1. 데이터마이닝과 기계학습

1.1. 데이터 마이닝

데이터 마이닝(data mining)이란 대량의 데이터에서 가치있는 지식을 발견하는 방법을 말한다. 데이터 마이닝은 마케팅, 의료, 금융 등 다양한 분야에서 활용된다.

1.2. 배경

1.2.1. 데이터의 폭발

- 1991년 CERN이 WWW를 공개
- 2000년대 초, 고속 인터넷 보급
- 2004년 웹2.0 (콘텐츠를 보는 것에서 콘텐츠를 생성하는 것으로)
- 2004년 페이스북 시작(2005년말 사용자 6백만명)
- 2005년 유튜브 시작 (서비스 시작 18 후 월 1억뷰 도달)
- 2007년 아이폰 등장

1.2.2. 알고리즘의 발달

- 1805년 르장드르의 최소제곱법
- 1958년 로젠블랏의 퍼셉트론(최초의 인공신경망)
- 1974년 DARPA의 AI 예산 삭감 (1차 AI 겨울)
- 1986년 역전파 알고리즘 (다층 인공신경망이 가능해짐)
- 1987년 Lisp Machines사의 파산 (2차 AI 겨울)
- 1989년 합성곱 신경망 발표
- 1991년 순환신경망 발표
- 1992년 SVM 발표
- 1997년 LSTM 발표
- 2006년 딥러닝 등장

1.2.3. 계산 능력의 발달

1946년 최초의 디지털 컴퓨터 ENIAC

1961년 1기가플롭스(초당 10억번 계산)을 위한 비용: 1,529억 달러 (2017년 가치 기준)

1997년 IBM 딥블루(11기가플롭스)가 체스에서 세계 챔피턴 가리 카스파로프를 상대로 이김.

1999년 NVIDIA가 GeForce 256 그래픽카드를 발표

2002년 아마존이 아마존 웹 서비스(AWS)를 발표. 대용량의 하드웨어를 웹을 통해 시간 단위로 빌려 쓸 수 있게됨.

2004년 구글이 발표한 맵리듀스 알고리즘을 바탕으로 2006년 Hadoop이 등장. '빅데이터'의 시대 개막

2013년 400달러짜리 가정용 게임기인 소니 플레이스테이션4의 속도가 1.84테라플롭스(초당 1조84백만번 계산). 1기가플롭스당 비용은 0.23달러

2017년말 기준 1기가플롭스당 비용은 0.03달러

1.3. 데이터 마이닝의 과정

데이터 마이닝은 다음과 같은 과정으로 진행된다.

- 데이터 생성 및 추출
- 전처리
- 탐색적 데이터 분석
- 기계 학습
- 해석 및 활용

먼저 데이터 생성 및 추출은 분석할 데이터를 만들거나 데이터베이스에서 추출하는 과정이다. 말로는 간단하지만 실제로는 가장 어려운 부분이라고 할 수 있다. 대부분의 경우 이 단계에서 현실적인 장벽에 가로막히기 때문이다.

전처리(preprocessing)는 데이터 분석에 앞서(pre-)를 분석하기 좋게 처리(processing)하는 과정이다. 대부분의 데이터는 분석 목적으로 수집된 것이 아니다. 예를 들어 금융 데이터의 경우를 살펴보자. 금융 데이터의 경우 데이터를 축적하는 1차 목적은 거래를 정확히 처리하는 것이다. 금융 데이터베이스에서는 이를 계정계 시스템이라고 한다. 분석을 위한 시스템은 정보계 시스템이라고 한다. 은행, 보험, 신용카드 등 금융회사의 입장에서는 정보계보다 계정계 시스템이 더 중요하다.

계정계 시스템은 정확성이 중요하기 때문에 매우 보수적이다. 프로그래밍 언어 중에 1950년대에 만들어진 COBOL이라는 언어가 있는데, 21세기가 된 지금도 한국 금융회사 계정계 시스템 중에는 COBOL로 작성된 부분이 돌아가고 있다. 1990년대에 만들어진 Java도 국내에서는 2013년에야 전북은행에서 도입되기도 하였다. 이렇든 계정계 시스템은 거래의 정확성에 중심을 두고 짜여져 있기 때문에 데이터가 분석하기에 불편한 형태로 쌓여있는 경우가 많다.

반대로 거래의 정확성과 관련없는 부분은 매우 허술한 경우가 많다. 신용카드에서 사용자의 성별이 누락되어 있거나 가맹점의 상호가 잘못되어 있더라도 거래를 처리하는데는 문제가 없기 때문에 그대로 방치되는 경우가 흔하다. 실제로 유명 빵집 체인의 경우 데이터베이스 내에 상호가 제각각으로 입력되어 있기도 하다.

전처리는 데이터 마이닝에서 가장 손이 많이 가는 부분이다. 이론보다도 많은 경험을 통해 빠른 처리 능력을 갖추는 것이 중요하다.

탐색적 데이터 분석(Explartory Data Analysis)은 특정한 가설 없이 데이터의 특성을 살펴보는(탐색) 분석 과정이다. 데이터에서 평균, 분산, 분포를 뽑아보거나 집단 간 비교, 변수들 사이의 상관관계를 수치적, 시각적으로 본다. 탐색적 데이터 분석을 통해 분석가는 데이터의 특성을 이해하고 분석의 방향성을 설정할 수 있게 된다. 많은 경우에는 이정도만으로도 분석을 마치고 해석 및 활용 단계로 넘어가기도 한다.

기계 학습(machine learning)은 컴퓨터에게 데이터 패턴을 학습시켜서 예측하게 하는 과정이다. 데이터 마이닝에서 그 중요성이 점점 더 커지고 있는 부분이나, 현업에서는 생략하는 경우도 흔하다. 기계학습의 결과는 직접적인 해석이 어렵고 컴퓨터를 직접 서비스에 활용해야 한다. 그런데 충분한 성능이 나오지 않는다면 서비스에도 활용하기 어렵게 된다.

해석 및 활용은 데이터 분석의 결과를 해석해서 비즈니스 계획을 수립, 실행하거나 또는 기계학습의 경과를 서비스에 붙여서 활용하는 단계이다.

1.4. 기계학습

컴퓨터는 입력된 자료를 바탕으로 계산한 결과를 출력한다. 출력되는 결과의 성격에 따라 기계학습은 지도 학습, 비지도 학습, 강화 학습로 구분된다.

1.4.1. 지도 학습

지도 학습(supervised learning)은 출력되는 결과의 올바른 답을 알고 있는 문제이다. 예를 들어 X레이 사진에서 암을 진단하는 문제의 경우 입력은 X레이 사진, 출력은 암 진단(양성/음성)이 된다. 이 경우 X레이 사진과 각 사진의 진단이 있으면 컴퓨터에 이 두 가지를 넣고 학습시켜서 새로운 X레이 사진을 입력으로 주었을 때 암을 진단하게 할 수 있다.

지도 학습은 기계 학습에서 가장 널리 다뤄지는 문제이다. 기계 번역, 추천 시스템, 이미지 인식 등이 모두 지도 학습에 속한다.

지도 학습은 출력의 형태에 따라 회귀(regression)와 분류(classification)로 나뉜다. 회귀는 연속적인 값(예: 온도, 나이, 가격, 만족도 등)을, 분류는 이산적인 값(예: 성별, 물체의 종류, 질병 진단 등)을 출력한다.

지도학습을 비스지스에 활용하는 사례로는 다음과 같은 것들이 있다.

- 제품의 특성, 가격 등으로 판매량 예측
- 채무자의 직업, 소득 등으로 상환 가능성 예측
- 피부 손상의 사진으로 상처가 치료될 때까지 기간을 예측
- 이메일의 내용으로 스팸 여부 판단

- 건물의 여러 가지 요소를 바탕으로 화재 위험 진단

1.4.2. 비지도 학습

비지도 학습(unsupervised learning)은 출력되야할 결과의 답을 모르는 문제이다. 예를 들어 고객들을 비슷한 몇 가지 집단으로 나눠서 마케팅을 하고자 할 때 어떤 고객이 어떤 집단에 속하게 될지는 실제로 나눠보기 전에는 알 수 없다. 이런 비지도 학습에서는 '비슷함'만 정의하고 컴퓨터에게 알아서 고객을 나누게 한다. 컴퓨테에게 정답을 알려주지 않기 때문에 '비지도'학습이라고 한다.

비지도 학습도 출력의 형태에 따라 차원 축소(dimensionality reduction)와 군집(clustering)으로 나눈다. 차원 축소는 연속적인 값을, 군집은 이산적인 값을 출력한다. 앞의 고객들을 집단으로 나누는 경우가 군집에 해당한다.

차원 축소의 조금 낯선 개념일 수도 있는데, 우리가 익숙한 예가 있다. 시험의 '총점'이다. '총점'은 여러 과목의 점수를 더해서 하나의 점수로 만든다. 세 과목의 점수를 더 해 하나의 총점으로 만들면 3차원에서 1차원으로 줄어드는 것과 같다. 그래서 차원 '축소'라고 하는 것이다. 기계학습에서 차원 축소는 단순히 총점을 내는 것보다는 훨씬 복잡한 방법이지만 기본적인 개념은 동일하다.

비지도 학습은 비즈니스에 직접적으로 활용하는 경우는 지도학습보다 적다. 마케팅에서 고객들을 군집하거나, 추천시스템에서 활용되기도 한다.

비지도 학습은 지도 학습의 전처리 과정으로 활용되기도 한다. 차원을 축소하면 입력의 크기가 줄어들기 때문에 지도 학습을 할 때 패턴을 파악하기가 더 쉬워질 수 있기 때문이다.

비지도학습을 비즈니스에 활용하는 사례로는 다음과 같은 것들이 있다.

- 소비패턴이 비슷한 고객들끼리 군집화
- 고객과 시청패턴이 비슷한 다른 고객들이 즐겨보는 영화를 추천 (추천시스템)
- 게시판 등에서 고객 의견을 비슷한 것끼리 묶어서 정리

1.4.3. 강화 학습

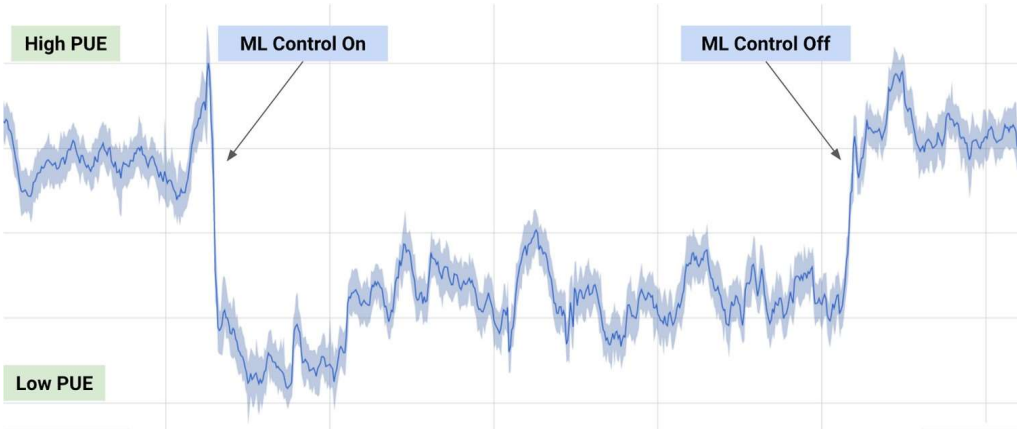
강화 학습(reinforcement learning)은 원래 심리학 용어로 동물에게 보상과 처벌을 통해 학습을 시키는 것을 말한다. 기계학습에서 강화 학습은 이러한 심리학적 과정을 컴퓨터로 흉내낸 것이다.

강화학습은 지도 학습과 비슷하지만 정답 대신 그 답이 얼마나 '좋은 지'만 아는 경우에 활용한다. 예를 들어 인터넷 쇼핑몰에 똑같은 버튼에 "지금 구매하세요"와 "주문하기" 두 가지 문구를 쓸 수 있다고 해보자. 어떤 버튼이 더 좋은지는 처음부터 알 수 없다. 하지만 고객들에게 버튼을 보여주었을 때 고객들이 그 버튼을 누른다면 그 버튼은 '좋다'고 말할 수 있다. 강화학습을 적용하면 컴퓨터가 시행착오를 거쳐 더 나은 문구의 버튼을 선택하게 할 수 있다.

쇼핑몰의 버튼은 한 단계로 끝나는 문제지만 여러 단계의 문제에도 강화학습을 적용할 수 있다. 바둑의 경우 '신의 한 수'가 무엇인지는 알 수 없다. 그러나 한 국의 바둑이 끝났을 때 승패는 바둑의

규칙을 아는 사람이라면 누구나 알 수 있다. 바둑에 강화학습을 적용하면 컴퓨터가 승패로부터 수를 되짚어서 '신의 한 수'에 가까운 수를 스스로 터득할 수 있게 된다. 구글의 알파고(AlphaGo)는 이러한 방식으로 바둑을 깨우쳐 세계 정상급 바둑기사인 한국의 이세돌과 중국의 커제를 상대로 압도적인 승리를 거두었다.

강화학습을 비즈니스에 활용한 사례는 아직 많지 않다. 알파고를 만든 구글 딥마인드는 2016년 강화학습을 통해 구글 데이터 센터의 냉방 비용을 40% 절감했다고 밝혔다.



앞으로 강화학습은 자율 주행 자동차, 실시간 트레이딩 등에 활용될 수 있을 것으로 기대된다.

1.4.4. 기계학습 모형

모형(model)이란 데이터의 패턴을 일정한 수학적 형태로 표현한 것이다. 기계학습은 이러한 모형을 바탕으로 이뤄진다.

지도 학습에서 널리 사용되는 기계 학습 모형에는 선형 모형, 최근접 이웃, 인공 신경망, 의사결정 나무 등이 있다.

선형 모형(linear model)은 데이터의 패턴이 직선적인 형태를 가지고 있다는 가정에 바탕을 둔다. 단순한 모형이지만 결과가 안정적이고 해석이 쉽기 때문에 널리 쓰인다.

최근접 이웃(nearest neighbor)은 새로운 사례가 입력으로 들어왔을 때 기존의 사례 중에서 가장 유사한(최근접) 사례들(이웃)을 찾아 이들의 평균을 내거나 다수결하여 출력한다. 간단하지만 정확한 방법이다. 다만 기존의 사례가 너무 많거나 복잡한 데이터의 경우에는 사용하기 어렵다는 단점이 있다.

인공 신경망과 의사 결정 나무는 연구와 응용 양쪽에서 최근 가장 활발한 모형이다. 인공 신경망(artificial neural network)은 생물의 신경망, 인간의 두뇌와 같은 것을 모방하는 방법이다. 학습이 느리고 많은 데이터가 필요하다는 단점이 있었으나 빅데이터의 등장과 컴퓨터 속도의 향상으로 최근 각광을 받게 되었다. 흔히 말하는 딥러닝(deep learning)이 인공 신경망에 바탕을 둔 것이다. 인공 신경망은 특히 언어, 소리, 이미지 같은 비정형(unstructured) 데이터에 강점이 있다.

의사 결정 나무(decision tree)는 스무고개와 같은 식으로 예/아니오로 답할 수 있는 질문들을 거쳐 예측을 할 수 있는 방법이다. 데이터를 바탕으로 이러한 질문의 종류와 순서를 결정하는데 정확

성이 매우 높다. 최근의 경향은 수 백 개의 의사 결정 나무들을 합쳐서 예측 능력을 높이는 앙상블(ensemble) 방법을 사용한다. 의사 결정 나무는 표의 형태로 나타낼 수 있는 정형(structured) 데이터에 강점이 있다.

1.4.5. 기계학습의 과정

기계 학습의 과정은 다음과 같은 순서로 진행된다.

- 데이터 분할
- 다양한 모형과 하이퍼 파라미터로 훈련
- 테스트
- 모형 선택

데이터 분할은 데이터를 훈련용(training)과 테스트용(test)으로 나누는 것이다. 컴퓨터가 데이터를 모두 '외워' 버리면 우리가 가진 데이터에서는 성능이 높아 보일 수도 있으나 실제에 적용하면 성능이 떨어질 수 있다. 따라서 컴퓨터가 단순히 데이터를 '외웠'는지 아니면 데이터의 패턴을 잘 깨우쳤는지 구별하기 위해 데이터 중 일부를 테스트용으로 나눠놓는 것이다.

데이터를 나눈 후에는 다양한 모형에 학습 또는 훈련을 시킨다. 대부분의 기계학습 모형들은 모형의 특성을 조절하는 하이퍼 파라미터(hyperparameter)를 가지고 있다. 같은 모형이라도 하이퍼 파라미터에 따라 성격이 달라진다. 따라서 다양한 모형과 또 모형마다 다양한 하이퍼 파라미터로 훈련을 시킨다.

이렇게 다양한 모형/하이퍼파라미터를 시도하는 이유는 실제로 학습을 시켜보기 전에는 어떤 것이 잘 작동할지 알 수 없기 때문이다. 모형이 잘 작동하려면 모형의 형태와 데이터가 가진 패턴의 형태가 맞아야 한다. 그런데 데이터가 가진 패턴의 형태는 매우 복잡해서 우리가 파악하기 어렵기 때문에 일단 학습을 시켜보고 잘 작동하는 모형을 고른다.

테스트는 말 그대로 앞에서 훈련시킨 다양한 모형들을 테스트 데이터로 테스트 하는 것이다.

모형 선택은 테스트에서 성능이 가장 좋은 모형을 선택한다. 여기서는 여러 가지 비즈니스적인 고려가 들어가기도 한다. 예를 들면 예측 자체는 잘하지만 계산이 오래 걸리는 모형보다 조금 예측이 부정확하더라도 계산이 빠른 모형을 선택할 수도 있다.

1.4.6. 기계학습과 통계학의 차이

기계학습과 통계학은 모두 데이터에서 패턴을 발견하는 데 초점을 맞춘 학문이다. 둘의 내용은 거의 비슷하지만 접근 방식에 약간의 차이가 있다. 간단히 말하자면 통계학은 수학 또는 과학의 하위 분과이고, 기계학습은 공학의 하위 분과라고 할 수 있다.

현대적인 통계학은 19세기 후반에서 20세기 초반에 성립한 수학의 응용 분야이다. 주로 데이터의 패턴을 **해석**하고 인과관계를 이해하는데 많은 비중을 두고 발전해왔다.

기계학습은 20세기 후반에 발전한 인공지능의 한 분야로 통계학에서 많은 영향을 받았다. 혹자는 **통계적 학습**(statistical learning)이라고 부르기도 한다. 기계학습은 통계학을 인공지능을 만들기 위한 수단으로서 활용하기 때문에 데이터에 나타낸 패턴을 해석하거나 인과 관계를 이해하는데는 큰 비중을 두지 않는다.

1.5. 최근접 이웃(kNN)

UCI 기계학습 저장소에서 와인 퀄리티 데이터셋을 다운 받는다.

- 레드 와인 데이터
- 화이트 와인 데이터

1.5.1. 데이터 불러오기

```
data.url = paste0(
  'https://archive.ics.uci.edu/ml/',
  'machine-learning-databases/wine-quality/',
  'winequality-red.csv')
```

`read.csv` 함수를 이용하여 데이터를 읽어들인다. `data.url` 이 인터넷 주소이기 때문에 `url` 함수에 넣어준다. `sep = ';'` 은 이 파일이 세미콜론(;)으로 구분되어 있기 때문에 지정해주는 것이다.

```
red.wine = read.csv(url(data.url), sep = ';')
```

읽어온 파일의 앞부분을 확인해보자.

```
head(red.wine)
```

```
fixed.acidity volatile.acidity citric.acid residual.sugar chlorides
1           7.4             0.70        0.00            1.9     0.076
2           7.8             0.88        0.00            2.6     0.098
3           7.8             0.76        0.04            2.3     0.092
4          11.2             0.28        0.56            1.9     0.075
5           7.4             0.70        0.00            1.9     0.076
6           7.4             0.66        0.00            1.8     0.075
  free.sulfur.dioxide total.sulfur.dioxide density   pH sulphates alcohol
1                  11                   34  0.9978 3.51      0.56     9.4
2                  25                   67  0.9968 3.20      0.68     9.8
3                  15                   54  0.9970 3.26      0.65     9.8
4                  17                   60  0.9980 3.16      0.58     9.8
5                  11                   34  0.9978 3.51      0.56     9.4
6                  13                   40  0.9978 3.51      0.56     9.4
  quality
1       5
2       5
3       5
4       6
5       5
6       5
```

데이터의 형태를 확인해보자.

```
dim(red.wine)
```

```
[1] 1599   12
```

1.5.2. caret

`caret` 패키지를 불러온다.

```
library(caret)
```

1.5.3. 데이터 분할

```
set.seed(1234)
```

```
idx = createDataPartition(red.wine$quality, p=.8, list=F)
```

```
data.train = red.wine[idx, ]
data.test = red.wine[-idx, ]
```

```
dim(data.train)
```

```
[1] 1281   12
```

```
dim(data.test)
```

```
[1] 318  12
```

1.5.4. 전처리 없이 학습

전처리

전처리는 일단 생략하도록 하자.

학습

```
knn = train(
    quality ~ .,
    data=data.train,
    method='knn',
    tuneGrid=data.frame(.k = 3))
```

테스트

훈련용 데이터

```
y.train.pred = predict(knn, data.train)
```

```
RMSE(data.train$quality, y.train.pred)
```

```
[1] 0.540363
```

```
R2(data.train$quality, y.train.pred)
```

```
[1] 0.5602867
```

테스트용 데이터

```
y.pred = predict(knn, data.test)
```

```
RMSE(data.test$quality, y.pred)
```

```
[1] 0.7743401
```

```
R2(data.test$quality, y.pred)
```

```
[1] 0.1520506
```

1.5.5. 전처리를 포함시킨 학습

전처리 & 학습

```
knn.2 = train(
    quality ~ .,
    data=data.train,
    method='knn',
    preProc = c('center', 'scale'),
    tuneGrid=data.frame(.k = 3))
```

테스트

```
y.train.pred = predict(knn.2, data.train)
```

```
RMSE(data.train$quality, y.train.pred)
```

```
[1] 0.4906523
```

```
R2(data.train$quality, y.train.pred)
```

```
[1] 0.6381152
```

테스트용 데이터

```
y.pred = predict(knn.2, data.test)
```

```
RMSE(data.test$quality, y.pred)
```

```
[1] 0.6679292
```

```
R2(data.test$quality, y.pred)
```

```
[1] 0.3177571
```

1.5.6. 다양한 하이퍼파라미터로 학습

```
knn.3 = train(
    quality ~ .,
    data=data.train,
    method='knn',
    preProc = c('center', 'scale'),
    tuneGrid=data.frame(.k = seq(1, 50, 5)))
```

```
knn.3
```

```
k-Nearest Neighbors

1281 samples
  11 predictor

Pre-processing: centered (11), scaled (11)
Resampling: Bootstrapped (25 reps)
Summary of sample sizes: 1281, 1281, 1281, 1281, 1281, 1281, ...
Resampling results across tuning parameters:

  k   RMSE       Rsquared
   1  0.8325262  0.2262511
   6  0.7359568  0.2566581
  11  0.7071393  0.2858751
  16  0.6947602  0.3021197
  21  0.6906320  0.3074262
  26  0.6885886  0.3104781
  31  0.6865187  0.3144305
  36  0.6847788  0.3183983
  41  0.6830465  0.3230313
  46  0.6816641  0.3268138

RMSE was used to select the optimal model using  the smallest value.
The final value used for the model was k = 46.
```

테스트

```
y.train.pred = predict(knn.3, data.train)
```

```
RMSE(data.train$quality, y.train.pred)
```

```
[1] 0.6544798
```

```
R2(data.train$quality, y.train.pred)
```

```
[1] 0.3667184
```

```
y.pred = predict(knn.3, data.test)
```

```
RMSE(data.test$quality, y.pred)
```

```
[1] 0.627078
```

```
R2(data.test$quality, y.pred)
```

```
[1] 0.3584239
```