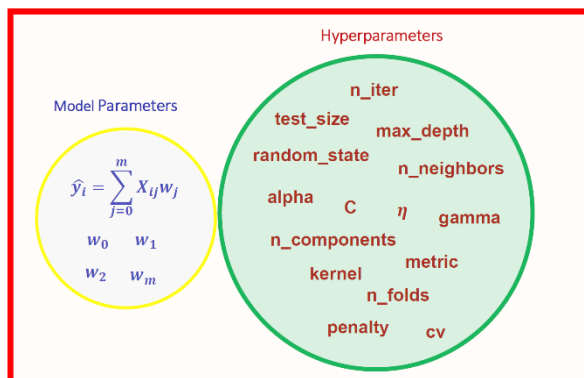
이름에서부터 알 수 있듯, 일부 데이터셋 하나만 별도로 떼내 그것을 검증 데이터셋으로 사용하는 방식. 데이터의 관측치가 n 개라면 각각의 관측치를 하나의 fold로 보는데 (많은 양의 음식을 소분하는 개념을 'fold'에 대입해보자), 그렇기 때문에 총 n 번의 실행을 거친다. 시간이 오래 걸리고, 컴퓨팅 파워를 많이 요구하는 작업이다.

2) K-fold Cross Validation

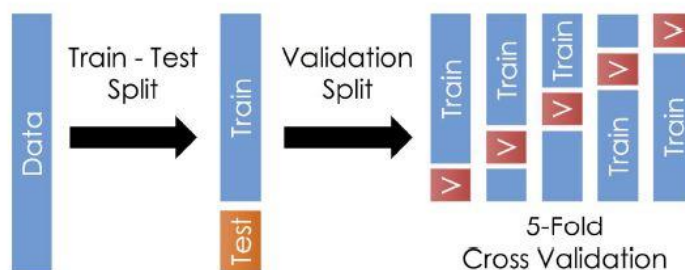
전체 데이터를 k 개의 그룹으로 나눈다. 여기서 k 는 주로 5 또는 10을 많이 사용한다. 이들 데이터가 k 개의 그룹으로 나뉘었을 때, 한 그룹씩 돌아가면서 검증 데이터로 사용하고 나머지 그룹들은 학습 데이터로 사용하는 방법이다. 각 과정에서 변수 선택을 진행하거나 모델의 *하이퍼 파라미터를 튜닝하면서 모델의 성능을 높여 나간다.

* 하이퍼 파라미터 (hyperparameter)?

하이퍼 파라미터는 모델의 성능에 직접적인 영향을 끼치는 변수의 값들로, 학습 데이터셋과 독립적으로 작용하며 때로 사용자의 별도 지정을 통해 설정되는 파라미터.



[그림 16] 파라미터와 하이퍼파라미터의 비교



[그림 17]

B. 차원의 저주 (Curse of Dimensionality)









학습 데이터에 대해서만 좋은 성능을 보이고 예측의 대상이 되는 테스트 데이터에 대해서는 그 정확도가 현저히 낮게 나오는 상황, 검증 데이터에 대한 예측 성능이 학습 데이터에 대한 예측 성능에 비해 현저하게 떨어지는 현상에 대해 우리는 '오버피팅'이라는 이름표를 붙여주었다. 샘플링을 하는 과정에서 적절하지 않은 방법을 취할 경우 오버피팅이 일어날 수도 있지만 feature들, 다시 말해 독립 변수의 개수가 너무 많아도 오버피팅이 발생할 수 있다.

3주차에 구체적으로 다룰 비지도학습 중 클러스터링의 예시를 통해 개념을 이해해보자. 우리의 목표는 주어진 사탕들에 대해, 비교적 매운 맛을 내는 붉은 계열의 사탕 (사탕이 매울 수 있는가는 의문)과 달콤한 맛의 푸른 계열 사탕으로 나누는 것이다. 앞선 문장에 기술했듯, 그저 사탕들을 '붉은 듯한'/'푸른 듯한'의 기준으로 나누어 분류를 할 경우 문제는 너무나도 간단해진다.

	Reddish	Bluish
	1	0
	1	0
	1	0
	1	0
	0	1
	0	1
	0	1

[그림 18]

위의 [그림 18]처럼 '해당된다'의 여부에 1, '해당되지 않는다'의 여부에 0의 값을 부여하여 분류를 시행했을 때 이 결과는 데이터의 대표성과 더불어 그 구체적인 경향성까지 잘 잡아낸 것을 확인할 수 있다. 근데 만약, 다른 제약 조건들을 더 덧붙인다면? 색을 표현하는 데 있어서 단순히 '붉다', '푸르다'와 같이 기술하지 않고 더욱 구체적인 표현들을 새로운 기준으로 삼는다면 어떻게 될까?

	Red	Maroon	Pink	Flamingo	Blue	Turquoise	Seaweed	Ocean
	1	0	0	0	0	0	0	0
	0	1	0	0	0	0	0	0
	0	0	1	0	0	0	0	0
	0	0	0	1	0	0	0	0
	0	0	0	0	1	0	0	0
	0	0	0	0	0	1	0	0
	0	0	0	0	0	0	1	0
	0	0	0	0	0	0	0	1

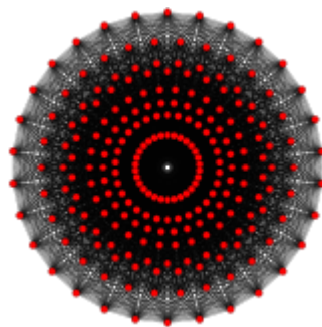
[그림 19]

위 [그림 19]에서는 사탕의 색을 정의하는데 있어서 더욱 다채로운 표현을 사용했음을 알 수 있다. 여기서 눈여겨볼 점은 하나의 색 기준에 대해 하나의 사탕만이 대응이 되었다는 점이다. 이러한 데이터를 컴퓨터에 입력하고 모델링을 시도한다면 컴퓨터는 다음과 같은 사고체계를 거치게 된다 (~~컴퓨터가 '사고'를 할 수 있느냐에 대한 문제는 여러 논란이 있지만..~~):-

- 1) 알고리즘의 단계에서 보았을 때, 각각의 색깔 간 연관성을 파악할 수 없다.
- 2) 고로, 각각의 사탕은 각자 한 개씩의 클러스터를 형성한다.
- 3) 에라 모르겠다, 8개를 그냥 4개/4개로 나누자.

오버피팅의 측면에서 문제가 되는 요소는 위의 2번 사고체계와 관련이 있다. 하나의 색에 대해 단 하나의 사탕만이 대응된 형태이므로 컴퓨터는 패턴을 파악하는데 어려움을 겪는다. 현재의 상황에서 오버피팅이 일어날 것이라는 사실은 당연하다. 학습하지 않은, 전혀 새로운 색에 대해서는 생뚱 맞은 답을 내놓을 것이 분명하다. 이러한 문제를 상쇄시키기 위해서는 현재 주어진 데이터들의 관측치 n 개 보다 훨씬 많은 관측치를 요구하는데, 현실적으로 데이터를 추가적으로 수집하는 것은 많은 어려움을 초래한다.

'차원'이라는 표현에 유의하여 이해해보자. [그림 18]에서 사탕들의 붉은 계열/푸른 계열 포함 여부를 좌표의 값으로 표현한다고 하면 이는 2차원 평면의 형태를 보인다. 마찬가지로 [그림 19]의 기준들, 즉 독립 변수에 대해 차원으로 나타낸다면 이는 8차원의 구조관계를 지닐 것이다. 8차원이라...! 상상이나 해보았나? 8차원 공간에서 변수 간의 관계를 정의한다면 아래 ([그림 20])와 같은 관계망이 그려질 것이다.

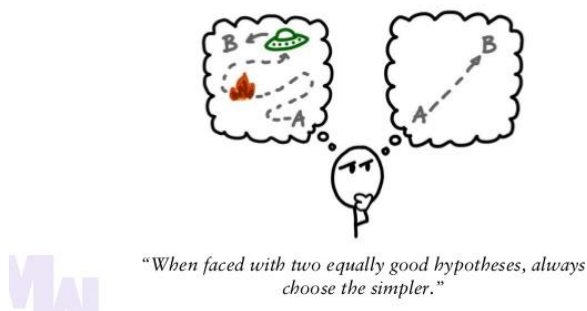


[그림 20]

8차원 공간에서 각 관측치의 경향성을 설명하실 수 있는 분? 아마 불가능할 것이다. 모델링을 하는 데 있어서 데이터의 추가적 수집을 요구한다는 점, 추가 관측치 수집이 어려워 그대로 모델링을 실시할 때 그 성능이 현저히 떨어진다는 점, 이에 더불어 해석이 불가능하다는 점 등 설명변수의 개수가 너무 많아 그 차원의 값이 크다면 여러 문제점이 발생한다. 이를 두고 우리는 '**차원의 저주 (Curse of Dimensionality)**'라 표현한다.

‘Simple is best’라고 했다. Occam’s Razor라는 표현 역시 이에 대해 설명한다. 어떠한 현상을 설명하는 데 있어 여러 조건이 동일하게 제시된다면 복잡하게 생각하지 말고 항상 더 간결한 설명을 고르라는 것이다. 그렇다고 해서 데이터들의 패턴을 파악하기 위해 동원되는 주요한 변수들을 누락해서는 안 되겠지만, 영국의 철학자 Occam은 지나치게 많은 요소들을 고려하여 길을 잃는 현상 (차원의 저주)을 경계하고 지양해야 함을 강조한다.

Occam's Razor



[그림 21]

어떠한 조치를 취해줌으로서 데이터들을 정의 내리는 설명 변수들의 개수를 줄일 수 있을까? 흔히 '차원 축소 (Dimensionality Reduction)'라 불리는 방법들에 주목해보자. 이와 관련한 내용은 우리 교안의 범위를 넘어가기 때문에 자세히 다루지는 않을 것이지만, 간단히 언급만 하고 넘어가려 한다 (회귀팀, 선대팀의 교안을 함께 보시면 더욱 좋습니다).

변수 선택법, 즉 **feature selection** 방법으로는 데이터의 특성을 가장 잘 설명하는 변수를 하나씩 더해가며 모델을 적합 시키는 전진 선택법 (forward selection)과, 반대로 변수를 하나씩 제거해가며 변수를 골라내는 후진 선택법 (backward elimination) 등이 있다. **Feature extraction** 방법은 고차원의 데이터를 저차원 공간의 데이터로 변환하는 것으로, 이 과정을 거치고 나면 이들 feature의 종류와 값 자체가 바뀌게 된다. Feature extraction의 대표적인 방법으로는 PCA (Principal Component Analysis)가 있다.

Early Stopping이라고 모델 학습 과정을 조기에 중단시키는 방법 또한 오버피팅을 예방할 수 있는데, 학습에 소요되는 시간을 제약하거나 '검증 데이터 성능이 90% 이상일 때 종료한다'와 같이 학습 중단 조건을 설정 (callback) 하는 방법이 있다. (이 부분은 딥러닝 팀의 교안을 참고하여 주세요).

실습 타임

간단한 실습을 해보겠습니다! 학습 목표는 아래와 같습니다!

1. R (kc_house_data.csv를 사용합니다)

- createDataPartition() 함수 이용, train-test split 원리 이해하기.
 - feature engineering은 모델링의 기본. 변수선택법에는 다양한 종류가 있더라~
 - cross validation 원리를 바탕으로, 직접 코드 확인해보기.
- (자세한 내용은 회귀팀 교안 참조. 일반적인 모델링 전 과정 설명하기 위해 가져왔어요)
- 회귀 문제에서의 성능 평가 지표로서 RMSE를 쓰더라~

2. Python (loan_data.csv를 사용합니다)

- train_test_split() 함수 이용, sklearn 라이브러리의 기본 기능 익히기.
- hyperparameter tuning ~ RandomizedSearchCV와 GridSearchCV 비교해보기.
- 분류 문제에서의 성능 평가 지표로서 활용되는 confusion matrix와 classification report 역시 확인해볼 수 있더라~