

December 10, 2025

- 강의명: CS109A: 데이터 과학 입문
- 주차: Lecture 01
- 교수명: Pavlos Protopapas, Kevin Rader, Chris Gumb
- 목적: Lecture 01의 핵심 개념 학습

▣ 핵심 요약

본 문서는 CS109A '데이터 과학 입문' 과정의 첫 번째 강의와 실습 세션을 통합한 종합 노트입니다. 데이터 과학의 정의, 역사적 발전, 5단계 프로세스 등 핵심 이론을 다룹니다. 또한 과정의 상세한 평가 기준, 특히 중요한 출석 정책을 설명합니다. 마지막으로, 파이썬(Python)을 활용한 웹 스크레이핑(Web Scraping) 기초 실습을 단계별로 상세히 안내하여 이론과 실습을 연결합니다.

Contents

1	과정 개요 (Course Overview)	2
1.1	CS109A: 데이터 과학의 첫걸음	2
1.2	예상 학습 로드맵	2
1.3	과정의 3대 목표	2
2	데이터 과학(Data Science)이란 무엇인가?	3
2.1	데이터 과학의 정의: 3단계 접근	3
2.2	데이터 과학의 역사적 발전 4단계	3
2.3	데이터 과학의 3대 구성 요소	3
2.4	데이터 과학의 잠재력과 위험성	4
3	데이터 과학의 5단계 프로세스	5
4	CS109A 과정 상세 안내	6
4.1	교수진 및 조교(TAs)	6
4.2	학습 철학: "Wax on, Wax off"	6
4.3	학습 도구	6
4.4	도움 받는 방법 (How to get help)	6
5	평가 및 주요 정책	7

5.1	5대 평가 요소 및 비중	7
5.2	숙제(Homework) 상세	7
5.3	선수과목 (Prerequisites) 진단	7
5.4	출석 및 지각 정책	8
6	실습 (Section 1): 웹 스크레이핑 입문	9
6.1	웹 스크레이핑이란?	9
6.2	실습 라이브러리	9
6.3	1단계: 웹 스크레이핑 윤리 및 규칙 확인	9
6.4	2단계: HTML 기초와 브라우저 '검사' 도구	9
6.5	3단계: requests로 데이터 가져오기	10
6.6	4단계: BeautifulSoup로 HTML 파싱하기	10
6.7	5단계: 데이터 추출 및 구조화(노벨상 예제)	11
6.8	6단계: pandas로 데이터 프레임 변환	12
6.9	(심화) 비동기(Async) 스크레이핑	12
7	자주 묻는 질문 (FAQ)	13

1 과정 개요 (Course Overview)

1.1 CS109A: 데이터 과학의 첫걸음

본 과정(CS109A)은 데이터 과학과 인공지능(AI) 분야의 전문가가 되기 위한 여정의 시작입니다. 최신 모델(예: LLM)을 단순히 사용하는 것을 넘어, 그 근간이 되는 **기초 원리(Fundamentals)**를 탄탄히 다지는 것을 목표로 합니다.

이 과정은 무술을 배울 때 기본자세(예: "Wax on, Wax off")를 반복 숙달하는 것과 같습니다. 때로는 지루하게 느껴질 수 있지만, 이 기초가 없으면 복잡한 모델을 제대로 이해하고 활용할 수 없습니다.

1.2 예상 학습 로드맵

본 과정은 데이터 과학의 전체 흐름을 따르며 다음과 같은 순서로 진행됩니다.

1. 데이터 수집 및 탐색 (Weeks 1-2): 웹 스크레이핑, 데이터 정제(Wrangling), 탐색적 데이터 분석(EDA) 및 시각화.
2. 회귀 (Regression) (Weeks 3-5): K-최근접 이웃(KNN), 선형 회귀, 다중/다항 회귀, 모델 선택(Cross Validation), 추론, 정규화(Ridge, Lasso).
3. 베이지안 모델링 (Bayesian) (Week 6): 베이지안 추론 프레임워크, 베이지안 선형 회귀.
4. 분류 (Classification) (Weeks 7-9): KNN 분류, 로지스틱 회귀, 계층적 모델링. (중간고사 포함)
5. 데이터 이슈 (Data Issues) (Week 10): 결측치(Missingness), 인과 추론(Causal Inference), 편향성 및 윤리.
6. 트리 기반 모델 (Tree-Based) (Weeks 11-14): 의사결정 나무, 배깅(Bagging), 랜덤 포레스트(Random Forest), 부스팅(Boosting).

1.3 과정의 3대 목표

본 과정은 이론, 실습, 그리고 실제 영향력이라는 세 가지 축을 중심으로 구성됩니다.

- **이론과 직관 (Theory/Intuition):** 통계 분석 및 머신러닝의 핵심 개념을 이해하고, 모델 평가 지표를 학습하며, 분석 결과로부터 통찰력을 추출합니다.
- **실습 (Practice):** 파이썬 라이브러리(Pandas, Scikit-learn 등)를 사용하여 머신러닝 및 딥러닝 모델을 구현하고, 다양한 종류의 데이터를 다루는 법을 배웁니다.
- **영향력 (Impact):** 데이터 과학을 사용해 실제 문제를 해결하고, 그 과정에서 발생할 수 있는 사회적, 윤리적 영향을 평가합니다.

2 데이터 과학(Data Science)이란 무엇인가?

2.1 데이터 과학의 정의: 3단계 접근

데이터 과학을 이해하는 가장 좋은 방법은 그 역사적 맥락과 구성 요소를 살펴보는 것입니다.

- **1단계 (핵심 요약):** 데이터 과학은 데이터로부터 의미 있는 통찰과 가치를 추출하는 모든 과정을 다루는 융합 학문입니다.
- **2단계 (비유):** 데이터 과학자는 데이터라는 원석을 캐내어(수집), 불순물을 제거하고(정제), 세공하여(모델링), 아름다운 보석(통찰)으로 만들어내는 장인과 같습니다.
- **3단계 (기술적 설명):** 정형/비정형 데이터를 수집, 관리, 탐색하고, 통계 및 머신러닝 모델을 적용하여 패턴을 발견하거나 미래를 예측하며, 그 결과를 시각화하여 설득력 있게 전달하는 전 과정을 포함합니다.

2.2 데이터 과학의 역사적 발전 4단계

인류가 세상을 이해하는 방식은 다음과 같이 발전해왔으며, 데이터 과학은 가장 최신의 패러다임입니다.

1. 경험적 관찰 (Empirical Observation) (고대)

- 밤하늘의 별을 세거나 농작물 수확량을 기록하는 등, 직접적인 관찰과 경험을 통해 데이터를 수집했습니다.
- 이는 통계학의 초기 형태라 볼 수 있습니다.

2. 방정식 (Equations) (근대 과학)

- 뉴턴, 아인슈타인 등이 등장하며 세상이 작동하는 근본 원리(First Principles)를 수학 방정식으로 설명하기 시작했습니다.
- 예: $F = ma$, $E = mc^2$

3. 컴퓨팅 (Computation) (20세기)

- 1단계의 방정식들이 너무 복잡하여 손으로 풀기 어려워지자, 컴퓨터를 사용하여 시뮬레이션하고 해를 구하기 시작했습니다.

4. 데이터 과학 (Data Science) (현대)

- 2단계(방정식)를 건너뛰는 경향이 나타납니다.
- 세상의 근본 원리(방정식)를 완벽히 이해하지 못하더라도, 방대한 데이터(1단계)와 강력한 컴퓨팅 (3단계)을 결합하여 세상의 작동 방식을 근사(approximate)하거나 예측합니다.

2.3 데이터 과학의 3대 구성 요소

데이터 과학은 세 가지 핵심 분야가 교차하는 지점에 있습니다.

- 컴퓨터 과학 / IT (Computer Science / IT): 데이터 수집, 저장, 처리, 소프트웨어 개발.
- 수학 / 통계 (Math & Statistics): 모델링, 가설 검증, 예측의 수학적 기반.
- 도메인 지식 / 비즈니스 (Domain Knowledge): 해당 분야(예: 천문학, 의학, 금융)에 대한 전문 지식.

주의사항

도메인 지식의 중요성

도메인 지식이 없는 데이터 과학은 ”엉뚱한 문제”를 풀 위험이 큽니다.

한 천문학자가 컴퓨터 과학자에게 데이터 분석을 의뢰했습니다. 한 달 후, 컴퓨터 과학자는 ”문제를 완벽히 해결했다”고 했지만, 알고 보니 그는 데이터의 의미와 천문학적 맥락을 전혀 이해하지 못해 완전히 잘못된 문제를 푼 것이었습니다.

데이터 과학은 자동차 정비소에 차를 맡기듯 문제를 던지고 끝내는 것이 아닙니다. 반드시 해당 분야의 전문가와 긴밀히 소통하며 문제 자체를 함께 정의해야 합니다.

2.4 데이터 과학의 잠재력과 위험성

데이터 과학은 강력한 도구이며, 그에 따른 잠재력과 위험성을 동시에 가집니다.

데이터 과학의 잠재력

- 질병 진단: 혈액 도말 샘플 이미지로 말라리아 감염 여부 진단.
- 신약 개발: 언어 모델(LM)을 활용하여 새로운 약물 조합 발견.
- 생성형 AI (Generative AI): 텍스트 프롬프트(예: ”안경 쓴 그리스인 교수”)로부터 이미지 생성.
- 자율 주행: 야간에도 안전하게 운행하는 자율주행 트럭.

데이터 과학의 위험성과 윤리

- 성별 편향 (Gender Bias): 특정 직군(예: 엔지니어) 채용 모델이 남성 지원자에게 유리하게 작동.
- 인종 편향 (Racial Bias): 미국 법원에서 사용되는 재판 위험도 예측 모델이 유색 인종에게 불리하게 편향됨.

이러한 위험성 때문에 우리는 데이터 과학을 맹목적으로 사용해서는 안 되며, 항상 비판적 사고(Critical Thinking)를 견지하고 모델의 공정성과 윤리성을 점검해야 합니다.

3 데이터 과학의 5단계 프로세스

데이터 과학 프로젝트는 일반적으로 다음 5단계를 순환하며 진행됩니다.

1. 흥미로운 질문하기 (Ask an interesting question)

- 가장 중요하고 첫 번째 단계입니다. "데이터가 있으니 뭔가 찾아봐"가 아니라, 명확한 가설이나 과학적 목표를 설정해야 합니다.
- (예: "우리가 예측/추정하려는 것은 무엇인가?", "모든 데이터가 있다면 무엇을 할 것인가?")

2. 데이터 획득하기 (Get the Data)

- 질문에 답하기 위해 필요한 데이터를 수집합니다.
- (예: "데이터는 어떻게 샘플링되었는가?", "어떤 데이터가 관련 있는가?", "라이선스나 개인정보 보호 문제는 없는가?")

3. 데이터 탐색하기 (Explore the Data - EDA)

- 데이터를 시각화하고 요약하며 패턴을 찾습니다. 이 단계에서 많은 시간을 절약할 수 있습니다. (때로는 이 단계만으로도 충분한 답을 얻기도 합니다.)
- (예: "데이터를 플롯팅해 보았는가?", "이상치(anomaly)나 심각한 오류는 없는가?", "어떤 패턴이 보이는가?")

4. 데이터 모델링하기 (Model the Data)

- 데이터의 패턴을 학습하거나 미래를 예측하는 통계/머신러닝 모델을 구축합니다.
- (예: "모델 구축(Build) -> 모델 학습(Fit) -> 모델 검증(Validate)")

5. 결과 전달/시각화하기 (Communicate/Visualize the Results)

- 분석 결과를 비전문가도 이해할 수 있도록 스토리텔링과 시각화를 통해 전달합니다.
- (예: "우리는 무엇을 배웠는가?", "결과가 말이 되는가(make sense)?", "효과적으로 스토리를 전달 할 수 있는가?")

4 CS109A 과정 상세 안내

4.1 교수진 및 조교(TAs)

- **Pavlos Protopapas:** 데이터 과학 석사 과정의 Scientific Director. 천문학과 머신러닝 연구를 수행하며, 요리자격증을 보유하고 있습니다.
- **Kevin Rader:** 통계학과 선임 지도교수(Senior Preceptor). 학부 교육을 담당하며, 스포츠 및 의학 분야 데이터 분석에 관심이 많습니다. (필라델피아 이글스 팬 - "Go Birds!")
- **Chris Gumb:** 지도교수(Preceptor). 약 30명에 달하는 TF 팀을 조율하고 과정 운영을 지원합니다.
- **Teaching Fellows (TFs):** 약 30명의 TF가 섹션(실습)과 오피스 아워를 담당합니다.

4.2 학습 철학: "Wax on, Wax off"

▣ 예제:

영화 <베스트 키드>에서 미야기 사부는 제자에게 가라데 대신 자동차 왁스 칠(Wax on, Wax off)만 반복시킵니다. 제자는 불평하지만, 이 무의미해 보이는 반복 작업이 실제 대련에서 방어 동작의 완벽한 기초가 되었음을 깨닫습니다.

CS109A도 마찬가지입니다. 데이터 탐색, 모델 학습, 검증 등 기본적인 절차를 반복 숙달시키는 과정이 많을 것입니다. 이는 단순히 코드를 'fit()' 시키는 것을 넘어, 모델이 내부에서 어떻게 작동하는지, 왜 그렇게 작동하는지를 깊이 이해하기 위한 필수 과정입니다.

4.3 학습 도구

본 과정은 두 가지 주요 플랫폼을 사용합니다.

- **Edstem:** 강의 슬라이드, 섹션(실습) 자료, 공지사항, 토론 포럼(Q&A)이 이루어지는 메인 허브입니다.
- **Canvas:** 강의 비디오 녹화본, 과제 제출, 공식 일정, 성적 확인에 사용됩니다.

4.4 도움 받는 방법 (How to get help)

문제가 생겼을 때 다음 순서로 도움을 요청하세요.

1. **Edstem (토론 포럼):** 가장 빠른 방법. 동료 학생이나 TF가 답변해줍니다. (개인적인 내용 제외)
2. **오피스 아워 (Office Hours):** 개념 이해나 과제에 대한 심층적인 도움이 필요할 때 가장 좋은 방법입니다.
3. **과정 헬프라인 (Email):** 수강 변경 등 개인적인 행정 문의.
4. **교수진 (Email):** 매우 사적인 문제나 민감한 사안.

5 평가 및 주요 정책

5.1 5대 평가 요소 및 비중

학생 평가는 5가지 요소를 합산하여 이루어집니다.

Table 1: CS109A 평가 요소 및 비중

평가 요소	비중	세부 내용
숙제 (Homework)	30%	HW0 (1%) + HW 1-5 (29%). 2인 1조(Pair) 작업 권장.
섹션 퀴즈	10%	섹션(실습) 시간 중 실시하는 30분 분량의 퀴즈 2회.
중간고사 (Midterm)	18%	섹션 시간 중 치르는 개념 파트 + 별도의 코딩 파트(Take-home)로 구성.
기말고사 (Final Exam)	22%	3시간 동안 지정된 좌석에서 치르는 시험. (개념 + 코딩)
프로젝트 (Project)	20%	3-5인 1조의 그룹 프로젝트. 공개된(public) 데이터를 활용하여 주제 제안 가능.

5.2 숙제(Homework) 상세

- **HW 0:** 과정 시작 시 배포되며, 선수과목(Prerequisites) 충족 여부를 스스로 진단하기 위한 목적입니다. (성실히 제출 시 1)
- **HW 1-5:** 2인 1조(Pair)로 제출하는 것을 적극 권장합니다.
- 제출 기한: (별도 공지 없는 한) 매주 화요일 오후 10시.

5.3 선수과목 (Prerequisites) 진단

HW0는 다음 3가지 영역에 대한 준비 상태를 점검합니다.

- **파이썬 (Python) 코딩:**
 - (필수) 프로그래밍 경험이 전혀 없다면 이 과정을 수강하기 매우 어렵습니다.
 - (괜찮음) 파이썬에 능숙하지 않더라도, 다른 언어(예: CS50 수강) 경험이 있다면 따라올 수 있습니다.
- **기초 수학 (Calculus):** 기본적인 미적분 지식이 필요합니다. (예: Math 1B 수준)
- **기초 통계/확률 (Stats/Probability):**
 - Stat 104 (데이터 분석 중심) 수강생이 가장 이상적입니다.
 - Stat 110 (수학적 확률론 중심)도 좋지만, 실제 데이터를 다루는 부분은 본 과정에서 새로 배워야 할 수 있습니다.
- **결론:** 수학/통계 지식의 공백은 TF와 교수진의 도움으로 메울 수 있지만, 코딩 경험의 부재는 심각한 장애물이 될 수 있습니다.

5.4 출석 및 지각 정책

주의사항

CS109A 출석 정책은 매우 중요하며 성적에 직접적인 영향을 미칩니다.

- 출석은 필수입니다 (On-campus 학생): 모든 강의와 세션은 출석이 요구됩니다.
- 성적 등급 자격 (Qualification): 출석률은 받을 수 있는 최고 성적을 제한하는 "자격" 요건입니다.
(이 출석률을 만족한다고 해당 성적을 보장하는 것은 아닙니다.)
 - A 등급을 받으려면 → 최소 **66%** (2/3) 출석 필요
 - A- 등급을 받으려면 → 최소 **50%** (1/2) 출석 필요
 - B+ 등급을 받으려면 → 최소 **33%** (1/3) 출석 필요
- 지각 제출권 (Late Days) 획득:
 - 출석(강의 또는 세션) 4회당 1개의 지각 제출권(Late Day)을 획득합니다.
 - (예: 24회 출석 시 6개의 Late Day 획득)
- DCE 학생은 출석 확인이 어려운 점을 감안하여 자동으로 **4개의 Late Day**가 부여됩니다.
- Late Day 사용:
 - 획득한 Late Day는 숙제(HW) 제출 시 사용할 수 있습니다.
 - 한 숙제당 최대 **2개의 Late Day**만 사용할 수 있습니다.

6 실습 (Section 1): 웹 스크레이핑 입문

6.1 웹 스크레이핑이란?

웹 스크레이핑 (Web Scraping)은 웹사이트에서 프로그래밍 방식을 통해 자동으로 데이터를 추출하고 수집하는 기술입니다.

첫 번째 실습과 숙제(HW1)는 이 기술을 사용하여 노벨상(Nobel Prize) 웹사이트에서 데이터를 수집하는 것을 목표로 합니다.

6.2 실습 라이브러리

- **requests:** 웹사이트에 접속하여 원본 HTML 코드를 가져오는 라이브러리. (HTTP 요청)
- **BeautifulSoup:** 가져온 HTML 코드를 파이썬이 다루기 쉬운 객체 구조로 변환(Parsing)하고, 원하는 정보를 쉽게 찾도록 도와주는 라이브러리.
- **pandas:** 추출한 데이터를 표(DataFrame) 형태로 정리하고 분석하는 라이브러리.
- **matplotlib:** 데이터를 시각화하는 라이브러리.

6.3 1단계: 웹 스크레이핑 윤리 및 규칙 확인

주의사항

모든 웹사이트가 데이터 수집을 허용하는 것은 아닙니다.

- **robots.txt 확인:** 웹사이트 도메인 뒤에 /robots.txt를 붙여(예: google.com/robots.txt) 어떤 페이지의 수집을 허용/금지하는지 확인해야 합니다.
- **과도한 요청 금지 (Rate Limit):** 서버에 부담을 주지 않도록 짧은 시간에 너무 많은 요청을 보내지 않아야 합니다. (예: "1분에 500회 요청 제한")

6.4 2단계: HTML 기초와 브라우저 '검사' 도구

웹사이트는 HTML(HyperText Markup Language)이라는 언어로 구성됩니다.

- **요소 (Element):** <tag>로 시작하여 </tag>로 끝나는 전체 구조.
- **태그 (Tag):** 요소의 종류를 정의합니다. (예: <h1>(제목), <p>(문단), <a>(링크), <div>(구역))
- **속성 (Attribute):** 태그에 추가 정보를 제공합니다. (예: (링크 주소), <div class="...">(요소의 별명))

▣ 예제:

브라우저 '검사' 도구 활용하기

웹사이트에서 원하는 정보(예: 수상자 이름)가 어떤 태그와 클래스(class)로 구성되어 있는지 확인하는 가장 쉬운 방법입니다.

1. 웹페이지에서 원하는 부분에 마우스 오른쪽 클릭
2. '검사(Inspect)' 메뉴 선택
3. 개발자 도구가 열리면, 왼쪽 상단의 '선택 도구(Picker Tool)' 아이콘(화살표 모양)을 클릭

4. 페이지에서 원하는 요소를 클릭하면, 해당 요소의 HTML 코드가 하이라이트됩니다.

6.5 3단계: requests로 데이터 가져오기

먼저 웹페이지의 HTML 소스 코드를 가져와야 합니다.

```

1 import requests
2
3 # 1. 수집할웹사이트 URL
4 url = "https://www.nobelprize.org/all-nobel-prizes/"
5
6 # 2. HTTP GET 요청보내기
7 response = requests.get(url)
8
9 # 3. 상태코드확인이면 (200 성공)
10 print(f"상태 코드: {response.status_code}")
11
12 # 4. HTML 텍스트내용확인일부만 ()
13 html_text = response.text
14 print(html_text[:200])

```

Listing 1: requests를 사용한 HTML 코드 요청

- **Status Code 200:** 요청이 성공적으로 완료되었음을 의미합니다. (OK)
- **Status Code 404:** 해당 URL을 찾을 수 없음을 의미합니다. (Not Found)

6.6 4단계: BeautifulSoup로 HTML 파싱하기

`response.text`는 다루기 힘든 거대한 문자열입니다. 이를 BeautifulSoup를 이용해 ”수프(soup)” 객체로 만듭니다.

```

1 from bs4 import BeautifulSoup
2
3 # 1. 'html.parser'를 이용해를 html_text soup 객체로변환
4 soup = BeautifulSoup(html_text, 'html.parser')
5
6 # 2. 원하는정보찾기예 (: 페이지제목 )
7 page_title = soup.title.get_text()
8 print(f"페이지 제목: {page_title}")
9
10 # 3. CSS 선택자(Selector)로 특정요소찾기
11 # 예 (: 클래스가 'card-prize인' 모든 div 요소)
12 prize_blocks = soup.select('div.card-prize')
13 print(f"총 수상블록개수 : {len(prize_blocks)}")
14
15 # 4. 첫번째블록에서텍스트만추출공백 ( 제거 )
16 first_block = prize_blocks[0]
17 block_text = first_block.get_text().strip()
18 print(block_text)

```

Listing 2: BeautifulSoup로 HTML 파싱하기

6.7 5단계: 데이터 추출 및 구조화 (노벨상 예제)

여러 개의 수상 블록(prize_blocks)을 순회하며 원하는 정보를 추출하여 리스트와 딕셔너리로 저장합니다.

```

1 import re # 정규표현식 (Regular Expression) 라이브러리
2 from collections import defaultdict, Counter
3
4 # 1. 람다(lambda)를 이용한 간단한 헬퍼 함수 정의
5 get_title = lambda block: block.select_one('h3').get_text().strip()
6 get_year = lambda block: re.search(r'(\d{4})', block.select_one('h3').get_text()
7     ).group(1)
8 get_description = lambda block: block.select_one('blockquote').get_text().
9     strip()
10
11 # 2. 데이터를 저장할 리스트
12 nobel_data = []
13
14 # 3. 모든 수상 블록을 순회 (loop)
15 for block in prize_blocks:
16     try:
17         title = get_title(block)
18         year = get_year(block)
19         description = get_description(block)
20
21         # 4. 딕셔너리 형태로 저장
22         nobel_data.append({
23             'title': title,
24             'year': int(year),
25             'description': description
26         })
27     except Exception as e:
28         print(f"데이터 추출 중 오류 발생 : {e}")
29
30 # 5. 예제() 고유한 수상 분야 찾기
31 unique_titles = set(item['title'] for item in nobel_data)
32 print(f"고유 수상 분야 : {unique_titles}")
33
34 # 6. 예제() 경제학상이 처음 수여된 연도 찾기
35 econ_years = [item['year'] for item in nobel_data if 'Economic Sciences' in
36     item['title']]
37 first_econ_year = min(econ_years)
38 print(f"경제학상 최초 수여 연도 : {first_econ_year}")
39
40 # 7. 예제() 연도별 수상자 수 집계 (defaultdict 사용)
41 winners_per_year = defaultdict(int)

```

```

39 for item in nobel_data:
40     winners_per_year[item['year']] += 1 # 으로 0 자동초기화됨
41 print(f"년 2023 수상자수 : {winners_per_year[2023]}")

```

Listing 3: 반복문을 통한 데이터 추출 및 구조화

6.8 6단계: pandas로 데이터 프레임 변환

스크레이핑한 데이터(딕셔너리 리스트)는 pandas의 DataFrame으로 변환하면 분석하기 매우 용이합니다.

```

1 import pandas as pd
2
3 # 1. 리스트를 데이터프레임으로 변환
4 df = pd.DataFrame(nobel_data)
5
6 # 2. 데이터프레임상위개 5 확인
7 print(df.head())
8
9 # 3. CSV 파일로 저장
10 df.to_csv('nobel_prizes.csv', index=False)

```

Listing 4: pandas DataFrame으로 변환 및 저장

6.9 (심화) 비동기(Async) 스크레이핑

수백, 수천 개의 페이지를 스크레이핑할 때는 한 번에 하나씩 요청하면 매우 느립니다.

비동기(Asynchronous) 프로그래밍 (예: asyncio, httpx 라이브러리)은 여러 개의 요청을 동시에 처리하여 속도를 높이는 고급 기법입니다.

이는 서버의 "Rate Limit"(요청 제한)을 존중하면서도 효율적으로 데이터를 수집하기 위해 사용됩니다. (예: "1초에 5개씩" 또는 "한 번에 10개씩 블어서(batch) 요청")

7 자주 묻는 질문 (FAQ)

- Q: 이 수업을 청강(Audit) 할 수 있나요?
A: 네, 가능합니다. 다만 청강으로 수강한 경우, 나중에 동일 과목을 학점 이수(for grade)로 다시 수강할 수 없습니다.
- Q: 비동기(Asynchronous) 방식(녹화본 시청)으로만 수강할 수 있나요?
A: (대학생) 허용되지 않습니다. (대학원생) 권장하지 않습니다. 출석은 이 수업의 매우 중요한 부분이며, 출석률이 33% 미만일 경우 B+ 이상의 성적을 받을 자격이 박탈됩니다.
- Q: 선수과목이 부족한데 수강할 수 있을까요?
A: HW0를 풀어보고 판단하세요. 수학/통계 지식은 도움을 받아 채울 수 있지만, 프로그래밍(코딩) 경험이 전혀 없다면 수강을 다음 학기로 미루는 것을 강력히 권장합니다.
- Q: 중간고사 기간에 여행 계획이 있습니다. 시험을 일찍 보거나 미룰 수 있나요?
A: 아니요. 중간고사(10/22-24주간)와 기말고사(12/11 예정)는 지정된 날짜에만 치러야 합니다. 일정을 미리 확인하세요.
- Q: 제가 가진 개인 프로젝트 아이디어를 사용해도 되나요?
A: 네, 가능합니다. 단, 데이터가 공개(public)되어 있어야 하며, 3-5명의 그룹 프로젝트로 진행해야 합니다.

December 10, 2025

- 강의명: CS109A: 데이터 과학 입문
- 주차: Lecture 02
- 교수명: Pavlos Protopapas, Kevin Rader, Chris Gumb
- 목적: Lecture 02의 핵심 개념 학습

▣ 핵심 요약

본 문서는 데이터 과학의 핵심 구성요소인 '데이터'가 무엇인지 정의하고, 데이터를 수집, 저장, 분류하는 방법을 다룹니다. 또한, 데이터의 특성을 요약하는 '기술 통계' 방법(평균, 분산 등)과, 데이터에 숨겨진 패턴을 찾는 '시각화'의 중요성(앤스콤 4중주) 및 다양한 시각화 기법(히스토그램, 산점도, 박스 플롯 등)을 설명합니다. 이 자료는 데이터 과학 프로세스의 2, 3단계(수집 및 탐색)의 기초를 다룹니다.

Contents

1	핵심 용어 정리	2
2	핵심 개념 1: 데이터란 무엇인가? (What is Data?)	3
2.1	데이터의 정의	3
2.2	데이터 수집 방법 (어디서 오는가?)	3
2.3	온라인 데이터 수집 3가지 방법	3
3	핵심 개념 2: 데이터의 유형과 구조	5
3.1	데이터 유형 (Data Types)	5
3.2	데이터 저장 구조	5
3.3	정형 데이터 (Tabular Data) 집중 탐구	5
3.4	변수의 유형: 분석의 첫걸음	6
3.5	데이터의 혼란 문제점과 "Tidy Data"	6
4	핵심 개념 3: 탐색적 데이터 분석 (EDA)	8
4.1	모집단 (Population) vs. 표본 (Sample)	8
4.2	표본 추출 편향 (Sampling Bias)	8
5	절차/방법 1: 기술 통계 (Descriptive Statistics)	9

5.1	데이터의 ”중심” 측정 (Measures of Center)	9
5.1.1	평균 (Mean)	9
5.1.2	중앙값 (Median)	9
5.1.3	평균 vs 중앙값: 왜도 (Skewness)	9
5.1.4	최빈값 (Mode)	10
5.2	데이터의 ”퍼짐” 측정 (Measures of Spread)	10
5.2.1	범위 (Range)	10
5.2.2	분산 (Variance)	10
5.2.3	표준편차 (Standard Deviation)	11
6	절차/방법 2: 기본 시각화 (Basic Visualizations)	12
6.1	시각화의 중요성 (앤스콤 4중주)	12
6.2	기본 플롯 유형 비교	13
6.3	3개 이상의 변수 시각화하기	13
7	부록 1: 데이터 시각화의 역사 (Historical Interlude)	15
8	부록 2: 효과적인 시각화 원칙 (Effective Visualization)	16
8.0.1	1. 그래픽 무결성 (Graphical Integrity)	16
8.0.2	2. 단순성 (Keep it simple)	16
8.0.3	3. 올바른 표현 (Use the right display)	16
8.0.4	4. 전략적인 색상 사용 (Use color strategically)	16
8.0.5	5. 청중 이해 (Know your audience)	17
9	학습 체크리스트	18
10	빠르게 훑어보기 (1-Page Summary)	19

1 핵심 용어 정리

데이터 분석을 시작하기 위해 꼭 알아야 할 기본 용어들입니다.

Table 1: 데이터 분석 핵심 용어

용어	쉬운 설명	원어	비고
데이터	관찰을 통해 수집된 사실, 값, 정보의 집합.	Data	단수형은 Datum. (Data는 복수형)
정형 데이터	엑셀 시트처럼 행과 열로 명확히 구조화된 데이터.	Tabular Data	"Tidy Data"라고도 함.
관측치	분석 대상의 개별 단위 (엑셀의 '행').	Observation	예: 한 명의 사람, 한 개의 영화.
변수	측정하려는 특성 (엑셀의 '열').	Variable	예: 나이, 평점. (Feature라고도 함)
모집단	연구 대상이 되는 전체 집단.	Population	예: 이 수업을 듣는 '모든' 학생.
표본	모집단에서 추출한 '일부' 대 표 집합.	Sample	예: 오늘 수업에 '출석한' 학생.
EDA	시각화와 통계를 통해 데이터의 패턴을 탐색하는 과정.	Exploratory Data Analysis	"탐색적 데이터 분석"
[중심 측정]			
평균	모든 값을 더해 개수로 나눈 값. (무게 중심)	Mean (\bar{x})	이상치(Outlier)에 매우 민감함.
중앙값	데이터를 순서대로 나열했을 때 딱 중간에 있는 값.	Median	이상치에 둔감함(Robust).
최빈값	데이터에서 가장 자주 등장하는 값.	Mode	범주형 데이터에서 주로 사용.
[퍼짐 측정]			
분산	데이터가 평균에서 얼마나 멀리 퍼져있는지의 정도.	Variance (s^2)	단위가 원래 단위의 '제곱'이 됨.
표준편차	분산에 제곱근을 씌운 값.	Standard Dev. (s)	원래 데이터와 단위가 동일해 해석이 쉬움.
[시각화]			
히스토그램	수량형 데이터의 '분포'를 막대로 표현.	Histogram	구간(bin) 너비에 따라 모양이 변함.
막대 그래프	범주형 데이터의 '빈도'를 막대로 비교.	Bar Plot	막대 순서를 바꿔도 의미가 통함.
산점도	두 수량형 변수 간의 '관계'를 점으로 표현.	Scatter Plot	방향, 강도, 형태, 이상치를 봄.
박스 플롯	범주형 그룹 간 수량형 데이터의 분포를 '요약' 비교.	Box Plot	중앙값, 사분위수, 이상치를 보여줌.

2 핵심 개념 1: 데이터란 무엇인가? (What is Data?)

데이터 과학(Data Science)은 이름 그대로 '데이터'에서 시작합니다.

2.1 데이터의 정의

- **데이터(Data):** 관찰이나 측정을 통해 얻은 여러 개의 정보 조각들입니다. (복수형)
- **데이터(Datum):** 정보 조각 '하나'를 의미합니다. (단수형)

과거에는 데이터가 주로 숫자(Numeric)였지만, 현대에는 텍스트, 이미지, 소리 등 모든 것이 데이터가 될 수 있습니다.

2.2 데이터 수집 방법 (어디서 오는가?)

데이터는 크게 세 가지 경로로 얻을 수 있습니다.

1. 내부 소스 (Internal Sources):

- 조직이나 개인이 직접 수집한 1차 데이터입니다.
- 예: 과학 실험 결과, 임상 시험 데이터, 회사 내부의 판매 기록.

2. 기존 외부 소스 (Existing External Sources):

- 이미 누군가 수집/가공하여 공개한 데이터입니다.
- 예: 정부 공공 데이터 포털, Kaggle 데이터셋, 스포츠 기록 사이트.

3. 수집이 필요한 외부 소스 (External Sources Requiring Collection):

- 외부에 존재하지만, 가져오려면 별도의 노력이 필요한 데이터입니다.
- 이 강의에서 주목하는 방식이며, 주로 온라인 데이터를 의미합니다.

2.3 온라인 데이터 수집 3가지 방법

온라인에서 데이터를 가져오는 대표적인 3가지 기술입니다.

1. **API (Application Programming Interface)** • **개념:** 회사가 외부 사용자가 자신의 데이터나 서비스에 "합법적으로" 접근할 수 있도록 열어둔 '공식 창구'입니다.
 - **특징:** 보통 사용량 제한이 있거나 유료입니다. 안정적이고 정확한 데이터를 제공받습니다. (예: 구글 지도 API, 스포티파이 API)
 - **필요 기술:** 파이썬(Python)과 각 API의 사용 설명서(Dictionary)를 읽는 능력.
2. **RSS (Rich Site Summary)** • **개념:** 블로그나 뉴스 사이트처럼 '자주 업데이트되는' 콘텐츠를 요약하여 스트림(Stream) 형태로 제공하는 규격입니다.
 - **특징:** 무료이며, 주로 새로운 게시물의 제목, 요약, 링크를 받아볼 때 사용됩니다.
3. **웹 스크래핑 (Web Scraping)** • **개념:** 웹사이트의 HTML 코드에서 직접 필요한 정보를 '추출'하는 기술입니다.
 - **특징:** API가 없거나 유료 API를 우회하고 싶을 때 사용됩니다. (예: 위키피디아의 표(table) 정보를 긁어오는 것)

주의사항

웹 스크래핑의 윤리적/법적 문제

웹 스크래핑은 강력하지만 매우 조심해야 하는 기술입니다.

- **서비스 약관(Terms of Service) 위반:** 많은 웹사이트가 스크래핑을 명시적으로 금지합니다.
- **개인정보 침해:** 사용자의 비공개 정보를 수집하면 안 됩니다.
- **서버 부하:** 과도한 스크래핑은 대상 웹사이트의 서버를 마비시킬 수 있습니다 (DoS 공격과 유사).
- **해악(Harm):** 수집한 데이터를 통해 사생활을 침해하거나 불법적인 용도로 사용해서는 안 됩니다.
항상 데이터를 수집하기 전에 ”이 데이터를 사용해도 되는가?”를 먼저 질문해야 합니다.

3 핵심 개념 2: 데이터의 유형과 구조

데이터를 수집했다면, 그 형태와 유형을 파악해야 합니다.

3.1 데이터 유형 (Data Types)

- **원자적 유형 (Atomic Types):** 더 이상 쪼갤 수 없는 기본 단위입니다.
 - **수량형 (Numeric):** 정수(Integers, 예: 109), 실수(Floats, 예: 3.14)
 - **부울형 (Boolean):** 참/거짓 (True/False, Yes/No, 1/0)
 - **문자열 (Strings):** 텍스트 (예: "Hello")
- **복합 유형 (Compound Types):** 원자적 유형들이 모여 구성됩니다.
 - **리스트 (Lists):** 순서가 있는 값의 모음 (예: [1, 2, 3])
 - **사전 (Dictionaries):** '키 (Key)'와 '값 (Value)'이 짹을 이룬 모음 (예: {"name": "Kevin"})

3.2 데이터 저장 구조

정형 데이터 (Tabular Data) 가장 중요합니다. 엑셀 시트나 CSV 파일처럼 2차원 테이블(표) 형태입니다. 대부분의 데이터 분석 패키지(예: Pandas)는 이 형태를 기본으로 가정합니다.

반정형 데이터 (Semistructured Data) JSON, XML처럼 키-값 쌍으로 이루어져 있지만, 정형 데이터처럼 엄격한 행/열 구조를 따르지 않을 수 있습니다.

비정형 데이터 (Unstructured Data) 텍스트 문서, 이미지, 오디오 파일 등 구조가 없는 데이터입니다.

3.3 정형 데이터 (Tabular Data) 집중 탐구

정형 데이터는 데이터 분석의 표준입니다.

- **관측치 (Observations):** 표의 행 (Row). 분석하려는 개별 대상 하나하나를 의미합니다. (예: 영화 1개, 학생 1명)
- **변수 (Variables):** 표의 열 (Column). 관측치에서 측정한 특정 속성입니다. (예: 영화 평점, 학생 나이)

```

1 # imbd_top_1000.csv 파일을 읽어들임
2 imbd = pd.read_csv('imbd_top_1000.csv')
3 # 처음 5(head)을 출력
4 imbd.head()

```

Listing 1: Pandas를 이용한 정형 데이터(CSV) 로딩 예시

[lst:pandas의 출력 결과: IMDB 영화 목록 표]

각 행 (Row)은 영화 1개(관측치)를 나타내며, 각 열 (Column)은 Series_Title, Released_Year, IMDB_Rating 등(변수)을 나타냅니다.

3.4 변수의 유형: 분석의 첫걸음

주의사항

왜 변수 유형을 구분해야 하나요?

변수의 유형에 따라 사용할 수 있는 요약 방법(통계)과 시각화가 완전히 달라지기 때문입니다. 예를 들어, '키'는 평균을 낼 수 있지만 '좋아하는 색깔'은 평균을 낼 수 없습니다.

변수는 크게 두 가지로 나뉩니다.

1. 수량형 변수 (Quantitative / Numeric Variable)

- 숫자로 측정되며, 산술 연산(+, -)이 의미가 있습니다.
- 이산형 (Discrete):** 값이 정수처럼 딱딱 떨어져 셀 수 있습니다. (예: 형제자매 수, 주사위 눈금)
- 연속형 (Continuous):** 값이 특정 범위 내에서 무한히 많은 값을 가질 수 있습니다. (예: 키, 몸무게, 온도)

2. 범주형 변수 (Categorical Variable)

- 값이 몇 개의 그룹이나 범주로 나뉩니다.
- 순서형 (Ordinal):** 범주 간에 자연스러운 순서가 있습니다. (예: 학점 A, B, C / 만족도 '높음', '중간', '낮음')
- 명목형 (Nominal):** 범주 간에 순서가 없습니다. (예: 혈액형 A, B, O / 좋아하는 애완동물 '개', '고양이', '쥐')

3.5 데이터의 혼란 문제점과 "Tidy Data"

현실의 데이터는 깨끗하지 않습니다.

- 결측치 (Missing values):** 값이 비어있습니다. (이 값을 버릴까요? 아니면 추측해서 채울까요?)
- 오입력값 (Wrong values):** 잘못된 값이 입력되었습니다. (예: 나이 200세)
- 형식 불일치 (Format mismatch):** 두 데이터를 합치려는데 형식이 다릅니다.
- 지저분한 데이터 (Messy Data):** 데이터가 분석하기 어려운 형태로 되어 있습니다.

□ 예제: 지저분한(Messy) 데이터 변환하기

다음은 주말 농산물 배송 횟수를 기록한 "지저분한" 표입니다.

Before: (지저분한 형식)

왜 이 형식이 나쁜가요?

- '관측치 1개 = 행 1개' 원칙이 깨졌습니다. 'Morning' 행 하나에 3개의 관측치(금요일 아침, 토요일 아침, 일요일 아침)가 들어있습니다.
- 'Friday'는 변수 이름이어야 하는데, '값'처럼 취급되고 있습니다.
- 이 상태로는 "평균 배송 횟수는?" 또는 "요일별 배송 횟수 합계는?"을 계산하기 어렵습니다.

After: (정형/Tidy 형식) 이 데이터를 분석하기 쉬운 '정형(Tabular)' 또는 'Tidy' 형식으로 바꾸면 다음과 같습니다.

왜 이 형식이 좋은가요?

- 1 관측치 = 1 행: '금요일 아침'이라는 관측치 하나가 행 하나를 차지합니다.
- 1 변수 = 1 열: '시간', '요일', '횟수'라는 명확한 변수(열)가 생겼습니다.
- 이제 Pandas 같은 도구로 'Number' 열의 평균을 구하거나, 'Day' 별로 그룹화하여 합계를 구하는 것이 매우 쉬워집니다.

4 핵심 개념 3: 탐색적 데이터 분석 (EDA)

탐색적 데이터 분석 (Exploratory Data Analysis, EDA)은 수집한 데이터를 본격적으로 모델링하기 전에, 시각화나 간단한 통계 기법을 통해 데이터의 구조와 패턴을 파악하고, 이상치나 잠재적인 문제점을 발견하는 과정을 말합니다.

EDA는 ”데이터와 친해지는 과정”이며, 데이터 과학 프로세스의 3단계에 해당합니다.

4.1 모집단 (Population) vs. 표본 (Sample)

- **모집단 (Population):** 내가 궁극적으로 알고 싶은 대상 '전체'입니다. (예: 하버드의 모든 학생)
- **표본 (Sample):** 모집단 전체를 조사하기는 불가능하므로, 그중 '일부'를 뽑아서 조사한 것입니다. (예: 오늘 CS109A 수업에 온 학생)

우리는 '표본'을 분석해서 '모집단'의 특성을 추측합니다. 이때 가장 중요한 것은 표본이 모집단을 잘 대표해야 한다는 것입니다.

4.2 표본 추출 편향 (Sampling Bias)

표본이 모집단을 잘 대표하지 못할 때 '편향(Bias)'이 발생했다고 말합니다.

선택 편향 (Selection Bias): 특정 하위 그룹이 다른 그룹보다 표본으로 더 잘 선택되는 경우.

무응답/자발적 참여 편향 (Non-response/Volunteer Bias): 응답하기 쉬운 대상만 응답하거나(예: 수업에 출석한 학생), 특정 주제에 열성적인 사람들(예: 앱 얼리 어답터)만 자발적으로 참여하는 경우.

□ 예제: 잘못된 표본 추출 예시

사례 1: 수업 출석률

- **목표:** CS109A 전체 학생의 평균 만족도 조사.
- **표본:** 오늘 수업에 '출석한' 학생들.
- **문제점:** 수업에 결석한 학생들(어쩌면 만족도가 매우 낮아서 안 온 학생들)의 의견이 반영되지 않습니다. (무응답 편향) 이 표본의 만족도 점수는 실제 모집단(전체 학생)의 점수보다 높게 나올 가능성이 큽니다.

사례 2: 신규 앱 기능 테스트

- **목표:** 새로운 앱 기능이 모든 사용자에게 효과가 있는지 테스트.
- **표본:** 신규 기능을 자발적으로 신청한 '얼리 어답터' 그룹.
- **문제점:** 얼리 어답터들은 원래 새로운 기능에 호의적이고 IT 활용도가 높은 집단입니다. (자발적 참여 편향) 이들이 새 기능을 좋아한다고 해서, 변화를 싫어하는 일반 대중(모집단)도 좋아할 것이라고 일반화할 수 없습니다.

5 절차/방법 1: 기술 통계 (Descriptive Statistics)

기술 통계는 수집한 데이터(표본)의 특성을 몇 개의 숫자로 요약하는 방법입니다. 주로 데이터의 '중심'이 어디인지, '얼마나 펴져있는지'를 봅니다.

5.1 데이터의 "중심" 측정 (Measures of Center)

5.1.1 평균 (Mean)

- 정의: 모든 값을 더한 뒤, 값의 개수(n)로 나눈 값.
- 직관: 데이터 분포의 "무게 중심" 또는 "균형점".
- 공식: $\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i = \frac{x_1 + x_2 + \dots + x_n}{n}$
- 단점: 이상치(Outlier)에 매우 민감합니다.
- 예시:
 - 데이터: [1, 2, 3, 4, 5] → 평균: $(1 + 2 + 3 + 4 + 5)/5 = 3$
 - 데이터: [1, 2, 3, 4, 100] → 평균: $(1 + 2 + 3 + 4 + 100)/5 = 22$
 - '100'이라는 극단값 하나 때문에 무게 중심이 3에서 22로 확 이동했습니다.

5.1.2 중앙값 (Median)

- 정의: 데이터를 크기순으로 정렬했을 때, 정확히 '가운데'에 위치한 값.
 - 데이터 개수(n)가 홀수: 가운데 1개 값.
 - 데이터 개수(n)가 짝수: 가운데 2개 값의 평균.
- 장점: 이상치에 거의 영향을 받지 않습니다. (Robust)
- 예시: (위의 예시를 다시 사용)
 - 데이터 (정렬됨): [1, 2, 3, 4, 5] → 중앙값: 3
 - 데이터 (정렬됨): [1, 2, 3, 4, 100] → 중앙값: 3
 - '100'이 아니라 '1000'이 되어도 중앙값은 여전히 3입니다.

5.1.3 평균 vs 중앙값: 왜도 (Skewness)

평균과 중앙값의 차이는 데이터 분포의 '비대칭성(왜도)'을 알려줍니다.

- 대칭 분포 (Symmetric): 평균 \approx 중앙값 (예: 정규분포)
- 오른쪽 꼬리 분포 (Right-skewed): 평균 > 중앙값
 - 소수의 매우 큰 값(outlier)이 평균을 오른쪽으로 끌어당깁니다.
 - 예: 개인 소득 분포. (대부분은 중간 소득, 소수의 재벌이 평균을 높임)
- 왼쪽 꼬리 분포 (Left-skewed): 평균 < 중앙값
 - 소수의 매우 작은 값이 평균을 왼쪽으로 끌어당깁니다.

[이미지 플레이스홀더: 오른쪽 꼬리 분포(Right-skewed) 그래프]

대부분의 데이터가 왼쪽에 몰려있고, 긴 꼬리가 오른쪽으로 뻗어 있음.
(중앙값)이 (평균)보다 왼쪽에 위치함.

5.1.4 최빈값 (Mode)

- 정의: 데이터에서 가장 '자주' 등장하는 값.
- 용도: 범주형 데이터의 중심을 나타낼 때 사용합니다.
- 예시: (선후하는 애완동물) ["개", "고양이", "개", "쥐", "고양이", "개"] → 최빈값: "개"
- 범주형 데이터는 순서가 없으므로 평균이나 중앙값을 계산하는 것이 의미가 없습니다.

□ 예제: 어느 것이 더 빠를까? (Mean vs Median)

질문: 수십억 개의 데이터가 있을 때, 평균과 중앙값 중 무엇이 더 빠를까요?

답변: 평균이 훨씬 빠릅니다.

- 평균 ($O(n)$): 데이터를 한 번만 훑으면서 합계와 개수만 알면 됩니다.
- 중앙값 ($O(n \log n)$): '중간' 값을 찾으려면, 먼저 모든 데이터를 정렬(Sorting)해야 합니다. 정렬은 매우 비싼 연산입니다.

이러한 연산 속도 차이도 평균이 자주 사용되는 이유 중 하나입니다.

5.2 데이터의 "퍼짐" 측정 (Measures of Spread)

데이터가 중심에 밀집해 있는지, 아니면 넓게 퍼져 있는지 측정합니다.

5.2.1 범위 (Range)

- 정의: 최대값(Max) - 최소값(Min).
- 단점: 데이터 양 끝의 극단값 2개에만 의존하므로, 분포 전체의 퍼짐을 잘 설명하지 못합니다.

5.2.2 분산 (Variance)

- 정의: 데이터가 평균(\bar{x})으로부터 '평균적으로 얼마나 멀리 떨어져 있는지'를 나타내는 값.
- 계산(직관):
 - 각 데이터가 평균과 얼마나 차이 나는지(편차: $x_i - \bar{x}$) 계산.
 - 편차를 '제곱' 함 (음수를 없애고, 멀리 떨어진 값에 더 큰 가중치를 주기 위해).
 - 제곱한 값들의 평균을 냅.
- 공식 (표본 분산): $s^2 = \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2$
- 단점: 값을 '제곱' 했기 때문에, 원래 데이터와 단위가 맞지 않습니다. (예: 키(cm)의 분산은 cm^2 가 됨)

□ 예제: Q&A: 왜 n 이 아니라 $n - 1$ 로 나누나요?

질문: 평균은 n 으로 나누는데, 왜 분산은 $n - 1$ 로 나누나요?

답변(직관): "퍼짐(spread)"을 측정하려면 최소 몇 개의 데이터가 필요할까요? 만약 데이터가 1개 ([5])만 있다면, 이 데이터가 얼마나 퍼져 있는지 말할 수 없습니다. 분산 공식의 분모에 $n = 1$ 을 넣어보면 $\frac{1}{1-1} = \frac{1}{0}$ 이 되어 정의되지 않습니다. 즉, 이 공식은 "분산을 계산하려면 최소 2개 이상의 데이터가 필요하다"는 직관을 반영하고 있습니다.

답변(기술): 우리가 가진 '표본'의 분산(s^2)을 가지고 '모집단'의 분산(σ^2)을 추정할 때, n 으로 나누면 실제보다 분산이 작게 추정되는 경향(편향)이 생깁니다. $n - 1$ 로 나누면 이 편향이 보정되어 모집단의

분산을 더 잘 추정할 수 있습니다. (통계 용어로 '자유도(Degrees of Freedom)'를 고려한 '불편추정량(unbiased estimator)'이라고 합니다.)

5.2.3 표준편차 (Standard Deviation)

- 정의: 분산에 제곱근($\sqrt{\cdot}$)을 씌운 값.
- 공식: $s = \sqrt{s^2} = \sqrt{\frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2}$
- 장점: 분산의 "단위 문제"를 해결합니다. 원래 데이터와 단위가 동일해져 해석이 매우 직관적입니다.
- 직관: "데이터가 평균으로부터 '평균적으로' 이 정도(s) 떨어져 있다."

6 절차/방법 2: 기본 시각화 (Basic Visualizations)

EDA의 꽃은 시각화입니다. 숫자는 우리를 속일 수 있지만, 그림은 그렇지 않습니다.

6.1 시각화의 중요성 (앤스콤 4중주)

주의사항

앤스콤의 4중주 (Anscombe's Quartet)는 ”왜 통계 요약치만 보면 안 되는지”를 보여주는 고전적인 예시입니다.

여기 4개의 서로 다른 (X, Y) 데이터셋이 있습니다.

Dataset I		Dataset II		Dataset III		Dataset IV	
X	Y	X	Y	X	Y	X	Y
10.0	8.04	10.0	9.14	10.0	7.46	8.0	6.58
8.0	6.95	8.0	8.14	8.0	6.77	8.0	5.76
13.0	7.58	13.0	8.74	13.0	12.74	8.0	7.71
9.0	8.81	9.0	8.77	9.0	7.11	8.0	8.84
11.0	8.33	11.0	9.26	11.0	7.81	8.0	8.47
14.0	9.96	14.0	8.10	14.0	8.84	8.0	7.04
6.0	7.24	6.0	6.13	6.0	6.08	8.0	5.25
4.0	4.26	4.0	3.10	4.0	5.39	19.0	12.50
12.0	10.84	12.0	9.13	12.0	8.15	8.0	5.56
7.0	4.82	7.0	7.26	7.0	6.42	8.0	7.91
5.0	5.68	5.0	4.74	5.0	5.73	8.0	6.89
평균/분산		평균/분산		평균/분산		평균/분산	
X Avg: 9.0		X Avg: 9.0		X Avg: 9.0		X Avg: 9.0	
Y Avg: 7.50		Y Avg: 7.50		Y Avg: 7.50		Y Avg: 7.50	
X Var: 11.0		X Var: 11.0		X Var: 11.0		X Var: 11.0	
Y Var: 4.12		Y Var: 4.12		Y Var: 4.12		Y Var: 4.12	
상관계수: 0.816		상관계수: 0.816		상관계수: 0.816		상관계수: 0.816	

놀랍게도, 이 4개 데이터셋의 X/Y 평균, X/Y 분산, 상관계수가 모두 동일합니다. 숫자만 보면 이 4개 셋은 ”똑같은” 데이터처럼 보입니다.

하지만 시각화하면 진실이 드러납니다.

[이미지 플레이스홀더: 앤스콤 4중주 산점도]

Dataset I: 점들이 완만한 선형 관계를 보임 (정상)

Dataset II: 점들이 위로 볼록한 2차 곡선(포물선) 모양을 보임 (비선형)

Dataset III: 거의 완벽한 직선 위에 있으나, Y축 방향의 이상치 1개가 존재함

Dataset IV: 모든 점이 $X=8$ 에 수직으로 있으나, X축 방향의 이상치 1개가 존재함

교훈: 절대 요약 통계치만 믿지 말고, 항상 데이터를 시각화해야 합니다.

6.2 기본 풀롯 유형 비교

시각화는 ”내가 무엇을 보고 싶은가?”에 따라 종류가 나뉩니다.

Table 2: 지저분한 데이터 예시 (*Messy Data*)

	Friday	Saturday	Sunday
Morning	15	158	10
Afternoon	2	90	20
Evening	55	12	45

- 히스토그램 vs. 막대 그래프: 둘 다 막대를 사용하지만, 히스토그램은 수량형 변수를 ’구간(bin)’으로 나눠 그리고 (막대들이 붙어 있음), 막대 그래프는 범주형 변수의 ’범주’별로 그립니다 (막대들이 떨어져 있음).
- 파이 차트 (Pie Chart)는 왜 별로 일까요? 인간의 눈은 ’각도’의 미세한 차이를 ’길이’의 차이보다 훨씬 못 알아봅니다. 비슷한 비율을 비교할 때는 파이 차트보다 막대 그래프가 훨씬 효과적입니다.

6.3 3개 이상의 변수 시각화하기

3개 이상의 고차원 데이터를 2D 화면에 표현하는 것은 어렵습니다.

- 나쁜 예: 3D 산점도 (3D Scatter Plot) 3개의 수량형 변수를 X, Y, Z축에 매핑하는 것은 그럴듯해 보이지만, 2D 모니터에서는 깊이감이 왜곡되어 ”산점도 구름(scatter cloud)”처럼 보일 뿐, 관계 파악이 거의 불가능합니다.
- 좋은 예: 미적 매핑 (Aesthetic Mapping) 활용 X축과 Y축 외에, 색상(Color), 크기(Size), 모양(Shape), 애니메이션(Animation) 등 추가적인 시각 요소를 사용하여 변수를 표현합니다.

□ 예제: 사례 연구: 갭마인더(Gapminder)의 5변수 시각화

한스 로슬링의 ”Wealth & Health of Nations” 시각화는 5개의 변수를 하나의 차트에 훌륭하게 녹여 냈습니다.

[이미지 플레이스홀더: 갭마인더 차트 (1950년 스냅샷)]

X축: 소득, Y축: 기대 수명. 점들이 분포해 있음.

이 차트는 다음 5가지 변수를 동시에 보여줍니다.

- 변수 1 (수량형): 1인당 소득 → X축 위치
- 변수 2 (수량형): 기대 수명 → Y축 위치
- 변수 3 (수량형): 국가 인구 수 → 원의 크기 (Size)

- **변수 4 (범주형):** 대륙 (아시아, 유럽...) → 원의 색상 (Color)
 - **변수 5 (시간형):** 연도 (1800 2020) → 애니메이션 (Animation)
- 매핑 전략:
- 수량형 변수(인구) → 크기: 크고 작음으로 양을 표현하기 좋음.
 - 범주형 변수(대륙) → 색상: 그룹을 구분하기 좋음.

7 부록 1: 데이터 시각화의 역사 (Historical Interlude)

데이터 시각화는 최근 기술이 아닌, 오래된 데이터 과학의 한 분야입니다.

존 스노 (John Snow, 1854) • 시각화: 런던 콜레라 발병 지도

- 내용: 콜레라 사망자 발생 위치를 지도에 점(dot)으로 찍었습니다.
- 결과: 특정 '펌프'(Broad Street Pump) 주변에 사망자가 밀집된 것을 시각적으로 확인하고, 펌프를 폐쇄하여 전염병의 원인이 '오염된 물'임을 증명했습니다.

플로렌스 나이팅게일 (Florence Nightingale, 1858) • 시각화: 로즈 차트 (Rose Chart / Coxcomb)

- 내용: 크림 전쟁 당시 사망 원인을 월별로 시각화했습니다. (파란색: 예방 가능한 질병, 빨간색: 부상, 검은색: 기타)
- 결과: 전투로 인한 사망(빨간색)보다, 열악한 위생으로 인한 질병 사망(파란색)이 압도적으로 많음을 보여주어 병원 위생 개혁을 이끌어냈습니다.

샤를 미나르 (Charles Minard, 1869) • 시각화: 나폴레옹의 러시아 원정 지도

- 내용: 단 하나의 차트에 나폴레옹 군대의 규모(선의 굵기), 이동 경로(지리), 방향(진격/후퇴), 시간, 그리고 후퇴 시의 기온 변화(하단 그래프)를 모두 담았습니다.
- 결과: 42만 대군이 모스크바로 진격했다가 1만 명만 돌아오는 과정을 처참하게 보여주는, 데이터 시각화 역사상 최고의 걸작 중 하나로 꼽힙니다.

[이미지 플레이스홀더: 미나르의 나폴레옹 행군도]

8 부록 2: 효과적인 시각화 원칙 (Effective Visualization)

좋은 시각화를 만들기 위한 5가지 원칙입니다.

8.0.1 1. 그래픽 무결성 (Graphical Integrity)

”데이터로 거짓말을 하지 말아야 합니다.”

□ 예제: 잘못된 예: 2020년 미국 대선 지도

- **지리적 면적 지도 (A):** 각 '카운티(County)'의 면적을 기준으로 승리한 정당(빨간색/파란색)을 칠합니다. → 결과: 미국 전역이 빨갛게 보입니다.
- **인구 기반 지도 (B):** 각 카운티의 '인구 수'에 비례하여 원의 크기를 조정한 점(dot) 지도를 만듭니다. → 결과: 인구가 밀집된 해안가와 도시에 파란색 점이 집중되고, 인구가 적은 중부 내륙에 빨간색 점이 흩어져 보입니다.
- **결론:** (A) 지도는 땅이 넓지만 인구가 적은 지역을 과대평가하여 ”미국 대부분이 빨간색을 지지 한다”는 잘못된 인상을 줍니다. (B) 지도가 실제 득표 수에 더 가까운 '무결성'을 가집니다.

8.0.2 2. 단순성 (Keep it simple)

”불필요한 장식을 피해야 합니다. (차트 정크 금지)”

- **차트 정크(Chart Junk):** 데이터 이해에 도움이 되지 않는 모든 시각적 요소를 말합니다.
- **나쁜 예:** 3D 효과가 들어간 막대 그래프, 혼란한 배경색, 의미 없는 그림자, 지나치게 복잡한 범례.
- **좋은 예:** 데이터 잉크 비율(Data-Ink Ratio)을 높여, 꼭 필요한 선과 점, 텍스트만 남깁니다.

8.0.3 3. 올바른 표현 (Use the right display)

인간의 뇌가 정보를 더 효율적으로 처리하는 시각적 수단이 있습니다.

[이미지 플레이스홀더: 시각적 표현의 효율성 계층]
 (가장 효율적 / 정량적) → 1. 위치 (**Position**) (예: 산점도)
 ↓→ 2. 길이 (**Length**) (예: 막대 그래프)
 ↓→ 3. 기울기 (**Slope**)
 ↓→ 4. 각도 (**Angle**) (예: 파이 차트)
 ↓→ 5. 면적 (**Area**) (예: 버블 차트)
 ↓→ 6. 강도 (**Intensity**) / 색상 (**Color**)
 (가장 비효율적 / 범주형) → 7. 모양 (**Shape**)

교훈: 같은 데이터라도 '면적'이나 '각도'로 표현하는 것보다, '위치'나 '길이'로 표현하는 것이 훨씬 더 정확한 비교를 가능하게 합니다. (이것이 파이 차트보다 막대 그래프가 나은 이유입니다.)

8.0.4 4. 전략적인 색상 사용 (Use color strategically)

색상은 강력하지만, 잘못 사용하면 혼란을 줍니다.

- **정성적 (Qualitative):** 범주를 구분할 때. (예: 대류별 색상) 5~8개 이하의 색상 사용을 권장합니다.

- **순차적 (Sequential):** 값이 낮음에서 높음으로 갈 때. (예: 연한 녹색 → 진한 녹색)
- **발산형 (Diverging):** 값이 '0'이나 '평균'을 기준으로 양쪽으로 갈라질 때. (예: 파란색 ← 흰색 → 빨간색)
- **주의 1:** 무지개색(Rainbow Colormap) 금지! 무지개색은 순서가 명확하지 않고(노란색이 녹색보다 높은가?), 특정 부분이 불필요하게 강조됩니다.
- **주의 2:** 색맹/색약 고려 (Color Blindness) 인구의 상당수가 적록색약입니다. 빨간색과 녹색을 동시에 사용한 비교는 피해야 합니다.

8.0.5 5. 청중 이해 (Know your audience)

시각화의 목적이 무엇인지, 청중이 무엇을 알고 싶어 하는지 알아야 합니다.

- **탐색적(Exploratory):** 스스로 데이터를 탐색하기 위한 (중립적인) 시각화. (예: 내부용 대시보드)
- **설명적(Explanatory):** 청중에게 특정 메시지나 주장을 전달하기 위한 (의견이 담긴) 시각화. (예: 신문 기사의 ”이라크의 피의 대가” 그래프)

9 학습 체크리스트

이 강의를 올바르게 이해했는지 다음 질문에 답해보세요.

데이터(Data)와 데이터(Datum)의 차이를 설명할 수 있는가?

API, RSS, 웹 스크래핑의 차이점과 스크래핑 시 윤리적 문제점을 아는가?

'정형 데이터(Tidy Data)'의 3가지 원칙 (1행=1관측치, 1열=1변수)을 아는가?

수량형 변수(이산형/연속형)와 범주형 변수(순서형/명목형)를 구분할 수 있는가?

모집단과 표본의 차이를 알고, '표본 편향'의 예시를 2가지 들 수 있는가?

평균과 중앙값의 차이를 설명하고, '오른쪽 꼬리 분포'에서 둘의 대소 관계(평균 > 중앙값)를 아는가?

분산(s^2) 대신 표준편차(s)를 주로 사용하는 이유(단위 문제)를 설명할 수 있는가?

분산을 계산할 때 n 이 아닌 $n - 1$ 로 나누는 직관적인 이유를 설명할 수 있는가?

앤스컴 4중주(Anscombe's Quartet)가 주는 교훈("항상 시각화하라")을 아는가?

히스토그램과 막대 그래프의 차이점(수량형 vs. 범주형)을 아는가?

파이 차트보다 막대 그래프가 권장되는 이유(각도 vs. 길이)를 아는가?

3개 이상의 변수를 시각화할 때 '미적 매핑'(색상, 크기 등)을 활용하는 법을 아는가?

'차트 정크'를 피하고, 색상을 전략적으로 사용해야 함을 이해했는가?

10 빠르게 훑어보기 (1-Page Summary)

1. 데이터 수집 (Getting Data)

- **API:** 공식적이고 안정적인 창구 (유료/제한 있음)
- **RSS:** 블로그/뉴스 스트림 (무료, 요약본)
- **웹 스크래핑:** HTML에서 직접 추출 (강력하지만 법적/윤리적 위험)

2. 데이터 구조 (Data Structure)

정형 데이터 (Tidy Data)가 목표!

- 1 행 = 1 관측치 (Observation)
- 1 열 = 1 변수 (Variable)
- 1 테이블 = 1 종류의 데이터

지저분한 (Messy) 데이터는 이 원칙에 맞게 변형 (*Tidying*)해야 함!

3. 변수 유형 (Variable Types) - (중요!)

- **수량형 (Quantitative):** 숫자로 연산 가능.
 - **이산형 (Discrete):** 셀 수 있음 (예: 형제 수)
 - **연속형 (Continuous):** 측정 함 (예: 키)
- **범주형 (Categorical):** 그룹으로 구분.
 - **명목형 (Nominal):** 순서 없음 (예: 애완동물 종류)
 - **순서형 (Ordinal):** 순서 있음 (예: 학점)

4. 기술 통계 (Descriptive Statistics)

중심 (Center): • 평균 (Mean): 무게 중심. (이상치에 민감)

- 중앙값 (Median): 순서상 중앙. (이상치에 둔감)
- 최빈값 (Mode): 최고 빈도. (범주형 데이터용)

퍼짐 (Spread): • 분산 (Variance): 퍼진 정도 (단위가 2 됨)

- 표준편차 (Std Dev): 퍼진 정도 (단위가 원본과 동일 → 해석 용이)

5. 핵심 시각화 (Key Visualizations)

앤스컴 4중주 (Anscombe's Quartet) 교훈: "숫자(통계)만 보지 말고, 항상 그래프를 그려라!"

히스토그램 (Histogram) 수량형 변수 1개의 분포 확인. (빈(bin) 너비에 민감)

막대 그래프 (Bar Plot) 범주형 변수 1개의 빈도 비교.

산점도 (Scatter Plot) 수량형 변수 2개의 관계 확인. (방향, 강도, 형태, 이상치)

박스 플롯 (Box Plot) & 바이올린 플롯 (Violin Plot) (범주형) 그룹 간 (수량형) 변수의 분포 비교. (바이올린이 더 많은 모양 정보 제공)

5변수 시각화 (Gapminder) X축, Y축, 크기 (Size), 색상 (Color), 애니메이션 (Time)

December 10, 2025

- 강의명: CS109A: 데이터 과학 입문
- 주차: Lecture 03
- 교수명: Pavlos Protopapas, Kevin Rader, Chris Gumb
- 목적: Lecture 03의 핵심 개념 학습

Contents

1 개요	3
2 용어 정리	4
3 핵심 개념: Pandas 자료구조	5
3.1 Series: 1차원 데이터의 마법 지팡이	5
3.1.1 Series 생성하기	5
3.1.2 Series의 주요 속성	6
3.1.3 Series 기본 메서드	7
3.2 DataFrame: 2차원 데이터의 강력한 테이블	9
3.2.1 DataFrame 생성하기	9
3.2.2 DataFrame의 주요 속성	10
3.2.3 DataFrame 기본 메서드	11
4 절차/방법: 데이터 다루기 실전	13
4.1 1단계: 데이터 불러오기 및 초기 검사	13
4.2 2단계: 데이터 정제	15
4.2.1 컬럼명 변경	15
4.2.2 불필요한 컬럼 제거	15
4.2.3 데이터 타입 변환	15
4.2.4 텍스트 정규화	17
4.2.5 중복 데이터 확인 및 처리	17
4.2.6 결측치 확인 및 처리	18
4.3 3단계: 데이터 선택, 필터링, 정렬	19
4.3.1 Boolean Indexing (마스크 활용)	19
4.3.2 loc 및 iloc 사용	19
4.3.3 데이터 정렬	20
4.4 4단계: 데이터 변환 및 집계	21
4.4.1 값 변환 (replace)	21
4.4.2 문자열 분리 및 확장 (str.split, explode)	21
4.4.3 함수 적용 (apply)	22
4.4.4 데이터 그룹화 및 집계 (groupby, agg)	22
4.4.5 교차 분석표 (crosstab)	23
4.5 5단계: 데이터 저장	24
5 실습 코드 종합 예제	25
6 체크리스트	27
7 FAQ (자주 묻는 질문)	28
8 빠르게 훑어보기 (1페이지 요약)	31
9 부록: 환경 설정 및 추가 정보	32
9.1 Python 환경 설정	32
9.2 Harvard OnDemand (클라우드 환경)	32
9.3 Jupyter Notebook 사용 팁	32
9.4 추가 학습 자료	33

1 개요

▣ 핵심 요약

이 문서는 Harvard CS1090A 데이터 과학 입문 강의의 Pandas 기초 부분을 다룹니다. Pandas는 파이썬에서 표 형태의 데이터를 다루는 데 필수적인 라이브러리입니다. 이 노트는 Pandas의 핵심 자료구조인 Series와 DataFrame의 개념, 생성 방법, 데이터 로딩, 검사, 정제, 기본적인 데이터 분석 방법을 설명합니다. 처음 배우는 사람도 쉽게 이해할 수 있도록 예시와 함께 단계별로 설명합니다. 이 노트만으로도 Pandas의 기본을 익히고 실습할 수 있도록 구성했습니다.

주요 학습 목표:

- Pandas의 기본 자료구조인 Series와 DataFrame 이해하기
- 다양한 방법으로 Series와 DataFrame 생성하기
- CSV 파일 등 외부 데이터를 Pandas DataFrame으로 불러오기
- 데이터를 검사하고 요약하는 기본 방법 익히기 ('head', 'info', 'describe' 등)
- 데이터 정제 기법 배우기 (컬럼명 변경, 타입 변환, 결측치 및 중복값 처리 등)
- Boolean indexing, loc, iloc을 이용한 데이터 선택 및 필터링
- 데이터 정렬, 집계, 그룹화 기초 ('sort_values', 'groupby', 'agg')
- 정제된 데이터를 파일로 저장하기 ('to_csv')

참고: 이 노트는 실제 강의 내용과 코드를 바탕으로 재구성되었으며, 추가적인 설명과 예시가 포함되어 있습니다.

2 용어 정리

데이터 분석 여정에서 자주 만나게 될 Pandas 관련 용어들을 미리 알아두면 학습에 큰 도움이 됩니다. 아래 표는 이 노트에서 사용되는 주요 용어들을 정리한 것입니다.

용어	쉬운 설명	원어	비고
Pandas	파이썬으로 표 형태 데이터를 쉽게 다루게 해주는 도구 모음	Pandas	데이터 분석의 필수 라이브러리
Series	1차원 배열 형태의 데이터 (하나의 열)	Series	값과 인덱스로 구성됨
DataFrame	2차원 표 형태의 데이터 (여러 개의 열)	DataFrame	Series 여러 개가 모인 것
Index	데이터의 각 행(row)을 식별하는 이름표 또는 번호	Index	기본은 0부터 시작하는 숫자
dtype	데이터의 종류 (숫자, 문자열, 날짜 등)	Data Type	<code>int64, float64, object, bool, datetime64, category</code> 등
NaN	데이터가 없음을 나타내는 특별한 값	Not a Number	결측치(Missing Value)라고도 함
Boolean Indexing	참/거짓(True/False) 값으로 원하는 데이터만 골라내는 방법	Boolean Indexing	조건에 맞는 데이터를 필터링할 때 사용
loc	이름표(label) 기반으로 데이터를 선택하는 방법	loc (Label-based)	예: <code>df.loc[3], df.loc['row_name']</code>
iloc	위치(position) 기반으로 데이터를 선택하는 방법	iloc (Integer position-based)	예: <code>df.iloc[0], df.iloc[0:5]</code>
Method Chaining	여러 함수(메서드)를 절()으로 연결하여 순차적으로 실행하는 것	Method Chaining	코드를 간결하게 만들. 예: <code>df.sort_values().head()</code>
GroupBy	특정 기준(컬럼 값)에 따라 데이터를 그룹으로 묶는 작업	GroupBy	그룹별 통계 계산 등에 사용
Aggregation	그룹으로 묶인 데이터에 대해 요약 통계(평균, 합계 등)를 계산하는 것	Aggregation	<code>agg(), mean(), sum(), count()</code> 등
Crosstab	두 변수(컬럼) 간의 빈도수를 표 형태로 교차 분석하는 것	Cross-tabulation	범주형 변수 간의 관계 파악에 유용
Explode	하나의 셀에 리스트 형태로 들어있는 값을 여러 행으로 펼치는 작업	Explode	콤마로 구분된 문자열 등을 분리할 때 사용

3 핵심 개념: Pandas 자료구조

Pandas는 파이썬에서 기본적으로 제공하지 않는, 데이터 분석에 특화된 두 가지 핵심 자료구조를 제공합니다: **Series**와 **DataFrame**.

3.1 Series: 1차원 데이터의 마법 지팡이

Series란?

Series는 마치 라벨(이름표)이 붙어 있는 1차원 배열과 같습니다. 엑셀 시트의 한 열(column)이나, 키(key)와 값(value)으로 이루어진 사전(dictionary)을 떠올리면 이해하기 쉽습니다. 각 데이터 값에는 고유한 이름표, 즉 인덱스(index)가 붙어 있어 데이터를 쉽게 찾고 다룰 수 있습니다.

왜 Series가 필요한가? 파이썬의 기본 리스트(list)도 1차원 데이터를 저장할 수 있지만, Series는 다음과 같은 장점을 가집니다.

- **명시적인 인덱스:** 단순한 숫자 위치뿐만 아니라 원하는 문자열 등으로 인덱스를 지정할 수 있습니다.
- **NumPy 기반 성능:** 내부적으로 고성능 NumPy 배열을 사용하므로 대량 데이터 처리 속도가 빠릅니다.
- **데이터 정렬 및 연산 용이성:** 인덱스를 기준으로 데이터를 정렬하거나, Series 간의 연산을 쉽게 수행할 수 있습니다.
- **다양한 데이터 타입 지원 및 결측치 처리:** 숫자, 문자열뿐 아니라 다양한 데이터 타입을 지원하며, 데이터가 없는 경우(결측치, NaN)를 효과적으로 다룰 수 있습니다.

3.1.1 Series 생성하기

가장 기본적인 방법은 파이썬 리스트를 사용하는 것입니다.

```

1 import pandas as pd
2 import numpy as np
3
4 # 숫자리스트로 Series 생성
5 numbers = [1, 3, 5, np.nan, 6, 8]
6 s = pd.Series(numbers)
7 print(s)

```

Listing 1: 파이썬 리스트로 Series 생성

결과:

```

0    1.0
1    3.0
2    5.0
3    NaN
4    6.0
5    8.0
dtype: float64

```

왼쪽의 0, 1, 2...는 자동으로 생성된 기본 인덱스이고, 오른쪽은 리스트의 값입니다. `np.nan`은 데이터가 없음을 의미하는 결측치입니다. 데이터 타입(`dtype`)은 실수(`float64`)로 자동 지정되었습니다. (`NaN` 때문에)

인덱스를 직접 지정할 수도 있습니다.

```

1 # 문자열리스트와사용자정의인덱스
2 data = ['a', 'b', 'c']
3 index = [5, 3, 1]
4 s_custom_index = pd.Series(data, index=index)
5 print(s_custom_index)

6
7 # 인덱스를문자열로지정
8 s_string_index = pd.Series([-3, -2, -1], index=['foo', 'bar', 'baz'])
9 print(s_string_index)

```

Listing 2: 인덱스를 지정하여 Series 생성

결과:

```

5    a
3    b
1    c
dtype: object

foo    -3
bar    -2
baz    -1
dtype: int64

```

인덱스가 숫자가 아닌 문자열이어도 되며, 순서대로가 아니어도 됩니다.

3.1.2 Series의 주요 속성

Series 객체는 데이터 자체 외에도 유용한 정보를 속성으로 가지고 있습니다.

```

1 l = [-3, -2, -1]
2 series = pd.Series(l, name='ints!', dtype=str) # 이름과타입지정
3 series.index = ['foo', 'bar', 'foo'] # 인덱스변경중복 ( 가능!)
4
5 print(f"값 (Values): {series.values}")
6 print(f"인덱스 (Index): {series.index}")
7 print(f"이름 (Name): {series.name}")
8 print(f"데이터 타입 (dtype): {series.dtype}")
9 print(f"값의 타입: {type(series.values)}")

```

Listing 3: Series 속성 확인

결과:

```
값 (Values): ['-3' '-2' '-1']
```

인덱스 (Index): Index(['foo', 'bar', 'foo'], dtype='object')

이름 (Name): ints!

데이터 타입 (dtype): object

값의 타입: <class 'numpy.ndarray'>

- **values:** Series의 실제 데이터 값들을 NumPy 배열 형태로 반환합니다. (주의: NumPy 배열은 모든 원소가 동일한 데이터 타입이어야 합니다. 만약 다양한 타입의 데이터가 Series에 있다면, 가장 포괄적인 타입인 object로 변환됩니다. object 타입은 실제 값 대신 메모리 주소를 저장하므로 성능 저하의 원인이 될 수 있습니다.)
- **index:** Series의 인덱스 객체를 반환합니다. 인덱스는 단순한 숫자가 아니라, 데이터를 식별하는 라벨이며 중복될 수도 있습니다.
- **name:** Series의 이름을 문자열로 반환합니다. DataFrame의 컬럼명으로 사용됩니다.
- **dtype:** Series에 저장된 데이터의 타입을 알려줍니다.

3.1.3 Series 기본 메서드

Series에는 데이터를 탐색하고 요약하는 데 유용한 여러 메서드가 내장되어 있습니다.

```

1 data = [1, 3, 5, np.nan, 6, 8]
2 s = pd.Series(data)
3
4 print("--- 처음 2개 데이터 (head) ---")
5 print(s.head(2))
6
7 print("\n--- 마지막 2개 데이터 (tail) ---")
8 print(s.tail(2))
9
10 print("\n--- 기술통계요약 (describe) ---")
11 print(s.describe())

```

Listing 4: Series 기본 메서드 사용

결과:

--- 처음 2개 데이터 (head) ---

0 1.0

1 3.0

dtype: float64

--- 마지막 2개 데이터 (tail) ---

4 6.0

5 8.0

dtype: float64

--- 기술 통계 요약 (describe) ---

count 5.000000

mean 4.600000

std 2.701851

```
min      1.000000
25%     3.000000
50%     5.000000
75%     6.000000
max     8.000000
dtype: float64
```

- `head(n)`: Series의 처음 n개 데이터를 보여줍니다. (기본값 n=5)
- `tail(n)`: Series의 마지막 n개 데이터를 보여줍니다. (기본값 n=5)
- `describe()`: 수치형 데이터의 경우 개수(count), 평균(mean), 표준편차(std), 최솟값(min), 사분위수(25

3.2 DataFrame: 2차원 데이터의 강력한 테이블

DataFrame 란?

DataFrame은 여러 개의 Series가 같은 인덱스를 공유하며 모여 있는 2차원 표 형태의 자료구조입니다. 엑셀 스프레드시트나 데이터베이스 테이블과 매우 유사합니다. 각 열(column)은 하나의 Series에 해당하며, 고유한 컬럼 이름을 가집니다. 각 행(row)은 인덱스로 식별됩니다.

왜 DataFrame이 필요한가? Series가 1차원 데이터를 다루는데 유용하다면, DataFrame은 다음과 같은 이유로 2차원 데이터를 다루는데 필수적입니다.

- 표 형태 데이터 처리:** 엑셀, CSV 파일 등 일반적인 표 형태 데이터를 직관적으로 표현하고 조작할 수 있습니다.
- 다양한 데이터 타입:** 각 열(Series)마다 다른 데이터 타입(숫자, 문자, 날짜 등)을 가질 수 있습니다.
- 유연한 인덱싱 및 선택:** 행과 열의 이름 또는 위치를 이용해 원하는 데이터를 쉽게 선택하고 필터링할 수 있습니다.
- 강력한 데이터 정제 및 변환 기능:** 결측치 처리, 데이터 타입 변환, 데이터 합치기, 그룹화 등 다양한 데이터 처리 기능을 제공합니다.
- 다양한 입출력 지원:** CSV, Excel, 데이터베이스 등 다양한 형식의 데이터를 쉽게 읽고 쓸 수 있습니다.

3.2.1 DataFrame 생성하기

DataFrame을 만드는 방법은 여러 가지가 있습니다.

1. 딕셔너리의 리스트 사용 (행 단위 생성) 각 딕셔너리가 하나의 행(row)이 되고, 딕셔너리의 키가 열(column) 이름이 됩니다.

```

1 import pandas as pd
2
3 data = [{'fruit': 'apple', 'color': 'red'},
4          {'fruit': 'grape', 'color': 'purple'}]
5
6 df_fruits = pd.DataFrame(data)
7 print(df_fruits)

```

Listing 5: 딕셔너리의 리스트로 DataFrame 생성

결과:

	fruit	color
0	apple	red
1	grape	purple

2. 리스트의 딕셔너리 사용 (열 단위 생성) 딕셔너리의 키가 열(column) 이름이 되고, 값으로 주어지는 리스트가 해당 열의 데이터가 됩니다.

```

1 import pandas as pd
2
3 data = {'fruit': ['apple', 'grape'],
4          'color': ['red', 'purple']}

```

```

5
6 df_fruits_alt = pd.DataFrame(data)
7 print(df_fruits_alt)

```

Listing 6: 리스트의 덱셔너리로 DataFrame 생성

결과:

```

fruit    color
0  apple     red
1  grape   purple

```

3. NumPy 배열 사용 2차원 NumPy 배열로부터 DataFrame을 생성할 수 있습니다. 이때 열 이름을 명시적으로 지정해주는 것이 좋습니다.

```

1 import pandas as pd
2 import numpy as np
3
4 # 차원 2 NumPy 배열 생성 (: 100x2 크기, 다변량정규분포데이터 )
5 data_np = np.random.multivariate_normal(mean=[0, -1], cov=[[1, 0.5], [0.5, 1]], size=100)
6
7 # NumPy 배열과 컬럼이름을 사용하여 DataFrame 생성
8 df_np = pd.DataFrame(data=data_np, columns=["X", "Y"])
9 print(df_np.head()) # 처음 5 행만 출력

```

Listing 7: NumPy 배열로 DataFrame 생성

결과(값은 실행 시마다 다름):

	X	Y
0	-1.558152	-1.781442
1	-0.114449	-2.243731
2	-1.675996	-2.528247
3	0.308387	-1.627142
4	-1.138339	-1.671097

3.2.2 DataFrame의 주요 속성

DataFrame도 Series와 유사하게 유용한 속성들을 제공합니다.

```

1 # 위의 df_np DataFrame 사용
2 print(f"형태 (Shape): {df_np.shape}")
3 print(f"컬럼 (Columns): {df_np.columns}")
4 print(f"인덱스 (Index): {df_np.index}")
5 print(f"값 (Values): \n{df_np.values[:2]})" # 값은 NumPy 배열, 처음 행만 2 출력

```

Listing 8: DataFrame 속성 확인

결과(값은 실행 시마다 다름):

형태 (Shape): (100, 2)

컬럼 (Columns): Index(['X', 'Y'], dtype='object')

인덱스 (Index): RangeIndex(start=0, stop=100, step=1)

값 (Values):

[[-1.55815228 -1.7814424]]

[[-0.11444917 -2.24373138]]

- **shape**: DataFrame의 형태(행 개수, 열 개수)를 튜플로 반환합니다.
- **columns**: DataFrame의 열 이름(컬럼명)들을 Index 객체로 반환합니다.
- **index**: DataFrame의 행 이름(인덱스)들을 Index 객체로 반환합니다.
- **values**: DataFrame의 모든 데이터 값을 2차원 NumPy 배열 형태로 반환합니다.

3.2.3 DataFrame 기본 메서드

DataFrame의 데이터를 빠르게 파악하기 위한 기본 메서드들입니다.

```

1 # 위의 df_np DataFrame 사용
2 print("--- 처음 3개 데이터 (head) ---")
3 print(df_np.head(3))
4
5 print("\n--- 마지막 2개 데이터 (tail) ---")
6 print(df_np.tail(2))
7
8 print("\n--- 요약정보 (info) ---")
9 df_np.info()
10
11 print("\n--- 기술통계요약 (describe) ---")
12 print(df_np.describe())

```

Listing 9: DataFrame 기본 메서드 사용

결과(값은 실행 시마다 다름):

--- 처음 3개 데이터 (head) ---

X Y

0	-1.558152	-1.781442
1	-0.114449	-2.243731
2	-1.675996	-2.528247

--- 마지막 2개 데이터 (tail) ---

X Y

98	-0.210365	-2.146889
99	1.032841	0.338218

--- 요약 정보 (info) ---

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 100 entries, 0 to 99

Data columns (total 2 columns):

```
#   Column Non-Null Count Dtype
---  -----
0    X        100 non-null    float64
1    Y        100 non-null    float64
dtypes: float64(2)
memory usage: 1.7 KB
```

--- 기술 통계 요약 (describe) ---

	X	Y
count	100.000000	100.000000
mean	-0.088686	-0.963459
std	0.981885	0.989715
min	-2.428570	-3.131102
25%	-0.781459	-1.639194
50%	-0.122675	-0.970172
75%	0.540450	-0.340263
max	2.308703	1.638069

- `head(n)`: DataFrame의 처음 n개 행을 보여줍니다. (기본값 n=5)
- `tail(n)`: DataFrame의 마지막 n개 행을 보여줍니다. (기본값 n=5)
- `info()`: DataFrame의 전반적인 정보를 요약하여 보여줍니다. 인덱스 타입, 컬럼 정보(이름, non-null 개수, 데이터 타입), 메모리 사용량 등을 확인할 수 있습니다. 데이터 타입을 확인하고 결측치 유무를 파악하는 데 매우 유용합니다.
- `describe()`: 수치형 컬럼들에 대한 기술 통계량을 계산하여 보여줍니다. 데이터의 분포를 빠르게 파악할 수 있습니다.

4 절차/방법: 데이터 다루기 실전

이제 실제 데이터를 Pandas DataFrame으로 불러와서 검사하고, 필요한 형태로 정제하고, 간단한 분석을 수행하는 과정을 단계별로 살펴보겠습니다. 예제 데이터로는 CS1090A 학생 설문조사 결과를 사용합니다.

4.1 1단계: 데이터 불러오기 및 초기 검사

가장 먼저 할 일은 데이터를 DataFrame으로 읽어 들이는 것입니다. CSV(Comma Separated Values) 파일은 가장 흔한 데이터 형식 중 하나이며, pd.read_csv() 함수를 사용합니다.

```

1 import pandas as pd
2
3 # CSV 파일을으로 DataFrame 불러오기
4 # 'data/cs1090a_survey_raw.csv' 파일이 있다고 가정
5 # 실제 파일 경로는 환경에 맞게 수정해야 합니다
6 try:
7     df = pd.read_csv("data/cs1090a_survey_raw.csv")
8     print("CSV 파일로딩 성공 !")
9 except FileNotFoundError:
10     print("오류: CSV 파일을 찾을 수 없습니다 . 파일 경로를 확인하세요 .")
11     # 예제 진행을 위해 임시 DataFrame 생성
12     data = {'Timestamp': ['9/9/2025 17:43:09'],
13             'What\'s your Harvard affiliation?\n': ['Bachelor\'s (CS)'],
14             'Have you used Jupyter Notebooks before?': ['Yes'],
15             'How many years of Python programming experience do you have?\n'
16             'Choose the range that best fits your experience.): ['Less
17             than 1 year'],
18             'Rate your current Pandas skill level.): [2],
19             'What is your primary OS?': ['MacOS'],
20             'Do you use normally code/browse in dark mode?': ['No'],
21             'What languages do you speak? \n(Comma separated)\n\nExample:
22             Hittite, Elvish, Cornish, Klingon': ['English'],
23             'Which continents have you visited?': ['Africa, Asia, Europe,
24             North America, South America'],
25             'When were you born?': ['11/15/2004'],
26             'What time do you usually wake up in the morning?': ['10:00:00 AM'
27             ],
28             'What time do you usually go to bed?': ['12:00:00 AM'],
29             'Favorite Season?': ['Spring'],
30             'Where do you usually get your caffeine?': ['Tea'],
            'Which kind of pet do you prefer?': ['Pet rock'],
            'What\'s your favorite movie?': [None], # 예시로 None 사용
            'What movie genres do you particularly enjoy?\n(select as many as you
            like)': ['Comedy'],
            'List up to 3 of your hobbies.\n(comma separated)\n\nExample:
            playing kazoo, bird watching, stamp collecting': ['tennis,
            movies, eating'],
            'How was HW0?': [None]}
```

```
31 df = pd.DataFrame(data)
```

Listing 10: CSV 파일 읽어오기

데이터를 불러온 후에는 어떤 데이터가 들어 있는지 확인하는 것이 중요합니다. `head()`, `shape`, `columns`, `info()`, `describe()` 등을 사용합니다.

```
1 # 처음개 5 행확인
2 print(" --- 데이터미리보기 (head) ---")
3 print(df.head())
4
5 # 데이터크기확인행 (, 열개수 )
6 print(f"\n --- 데이터크기 (shape) ---")
7 print(f"{df.shape[0]} 행, {df.shape[1]} 열")
8
9 # 컬럼명리스트확인
10 print("\n --- 컬럼명 (columns) ---")
11 print(list(df.columns))
12
13 # 데이터요약정보확인결측치 (, 데이터타입 )
14 print("\n --- 요약정보 (info) ---")
15 df.info()
16
17 # 수치형데이터기술통계확인
18 print("\n --- 기술통계 (describe) ---")
19 print(df.describe())
```

Listing 11: 데이터 초기 검사

초기 검사 결과 (예시 데이터 기준):

- `head()`: 데이터의 실제 값과 컬럼명을 눈으로 확인할 수 있습니다. 컬럼명이 너무 길거나 줄바꿈 문자가 포함되어 지저분해 보입니다. 일부 값은 비어 있는 것 같습니다 (NaN).
- `shape`: 데이터의 전체 크기를 알려줍니다. 예제 데이터는 175행 19열입니다.
- `columns`: 모든 컬럼명을 리스트로 보여줍니다. 너무 길고 특수문자가 있어 다루기 불편합니다.
- `info()`: 각 컬럼의 non-null 값 개수와 데이터 타입을 보여줍니다. 대부분의 컬럼이 `object` 타입(주로 문자열)이며, 'Rate your current Pandas skill level.' 컬럼만 `int64`(정수) 타입입니다. Non-null 개수를 보면 여러 컬럼에 결측치가 있음을 알 수 있습니다 ('dob', 'wake_time', 'fav_movie', 'hobbies', 'hw0' 등).
- `describe()`: 현재는 'Rate your current Pandas skill level.' 컬럼에 대한 통계만 보여줍니다. 평균 2.5, 표준편차 1.1 정도임을 알 수 있습니다.

4.2 2단계: 데이터 정제

초기 검사 결과를 바탕으로 데이터를 분석하기 좋은 형태로 만드는 정제 작업을 수행합니다.

4.2.1 컬럼명 변경

길고 복잡한 컬럼명은 사용하기 불편하므로, 짧고 의미 있는 이름으로 변경합니다. 일반적으로 소문자와 언더스코어(_)를 사용하는 것이 좋습니다.

```

1 # 새컬럼명리스트정의
2 new_cols = [
3     "timestamp", "program", "jupyter", "python_exp", "pandas_skill", "os", "
4         dark_mode",
5     "languages", "continents", "dob", "wake_time", "sleep_time", "fav_season",
6     "caffeine", "pet", "fav_movie", "fav_genres", "hobbies", "hw0"
7 ]
8
9 # 의 DataFrame columns 속성에새리스트할당
10 df.columns = new_cols
11
12 # 변경된컬럼명확인
13 print("--- 변경된컬럼명확인      (head) ---")
14 print(df.head())

```

Listing 12: 컬럼명 변경

이제 df.program이나 df['pandas_skill']처럼 훨씬 간결하게 컬럼에 접근할 수 있습니다.

4.2.2 불필요한 컬럼 제거

분석에 사용하지 않을 컬럼은 제거하여 DataFrame을 가볍게 만듭니다. 여기서는 'timestamp' 컬럼을 제거합니다. drop() 메서드를 사용하며, axis=1은 열을 기준으로 제거하라는 의미입니다. (axis=0은 행 기준)

```

1 # 'timestamp' 컬럼제거 (axis은=1 열을의미 )
2 # inplace=True 사용하면원본이 DataFrame 바로수정됨
3 # 여기서는수정된결과를다시에      df 할당
4 df = df.drop("timestamp", axis=1)
5
6 # 제거확인 (shape 및 head)
7 print(f"제거 후데이터크기 : {df.shape}")
8 print(df.head(2))

```

Listing 13: 컬럼 제거

컬럼 개수가 19개에서 18개로 줄어든 것을 확인할 수 있습니다.

4.2.3 데이터 타입 변환

현재 대부분의 컬럼이 object 타입입니다. 분석 목적에 맞게 적절한 데이터 타입으로 변환해야 합니다.

Boolean 타입 변환: 'Yes'/'No' 형태의 데이터를 True/False로 변환합니다.

```

1 # 'dark_mode' 컬럼값이 'yes'이면 True, 아니면로 False 변환
2 # .str.lower()를 먼저적용하여대소문자구분없이처리
3 df['dark_mode'] = (df['dark_mode'].str.lower() == 'yes')
4
5 # 'jupyter' 컬럼도동일하게처리
6 df['jupyter'] = (df['jupyter'].str.lower() == 'yes')
7
8 # 타입변경확인
9 print("--- dark_mode dtype ---")
10 print(df['dark_mode'].dtype)
11 print("--- jupyter dtype ---")
12 print(df['jupyter'].dtype)

```

Listing 14: Boolean 타입 변환

이제 bool 타입이 되어 논리 연산에 사용하기 편리해졌습니다.

Category 타입 변환: 정해진 몇 가지 값 중 하나를 가지는 범주형 데이터는 category 타입으로 변환하는 것이 좋습니다. 메모리 효율성을 높이고, 해당 컬럼이 가질 수 있는 값을 제한할 수 있습니다.

```

1 # 'os' 컬럼을 category 타입으로변환
2 df['os'] = df['os'].astype('category')
3
4 # 타입및카테고리확인
5 print(df['os'].dtype)
6 print(df['os'].cat.categories)

```

Listing 15: Category 타입 변환

순서 있는 Category 타입 변환 (Ordinal): 'python_exp'처럼 순서가 있는 범주형 데이터는 순서를 명시하여 category 타입으로 변환합니다. 이렇게 하면 크기 비교나 정렬이 의미 있게 가능해집니다. CategoricalDtype을 사용합니다.

```

1 from pandas.api.types import CategoricalDtype
2
3 # Python 경험순서정의
4 experience_order = ['Less than 1 year', '1-2 years', '2-4 years', '4+ years']
5
6 # 순서있는 CategoricalDtype 생성
7 experience_dtype = CategoricalDtype(categories=experience_order, ordered=True)
8
9 # 'python_exp' 컬럼을정의된타입으로변환
10 # 원본컬럼을유지하고새컬럼 'python_experience' 생성강의 ( 노트방식 )
11 # 실제로는 df['python_exp'] = df['python_exp'].astype(experience_dtype) 처럼덮어
12 # 쓰는경우가많음
12 df['python_experience'] = df['python_exp'].astype(experience_dtype)
13
14 # 타입및순서확인
15 print(df['python_experience'].dtype)
16 print(df['python_experience'].head())
17 # 예: 경험이 '1-2 years'보다 많은 사람필터링
18 print("\n--- 경험 '1-2 years' 초과 ---")

```

```
19 print(df[df['python_experience'] > '1-2 years']['python_experience'].head())
```

Listing 16: 순서 있는 Category 타입 변환

Datetime 타입 변환: 날짜/시간 관련 문자열은 datetime 타입으로 변환해야 날짜 계산 등을 수행할 수 있습니다. pd.to_datetime() 함수를 사용하며, errors='coerce' 옵션은 변환 불가능한 값을 NaT(Not a Time)로 처리합니다.

```
1 # 'dob' 컬럼을 datetime 타입으로변환
2 df['dob'] = pd.to_datetime(df['dob'], errors='coerce')
3
4 # 타입확인
5 print(df['dob'].dtype)
```

Listing 17: Datetime 타입 변환

시간 정보를 가진 'wake_time', 'sleep_time'도 유사하게 처리할 수 있으나, 시간만 다룰 경우 다른 방식이 더 적합할 수 있습니다. 이 노트에서는 일단 변환하지 않습니다.

4.2.4 텍스트 정규화

사용자가 직접 입력한 텍스트 데이터는 대소문자가 섞여 있거나 앞뒤 공백이 있을 수 있습니다. 분석의 일관성을 위해 모두 소문자로 변환하고 불필요한 공백을 제거하는 것이 좋습니다. str.lower()와 str.strip() 메서드를 사용합니다.

```
1 # object 타입주로 (문자열) 컬럼만선택
2 str_cols = df.select_dtypes(include='object').columns
3
4 # 각문자열컬럼에대해소문자변환적용
5 for c in str_cols:
6     # 결측치가있을수있으므로 .str 접근자사용
7     df[c] = df[c].str.lower()
8     # 필요시앞뒤공백제거 : df[c] = df[c].str.strip()
9
10 # 변경확인예 (: program 컬럼)
11 print(df['program'].head())
```

Listing 18: 텍스트 소문자 변환

4.2.5 중복 데이터 확인 및 처리

완전히 동일한 행이 중복되어 있는지 확인합니다. duplicated()는 각 행이 중복인지 여부를 boolean Series로 반환하고, sum()을 사용하면 중복된 행의 개수를 알 수 있습니다.

```
1 duplicate_rows = df.duplicated().sum()
2 print(f"중복된 행의개수 : {duplicate_rows}")
3
4 # 만약중복행이있다면제거 :
5 # df = df.drop_duplicates()
```

Listing 19: 중복 행 확인

예제 데이터에는 중복 행이 없습니다.

4.2.6 결측치 확인 및 처리

데이터에 값이 없는 경우(NaN, NaT 등)를 결측치라고 합니다. `info()` 메서드로 컬럼별 non-null 개수를 확인하거나, `isna().sum()`으로 컬럼별 결측치 개수를 직접 확인할 수 있습니다.

```
1 print("--- 컬럼별결측치개수 ---")
2 print(df.isna().sum())
```

Listing 20: 컬럼별 결측치 개수 확인

결측치가 많은 컬럼('hw0', 'fav_movie', 'hobbies' 등)과 적은 컬럼이 있습니다.

결측치를 처리하는 방법은 여러 가지입니다.

- 제거:** `dropna()` 메서드를 사용하여 결측치가 포함된 행 또는 열을 제거합니다. 특정 컬럼에 결측치가 있는 행만 제거할 수도 있습니다 (`subset` 인자 사용). 분석에 필수적인 정보가 없는 행을 제거할 때 사용합니다.
- 대체:** `fillna()` 메서드를 사용하여 결측치를 특정 값(예: 0, 평균값, 최빈값, 'Unknown' 등)으로 채웁니다. 데이터 손실을 최소화하고 싶을 때 사용합니다.

```
1 # 예시 : 'hobbies' 컬럼에 결측치가 있는 행을 제거한 결과 확인 원본      (변경 X)
2 df_dropped_hobbies = df.dropna(subset=['hobbies'])
3 print(f"원본 행 개수 : {df.shape[0]}")
4 print(f"'hobbies' 결측치 제거 후 행 개수 : {df_dropped_hobbies.shape[0]}")
5
6 # 예시 : 'languages' 컬럼에 결측치가 있는 행을 실제로 제거 원본      (변경 O)
7 print(f"\n제거 전 'languages' 결측치 개수 : {df['languages'].isna().sum()}")
8 df.dropna(subset=['languages'], inplace=True) # inplace=True는 True 원본을 직접 수정
9 print(f"제거 후 'languages' 결측치 개수 : {df['languages'].isna().sum()}")
10 print(f"최종 데이터 크기 : {df.shape}")
```

Listing 21: 결측치 처리 예시 (제거)

주의사항

결측치 처리는 분석 결과에 큰 영향을 미칠 수 있으므로 신중하게 결정해야 합니다. 무조건 제거하거나 특정 값으로 채우기보다는, 데이터의 특성과 분석 목적을 고려하여 가장 적절한 방법을 선택해야 합니다. 경우에 따라서는 결측치 자체를 하나의 정보로 활용할 수도 있습니다.

4.3 3단계: 데이터 선택, 필터링, 정렬

정제된 데이터에서 원하는 부분만 선택하거나 특정 조건에 맞는 데이터를 필터링하고, 필요에 따라 정렬하는 방법을 알아봅니다.

4.3.1 Boolean Indexing (마스크 활용)

가장 강력하고 흔하게 사용되는 방법입니다. 특정 조건(예: 'pandas_skill' > 3)을 DataFrame이나 Series에 적용하면, 각 행(또는 원소)이 조건을 만족하는지 여부를 나타내는 True/False 값으로 이루어진 Series(이를 마스크(mask)라고 부름)가 반환됩니다. 이 마스크를 DataFrame의 인덱서([])에 넣어주면 조건이 True인 행들만 선택됩니다.

```

1 # 조건: Pandas 스킬레벨이보다 3 큰경우
2 mask = df['pandas_skill'] > 3
3
4 # 마스크를사용하여조건에맞는행들만선택
5 df_high_skill = df[mask]
6
7 # 결과확인 (pandas_skill 컬럼만출력 )
8 print(df_high_skill['pandas_skill'].head())
9
10 # 조건: dark 를 mode 사용하고 (True) 가 OS 인 MacOS 경우
11 # 여러조건은 & (AND), | (OR) 연산자로연결하며 , 각조건은괄호로묶어야함
12 mask_dark_mac = (df['dark_mode'] == True) & (df['os'] == 'macos')
13 df_dark_mac = df[mask_dark_mac]
14
15 # 결과확인 (os, dark_mode 컬럼및크기출력 )
16 print("\n--- Dark Mode 사용자 (MacOS) ---")
17 print(df_dark_mac[['os', 'dark_mode']].head())
18 print(f"해당 학생수 : {df_dark_mac.shape[0]}")
```

Listing 22: Boolean Indexing으로 필터링

4.3.2 loc 및 iloc 사용

loc과 iloc은 특정 행과 열을 선택하는 더 정교한 방법입니다.

- loc: 라벨(이름) 기반 인덱싱. 행과 열의 이름(인덱스 라벨, 컬럼명)을 사용합니다. 슬라이싱 시 끝점을 포함합니다.
- iloc: 위치(정수) 기반 인덱싱. 0부터 시작하는 행과 열의 순서(위치)를 사용합니다. 슬라이싱 시 끝점을 제외합니다(파이썬 기본 슬라이싱과 동일).

```

1 # 예시 DataFrame 생성
2 data = {'A': [10, 20, 30, 40], 'B': [50, 60, 70, 80]}
3 index = ['r1', 'r2', 'r3', 'r4']
4 df_example = pd.DataFrame(data, index=index)
5 print("--- 예시 DataFrame ---")
6 print(df_example)
7
8 # loc 사용예시
```

```

9 print("\n--- loc 사용 ---")
10 # 행 'r2' 선택 (Series 반환)
11 print(f"행 'r2':\n{df_example.loc['r2']}") 
12 # 행 'r1', 'r3'과 열 'B' 선택 (DataFrame 반환)
13 print(f"\n행n 'r1', 'r3', 열 'B':\n{df_example[['r1', 'r3'], 'B']}") 
14 # 행 'r2'부터 'r4까지', 열 'A' 선택 (Series 반환)
15 print(f"\n행n 'r2':'r4', 열 'A':\n{df_example.loc['r2':'r4', 'A']}") 
16
17 # iloc 사용 예시
18 print("\n--- iloc 사용 ---")
19 # 첫번째 행 위치 (0) 선택 (Series 반환)
20 print(f"첫 번째 행 (iloc[0]):\n{df_example.iloc[0]}") 
21 # 첫번째, 세번째 행 위치 (0, 2)과 두번째 열 위치 (1) 선택 (Series 반환)
22 print(f"\n행n 0, 2, 열 1:\n{df_example.iloc[[0, 2], 1]}") 
23 # 첫 두 행 위치 (0, 1)과 모든 열 선택 (DataFrame 반환)
24 print(f"\n처음n 두 행 (iloc[0:2]):\n{df_example.iloc[0:2]}") 

```

Listing 23: loc 및 iloc 사용 예시

주의사항

loc과 iloc의 차이를 명확히 이해하는 것이 중요합니다. 특히 정수 인덱스를 사용할 때 혼동하기 쉽습니다. df.loc[0]은 인덱스 라벨이 '0'인 행을 찾고, df.iloc[0]은 첫 번째 위치에 있는 행을 찾습니다. 데이터 정렬이나 필터링 후 인덱스가 순차적이지 않을 때 iloc이 유용할 수 있습니다.

4.3.3 데이터 정렬

특정 컬럼의 값을 기준으로 행을 정렬할 때는 sort_values() 메서드를 사용합니다.

```

1 # 'pandas_skill' 기준으로 내림차순 정렬 (ascending=False)
2 df_sorted_skill = df.sort_values(by='pandas_skill', ascending=False)
3 print(df_sorted_skill[['program', 'pandas_skill']].head())
4
5 # 여러 컬럼 기준으로 정렬 예 (: 'program' 오름차순, 'age' 내림차순)
6 # df_sorted_multi = df.sort_values(by=['program', 'age'], ascending=[True, False])
7 # print(df_sorted_multi[['program', 'age']].head())

```

Listing 24: 데이터 정렬

주의: 정렬 후에는 인덱스가 뒤섞이게 됩니다. 필요하다면 reset_index(drop=True)를 사용하여 인덱스를 0부터 다시 부여하는 것이 좋습니다.

4.4 4단계: 데이터 변환 및 집계

데이터를 분석하기 좋은 형태로 변환하거나 그룹별로 요약 통계를 계산합니다.

4.4.1 값 변환 (replace)

특정 값을 다른 값으로 바꿀 때 사용합니다. 정규 표현식(regex)을 사용하여 패턴 기반의 변환도 가능합니다.

```

1 # 'program' 컬럼에서 'certificate' 포함된 문자열을 하나로 통일
2 df['program'] = df['program'].replace(r".*certificate.*",
3                                     "graduate certificate [extension school]"
4                                     ,
4                                     regex=True)
5
6 # 변경 확인 (value_counts)
7 print(df['program'].value_counts().head())

```

Listing 25: 값 변환 (replace)

4.4.2 문자열 분리 및 확장 (str.split, explode)

'languages', 'hobbies', 'fav_genres'처럼 콤마 등으로 구분된 여러 값이 하나의 문자열로 들어있는 경우, 각 값을 분리하여 분석하기 쉽게 만듭니다.

1. 문자열 분리 (str.split): 특정 구분자(예: ',')를 기준으로 문자열을 나누어 리스트로 만듭니다.

2. 데이터 확장 (explode): 리스트가 들어있는 컬럼을 기준으로, 리스트의 각 원소가 별도의 행이 되도록 DataFrame을 확장합니다.

```

1 # 'languages' 컬럼을 콤마와 공백 (' , ') 기준으로 분리하여 리스트 생성
2 split_languages = df['languages'].str.split(' , ')
3 print("--- 분리 후 (Series of lists) ---")
4 print(split_languages.head())
5
6 # 'languages' 컬럼을 기준으로 explode 수행원본 (변경 X)
7 df_exploded = df.explode('languages')
8 print("\n--- Explode 후 (languages 컬럼만 확인) ---")
9 # 예: 첫번째 학생 (index 0)이 여러 행으로 나타나는지 확인
10 print(df_exploded[df_exploded.index == 0]['languages'])
11
12 # 전체 언어 목록 확인 중복 (제거 및 정규화 포함)
13 all_languages = df['languages'].str.split(' , ').explode()
14 # 'mandarin' 관련 표현통일 예시 ()
15 all_languages = all_languages.replace(r'.*mandarin.*|mandarin|(chinese$)', 
16                                         'chinese (mandarin)', regex=True).str.
17                                         strip()
18 unique_languages = all_languages.unique()
19 print(f"\n--- 고유 언어 개수 : {len(unique_languages)} ---")
20 # print(sorted(list(unique_languages))) # 정렬하여 출력 생략 ()

```

Listing 26: 문자열 분리 및 확장

`explode`는 각 언어별 빈도수를 계산하거나, 특정 언어 사용자 그룹을 분석할 때 유용합니다.

4.4.3 함수 적용 (apply)

Series나 DataFrame의 각 원소 또는 행/열에 대해 사용자 정의 함수나 내장 함수를 적용할 때 사용합니다. 예를 들어, `split`된 리스트의 길이를 계산하여 각 학생이 구사하는 언어의 수를 계산할 수 있습니다.

```

1 # 'languages' 컬럼을 분리한 리스트의 길이를 계산하여      'num_languages' 컬럼 생성
2 # 결측치 (NaN)가 있는 경우 오류가 발생할 수 있으므로      , fillna('') 등으로 사전 처리 필요
3 # 여기서는 'languages' 결측치를 이미 제거했으므로 바로 적용
4 df['num_languages'] = df['languages'].str.split(',').apply(len)
5
6 # 가장 많은 언어를 구사하는 경우 확인
7 max_languages = df['num_languages'].max()
8 print(f"가장 많은 언어 구사수 : {max_languages}")
9 print(df.loc[df['num_languages'].idxmax(), ['languages', 'num_languages']]) # idxmax() 는 최댓값의 인덱스 반환

```

Listing 27: apply를 이용한 계산

4.4.4 데이터 그룹화 및 집계 (groupby, agg)

특정 컬럼(들)의 값을 기준으로 데이터를 그룹화하고, 각 그룹에 대해 집계 함수(평균, 합계, 개수 등)를 적용하여 요약 통계를 계산합니다.

```

1 # 'program' 별 평균 'pandas_skill' 계산
2 avg_skill_by_program = df.groupby('program')['pandas_skill'].mean()
3 print("--- 프로그램 별 평균 Pandas 스킬상위 (개5) ---")
4 print(avg_skill_by_program.sort_values(ascending=False).head())
5
6 # 여러 집계 함수를 동시에 적용 예 (: 평균 나이와 학생수 )
7 # 'age' 컬럼이 필요 앞서 (로부터 dob 계산했다고 가정)
8 if 'age' in df.columns:
9     program_summary = df.groupby('program').agg(
10         avg_age=('age', 'mean'),          # 'age' 컬럼에 mean 함수 적용
11         count=('program', 'size')       # 'program' 컬럼에 size 함수 개수 (세기) 적용
12     )
13     # 학생수가 명 2 이상인 프로그램만 필터링하고 평균 나이 기준 정렬
14     program_summary_filtered = program_summary[program_summary['count'] >= 2].sort_values('avg_age')
15     print("\n--- 주요 프로그램 별 평균 나이 및 학생수 ---")
16     print(program_summary_filtered)
17 else:
18     print("\n'age' 컬럼이 없어 프로그램 별 요약 통계를 계산할 수 없습니다 .")

```

Listing 28: groupby 및 agg 사용

4.4.5 교차 분석표 (crosstab)

두 범주형 변수 간의 관계를 파악하기 위해 각 조합에 해당하는 빈도수를 표 형태로 보여줍니다. 예를 들어, 운영체제(os)와 다크 모드(dark_mode) 선호도 간의 관계를 볼 수 있습니다.

```
1 # 'os'와 'dark_mode' 간의 교차표 생성
2 os_darkmode_crosstab = pd.crosstab(df['os'], df['dark_mode'])
3 print("--- 별 OS Dark Mode 선호도 교차표 ---")
4 print(os_darkmode_crosstab)
5
6 # 비율로 보고 싶다면 normalize 인자 사용
7 # os_darkmode_crosstab_ratio = pd.crosstab(df['os'], df['dark_mode'],
8 #                                              normalize='index')
9 # print("\n--- 별 OS Dark Mode 선호도 비율 ---")
# print(os_darkmode_crosstab_ratio)
```

Listing 29: crosstab 사용

4.5 5단계: 데이터 저장

정제 및 분석된 DataFrame을 나중에 다시 사용하기 위해 파일로 저장합니다. CSV 파일로 저장하는 것이 일반적이며, `to_csv()` 메서드를 사용합니다. `index=False` 옵션을 주면 DataFrame의 인덱스가 파일에 저장되지 않습니다.

```
1 # 정제된을 DataFrame 'survey_final.csv' 파일로저장인덱스 ( 제외 )
2 try:
3     df.to_csv('survey_final.csv', index=False)
4     print("O|DataFrame 'survey_final.csv' 파일로저장되었습니다 .")
5 except Exception as e:
6     print(f"파일 저장중오류발생 : {e}")
```

Listing 30: DataFrame을 CSV 파일로 저장

5 실습 코드 종합 예제

다음은 위에서 설명한 주요 Pandas 작업을 연속적으로 보여주는 코드 예제입니다.

```

1 import pandas as pd
2 import numpy as np
3 from pandas.api.types import CategoricalDtype
4
5 # --- 1. 데이터로딩 ---
6 try:
7     df = pd.read_csv("data/cs1090a_survey_raw.csv")
8 except FileNotFoundError:
9     # 파일없을경우임시데이터사용위 ( 예제참고 )
10    data = {'Timestamp': ['9/9/2025 17:43:09'], 'What\'s your Harvard
11        affiliation?\n': ['Bachelor\'s (CS)'], 'Have you used Jupyter
12        Notebooks before?': ['Yes'], 'How many years of Python programming
13        experience do you have?\n(Choose the range that best fits your
14        experience.)': ['Less than 1 year'], 'Rate your current Pandas skill
15        level.': [2], 'What is your primary OS?': ['MacOS'], 'Do use normally
16        code/browse in dark mode?': ['No'], 'What languages do you speak? \n(
17        Comma separated)\n\nExample: Hittite, Elvish, Cornish, Klingon': [
18        'English'], 'Which continents have you visited?': ['Africa, Asia,
19        Europe, North America, South America'], 'When were you born?': [
20        '11/15/2004'], 'What time do you usually wake up in the morning?': [
21        '10:00:00 AM'], 'What time do you usually go to bed?': ['12:00:00 AM'],
22        'Favorite Season?': ['Spring'], 'Where do you usually get your
23        caffeine?': ['Tea'], 'Which kind of pet do you prefer?': ['Pet rock'],
24        'What\'s your favorite movie?': [None], 'What movie genres do
25        particularly enjoy?\n(select as many as you like)': ['Comedy'], 'List
26        up to 3 of your hobbies.\n(comma separated)\n\nExample: playing kazoo,
27        bird watching, stamp collecting': ['tennis, movies, eating'], 'How
28        was HW0?': [None]}
29 df = pd.DataFrame(data)
30
31 # --- 2. 데이터정제 ---
32 # 컬럼명변경
33 new_cols = ["timestamp", "program", "jupyter", "python_exp", "pandas_skill",
34             "os", "dark_mode", "languages", "continents", "dob", "wake_time",
35             "sleep_time", "fav_season", "caffeine", "pet", "fav_movie", "fav_genres",
36             "hobbies", "hw0"]
37 df.columns = new_cols
38
39 # 불필요한컬럼제거
40 df = df.drop("timestamp", axis=1)
41
42 # 데이터타입변환 (Boolean, Category, Datetime)
43 df['dark_mode'] = (df['dark_mode'].str.lower() == 'yes')
44 df['jupyter'] = (df['jupyter'].str.lower() == 'yes')
45 df['os'] = df['os'].astype('category')
46 experience_order = ['Less than 1 year', '1-2 years', '2-4 years', '4+ years']

```

```

26 experience_dtype = CategoricalDtype(categories=experience_order, ordered=True)
27 df['python_experience'] = df['python_exp'].astype(experience_dtype)
28 df['dob'] = pd.to_datetime(df['dob'], errors='coerce')
29
30 # 텍스트정규화소문자 (변환)
31 str_cols = df.select_dtypes(include='object').columns
32 for c in str_cols:
33     df[c] = df[c].str.lower().str.strip() # 공백제거추가
34
35 # 중복행제거있는 (경우)
36 df = df.drop_duplicates()
37
38 # 결측치처리예 (: 'languages'가 '비어있는행제거')
39 df.dropna(subset=['languages'], inplace=True)
40
41 # 파생변수생성예 (: 나이, 언어수)
42 now = pd.Timestamp.now()
43 df['age'] = np.floor((now - df['dob']).dt.days / 365.25).astype('Int64')
44 df['num_languages'] = df['languages'].str.split(', ').apply(len)
45
46 # --- 3. 데이터분석예시 ---
47 # Pandas 스킬 4 이상인학생필터링
48 df_high_skill = df[df['pandas_skill'] >= 4]
49
50 # 프로그램별평균나이계산학생 (수명 2 이상)
51 program_summary = df.groupby('program').agg(avg_age=('age', 'mean'), count=('program', 'size'))
52 program_summary_filtered = program_summary[program_summary['count'] >= 2].sort_values('avg_age')
53
54 # --- 4. 결과확인 ---
55 print("--- Pandas 스킬 4 이상학생일부 () ---")
56 print(df_high_skill[['program', 'pandas_skill']].head())
57
58 print("\n--- 주요프로그램별평균나이및학생수 ---")
59 print(program_summary_filtered)
60
61 # --- 5. 데이터저장 ---
62 # df.to_csv('survey_final.csv', index=False)
63 # print("\n정제된 n 데이터가 'survey_final.csv'로 저장되었습니다.")

```

Listing 31: Pandas 데이터 처리 파이프라인 예제

6 체크리스트

Pandas를 사용하여 데이터를 처리할 때 다음 사항들을 점검하면 실수를 줄이고 효율적인 분석을 수행하는 데 도움이 됩니다.

- ✓ **데이터 로딩:** 데이터가 올바르게 DataFrame으로 로드되었는가? (`head()`, `shape` 확인)
- ✓ **컬럼명 확인 및 변경:** 컬럼명이 너무 길거나 복잡하지 않은가? 의미 있고 사용하기 쉬운 이름으로 변경했는가? (소문자, 언더스코어 사용 권장)
- ✓ **불필요한 데이터 제거:** 분석에 사용하지 않을 행이나 열은 제거했는가? (`drop`)
- ✓ **데이터 타입 확인 및 변환:** 각 컬럼의 데이터 타입(`dtypes`, `info`)이 적절한가? 숫자여야 할 컬럼이 문자열(`object`)은 아닌가? 날짜, 범주형, boolean 등으로 변환이 필요한 컬럼은 없는가? (`astype`, `to_datetime`, `CategoricalDtype`)
- ✓ **텍스트 정규화:** 문자열 데이터에 대소문자나 앞뒤 공백 등 일관성 없는 부분이 있는가? (`str.lower()`, `str.strip()`, `replace`)
- ✓ **중복값 확인 및 처리:** 완전히 동일한 행이 중복되어 있는가? (`duplicated()`, `drop_duplicates`)
- ✓ **결측치 확인 및 처리:** 결측치(NaN)가 어디에 얼마나 있는지 파악했는가? (`isna().sum()`, `info`) 분석 목적에 맞게 결측치를 제거(`dropna`)하거나 적절한 값으로 대치(`fillna`)했는가?
- ✓ **인덱스 확인 및 재설정:** 데이터 필터링, 정렬 후 인덱스가 순차적이지 않게 되었는가? 필요하다면 인덱스를 재설정했는가? (`reset_index`)
- ✓ **데이터 선택/필터링 검증:** Boolean indexing, `loc`, `iloc` 등을 사용하여 원하는 데이터를 정확히 선택했는지 확인했는가?
- ✓ **집계 결과 검증:** `groupby`, `agg`, `crosstab` 등의 집계 결과가 예상과 일치하는가? 집계 함수를 올바르게 사용했는가?
- ✓ **결과 저장:** 최종적으로 정제되거나 분석된 데이터를 저장할 필요가 있는가? (`to_csv`)

7 FAQ (자주 묻는 질문)

Pandas를 처음 배울 때 흔히 궁금해하거나 혼동하는 부분들을 정리했습니다.

Q1: df[column_name] 과 df.column_name 중 어떤 것을 써야 하나요?

- df['column_name'] (대괄호와 문자열 사용) 방식이 더 안전하고 권장됩니다. 컬럼 이름에 공백이나 특수문자가 있거나, DataFrame의 기존 메서드 이름(예: 'head', 'count')과 겹치는 경우에도 사용할 수 있습니다.
- df.column_name (점 표기법) 방식은 타이핑이 간편하지만, 위와 같은 제약 조건이 있습니다. 간단하고 이름 충돌 가능성이 없는 경우에만 사용하는 것이 좋습니다.

Q2: loc과 iloc은 언제 사용하나요? 왜 이렇게 복잡하게 나뉘어 있나요?

- 데이터를 선택하는 기준이 명확히 달릅니다. loc은 사람이 지정한 이름표(label)를 사용하고, iloc은 컴퓨터가 내부적으로 관리하는 순서(position, 0부터 시작하는 정수)를 사용합니다.
- 예를 들어, 학생 명단에서 '홍길동' 학생의 정보를 찾을 때는 이름표(loc['홍길동'])를 쓰는 것이 직관적입니다. 반면, 명단에서 그냥 '첫 번째 학생'의 정보를 볼 때는 위치(iloc[0])를 쓰는 것이 편합니다.
- 데이터를 정렬하거나 필터링하면 원래의 순서와 이름표가 달라질 수 있기 때문에, 이 두 가지 방법을 명확히 구분하여 사용해야 원하는 데이터를 정확히 찾을 수 있습니다.

Q3: 데이터를 정렬하거나 필터링한 후에 왜 reset_index()를 자주 사용하나요?

- sort_values()나 boolean indexing으로 DataFrame을 조작하면, 기존의 인덱스 라벨은 그대로 유지된 채 행의 순서만 바뀝니다. 예를 들어, 원래 100번 인덱스였던 행이 정렬 후 첫 번째 행이 되어도 인덱스 라벨은 여전히 100입니다.
- 이렇게 되면 iloc[0] (첫 번째 위치의 행)과 loc[0] (인덱스 라벨이 0인 행)이 다른 행을 가리키게 되어 혼란을 야기할 수 있습니다.
- reset_index(drop=True)를 사용하면 현재 행 순서에 맞게 인덱스를 0부터 다시 깔끔하게 부여해주세요. 이후 작업에서 혼동을 줄일 수 있습니다. (drop=False로 하면 기존 인덱스가 새로운 컬럼으로 추가됩니다.)

Q4: 컬럼의 데이터 타입(dtype)이 왜 중요한가요? 그냥 object로 두면 안 되나요?

- 성능: 숫자(int, float), boolean(bool) 등 명확한 타입은 메모리를 효율적으로 사용하고 계산 속도도 빠릅니다. 반면 object 타입은 다양한 종류의 데이터를 담을 수 있지만, 내부적으로는 메모리 주소를 저장하는 방식이라 처리 속도가 느리고 메모리도 많이 차지합니다.
- 기능: 날짜/시간 타입(datetime64)으로 변환해야 날짜 관련 연산(예: 두 날짜 간의 차이 계산)이 가능합니다. 범주형 타입(category)은 메모리를 절약하고 특정 통계 분석에 유용합니다. 숫자 타입이어야 평균, 합계 등 수학적 연산이 가능합니다.
- 정확성: '1', '2', '3'이 문자열(object)로 저장되어 있다면 숫자 크기 비교나 덧셈이 제대로 되지 않습니다. 반드시 적절한 숫자 타입으로 변환해야 합니다.

Q5: NumPy 배열과 Pandas Series/DataFrame은 무엇이 다른가요?

- **NumPy 배열(ndarray):** 주로 숫자로 이루어진 다차원 배열을 효율적으로 다루기 위한 라이브러리입니다. 모든 원소는 동일한 데이터 타입이어야 하며, 인덱스는 0부터 시작하는 정수 위치만 사용합니다. 수학/과학 계산에 강력합니다.

- **Pandas Series/DataFrame:** NumPy를 기반으로 만들어졌지만, 더 유연하고 다양한 기능을 제공합니다.
 - 명시적 인덱스: 숫자가 아닌 라벨 기반 인덱스를 사용할 수 있습니다.
 - 다양한 데이터 타입: DataFrame의 경우 열마다 다른 데이터 타입을 가질 수 있습니다.
 - 결측치 처리: NaN 값을 내장하여 쉽게 처리할 수 있습니다.
 - 데이터 조작 기능: 데이터 정렬, 필터링, 그룹화, 병합 등 데이터 분석에 특화된 고수준의 기능을 풍부하게 제공합니다.
 - 입출력 편의성: CSV, Excel 등 다양한 파일 형식을 쉽게 다룰 수 있습니다.
- 요약하면, NumPy는 고성능 수치 계산의 기반이고, Pandas는 NumPy를 활용하여 표 형태의 데이터를 편리하게 분석할 수 있도록 확장한 도구입니다.

8 빠르게 훑어보기 (1페이지 요약)

Pandas 핵심 요약

Pandas? 파이썬에서 표(테이블) 데이터를 다루는 강력하고 편리한 라이브러리.

주요 자료구조

- **Series:** 1차원 배열 + 인덱스(이름표). 엑셀의 한 열. `pd.Series(data, index=...)`
- **DataFrame:** 2차원 표. 여러 Series의 묶음. 엑셀 시트. `pd.DataFrame(data, columns=...)`

데이터 로딩 & 저장

- 읽기: `pd.read_csv('파일경로')` (주로 사용), `pd.read_excel()`, ...
- 저장: `df.to_csv('파일경로', index=False)`

기본 탐색

- `df.head(n)`: 처음 n개 행 보기
- `df.tail(n)`: 마지막 n개 행 보기
- `df.shape`: (행 개수, 열 개수) 확인
- `df.columns`: 열 이름 목록
- `df.index`: 행 이름(인덱스) 목록
- `df.info()`: 전체 요약 정보 (결측치, 타입 확인 필수!)
- `df.dtypes`: 열별 데이터 타입 확인
- `df.describe()`: 수치형 데이터 기술 통계 요약

데이터 정제

- 컬럼명 변경: `df.columns = [...], df.rename(columns=...)`
- 컬럼/행 제거: `df.drop(['col1', 'row1'], axis=1/0)`
- 타입 변환: `df['col'].astype('int'), pd.to_datetime(df['col'])`, ...
- 결측치 확인/처리: `df.isna().sum(), df.dropna(), df.fillna(value)`
- 중복값 확인/처리: `df.duplicated().sum(), df.drop_duplicates()`
- 텍스트 처리: `df['col'].str.lower(), .str.strip(), .str.replace(), .str.split()`

데이터 선택 & 필터링

- 열 선택: `df['col'], df[['col1', 'col2']]`
- 행 선택 (Boolean Indexing): `df[df['col'] > 10], df[(cond1) & (cond2)]`
- 라벨 기반 선택 (loc): `df.loc['row_label', 'col_label'], df.loc['start':'end']` (끝 포함)
- 위치 기반 선택 (iloc): `df.iloc[0, 1], df.iloc[0:5]` (끝 제외)

정렬 & 집계

- 정렬: `df.sort_values(by='col', ascending=False)`
- 값 개수: `df['col'].value_counts()`
- 그룹화: `df.groupby('col')`
- 집계: `grouped.agg('col1': 'mean', 'col2': 'count'), grouped.mean(), grouped.size()`
- 교차 분석: `pd.crosstab(df['col1'], df['col2'])`

9 부록: 환경 설정 및 추가 정보

9.1 Python 환경 설정

Pandas를 사용하기 위해서는 Python과 Pandas 라이브러리가 설치되어 있어야 합니다. 강의에서는 가상 환경 사용을 권장하며, uv 또는 conda와 같은 도구를 사용할 수 있습니다.

uv 사용 (권장): 과제 파일과 함께 제공되는 `requirements.txt` 파일을 이용하여 필요한 라이브러리가 모두 포함된 가상 환경을 생성할 수 있습니다.

1. uv 설치 (`pip install uv`)
2. 프로젝트 폴더에서 `uv venv` 실행하여 가상 환경 생성 (.venv 폴더 생성됨)
3. `source .venv/bin/activate` (macOS/Linux) 또는 `.venv\Scripts\activate` (Windows) 실행하여 가상 환경 활성화
4. `uv pip install -r requirements.txt` 실행하여 라이브러리 설치

각 과제(프로젝트)마다 별도의 가상 환경을 만드는 것이 좋습니다.

Conda 사용: Conda 환경을 사용하고 있다면, `conda install pandas` 명령어로 Pandas를 설치하거나, `requirements.txt` 파일을 이용하여 환경을 구성할 수 있습니다.

직접 설치: 기존 환경에 직접 설치하려면 `pip install pandas` 또는 `uv pip install pandas` 명령어를 사용합니다. 과제 진행 중 필요한 라이브러리가 없다는 오류가 발생하면 해당 라이브러리를 추가로 설치해주어야 합니다.

9.2 Harvard OnDemand (클라우드 환경)

로컬 환경 설정에 어려움을 겪는 경우, Harvard OnDemand에서 제공하는 클라우드 기반 JupyterLab 환경을 사용할 수 있습니다.

- Canvas의 'Harvard OnDemand' 링크를 통해 접속합니다.
- 'Interactive Apps' 메뉴에서 'JupyterLab CS1090A'를 선택합니다.
- 사용 시간 등을 설정하고 'Launch' 버튼을 클릭합니다.
- 잠시 후 생성된 환경에 접속하여 Jupyter 노트북을 사용합니다.
- 과제 파일이나 강의 자료(zip 파일)를 업로드하여 작업할 수 있습니다.
- 사용 시간이 만료되어도 작업 내용은 유지되므로, 다시 접속하여 이어서 작업할 수 있습니다.

주의사항

Harvard OnDemand는 제한된 자원이므로, 로컬 환경 설정이 가능한 경우에는 로컬 환경 사용을 권장합니다. 꼭 필요한 경우의 안전망으로 활용하세요.

9.3 Jupyter Notebook 사용 팁

- **Tab 자동 완성:** 코드 입력 중 Tab 키를 누르면 사용 가능한 변수, 함수, 메서드 목록을 보여주거나 자동 완성해줍니다. (df. 입력 후 Tab)
- **Shift + Tab 도움말:** 함수 괄호 안에서 Shift + Tab 키를 누르면 해당 함수의 설명(docstring)을

보여줍니다. 인자나 사용법을 확인할 때 유용합니다.

- **셀 실행 순서 주의:** 노트북 셀은 원하는 순서대로 실행할 수 있지만, 이로 인해 변수 상태가 꼬여 예상치 못한 오류가 발생할 수 있습니다. 문제가 발생하면 커널을 재시작(Kernel > Restart)하고 처음부터 순서대로 실행해보는 것이 좋습니다.

9.4 추가 학습 자료

- **Pandas 공식 문서:** <https://pandas.pydata.org/docs/> (가장 정확하고 방대한 정보 제공)
- **Pandas 10분 완성:** https://pandas.pydata.org/docs/user_guide/10min.html (빠른 시작 가이드)
- **강의 플랫폼(Ed) 자료:** Bedrock Data Science, Bedrock Codecasts & Tutorials 섹션의 추가 Pandas 자료 확인

December 10, 2025

- 강의명: CS109A: 데이터 과학 입문
- 주차: Lecture 04
- 교수명: Pavlos Protopapas, Kevin Rader, Chris Gumb
- 목적: Lecture 04의 핵심 개념 학습

Contents

1	개요	2
2	핵심 용어 정리	2
3	통계 모델링 시작하기	4
3.1	예측이란 무엇인가?	4
3.2	데이터의 두 가지 역할: 예측 변수(X)와 반응 변수(y)	4
3.3	데이터를 '행렬'로 표현하기 (The Design Matrix)	4
4	모델이란 무엇인가?	6
4.1	완벽한 모델 vs. 통계 모델	6
4.2	모델의 두 가지 목적: 추론(Inference) vs. 예측(Prediction)	6
5	k-최근접 이웃 (kNN) 알고리즘	7
5.1	가장 단순한 모델: "평균" 모델	7
5.2	kNN의 핵심 아이디어: "가까운 이웃" 참고하기	7
5.3	k 값에 따른 모델의 변화	7
5.4	k는 '하이퍼파라미터'입니다	8
6	모델 평가하기 (Error Evaluation)	9
6.1	"최고의" 모델이란 무엇인가?	9
6.2	데이터를 나누는 이유: 훈련, 검증, 테스트	9
6.3	손실 함수 (Loss Function): 오류를 숫자로 나타내기	9
7	최고의 모델 선택 및 최종 평가	11
7.1	검증 세트를 이용한 k 값 선택	11
7.2	모델이 "쓸 만한지" 평가하기: R^2 (R-squared)	11
8	kNN의 한계: 차원의 저주	13

8.1 kNN을 다차원으로 확장하기	13
8.2 ”차원의 저주”란 무엇인가?	13
9 학습 목표 요약 (Checklist)	14

1 개요

이 문서는 통계적 모델링의 기초, 특히 **k-최근접 이웃(kNN) 회귀** 모델에 대해 다룹니다.

우리의 목표는 'TV 광고 예산'과 같은 데이터를 사용하여 '제품 판매량'을 예측하는 것입니다.

kNN은 "가장 유사한 과거 사례(이웃)를 찾아 그들의 평균값으로 예측한다"는 매우 직관적인 아이디어에서 출발합니다.

모델을 만든 후에는 이 모델이 얼마나 '좋은지' 평가해야 합니다. 이를 위해 데이터를 **훈련/검증/테스트** 세트로 나누고, **평균 제곱 오차(MSE)**와 같은 '손실 함수'를 사용하여 오류를 측정합니다.

마지막으로, 여러 k값 (예: 이웃 1명, 10명, 70명) 중에서 '최적의' 모델을 선택하고, **R-squared (R^2)** 값을 통해 이 모델이 "단순히 평균을 추측하는 것보다 얼마나 더 나은지"를 객관적으로 평가합니다.

2 핵심 용어 정리

모델링을 학습하기 위해 자주 사용되는 기본 용어들을 정리했습니다.

용어	쉬운 설명 (직관)	원어	비고 (예시)
반응 변수	우리가 예측 하려는 '결과' 값입니다.	Response Variable (y)	제품 판매량, 주택 가격
예측 변수	예측을 위해 사용하는 '입력' 데이터입니다.	Predictor Variable (X)	TV/라디오 광고 예산
설계 행렬	모든 예측 변수 데이터를 모아 둔 행렬 ($n \times p$ 크기).	Design Matrix (X)	$n=관측치 수, p=예측 변수 수$
통계 모델	데이터 (X)와 결과 (y) 사이의 관계를 공식화하려는 시도.	Statistical Model (\hat{f})	완벽한 관계 f 를 추정 (\hat{f}) 합니다.
예측 (Prediction)	모델을 사용해 X 가 주어졌을 때 y 의 값을 맞추는 것.	Prediction	정확한 '값'을 맞추는 것이 목표.
추론 (Inference)	X 와 y 가 '어떤 관계'인지 이해하는 것.	Inference	'TV 예산이 1달러 오르면 판매량이 얼마나 오르는가?'에 대한 답.
비모수	모델의 형태를 미리 정하지 않고 데이터에 의존하는 방식.	Non-parametric	kNN 이 대표적. 유연합니다.
하이퍼파라미터	모델이 학습하는 것이 아니라, *사람*이 미리 정해주는 값.	Hyperparameter	kNN 에서의 k 값 (이웃수).
훈련/검증/테스트	데이터를 세 조각으로 나누는 것.	Train / Validation / Test	훈련: 모델 학습 (연습문제) 검증: 모델 선택 (모의고사) 테스트: 최종 성능 평가 (실제 시험)
손실 함수	모델이 얼마나 틀렸는지 (오류) 계산하는 함수.	Loss Function	이 값이 낮을수록 좋은 모델입니다.
MSE	(실제 값 - 예측 값)을 제곱하여 평균 낸 값.	Mean Squared Error	가장 널리 쓰이는 손실 함수.
R^2 (결정 계수)	모델이 "단순 평균"보다 얼마나 예측을 잘하는지 (0~1).	R-squared	1에 가까울수록 좋습니다. 0이면 평균과 같습니다.
차원의 저주	예측 변수 (p) 가 너무 많아질 때 발생하는 문제.	Curse of Dimensionality	데이터가 희박해져 kNN 성능이 저하됩니다.

3 통계 모델링 시작하기

3.1 예측이란 무엇인가?

우리는 종종 하나의 변수 값을 예측하기 위해 다른 변수들을 사용합니다.

- 예시 1: 동영상의 길이, 게시 날짜 등을 바탕으로 다음 주 TikTok 조회수 예측하기.
- 예시 2: 사용자의 이전 영화 평점, 인구통계학적 데이터를 바탕으로 Netflix 사용자가 높게 평가할 영화 예측하기.

이 강의에서는 간단한 예제인 광고 데이터셋을 사용합니다. 200개 시장에서 제품 판매량(Sales) 데이터와, 3개 매체(TV, 라디오, 신문)의 광고 예산(\$1,000 단위) 데이터를 가지고 있습니다.

우리의 목표: TV, 라디오, 신문 광고 예산을 알 때, 판매량을 예측하는 모델을 만드는 것입니다.

3.2 데이터의 두 가지 역할: 예측 변수(X)와 반응 변수(y)

데이터에는 비대칭성이 존재합니다. 우리가 예측하려는 '판매량'은 다른 변수들(광고 예산)에 의해 영향을 받거나, 측정하기 더 어렵거나, 더 중요할 수 있습니다.

- 반응 변수 (Response Variable, y): 우리가 예측하려는 목표 변수입니다.
 - 다른 이름: 종속 변수(Dependent Variable), 타겟(Target), 결과(Outcome).
 - 예: sales (판매량)
- 예측 변수 (Predictor Variables, X): 예측을 위해 사용하는 입력 변수들입니다.
 - 다른 이름: 독립 변수(Independent Variable), 특성(Features), 공변량(Covariates).
 - 예: TV, radio, newspaper (각 매체별 광고 예산)

데이터는 보통 n 개의 행(관측치)과 p 개의 열(예측 변수)로 구성됩니다.

3.3 데이터를 '행렬'로 표현하기 (The Design Matrix)

데이터를 수학적으로 다루기 위해 다음과 같이 표기합니다.

- X (설계 행렬, Design Matrix): n 개의 관측치(행)와 p 개의 예측 변수(열)를 가진 행렬입니다.
- y (반응 벡터, Response Vector): n 개의 관측치에 대한 n 개의 반응 변수 값을 가진 벡터입니다.

□ 예제:

데이터의 형태: X 와 y 만약 5개의 관측치($n = 5$)와 3개의 예측 변수($p = 3$)가 있다면 데이터는 다음과 같습니다.

	TV	radio	newspaper		sales
X (설계 행렬, 5×3)	230.1	37.8	69.2		22.1
	44.5	39.3	45.1	y (반응 벡터, 5×1)	10.4
	17.2	45.9	69.3		9.3
	151.5	41.3	58.5		18.5
	180.8	10.8	58.4		12.9

주의사항

Pandas와 Sklearn에서의 데이터 차원 (Shape) 데이터 분석 도구(Pandas, Sklearn)를 다룰 때, 벡터의 차원에 유의해야 합니다.

- `X.shape` (설계 행렬): 항상 (n, p) 형태의 2차원입니다. (예: $(5, 3)$)
- `y.shape` (반응 벡터): $(n,)$ 또는 $(n, 1)$ 형태일 수 있습니다.
 - $(n,)$: 1차원 벡터 (Pandas의 **Series**). 예: `df['sales']`
 - $(n, 1)$: 2차원 행렬 (Pandas의 **DataFrame**). 예: `df[['sales']]`
- 대부분의 머신러닝 라이브러리는 두 형태 모두를 받아들이지만, $(n,)$ 형태를 기본으로 하는 경우가 많습니다. 이 차이로 인해 오류가 발생하기 쉬우니 `.shape` 속성으로 항상 확인하는 습관을 들이는 것이 좋습니다.

4 모델이란 무엇인가?

4.1 완벽한 모델 vs. 통계 모델

데이터 X 와 y 사이의 관계를 찾는 것을 '모델링'이라고 합니다.

아이스크림 비유

- 진짜(True) 모델, f : 세상에 존재하는 '완벽한 아이스크림' 레시피. 모든 맛의 조합을 정확히 알고 있습니다. 하지만 우리는 이 레시피를 절대 알 수 없습니다.
- 통계(Statistical) 모델, \hat{f} : 우리가 가진 재료(데이터)로 그 '완벽한 아이스크림'을 따라 만들려는 * 시도*입니다.

우리는 X 와 y 사이에 어떤 '진짜' 관계 f 가 있다고 가정합니다.

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

- $f(X)$: X 에 의해 결정되는 Y 의 체계적인 정보 (우리가 찾고 싶은 완벽한 규칙).
- ϵ (엡실론): $f(X)$ 로 설명되지 않는 무작위 오차(Noise). 측정의 한계, 우리가 고려하지 못한 변수, 혹은 세상의 본질적인 무작위성 때문에 발생합니다.

통계 모델링이란, 우리가 가진 데이터(X, y)를 사용하여 이 알 수 없는 f 를 가장 잘 근사하는(따라하는) 함수 \hat{f} (f-hat)을 찾는 과정입니다.

4.2 모델의 두 가지 목적: 추론(Inference) vs. 예측(Prediction)

모델 \hat{f} 를 찾는 목적은 크게 두 가지로 나뉩니다.

구분	추론 (Inference)	예측 (Prediction)
목표	X 와 y 사이의 관계를 이해하는 것.	X 가 주어졌을 때 y 의 값을 정확히 맞추는 것.
비유	탐정	궁수 \square
주요 질문	"TV 예산이 판매량에 어떻게 영향을 미치는가?"	"TV 예산이 \$150k일 때 예상 판매량은 얼마인가?"
모델 형태	해석 가능한 단순한 모델 (예: 선형 회귀)	정확도 높은 유연한 모델 (예: kNN, 신경망)
모델(\hat{f}) 평가	\hat{f} 의 형태와 계수(parameter)가 중요한가? Yes	\hat{f} 의 형태는 중요하지 않음 (블랙박스여도 OK). No

이 강의에서는 먼저 이해하기 쉬운 **예측** 모델인 kNN부터 다룹니다.

5 k-최근접 이웃 (kNN) 알고리즘

5.1 가장 단순한 모델: ”평균” 모델

예측 변수 X 를 완전히 무시하고 예측하는 가장 단순한 모델은 무엇일까요? 바로 모든 y 값의 평균을 사용하는 것입니다.

$$\hat{y} = \bar{y} = \frac{1}{n} \sum_{i=1}^n y_i$$

이 모델은 TV 예산이 \$10k 이든 \$300k 이든 상관없이 항상 동일한 평균 판매량(예: 12.5)을 예측합니다. 이것은 매우 순진한(naive) 모델이지만, 나중에 다른 모델과 비교하기 위한 ’최소 기준선(Baseline)’ 역할을 합니다.

5.2 kNN의 핵심 아이디어: ”가까운 이웃” 참고하기

kNN은 평균 모델보다 훨씬 똑똑한, 매우 직관적인 아이디어에 기반합니다.

의사의 진단 비유 환자가 ”배가 아프다”고 하며 병원에 왔습니다. 의사는 머릿속으로 ’이번 주에 배가 아프다고 왔던 다른 환자 10명’을 떠올립니다. ”그 10명은 모두 특정 푸드트럭 음식을 먹고 장염에 걸렸었지. 이 환자도 장염일 가능성이 높다.” 즉, 나와 비슷한 사례(이웃)를 찾아 그들의 결과를 참고하여 현재 사례를 판단(예측)합니다.

kNN 알고리즘은 다음과 같이 작동합니다. (1차원 예시: TV 예산(x)만 사용)

1. 예측할 x_q 정하기: ”TV 예산이 \$150k 일 때(x_q) 판매량(y_q)은?”
2. 거리 계산: x_q 와 훈련 데이터에 있는 모든 x_i 사이의 거리를 계산합니다. (1차원에서는 $D(x_q, x_i) = |x_q - x_i|$)
3. k개의 이웃 찾기: x_q 와 가장 가까운 k 개의 데이터 포인트 (x_i, y_i) 를 찾습니다.
4. 평균 내기: 찾은 k 개 이웃들의 y_i 값들의 평균을 내어 \hat{y}_q 로 예측합니다.

$$\hat{y}_q = \frac{1}{k} \sum_{i \in \text{Neighbors}} y_{q_i}$$

5.3 k값에 따른 모델의 변화

k 값을 어떻게 정하느냐에 따라 모델의 형태가 극적으로 달라집니다.

- $k = 1$ (가장 가까운 1명):
 - 작동: 가장 가까운 이웃 1명의 y 값을 그대로 복사합니다.
 - 결과: 모델이 매우 뾰족하고 들쭉날쭉한 계단 형태가 됩니다. (Overfitting)
 - 문제점: 데이터의 아주 작은 잡음(noise)에도 민감하게 반응합니다. 변동성이 너무 큽니다.
- $k = 10$ (적당한 수의 이웃):
 - 작동: 가장 가까운 10명의 y 값을 평균냅니다.
 - 결과: $k = 1$ 보다 훨씬 부드러운 곡선(계단)이 되어 데이터의 전반적인 추세를 잘 따릅니다.
- $k = 70$ (또는 $k = n$, 아주 많은 이웃):
 - 작동: 모든 데이터의 y 값을 평균냅니다.
 - 결과: 모델이 매우 평坦한 계단 형태가 됩니다.

- **작동:** 훈련 데이터 70개 전부를 이웃으로 삼아 평균냅니다.
- **결과:** x_q 의 위치에 상관없이 항상 전체 데이터의 평균을 반환합니다.
- **문제점:** 5.1 절에서 본 ”가장 단순한 모델”과 같아집니다. 데이터의 지역적 특성을 완전히 무시합니다. (Underfitting)

$k = 1$ 은 너무 복잡하고, $k = 70$ 은 너무 단순합니다. 우리는 이 사이의 ”적절한” k 값을 찾아야 합니다.

5.4 k 는 ’하이퍼파라미터’입니다

kNN은 비모수(Non-parametric) 모델입니다. 이는 모델이 f 의 형태를 ’직선’이나 ’곡선’이라고 미리 가정하지 않고, 데이터 자체에 의존해 형태를 만들기 때문입니다.

하지만 kNN에도 k 라는 파라미터가 있습니다. 이 k 값은 모델이 데이터로부터 *학습하는 값*이 아니라, 우리가 *직접 정해줘야 하는 값*입니다. 이처럼 사람이 모델 학습 전에 직접 설정하는 값을 **하이퍼파라미터(Hyperparameter)**라고 부릅니다.

질문: $k = 1, k = 10, k = 70$ 중에서 ”최고의” k 는 어떻게 찾을 수 있을까요?

6 모델 평가하기 (Error Evaluation)

6.1 ”최고의” 모델이란 무엇인가?

k 값에 따라 수많은 모델이 나옵니다. ”최고의” 모델을 고르려면 ”좋다”는 것을 정의할 기준이 필요합니다. 우리는 모델의 예측 오류(Error)가 가장 작은 모델을 ”최고”라고 부를 것입니다.

6.2 데이터를 나누는 이유: 훈련, 검증, 테스트

모델의 오류를公正하게 평가하려면 데이터를 명확한 목적에 따라 나눠야 합니다.

시험 공부 비유

- **훈련 세트 (Train Set):** 모델을 학습시키는 데 사용합니다. (kNN에서는 이웃을 찾기 위한 데이터 풀)
 - 비유: 답안지가 있는 연습 문제집. 이걸로 공부합니다.
- **검증 세트 (Validation Set):** 학습된 모델들 중 최고의 하이퍼파라미터(k)를 선택하기 위해 사용합니다.
 - 비유: 모의고사. $k = 3$ 전략과 $k = 10$ 전략 중 모의고사 점수가 더 잘 나오는 전략을 선택합니다.
- **테스트 세트 (Test Set):** 선택된 최종 모델의 실제 성능을 딱 한 번 평가하기 위해 사용합니다.
 - 비유: 실제 수능 시험. 모의고사에 너무 최적화되었는지(과적합), 진짜 실력(일반화 성능)이 어느 정도인지 최종 평가합니다.

데이터 분할은 보통 무작위(Randomly shuffle)로 진행됩니다.

주의사항

데이터 오염 (Data Contamination) ”모델을 선택하기 위해(하이퍼파라미터 튜닝) 테스트 세트를 사용하는 사람들은 지옥의 특별한 자리가 마련되어 있습니다.” (강의 중 인용)

만약 모의고사(k 값 선택)에 수능 문제()를 미리 보고 푼다면, 그 점수는 진짜 실력이 아닙니다. 모델 선택(검증)에는 검증 세트만 사용하고, 테스트 세트는 최종 평가 전까지 절대 열어보지 않아야 합니다.

□ 예제:

scikit-learn의 `train_test_split` 파일 라이브러리인 scikit-learn은 데이터를 두 개로 조개는 `train_test_split()` 함수만 제공합니다. 훈련/검증/테스트 세 개로 나누려면 이 함수를 두 번 호출해야 합니다.

1. 전체 데이터를 `train_full`과 `test`로 나눕니다. (예: 80% / 20%)
2. 1번에서 나눈 `train_full`을 다시 `train`과 `validation`으로 나눕니다. (예: 75% / 25% → 전체의 60% / 20%)

6.3 손실 함수 (Loss Function): 오류를 숫자로 나타내기

검증 세트에서 모델의 오류를 계산하려면, ’오류’를 하나의 숫자로 표현해야 합니다.

- **잔차 (Residual):** 실제 값과 예측 값의 차이. $r_i = y_i - \hat{y}_i$
- **문제점:** 잔차를 그냥 더하면 양수 오류와 음수 오류가 상쇄되어 오류가 0처럼 보일 수 있습니다. (예: +5, -5 → 합 0)

- **해결책:** 오류를 모두 양수로 만들기 위해 제곱하거나 절댓값을 사용합니다.

주요 손실 함수

1. 평균 제곱 오차 (Mean Squared Error, MSE)

- **공식:** $MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$
- **특징:** 오류를 '제곱' 하므로, 큰 오류(Outlier)에 더 큰 패널티를 줍니다. 수학적으로 미분이 가능하여 최적화에 유리하며, 확률론적 정당성(오차가 정규분포를 따른다는 가정 하에 최적)이 있어 가장 널리 쓰입니다.

2. 평균 절대 오차 (Mean Absolute Error, MAE)

- **공식:** $MAE = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i|$
- **특징:** 오류의 '절댓값' 을 사용합니다. MSE보다 큰 오류에 덜 민감합니다.

3. 평균 제곱근 오차 (Root Mean Squared Error, RMSE)

- **공식:** $RMSE = \sqrt{MSE}$
- **특징:** MSE에 제곱근을 씌운 것입니다. 오류의 단위가 원래 y 의 단위(예: $\2 가 아닌 $\$$)와 같아져 해석이 더 직관적입니다.

7 최고의 모델 선택 및 최종 평가

7.1 검증 세트를 이용한 k값 선택

이제 우리는 “최고의 k ”를 찾는 명확한 절차를 갖게 되었습니다.

1. k 의 후보 리스트를 정합니다. (예: $k = 1, 2, 3, \dots, 10$)
2. 각 k 값에 대해 다음을 반복합니다:
 - 훈련 세트를 사용하여 kNN 모델을 구성합니다.
 - 검증 세트의 X_{val} 값을 입력하여 \hat{y}_{val} 을 예측합니다.
 - 검증 세트의 실제 y_{val} 과 예측 \hat{y}_{val} 사이의 MSE를 계산합니다.
3. 모든 k 값 중에서 검증 세트의 MSE가 가장 낮은 k 를 “최적의 하이퍼파라미터”로 선택합니다.

예를 들어, k 값에 따른 검증 MSE가 아래와 같다고 가정합시다. (그래프로 그리면 U자 형태가 나옴)

k Nearest Neighbors	Validation MSE
1	7.0
2	3.0
3	0.2
4	0.8
...	...
10	5.2

이 경우, 검증 MSE가 0.2로 가장 낮은 $k = 3$ 을 최고의 모델로 선택합니다.

주의사항

무작위 분할의 합정 만약 우리가 우연히 운이 나쁘게 훈련/검증 세트를 분할했다면 어떨까요? $k = 3$ 이 이 특정 분할에서는 최고였지만, 다른 분할에서는 $k = 5$ 가 최고일 수도 있습니다. 이러한 분할의 불안정성(Variance) 문제는 이후 교차 검증(Cross-Validation)과 같은 고급 기법으로 다루게 됩니다.

7.2 모델이 ”쓸 만한지” 평가하기: R^2 (R-squared)

우리는 $k = 3$ 이 ”우리가 가진 모델 중 최고”라는 것을 알아냈습니다. 하지만 이 최고 모델이 ”쓸 만한” 모델일까요?

NBA 선수 선발 비유 제가 ”저는 우리 학과 교수팀에서 최고의 농구 선수입니다!” ($\leftarrow k = 3$ 이 다른 k 보다 낫다)라고 말한다고 해서, NBA 팀이 저를 스카웃해야 할까요?

아닙니다. NBA 팀은 ”그래서 르브론 제임스나 다른 NBA 선수들과 비교하면 어떻습니까?” (\leftarrow 절대적인 기준선과 비교)라고 물어볼 것입니다.

$k = 3$ 모델의 MSE가 5.0이라고 합시다. 이 $MSE = 5.0$ 은 좋은 값일까요? 알 수 없습니다. y 의 단위에 따라 이 값은 달라집니다.

- Sales 단위를 ’1,000 units’에서 ’single units’로 바꾸면 (y 값이 1000배 커짐)

- MSE는 1000^2 배 커져 5.0 \rightarrow 5,000,000이 됩니다.

MSE 값 자체는 단위에 의존하므로, 모델이 ”얼마나 좋은지” 직관적으로 말해주지 못합니다. 우리는 단위에 의존하지 않는, 0 점(최악)에서 1 점(최고) 사이의 표준화된 점수가 필요합니다. 이것이 바로 R^2 (**R-squared**, 결정 계수)입니다.

R^2 (R-squared) R^2 는 우리의 모델(\hat{f})이 ”가장 단순한 모델”(평균 모델, \bar{y})에 비해 얼마나 오류를 줄였는지를 비율로 나타냅니다.

- 기준선 (최악):** 평균 모델의 오류 $\sum(y_i - \bar{y})^2$
- 우리 모델:** 우리 모델의 오류 $\sum(y_i - \hat{y}_i)^2$

$$R^2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y}_i)^2} = 1 - \frac{MSE(\text{우리 모델})}{MSE(\text{평균 모델})}$$

해석:

- $R^2 = 1$: 완벽한 모델 ($MSE(\text{모델}) = 0$).
- $R^2 = 0$: 우리 모델이 단순 평균 모델과 성능이 정확히 같음 (쓸모 없음).
- $R^2 < 0$ (음수): 우리 모델이 단순 평균 모델보다 더 나쁨. (훈련 세트에서는 거의 발생하지 않지만, 테스트 세트에서는 발생 가능)

8 kNN의 한계: 차원의 저주

8.1 kNN을 다차원으로 확장하기

지금까지는 'TV 예산' ($p = 1$) 하나만 사용했습니다. 만약 'TV', 'radio', 'newspaper' ($p = 3$)를 모두 사용하려면 어떻게 해야 할까요?

kNN의 핵심은 '거리'입니다. 다차원 공간에서 두 점 사이의 거리를 계산하면 됩니다. 가장 일반적인 방법은 유클리드 거리 (Euclidean distance)입니다. (두 점을 잇는 직선 거리, 피타고라스 정리의 확장)

두 점 $x = (x_1, \dots, x_p)$ 와 $x' = (x'_1, \dots, x'_p)$ 사이의 거리는 다음과 같습니다:

$$d(x, x') = \sqrt{\sum_{j=1}^p (x_j - x'_j)^2}$$

8.2 "차원의 저주"란 무엇인가?

kNN은 직관적이고 강력하지만, 예측 변수의 수 (p), 즉 차원 (Dimension)이 증가하면 심각한 문제에 직면합니다. 이를 차원의 저주 (Curse of Dimensionality)라고 합니다.

주의사항

차원이 높아질 때 kNN이 겪는 문제들

1. 데이터가 희박해집니다 (Sparse Data):

- 비유:** 학생 100명이 1차원(복도)에서 있으면 매우 빼빼합니다. 2차원(운동장)에서 있으면 들판처럼 희박해집니다. 3차원(체육관)에 퍼져 있으면 훨씬 더 희박합니다. 100차원 공간에서는 100명의 학생이 있어도 공간 대부분이 텅 비게 됩니다.
- 영향:** 데이터가 희박해지면, 어떤 점 x_q 에서 "가장 가까운 이웃"조차도 실제로는 엄청나게 멀리 떨어져 있게 됩니다. "이웃"이라는 개념 자체가 무의미해집니다.

2. 거리의 변별력이 사라집니다:

- 현상:** 차원이 매우 높아지면, 한 점에서 다른 모든 점까지의 거리가 거의 비슷해지는 현상이 발생합니다.
- 영향:** "가까운 이웃"과 "먼 이웃"을 구분하기 어려워집니다.

3. 특성 스케일링 (Feature Scaling) 문제:

- 문제:** 유클리드 거리는 값의 스케일에 매우 민감합니다.
- 예시:** 예측 변수로 '키'(150~190cm)와 '연봉'(30,000,000~100,000,000원)을 사용한다고 합시다. 거리 계산 $(x_j - x'_j)^2$ 에서, '연봉'의 차이(수백만)가 '키'의 차이(수십)를 완전히 압도해버립니다. 사실상 '키'는 무시되고 '연봉'만으로 이웃을 찾게 됩니다.
- 해결책:** kNN을 사용하기 전, 모든 예측 변수 (X)를 동일한 범위 (예: 0~1 사이)로 스케일링 (Scaling) 또는 정규화 (Normalization)하는 전처리가 필수적입니다.

결론: kNN은 차원이 낮을 때 (p 가 작을 때) 매우 효과적이고 직관적인 모델이지만, 차원이 높은 데이터에는 적합하지 않을 수 있습니다.

9 학습 목표 요약 (Checklist)

이 강의를 통해 다음을 수행할 수 있어야 합니다.

- 반응 변수(y)와 예측 변수(X), 그리고 설계 행렬을 정의 할 수 있다.
- 통계 모델링의 두 가지 목적(추론, 예측)의 차이점을 설명할 수 있다.
- k-최근접 이웃(kNN) 알고리즘이 비모수적 방식이며 ”유사한 사례”에 의존함을 설명할 수 있다.
- 1차원 데이터에서 kNN의 이웃을 찾고, 그들의 값을 평균내어 예측값을 계산할 수 있다.
- k 값이 모델의 유연성(복잡도)에 어떤 영향을 미치는지 설명할 수 있다 ($k = 1$ vs $k = n$).
- 모델 평가를 위해 데이터를 훈련/검증/테스트 세트로 나누는 이유를 설명할 수 있다.
- MSE, RMSE 등 오류 측정 지표(손실 함수)를 사용하여 모델의 성능을 정량화할 수 있다.
- 검증 세트의 MSE를 사용하여 최적의 하이퍼파라미터(k)를 선택하는 과정을 설명할 수 있다.
- MSE가 단위에 의존하는 문제를 설명하고, R^2 가 왜 필요한지 설명할 수 있다.
- R^2 의 값을 (0, 1, 음수) 해석할 수 있다.
- 유clidean 거리를 사용하여 kNN을 다차원 데이터로 확장하는 방법을 설명할 수 있다.
- 차원의 저주가 무엇인지, 그리고 특성 스케일링이 왜 kNN에 필수적인지 설명할 수 있다.

December 10, 2025

- 강의명: CS109A: 데이터 과학 입문
- 주차: Lecture 05
- 교수명: Pavlos Protopapas, Kevin Rader, Chris Gumb
- 목적: Lecture 05의 핵심 개념 학습

Contents

I	선형 회귀의 기초	1
1	개요 (Overview)	2
2	핵심 용어 정리	2
3	왜 선형 회귀를 사용할까요?	2
4	단순 선형 회귀 (Simple Linear Regression, SLR)	3
4.1	모델 정의: ”최적의 직선 찾기”	3
4.2	최적의 선 찾기: 손실 함수 (Loss Function)	3
4.3	최적화: 손실 최소화하기	4
II	다중 선형 회귀로의 확장	5
5	다중 선형 회귀 (Multi-Linear Regression, MLR)	6
5.1	모델 정의: ”최적의 초평면 찾기”	6
5.2	행렬 표기법 (Matrix Notation)	6
5.3	최적화: 다중 회귀의 정규 방정식	7
III	모델 활용 및 해석	7
6	Python scikit-learn을 이용한 실습	8
7	모델 파라미터 해석하기	8
8	모델 정확도를 위한 고려사항	9
8.1	스케일링 (Scaling)	9

8.2 다중공선성 (Collinearity)	9
8.3 범주형 예측 변수 (Qualitative Predictors)	10
IV 학습 점검	10
9 핵심 학습 체크리스트	11
10 초심자 FAQ	11
11 빠르게 훑어보기 (1-Page Summary)	12

Part I

선형 회귀의 기초

1 개요 (Overview)

선형 회귀(Linear Regression)는 데이터 과학과 기계 학습에서 가장 기본이 되는 핵심 모델입니다.

이 문서는 선형 회귀의 기초부터 실제 적용까지 초심자도 완벽히 이해할 수 있도록 구성되었습니다.

▣ 핵심 요약

이 문서의 핵심 요약:

- 선형 회귀란?** 하나 이상의 입력 변수(예측 변수)와 하나의 연속적인 출력 변수(반응 변수) 사이의 선형(직선) 관계를 모델링하는 기법입니다.
- 왜 중요한가?** 복잡한 모델(신경망 등)을 이해하는 기초가 되며, ”어떤 변수가 결과에 얼마나 영향을 미치는지” 해석하기 용이합니다.
- 학습 흐름:**
 - 단순 선형 회귀 (SLR):** 하나의 입력(X)으로 하나의 출력(Y)을 예측합니다. ($Y = \beta_0 + \beta_1 X$)
 - 다중 선형 회귀 (MLR):** 여러 개의 입력(X_1, X_2, \dots)으로 하나의 출력(Y)을 예측합니다. ($Y = \beta_0 + \beta_1 X_1 + \dots$)
 - 모델 학습:** ’최적의 선’을 찾기 위해 평균 제곱 오차(MSE)라는 손실 함수를 최소화합니다.
 - 모델 해석 및 함정:** 계수(coefficient)의 의미, 스케일링, 다중공선성, 범주형 변수 처리 방법을 배웁니다.

2 핵심 용어 정리

선형 회귀를 이해하기 위해 다음 용어들을 먼저 숙지해야 합니다.

Table 1: 선형 회귀 핵심 용어

용어	원어	쉬운 설명	비고
반응 변수	Response Variable	우리가 예측하려는 결과값 (Y)	종속 변수(Dependent Variable)라고도 함
예측 변수	Predictor Variable	결과를 예측하는 데 사용하는 입력값 (X)	특성(Feature), 독립 변수라고도 함
계수	Coefficient	예측 변수가 결과에 미치는 영향력 (β_1, β_2, \dots)	기울기(Slope)라고도 함
절편	Intercept	모든 예측 변수가 0일 때의 기본값 (β_0)	y 절편, 편향(Bias)이라고도 함
모델	Model	입력(X)을 출력(Y)으로 변환하는 수학 공식	$\hat{Y} = \beta_0 + \beta_1 X$
잔차	Residual	실제값(Y)과 모델의 예측값(\hat{Y})의 차이	$r = Y - \hat{Y}$, 즉 ’모델이 틀린 정도’
손실 함수	Loss Function	모델이 얼마나 ’못’ 하는지 측정하는 함수	이 함수의 값을 최소화하는 것이 학습의 목표
평균 제곱 오차	MSE	잔차들을 제곱하여 평균 낸 값	Mean Squared Error. 대표적인 손실 함수
학습/피팅	Training / Fitting	데이터로부터 최적의 계수(β)를 찾는 과정	손실 함수(MSE)를 최소화하는 과정
정규 방정식	Normal Equation	미분을 통해 MSE를 최소화하는 β 를 한 번에 찾는 공식	$\hat{\beta} = (X^T X)^{-1} X^T Y$

3 왜 선형 회귀를 사용할까요?

세상에는 KNN, 신경망 등 복잡하고 강력한 모델이 많습니다. 왜 단순해 보이는 선형 회귀부터 배울까요?

- 모든 모델의 기초:** 복잡한 딥러닝 모델도 결국은 선형 변환과 비선형 함수의 조합입니다. 선형 회귀의 원리(모델 정의 \rightarrow 손실 함수 \rightarrow 최적화)를 완벽히 이해하면, 다른 모든 기계 학습 모델을 이해하는 튼튼한 기반이 됩니다.
- 뛰어난 해석력 (Interpretability):** KNN 같은 모델은 ”왜” 그런 예측이 나왔는지 설명하기 어렵습니다 (Non-parametric). 하지만 선형 회귀는 ”어떤 변수가 결과에 얼마나 영향을 주는지” 명확하게

숫자로 보여줍니다.

□ 예제:

예시: KNN vs 선형 회귀

- **KNN (K-최근접 이웃):** "TV 광고 예산이 1억일 때 예상 매출은?" → "과거 1억과 비슷했던 3개 지점의 평균 매출이 10억이니, 10억일 겁니다." "TV 광고 예산을 2배로 늘리면 매출은?" → "음... 다시 계산해봐야 합니다." (직관적이지 않음)
- **선형 회귀:** "TV 광고 예산이 1억일 때 예상 매출은?" → "학습된 공식 $= 5 + 0.05 \times (TV)$ 에 따라 10억입니다." "TV 광고 예산을 2배로 늘리면 매출은?" → "계수(기울기)가 0.05이므로, 광고비가 1억 증가할 때마다 매출이 5억씩 증가하는 경향이 있습니다." (직관적 해석 가능)

4 단순 선형 회귀 (Simple Linear Regression, SLR)

가장 간단한 형태로, 하나의 예측 변수(X)가 하나의 반응 변수(Y)에 미치는 영향을 모델링합니다.

4.1 모델 정의: "최적의 직선 찾기"

우리는 X 와 Y 사이에 직선 관계가 있다고 가정합니다. 모든 데이터 포인트를 완벽하게 지나는 직선은 없으므로, 약간의 오차(ϵ)를 포함합니다.

$$Y = \beta_0 + \beta_1 X + \epsilon$$

- Y : 반응 변수 (예: 매출)
- X : 예측 변수 (예: TV 광고비)
- β_0 : 절편. X 가 0일 때의 Y 값 (광고비가 0일 때의 기본 매출)
- β_1 : 기울기 (계수). X 가 1단위 증가할 때 Y 의 평균적인 변화량 (광고비 1원 증가 시 매출 변화량)
- ϵ : 오차(Error). 모델이 설명하지 못하는 무작위성 (다른 요인들)

우리의 목표는 데이터를 가장 잘 설명하는 β_0 와 β_1 를 찾는 것입니다. 이 예측된 모델을 \hat{Y} (Y-hat)이라고 부릅니다.

$$\hat{Y} = \hat{\beta}_0 + \hat{\beta}_1 X$$

4.2 최적의 선 찾기: 손실 함수 (Loss Function)

수많은 직선 중에 "최적의 선"은 무엇일까요? 바로 "실제 데이터와 가장 가까운 선"입니다. 이 "가까운 정도"를 측정하는 것이 손실 함수입니다.

1. 잔차 (Residuals)

- 정의: 실제 값(Y_i)과 모델의 예측 값(\hat{Y}_i)의 차이입니다.
- 수식: $r_i = Y_i - \hat{Y}_i$
- 비유: 예측 선에서 실제 데이터 점까지의 "수직 거리"입니다. 이 거리가 짧을수록 좋은 모델입니다.

2. 평균 제곱 오차 (Mean Squared Error, MSE)

- 정의: 모든 데이터의 잔차(r_i)를 제곱하여 더한 뒤, 데이터 개수(n)로 나눈 값입니다.
- 수식: $L(\beta_0, \beta_1) = \text{MSE} = \frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2 = \frac{1}{n} \sum_{i=1}^n (Y_i - (\beta_0 + \beta_1 X_i))^2$

주의사항

Q: 왜 잔차를 그냥 더하지 않고 제곱하나요?

A: 만약 제곱하지 않고 그냥 더하면, 예측보다 위에 있는 점(잔차 > 0)과 아래에 있는 점(잔차 < 0)이 서로 상쇄되어, 실제로는 오차가 큼에도 불구하고 총합이 0에 가까워질 수 있습니다.

제곱을 하는 이유:

- 모든 잔차를 양수로 만듭니다. (상쇄 방지)
- 오차가 큰 값(Outlier)에 더 큰 페널티를 부여합니다. (10의 제곱 = 100, 2의 제곱 = 4)
- 수학적으로 미분하기 쉬워져 최적화에 유리합니다.

4.3 최적화: 손실 최소화하기

기계 학습의 핵심 3단계를 기억하세요.

기계 학습의 핵심 3단계 프로세스

- 모델 정의 (Define Model): $\hat{Y} = \beta_0 + \beta_1 X$ (직선이라고 가정)
- 손실 정의 (Define Loss): $\text{MSE} = \frac{1}{n} \sum (Y_i - \hat{Y}_i)^2$ (틀린 정도를 측정)
- 손실 최소화 (Minimize Loss): MSE가 가장 작아지는 β_0 와 β_1 을 찾는다.

MSE는 β_0 와 β_1 에 대한 2차 함수(3D 그릇 모양)입니다. 이 그릇의 가장 낮은 지점을 찾는 것이 목표입니다.

□ 예제:

비유: 산에서 가장 낮은 계곡 찾기

- 현재 위치: (β_0, β_1) 값
- 고도: MSE 값
- 목표: 고도(MSE)가 가장 낮은 지점 찾기
- 방법: 기울기(경사)가 0이 되는 지점을 찾습니다.

수학적으로 ”기울기가 0”인 지점은 미분(Derivative)을 통해 찾습니다. 손실 함수 L 을 β_0 와 β_1 각각에 대해 편미분(Partial Derivative)하여 0이 되는 지점을 찾습니다.

$$\frac{\partial L}{\partial \beta_0} = 0 \quad \text{and} \quad \frac{\partial L}{\partial \beta_1} = 0$$

이 두 방정식을 연립하여 풀면, MSE를 최소화하는 $\hat{\beta}_0$ 와 $\hat{\beta}_1$ 의 공식을 유도할 수 있습니다. 이를 정규 방정식 (Normal Equation) 또는 최소 제곱법 (Least Squares)이라고 합니다.

단순 선형 회귀의 정규 방정식 (Closed-form Solution)

복잡한 미분 과정(연쇄 법칙 포함)을 거치면 다음과 같은 깔끔한 공식을 얻을 수 있습니다.

$$\hat{\beta}_1 = \frac{\sum_{i=1}^n (X_i - \bar{X})(Y_i - \bar{Y})}{\sum_{i=1}^n (X_i - \bar{X})^2}$$

$$\hat{\beta}_0 = \bar{Y} - \hat{\beta}_1 \bar{X}$$

- \bar{X} : X 의 평균
- \bar{Y} : Y 의 평균

중요: 이 공식은 컴퓨터가 `.fit()` 명령을 실행할 때 내부적으로 계산하는 값입니다. 이처럼 최적의 해를 한 번의 계산으로 찾을 수 있는 경우는 매우 드물며, 선형 회귀의 강력한 특징입니다.

Part II

다중 선형 회귀로의 확장

5 다중 선형 회귀 (Multi-Linear Regression, MLR)

현실에서는 하나의 요인만으로 결과를 예측하기 어렵습니다. (예: 매출은 TV 광고비뿐만 아니라 라디오, 신문 광고비, 소셜 미디어 등에도 영향을 받음)

다중 선형 회귀는 여러 개의 예측 변수(X_1, X_2, \dots, X_p)를 사용하여 Y 를 예측합니다.

5.1 모델 정의: "최적의 초평면 찾기"

SLR이 2D 평면에서 '선'을 찾는 것이라면, MLR은 3D 공간에서 '평면'을, 그 이상의 p 차원 공간에서 '초평면(Hyperplane)'을 찾는 것입니다.

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_p X_p + \epsilon$$

- β_0 : 절편. 모든 예측 변수(X_1, \dots, X_p)가 0일 때의 Y 값.
- β_j : j 번째 예측 변수의 계수. 해석이 중요: "다른 모든 예측 변수가 고정되어 있다고 가정할 때," X_j 가 1단위 증가할 때 Y 의 평균 변화량.

5.2 행렬 표기법 (Matrix Notation)

변수가 많아지면 위 공식을 쓰기 번거롭습니다. 선형 대수(행렬)를 사용하면 매우 깔끔하게 표현할 수 있습니다.

n 개의 데이터와 p 개의 예측 변수가 있다고 가정합시다.

- **Y (반응 변수 벡터):** $n \times 1$ 행렬 (결과값 n 개)

$$Y = \begin{pmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_n \end{pmatrix}$$

- **X (설계 행렬, Design Matrix):** $n \times (p + 1)$ 행렬 (데이터 n 개, 변수 p 개 + 절편용 1)

$$X = \begin{pmatrix} 1 & X_{11} & X_{12} & \dots & X_{1p} \\ 1 & X_{21} & X_{22} & \dots & X_{2p} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & X_{n1} & X_{n2} & \dots & X_{np} \end{pmatrix}$$

- **β (계수 벡터):** $(p + 1) \times 1$ 행렬 (찾아야 할 파라미터 $p + 1$ 개)

$$\beta = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{pmatrix}$$

주의사항

Q: 왜 X 행렬에 '1'로 채워진 첫 번째 열이 있나요?

A: 수학적 트릭입니다. $Y = \beta_0 + \beta_1 X_1 + \dots$ 공식을 행렬 곱으로 표현하면 β_0 (절편)이 따로 떨어져 있어 불편합니다. X 에 1을 추가하고 β 에 β_0 를 포함시키면, 행렬 곱셈 $X\beta$ 의 첫 번째 항이 $(1 \times \beta_0) + (X_1 \times \beta_1) + \dots$ 가되어 절편을 자연스럽게 수식에 포함시킬 수 있습니다.

이제 다중 선형 회귀 모델은 단 세 개의 기호로 표현됩니다.

$$Y = X\beta + \epsilon$$

우리의 예측 모델은 $\hat{Y} = X\hat{\beta}$ 가 됩니다.

5.3 최적화: 다중 회귀의 정규 방정식

SLR과 마찬가지로, MSE를 최소화하는 $\hat{\beta}$ 벡터를 찾아야 합니다. 손실 함수 MSE를 행렬로 표현하면 다음과 같습니다.

$$L(\beta) = \text{MSE} = \frac{1}{n} \|Y - X\beta\|^2 = \frac{1}{n} (Y - X\beta)^T (Y - X\beta)$$

이 손실 함수를 β 벡터에 대해 미분하여 0으로 놓고 풀면 (선형 대수 연산 필요), $\hat{\beta}$ 를 구하는 강력한 공식을 얻습니다.

다중 선형 회귀의 정규 방정식 (The Normal Equation)

MSE를 최소화하는 계수 벡터 $\hat{\beta}$ 는 다음 공식으로 한 번에 계산됩니다.

$$\hat{\beta} = (X^T X)^{-1} X^T Y$$

- X^T : X 의 전치 행렬 (Transpose, 행과 열을 바꿈)
- $(\dots)^{-1}$: 역행렬 (Inverse, 행렬의 나눗셈)

이 공식이 바로 scikit-learn의 `reg.fit(X, y)` 명령이 내부적으로 수행하는 핵심 계산입니다.

Part III

모델 활용 및 해석

6 Python scikit-learn을 이용한 실습

이론적으로 유도된 정규 방정식을 직접 계산할 필요는 없습니다. Python의 scikit-learn 라이브러리가 이 모든 것을 대신해줍니다.

```

1 # 1. 라이브러리임포트
2 from sklearn.linear_model import LinearRegression
3 import pandas as pd
4 import numpy as np
5
6 # 2. 데이터준비예시 (: 광고데이터 )
7 # df = pd.read_csv('Advertising.csv')
8 # X = df[['TV', 'Radio', 'Newspaper']].values # 예측변수행렬 ()
9 # y = df['Sales'].values # 반응변수벡터 ()
10
11 # --- 가상(데이터생성) ---
12 X = np.array([[100, 20], [150, 30], [200, 25], [300, 40]])
13 y = np.array([11, 16, 18, 25])
14 #
15
16 # 3. 모델객체생성인스턴스화 ()
17 reg = LinearRegression()
18
19 # 4. 모델학습피팅 ()
20 # O| .fit() 한줄이  $(X^T X)^{-1} X^T Y$  계산을수행합니다 !
21 reg.fit(X, y)
22
23 # 5. 결과확인
24 print(f"계수 (beta_1, beta_2...): {reg.coef_}")
25 print(f"절편 (beta_0): {reg.intercept_}")
26
27 # 6. 새로운데이터로예측
28 new_data = np.array([[250, 35]]) # TV=250, Radio일=35 때?
29 prediction = reg.predict(new_data)
30 print(f"예측된 매출: {prediction[0]}")

```

Listing 1: scikit-learn을 이용한 선형 회귀 학습

7 모델 파라미터 해석하기

모델을 만드는 것보다 중요한 것은 결과를 해석하는 것입니다.

- 단순 선형 회귀(SLR)의 $\hat{\beta}_1$: "X가 1단위 증가할 때, Y는 평균적으로 $\hat{\beta}_1$ 만큼 변화한다." (예: $\hat{\beta}_1 = 0.05 \rightarrow$ "TV 광고비를 1천원 더 쓰면, 매출은 평균 50유닛 증가한다.")
- 다중 선형 회귀(MLR)의 $\hat{\beta}_j$: "다른 모든 변수(X_k)가 일정하다고 가정할 때," X_j 가 1단위 증가하면, Y는 평균적으로 $\hat{\beta}_j$ 만큼 변화한다." (예: $\hat{\beta}_{tv} = 0.04, \hat{\beta}_{radio} = 0.15 \rightarrow$ "라디오와 신문 광고비를 고정시킨 채, TV 광고비를 1천원 더 쓰면 매출은 평균 40유닛 증가한다.")

변수가 많을 때는 계수 값을 시각화하는 특성 중요도 그래프(Feature Importance Plot)를 사용합니다.

막대가 길수록(양/음 방향 모두) 해당 변수가 예측에 큰 영향을 미친다는 의미입니다.

8 모델 정확도를 위한 고려사항

모델을 그냥 만들고 끝내면 안 됩니다. 계수 값을 신뢰할 수 있는지, 모델이 안정적인지 확인해야 합니다.

8.1 스케일링 (Scaling)

주의사항

문제점: "단위"가 다르면 계수 비교가 불가능합니다.

'TV 광고비' (단위: 억 원)와 '라디오 광고비' (단위: 만 원)를 예측 변수로 사용했다고 가정해봅시다. TV 광고비가 1단위(1억) 변하는 것과 라디오 광고비가 1단위(1만원) 변하는 것은 크기 자체가 다릅니다.

이때 $\hat{\beta}_{tv} = 10$, $\hat{\beta}_{radio} = 0.1$ 이 나왔다고 해서 "TV 광고가 라디오보다 100배 중요하다"고 말할 수 없습니다. 변수의 스케일(단위)이 다르기 때문에 β 계수의 절대 크기를 직접 비교하는 것은 무의미 합니다.

해결책: 스케일링 모든 예측 변수 X 들을 학습 전에 비슷한 범위(스케일)로 변환합니다.

1. 표준화 (Standardization): 데이터를 평균 0, 표준편차 1인 분포로 변환합니다. (Z-score)

$$X_{\text{scaled}} = \frac{X - \text{mean}(X)}{\text{std}(X)}$$

2. 정규화 (Normalization): 데이터를 0과 1 사이의 범위로 변환합니다. (Min-Max Scaling)

$$X_{\text{scaled}} = \frac{X - \min(X)}{\max(X) - \min(X)}$$

스케일링을 수행한 후 모델을 학습시키면, β 계수들은 단위의 영향에서 벗어나 변수의 순수한 중요도를 (근사적으로) 비교할 수 있게 됩니다.

8.2 다중공선성 (Collinearity)

주의사항

문제점: 예측 변수끼리 너무 친한 경우

다중공선성이란 예측 변수들끼리 높은 상관관계를 갖는 상황을 말합니다. (예: $X_1 =$ '신용 한도', $X_2 =$ '신용 등급'. 두 변수는 거의 같은 정보를 담고 있음)

비유: 공로를 구분하기 힘든 두 가수 "두 명의 가수(예측 변수)가 정확히 똑같은 멜로디(정보)를 부르며 노래(반응 변수)의 인기에 기여하고 있습니다. 이때 노래 인기의 공로가 누구에게 몇 결과:

1. 모델의 전체적인 예측 성능(MSE)은 괜찮을 수 있습니다.
2. 하지만 개별 β 계수의 신뢰도가 박살납니다.
3. β 값이 비상식적으로 커지거나, 부호가 반대로 나올 수 있습니다.

4. 데이터를 조금만 바꿔도 β 값이 크게 널뛰기합니다. (불안정)
(예: '신용 한도'를 제거했더니 '신용 등급'의 β 값이 1.1에서 3.9로 갑자기 뛰어오름)

해결책:

- 시각화: 변수 간의 산점도 행렬(Scatter Matrix)을 그려 높은 상관관계를 확인합니다.
- 제거: 상관관계가 매우 높은 변수 중 하나를 제거합니다.

8.3 범주형 예측 변수 (Qualitative Predictors)

'성별' (Male/Female), '학생 여부' (Yes/No), '인종' (Asian/Caucasian/...)처럼 숫자가 아닌 텍스트 데이터는 어떻게 처리할까요?

해결책 1: 더미 변수 (Dummy Variables) (2개의 레벨을 가질 때)

컴퓨터가 이해하도록 0과 1로 바꿔줍니다. (예: '성별' 변수 \rightarrow 'is_female'이라는 새 변수 생성)

$$x_{\text{is_female}} = \begin{cases} 1 & \text{if person is female} \\ 0 & \text{if person is male} \end{cases}$$

이 변수를 모델에 포함시키면 ($Y = \beta_0 + \beta_1 x_{\text{is_female}}$) 해석이 매우 흥미로워집니다.

- Male** ($x = 0$): $Y = \beta_0 + \beta_1(0) = \beta_0 \rightarrow \beta_0$ (절편)는 남성의 평균 Y 값(기준선)이 됩니다.
- Female** ($x = 1$): $Y = \beta_0 + \beta_1(1) = \beta_0 + \beta_1 \rightarrow \beta_1$ 은 여성과 남성의 평균 Y 값 차이가 됩니다.

해결책 2: 원-핫 인코딩 (One-Hot Encoding) (3개 이상의 레벨을 가질 때)

(예: '인종' 변수 \rightarrow 'Asian', 'Caucasian', 'African American')

k 개의 레벨이 있다면, $k - 1$ 개의 더미 변수를 만듭니다. (하나를 기준선으로 삼음)

$$x_{\text{is_Asian}} = \begin{cases} 1 & \text{if Asian} \\ 0 & \text{else} \end{cases} \quad x_{\text{is_Caucasian}} = \begin{cases} 1 & \text{if Caucasian} \\ 0 & \text{else} \end{cases}$$

모델: $Y = \beta_0 + \beta_1 x_{\text{is_Asian}} + \beta_2 x_{\text{is_Caucasian}}$

- African American** (기준선, $x_1 = 0, x_2 = 0$): $Y = \beta_0$
- Asian** ($x_1 = 1, x_2 = 0$): $Y = \beta_0 + \beta_1$
- Caucasian** ($x_1 = 0, x_2 = 1$): $Y = \beta_0 + \beta_2$

$\rightarrow \beta_0$ 는 기준선(African American)의 평균 Y 가 되고, β_1 과 β_2 는 각각 기준선과의 차이를 나타냅니다.

Part IV

학습 점검

9 핵심 학습 체크리스트

이 문서를 다 읽은 후, 다음 질문에 답할 수 있는지 확인하세요.

선형 회귀가 KNN과 같은 다른 모델에 비해 갖는 장점(해석력)은 무엇인가?

'모델 학습(Training)'의 3단계 프로세스(모델 정의, 손실 정의, 손실 최소화)를 설명할 수 있는가?

손실 함수로 MSE를 사용할 때, 왜 잔차를 그냥 더하지 않고 '제곱'하는가?

단순 선형 회귀(SLR)의 β_1 계수의 의미를 정확히 설명할 수 있는가?

다중 선형 회귀(MLR)의 β_j 계수의 의미를 "다른 변수를 고정할 때"라는 조건과 함께 설명할 수 있는가?

`scikit-learn`의 `.fit()` 메소드가 내부적으로 어떤 수학적 계산(정규 방정식)을 수행하는지 아는가?

왜 변수 스케일링(Scaling)이 필요한가? (단위가 다른 변수 간 계수 비교 문제)

다중공선성(Collinearity)이 무엇이며, 왜 모델 '해석'에 문제를 일으키는지 설명할 수 있는가?

'성별'과 같은 범주형 데이터를 모델에 포함시키기 위한 '더미 변수' 기법을 설명할 수 있는가?

10 초심자 FAQ

주의사항

Q: 왜 손실 함수로 잔차의 '절대값'이 아닌 '제곱'(MSE)을 주로 쓰나요? **A:** 절대값(MAE, Mean Absolute Error)도 좋은 손실 함수입니다. 하지만 MSE를 더 선호하는 두 가지 이유가 있습니다. 1) MSE는 수학적으로 미분이 부드럽게 가능하여 최적화(가장 낮은 지점 찾기)에 유리합니다. 2) MSE는 오차가 큰 값(Outlier)에 제곱으로 페널티를 주므로, 모델이 큰 실수를 하지 않도록 유도하는 경향이 있습니다.

Q: `reg.fit(X, y)` 명령은 마법 상자인가요? 정확히 뭘 하는 거죠? **A:** 마법이 아닙니다! `.fit()`은 이 문서에서 배운 정규 방정식 $\hat{\beta} = (X^T X)^{-1} X^T Y$ 공식을 데이터 X 와 y 에 대해 정확히 계산하여, MSE를 최소화하는 $\hat{\beta}$ 벡터(즉, `reg.coef_` 와 `reg.intercept_`)를 찾아내는 과정입니다.

Q: 스케일링을 하면 모델의 예측 성능(MSE)이 좋아지나요? **A:** 단순 선형 회귀나 다중 선형 회귀에서는 스케일링이 예측 성능 자체에 영향을 주지 않습니다. (어차피 정규 방정식으로 최적의 해를 찾기 때문입니다.) 하지만 계수를 해석하고 비교하기 위해 스케일링이 필요합니다. (참고: 경사 하강법(Gradient Descent)을 사용하는 모델이나, 정규화(Ridge/Lasso)가 포함된 모델에서는 스케일링이 성능과 수렴 속도에 큰 영향을 줍니다.)

Q: 다중공선성이 높으면 모델이 "틀린" 건가요? **A:** "틀렸다"기보다는 "불안정하다"고 표현하는 것이 맞습니다. 모델의 예측 성능 자체는 여전히 높을 수 있습니다. (어차피 변수들이 비슷한 정보를 담고 있으므로) 하지만 "각 변수가 얼마나 중요한가"를 나타내는 β 계수 값을 신뢰할 수 없게 됩니다. 따라서 '예측'만이 목표라면 큰 문제가 아닐 수 있지만, '해석'이 목표라면 반드시 해결해야 합니다.

Q: 왜 k 개의 범주(예: 3개 인종)에 k 개가 아닌 $k - 1$ 개(2개)의 더미 변수를 쓰나요? **A:** k 개를 모두 사용하면 완벽한 다중공선성(Dummy Variable Trap)이 발생합니다. 예를 들어 x_{Asian} , $x_{\text{Caucasian}}$, $x_{\text{AfricanAmerican}}$ 3개를 모두 만들면, $x_{\text{Asian}} + x_{\text{Caucasian}} + x_{\text{AfricanAmerican}} = 1$ 이라는 완벽한 선형 관계가 생깁니다. 이는 X 행렬의 역행렬($X^T X$) $^{-1}$ 을 계산할 수 없게 만듭니다. 따라서 하나를 기준선(Baseline)으로 제외하여 이 문제를 피합니다.

11 빠르게 훑어보기 (1-Page Summary)

기계 학습 3단계 프로세스

모든지도 학습은 이 3단계를 따릅니다.

1. 모델 정의: 데이터의 관계를 어떤 함수(예: 직선)로 가정할지 선택합니다.
2. 손실 함수 정의: 모델의 예측이 실제와 얼마나 다른지(오차) 측정하는 기준(예: MSE)을 정합니다.
3. 손실 최소화: 손실이 최소가 되는 모델의 파라미터(예: β)를 수학적 방법(예: 정규 방정식, 경사 하강법)으로 찾습니다.

단순 선형 회귀 (SLR): $Y = \beta_0 + \beta_1 X$

- 목표: 2D 평면에서 데이터를 가장 잘 표현하는 직선을 찾는다.
- 해석: β_1 은 X 가 1단위 증가할 때 Y 의 평균 변화량이다.
- 해법: $\hat{\beta}_1 = \frac{\sum(X_i - \bar{X})(Y_i - \bar{Y})}{\sum(X_i - \bar{X})^2}$, $\hat{\beta}_0 = \bar{Y} - \hat{\beta}_1 \bar{X}$

다중 선형 회귀 (MLR): $Y = X\beta$

- 목표: $p + 1$ 차원 공간에서 데이터를 가장 잘 표현하는 초평면(Hyperplane)을 찾는다.
- 해석: β_j 는 다른 모든 변수가 고정되었을 때 X_j 가 1단위 증가할 때 Y 의 평균 변화량이다.
- 해법 (정규 방정식): $\hat{\beta} = (X^T X)^{-1} X^T Y$ (이것이 .fit()의 핵심!)

모델 해석의 3대 함정

1. 스케일링 문제 (Apple vs Orange): 단위(스케일)가 다른 변수들의 β 계수 크기는 직접 비교할 수 없다. → 해결: 표준화(Standardization) 후 비교
2. 다중공선성 문제 (Clones): 서로 상관관계가 높은 변수들은 β 계수 값은 불안정하게 만든다. → 해결: 상관관계 높은 변수 중 하나를 제거
3. 범주형 변수 문제 (Text): 'Male'/'Female' 같은 텍스트는 0/1로 변환(더미 변수)해야 한다. → 해결: k 개 레벨에 $k - 1$ 개 더미 변수 사용

December 10, 2025

- 강의명: CS109A: 데이터 과학 입문
- 주차: Lecture 06
- 교수명: Pavlos Protopapas, Kevin Rader, Chris Gumb
- 목적: Lecture 06의 핵심 개념 학습

Contents

1	개요	2
2	핵심 용어 정리	3
3	모델 평가와 과적합의 문제	4
3.1	학습 오차(Training Error)의 한계	4
3.2	일반화 오차(Generalization Error)의 중요성	4
3.3	과적합(Overfitting)이란 무엇인가?	4
3.4	모델 해석(Interpretation)의 함정	4
4	선형 회귀의 확장: 비선형성 다루기	6
4.1	선형 회귀의 4가지 핵심 가정	6
4.2	진단 도구: 잔차 분석(Residual Analysis)	6
4.3	확장 1: 상호작용 항 (Interaction Effect)	6
4.4	확장 2: 다항 회귀 (Polynomial Regression)	7
4.5	다항 회귀의 함정: 스케일링과 항의 개수	7
5	최적의 모델 찾기: 모델 선택 (Model Selection)	8
5.1	데이터 3-분할: 학습, 검증, 테스트	8
5.2	모델 선택 방법론	8
5.3	하이퍼파라미터 튜닝과 검증 세트	8
6	신뢰할 수 있는 평가: 교차 검증 (Cross-Validation)	10
6.1	단일 검증 세트의 문제점 (CV의 동기)	10
6.2	K-겹 교차 검증 (K-Fold Cross-Validation) 절차	10
6.3	LOOCV (Leave-One-Out Cross-Validation)	10
6.4	구현: sklearn과 neg_mean_squared_error	11

7 빠르게 훑어보기 (1페이지 요약)	12
8 초심자 FAQ	13

1 개요

이 문서는 머신러닝 모델을 만들고 평가하는 핵심 과정인 '모델 선택'에 대해 다릅니다.

모델의 성능은 학습 데이터가 아닌, **본 적 없는 새로운 데이터**로 평가해야 합니다. 이 과정에서 모델이 학습 데이터의 노이즈까지 암기하는 **'과적합'**이 가장 큰 문제입니다. 우리는 선형 회귀를 확장한 **'상호작용 항'**과 **'다항 회귀'**를 통해 더 복잡한 모델을 만들 수 있지만, 이는 과적합의 위험을 높입니다. '모델 선택'은 이 복잡성과 일반화 성능 사이의 균형점을 찾는 과정입니다. **'교차 검증(Cross-Validation)'**은 단일 검증 세트의 함정을 피하고 모델의 일반화 성능을 신뢰성 있게 추정하는 표준적인 방법입니다.

2 핵심 용어 정리

모델 선택과 평가 과정을 이해하기 위해 필수적인 용어들을 정리했습니다.

Table 1: 모델 선택 및 평가 핵심 용어

용어	쉬운 설명	원어	비고 (예시)
과적합	모델이 학습 데이터를 '암기'해버려서, 새로운 데이터에 대한 예측 성능이 떨어지는 현상.	Overfitting	시험 족보만 외우고 응용 문제를 못 푸는 학생.
일반화 오차	모델이 '처음 보는' 데이터에서 발생하는 오차. 이 오차를 최소화하는 것이 최종 목표.	Generalization Error	실전 모의고사 성적.
모델 선택	여러 모델 후보(예: 다항식 차수) 중에서 일반화 오차가 가장 낮을 것으로 기대되는 모델을 고르는 과정.	Model Selection	1차, 2차, 3차 함수 중 2차 함수를 선택.
하이퍼파라미터	모델이 학습하기 전에 '사람'이 미리 정해야 하는 값.	Hyperparameter	다항 회귀의 '차수(M)', KNN의 'K값'.
검증 세트	하이퍼파라미터 튜닝(모델 선택)을 위해 사용하는 데이터.	Validation Set	여러 모델을 테스트해보는 연습 문제지.
테스트 세트	모델 선택이 끝난 후, '단 한 번' 최종 성능을 보고하기 위해 사용하는 데이터. 절대 모델 선택에 사용하면 안 됨.	Test Set	최종 학기말 고사.
상호작용 항	한 예측 변수의 효과가 다른 예측 변수의 수준에 따라 달라지는 효과(시너지 효과).	Interaction Term	TV 광고 효과(X_1)가 라디오 광고(X_2)와 함께 할 때 더 커지는 현상 ($X_1 X_2$).
다항 회귀	X, X^2, X^3 등 예측 변수의 거듭제곱을 새로운 예측 변수처럼 사용해 비선형 관계를 학습하는 기법.	Polynomial Regression	$Y = \beta_0 + \beta_1 X + \beta_2 X^2$
잔차	모델의 '예측값'과 '실제값'의 차이. 모델이 잘 맞는지 진단하는 핵심 도구.	Residual	$e_i = y_i - \hat{y}_i$
동분산성	예측 변수 X 의 값에 관계없이 잔차의 분산이 일정한 것. (선형 회귀의 중요 가정)	Homoscedasticity	잔차 그래프가 깔때기 모양이 아님.
이분산성	X 가 커질수록 잔차의 변동 폭도 커지거나 작아지는 현상. (동분산성 가정이 깨진 상태)	Heteroscedasticity	잔차 그래프가 깔때기(fanning) 모양.
K-겹 교차 검증	학습 데이터를 K개의 '조각'으로 나눈 뒤, K-1 개로 학습하고 1개로 검증하는 과정을 K번 반복.	K-Fold Cross-Validation	데이터를 5조각(Fold)으로 나눔.

3 모델 평가와 과적합의 문제

3.1 학습 오차(Training Error)의 한계

모델을 평가할 때 MSE(평균 제곱 오차)나 R^2 (결정 계수) 같은 지표를 사용합니다. 하지만 모델을 학습시킨 학습 데이터(Training Data)로 계산된 오차(학습 오차)는 모델의 실제 성능을 보장하지 않습니다.

학습 오차는 믿을 수 없다

네 개의 서로 다른 데이터셋이 동일한 $MSE=1$ 을 가질 수 있습니다. 하지만 그래프를 보면 어떤 모델은 비선형 관계를 놓치고, 어떤 모델은 직선을 잘못 학습하고, 어떤 모델은 특이점에 과도하게 영향을 받습니다.

단순히 MSE가 낮다고 해서 좋은 모델이라고 말할 수 없습니다.

3.2 일반화 오차(Generalization Error)의 중요성

우리의 진짜 목표는 모델이 본 적 없는 새로운 데이터(Unseen Data)에서도 잘 작동하도록 하는 것입니다. 이때 '새로운 데이터'에서 발생하는 오차를 일반화 오차(Generalization Error)라고 부릅니다. 모델 선택의 목표는 이 일반화 오차를 최소화하는 모델을 찾는 것입니다.

3.3 과적합(Overfitting)이란 무엇인가?

과적합(Overfitting)은 모델이 학습 데이터의 패턴(경향성)뿐만 아니라, 데이터에 포함된 사소한 노이즈(Noise)나 특이 점(Outlier)까지 모두 '암기'해버리는 현상을 말합니다.

□ 예제: title

과적합된 모델은 마치 시험 범위의 모든 예제와 답을 통째로 암기한 학생과 같습니다.

- 학습 데이터 (시험 족보): 100점을 받습니다. (낮은 학습 오차)
- 새로운 데이터 (응용 문제): 암기한 내용과 조금이라도 다르면 전혀 풀지 못합니다. (높은 일반화 오차)

모델은 데이터의 '개념(Trend)'을 배워야지, 데이터 자체를 '암기(Noise)' 하면 안 됩니다.

과적합은 모델이 필요 이상으로 복잡할 때 발생합니다.

- 예측 변수(Feature)가 너무 많을 때
- 다항 회귀의 차수(Degree)가 너무 높을 때
- 상호작용 항이 너무 많을 때

3.4 모델 해석(Interpretation)의 함정

모델의 성능 지표(MSE)가 좋아 보여도, 반드시 모델의 계수(Coefficient)를 해석하여 상식에 맞는지 확인해야 합니다.

예제: title

TV 광고 예산(X)과 매출(Y)의 관계를 모델링한 두 가지 경우입니다.

- **사례 1:** $Y = -0.05X + 6.2$
- **문제:** 기울기가 음수(-0.05)입니다. 이는 TV 광고 예산을 늘릴수록 매출이 줄어든다는 뜻입니다.
(상식에 맞지 않습니다. 데이터에 오류가 있거나, 모델이 잘못되었을 수 있습니다.)
- **사례 2:** $Y = 0.02X - 0.5$
- **문제:** 절편이 음수(-0.5)입니다. 이는 광고 예산이 0일 때($X=0$) 매출이 음수가 된다는 뜻입니다.
(마찬가지로 상식에 맞지 않습니다.)

단순히 숫자에만 의존하지 말고, 모델이 현실을 잘 설명하는지 항상 비판적으로 검토해야 합니다.

4 선형 회귀의 확장: 비선형성 다루기

단순 선형 회귀는 강력하지만 현실의 복잡한 데이터를 설명하기엔 한계가 있습니다. 더 복잡한 모델을 만들기 전에, 선형 회귀의 기본 가정부터 확인해야 합니다.

4.1 선형 회귀의 4가지 핵심 가정

우리가 사용하는 MSE(평균 제곱 오차) 손실 함수는 다음 4가지 가정을 암묵적으로 전제합니다.

1. 선형성(Linearity): 예측 변수와 반응 변수 간에 직선적인 관계가 있다.
2. 독립성(Independence): 각 데이터의 오차(잔차)는 서로 독립적이다. (MSE는 단순히 오차 제곱을 더하기 때문에 이 가정이 필요합니다.)
3. 등분산성(Homoscedasticity): 모든 데이터 포인트에서 오차의 분산이 동일하다. (MSE는 모든 오차에 '가중치'를 두지 않기 때문에 이 가정이 필요합니다.)
4. 잔차의 정규성(Normality of Residuals): 잔차가 정규분포를 따른다. (오차를 '제곱'하는 방식은 정규분포 가정과 통계적으로 연결됩니다.)

이 외에도 '예측 변수 X는 오차가 없다(Fixed X)', '예측 변수 간 상관관계가 높지 않다(No Multicollinearity)' 등의 가정이 있습니다.

4.2 진단 도구: 잔차 분석(Residual Analysis)

위의 가정이 맞는지 확인하는 가장 좋은 방법은 잔차 분석입니다. 잔차($e = Y - \hat{Y}$)를 X축에, 예측 변수(X)나 예측 값(\hat{Y})을 Y축에 그려봅니다.

- 좋은 모델 (가정 만족): 잔차가 0을 기준으로 특별한 패턴 없이 무작위로 흩어져 있습니다. (White Noise처럼 보입니다.) 잔차의 히스토그램은 종 모양(정규분포)을 띕니다.
- 나쁜 모델 (선형성 위반): 잔차가 U자형, S자형 등 뚜렷한 패턴을 보입니다. 이는 모델이 데이터의 비선형적 경향을 놓치고 있다는 신호입니다.
- 나쁜 모델 (등분산성 위반): X 가 커질수록 잔차의 변동 폭이 커지거나(깔때기 모양, Fanning) 작아집니다. 이는 이분산성(Heteroscedasticity)을 의미합니다.

잔차 분석의 핵심

"잔차 플롯에서 어떤 패턴이든 보인다면, 심지어 용이나 성(Dragons flying over castles)이 상상되더라도, 모델의 기본 가정이 위반되었을 가능성이 높습니다."

4.3 확장 1: 상호작용 항(Interaction Effect)

현실에서는 한 변수의 효과가 다른 변수에 따라 달라지는 시너지 효과(Synergy Effect)가 흔합니다. 이를 모델링하는 것이 상호작용 항입니다.

□ 예제: title

모델 A: 상호작용 항 없음 $balance = \beta_0 + \beta_1 \times income + \beta_2 \times student$

- 비학생(student=0): $balance = \beta_0 + \beta_1 \times income$

- 학생(student=1): $balance = (\beta_0 + \beta_2) + \beta_1 \times income$

해석: 학생과 비학생은 기본 잔액(절편)만 다를 뿐, 소득(income)이 1 단위 증가할 때 잔액이 β_1 만큼 증가하는 기울기는 동일합니다. (두 개의 평행선)

모델 B: 상호작용 항 추가 $balance = \beta_0 + \beta_1 \times income + \beta_2 \times student + \beta_3 \times (income \times student)$

- 비학생(student=0): $balance = \beta_0 + \beta_1 \times income$

- 학생(student=1): $balance = (\beta_0 + \beta_2) + (\beta_1 + \beta_3) \times income$

해석: 학생과 비학생은 절편도 다르고(β_0 vs $\beta_0 + \beta_2$), 소득이 증가할 때의 기울기도 다릅니다(β_1 vs $\beta_1 + \beta_3$). 만약 $\beta_3 > 0$ 이라면, 학생은 소득이 증가할 때 비학생보다 대출 잔액이 더 가파르게 증가(더 많은 소비)한다는 의미입니다.

4.4 확장 2: 다항 회귀 (Polynomial Regression)

데이터가 명백한 곡선 형태일 때, 다항 회귀를 사용합니다. $Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \dots + \beta_M X^M$

다항 회귀의 ”속임수”: 왜 이것도 선형 회귀인가?

다항 회귀는 X 와 Y 의 관계는 비선형이지만, 통계적으로는 다중 선형 회귀의 특수한 경우입니다.

속임수(Trick): X^2 를 \tilde{X}_2 , X^3 를 \tilde{X}_3 라는 완전히 새로운 예측 변수로 취급합니다. $Y = \beta_0 + \beta_1 X_1 + \beta_2 \tilde{X}_2 + \dots + \beta_M \tilde{X}_M$

이렇게 변환하고 나면, 이 모델은 각 계수($\beta_0, \beta_1, \beta_2, \dots$)에 대해 선형입니다. 따라서 다중 선형 회귀를 푸는 것과 동일한 방식($\hat{\beta} = (\tilde{X}^T \tilde{X})^{-1} \tilde{X}^T y$)으로 β 값을 찾을 수 있습니다.

sklearn에서는 PolynomialFeatures로 $X, X^2, X^3 \dots$ 항들을 만든 후, LinearRegression을 fit시키면 됩니다.

4.5 다항 회귀의 함정: 스케일링과 항의 개수

다항 회귀 사용 시 주의사항

1. 특성 스케일링(Feature Scaling) 필수: X 의 범위가 100만 되어도 X^{10} 은 천문학적인 숫자가 됩니다. 이렇게 값의 범위 차이가 극심하면 컴퓨터가 β 값을 계산할 때 수치적으로 불안정해집니다. 다항 회귀 사용 전, StandardScaler 등을 사용해 X 의 범위를 (평균=0, 표준편차=1)로 표준화하는 것이 좋습니다.
2. PolynomialFeatures의 작동 방식: sklearn의 PolynomialFeatures는 기본적으로 '1' (절편 항), '상호작용 항' ($X_1 X_2$) 등을 모두 포함하여 생성합니다.
 - 이 도구가 '1'을 이미 만들었으므로, LinearRegression을 학습시킬 때 fit_intercept=False 옵션을 주어야 절편이 중복 계산되지 않습니다.
 - 예측 변수가 여러 개일 때 불필요한 상호작용 항이 너무 많이 생겨 과적합을 유발할 수 있습니다.

다항 회귀의 차수 M 을 몇으로 할지 정하는 것 자체가 하이퍼파라미터 튜닝이며, 모델 선택의 핵심 문제입니다.

- M 이 너무 낮으면 (예: 1차): 데이터의 곡선 트렌드를 못 잡아냄 (과소적합, Underfitting)
- M 이 너무 높으면 (예: 50차): 데이터의 모든 노이즈를 통과하는 구불구불한 선이 됨 (과적합, Overfitting)

5 최적의 모델 찾기: 모델 선택 (Model Selection)

모델 선택은 과소적합과 과적합 사이의 '최적점(Sweet Spot)'을 찾는 과정입니다.

5.1 데이터 3-분할: 학습, 검증, 테스트

이를 위해 데이터를 3가지 용도로 나눕니다.

데이터의 3가지 역할

- **학습 세트 (Training Set):** 모델을 학습(훈련)시키는데 사용. (모델의 β 계수들을 찾는데 사용)
- **검증 세트 (Validation Set):** 학습된 모델들 중 최적의 하이퍼파라미터(예: 다항식 차수 M)를 선택하는데 사용.
- **테스트 세트 (Test Set):** 모델 선택까지 모두 끝난 후, 우리가 선택한 최종 모델의 일반화 성능을 보고하기 위해 '단 한 번' 사용.

테스트 세트의 신성불가침 원칙

"테스트 세트를 사용해 모델을 선택하거나 튜닝하는 행위는 학계의 가장 큰 금기 중 하나입니다." (마치 모의고사 문제로 기말고사를 내는 것과 같습니다.) 테스트 세트는 모델 개발 과정에서 완전히 격리되어야 하며, 최종 성능 보고 시에만 사용해야 합니다.

5.2 모델 선택 방법론

예측 변수가 J 개 있을 때, 어떤 변수를 모델에 포함시킬지 고르는 방법입니다.

- **전체 탐색 (Exhaustive Search):** J 개의 변수로 만들 수 있는 모든 조합(2^J 개)의 모델을 다 만들어보고, 검증 세트 성능이 가장 좋은 것을 고릅니다. 변수가 10개만 돼도 1024개, 20개면 100만 개가 넘어 현실적으로 불가능합니다.
- **탐욕적 알고리즘 (Greedy Algorithms):** 매 순간 '지금 당장' 가장 좋아 보이는 선택을 하는 방식입니다.
 - 전진 선택법 (Forward Selection): 1. 아무 변수도 없는 모델(M_0)에서 시작. 2. J 개의 변수 중 1개를 추가했을 때 검증 오차가 가장 많이 줄어드는 변수를 추가(M_1). 3. M_1 에 $J - 1$ 개의 남은 변수 중 1개를 추가했을 때 검증 오차가 가장 많이 줄어드는 변수를 추가(M_2). 4. ... 변수를 J 개 다 쓸 때까지 반복. 5. 만들어진 M_0, M_1, \dots, M_J 중에서 검증 오차가 가장 낮았던 모델을 최종 선택.
 - (이 외에 후진 제거법, 단계적 선택법 등이 있습니다.)
 - 장점: 2^J 에 비해 $O(J^2)$ 수준으로 계산이 훨씬 빠릅니다.
 - 단점: 최적의 조합을 놓칠 수 있습니다. (예: X_1 만 있을 때보다 X_2 만 있을 때가 더 나빠도, $X_1 + X_2$ 조합이 $X_1 + X_3$ 조합보다 훨씬 좋을 수 있습니다.)

5.3 하이퍼파라미터 튜닝과 검증 세트

다항 회귀의 차수 M 을 찾는 것과 같은 하이퍼파라미터 튜닝이 모델 선택의 핵심입니다. $M = 1, 2, 3, \dots, 10$ 까지의 모델을 모두 학습 세트로 학습시킨 뒤, 각 모델의 성능을 검증 세트로 평가합니다.

모델 복잡도(Degree)에 따른 오차 그래프

- 학습 오차 (Training MSE): 모델이 복잡해 질수록(차수가 높아질수록) 학습 데이터를 더 잘 '암기' 할 수 있으므로 계속 감소합니다.
- 검증 오차 (Validation MSE):
 - 과소적합 영역 (Underfitting): 차수가 낮으면 트렌드를 못 잡아 오차가 높습니다.
 - 최적점 (Best Model): 트렌드는 잘 잡고 노이즈는 무시하는 지점에서 오차가 가장 낮아집니다.
 - 과적합 영역 (Overfitting): 차수가 너무 높으면 노이즈까지 암기하기 시작해, '새로운' 검증 데이터에서는 오차가 다시 증가합니다.

우리는 이 검증 오차 그래프(U자형 커브)에서 오차가 최소가 되는 지점(Minimum)의 차수 M 을 최 적의 하이퍼파라미터로 선택합니다.

6 신뢰할 수 있는 평가: 교차 검증 (Cross-Validation)

6.1 단일 검증 세트의 문제점 (CV의 동기)

만약 데이터를 학습/검증 세트로 딱 한 번만 나눈다면, 검증 세트가 우연히 특정 모델에 유리하게 뽑힐 수 있습니다.

□ 예제: title

데이터의 실제 트렌드는 3차 함수(노란색 선)에 가깝다고 가정해봅시다. 하지만 우리가 우연히 뽑은 검증 데이터(분홍색 점)가 1차 함수(녹색 선) 근처에 몰려있을 수 있습니다.

이 경우, 우리는 1차, 2차, 3차 모델을 검증 세트로 테스트한 후, ”검증 오차가 가장 낮은 1차 모델이 최고다!”라고 잘못된 선택을 하게 됩니다.

이는 우리가 학습 데이터에 과적합되는 것을 피하려다, 검증 데이터에 과적합되는 결과를 낳습니다.

6.2 K-겹 교차 검증 (K-Fold Cross-Validation) 절차

이 문제를 해결하기 위해, 데이터를 ’여러 번’ 다르게 쪼개서 검증하고 그 성능을 ’평균’내는 것이 교차 검증입니다. (K-Fold CV가 표준입니다.)

전제: 테스트 세트는 미리 분리해두고 절대 사용하지 않습니다. 남은 학습+검증 데이터를 가지고 다음을 수행합니다.

1. 전체 학습 데이터를 K개의 균등한 ’조각(Fold)’으로 나눕니다. (보통 K=5 또는 K=10 사용)

2. K 번의 반복(Iteration)을 수행합니다.

3. 1번째 반복:
 - 1번 조각(Fold 1)을 검증 세트로 사용합니다.
 - 나머지 K-1개 조각(Fold 2, 3, 4, 5)을 학습 세트로 사용해 모델을 학습합니다.
 - 학습된 모델로 Fold 1을 예측하여 검증 오차(MSE_1)를 계산합니다.

4. 2번째 반복:
 - 2번 조각(Fold 2)을 검증 세트로 사용합니다.
 - 나머지 K-1개 조각(Fold 1, 3, 4, 5)을 학습 세트로 사용해 모델을 학습합니다.
 - 학습된 모델로 Fold 2를 예측하여 검증 오차(MSE_2)를 계산합니다.

5. ... K번째 반복까지 동일하게 수행합니다.

6. 최종 CV 점수: K개의 검증 오차(MSE_1, \dots, MSE_K)를 평균냅니다. $CV(Model) = \frac{1}{K} \sum_{i=1}^K MSE_i^{val}$
모델 선택(예: 다항식 차수 M 찾기)을 할 때, $M = 1, M = 2, M = 3$ 등 각 후보에 대해 이 K-Fold CV 과정을 모두 수행하고, CV 점수가 가장 낮은 M 을 최종 모델로 선택합니다.

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

K-Fold CV의 극단적인 형태로, $K = N$ (데이터 포인트 개수)인 경우입니다.

- 총 N번의 반복을 수행합니다.
- 매 반복마다 데이터 1개만 검증 세트로 쓰고, 나머지 N-1개로 학습합니다.
- 장점: 편향(Bias)이 매우 낮습니다.

- 단점: N 번이나 모델을 학습시켜야 하므로 계산 비용이 매우 비쌉니다.

6.4 구현: sklearn과 neg_mean_squared_error

`sklearn.model_selection.cross_validate` 함수를 사용해 교차 검증을 쉽게 수행할 수 있습니다.

Scoring: 왜 'Negative' MSE를 쓰는가?

`sklearn`의 교차 검증 및 튜닝 도구는 점수(Score)를 최대화(Maximize)하도록 설계되어 있습니다.
(예: 정확도(Accuracy)는 높을수록 좋음)

하지만 MSE는 최소화(Minimize)해야 하는 '오차(Error)' 지표입니다. 따라서 MSE를 최소화하는 것은 -MSE를 최대화하는 것과 같습니다.

`cross_validate`의 `scoring` 매개변수에 'mse'가 아닌 '`neg_mean_squared_error`'를 전달해야 올바르게 작동합니다.

```

1 from sklearn.model_selection import cross_validate
2 from sklearn.linear_model import LinearRegression
3 from sklearn.preprocessing import PolynomialFeatures
4 from sklearn.pipeline import make_pipeline
5
6 # 예: 차3 다항회귀모델파이프라인생성
7 model = make_pipeline(
8     PolynomialFeatures(degree=3, include_bias=False),
9     LinearRegression(fit_intercept=True)
10    # 가PolynomialFeatures 을1 만들지않게 (include_bias=False)하고
11    # 이LinearRegression 절편을찾게 (fit_intercept=True) 할수있음
12    # 또는( 반대로설정 )
13 )
14
15 # 데이터 X, 와y 겹5-(cv=5) 교차검증수행
16 # 점수지표로 'neg_mean_squared_error' 사용
17 cv_results = cross_validate(
18     model,
19     X,
20     y,
21     cv=5,
22     scoring="neg_mean_squared_error",
23     return_train_score=True # 학습스코어도반환
24 )
25
26 # cv_results['test_score'] 에개의 5 음수 () MSE 값이들어있음
27 # 이값들의평균을내고부호를바꾸면최종 CV 점수(MSE)가 됨
28 final_cv_mse = -cv_results['test_score'].mean()

```

Listing 1: `sklearn`을 사용한 교차 검증 (의사 코드)

7 빠르게 훑어보기 (1페이지 요약)

모델링 핵심 요약 카드

1. 목표: 일반화 (Generalization)

모델의 진짜 성능은 '처음 보는' 데이터(Unseen Data)에서 나옵니다. 이때의 오차(일반화 오차)를 최소화하는 것이 목표입니다. 학습 데이터 오차(Training Error)는 중요하지 않습니다.

2. 적: 과적합 (Overfitting)

모델이 너무 복잡해져서(예: 너무 높은 차수, 너무 많은 변수) 학습 데이터의 '노이즈'까지 암기하는 현상입니다. 증상: 학습 오차는 매우 낮지만, 검증 오차(일반화 오차)는 매우 높습니다.

3. 확장: 비선형 모델링

- 상호작용 항 ($X_1 X_2$): 한 변수의 효과가 다른 변수에 따라 달라지는 '시너지' 효과를 모델링합니다.
- 다항 회귀 (X, X^2, X^3): 데이터의 곡선 트렌드를 잡습니다. (주의: 스케일링 필수!)

4. 전략: 모델 선택 (Model Selection)

과소적합(너무 단순)과 과적합(너무 복잡) 사이의 균형점을 찾는 과정입니다.

- 데이터 3-분할: 학습(훈련), 검증(모델 선택/튜닝), 테스트(최종 보고).
- 방법: 하이퍼파라미터(예: 차수 M)를 바꿔가며 검증 세트 오차(Validation MSE)가 U자 곡선을 그릴 때, 가장 낮은 지점을 선택합니다.

5. 무기: K-겹 교차 검증 (K-Fold CV)

단일 검증 세트는 '우연'에 의해 잘못된 모델을 선택할 수 있습니다. (검증 세트에 과적합)
해결: 데이터를 K조각으로 나눠, K번의 (학습/검증)을 반복하고 그 오차를 평균냅니다. 이 CV 점수를 기준으로 하이퍼파라미터를 선택하는 것이 가장 신뢰할 수 있습니다.

8 초심자 FAQ

Q: 다항 회귀가 왜 '선형' 회귀인가요? 너무 헛갈립니다.

A: Y 와 X 의 관계(그래프)는 곡선(비선형)이 맞습니다. 하지만 모델을 수식으로 볼 때, $Y = \beta_0 + \beta_1 X + \beta_2 X^2$ 에서 우리가 찾아야 할 값은 $\beta_0, \beta_1, \beta_2$ 입니다. 이 계수(Coefficient) β 에 대해서는 덧셈으로만 연결되어 있으므로 '계수에 대해 선형(Linear in parameters)'이라고 부릅니다. X^2 을 \tilde{X} 라는 새로운 변수로 보면 $Y = \beta_0 + \beta_1 X + \beta_2 \tilde{X}$ 가 되어 다중 '선형' 회귀와 형태가 똑같아집니다.

Q: K-겹 교차 검증에서 K는 몇으로 정해야 하나요?

A: 정답은 없지만, 관례적으로 **K=5** 또는 **K=10**을 가장 많이 사용합니다. K가 너무 작으면 (예: K=2) 검증 데이터의 변동성이 커서 불안정하고, K가 너무 크면(예: K=N, 즉 LOOCV) 계산 시간이 매우 오래 걸립니다. 5 또는 10이 계산 비용과 추정치의 안정성 사이의 적절한 타협점으로 알려져 있습니다.

Q: 검증 세트와 테스트 세트가 뭐가 다른 건가요? 둘 다 '평가'하는 것 아닌가요?

A: 역할이 완전히 다릅니다.

- **검증 세트 (Validation Set):** '모델을 고르기 위한' 평가 세트입니다. 마치 여러 벌의 옷(모델 후보)을 입어보고(테스트) 가장 잘 어울리는 옷(최적 모델)을 '선택'하는 과정입니다.
- **테스트 세트 (Test Set):** '선택이 끝난 후' 최종적으로 한 번만 평가하는 세트입니다. 가장 잘 고른 옷을 입고 나가서 사람들(새로운 데이터)에게 '평가 보고'를 받는 과정입니다.

검증 세트로는 여러 모델을 반복적으로 테스트하지만, 테스트 세트로는 최종 선택된 단 하나의 모델만 테스트해야 합니다.

Q: 특성 스케일링(StandardScaler)은 언제나 필요한가요?

A: 항상 필수는 아니지만, 사용하는 것이 훨씬 안전합니다. 단순 선형 회귀($Y = \beta_0 + \beta_1 X$)에서는 스케일링이 결과에 영향을 주지 않습니다. 하지만 다항 회귀($X^2, X^3\dots$), 정규화 회귀(Ridge, Lasso), **KNN**, **SVM**, 신경망 등 대부분의 고급 머신러닝 모델은 특성 간의 스케일 차이에 매우 민감합니다. 따라서 모델링 전 스케일링을 적용하는 것을 습관화하는 것이 좋습니다.

December 10, 2025

- 강의명: CS109A: 데이터 과학 입문
- 주차: Lecture 07
- 교수명: Pavlos Protopapas, Kevin Rader, Chris Gumb
- 목적: Lecture 07의 핵심 개념 학습

Contents

1	개요	2
2	용어 정리	3
3	핵심 개념: 편향-분산 트레이드오프 (Bias-Variance Tradeoff)	4
3.1	모델 오차의 두 가지 근원	4
3.2	편향(Bias)과 분산(Variance)의 정의	4
3.3	트레이드오프(Tradeoff) 관계	5
4	문제 진단: 과대적합과 모델 계수	6
5	해결책: 정규화 (Regularization)	7
5.1	정규화의 핵심 아이디어	7
5.2	λ (람다)의 역할: 패널티의 강도	7
5.3	두 가지 정규화 기법: L2 (Ridge) vs. L1 (Lasso)	7
5.4	계산적 차이 및 비교 요약	8
6	절차: 최적의 λ (람다) 찾기 (Hyperparameter Tuning)	9
6.1	방법 1: 단일 검증 세트 (Single Validation Set) 사용	9
6.2	K-Fold 교차 검증 (Cross-Validation)	10
7	FAQ 및 주요 질문	11
8	1페이지 요약: 빠른 훑어보기	12

1 개요

이 문서는 머신러닝 모델의 성능을 평가하고 개선하는 핵심 원리인 **일반화(Generalization)**, **편향-분산 트레이드오프(Bias-Variance Tradeoff)**, 그리고 **정규화(Regularization)** 기법에 대해 다룹니다.

우리의 최종 목표는 훈련(training) 데이터에만 잘 맞는 모델이 아니라, 한 번도 본 적 없는 새로운 데이터(test data)에서도 좋은 성능을 내는 **'일반화 성능이 뛰어난'** 모델을 만드는 것입니다.

모델이 너무 단순하면 훈련 데이터조차 제대로 학습하지 못하며(과소적합, High Bias), 모델이 너무 복잡하면 훈련 데이터의 노이즈까지 암기해버려 새로운 데이터에서 형편없는 성능을 보입니다(과대적합, High Variance).

이 문서는 과대적합의 주된 증상(모델 계수의 폭주)을 진단하고, 이를 해결하기 위해 손실 함수(Loss Function)에 **'섀널티'**를 부과하는 정규화 기법, 특히 **릿지(Ridge, L2)**와 **라쏘(Lasso, L1)**를 중점적으로 설명합니다.

마지막으로, 이 섀널티의 강도를 조절하는 하이퍼파라미터 λ (람다)를 찾기 위한 체계적인 절차로 **검증 세트(Validation Set)**와 **교차 검증(Cross-Validation)** 방법을 단계별로 학습합니다.

2 용어 정리

핵심 용어들을 미리 이해하면 학습에 도움이 됩니다.

Table 1: 핵심 용어 정리표

용어	원어 (Full Term)	쉬운 설명 (초심자용)	비고
일반화	Generalization	모델이 훈련 데이터가 아닌 '새로운 데이터'를 얼마나 잘 맞추는지의 능력.	높을수록 좋은 모델입니다.
과소적합	Underfitting	모델이 너무 단순해서 훈련 데이터조차 제대로 학습하지 못한 상태.	편향(Bias)이 높은 상태입니다.
과대적합	Overfitting	모델이 너무 복잡해서 훈련 데이터의 '노이즈'까지 암기해버린 상태.	새로운 데이터에서 성능이 급격히 저하됩니다. 분산(Variance)이 높은 상태입니다.
편향	Bias	모델의 예측이 '실제 정답'과 평균적으로 얼마나 멀리 떨어져 있는가.	'부정확성'. 과녁의 중심을 못 맞춤.
분산	Variance	훈련 데이터가 조금 바뀔 때 모델의 예측이 얼마나 크게 출렁이는가.	'비일관성'. 쏠 때마다 탄착군이 흘어짐.
정규화	Regularization	모델의 복잡도(주로 계수 값)에 패널티를 부과하여 과대적합을 막는 기법.	"모델이 너무 복잡해지지 마!"
릿지 (L2)	Ridge Regression	계수의 '제곱의 합'에 패널티를 주는 정규화. (L_2 Norm)	계수를 0에 가깝게 줄이지만 0으로 만들진 않음.
라쏘 (L1)	Lasso Regression	계수의 '절대 값의 합'에 패널티를 주는 정규화. (L_1 Norm)	불필요한 계수를 아예 0으로 만들어 '변수 선택' 효과.
λ (람다)	Lambda	정규화의 '강도'를 조절하는 하이퍼파라미터.	0이면 정규화 안함. ∞ 면 모든 계수가 0이 됨.
하이퍼파라미터	Hyperparameter	모델이 스스로 학습하는 값(β)이 아니라, '사람이 직접 설정' 해줘야 하는 값.	λ 나 K-Fold의 K 값 등.

3 핵심 개념: 편향-분산 트레이드오프 (Bias-Variance Tradeoff)

모델의 예측 오차(Error)는 우리가 어떻게 할 수 없는 부분과, 우리가 개선할 수 있는 부분으로 나뉩니다.

3.1 모델 오차의 두 가지 근원

1. 환원 불가능한 오차 (Irreducible Error)

이 오차는 데이터 자체에 내재된 **'무작위 노이즈'** 때문에 발생합니다. 아무리 완벽한 모델을 만들어도 이 오차는 절대 0이 될 수 없습니다.

□ 예제:

비유 (Aleatoric Error): 아무리 비싼 최고급 마이크(≈모델)를 사용해도, 녹음실 주변의 공사장 소음(≈노이즈)까지 함께 녹음되는 것과 같습니다. 이 소음은 마이크 성능으로 제거할 수 없습니다.

우리는 이 오차의 존재를 인정하고, 우리가 줄일 수 있는 오차에 집중해야 합니다.

2. 환원 가능한 오차 (Reducible Error)

이 오차는 우리가 **'모델을 잘못 선택'**했기 때문에 발생합니다. 우리의 임무는 이 오차를 최소화하는 것이며, 이 오차는 다시 '편향'과 '분산'이라는 두 가지 요소로 분해됩니다.

3.2 편향(Bias)과 분산(Variance)의 정의

모델의 성능을 사격에 비유하여 편향과 분산을 이해할 수 있습니다.

편향 (Bias): 과녁을 놓치다 (부정확성)

편향은 모델의 예측값이 실제 정답(과녁의 중심)과 평균적으로 얼마나 멀리 떨어져 있는지를 나타냅니다.

- **High Bias (높은 편향):** 모델이 너무 단순하여 데이터의 복잡한 패턴을 전혀 학습하지 못합니다. (예: S자 곡선 데이터를 직선으로 예측하려는 시도)
- **결과:** 과소적합 (Underfitting). 훈련 데이터에서도, 테스트 데이터에서도 모두 성능이 나쁩니다.
- **특징:** 훈련 데이터를 바꿔가며 여러 번 학습해도 예측 결과가 거의 변하지 않습니다(낮은 분산).

분산 (Variance): 탄착군이 흩어지다 (비일관성)

분산은 훈련 데이터가 조금만 바뀌어도 모델의 예측이 얼마나 크게 변동하는지를 나타냅니다.

- **High Variance (높은 분산):** 모델이 너무 복잡하여 훈련 데이터의 사소한 노이즈까지 '암기'해버립니다.
- **결과:** 과대적합 (Overfitting). 훈련 데이터에서는 완벽(0에 가까운 오차) 하지만, 새로운 테스트 데이터에서는 성능이 재앙 수준입니다.
- **특징:** 훈련 데이터 샘플이 조금만 달라져도 모델의 형태가 스파게티 면발처럼 마구 요동칩니다.

□ 예제:

시뮬레이션 예시 (스파게티 면발): 서로 다른 2,000개의 샘플 데이터셋을 뽑아서 모델을 2,000번 학습시켰다고 가정해봅시다.

- **단순한 선형 모델 (Low Variance):** 2,000개의 예측선이 모두 비슷하게 그려집니다. (안정적)
- **복잡한 10차 다항식 모델 (High Variance):** 2,000개의 예측선이 샘플 데이터의 노이즈에 민감하게 반응하여, 마치 스파게티 면발처럼 서로 얹히고 설켜 그려집니다. (불안정)

3.3 트레이드오프(Tradeoff) 관계

편향과 분산은 한쪽이 줄어들면 다른 한쪽이 늘어나는 **'시소 관계'**에 있습니다.

모델 복잡도에 따른 오차의 변화

모델 복잡도(X축) vs. 총 오차(Y축)

- 모델이 단순할수록 (좌측):
 - 편향(Bias)은 높고 (데이터를 못 맞춤)
 - 분산(Variance)은 낮습니다. (예측이 안정적)
- 모델이 복잡해질수록 (우측):
 - 편향(Bias)은 낮아지고 (훈련 데이터를 완벽히 맞춤)
 - 분산(Variance)은 급격히 높아집니다. (데이터에 과민 반응)

총 오차(Total Error = Bias² + Variance + Irreducible Error)는 U자 형태의 곡선을 그립니다.

우리의 목표는 이 U자 곡선의 가장 낮은 지점, 즉 총 오차를 최소화하는 최적의 복잡도를 찾는 것입니다.

사격 과녁 비유 요약표

Table 2: 편향과 분산의 4가지 시나리오 (사격 비유)

	Low Variance (낮은 분산 / 일관성 ↑)	High Variance (높은 분산 / 일관성 ↓)
High Bias (높은 편향 / 정확도 ↓)	과소적합 (Underfitting) <ul style="list-style-type: none"> • 과녁의 중심은 못 맞추지만, 쏜 지점에 계속 일관되게 쏘. • (예: 단순 선형 모델) 이상적인 모델 (Ideal) <ul style="list-style-type: none"> • 과녁의 중심(정답)에 정확하고 일관되게 쏘. • 우리의 목표! 	최악의 모델 (Worst) <ul style="list-style-type: none"> • 과녁의 중심도 못 맞추고, 쏠 때마다 아무데나 흩어짐. 과대적합 (Overfitting) <ul style="list-style-type: none"> • 평균적으로 과녁 중심 근처에 맞지만(훈련 데이터), 쏠 때마다 탄착군이 너무 흩어짐. • (예: 고차 다항식 모델)
Low Bias (낮은 편향 / 정확도 ↑)		

4 문제 진단: 과대적합과 모델 계수

질문: ”모델이 과대적합(High Variance) 되었는지 어떻게 알 수 있습니까?”

답변: ”모델의 계수(coefficients, β) 값을 확인하면 됩니다.”

모델이 과대적합 상태일 때, 즉 훈련 데이터의 노이즈에 과민하게 반응할 때, 모델의 **계수(β_j) 값들은 비정상적으로 커지거나 극단적으로 불안정한 값**을 갖게 됩니다.

□ 예제:

계수 값 분포 비교 (Violin Plots): 서로 다른 2,000 개의 샘플 데이터로 2,000 개의 모델을 학습시킨 경우의 계수 분포입니다.

- 단순 선형 모델 (Low Variance): β_0, β_1 계수 값들이 0.0에서 1.25 사이의 좁은 범위에서 안정적으로 분포합니다.
- 복잡한 10차 다항식 모델 (High Variance): $\beta_5, \beta_8, \beta_9$ 등의 고차항 계수 값들이 $1e9$ (즉, 10 억) 스케일로 폭주하며, 매우 넓은 범위(큰 분산)를 갖습니다.

주의사항

과대적합의 핵심 증상: 높은 분산(High Variance) \rightarrow 불안정하고 극단적인 계수(Large β) 값
따라서, 과대적합을 막기 위한 해결책은 ”모델의 계수 값이 너무 커지지 않도록 억제하는 것”입니다.
이것이 바로 ’정규화(Regularization)’의 핵심 아이디어입니다.

5 해결책: 정규화 (Regularization)

5.1 정규화의 핵심 아이디어

정규화는 모델의 손실 함수(Loss Function)를 수정하여, 모델이 두 가지 목표를 동시에 달성하도록 강제합니다.

1. 목표 1: 데이터를 잘 맞춰라. (기존의 목표) \rightarrow 손실 함수(MSE)를 최소화. $\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$
2. 목표 2: 계수 값을 작게 유지해라. (새로운 목표: 과대적합 방지) \rightarrow 계수의 크기에 대한 '패널티 항'을 최소화.

새로운 정규화 손실 함수:

$$\mathcal{L}_{\text{REG}} = (\text{기존 MSE}) + \lambda \times (\text{패널티 항})$$

$$\mathcal{L}_{\text{REG}} = \frac{1}{n} \sum_{i=1}^n (y_i - \beta^\top x_i)^2 + \lambda L_{\text{reg}}$$

모델은 이제 MSE만 줄이는 것이 아니라, (MSE + $\lambda \times$ 패널티)의 총합을 최소화해야 합니다.

5.2 λ (람다)의 역할: 패널티의 강도

λ (람다)는 패널티의 '강도'를 조절하는 하이퍼파라미터입니다.

- 만약 $\lambda = 0$ 이라면: 패널티가 0이 됩니다. 이는 일반적인 선형 회귀와 같으며 과대적합의 위험이 있습니다.
- 만약 $\lambda \rightarrow \infty$ (무한대)라면: 패널티가 너무 강력해집니다. 모델은 MSE를 무시하고 오직 패널티($\sum \beta_j^2$ 또는 $\sum |\beta_j|$)를 0으로 만드는 데만 집중합니다. 그 결과 모든 계수 β_j 가 0이 되어, 모델은 단순한 수평선(평균값)이 됩니다(과소적합, High Bias).

▣ 핵심 요약

우리의 목표는 훈련 데이터와 검증 데이터를 사용하여 편향과 분산 사이의 균형을 잡는 '최적의 λ '를 찾는 것입니다.

5.3 두 가지 정규화 기법: L2 (Ridge) vs. L1 (Lasso)

패널티 항(L_{reg})을 어떻게 정의하느냐에 따라 릿지(Ridge)와 라쏘(Lasso)로 나뉩니다.

1. L2 정규화: 릿지 회귀 (Ridge Regression)

패널티 항: 계수의 **제곱의 합** (L_2 Norm) $\rightarrow L_{\text{reg}} = \sum_{j=1}^J \beta_j^2$

최종 손실 함수:

$$\mathcal{L}_{\text{RIDGE}} = \frac{1}{n} \sum_{i=1}^n (y_i - \beta^\top x_i)^2 + \lambda \sum_{j=1}^J \beta_j^2$$

- **특징:** 계수 값이 커질수록 패널티가 '제곱'으로 증가하므로, 매우 큰(튀는) 계수 값을 강력하게 억제합니다. 모든 계수를 0에 '가깝게' 줄이지만, 정확히 0으로 만들지는 않습니다.
- **장점:** 계산이 빠릅니다 (수학적인 공식, 즉 'Analytical Solution'이 존재함). 다중공선성 (Multi-collinearity: 예측 변수 간 강한 상관관계)이 있을 때 모델을 안정화시키는 데 매우 효과적입니다.
- **적합한 상황:** 모든 변수(feature)가 예측에 어느 정도 기여한다고 판단될 때 사용합니다.

2. L1 정규화: 라쏘 회귀 (Lasso Regression)

패널티 항: 계수의 **절대값의 합** (L_1 Norm) $\rightarrow L_{\text{reg}} = \sum_{j=1}^J |\beta_j|$

최종 손실 함수:

$$\mathcal{L}_{\text{LASSO}} = \frac{1}{n} \sum_{i=1}^n (y_i - \beta^\top x_i)^2 + \lambda \sum_{j=1}^J |\beta_j|$$

- **특징:** 중요하지 않은 변수의 계수는 '정확히 0'으로 만들어 버립니다. 이는 모델에서 해당 변수를 아예 '제거'하는 것과 같은 효과를 줍니다.
- **장점:** 모델의 복잡도를 근본적으로 낮추는 **자동 변수 선택(Feature Selection)** 기능이 있습니다. 해석하기 쉬운 단순한 모델을 만듭니다.
- **적합한 상황:** 수백, 수천 개의 변수 중 실제 중요한 변수는 몇 개 안 된다고 의심될 때 매우 유용합니다.

주의사항

주의: 절편(β_0)은 정규화하지 않습니다. L_1, L_2 패널티 항은 β_1 부터 β_J 까지만 적용됩니다. 절편(intercept) β_0 는 특정 변수와 연결된 민감도가 아니라, 모델 전체의 기본 '높낮이(offset)'를 조절할 뿐이므로 패널티 대상에서 제외합니다.

5.4 계산적 차이 및 비교 요약

- **릿지(Ridge):** 행렬을 이용한 명확한 수학 공식 (Analytical Solution)으로 β 값을 한 번에 계산할 수 있습니다.

$$\hat{\beta}_{\text{Ridge}} = (X^\top X + \lambda I)^{-1} X^\top Y$$

- **라쏘(Lasso):** 절대값 함수는 0에서 미분이 불가능하므로, 이런 수학 공식이 없습니다. 대신 '수치적 최적화(Numerical Optimization)' 기법 (예: Solver, Coordinate Descent)을 사용하여 반복적으로 β 값을 찾아가야 하므로, 릿지보다 계산 속도가 느릴 수 있습니다.

Table 3: Ridge (L^2) vs. Lasso (L^1) 비교 요약

구분	릿지 회귀 (Ridge, L2)	라쏘 회귀 (Lasso, L1)
패널티 항	계수의 제곱의 합 ($\sum \beta_j^2$)	계수의 절대값의 합 ($\sum \beta_j $)
계수 축소	계수를 0에 가깝게 줄임 (0은 안 됨)	불필요한 계수를 정확히 0으로 만듦
핵심 기능	모델 안정화, 다중공선성 제어	변수 선택 (Feature Selection)
계산 방식	빠름 (Analytical Solution 존재)	느림 (Numerical Solver 필요)
도형적 해석	패널티 영역이 '원' (Circle) 형태	패널티 영역이 '다이아몬드' (Diamond) 형태

6 절차: 최적의 λ (람다) 찾기 (Hyperparameter Tuning)

λ 는 하이퍼파라미터입니다. 즉, 모델이 스스로 학습하는 값이 아니라 우리가 정해줘야 하는 값입니다. 최적의 λ 를 찾는 과정을 '튜닝(Tuning)'이라고 합니다.

주의사항

하이퍼파라미터 튜닝의 철칙:

- 훈련(Train) 데이터로 튜닝하면 안 됩니다. (모델이 $\lambda = 0$ 을 선호하게 됨)
- 테스트(Test) 데이터로 튜닝하면 절대 안 됩니다. (정보 유출, 즉 'Cheating' 임)
- 오직 검증(Validation) 데이터 또는 교차 검증(Cross-Validation)을 사용해야 합니다.

6.1 방법 1: 단일 검증 세트 (Single Validation Set) 사용

가장 기본적인 방법으로, 데이터를 Train / Validation / Test 세 부분으로 나누어 진행합니다.

- Step 1: 데이터 분할 (Split Data) 데이터를 훈련(Train), 검증(Validation), 테스트(Test)용으로 3-way 분할합니다.
- Step 2: λ 후보군 설정 (Select λ range) 테스트할 λ 값들의 목록을 만듭니다. (예: '[0.0001, 0.001, 0.01, 0.1, 1, 10, 100]')
- Step 3: 모델 훈련 (Train Models) 훈련(Train) 데이터를 사용하여, 각 λ 후보마다 정규화(Ridge/ Lasso) 모델을 학습시키고 계수(β_λ)를 얻습니다.
- Step 4: 검증 및 MSE 기록 (Validate) 검증(Validation) 데이터를 사용하여, 3단계에서 얻은 각 모델(β_λ)의 성능(MSE)을 측정합니다.

주의사항

매우 중요! 이 단계에서 성능을 측정할 때는 패널티 항(λL_{reg})을 제외하고, 순수한 MSE 값만 계산합니다.

이유: λ 는 모델을 '훈련' 시킬 때 과대적합을 막기 위한 도구일 뿐, 모델의 '순수한 예측 성능'을 평가할 때는 MSE(실제 정답과의 차이)만 보는 것이 타당합니다.

- Step 5: 최적 λ^* 선정 (Select Best λ) 4단계에서 기록한 MSE 값들 중, 가장 낮은 MSE를 기록한 λ 를 최종 λ^* (람다-스타)로 선정합니다.
- Step 6: (권장) 모델 재훈련 (Refit Model) 최적의 하이퍼파라미터 λ^* 를 찾았으므로, 이제 '검증'

세트'의 임무는 끝났습니다. 훈련(Train) 데이터와 검증(Validation) 데이터를 다시 하나로 합친 더 큰 데이터셋을 사용하여, λ^* 값으로 모델을 단 한 번 재훈련합니다.

이유: 모델 선택이 끝났으니, 검증에 썼던 데이터도 훈련에 사용하여 최종 모델이 조금이라도 더 많은 정보를 학습하게 합니다.

7. **Step 7: 최종 평가(Final Report)** 지금까지 단 한 번도 사용하지 않은 테스트(Test) 데이터를 사용하여, 6단계에서 얻은 최종 모델의 성능(MSE)을 평가하고 이 점수를 보고합니다.

6.2 방법 2: K-Fold 교차 검증 (Cross-Validation)

단일 검증 세트는 데이터가 어떻게 분할되었느냐에 따라 '운' 좋게 특정 λ 에 유리한 결과가 나올 수 있습니다. (e.g., 검증 데이터가 우연히 직선 형태)

K-Fold 교차 검증(CV)은 이 '운'의 요소를 제거하여 더 안정적이고 신뢰할 수 있는 λ 를 찾는, 더 강력한 방법입니다.

1. **Step 1: 데이터 분할(Split Data)** 데이터를 (훈련+검증)용 훈련 풀(Training Pool)과 테스트(Test)용으로 2-way 분할합니다.
2. **Step 2: λ 후보군 선정(Select λ range)** (\circ) 전과 동일. 예: '[0.001, 0.1, 1, 10, 100]'
3. **Step 3: K-Fold 분할(Split K-Folds)** 훈련 풀(Training Pool)을 K개 (예: 5개)의 '폴드(fold)'로 균등하게 나눕니다.
4. **Step 4: K-Fold CV 루프 실행(Run CV Loop)** K번 반복합니다. (예: $k = 1$ 부터 5까지)
 - **For $k = 1$:** Fold 1을 '검증용', Fold 2~5를 '훈련용'으로 사용.
 - **For $k = 2$:** Fold 2를 '검증용', Fold 1, 3~5를 '훈련용'으로 사용.
 - ... (K번 반복) ...
 각 k 번째 반복마다, 모든 λ 후보에 대해 모델을 훈련하고 검증용 폴드의 MSE를 기록합니다.
5. **Step 5: 평균 MSE 계산(Average MSEs)** 4단계가 끝나면, 각 λ 후보마다 K개의 MSE 값 (예: $\lambda = 0.1$ 일 때 5개의 MSE)이 쌓입니다. 각 λ 별로 K개의 MSE 값의 평균을 계산합니다. (예: ' $\text{AvgMSE}[\lambda = 0.1] = (\text{MSE}_{k1} + \dots + \text{MSE}_{k5})/5$ '')
6. **Step 6: 최적 λ^* 선정(Select Best λ)** 5단계에서 계산한 평균 MSE 값들 중, 가장 낮은 평균 MSE를 기록한 λ 를 최종 λ^* 로 선정합니다.
7. **Step 7: 모델 재훈련(Refit Model)** K-Fold CV는 오직 λ^* 를 찾기 위한 과정이었습니다. 이제 훈련 풀 전체(1~5 Fold 모두)를 사용하여, λ^* 값으로 모델을 단 한 번 재훈련합니다.
이유: K개의 모델 중 하나를 고르는 것이 아니라, 찾은 최적의 λ 를 사용하여 *모든* 훈련 데이터를 학습한 최종 모델을 얻기 위함입니다.
8. **Step 8: 최종 평가(Final Report)** 지금까지 단 한 번도 사용하지 않은 테스트(Test) 데이터로 7단계의 최종 모델 성능을 평가하고 보고합니다.

7 FAQ 및 주요 질문

Q: 릿지(Ridge)와 라쏘(Lasso) 중 무엇을 써야 하나요?

A: 정답은 없습니다. 상황에 따라 다르며, 둘 다 시도하고 교차 검증(CV) 점수를 비교하는 것이 가장 좋습니다.

- **라쏘(Lasso)가 유리할 때:** 변수가 수백 수천 개로 매우 많고, 그 중 '소수의 핵심 변수'만 예측에 중요하다고 의심될 때. 라쏘가 불필요한 변수들을 0으로 만들어 변수 선택(Feature Selection)을 자동으로 해줍니다.
- **릿지(Ridge)가 유리할 때:** 모든 변수가 예측에 조금씩이라도 기여한다고 생각될 때. 특히 변수들 간에 강한 상관관계(다중공선성)가 있을 때, 라쏘보다 더 안정적인 성능을 보입니다.

Q: λ 탐색 범위를 정했는데, 최적값이 범위의 경계(예: 0.001 또는 100)에서 나왔습니다.

A: 탐색 범위를 더 넓혀야 합니다. 만약 $\lambda = 100$ 에서 MSE가 최소였다면, 이는 $\lambda = 1000, 10000$ 일 때 MSE가 더 낮아질 가능성이 있다는 신호입니다. 최적의 λ 는 U자형 MSE 곡선의 '바닥'에 있어야 합니다. 탐색 범위의 경계에서 최적값이 나왔다면, 아직 U자 곡선의 바닥을 찾지 못했다는 뜻이므로 범위를 확장하여 다시 시도해야 합니다.

Q: (단일 검증) Step 6에서 왜 검증(Validation) 세트를 다시 훈련(Train) 세트에 합쳐서 재훈련하나요?

A: 검증 세트의 유일한 임무는 '최적의 하이퍼파라미터(λ^*)를 찾는 것'이었습니다. 일단 λ^* 를 찾았다면, 검증 세트는 더 이상 필요 없습니다. 이 데이터를 '버리기'보다는, 최종 모델을 훈련시킬 때 훈련 데이터에 다시 합쳐서 모델이 조금이라도 더 많은 데이터를 학습하게 하는 것이 성능에 유리합니다.

단, 테스트(Test) 세트는 절대 훈련에 사용해서는 안 됩니다.

Q: K-Fold CV에서 K는 몇으로 정해야 하나요?

A: 일반적으로 $K=5$ 또는 $K=10$ 을 가장 많이 씁니다. K 역시 하이퍼파라미터지만, K 값을 튜닝하기 위해 또 CV를 하는 것은 비효율적일 수 있습니다. (예: $K = 5$ 일 때와 $K = 10$ 일 때의 최종 성능 차이가 미미한 경우가 많음)

$K = 5$ 또는 $K = 10$ 정도로도 충분히 안정적인 λ 값을 찾을 수 있습니다.

8 1페이지 요약: 빠른 훑어보기

▣ 핵심 요약

핵심 목표: 일반화 (Generalization) 훈련 데이터가 아닌, '새로운 데이터'를 잘 맞추는 모델을 원한다.

1. 문제: 편향 vs. 분산 (Bias vs. Variance)

- **High Bias (과소적합):** 모델이 너무 단순함. (예측이 정답과 떨어짐)
- **High Variance (과대적합):** 모델이 너무 복잡함. (예측이 데이터마다 널뛰기 함)

2. 진단: 과대적합의 징후

모델의 계수(β) 값이 비정상적으로 크고 불안정해진다. (\rightarrow "계수 값을 줄여야 한다!")

3. 해결: 정규화 (Regularization)

손실 함수에 '패널티 항'을 추가하여 계수 값이 커지는 것을 억제한다.

$$\text{New Loss} = \text{MSE} + \lambda \times (\text{Penalty})$$

4. 방법: Ridge (L2) vs. Lasso (L1)

- **Ridge (L2)** $\rightarrow \sum \beta_j^2$: 계수를 0에 가깝게 줄인다. (안정성 \uparrow)
- **Lasso (L1)** $\rightarrow \sum |\beta_j|$: 계수를 정확히 0으로 만든다. (변수 선택 가능)

5. 튜닝: 최적의 λ 찾기 (by CV)

1. λ 후보 목록을 만든다. (e.g., [0.01, 0.1, 1, 10])
2. 각 λ 마다 K-Fold CV를 실행하여 평균 검증 MSE를 계산한다.
3. 평균 검증 MSE가 가장 낮은 λ^* 를 선택한다.
4. (훈련+검증) 전체 데이터로 λ^* 를 적용하여 최종 모델을 재훈련한다.
5. 테스트 (Test) 세트로 최종 성능을 딱 1번 보고한다.

December 10, 2025

- 강의명: CS109A: 데이터 과학 입문
- 주차: Lecture 08
- 교수명: Pavlos Protopapas, Kevin Rader, Chris Gumb
- 목적: Lecture 08의 핵심 개념 학습

▣ 핵심 요약

이 문서는 선형 회귀 모델의 단순한 예측을 넘어, 우리가 얻은 모델이 얼마나 신뢰할 수 있는지(정확성), 모델에 포함된 변수들이 실제로 의미가 있는지(유의성), 그리고 모델의 예측이 얼마나 확실한지(예측 구간)를 평가하는 통계적 추론 방법을 다룹니다.

데이터에 존재하는 불확실성을 이해하고, '부트스트래핑'이라는 시뮬레이션 기법을 통해 회귀 계수 ($\hat{\beta}$)의 분포를 추정합니다. 이를 바탕으로 신뢰구간을 계산하고, t -검정 및 p-값을 이용해 각 예측 변수의 중요도와 통계적 유의성을 검증하는 방법을 배웁니다. 마지막으로 모델의 '평균 예측'에 대한 신뢰구간과 '개별 예측'에 대한 예측구간의 차이점을 명확히 구분합니다.

Contents

1 개요: 왜 '추론'이 필요한가?	2
2 핵심 용어 정리	3
3 1부: 추정치의 정확성 평가 (Accuracy of Estimates)	4
3.1 문제 제기: 불확실성은 어디에서 오는가?	4
3.2 부트스트래핑 (Bootstrapping): '평행 우주' 시뮬레이션	4
3.3 신뢰구간: β 의 분포에서 정확성 찾기	4
3.3.1 신뢰구간 (Confidence Interval, CI) 계산법	5
3.3.2 표준 오차 (Standard Error, SE)	5
4 2부: 예측 변수의 유의성 평가 (Significance)	6
4.1 무엇이 '중요한' 예측 변수인가?	6
4.2 t -검정 통계량: 신뢰도를 반영한 중요도	6
4.3 p-값: '가장 중요한' 것이 '유의미'한가?	6
4.4 p-값의 정의와 해석	7

5 3부: 예측의 불확실성 (Uncertainty in Predictions)	8
5.1 함수 f 의 신뢰구간 (CI for f)	8
5.2 새로운 y 의 예측구간 (PI for y)	8
5.3 구간의 '나팔' 모양	8
6 빠르게 훑어보기 (1-Page Summary)	10

1 개요: 왜 '추론'이 필요한가?

선형 회귀 모델을 학습시키면 $\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x$ 와 같은 식을 얻습니다. 예를 들어, TV 광고 예산(x)과 매출(y) 간의 관계가 $\hat{y} = 1.01x + 0.005$ 라고 가정해 봅시다.

이 식의 해석은 ”TV 광고 예산을 \$1,000 늘리면 매출이 \$1,010 증가한다(순이익 \$10)”입니다. 하지만 이 ’1.01’이라는 숫자는 우리가 가진 *하나의* 데이터 샘플로부터 얻은 *추정치*($\hat{\beta}_1$) 일 뿐입니다.

만약 우리가 다른 날짜에 데이터를 다시 수집했다면(다른 ’실현’ or ’realization’) 어땠을까요? 아마도 $\hat{\beta}_1 = 1.03$ 이나 $\hat{\beta}_1 = 0.98$ 처럼 미묘하게 다른 값을 얻었을 것입니다.

통계적 추론(Inference)은 이러한 불확실성을 다루는 학문입니다. 이 문서의 목표는 다음과 같은 질문에 답하는 것입니다.

- 정확성 (Accuracy): 우리가 얻은 $\hat{\beta}_1 = 1.01$ 이라는 값은 얼마나 정확하고 신뢰할 수 있는가? (1부)
- 유의성 (Significance): $\hat{\beta}_1$ 이 ’0’과 충분히 멀리 떨어져 있는가? 즉, TV 광고(x)가 매출(y)에 *정말로* 영향을 미치는가, 아니면 그냥 우연인가? (2부)
- 예측 불확실성 (Prediction Uncertainty): 모델이 예측한 값 \hat{y} 는 얼마나 믿을 수 있는가? (3부)

2 핵심 용어 정리

본격적인 학습에 앞서 주요 용어들을 정리합니다.

Table 1: 선형 회귀 추론 핵심 용어

용어	원어 (English)	쉬운 설명
추론	Inference	제한된 데이터(샘플)를 가지고 더 큰 모집단의 특성을 추측하는 과정
$\hat{\beta}$ (계수 추정치)	Coefficient Estimate	우리가 가진 데이터로 계산한 회귀 계수. (e.g., 1.01)
부트스트래핑	Bootstrapping	원본 데이터에서 '복원 추출'을 반복하여 가상의 데이터셋들을 만드는 기법
복원 추출	Sampling with Replacement	데이터를 뽑은 후 다시 집어 넣고 다음 데이터를 뽑는 방식 (중복 허용)
신뢰구간	Confidence Interval (CI)	실제 모수(e.g., 진짜 β_1)가 포함될 것이라 95% 신뢰하는 범위
표준 오차	Standard Error (SE)	$\hat{\beta}$ 값들이 평균으로부터 얼마나 흩어져 있는지를 나타내는 표준편차
가설 검정	Hypothesis Testing	"효과가 없다"(H_0)는 주장이 맞는지 데이터로 검증하는 절차
귀무가설 (H_0)	Null Hypothesis	"아무런 효과가 없다"는 기본 가정. (e.g., " $\beta_1 = 0$ 이다.")
t-검정 통계량	t-test statistic	계수 값이 0으로부터 표준 오차의 몇 배만큼 떨어져 있는지(신호 대 잡음비)
p-값	p-value	귀무가설(H_0)이 맞다고 할 때, 현재 데이터(혹은 더 극단적인)가 *우연히* 관찰될 확률. (작을수록 H_0 가 틀렸다고 확신)
예측구간	Prediction Interval (PI)	*새로운* 데이터 포인트 y 하나가 존재할 것이라 95% 신뢰하는 범위

3 1부: 추정치의 정확성 평가 (Accuracy of Estimates)

3.1 문제 제기: 불확실성은 어디에서 오는가?

우리가 가진 데이터는 완벽하지 않습니다. 불확실성(오차)의 원인은 크게 두 가지입니다.

1. 우연적 오차 (Aleatoric / Irreducible Error, ϵ)

- 시스템에 본질적으로 내재된 무작위성 또는 노이즈입니다.
- 예: 동일한 광고비를 써도 날씨, 경쟁사 프로모션 등 '측정되지 않은' 요인 때문에 매출이 매번 다르게 나옵니다.
- 이는 모델을 아무리 개선해도 줄일 수 없는 '축소 불가능한 오차'입니다.

2. 인식론적 오차 (Epistemic / Misspecification Error)

- 우리가 모델을 잘못 설정했거나(e.g., 비선형인데 선형으로 가정) 데이터가 부족해서 발생하는 오차입니다.
- 우리가 가진 데이터는 수많은 가능한 현실 중 하나의 '실현(realization)' 일 뿐입니다.

이러한 오차 때문에, 우리가 데이터를 다시 수집하면 $\hat{\beta}$ 값도 계속 바뀔 것입니다. 우리의 목표는 이 $\hat{\beta}$ 가 얼마나 변동(Variability)하는지 파악하는 것입니다.

3.2 부트스트래핑 (Bootstrapping): '평행 우주' 시뮬레이션

$\hat{\beta}$ 의 변동성을 알려면 "평행 우주"에서 데이터를 여러 번 가져와 $\hat{\beta}$ 를 여러 번 계산해 보면 됩니다. 하지만 현실에선 불가능합니다.

해결책: 부트스트래핑 (Bootstrapping) 우리가 가진 원본 데이터셋(크기 N)을 '모집단' 그 자체라고 간주하고, 이로부터 가상의 '평행 우주' 데이터셋들을 생성하는 기법입니다.

▣ 핵심 정보

직관적 비유: 주머니 속 공 뽑기

1. 우리에게 $N = 5$ 개의 공(데이터)이 담긴 주머니(원본 데이터셋)가 있습니다. (공 번호: 1, 3, 5, 8, 9)
2. 이 주머니에서 공을 하나 꺼내 번호를 확인하고(e.g., 8번), 복제본을 만든 뒤, 새 주머니로 옮깁니다.
3. 중요: 꺼냈던 8번 공은 다시 원본 주머니에 집어넣습니다. (이것이 '복원 추출'입니다.)
4. 다시 주머니에서 공을 꺼냅니다. 아까 뽑았던 8번이 또 나올 수도 있습니다. (e.g., 8번) 복제본을 새 주머니로 옮깁니다.
5. 이 과정을 원본 크기 $N = 5$ 가 될 때까지 반복합니다.
6. 결과:
 - 원본 샘플: {1, 3, 5, 8, 9}
 - 첫 번째 부트스트랩 샘플: {8, 8, 3, 5, 1} (8번 중복, 9번 누락)
7. 이 16의 과정을 S 번(e.g., 1000번) 반복하여 S 개의 '부트스트랩 샘플'(평행 우주)을 만듭니다.

3.3 신뢰구간: β 의 분포에서 정확성 찾기

부트스트래핑으로 S 개의 '가상' 데이터셋을 만들었습니다. 이제 $\hat{\beta}$ 의 분포를 찾을 수 있습니다.

절차:

1. S 개의 각 부트스트랩 샘플에 대해 선형 회귀 모델을 학습시킵니다.
2. S 개의 서로 다른 $\hat{\beta}^{(1)}, \hat{\beta}^{(2)}, \dots, \hat{\beta}^{(S)}$ 값들을 얻습니다.

3. 이 값들을 모아 히스토그램을 그리면 $\hat{\beta}$ 의 (근사적인) 샘플링 분포가 됩니다.

이 분포가 바로 $\hat{\beta}$ 의 불확실성을 시각적으로 보여줍니다. 분포가 좁으면(표준편차가 작으면) 추정치가 매우 정확한 것이고, 넓으면(표준편차가 크면) 부정확한 것입니다.

[참고 이미지: $\hat{\beta}_1$ 의 부트스트랩 분포 - TV 광고는 변동성이 작고, Radio 광고는 변동성이 큼]

3.3.1 신뢰구간 (Confidence Interval, CI) 계산법

이 분포를 사용해 ”실제 β 값이 존재할 것이라 95% 신뢰하는 구간”을 계산할 수 있습니다.

방법 (가정 없는 백분위수 방식):

1. S 개의 $\hat{\beta}$ 값들을 크기순으로 정렬합니다. (e.g., $S = 1000$ 개)
2. 95% 신뢰구간을 원한다면, 하위 2.5%와 상위 2.5%를 잘라냅니다.
3. 즉, 정렬된 값 중에서 [2.5 백분위수]와 [97.5 백분위수] 값을 찾습니다.
4. (e.g., 1000개 샘플 중 25번째 값과 975번째 값)
5. 예시: [11.50, 12.26, 12.81, ..., 15.21]
 - ‘np.percentile(betas, 2.5)’ → 12.80 (하한)
 - ‘np.percentile(betas, 97.5)’ → 13.71 (상한)
 - 95% CI = [12.80, 13.71]

해석: 우리의 $\hat{\beta}$ 추정치가 이 과정을 통해 얻어졌을 때, 실제(참) β 값이 [12.80, 13.71] 구간 내에 존재한다고 95% 신뢰할 수 있습니다.

3.3.2 표준 오차 (Standard Error, SE)

신뢰구간 외에 불확실성을 요약하는 또 다른 값입니다.

- 표준 오차 (SE): S 개 $\hat{\beta}$ 값들의 표준편차입니다. ($\sigma_{\hat{\beta}}$)
- 근사적 신뢰구간: 만약 $\hat{\beta}$ 의 분포가 정규분포(종 모양)를 따른다고 *가정*한다면, 95% CI는 대략 다음과 같이 근사할 수 있습니다.

$$95\% \text{ CI} \approx [\mu_{\hat{\beta}} - 2 \times SE_{\hat{\beta}}, \quad \mu_{\hat{\beta}} + 2 \times SE_{\hat{\beta}}]$$

(여기서 $\mu_{\hat{\beta}}$ 는 S 개 $\hat{\beta}$ 값들의 평균입니다.)

주의사항

표준 편차(Standard Deviation) vs. 표준 오차(Standard Error)

- 표준 편차 (SD): 데이터 포인트(y)가 평균(\bar{y})으로부터 얼마나 흩어져 있는가? (데이터 자체의 변동성)
- 표준 오차 (SE): 추정치($\hat{\beta}$)가 실제 모수(β)로부터 얼마나 흩어져 있을 것으로 *예상*되는가? (추정치의 변동성)

부트스트래핑에서는 $\hat{\beta}$ 샘플들의 표준편차를 표준 오차(SE)로 사용합니다.

4 2부: 예측 변수의 유의성 평가 (Significance)

4.1 무엇이 '중요한' 예측 변수인가?

이제 우리는 각 예측 변수(e.g., TV, Radio, Newspaper)의 $\hat{\beta}$ 분포를 알고 있습니다. 어떤 변수가 결과(매출)에 가장 큰 영향을 미칠까요?

- 단순한 방법: $\hat{\beta}$ 의 평균 값 $|\mu_{\hat{\beta}}|$ 이 가장 큰 변수.
- 문제점: Figure ??에서 보듯이, Radio 광고 ($\mu_{\beta_1} = -0.05, \sigma_{\beta_1} = 0.1$)는 평균은 0에 가깝지만 변동성이 매우 큽니다. 이는 실제 값이 0.15일 수도, -0.25일 수도 있음을 의미합니다 (신뢰할 수 없음).
- 반면 TV 광고 ($\mu_{\beta_1} = 0.05, \sigma_{\beta_1} = 0.005$)는 값은 작지만 변동성이 매우 적어, 0이 아니라고 확실히 말할 수 있습니다.

즉, '중요도'는 계수의 크기(신호)뿐만 아니라 불확실성(잡음)도 함께 고려해야 합니다.

4.2 \hat{t} -검정 통계량: 신뢰도를 반영한 중요도

이 '신호 대 잡음비'를 측정하는 지표가 바로 \hat{t} -검정 통계량입니다. (강의에서는 \sqrt{n} 을 생략한 \hat{t} -test hat 을 사용)

$$\hat{t}\text{-test} = \frac{\mu_{\hat{\beta}}}{\sigma_{\hat{\beta}}} = \frac{\text{계수 평균 (신호)}}{\text{표준 오차 (잡음)}}$$

이 값은 "계수 평균이 0으로부터 표준 오차의 몇 배만큼 떨어져 있는가?"를 의미합니다. \hat{t} 통계량의 절댓값이 클수록, 그 변수는 불확실성에 비해 더 강력한 신호를 가집니다.

▣ 핵심 요약

특정 중요도 순위의 변화

캘리포니아 주택 가격 데이터 예시에서, 중요도 순위는 어떤 지표를 쓰느냐에 따라 달라집니다.

- '평균 방수(AveBedrms)'는 계수 자체는 크지만, 부트스트래핑 결과 불확실성(SE)이 매우 커서 \hat{t} -test 순위는 낮아졌습니다.
- 반면 '중간 소득(MedInc)'은 계수 값도 크고 불확실성도 낮아, 가장 신뢰할 수 있는 중요한 예측 변수로 선정되었습니다.

4.3 p-값: '가장 중요한' 것이 '유의미'한가?

\hat{t} -test로 '가장 중요한' 변수(MedInc)를 찾았습니다. 하지만 이런 질문이 남습니다.

"이 변수들이 모두 쓰레기(junk)이고, MedInc는 단지 '쓰레기 중 유품'인 것은 아닐까?"

즉, MedInc가 매출에 미치는 영향이 실제로 존재하는지, 아니면 우리가 가진 데이터에서 우연히 그렇게 보인 것인지 검증해야 합니다. 이것이 가설 검정(Hypothesis Testing)입니다.

- 귀무가설 (H_0): "이 예측 변수는 y 에 아무런 영향을 미치지 않는다."
 - (수학적 표현: 실제 $\beta = 0$ 이다.)
 - (이 경우, 우리가 관찰한 \hat{t} -test 값은 순전히 우연(random chance)의 산물이다.)
- 대립가설 (H_1): "이 예측 변수는 y 에 유의미한 영향을 미친다."

- (수학적 표현: $\beta \neq 0$ 이다.)

4.4 p-값의 정의와 해석

가설 검증은 H_0 가 맞다고 가정하고 시작합니다.

검증 절차:

1. H_0 가 맞다고 가정합니다. (즉, x 와 y 는 아무 관계가 없다. $\beta = 0$ 이다.)
2. 이 가정 하에서, 데이터를 랜덤 노이즈로 간주하고 t -test 값을 계산합니다.
3. 이 과정을 수없이 반복하면, '순전히 우연'에 의해 발생하는 t -test 값들의 분포(Null Distribution)를 얻을 수 있습니다. (이 분포는 **Student's t-distribution**으로 알려져 있습니다.)
4. 핵심 질문: 이 '우연의 분포'에서, 우리가 실제 데이터로 계산한 t -test 값 (e.g., $t^* = 49$) 또는 그보다 더 극단적인 값이 관찰될 확률은 얼마인가?
5. 이 확률이 바로 **p-값 (p-value)**입니다.

$$p\text{-value} = P(|t_{\text{random}}| \geq |t^*_{\text{our_model}}| \mid H_0 \text{ is true})$$

[참고 이미지: p-값의 시각적 이해 - t-분포 곡선에서 극단적인 꼬리 영역의 면적]

p-값 해석:

- **p-값이 크다 (e.g., p = 0.5):**
 - ”우리가 관찰한 t^* 값은 ‘아무 효과가 없다’고 가정해도 50%의 확률로 우연히 발생할 수 있다.”
 - $\rightarrow H_0$ 를 기각할 근거가 부족하다. (변수가 유의미하다고 말할 수 없다.)
- **p-값이 작다 (e.g., p = 0.01):**
 - ”우리가 관찰한 t^* 값은 ‘아무 효과가 없다’고 가정하면 단 1%의 확률로만 우연히 발생할 수 있다. (매우 희귀한 일)”
 - \rightarrow ”우연이라기엔 너무 극단적이다. H_0 가정이 틀렸을 것이다.”
 - $\rightarrow H_0$ 를 기각하고 H_1 을 채택한다. (변수가 통계적으로 유의미하다고 결론)

유의 수준 (Significance Level): 통상적으로 $p < 0.05$ (5%)를 H_0 기각의 기준으로 삼습니다.

5 3부: 예측의 불확실성 (Uncertainty in Predictions)

지금까지 β 계수 자체의 불확실성을 다뤘습니다. 이제 모델의 예측값 \hat{y} 의 불확실성을 다룹니다.

5.1 함수 f 의 신뢰구간 (CI for f)

부트스트래핑을 통해 S 개의 다른 모델 $\hat{f}^{(i)}(x) = \hat{\beta}_0^{(i)} + \hat{\beta}_1^{(i)}x$ 을 얻었습니다. 이 S 개의 회귀선을 모두 그리면 '스파게티 플롯(spaghetti plot)' 이 됩니다.

[참고 이미지: 1000개의 부트스트랩 모델(회귀선)을 그린 '스파게티 플롯']

이 플롯은 $\hat{\beta}$ 의 불확실성이 $\hat{f}(x)$ 예측의 불확실성으로 어떻게 전파되는지 보여줍니다.

- 특정 x 값(e.g., TV Budget = 200)에서, 1000개의 다른 예측값 $\hat{f}^{(i)}(200)$ 이 나옵니다.
- 이 값들의 95% 백분위수 구간을 계산할 수 있습니다.
- 이 작업을 모든 x 에 대해 수행하여 연결하면, 함수 f 에 대한 95% 신뢰구간(Confidence Interval) 밴드(band)가 만들어집니다.

해석: ”우리는 실제 평균 응답을 나타내는 회귀선 $f(x)$ 가 95%의 신뢰로 이 밴드(연한 하늘색 영역) 안에 있다고 믿는다.”

5.2 새로운 y 의 예측구간 (PI for y)

$f(x)$ 의 신뢰구간은 모델 자체의 불확실성(즉, $\hat{\beta}$ 가 β 와 다를 수 있는 오차)만 반영합니다.

하지만 우리가 예측하려는 새로운 관측치 y 는 모델 $f(x)$ 에 더해 축소 불가능한 오차 ϵ 까지 포함하고 있습니다. ($y = f(x) + \epsilon$)

[참고 이미지: 신뢰구간(CI)과 예측구간(PI)의 비교 - PI가 ϵ 오차를 추가로 포함하므로 항상 더 넓음]

따라서 '새로운 y 값'이 존재할 범위를 나타내는 예측구간(Prediction Interval)은 $f(x)$ 의 신뢰구간보다 항상 더 넓습니다.

주의사항

신뢰구간(CI) vs. 예측구간(PI)

두 개념은 예측 대상이 다릅니다.

- 신뢰구간(CI): ”TV 예산이 \$200,000 일 때, 평균 매출($f(x)$)은 95% 신뢰로 [\$16.5M, \$17.5M] 사이에 있을 것이다.”
 - (오직 $\hat{\beta}$ 의 불확실성만 고려)
- 예측구간(PI): ”TV 예산이 \$200,000 인 새로운 매장 하나의 개별 매출(y)은 95% 확률로 [\$14.0M, \$20.0M] 사이에 있을 것이다.”
 - ($\hat{\beta}$ 의 불확실성 + 개별 오차 ϵ 의 불확실성 모두 고려)

5.3 구간의 '나팔' 모양

Figure ??에서 볼 수 있듯이, 신뢰구간과 예측구간은 데이터의 중심(e.g., x 의 평균 \bar{x})에서 가장 좁고, 양 끝으로 갈수록 넓어지는 '나팔' 또는 '깔때기' 모양을 띍니다.

이유: 회귀선은 데이터의 중심(\bar{x}, \bar{y})을 축으로 회전하는 경향이 있습니다. $\hat{\beta}_1$ (기울기)의 작은 불확실성

이라도, 중심에서 멀어질수록 \hat{y} 예측값에 큰 차이를 만들어내기 때문입니다.

6 빠르게 훑어보기 (1-Page Summary)

▣ 핵심 요약

1. 부트스트래핑 (Bootstrapping)

- 목적: $\hat{\beta}$ 추정치의 불확실성(변동성)을 알기 위해.
- 방법: 원본 데이터(크기 N)에서 복원 추출을 N 번 수행하여 '부트스트랩 샘플'을 만든다. 이 과정을 S 번 반복한다.
- 결과: S 개의 모델과 S 개의 $\hat{\beta}$ 분포를 얻는다.

▣ 핵심 정보

2. 신뢰구간 (CI) vs. 예측구간 (PI)

- 신뢰구간 (CI): 평균 응답 $\hat{f}(x)$ 의 불확실성.
 - (오차 원인: $\hat{\beta}$ 의 불확실성)
- 예측구간 (PI): 새로운 관측치 y 의 불확실성.
 - (오차 원인: $\hat{\beta}$ 의 불확실성 + 축소 불가능한 오차 ϵ)
 - 항상 PI가 CI보다 넓다.

3. t -test 와 p-value (유의성 검증)

- 질문: 이 변수가 y 에 미치는 영향이 우연인가, 실제인가?
- 귀무가설 (H_0): 우연이다 ($\beta = 0$).
- t -test 통계량: $\frac{\hat{\beta}}{\sigma_{\hat{\beta}}}$ (신호 대 잡음비). 이 값이 0에서 얼마나 먼가?
- p-value: H_0 가 맞다고 가정할 때, 우리가 관찰한 t -test 값(혹은 더 극단적인 값)이 우연히 나올 확률.
- 결론: $p < 0.05$ 이면, ”우연히 일어날 확률이 5% 미만이므로 H_0 는 틀렸을 것이다. 이 변수는 유의미하다.”

December 10, 2025

- 강의명: CS109A: 데이터 과학 입문
- 주차: Lecture 09
- 교수명: Pavlos Protopapas, Kevin Rader, Chris Gumb
- 목적: Lecture 09의 핵심 개념 학습

Contents

I 개요	2
II 핵심 용어 정리	2
III 데이터 분석 예시: 주택 가격 예측	4
1 문제 정의 및 데이터 탐색 (EDA)	4
1.1 데이터 전처리 (Cleaning)	4
2 데이터 시각화 및 주요 발견	5
2.1 발견 1: 이분산성 (Heteroscedasticity)	5
2.2 발견 2: 다중공선성 (Collinearity)	5
2.3 모델링 라이브러리: statsmodels	6
IV 복습: 모델 검증 및 규제	7
3 교차 검증 (Cross-Validation)의 활용	7
4 데이터 표준화 (Standardization)	7
5 규제 모델 궤적도 (Trajectory Plots) 해석	7
V 확률론의 기초	9
6 확률과 확률 변수	9

7 핵심 구분: PMF vs. PDF (이산형 vs. 연속형)	9
8 주요 확률 분포	10
8.1 베르누이 분포 (Bernoulli Distribution)	10
8.2 이항 분포 (Binomial Distribution)	10
8.3 정규 분포 (Normal Distribution)와 중심극한정리 (CLT)	10
 VI 최대가능도추정 (MLE)과 선형 회귀의 연결	 12
9 추론: 확률의 역방향 문제	12
10 가능성 함수 (Likelihood Function)	12
11 로그 가능성 (Log-Likelihood)와 MLE	12
12 OLS와 MLE의 통합 (The Big Connection)	13
 VII 통계적 추론: 불확실성 정량화	 15
13 점 추정(Point Estimate)의 한계	15
14 신뢰 구간 (Confidence Interval, CI)	15
14.1 표준 오차 (Standard Error, SE)	15
15 가설 검정 (Hypothesis Testing)	15
16 부트스트래핑 vs. 공식: 최종 비교	16
 VIII 빠르게 훑어보기 (1페이지 요약)	 18

Part I

개요

강의 핵심 요약

이번 강의는 데이터 과학의 핵심인 **선형 회귀** 모델을 **확률론적 관점**에서 재해석합니다.

- **왜 확률인가?**: 우리가 가진 데이터는 더 큰 모집단(혹은 데이터 생성 프로세스)에서 나온 하나의 '무작위 실현' 일 뿐이므로, 불확실성을 다루기 위해 확률론이 필요합니다.
- **핵심 연결 고리:** 최소제곱법(OLS)을 사용하여 평균제곱오차(MSE)를 최소화하는 것은, 만약 잔차(residuals)가 정규 분포를 따른다고 가정한다면, 통계적 추론 방법인 최대가능도추정(MLE)을 수행하는 것과 수학적으로 동일함을 보입니다.
- **주요 개념:** 확률 변수, PMF/PDF, 정규 분포, 이항 분포, 가능도(Likelihood), 최대가능도추정(MLE)의 기본 원리를 학습합니다.
- **추론 방법 비교:** 모델의 불확실성을 측정하는 두 가지 방법, 즉 공식(t-검정) 기반의 신뢰구간과 부트스트래핑(Bootstrapping)을 비교합니다.
- **실전 결론:** 모델의 가정(예: 등분산성)이 깨졌을 때, 공식 기반 추론은 잘못된(지나치게 낙관적 인) 결론을 내릴 수 있으며, 가정이 적은 부트스트래핑이 더 신뢰할 수 있는 대안이 됩니다.

Part II

핵심 용어 정리

강의에서 다루는 주요 확률 및 통계 용어를 정리합니다.

용어	쉬운 설명(직관)	원어	비고(예시)
확률 변수	어떤 무작위 현상의 결과를 '숫자'로 바꿔주는 변수	Random Variable (RV)	동전 던지기(현상) \rightarrow 앞면=1, 뒷면=0 (RV)
PMF	이산 확률 변수(셀 수 있는)가 특정 값을 가질 '확률'	Probability Mass Func.	주사위 '3'이 나올 확률 = 1/6
PDF	연속 확률 변수(셀 수 없는)의 '상대적 가능성'(밀도)	Probability Density Func.	키가 170cm 일 확률은 0이지만, 170-171cm 사이 일 '확률'은 계산 가능(곡선 아래 면적)
정규 분포	자연 현상에서 가장 흔히 발견되는 종 모양(bell-shaped)의 연속 분포	Normal (Gaussian) Dist.	사람들의 키, 시험 성적 등
CLT	'많은' 샘플의 '평균'은, 원래 데이터가 어떻든 상관없이, 정규 분포를 따른다는 마법 같은 정리	Central Limit Theorem	모집단이 정규분포가 아니어도 \bar{X} 는 정규분포를 따름
이항 분포	n 번의 독립 시도에서 k 번 성공할 확률(이산 분포)	Binomial Distribution	동전을 10번 던져 앞면이 3번 나올 확률
가능도	'데이터가 관찰된 지금', 어떤 모델(파라미터)이 가장 그럴듯한지에 대한 '믿음의 정도'	Likelihood	$P(\text{Data} \theta)$ 가 아닌 $L(\theta \text{Data})$ 로 관점을 바꾼 것
MLE	가능도를 '최대화'하는 파라미터를 찾는 추정 방법. "이 데이터가 나올 확률이 가장 높은 모델은 무엇인가?"	Max. Likelihood Est.	동전을 10번 던져 앞면이 7번 나왔다면, $p = 0.7$ 을 MLE로 추정
로그 가능도	가능도 함수에 로그(log)를 씌운 것. 미분을 쉽게 하기 위해 사용. (곱셈 \rightarrow 덧셈)	Log-Likelihood	$\log(L(\theta \text{Data}))$
통계적 추론	샘플 데이터를 이용해 모집단의 특성(파라미터)에 대해 추측하는 과정	Statistical Inference	샘플 평균으로 모집단 평균을 추측
표준 오차	추정치의 불확실성(변동성)을 나타내는 값. (추정치의 표준 편차)	Standard Error (SE)	$\hat{\beta}_1$ 의 SE: "나의 기울기 추정치가 평균적으로 얼마나 벗나갈까?"
신뢰 구간	모집단 파라미터가 존재할 것이라고 '신뢰'하는 구간(예: 95%가설 검정)	H_0 (귀무가설)과 H_A (대립가설)을 세우고, 데이터로 H_0 을 기각할지 결정하는 절차	Hypothesis Testing
$H_0: \beta_1 = 0$ (효과 없음)			
p-value	귀무가설(H_0)이 '맞다'고 가정할 때, 지금 관찰된 데이터 혹은 더 극단적인 데이터가 나올 확률	p-value	p-value가 낮으면 (< 0.05), " H_0 이 맞는데 이런 일이? $\rightarrow H_0$ 을 기각하자"
다중공선성	예측 변수(X)들끼리 강한 상관관계를 보이는 현상	Collinearity	'방의 개수'와 '집 크기(sqft)'가 강하게 비례하는 경우
이분산성	오차(잔차)의 분산이 일정하지 않은 현상	Heteroscedasticity	집 크기가 클수록 가격 예측 오차의 변동 폭도 커짐

Part III

데이터 분석 예시: 주택 가격 예측

강의는 캘리포니아/서머빌 지역의 주택 가격 데이터를 분석하는 예시로 시작합니다.

1 문제 정의 및 데이터 탐색 (EDA)

- 데이터 소스: Redfin.com (온라인 부동산 사이트)
- 데이터 규모: $n = 592$ 개의 주택 판매 기록
- 질문 (목표): 어떤 변수(특성)가 주택 판매 가격과 연관되어 있는가?
- 반응 변수 (Y): price (주택 판매 가격)
- 예측 변수 (X):
 - type: 주택 유형 (콘도, 단독주택, 다가구 등) - (범주형)
 - beds: 침실 수 - (이산형)
 - baths: 욕실 수 - (이산형)
 - sqft: 면적 (제곱피트) - (연속형)
 - lotsize: 대지 크기 - (연속형)
 - year: 건축 연도 - (이산형)
 - dist: 하버드 스퀘어 T-stop(지하철역)까지의 거리 - (연속형)

1.1 데이터 전처리 (Cleaning)

분석 전, 몇 가지 데이터 정리 작업이 수행되었습니다.

1. 결측치 처리 (Missing Data):

- lotsize (대지 크기)와 hoa (주택소유자협회비) 변수에서 많은 결측치 (NA)가 발견되었습니다.
- lotsize는 주로 콘도나 타운하우스에서 결측되었고, hoa는 그 반대였습니다.
- 이는 데이터가 누락된 것이 아니라 '해당 없음'을 의미할 가능성이 높습니다.
- 따라서 이 결측치들을 0으로 대체 (Imputation) 했습니다. 이는 합리적인 가정입니다.

2. 스케일 조정 (Rescaling):

- price 변수의 단위를 '달러 (\$)'에서 '천 달러 (\$1000s)'로 변경했습니다.
- 예: \$1,250,000 → 1250.
- 이는 모델의 계수 (coefficient)를 해석하기 쉽게 만듭니다.

3. 타입 변환 (Type Conversion):

- zip (우편번호)는 숫자 (int)로 저장되어 있었지만, 실제로는 순서나 크기가 없는 범주형 데이터입니다.
- 따라서 문자열 (string) 타입으로 변환하여 모델이 이를 연속형 숫자로 오해하지 않도록 했습니다.

2 데이터 시각화 및 주요 발견

데이터 탐색(EDA)을 통해 두 가지 중요한 통계적 문제를 발견했습니다.

2.1 발견 1: 이분산성 (Heteroscedasticity)

- price (가격)와 sqft (면적)의 산점도(scatter plot)를 확인했습니다.
- 현상: 면적(sqft)이 작은 집들은 가격 변동성이 작은 반면 (즉, 예측 오차가 적음), 면적이 큰 집들은 가격 변동성이 매우 커집니다 (즉, 예측 오차가 큼).
- 정의: 이처럼 예측 변수(X)의 값에 따라 오차(잔차)의 분산이 일정하지 않은 현상을 **이분산성(Heteroscedasticity)**이라고 합니다.
- 문제점: 이는 표준적인 선형 회귀의 기본 가정('오차의 분산은 일정하다')을 위배합니다. 이 가정이 깨지면, 모델이 추정한 계수($\hat{\beta}$)는 괜찮을지 몰라도, 그 계수의 표준 오차(SE)와 신뢰 구간(CI) 계산이 부정확해집니다.

2.2 발견 2: 다중공선성 (Collinearity)

여러 예측 변수를 모두 포함한 다중 선형 회귀 모델을 피팅했습니다.

다중 회귀 모델 결과 (일부)				
	coef	std err	t	P> t
Intercept	-1949.0670	745.203	-2.615	0.009
sqft	0.6411	0.044	14.720	0.000
beds	-89.9345	23.532	-3.822	0.000
baths	198.4646	31.332	6.334	0.000
...				

- 이상한 점: beds (침실 수)의 계수가 음수(-89.9)로 나타났습니다.
- 직관적 해석: ”다른 모든 변수(면적, 욕실 수 등)를 고정한 채로, 침실 수만 1개 늘리면 집값이 약 \$89,934 하락한다.”
- 이게 말이 되나?: 상식적으로 침실이 많으면 집값이 올라야 합니다. 이것은 모델의 오류일까요?
- 원인 (다중공선성): 아닙니다. 이는 beds와 sqft 간의 강한 다중공선성(Collinearity) 때문입니다.
- 올바른 해석 (비유):
 - ’다른 모든 변수를 고정한다’는 것은, 특히 ’총 면적(sqft)을 고정한다’는 의미입니다.
 - 즉, 동일한 총 면적의 집에서 침실 수를 1개 늘린다는 것은, 기존의 방들을 조개서 더 작고 답답한 침실들을 만든다는 뜻입니다.
 - 따라서 ”집이 넓어지지 않는데 방만 얹지로 구겨 넣으면 (cramming another bedroom) 집의 가치가 떨어진다”는 합리적인 해석이 가능합니다.

2.3 모델링 라이브러리: statsmodels

- 데이터 과학에서는 `sklearn` 라이브러리를 예측에 주로 사용하지만, 통계적 추론과 해석에는 `statsmodels` 라이브러리가 더 편리할 수 있습니다.
- `statsmodels`는 R 언어처럼 공식(formula)을 사용하여 모델을 정의할 수 있게 해줍니다. (예: "price ~ sqft + type + dist")
- 이는 특히 범주형 변수(type)를 다룰 때 자동으로 더미 변수(dummy variable)를 생성해주는 등 편리함을 제공합니다.

Part IV

복습: 모델 검증 및 규제

본격적인 학률론에 들어가기 전, 이전 강의의 핵심 개념인 교차 검증과 규제를 복습합니다.

3 교차 검증 (Cross-Validation)의 활용

Q: 교차 검증(CV)은 언제 사용하나요?

교차 검증은 특정 모델에 국한된 기술이 아니라, 모델 선택(Model Selection)과 관련된 모든 의사결정에 사용할 수 있는 일반적인 도구입니다.

- A. k-NN 모델에서 최적의 k (이웃 수)를 고를 때
- B. Ridge/Lasso 모델에서 최적의 λ (규제 강도)를 고를 때
- C. 선형 회귀에서 어떤 예측 변수(feature) 조합이 최선인지 고를 때
- D. 서로 다른 모델 계열(예: k-NN vs. 선형 회귀) 중 어느 것이 더 나은지 비교할 때

정답: A, B, C, D 모두. 이 모든 것은 '모델을 선택'하는 과정이며, CV는 이 선택의 성능을 평가하는 표준 방법입니다.

4 데이터 표준화 (Standardization)

- Q: 예측 변수(X)들을 표준화(Standardize)해야 하는 경우는 언제인가요?
- A: (D) "예측 변수들을 동등하게 처리(treat equally)하고 싶을 때"입니다.
- 이유:
 - k-NN (거리 기반): 표준화를 하지 않으면, '집값'(단위: \$1000s) 같은 큰 스케일의 변수가 '방 개수' (단위: 1) 같은 작은 스케일의 변수보다 거리에 훨씬 큰 영향을 미칩니다. 즉, 스케일이 큰 변수에 편향됩니다.
 - Ridge/Lasso (규제 기반): 규제는 계수(β)의 '크기'에 폐널티를 줍니다. 만약 sqft가 피트(ft)가 아닌 인치(inch) 단위라면, 스케일이 커져서 계수(β) 값은 0에 매우 가까워질 것입니다. 스케일이 다르면 폐널티가 불공평하게 적용되므로, 표준화가 권장됩니다.
- 주의: 표준화는 '항상(Always)' 정답은 아닙니다. 예를 들어, 범주형 변수를 더미(0/1)로 만들었을 때, 이 변수들을 다른 연속형 변수와 동일한 스케일로 맞추는 것이 오히려 모델 해석을 어렵게 하거나 성능을 저하시킬 수도 있습니다.

5 규제 모델 궤적도 (Trajectory Plots) 해석

Lasso와 Ridge 모델에서 규제 강도(λ 또는 α)를 변화시킬 때, 각 변수의 회귀 계수(β)가 어떻게 변하는지 그린 그래프입니다.

Lasso vs. Ridge 쿼적도

- **Lasso (라쏘)**: λ 가 커지면 계수가 정확히 0이 됩니다. 0이 된 변수는 모델에서 '탈락'한 것 이므로, 특성 선택(Feature Selection)에 유용합니다.
 - **Ridge (릿지)**: λ 가 커져도 계수가 0에 가까워질 뿐, 정확히 0이 되지는 않습니다. 모든 변수를 유지하되 영향력을 줄입니다.
-
- **이상적인 쿼적도**: 교과서에 나오는 쿼적도는 매우 매끄러운 곡선을 그리며 0으로 수렴합니다. 이는 모든 예측 변수(X)들이 서로 독립(independent)이라고 가정한, 비현실적인 상황입니다.
 - **현실적인 쿼적도**: 실제 데이터에서는 다중공선성(Collinearity) 때문에 쿼적도가 매우 지저분합니다.
 - **다중공선성의 징후**:
 - 특정 계수가 0으로 수렴하다가 갑자기 부호가 바뀌거나 (0을 통과함)
 - 0으로 수축하는 대신 오히려 일시적으로 값이 커졌다가 줄어드는 경우
 - 이는 λ 가 커짐에 따라 한 변수(A)가 페널티를 받아 영향력이 줄어들 때, 그와 상관관계가 높은 다른 변수(B)가 A의 예측력(power)을 '넘겨받아' 계수가 커지는 현상을 나타냅니다.

Part V

확률론의 기초

데이터 과학 모델의 근간이 되는 확률 이론의 기본 개념들을 복습합니다.

6 확률과 확률 변수

- 확률(Probability)이란?
 - 정의: 어떤 사건이 발생할 장기적인 상태 빈도(long-run, relative frequency).
 - 범위: 0 (절대 발생 안 함)에서 1 (항상 발생) 사이의 값을 가집니다.
- 왜 데이터 과학에서 확률이 중요한가?
 - 우리가 가진 데이터(샘플)는 더 큰 데이터 생성 프로세스(Data Generating Process) 또는 모집단에서 나온 하나의 무작위적인 실현(random realization)에 불과합니다.
 - 확률론은 이 '불확실성'을 수학적으로 다루고, 샘플을 넘어선 모집단에 대한 추론을 가능하게 하는 언어입니다.
- 확률 변수(Random Variable, RV)란?
 - 무작위적인 현상(phenomenon)의 결과를 숫자(numeric outcome)로 대응시키는 함수(또는 변수)입니다.
 - 예시: "하버드 학생이 Mac 노트북을 사용하는가?"라는 현상을 조사할 때
 - $X_1 =$ 첫 번째 학생의 응답
 - $X_1 = 1$ (Mac 사용자), $X_1 = 0$ (그 외 사용자) $\rightarrow X_1$ 은 확률 변수입니다.

7 핵심 구분: PMF vs. PDF (이산형 vs. 연속형)

확률 변수는 크게 두 종류로 나뉘며, 이에 따라 확률을 기술하는 방식이 달라집니다.

이산형 확률 변수 (Discrete RV)

- 정의: 변수가 가질 수 있는 값이 '셀 수 있는' 경우 (예: 0, 1, 2, ... 또는 정수 값).
- 확률 함수: 확률 질량 함수 (Probability Mass Function, PMF).
- 특징: $P(X = x)$ 값이 특정 '확률'을 가집니다. 막대 그래프(bar chart)로 시각화됩니다.
- 예시: 동전 던지기 (0, 1), 주사위 굴리기 (1, 2, 3, 4, 5, 6), 침실 수 (1, 2, 3, ...).

연속형 확률 변수 (Continuous RV)

- 정의: 변수가 특정 범위 내의 '모든 실수' 값을 가질 수 있는 경우.
- 확률 함수: 확률 밀도 함수 (Probability Density Function, PDF).
- 특징:
 - $f(x)$ 값(곡선의 높이)은 확률이 아니라 '밀도(density)' 또는 '상대적 가능성'입니다.
 - 특정 값에서의 확률은 0입니다. (예: $P(\text{키} = 175.000\ldots\text{cm}) = 0$).
 - 확률은 항상 '구간'으로 계산되며, 이는 PDF 곡선 아래의 면적(Area)과 같습니다.
- 예시: 사람의 키, 주택 가격(price), 지하철역까지의 거리(dist).

8 주요 확률 분포

8.1 베르누이 분포 (Bernoulli Distribution)

- 정의: '단 한 번'의 시도($n = 1$)에서 '성공(1)' 또는 '실패(0)' 두 가지 결과만 나오는 분포. (가장 단순한 이산 분포)
- 파라미터: p (성공할 확률)
- 예시: 동전 1번 던지기 (앞면=1, 뒷면=0), 학생 1명이 Mac 유저(1)인지 아닌지(0).
- PMF: $P(X = x) = p^x(1 - p)^{1-x}$, (단, $x \in \{0, 1\}$)

8.2 이항 분포 (Binomial Distribution)

- 정의: '서로 독립'인 베르누이 시도를 n 번 반복했을 때, '성공 횟수(x)'를 나타내는 이산 분포.
- 파라미터: n (총 시도 횟수), p (각 시도의 성공 확률)
- 예시: 동전을 10번($n = 10$) 던졌을 때, 앞면이 3번($x = 3$) 나올 확률 (단, $p = 0.5$).
- PMF: $P(X = x) = \binom{n}{x} p^x (1 - p)^{n-x}$
- $\binom{n}{x}$ (n choose x)의 의미: " n 번의 시도 중 성공이 x 번 발생하는 모든 경우의 수"를 의미합니다. (예: HHH...TTT, HTHTH... 등). 이항 계수(Binomial Coefficient)라고 부릅니다.

8.3 정규 분포 (Normal Distribution)와 중심극한정리 (CLT)

- 정의: 자연계와 사회 현상에서 가장 흔하게 발견되는 종 모양(bell-shaped)의 연속 분포. 가우시안 분포(Gaussian Distribution)라고도 합니다.
- 파라미터: μ (평균, 분포의 중심), σ^2 (분산, 분포의 퍼짐 정도)
- 표준 정규 분포 (Standard Normal): 평균이 0이고 표준편차가 1인 ($Z \sim N(0, 1)$) 특별한 정규 분포.
- 표준화 (Standardization): 어떤 정규 분포 $X \sim N(\mu, \sigma^2)$ 라도, $Z = \frac{X-\mu}{\sigma}$ 공식을 통해 표준 정규 분포로 변환할 수 있습니다. Z 값은 "평균에서 몇 표준편차만큼 떨어져 있는가?"를 의미합니다.

핵심 원리: 중심극한정리 (Central Limit Theorem)**Q: 왜 정규 분포가 이렇게 중요한가요?****A: 중심극한정리(CLT) 때문입니다.**

CLT란? ”모집단이 어떤 분포(정규분포가 아니어도 됨)를 따르든 상관없이, 거기서 뽑은 샘플의 크기 n 이 충분히 크다면, 그 샘플들의 평균(\bar{X})이 이루는 분포는 정규 분포에 근사한다.”

- **직관적 예시 (신장):** 한 사람의 키(Height)는 수많은 작은 요인들(유전, 영양, 수면, ...)의 '합' 또는 '평균'으로 결정됩니다. CLT에 의해, 이렇게 많은 요인들이 합쳐진 결과물은 정규 분포를 따르게 됩니다.

- **수학적 표현:** $X_i \sim (\mu, \sigma^2)$ 일 때, $\bar{X} \sim N(\mu, \frac{\sigma^2}{n})$

- **중요한 함의:**

- 샘플 평균(\bar{X})의 평균은 모집단 평균(μ)과 같습니다.
- 샘플 평균(\bar{X})의 분산은 n 이 커질수록 작아집니다 (n 으로 나눔).
- (직관: 샘플 1개를 뽑는 것보다, 100개를 뽑아 평균내는 것이 훨씬 더 안정적이고 실제 평균에 가깝습니다.)

Part VI

최대가능도추정 (MLE)과 선형 회귀의 연결

이번 강의의 가장 핵심적인 부분으로, OLS 회귀 모델이 어떻게 확률론적 MLE와 연결되는지 설명합니다.

9 추론: 확률의 역방향 문제

확률과 통계적 추론은 동전의 양면과 같습니다.

확률 (Probability) vs. 추론 (Inference)

- 확률 (Deduction): 모델(파라미터) → 데이터
 - 질문: ”공정한 동전($p = 0.5$)이 주어졌을 때, 10번 던져 앞면이 8번 나올 확률($P(\text{Data}|\text{Model})$)은 얼마인가?”
 - 방향: 원인 → 결과
- 통계적 추론 (Inference): 데이터 → 모델(파라미터)
 - 질문: ”동전을 10번 던져 앞면이 8번 나왔다(Data). 이 동전은 공정한가($p = 0.5$), 아니면 편향되었는가($p = 0.8$)? 어떤 모델이 이 데이터를 가장 잘 설명하는가?”
 - 방향: 결과 → 원인 (우리가 데이터 과학에서 하는 일!)

10 가능성 함수 (Likelihood Function)

- 정의: 가능성 함수 $L(\theta|\text{data})$ 는 PMF 또는 PDF와 수학적으로는 동일한 함수입니다.
- 관점의 차이:
 - PDF/PMF, $f(\text{data}|\theta)$: θ (파라미터)를 고정하고 data를 변수로 봅니다.
 - Likelihood, $L(\theta|\text{data})$: data를 (우리가 관찰한) 고정된 값으로 보고, θ (파라미터)를 변수로 봅니다.
- 의미: 이 함수는 ”관찰된 데이터 하에서, 이 파라미터 θ 가 얼마나 그럴듯한지(likely)”를 측정합니다.
- 독립 가정: 만약 n 개의 데이터가 모두 독립적으로(i.i.d.) 샘플링되었다면, 전체의 가능성은 각 데이터 포인트의 가능성의 곱(product)으로 표현됩니다.

$$L(\theta|x_1, \dots, x_n) = \prod_{i=1}^n L(\theta|x_i) = \prod_{i=1}^n f(x_i|\theta)$$

11 로그 가능성 (Log-Likelihood)과 MLE

- 문제점: 수많은 확률 값을 ’곱하는’ 것은 수학적으로 매우 다루기 어렵습니다. (값이 0에 가깝게 작아지거나, 미분이 복잡해짐)
- 해결책: 로그 가능성 (Log-Likelihood) $l(\theta) = \log(L(\theta))$ 를 사용합니다.
- 이유:
 1. log 함수는 단조 증가(monotonic) 함수이므로, $L(\theta)$ 를 최대화하는 θ 값은 $l(\theta)$ 를 최대화하는 θ 값과

동일합니다.

2. 로그의 성질 ($\log(A \times B) = \log(A) + \log(B)$) 덕분에, 거대한 곱셈(Product)이 간단한 덧셈(Sum)으로 바뀝니다.

$$l(\theta|\text{data}) = \log \left(\prod_{i=1}^n f(x_i|\theta) \right) = \sum_{i=1}^n \log(f(x_i|\theta))$$

- 최대가능도추정 (Maximum Likelihood Estimator, MLE):
 - 정의: 이 로그 가능도 함수 $l(\theta)$ 를 최대화하는 파라미터 $\hat{\theta}$ 를 찾는 방법입니다.
 - 직관: ”내가 가진 데이터를 만들어 냈을 가능성인 가장 높은 파라미터(모델)를 찾겠다!”
 - 예시: 데이터 [3, 5, 10]이 $N(\mu, \sigma^2 = 4)$ 에서 나왔다고 가정할 때, 이 데이터의 로그 가능도 함수는 $\mu = 6$ 일 때 최대가 됩니다. 따라서 MLE $\hat{\mu} = 6$ 이며, 이는 샘플 평균(\bar{x})과 같습니다.
- MLE를 찾는 방법:
 1. 분석적 방법 (Analytical): $l(\theta)$ 를 θ 에 대해 미분하고, 그 값을 0으로 만드는 θ 를 찾습니다. (수학적으로 해가 구해지는 경우)
 2. 수치적 방법 (Numerical): 해가 복잡할 때 컴퓨터를 사용합니다. 경사하강법(Gradient Descent)을 이용해 ’음의’ 로그 가능도 (Negative Log-Likelihood)를 최소화(minimize)합니다. (최대화 → 음수 최소화)

12 OLS와 MLE의 통합 (The Big Connection)

이제, 왜 우리가 선형 회귀에서 손실 함수로 평균제곱오차(MSE)를 사용했는지에 대한 확률론적 정당성을 찾게 됩니다.

핵심 결론: OLS는 MLE의 특별한 경우이다

가정: 선형 회귀 모델 $Y_i = \beta_0 + \beta_1 X_i + \epsilon_i$ 에서, 잔차(error term) ϵ_i 가 서로 독립이며 평균이 0인 정규 분포 $\epsilon_i \sim N(0, \sigma^2)$ 를 따른다고 가정하자.

1. 모델 재정의: 위 가정은 Y_i 자체가 X_i 에 조건부로 정규 분포를 따른다는 의미입니다.

$$Y_i | X_i \sim N(\text{mean} = \beta_0 + \beta_1 X_i, \text{variance} = \sigma^2)$$

2. 가능도 함수 작성: n 개의 모든 데이터에 대한 (로그) 가능도 함수를 작성합니다. 정규 분포 PDF 공식을 사용하고 log를 써워 덧셈으로 만듭니다.

$$l(\beta_0, \beta_1, \sigma^2 | \text{Data}) = \sum_{i=1}^n \log \left(\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2} \left(\frac{Y_i - (\beta_0 + \beta_1 X_i)}{\sigma} \right)^2} \right)$$

3. 함수 정리: 로그를 풀면 두 부분으로 나뉩니다.

$$l(\dots) = \underbrace{\sum_{i=1}^n \log \left(\frac{1}{\sqrt{2\pi\sigma^2}} \right)}_{(A) \beta \text{와 무관한 상수항}} - \underbrace{\frac{1}{2\sigma^2} \sum_{i=1}^n (Y_i - (\beta_0 + \beta_1 X_i))^2}_{(B) \beta \text{가 포함된 항}}$$

4. MLE 목표: 우리의 목표는 이 $l(\dots)$ 함수를 최대화하는 β_0 와 β_1 를 찾는 것입니다 (MLE).

5. 결론:

- (A) 부분은 β_0, β_1 와 아무 상관이 없으므로 무시할 수 있습니다.
- $l(\dots)$ 를 최대화하려면, (B) 부분을 최대화해야 합니다.
- (B) 앞에는 음수(-) 부호가 붙어 있으므로, (B) 를 최대화하는 것은 괄호 안의 $\sum(\dots)^2$ 부분을 최소화하는 것과 같습니다.
- $\sum_{i=1}^n (Y_i - (\beta_0 + \beta_1 X_i))^2 \leftarrow$ 이것이 바로 오차제곱합 (Sum of Squared Errors, SSE)입니다!

최종 요약: '잔차가 정규 분포를 따른다'고 가정한 상태에서 MLE를 찾는 것은, 수학적으로 OLS(최소제곱법)를 사용하여 SSE(혹은 MSE)를 최소화하는 것과 정확히 일치합니다.

이는 우리가 왜 손실 함수로 MSE를 사용하는지에 대한 강력한 이론적 근거를 제공합니다.

Part VII

통계적 추론: 불확실성 정량화

모델이 $\hat{\beta}_1 = 0.5898$ 이라는 '하나의 값'(점 추정치)을 주었지만, 이 추정치가 얼마나 신뢰할 수 있는지(불확실성)를 알아야 합니다.

13 점 추정(Point Estimate)의 한계

- 우리가 얻은 $\hat{\beta}_1 = 0.5898$ 은 $n = 592$ 개의 '샘플'로 계산한 값입니다.
- 만약 우리가 다른 592개의 샘플을 뽑았다면, $\hat{\beta}_1$ 값은 0.59나 0.57처럼 약간 다른 값이 나왔을 것입니다.
- 즉, 우리의 추정치($\hat{\beta}$) 자체도 '불확실성'을 가집니다.
- 통계적 추론의 목표는 이 불확실성을 정량화하는 것입니다. (예: "진짜 β_1 이 0.6 일 가능성은?", "0 일 가능성은?")
- 우리는 두 가지 방법(신뢰 구간, 가설 검정)을 사용합니다.

14 신뢰 구간(Confidence Interval, CI)

- 목표:** "진짜 β_1 (모집단의 기울기)이 존재할 것이라고 95% 신뢰하는 구간을 제공하자."
- 방법 1 (부트스트래핑, Bootstrapping):** (이전 강의에서 배움)
 - 원본 데이터(592개)에서 중복을 허용하여 592개를 다시 뽑습니다 (Resampling).
 - 모델을 다시 피팅하여 $\hat{\beta}_1^*$ (부트스트랩 추정치)를 기록합니다.
 - 이 과정을 1000번 반복하여 $\hat{\beta}_1^*$ 의 분포를 만듭니다.
 - 이 분포의 하위 2.5%와 상위 97.5% 지점을 찾아 95% 신뢰 구간으로 삼습니다.
- 방법 2 (공식 기반, Formula-based):** (이번 강의)
 - 확률 이론(CLT, t-분포)을 바탕으로 신뢰 구간을 계산하는 공식을 사용합니다.
 - 신뢰 구간 공식: 추정치 \pm (임계값) \times (표준 오차)
 - $$\hat{\beta}_1 \pm t^* \cdot \hat{SE}(\hat{\beta}_1)$$

14.1 표준 오차(Standard Error, SE)

- 정의:** 표준 오차($SE(\hat{\beta}_1)$)는 추정치($\hat{\beta}_1$)의 표준 편차입니다.
- 직관:** "우리의 추정치가 평균적으로 얼마나 부정확한가?" (불확실성의 크기)
- 선형 회귀의 가정(정규성, 등분산성 등)이 모두 맞는다면, $\hat{SE}(\hat{\beta}_1)$ 를 계산하는 수학 공식이 존재합니다.

15 가설 검정(Hypothesis Testing)

- 목표:** "특정 가설(예: 'sqft는 가격에 영향이 없다')이 맞는지 틀리는지 데이터로 검증하자."

- 단계 (t-검정 예시):

1. 가설 설정:

- H_0 (귀무가설): $\beta_1 = 0$ (sqft는 가격과 관계가 없다. 기울기는 0이다.)
- H_A (대립가설): $\beta_1 \neq 0$ (sqft는 가격과 관계가 있다. 기울기는 0이 아니다.)

2. 검정 통계량 (Test Statistic) 계산:

$$t = \frac{\hat{\beta}_1 - 0}{\hat{SE}(\hat{\beta}_1)}$$

- 직관: ”우리가 관찰한 기울기($\hat{\beta}_1$)가, 0으로부터 몇 표준 오차(SE)만큼 떨어져 있는가?”
- t 값이 크다는 것은 $\hat{\beta}_1$ 이 0과 매우 멀리 떨어져 있다는 뜻이며, 이는 H_0 에 불리한 증거입니다.

3. p-value 계산:

- ” H_0 이 정말 맞다($\beta_1 = 0$)고 가정할 때, 지금 우리가 관찰한 t 값만큼 (혹은 더) 극단적인 t 값이 우연히 나올 확률은 얼마인가?”

4. 결정:

- p-value가 매우 낮으면 (관례적으로 < 0.05), ”우연이라고 보기엔 너무 드문 일이다. H_0 이 틀린 것 같다.” $\rightarrow H_0$ 을 기각(Reject)합니다.
- ”sqft는 가격에 통계적으로 유의미한(statistically significant) 영향을 미친다”고 결론 내립니다.

16 부트스트래핑 vs. 공식: 최종 비교

주택 가격 예시(price ~ sqft)로 돌아가, 두 가지 방법으로 계산된 β_1 (sqft의 계수)의 95% 신뢰 구간을 비교합니다.

가정 위반의 결과: 부트스트래핑의 승리

- 방법 1 (부트스트래핑 결과):
 - CI: [0.487, 0.705]
 - 구간의 너비: $0.705 - 0.487 = 0.218$
- 방법 2 (공식 기반 `statsmodels` 결과):
 - CI: [0.544, 0.636]
 - 구간의 너비: $0.636 - 0.544 = 0.092$

결과 분석:

1. 공식 기반의 신뢰 구간이 부트스트래핑 신뢰 구간보다 훨씬 좁습니다(narrower).
2. 이는 공식 기반 방법이 ”우리의 추정치가 매우 정확하다”고 지나치게 낙관적인(overly optimistic) 결론을 내렸음을 의미합니다.
3. 왜 이런 일이 발생했는가?
 - 공식 기반 방법은 선형 회귀의 모든 가정이 완벽하게 충족될 때만 정확합니다.
 - 하지만 우리는 EDA 과정에서 이분산성(Heteroscedasticity)을 발견했습니다. 이는 ’오차의 분산이 일정하다’는 핵심 가정을 위반한 것입니다.
 - 가정이 깨졌기 때문에, 공식을 통해 계산된 표준 오차(\hat{SE}) 값이 잘못(너무 작게) 계산된 것입니다.
4. 최종 결론:
 - 부트스트래핑은 데이터의 실제 분포(이분산성 포함)를 그대로 사용하여 추정치의 불확실성을 계산합니다.
 - 따라서 회귀 모델의 가정이 위반되었을 때, 부트스트래핑이 공식 기반 방법보다 더 정직하고 신뢰할 수 있는(reliable) 불확실성 추정치를 제공합니다.

Part VIII

빠르게 훑어보기 (1페이지 요약)

1. 핵심 연결: OLS

최소제곱법(OLS)을 사용하여 MSE(오차제곱합)를 최소화하는 것은, 만약 ”잔차가 정규 분포를 따른다”고 가정한다면, 확률론적인 최대가능도추정(MLE)을 수행하는 것과 수학적으로 완벽하게 동일합니다.

이것이 우리가 회귀 모델의 손실 함수로 MSE를 사용하는 강력한 이론적 근거입니다.

2. PMF vs. PDF (핵심 구분)

- **PMF (확률 질량 함수):** 이산형 (셀 수 있음, 예: 방 개수).
 - $P(X = 3) \rightarrow$ ”방 3개 일 확률”. 확률(Mass)을 가짐.
 - (시각화: 막대 그래프)
- **PDF (확률 밀도 함수):** 연속형 (셀 수 없음, 예: 집 면적).
 - $P(X = 1500.0) = 0 \rightarrow$ ”정확히 1500.0 sqft 일 확률은 0”.
 - 확률은 항상 구간(면적)으로 계산됨. (예: $P(1500 < X < 1501)$)
 - (시각화: 부드러운 곡선)

3. 중심극한정리 (CLT)

왜 정규 분포(종 모양)가 중요한가?

모집단이 주사위처럼 생긴 분포(Uniform)이든, 빠딱한 분포(Skewed)이든 상관없이, 거기서 뽑은 샘플의 평균(\bar{X})은 n 이 충분히 크다면 무조건 정규 분포를 따릅니다.

4. 다중공선성 (Collinearity) 함정

”침실 수(beds)의 계수가 음수(-89.9)가 나왔습니다. 오류인가요?”

아닙니다. 이는 beds와 sqft가 강하게 연관(collinear)되어 있기 때문입니다. 해석: ”다른 변수, 특히 총 면적(sqft)을 고정한 채로 침실 수만 1개 늘리면 (즉, 방을 양지로 조개면) 집 값이 \$89.9k 하락한다”는 합리적인 의미입니다.

5. 이분산성 (Heteroscedasticity)과 추론

”모델의 신뢰 구간(CI)은 어떤 방법을 믿어야 하나요?”

- 공식 기반 CI (t-검정): [0.544, 0.636] (좁음 → 지나치게 낙관적)
- 부트스트랩 CI (Resampling): [0.487, 0.705] (넓음 → 더 정직함)

결론: 우리 데이터는 ’이분산성’ (집이 클수록 오차가 커짐)을 보였고, 이는 공식 기반 방법의 가정을 위배합니다. 따라서 가정이 깨졌을 때는 부트스트래핑이 더 신뢰할 수 있는 불확실성 추정 방법입니다.

December 10, 2025

- 강의명: CS109A: 데이터 과학 입문
- 주차: Lecture 10
- 교수명: Pavlos Protopapas, Kevin Rader, Chris Gumb
- 목적: Lecture 10의 핵심 개념 학습

▣ 핵심 요약

이 문서의 핵심 요약

이 문서는 통계적 추론의 기본(신뢰구간, 가설 검정)을 복습하고, '가능도(Likelihood)' 개념을 소개합니다.

궁극적으로, 이 개념들을 조합하여 '베이즈 추론(Bayesian Inference)'의 핵심 원리를 설명합니다. 베이즈 추론의 핵심은 '데이터(증거)'를 바탕으로 '사전의 믿음(Prior)'을 '사후의 믿음(Posterior)'으로 갱신하는 것입니다.

진단 키트 예시를 통해 베이즈 정리의 직관을 배우고, 이를 통계 모델의 모수(θ) 추정에 적용하는 방법을 다룹니다.

Contents

1	핵심 용어 정리	2
2	통계적 추론 복습 (Inference Review)	3
2.1	모집단 모델 vs. 표본 모델	3
2.2	신뢰 구간 (Confidence Intervals)	3
2.2.1	공식 기반 신뢰 구간	3
2.2.2	왜 Z분포가 아닌 t-분포를 사용할까?	4
2.3	가설 검정 (Hypothesis Testing)	4
2.4	모델 가정과 추론 방법의 선택	4
3	신뢰 구간 vs. 예측 구간 (CI vs. PI)	6
4	가능도 (Likelihood)	7
4.1	정의: PDF를 뒤집어 생각하기	7
4.2	최대 가능도 추정 (Maximum Likelihood Estimation, MLE)	7

5 베이즈 정리 (Bayes' Rule)	9
5.1 기본 공식	9
5.2 베이즈 정리의 직관: 믿음의 갱신 (Belief Update)	9
6 베이즈 추론 (Bayesian Inference)	11
6.1 베이즈 추론의 핵심 공식	11
6.2 빈도주의 (Frequentist) vs. 베이지안 (Bayesian)	11
6.3 이산적 베이즈 추론 예시: 3개의 동전	12
6.4 미리보기: 연속적인 모수 (Normal-Normal 모델)	12
7 참고: 'statsmodels' 결과표 분석	14
8 학습 체크리스트	16
9 FAQ (자주 묻는 질문)	16
10 한눈에 훑어보기 (1-Page Summary)	18
11 부록: 몬티 홀 문제 (Monty Hall Problem)	19

1 핵심 용어 정리

본격적인 학습에 앞서, 이번 강의에서 사용되는 핵심 용어들을 정리합니다.

핵심 용어집 (Glossary)

용어 (한글)	쉬운 설명	용어 (영어)	기호/비고
모수	우리가 알고 싶은 모집단의 실제 값 (e.g., 평균, 회귀계수)	Parameter	θ, β_1, μ
추정량	샘플 데이터를 이용해 계산한 '모수 추정치' (e.g., 표본평균)	Estimate / Estimator	$\hat{\theta}, \hat{\beta}_1, \bar{X}$
표준 오차	추정량이 얼마나 부정확한지(퍼져 있는지) 나타내는 척도	Standard Error (SE)	$\hat{SE}(\hat{\beta}_1)$
신뢰 구간	모수가 포함될 것이라 '신뢰'하는 구간(점 추정 + 불확실성)	Confidence Interval (CI)	e.g., 95% CI
예측 구간	'새로운 단일 관측치'가 존재할 것이라 '예측'하는 구간	Prediction Interval (PI)	항상 신뢰구간보다 넓음
귀무 가설	우리가 기각하고 싶은 가설 (e.g., "관계가 없다", "차이가 없다")	Null Hypothesis	$H_0 : \beta_1 = 0$
p-값	귀무가설이 맞다고 가정할 때, 현재 데이터를 얻을 확률	p-value	$p\text{-value} < 0.05$
가능도	관찰된 데이터를 기반으로, 특정 모수가 얼마나 '그럴듯한지'	Likelihood	$L(\theta \text{data})$
최대가능도추정	가능도를 최대화하는 모수 θ 를 찾는 방법	Max Likelihood Est. (MLE)	
사전 확률	데이터를 보기 전, 모수에 대해 갖는 초기 믿음(확률)	Prior Probability	$P(\theta), f(\theta)$
사후 확률	데이터를 본 후, 개신된 모수에 대한 믿음(확률)	Posterior Probability	$P(\theta \text{data}), f(\theta X)$
민감도	실제 '참'인 것을 '참'이라고 예측할 확률 (True Positive Rate)	Sensitivity	$P(T+ D+)$
특이도	실제 '거짓'인 것을 '거짓'이라고 예측할 확률 (True Negative Rate)	Specificity	$P(T- D-)$

Table 1: 제 10강의 핵심 통계 용어 정리

2 통계적 추론 복습 (Inference Review)

데이터 분석은 샘플(표본)을 통해 모집단의 특성을 알아내는 '추론' 과정입니다. 단순히 모델을 만드는 것을 넘어, 이 모델이 얼마나 신뢰할 수 있는지 평가해야 합니다.

2.1 모집단 모델 vs. 표본 모델

우리가 관심 있는 것은 모집단의 '진짜' 관계(Population Model)이지만, 우리가 가진 것은 샘플 데이터로 '추정'한 모델(Estimated Model)뿐입니다.

- **모집단 모델 (이론):** $Y_i = \beta_0 + \beta_1 X_i + \epsilon_i$
 - 우리가 찾고 싶은 진짜 값 β_0, β_1
 - ϵ_i : 측정 불가능한 오차 ($N(0, \sigma^2)$ 가정)
- **표본 모델 (추정):** $\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i$ (e.g., $\hat{y} = 247.4 + 0.5898 \cdot x$)
 - 우리가 가진 데이터로 계산한 추정치 $\hat{\beta}_0, \hat{\beta}_1$
 - 이 값은 샘플이 달라지면 계속 바뀌는 '하나의 추측'에 불과합니다.

2.2 신뢰 구간 (Confidence Intervals)

점 추정(Point Estimate) $\hat{\beta}_1 = 0.5898$ 은 하나의 추측일 뿐, 진짜 β_1 이 0.6일지, 0.7일지, 아니면 0일지 알 수 없습니다. 이 '불확실성'을 표현하는 방법이 신뢰 구간(CI)입니다.

신뢰 구간을 만드는 방법은 크게 두 가지가 있습니다.

1. **부트스트랩 (Bootstrap):** 데이터를 반복 재추출(resampling)하여 $\hat{\beta}_1$ 의 경험적 분포를 만들 (컴퓨터 파워)
2. **공식 기반 (Formula-based):** 확률 이론(정규분포, t-분포)에 기반한 공식을 사용 (수학)

2.2.1 공식 기반 신뢰 구간

신뢰 구간은 추정치에서 표준 오차(Standard Error, SE)의 배수만큼 더하고 빼서 만듭니다.

$$95\% \text{ 신뢰 구간} = \hat{\beta}_1 \pm t^* \cdot \hat{SE}(\hat{\beta}_1)$$

- $\hat{SE}(\hat{\beta}_1)$: 추정치 $\hat{\beta}_1$ 의 불확실성을 나타내는 척도.
- t^* : 신뢰 수준(e.g., 95%)을 결정하는 값. (샘플이 충분히 크면 약 $1.96 \approx 2$)

□ 예제: title

추정치의 불확실성(SE)을 어떻게 줄일 수 있을까요? $SE(\hat{\beta}_1) \approx \frac{\hat{\sigma}_\epsilon}{\sqrt{\sum(x_i - \bar{x})^2}}$ 공식에서 힌트를 얻을 수 있습니다.

- 1. **데이터(n) 늘리기:** 샘플 크기 n 이 커지면 분모의 $\sum(\dots)$ 항이 커져 SE 가 줄어듭니다. (가장 확실한 방법)
- 2. **X의 범위 넓히기:** X 값들이 넓게 퍼져있을수록($Var(X) \uparrow$), 분모가 커져 SE 가 줄어듭니다. (좁은 범위의 X 로 예측하면 불안정함)
- 3. **노이즈(σ_ϵ) 줄이기:** 데이터가 회귀선 주변에 빽빽하게 모여있을수록($\hat{\sigma}_\epsilon \downarrow$), 분자인 $\hat{\sigma}_\epsilon$ 이 작아져

SE 가 줄어듭니다. (더 정확한 모델)

2.2.2 왜 Z분포가 아닌 t-분포를 사용할까?

- 정규분포 (Z): 모집단의 실제 노이즈(σ_ϵ)를 알고 있을 때 사용합니다.
- t-분포 (t): σ_ϵ 을 모르기 때문에 데이터의 잔차(residual)로 $\hat{\sigma}_\epsilon$ 를 추정했을 때 사용합니다.

σ_ϵ 를 추정하는 과정에서 추가적인 불확실성이 발생합니다. t-분포는 이 불확실성을 반영하기 위해 정규분포보다 '꼬리가 두꺼운(fatter tails)' 형태를 가집니다. (단, 샘플 크기 n 이 50개 이상으로 매우 커지면 t-분포는 정규분포와 거의 동일해집니다.)

2.3 가설 검정 (Hypothesis Testing)

"이 변수가 Y와 정말 관련이 있을까?" (e.g., β_1 이 0이 아닐까?) 를 통계적으로 답하는 과정입니다.

가설 검정 5단계

1. 가설 설정:

- 귀무 가설 (H_0): 관계가 없다. ($H_0 : \beta_1 = 0$)
- 대립 가설 (H_A): 관계가 있다. ($H_A : \beta_1 \neq 0$)

2. 검정 통계량 선택: t-통계량 $t = \frac{\hat{\beta}_1 - 0}{\hat{SE}(\hat{\beta}_1)}$ (추정치가 0으로부터 표준 오차의 몇 배만큼 떨어져 있는가?)

3. 데이터 수집 및 계산: 샘플 데이터로 $\hat{\beta}_1$ 과 $\hat{SE}(\hat{\beta}_1)$ 을 계산하여 t 값 확인.

4. 결정 (p-value):

- p-value란? H_0 가 맞다(즉, $\beta_1 = 0$) 고 가정할 때, 우리가 관찰한 t 값보다 더 극단적인 값이 나올 확률.
- p-value가 매우 작으면 (e.g., < 0.05), " H_0 가 맞다고 보기엔 너무 희귀한 일이 일어났군. H_0 를 기각 하자!"

5. 결론 도출: "p-value가 0.001 이므로, X 와 Y 사이에는 통계적으로 유의미한 연관성이 있다."

가설 검정을 위한 부트스트랩? → 순열 검정!

신뢰 구간(CI)을 만들 때는 부트스트랩(Bootstrap) 이 유용합니다. 하지만 가설 검정 (p-value) 에는 부트스트랩을 쓰면 안 됩니다. (Type I Error 증가 문제)

가설 검정 시에는 순열 검정(Permutation Test)을 사용해야 합니다.

- 부트스트랩 (CI용): H_0 와 무관하게 원본 데이터에서 (복원) 재추출.
- 순열 검정 (p-value용): H_0 가 맞다(X, Y 관계 없음) 고 가정하고, Y 값의 순서를 무작위로 섞은 (shuffle) 후 X 와 다시 매칭하여 β_1^* 을 계산.

2.4 모델 가정과 추론 방법의 선택

공식 기반 추론(t-분포)은 빠르고 편리하지만, 강력한 모델 가정(Assumptions) 이 필요합니다.

- Linearity (선형성): 관계는 선형이다.
- Independence (독립성): 오차(residual)들은 서로 독립이다. (가장 중요)
- Normality (정규성): 오차는 정규분포를 따른다. (n 이 크면 (e.g., >50) 덜 중요)

- Equal Variance (등분산성): 오차의 분산은 X 값에 상관없이 일정하다. (중요)

사례: 공식(statsmodels) vs. 부트스트랩

강의에서 주택 가격 데이터($n = 592$)를 분석한 결과, 잔차 그림에서 X 가 커질수록 분산이 커지는 '부채꼴 모양(fanning out)'이 나타났습니다. 이는 등분산성(E) 가정이 위반되었음을 의미합니다. 이때 두 방법으로 구한 β_1 (sqft)의 95% CI는 다음과 같습니다.

- 공식 (statsmodels): (0.544, 0.636) → 가정이 깨져서 잘못된(너무 좁은) 구간.
- 부트스트랩: (0.487, 0.705) → 등분산성 가정이 필요 없으므로, 이 상황에서 더 신뢰할 수 있는 구간.

결론: 모델 가정이 깨졌을 때는 부트스트랩 같은 비모수적(non-parametric) 방법이 더 강건(robust) 합니다.

3 신뢰 구간 vs. 예측 구간 (CI vs. PI)

통계적 추론에서 가장 혼동하기 쉬운 두 가지 '구간'이 있습니다. 강의 중 두 번째 퀴즈가 이 차이점을 정확히 짚었습니다.

- **신뢰 구간 (Confidence Interval):**

- 질문: "2860 평방피트(x)를 가진 집단의 평균 가격은 어디쯤 있을까?"
- 의미: 회귀선 자체의 불확실성. (e.g., $\hat{y} \pm t^* \cdot SE(\text{fit})$)

- **예측 구간 (Prediction Interval):**

- 질문: "2860 평방피트(x)를 가진 새로운 집 한 채의 가격은 어디쯤 있을까?"
- 의미: 회귀선의 불확실성 + 개별 데이터의 고유한 변동성(노이즈).

예측 구간(PI)은 항상 더 넓습니다

예측 구간은 두 가지 불확실성을 모두 고려해야 합니다.

1. 모델(회귀선)의 불확실성: $\hat{\beta}_0, \hat{\beta}_1$ 추정치의 불확실성. (신뢰구간이 다루는 영역)
2. 개별 데이터의 불확실성 (Irreducible Error, $\hat{\sigma}_\epsilon$): 아무리 모델이 완벽해도, 개별 주택 가격은 '진짜' 평균선 주위에 흩어져 있습니다.

따라서 예측 구간의 표준 오차는 두 불확실성을 모두 합산합니다.

$$\text{PI} = \hat{y} \pm t^* \cdot \sqrt{(SE(\text{fit}))^2 + \hat{\sigma}_\epsilon^2}$$

(강의 퀴즈의 D번 보기 $247.4 + 0.5898(2860) \pm 2\sqrt{0.023^2 + \hat{\sigma}^2}$ 와 동일한 형태)

4 가능도 (Likelihood)

베이즈 추론을 이해하기 위한 핵심 구성요소인 '가능도'에 대해 알아봅니다.

4.1 정의: PDF를 뒤집어 생각하기

지금까지 우리는 확률 밀도 함수(PDF)를 $P(\text{data}|\theta)$ 로 생각했습니다. (e.g., $\mu = 0, \sigma = 1$ 일 때, $x = 2$ 가 나올 확률은?)

가능도 (Likelihood)는 이 관점을 뒤집습니다.

- 데이터(x)를 '고정된 관찰 값'으로 봅니다.
- 모수(θ)를 '변수'로 봅니다.
- 질문: "우리가 관찰한 데이터 x 가 있을 때, 어떤 θ 값이 이 데이터를 가장 '그럴듯하게(likely)' 설명하는가?"

$$L(\theta|\text{data}) = P(\text{data}|\theta)$$

수학적 형태는 PDF와 같지만, θ 에 대한 함수라는 점이 다릅니다.

4.2 최대 가능도 추정 (Maximum Likelihood Estimation, MLE)

가능도 $L(\theta|\text{data})$ 를 최대로 만드는 θ 값을 찾는 것을 **MLE**라고 부릅니다.

- 데이터: x_1, x_2, \dots, x_n (서로 독립이라 가정)
- 가능도: $L(\theta) = \prod_{i=1}^n P(x_i|\theta)$ (모든 데이터가 동시에 관찰될 확률)

□ 예제: title

여러 확률을 곱하는 것(\prod)은 계산이 복잡하고 언더플로우(underflow) 위험이 있습니다. 대신 로그(\log)를 취하면 곱셈이 덧셈(\sum)으로 바뀝니다.

$$\log(L(\theta)) = \sum_{i=1}^n \log(P(x_i|\theta))$$

$L(\theta)$ 를 최대화하는 θ 는 $\log(L(\theta))$ 를 최대화하는 θ 와 동일합니다.

예시: $\sigma^2 = 4$ 인 정규분포에서 [3, 5, 10] (평균 6) 데이터가 관찰됨. $\log(L(\mu|\text{data}))$ 그래프는 $\mu = 6$ 일 때 정확히 최댓값을 가집니다. 즉, $\hat{\mu}_{MLE} = 6 = \bar{X}$ 입니다.

MLE를 찾는 방법

- 수학적 방법: $\log(L(\theta))$ 를 θ 에 대해 미분하여 0이 되는 지점을 찾습니다.
- 컴퓨터 방법: 경사 하강법(Gradient Descent)을 사용합니다. (단, 경사 하강법은 '최소화' 알고리즘이므로, $\log(L(\theta))$ 대신 음의 로그 가능도 (Negative Log-Likelihood) $-\log(L(\theta))$ 를 최소화합니다.)

OLS와 MLE의 중요한 관계

선형 회귀(Ordinary Least Squares, OLS)와 MLE는 깊은 관련이 있습니다.

만약 선형 회귀의 오차(ϵ_i)가 정규분포를 따른다고 가정한다면, 가능도 $L(\beta_0, \beta_1, \sigma^2 | \text{data})$ 를 계산할 수 있습니다.

이때, 음의 로그 가능도(Negative Log-Likelihood)를 최소화하는 것은 수학적으로 오차 제곱합(Sum of Squared Errors, SSE) $\sum(y_i - (\beta_0 + \beta_1 x_i))^2$ 을 최소화하는 것과 정확히 동일합니다.

결론: OLS는 '오차가 정규분포를 따른다'는 가정 하의 MLE입니다.

5 베이즈 정리 (Bayes' Rule)

베이즈 추론의 기반이 되는 베이즈 정리는 '조건부 확률'을 뒤집는 방법을 제공합니다.

5.1 기본 공식

두 사건 A, B에 대하여 B가 일어났을 때 A가 일어날 확률은 다음과 같습니다.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

□ 예제: title

어떤 수업에 STAT 전공자(A)가 40%, CS 전공자(B)가 60%이며, 둘 다 전공하는 학생(A and B)은 20%라고 가정해봅시다.

- 질문 1: STAT 전공자(A) 중에서 CS도 전공(B)할 확률은?

$$P(B|A) = \frac{P(A \text{ and } B)}{P(A)} = \frac{0.20}{0.40} = 0.50 \quad (50\%)$$

- 질문 2: CS 전공자(B) 중에서 STAT도 전공(A)할 확률은?

$$P(A|B) = \frac{P(A \text{ and } B)}{P(B)} = \frac{0.20}{0.60} = 0.333 \quad (33.3\%)$$

이처럼 $P(A|B)$ 와 $P(B|A)$ 는 전혀 다른 값입니다. 베이즈 정리는 한쪽을 알 때 다른 쪽을 계산할 수 있게 해줍니다.

5.2 베이즈 정리의 직관: 믿음의 갱신 (Belief Update)

베이즈 정리는 '증거(Evidence)'를 바탕으로 '사전의 믿음(Prior Belief)'을 '사후의 믿음(Posterior Belief)'으로 갱신하는 논리적 과정입니다.

□ 예제: title

어떤 사람이 임신했을 확률($D+$)이 30%라고 '초기에 믿고(Prior)' 있습니다. 이 사람이 사용한 키트의 정보는 다음과 같습니다.

- 민감도 (Sensitivity): $P(T+|D+) = 0.97$ (실제 임신 시, '양성'이 나올 확률)
- 특이도 (Specificity): $P(T-|D-) = 0.99$ (\rightarrow 실제 비임신 시, '양성'이 나올 확률 $P(T+|D-) = 1 - 0.99 = 0.01$)

이 사람이 검사 후 '양성($T+$)'이라는 증거(Evidence)를 얻었습니다. 이제 이 사람이 실제 임신($D+$)했을 확률 $P(D+|T+)$ 은 얼마일까요?

베이즈 정리를 사용합니다.

$$P(D+|T+) = \frac{P(T+|D+)P(D+)}{P(T+)}$$

여기서 $P(T+)$ 는 양성이 나올 모든 경우의 확률 (Law of Total Probability)입니다.

$$P(T+) = P(T+|D+)P(D+) + P(T+|D-)P(D-)$$

$$P(T+) = (0.97 \times 0.30) + (0.01 \times 0.70) = 0.291 + 0.007 = 0.298$$

따라서 사후 확률은:

$$P(D+|T+) = \frac{0.291}{0.298} \approx 0.9765 \quad (97.65\%)$$

결론: 30%였던 '믿음'이 '양성'이라는 '증거'를 통해 97.65%로 '갱신'되었습니다.

6 베이즈 추론 (Bayesian Inference)

이제 베이즈 정리를 통계적 모수(θ) 추론에 적용합니다.

6.1 베이즈 추론의 핵심 공식

진단 키트 예시의 구성요소를 통계 모델의 모수(θ)와 데이터(X)로 치환합니다.

- $D+$ (실제 상태) $\rightarrow \theta$ (우리가 모르는 실제 모수)
- $T+$ (검사 결과) $\rightarrow X$ (우리가 관찰한 데이터)

베이즈 추론 공식

$$f(\theta|X) = \frac{f(X|\theta) \cdot f(\theta)}{f(X)}$$

이 공식의 각 요소를 이해하는 것이 베이즈 추론의 전부입니다.

- $f(\theta|X)$ — 사후 분포 (Posterior Distribution)
 - 의미: 데이터를 관찰한 후에 생긴 된 θ 에 대한 믿음의 분포.
 - 결과물: 이것이 우리가 얻고자 하는 최종 결과입니다.
- $f(X|\theta)$ — 가능성도 (Likelihood)
 - 의미: 특정 모수 θ 를 가정했을 때, 현재 데이터 X 가 관찰될 확률.
 - 역할: 데이터 X 가 θ 의 어떤 값을 지지하는지 알려주는 '증거'의 역할. (앞서 배운 MLE의 그 가능성도입니다.)
- $f(\theta)$ — 사전 분포 (Prior Distribution)
 - 의미: 데이터를 관찰하기 전에 θ 에 대해 우리가 가졌던 초기 믿음.
 - 역할: 이전 연구나 상식 등, 데이터 외의 정보를 모델에 반영합니다.
- $f(X)$ — 증거 (Evidence) 또는 정규화 상수
 - 의미: $f(X) = \int f(X|\theta) f(\theta) d\theta$. (모든 θ 에 대해 가능성도를 평균낸 값)
 - 역할: 사후 분포 $f(\theta|X)$ 의 총합이 1이 되도록 맞춰주는 '상수' 역할. 계산이 복잡해서 종종 생략하고 비례 관계로 표현합니다.

더 간단한 표현: Posterior \propto Likelihood \times Prior

6.2 빈도주의 (Frequentist) vs. 베이지안 (Bayesian)

전통적인 통계(빈도주의)와 베이즈 추론은 '확률'과 '모수'를 바라보는 근본적인 관점이 다릅니다.

항목	빈도주의 (Frequentist) (e.g., OLS, MLE)	베이지안 (Bayesian)
모수(θ) 관점	고정된 상수. (단지 우리가 모를 뿐)	확률 변수. (믿음의 분포를 가짐)
확률의 정의	장기적인 실험에서 발생하는 빈도	주관적 믿음의 정도 (Belief)
핵심 질문	" θ 가 0이라는 H_0 를 기각할 수 있는가?"	"데이터 X 를 본 후, θ 에 대한 믿음은?"
결과물	점 추정치 $\hat{\theta}$ 와 신뢰구간(CI)	사후 분포 $f(\theta X)$ (전체 분포)
구간의 해석	"이 절차로 CI를 100번 만들면, 95개는 θ 를 포함한다."	" θ 가 이 구간에 있을 확률(믿음)이 95%이다."

Table 2: 빈도주의와 베이지안 추론의 관점 비교

6.3 이산적 베이즈 추론 예시: 3개의 동전

베이즈 추론이 '믿음을 갱신'하는 과정을 간단한 이산적 예시로 살펴봅니다.

□ 예제: title

주머니 속에 3개의 동전(θ)이 있습니다: $p = 0.1$ (가짜), $p = 0.5$ (공정), $p = 0.9$ (가짜). 이 중 하나를 랜덤하게 뽑아 4번($n = 4$) 던졌더니, 3번의 앞면(H)과 1번의 뒷면(T)이라는 데이터(X)가 나왔습니다.

질문: 우리는 3개의 동전 중 어떤 것을 뽑았다고 믿어야 할까요?

1. 사전 분포 (Prior) $f(\theta)$ 설정 데이터를 보기 전, 각 동전을 뽑을 확률은 공평하게 $1/3$ 입니다.

$$P(p = 0.1) = 1/3, \quad P(p = 0.5) = 1/3, \quad P(p = 0.9) = 1/3$$

2. 가능성 (Likelihood) $f(X|\theta)$ 계산 각 동전(θ)을 가정했을 때, '3H 1T' 데이터(X)가 나올 확률 (이항분포)

- $L(p = 0.1) = P(X|p = 0.1) = \binom{4}{3}(0.1)^3(0.9)^1 = 4 \times 0.001 \times 0.9 = \mathbf{0.0036}$

- $L(p = 0.5) = P(X|p = 0.5) = \binom{4}{3}(0.5)^3(0.5)^1 = 4 \times 0.125 \times 0.5 = \mathbf{0.2500}$

- $L(p = 0.9) = P(X|p = 0.9) = \binom{4}{3}(0.9)^3(0.1)^1 = 4 \times 0.729 \times 0.1 = \mathbf{0.2916}$

(데이터 X 는 $p = 0.9$ 일 가능성은 가장 그럴듯하게 봅니다.)

3. 사후 분포 (Posterior) \propto Likelihood \times Prior

- $P(p = 0.1|X) \propto 0.0036 \times (1/3) = 0.0012$

- $P(p = 0.5|X) \propto 0.2500 \times (1/3) = 0.0833$

- $P(p = 0.9|X) \propto 0.2916 \times (1/3) = 0.0972$

4. 정규화 (Normalize) \rightarrow 총합을 1로 만들기 총합 = $0.0012 + 0.0833 + 0.0972 = 0.1817$

- $P(p = 0.1|X) = 0.0012/0.1817 \approx \mathbf{0.007} \quad (0.7\%)$

- $P(p = 0.5|X) = 0.0833/0.1817 \approx \mathbf{0.458} \quad (45.8\%)$

- $P(p = 0.9|X) = 0.0972/0.1817 \approx \mathbf{0.535} \quad (53.5\%)$

결론: '3H 1T'라는 데이터를 관찰한 후, 우리의 믿음은 $(1/3, 1/3, 1/3)$ 에서 $(0.7\%, 45.8\%, 53.5\%)$ 로 갱신되었습니다. 이제 우리는 $p = 0.9$ 동전을 뽑았다고 가장 강하게 믿게 되었습니다.

6.4 미리보기: 연속적인 모수 (Normal-Normal 모델)

위 예시는 모수 p 가 3개의 값만 가지는 이산적(discrete) 경우였습니다. 실제로는 모수(μ, β_1 등)가 연속적(continuous)인 경우가 많습니다.

- 모델: $X_i \sim N(\mu, \sigma^2)$ (단, σ^2 은 있다고 가정)
- 사전 분포 (Prior): μ 에 대한 초기 믿음. (e.g., $\mu \sim N(\mu_0, \sigma_0^2)$)
- 데이터: $X = \{x_1, \dots, x_n\}$ (표본 평균 \bar{X})
- 사후 분포 (Posterior): $f(\mu|X)$ 는 놀랍게도 또 다른 정규분포가 됩니다.

이때 사후 분포의 평균(우리의 최종 μ 추정치)은 다음과 같습니다.

$$\hat{\mu}_{\text{posterior}} = \frac{(\sigma^2 \cdot \mu_0) + (n\sigma_0^2 \cdot \bar{X})}{\sigma^2 + n\sigma_0^2}$$

이 식은 사전 믿음(μ_0)과 데이터(\bar{X})의 가중 평균입니다.

데이터가 사전 믿음을 이긴다

위 가중 평균식에서 샘플 크기 n 이 매우 커지면 ($n \rightarrow \infty$) 어떻게 될까요?

- μ_0 의 가중치 $\rightarrow 0$
- \bar{X} 의 가중치 $\rightarrow 1$

결론: 데이터가 충분히 많아지면, 초기에 어떤 사전 믿음(μ_0)을 가졌든 상관없이 사후 분포는 데이터가 말해주는 값(\bar{X})으로 수렴합니다.

7 참고: ‘statsmodels’ 결과표 분석

강의에서 사용된 ‘statsmodels’ (Python 라이브러리)의 OLS 회귀분석 결과표를 해석하는 방법입니다.

단순 선형 회귀: ‘price ~ sqft’

```

1 =====
2 Dep. Variable:                  price      R-squared:
3                               0.519
4 Model:                          OLS      Adj. R-squared:
5                               0.518
6 Method: Least Squares      F-statistic:
7                               635.6
8 Date:      Tue, 03 Oct 2023   Prob (F-statistic):
9                               9.97e-96
10 Time:          22:00:05      Log-Likelihood:
11                               -4566.2
12 No. Observations:             592      AIC:
13                               9136.           BIC:
14 Df Residuals:                 590
15 Df Model:                      1
16 Covariance Type:            nonrobust
17 =====
18
19              coef      std err      t      P>|t|      [0.025
20                  0.975]
21 -----
22 Intercept    247.4382     45.388     5.452      0.000     158.296
23             336.581
24 sqft        0.5898      0.023     25.211      0.000      0.544
25             0.636
26 =====

```

Listing 1: 단순 선형 회귀(OLS) 결과

핵심 해석:

- **coef (계수):** ‘Intercept’($\hat{\beta}_0$) = 247.4, ‘sqft’($\hat{\beta}_1$) = 0.5898 (평방 피트가 1 증가할 때마다 가격이 0.5898 (천 달러) 증가)
- **std err (표준 오차):** ‘sqft’의 $\hat{SE}(\hat{\beta}_1)$ = 0.023 (0.5898이라는 추정치의 불확실성 정도)
- **t (t-통계량):** $25.211 = (0.5898 - 0) / 0.023$ ($H_0 : \beta_1 = 0$ 으로부터 25 표준오차만큼 떨어져 있음)
- **P>|t| (p-값):** 0.000 ($\beta_1 = 0$ 인데 이런 결과가 나올 확률이 0에 가까움 $\rightarrow H_0$ 기각)
- **[0.025 0.975] (95% 신뢰구간):** (0.544, 0.636) (모집단의 실제 β_1 이 0.544와 0.636 사이에 있을 것이라 95% 신뢰 함)

다중 선형 회귀: ‘price ~ sqft + dist + ... + type’

	coef	std err	t	P> t
	[0.025	0.975]		
Intercept	-1949.0670	745.203	-2.615	0.009
type[T.multifamily]	-3412.677	-485.457	-5.839	0.000
type[T.singlefamily]	-604.352	-300.119	6.145	0.000
type[T.townhouse]	335.7612	54.642	-1.344	0.179
sqft	228.441	443.081	14.720	0.000
dist	-188.111	35.237	-8.634	0.000
beds	0.556	0.6411	-3.822	0.000
baths	-213.018	-134.067	23.532	0.000
year	136.928	198.4646	6.334	0.000
	0.468	260.002	0.388	3.169
	1.992			0.002

Listing 2: 다중 선형 회귀(OLS) 결과 (일부)

핵심 해석 (다중 회귀):

- 계수 해석의 변화: ‘sqft’의 계수가 0.5898 → 0.6411로 변경되었습니다. 이는 다른 변수(dist, beds 등)의 효과를 ‘통제’했기 때문입니다.
- ‘beds’ 계수 (-89.93): ”다른 모든 변수(sqft, baths, dist 등)가 동일하게 고정되어 있다면, 침실 수가 1개 증가할 때 오히려 가격이 89.9 (천 달러) 감소한다.”
- 인과관계 주의: 이는 상관관계(association)일 뿐, 침실을 늘리면 집값이 떨어진다는 인과관계(causation)를 의미하지 않습니다. (e.g., 같은 크기의 집에 침실만 무리하게 늘리면 가치가 떨어질 수 있음)

8 학습 체크리스트

이 강의를 통해 다음 질문에 답할 수 있는지 확인해 보세요.

최소 학습 목표 (Checklist)

- 모집단 모델(β_1)과 표본 모델($\hat{\beta}_1$)의 차이를 설명할 수 있는가?
- 표준 오차(SE)가 무엇인지, 그리고 SE를 줄이는 3가지 방법을 아는가?
- 왜 Z-분포 대신 t-분포를 사용하며, 둘의 형태상 차이는 무엇인가?
- p-value가 0.05보다 작다는 것이 (e.g., $H_0 : \beta_1 = 0$ 일 때) 무엇을 의미하는지 설명할 수 있는가?
- 신뢰 구간과 예측 구간의 차이를 '질문'과 '불확실성'의 관점에서 설명할 수 있는가?
- 가능성(Likelihood)과 확률(Probability)과 어떻게 다른지, $L(\theta|X)$ 로 설명할 수 있는가?
- OLS(최소제곱법)와 MLE(최대가능도추정)의 관계는 무엇인가? (힌트: 정규분포 가정)
- 베이즈 정리를 '믿음의 갱신' 과정(사전확률 \rightarrow 증거 \rightarrow 사후확률)으로 설명할 수 있는가?
- 베이즈 추론 공식의 4가지 요소($f(\theta|X)$, $f(X|\theta)$, $f(\theta)$, $f(X)$)를 각각 명명하고 설명할 수 있는가?
- 빈도주의자와 베이지안이 '모수(θ)'를 바라보는 근본적인 관점 차이는 무엇인가?

9 FAQ (자주 묻는 질문)

초심자가 흔히 가질 수 있는 질문들을 정리했습니다.

Q: 부트스트랩 샘플 수(B)를 1000번에서 100만 번으로 늘리면 신뢰구간이 더 좁아지나요?

A: 아닙니다. 부트스트랩 샘플 수 B 를 늘리는 것은 단지 우리가 추정한 샘플링 분포($\hat{\beta}_1$ 의 히스토그램)를 더 '매끄럽게(smoothen)' 만들 뿐입니다. 이는 계산의 정밀도를 높이는 것일지, 원본 추정치의 근본적인 불확실성을 줄여주지 않습니다.

신뢰 구간을 좁히는(불확실성을 줄이는) 유일한 방법은 원본 데이터의 샘플 크기 n 을 늘리는 것입니다. (e.g., 500개 훈련 데이터 \rightarrow 5000개 훈련 데이터)

□ 예제: title

A: 좋은 지적입니다. 하지만 '근거 없는 찍기'와는 다릅니다.

1. **정보 사전확률(Informative Prior):** 과거의 연구 결과나 해당 분야의 전문가적 합의 (e.g., "회귀 계수가 0에서 10 사이일 것")를 반영합니다.
2. **무정보 사전확률(Uninformative Prior):** 정말 아는 것이 없을 때, 모든 가능성을 평평하게(uniform) 두는 사전확률을 사용합니다. (e.g., p 가 0~1 사이 모든 값의 확률이 동일)
3. **데이터의 힘:** 가장 중요한 점은, 데이터(n)가 충분히 많아지면, 초기에 어떤 사전확률을 설정했더라도 사후확률(결과)은 거의 같은 결론으로 수렴한다는 것입니다. 데이터가 주관적인 믿음을 '압도'합니다.

□ 예제: title

A: 상황에 따라 다릅니다. (It depends.)

- 빈도주의 (Frequentist): 고전적인 방법이며, 계산이 간단하고 빠릅니다. 객관적인 '절차'를 중시하는 분야(e.g., 의학 임상시험)에서 표준으로 사용됩니다.
- 베이지안 (Bayesian): 계산이 복잡하지만(최근 10+년간 MCMC 등 컴퓨터 파워로 해결), '사전 지식'을 모델에 통합할 수 있습니다. 데이터가 적거나(n 이 작을 때), 모수에 대한 불확실성을 '분포' 자체로 얻고 싶을 때 매우 강력합니다.

현대 데이터 과학에서는 두 가지 접근법을 모두 이해하고, 문제 상황에 맞게 사용하거나 두 결과를 비교 분석하는 경우가 많습니다.

10 한눈에 훑어보기 (1-Page Summary)

신뢰 구간 (Mean) vs. 예측 구간 (Single)

- 신뢰 구간 (CI): ”2860 sqft 집들의 평균 가격은?” (불확실성 1개: 모델)
- 예측 구간 (PI): ”2860 sqft 집 한 채의 가격은?” (불확실성 2개: 모델 + 개별 노이즈 $\hat{\sigma}_\epsilon$)
- → PI는 항상 CI보다 넓습니다.

OLS

선형 회귀에서 오차(ϵ)가 정규분포를 따른다고 가정하면, 오차 제곱합을 최소화(OLS)하는 것은 가능도를 최대화(MLE)하는 것과 수학적으로 동일합니다.

베이즈 정리의 핵심 흐름

사전 믿음 (Prior) ($P(D+) = 30\%$) + 증거 (Evidence) (테스트 ’양성’) \rightarrow 사후 믿음 (Posterior) ($P(D+|T+) = 97.7\%$)

베이즈 추론의 심장

$$\underbrace{f(\theta|X)}_{\text{사후 분포}} \propto \underbrace{f(X|\theta)}_{\text{가능도}} \times \underbrace{f(\theta)}_{\text{사전 분포}}$$

(Posterior \propto Likelihood \times Prior)

빈도주의 vs. 베이지안

- 빈도주의자: 모수(θ)는 고정된 값. 나의 추정치($\hat{\theta}$)가 확률적.
- 베이지안: 모수(θ)는 확률 변수(분포). 나의 믿음이 데이터에 따라 갱신됨.

11 부록: 몬티 홀 문제 (Monty Hall Problem)

강의에서 간단히 언급된 몬티 홀 문제는 조건부 확률과 베이즈 정리의 직관을 보여주는 유명한 예시입니다.

문제 상황

1. 3개의 문(Door 1, 2, 3) 뒤에 1개는 자동차(Car), 2개는 염소(Goat)가 있습니다.
2. 당신은 하나의 문(e.g., Door 1)을 선택합니다.
3. 진행자(Monty)는 당신이 선택하지 않은 두 개의 문(Door 2, 3) 중에서, 염소가 있는 문 하나를 열어서 보여줍니다. (e.g., Door 3에 염소가 있다고 열어줌)
4. 진행자가 묻습니다: ”처음 선택(Door 1)을 유지하시겠습니까, 아니면 남은 문(Door 2)으로 바꾸시겠습니까?”

결론: 무조건 바꾸는 것이 유리합니다.

- 유지(Stay) 할 경우 승률: 1/3
- 변경(Switch) 할 경우 승률: 2/3

□ 예제: title

시나리오 1: 맨 처음에 '자동차'를 선택한 경우 (확률 1/3)

- 당신: Door 1 (Car) 선택
- 진행자: Door 2(Goat) 또는 Door 3(Goat) 중 하나를 열어줌
- 당신: Door 2(Goat) 또는 Door 3(Goat)으로 바꾼다. → 패배!

시나리오 2: 맨 처음에 '염소'를 선택한 경우 (확률 2/3)

- 당신: Door 1 (Goat) 선택
- 진행자: Door 2(Car), Door 3(Goat) 중 염소가 있는 문(Door 3)을 반드시 열어야 함
- 당신: 남은 문(Door 2)으로 바꾼다. → 승리!

요약:

- 처음 선택을 '유지' 하면, 처음에 자동차를 골랐을 때(1/3)만 이깁니다.
- 처음 선택을 '변경' 하면, 처음에 염소를 골랐을 때(2/3) 항상 이깁니다.

진행자의 행동($P(\text{염소 문 열기} | \text{내 선택})$)은 새로운 정보를 제공하며, 이 정보가 조건부 확률을 변경하여 바꾸는 것의 승률을 2/3로 높여줍니다. (100개 문으로 확장하면, 99/100의 확률로 이길 수 있습니다.)

December 10, 2025

- 강의명: CS109A: 데이터 과학 입문
- 주차: Lecture 11
- 교수명: Pavlos Protopapas, Kevin Rader, Chris Gumb
- 목적: Lecture 11의 핵심 개념 학습

Contents

1 베이지안 용어 정리	1
2 베이지안 추론이란 무엇인가?	3
2.1 1. 빈도주의 (Frequentist) 관점	3
2.2 2. 베이지안 (Bayesian) 관점	3
3 베이즈 정리: 믿음을 업데이트하는 공식	5
4 사전분포(Prior) 선택하기	7
4.1 1. 정보적 사전분포 (Informative Prior)	7
4.2 2. 비정보적 사전분포 (Uninformative Prior)	7
4.3 3. 컬레 사전분포 (Conjugate Prior)	7
5 핵심 예제: 정규-정규 모델 (Normal-Normal Model)	9
6 베이지안 추정: 점과 구간	9
6.1 점 추정 (Point Estimation)	9
6.2 구간 추정 (Interval Estimation)	10
7 베이지안 선형 회귀 (Bayesian Linear Regression)	11
8 심화: 릿지(Ridge)와 라쏘(Lasso)의 베이지안 해석	12
8.1 1. 릿지(Ridge) = MAP + 정규(Normal) 사전분포	12
8.2 2. 라쏘(Lasso) = MAP + 라플라스(Laplace) 사전분포	13
9 계산 방법: MCMC와 갑스 샘플링	14
9.1 문제: 사후 분포가 너무 복잡할 때	14
9.2 해결: 시뮬레이션 (MCMC)	14
9.3 갑스 샘플링 (Gibbs Sampling)	14

10 자주 묻는 질문 (FAQ) 및 점검	16
11 빠르게 훑어보기 (1-Page Summary)	16

▣ 핵심 요약

이 문서는 베이지안(Bayesian) 모델링의 핵심 개념을 설명합니다.

베이지안 추론은 **데이터(증거)를 바탕으로 기존의 믿음(사전 확률)을 업데이트**하여, 더 합리적인 새로운 믿음(사후 확률)을 도출하는 과정입니다.

단 하나의 '정답'을 찾는 대신, **파라미터의 '가능성'을 확률 분포로 표현**합니다.

이 문서를 통해 베이즈 정리의 기본 원리를 이해하고, 이것이 어떻게 릿지(Ridge), 라쏘(Lasso)와 같은 머신러닝 회귀 모델과 연결되는지 학습합니다.

1 베이지안 용어 정리

베이지안 모델링을 처음 접할 때 가장 혼란스러운 것은 용어입니다. 핵심 용어들을 일상적인 언어로 먼저 정리합니다.

용어	쉬운 설명 (직관)	원어	비고
베이즈 정리	나의 '기존 믿음'과 '새로운 증거'를 합쳐 '최종 결론'을 내는 수학 공식	Bayes' Rule / Theorem	$P(A B) = \frac{P(B A)P(A)}{P(B)}$
사전 확률	데이터를 보기 전, 내가 이미 가지고 있던 믿음(가설)	Prior Probability	$P(A)$
가능도	나의 '믿음'이 맞다고 가정할 때, 이 '증거'가 나타날 확률	Likelihood	$P(B A)$
사후 확률	'증거'를 반영하여 새롭게 업데이트 된 '최종 믿음'(결론)	Posterior Probability	$P(A B)$
증거	그냥 '증거'가 관찰될 확률(전체 확률). 주로 정규화 상수로 쓰임.	Evidence / Marginal	$P(B)$
빈도주의	파라미터(예: 평균)는 '고정된 값'이며, 데이터가 무작위라고 보는 관점.	Frequentist	베이지안과 대비되는 통계학의 주류 관점.
베이지안	파라미터 자체를 '확률 변수'(믿음의 대상)로 보는 관점.	Bayesian	주관적 확률(belief)을 다룸.
초매개변수	'사전 확률'을 정의하기 위해 필요 한 또 다른 파라미터. (예: $\mu \sim N(\mu_0, \sigma_0^2)$ 에서 μ_0, σ_0^2)	Hyperparameter	Prior의 파라미터.
켤레 사전분포	계산을 편하게 해주는 '특수 조합'. (예: 이항-베타, 정규-정규)	Conjugate Prior	특정 가능도와 결합 시, 사후 분포가 사전 분포와 '같은 가족'이 됨.
MAP	사후 확률 분포에서 '가장 높은 지점'(최 빈값)을 추정치로 사용.	Max A Posteriori	사후 확률을 최대화하는 파라미터.
신용 구간	"파라미터가 이 구간 안에 있을 확률이 95%다." (매우 직관적)	Credible Interval	베이지안 베전의 구간 추정.
신뢰 구간	"같은 실험 100번 시, 95개의 구간이 '참 값'을 포함할 것이다."	Confidence Interval	빈도주의 베전의 구간 추정. (해석이 까다로움)

2 베이지안 추론이란 무엇인가?

통계학에는 세상을 바라보는 두 가지 큰 관점이 있습니다: 빈도주의(Frequentist)와 베이지안(Bayesian)입니다.

2.1 1. 빈도주의 (Frequentist) 관점

빈도주의는 우리가 흔히 배우는 전통적인 통계학입니다.

* 핵심 가정: 우리가 찾으려는 파라미터(모수, θ , 예: 전교생의 실제 평균 키)는 하나의 '고정된' 상수입니다.
 * 데이터: 이 '참 값'을 알기 위해 우리가 뽑는 표본(데이터, X)이 무작위(random)입니다.
 * 목표: 데이터를 많이 뽑아서 저 '고정된 참 값'(θ)을 정확히 추정(estimate)하는 것입니다.

□ 예제: 빈도주의 비유: 고정된 보물 찾기

어딘가에 하나의 '보물'(θ)이 숨겨져 있습니다.

우리는 이 보물의 위치를 모르지만, 보물에 대한 힌트(X , 데이터)를 여러 번 얻을 수 있습니다.

빈도주의는 이 힌트(데이터)들을 조합하여 "보물은 (x, y) 지점에 있을 것이다"라고 하나의 지점을 추정하는 방식입니다.

2.2 2. 베이지안 (Bayesian) 관점

베이지안 통계학은 '믿음(belief)'의 관점에서 접근합니다.

* 핵심 가정: 우리가 가진 데이터(X)는 '고정된' 관찰 값입니다. (일단 관찰했으므로)
 * 파라미터: 오히려 우리가 모르는 파라미터(θ)가 무작위(random)이며, 확률 분포를 가집니다.
 * 목표: 데이터(X)를 관찰함으로써, 파라미터 θ 에 대한 우리의 믿음(belief)을 업데이트하는 것입니다.

□ 예제: 베이지안 비유: 확률적인 보물 지도

베이지안은 보물이 "어디쯤 있을지"에 대한 '믿음의 지도'(사전 확률)로 시작합니다. (예: "A 지역 60%, B 지역 40%")

이때 힌트(X , 데이터)를 하나 얻습니다. (예: "보물은 강 근처에 있다.")

이 힌트를 바탕으로, "A 지역이 강 근처일 확률"과 "B 지역이 강 근처일 확률"을 계산하여, 원래의 믿음을 업데이트한 '새로운 지도'(사후 확률)를 만듭니다. (예: "A 지역 80%, B 지역 20%")

colback=mygray, colframe=darkgray, breakable, title=□ 빈도주의 vs. 베이지안 핵심 비교

- **빈도주의 (Frequentist):**

- 파라미터(θ): 고정된 상수 (Unknown constant)
- 데이터(X): 랜덤 변수 (Random variable)
- 해석: ”무한히 반복하면, 95%의 신뢰 구간이 참 값을 포함한다.”

- **베이지안 (Bayesian):**

- 파라미터(θ): 랜덤 변수 (Random variable)
- 데이터(X): 고정된 관찰 값 (Fixed data)
- 해석: ”우리의 데이터에 따르면, 95% 확률로 파라미터가 이 신용 구간 안에 있다.”

베이지안의 ’신용 구간’이 우리가 일상적으로 ”확률”을 이해하는 방식과 더 가깝습니다.

3 베이즈 정리: 믿음을 업데이트하는 공식

베이지안 추론은 베이즈 정리(Bayes' Rule)라는 하나의 공식에서 출발합니다.

$$P(\theta|X) = \frac{P(X|\theta)P(\theta)}{P(X)}$$

이 공식의 각 항은 다음과 같은 의미를 가집니다.

- $P(\theta|X)$: 사후 확률 (Posterior)
 - ”데이터 X 를 관찰한 후에, 파라미터 θ 에 대한 우리의 최종 믿음”
 - 이것이 우리가 구하고자 하는 결과물입니다.
- $P(X|\theta)$: 가능성 (Likelihood)
 - ”파라미터 θ 가 참이라고 가정할 때, 데이터 X 가 관찰될 가능성”
 - 이것은 우리가 가진 데이터로부터 계산되는 증거의 힘입니다.
- $P(\theta)$: 사전 확률 (Prior)
 - ”데이터 X 를 관찰하기 전에, 파라미터 θ 에 대해 우리가 가졌던 초기 믿음”
 - 이것은 우리의 주관적인 지식이나 가정입니다.
- $P(X)$: 증거 (Evidence)
 - ”그냥 데이터 X 가 관찰될 전체 확률”
 - 실제 계산에서는 θ 와 관련이 없으므로, $P(\theta|X)$ 의 합이 1이 되도록 만드는 정규화 상수(Normalizing Constant) 역할을 합니다.

따라서 베이즈 정리는 다음과 같이 요약할 수 있습니다.

▣ 핵심 요약

사후 확률 (최종 믿음) \propto 가능성 (증거의 힘) \times 사전 확률 (초기 믿음)

$$f(\theta|X) \propto f(X|\theta)f(\theta)$$

▣ 예제: 동전 뒤집기 예제 (이산 확률)

주머니 속에 3개의 동전이 있습니다:

- 동전 A: 앞면이 나올 확률 $p = 0.1$ (불량 동전)
- 동전 B: 앞면이 나올 확률 $p = 0.5$ (공정 동전)
- 동전 C: 앞면이 나올 확률 $p = 0.9$ (불량 동전)

1. **사전 확률 (Prior):** 내가 동전 하나를 무작위로 뽑았습니다. 이 동전이 A, B, C일 확률은 얼마일까요? 데이터가 없으므로, 각각 $1/3$ 입니다.

$$P(\theta = 0.1) = 1/3 \quad | \quad P(\theta = 0.5) = 1/3 \quad | \quad P(\theta = 0.9) = 1/3$$

2. **데이터 (Data):** ◎ 동전을 4번 던졌더니, 앞면이 3번, 뒷면이 1번 나왔습니다. ($X = 3$)

3. **가능도 (Likelihood):** 각 동전(가설)이 이 데이터를 만들어낼 가능성은 얼마일까요? (이항 분포 사용: $\binom{4}{3} p^3 (1-p)^1$)

- $P(X = 3|\theta = 0.1) = \binom{4}{3} (0.1)^3 (0.9)^1 = 4 \times 0.001 \times 0.9 = 0.0036$
- $P(X = 3|\theta = 0.5) = \binom{4}{3} (0.5)^3 (0.5)^1 = 4 \times 0.125 \times 0.5 = 0.2500$
- $P(X = 3|\theta = 0.9) = \binom{4}{3} (0.9)^3 (0.1)^1 = 4 \times 0.729 \times 0.1 = 0.2916$

데이터는 동전 C ($p = 0.9$) 일 가능성을 가장 높게 시사합니다.

4. 사후 확률 (Posterior): 이제 $Posterior \propto Likelihood \times Prior$ 를 계산합니다.

- $P(\theta = 0.1|X = 3) \propto 0.0036 \times (1/3) \approx 0.0012$
- $P(\theta = 0.5|X = 3) \propto 0.2500 \times (1/3) \approx 0.0833$
- $P(\theta = 0.9|X = 3) \propto 0.2916 \times (1/3) \approx 0.0972$

5. 정규화 (Normalize): 사후 확률의 총합($0.0012 + 0.0833 + 0.0972 = 0.1817$)으로 나누어 합이 1이 되게 합니다.

- $P(\theta = 0.1|X = 3) = 0.0012/0.1817 \approx \mathbf{0.7\%}$
- $P(\theta = 0.5|X = 3) = 0.0833/0.1817 \approx \mathbf{45.8\%}$
- $P(\theta = 0.9|X = 3) = 0.0972/0.1817 \approx \mathbf{53.5\%}$

결론: 데이터를 보기 전 우리의 믿음은 (33%, 33%, 33%) 였지만, "4번 중 3번 앞면"이라는 데이터를 본 후, 우리의 믿음은 (0.7%, 45.8%, 53.5%)로 업데이트되었습니다. 우리는 이제 이 동전이 $p = 0.9$ 동전(C)이라고 가장 강하게 믿게 되었습니다.

4 사전분포(Prior) 선택하기

베이지안 모델링의 가장 중요하고 주관적인 부분이 바로 사전분포(Prior, $f(\theta)$)를 정하는 것입니다. 사전분포는 ”내가 데이터를 보기 전에 파라미터에 대해 무엇을 알고 있는가?”를 확률 분포로 표현하는 것입니다. 사전분포를 선택하는 3가지 주요 접근 방식이 있습니다.

4.1 1. 정보적 사전분포 (Informative Prior)

이전 연구, 전문가의 의견, 또는 과거의 데이터를 바탕으로 ’정보가 있는’ 사전분포를 설정합니다.

* 목적: 이미 알고 있는 지식을 모델에 적극적으로 반영합니다. * 예시: * 내일 정오의 기온(μ)을 예측하는 모델을 만든다고 가정합니다. * 사전분포: ”어제 정오 기온이 20도였고, 최근 30일간 일교차 표준편차가 2도였다”는 정보를 바탕으로, * $\mu \sim N(\mu_0 = 20, \sigma_0^2 = 2^2)$ 처럼 정규분포를 설정할 수 있습니다. * 이는 ”내일 기온도 20도 근처일 것이고, 95% 확률로 16도에서 24도 사이일 것이다”라는 강한 믿음을 표현합니다.

4.2 2. 비정보적 사전분포 (Uninformative Prior)

파라미터에 대해 아는 것이 거의 없거나, 의도적으로 데이터의 영향력을 극대화하고 싶을 때 사용합니다. ”최소한의 정보”를 제공하는 사전분포입니다.

* 목적: 사전 지식의 영향을 최소화하고, 데이터(X)가 사후 분포를 거의 결정하도록 만듭니다. * 예시: * 새로운 치료법의 성공 확률 p (0에서 1 사이)를 모델링 합니다. * 사전분포: p 에 대해 아는 것이 전혀 없으므로, 0과 1 사이의 모든 값이 동일하게 가능하다고 가정합니다. * $p \sim Uniform(0, 1)$ (0과 1 사이의 균등 분포) * 이는 ”모든 확률값이 공평하다”는 약한 믿음을 표현합니다.

4.3 3. 결례 사전분포 (Conjugate Prior)

수학적, 계산적 편의성을 위해 특정 ’궁합이 잘 맞는’ 사전분포-가능도 조합을 사용하는 것입니다.

* 정의: 어떤 가능성도 함수 $f(X|\theta)$ 에 대해, 특정 사전분포 $f(\theta)$ 를 사용했더니, 그 결과물인 사후분포 $f(\theta|X)$ 가 사전분포와 동일한 분포 가족(family)이 되는 경우, 이 사전분포를 ’결례 사전분포’라고 부릅니다. * 비유: ”파란색 물감(Prior) + 노란색 물감(Likelihood) = 초록색 물감(Posterior)” 결례 관계는 ’파란색 + 노란색 = 초록색’이라는 공식을 아는 것과 같습니다. 만약 결례가 아니면, ”파란색 + 분홍색 = ??” 처럼, 그 결과를 한눈에 알 수 없고 계산이 복잡해집니다. * 왜 사용하는가? 복잡한 적분 계산 ($P(X) = \int f(X|\theta)f(\theta)d\theta$)을 피하고, 사후 분포의 파라미터를 간단한 공식으로 바로 유도할 수 있게 해줍니다.

주요 컬렉션 분포 조합				
가능도 (데이터)	파라미터	컬렉션 분포	→	사후 분포
Binomial (이항)	p (성공 확률)	Beta (베타)	→	Beta (베타)
Poisson (푸아송)	λ (비율)	Gamma (감마)	→	Gamma (감마)
Normal (정규)	μ (평균)	Normal (정규)	→	Normal (정규)
Normal (정규)	$1/\sigma^2$ (정밀도)	Gamma (감마)	→	Gamma (감마)
Exponential (지수)	λ (비율)	Gamma (감마)	→	Gamma (감마)

5 핵심 예제: 정규-정규 모델 (Normal-Normal Model)

베이지안 추론의 가장 기본이 되는 ”정규-정규” 모델을 통해 사전-사후 분석이 어떻게 작동하는지 살펴봅니다. 이는 ”데이터(가능도)도 정규분포를 따르고, 파라미터(사전분포)도 정규분포를 따른다”는 의미입니다.

문제 설정:

- 가능도 (데이터): $X_1, \dots, X_n \sim N(\mu, \sigma^2)$ (우리가 관찰한 데이터는 평균이 μ 이고 분산이 σ^2 인 정규분포에서 나왔다.)
- 가정: σ^2 (데이터의 분산)는 이미 알고 있다고 가정합니다.
- 파라미터: μ (데이터의 평균)는 모른다.
- 사전분포 (초기 믿음): μ 역시 정규분포를 따를 것이라고 가정합니다. $\mu \sim N(\mu_0, \sigma_0^2)$ (여기서 μ_0 는 ’사전 평균’, σ_0^2 는 ’사전 분산’이며, 둘 다 초매개변수(Hyperparameter)입니다.)

결과 (사후 분포): 결코 관계에 의해, 사후 분포 $f(\mu|X)$ 역시 정규분포가 됩니다! $f(\mu|X) \sim N(\mu_n, \sigma_n^2)$ 이때 업데이트된 사후 평균(μ_n)과 사후 분산(σ_n^2)은 다음과 같습니다.

$$\mu_n = \frac{\sigma^2 \mu_0 + n \sigma_0^2 \bar{X}}{\sigma^2 + n \sigma_0^2} \quad | \quad \sigma_n^2 = \frac{\sigma^2 \sigma_0^2}{\sigma^2 + n \sigma_0^2}$$

▣ 핵심 요약

정규-정규 모델의 직관적 해석

위 공식은 복잡해 보이지만, 매우 중요한 직관을 담고 있습니다.

1. 사후 평균(μ_n) = ”사전 믿음과 데이터의 가중 평균” * μ_n 은 사전 평균(μ_0)과 데이터 평균(\bar{X})의 가중 평균입니다. * 데이터가 많아지면 ($n \rightarrow \infty$)? 분자의 $n \sigma_0^2 \bar{X}$ 항이 압도적으로 커집니다. μ_n 은 \bar{X} (데이터 평균)에 수렴합니다. 결론: 충분한 데이터는 결국 사전 믿음을 이깁니다. * 사전 믿음이 매우 강하면 ($\sigma_0^2 \rightarrow 0$)? ”나의 초기 믿음 μ_0 는 절대 틀리지 않아!” (분산=0) μ_n 은 μ_0 (사전 평균)에 수렴합니다. 데이터(\bar{X})는 무시됩니다. * 사전 믿음이 매우 약하면 ($\sigma_0^2 \rightarrow \infty$)? ”나는 아무것도 몰라!” (분산=무한대) μ_n 은 \bar{X} (데이터 평균)에 수렴합니다.
2. 사후 분산(σ_n^2) = ”데이터가 많을수록 확신이 커진다” * σ_n^2 의 분모에 $n \sigma_0^2$ 항이 있습니다. * 데이터가 많아질수록 ($n \rightarrow \infty$), 분모가 커져서 σ_n^2 는 0에 수렴합니다. * 결론: 데이터를 더 많이 관찰할수록, 우리의 사후 믿음(추정)은 더욱 확실해집니다. (분산 감소)

6 베이지안 추정: 점과 구간

사후 분포 $f(\theta|X)$ 는 파라미터 θ 에 대한 우리의 최종 믿음을 나타내는 ’분포’입니다. 하지만 종종 보고를 위해 하나의 ’값’(점 추정)이나 ’구간’(구간 추정)이 필요합니다.

6.1 점 추정 (Point Estimation)

사후 분포(PDF)를 대표하는 하나의 값을 뽑는 방법입니다.

- 사후 평균 (Posterior Mean): $E[\theta|X]$. 사후 분포의 기댓값(평균)을 사용합니다.
- 사후 최빈값 (Posterior Mode, MAP): $\text{argmax}_\theta f(\theta|X)$. 사후 분포에서 확률 밀도가 가장 높은 지점

(가장 높은 봉우리)을 사용합니다. 이를 **MAP (Maximum A Posteriori)** 추정이라고 부릅니다.

- 사후 중앙값 (**Posterior Median**): 사후 분포의 50% 지점을 사용합니다.

6.2 구간 추정 (Interval Estimation)

주의사항

신용 구간 (Credible Interval) vs. 신뢰 구간 (Confidence Interval)

이 들은 매우 다르며, 베이지안의 '신용 구간'이 훨씬 직관적입니다.

- 베이지안 95% 신용 구간 (Credible Interval)
 - 해석: "관찰된 데이터를 바탕으로, 파라미터 θ 가 이 구간 안에 있을 확률이 95%이다."
 - 계산: 사후 분포 $f(\theta|X)$ 의 양쪽 꼬리 2.5%를 잘라내고 남은 95%의 중앙 구간.
 - 직관: 우리가 원하는 확률적 해석과 일치합니다.
- 빈도주의 95% 신뢰 구간 (Confidence Interval)
 - 해석: "이 실험(표본 추출)을 100번 반복하면, 100개의 '신뢰 구간' 중 95개의 구간이 '고정된 참 값 θ '를 포함할 것이다."
 - 주의: 이미 계산된 하나의 신뢰 구간(예: [10, 20])을 보고 " θ 가 [10, 20] 사이에 있을 확률이 95%다"라고 말하면 틀립니다. 빈도주의에서 θ 는 고정된 값이라 확률이 없으며, 확률은 '구간'에 걸려있습니다.

7 베이지안 선형 회귀 (Bayesian Linear Regression)

이러한 베이지안 원리를 선형 회귀 모델 $y = \beta_0 + \beta_1 x + \epsilon$ 에 적용할 수 있습니다.

1. 가능성 (데이터 모델): 선형 회귀의 기본 가정은 ”오차(ϵ)가 정규분포를 따른다”는 것입니다. 이는 y 역시 x 에 따라 평균이 변하는 정규분포를 따른다는 의미입니다.

$y_i \sim N(\mu_i, \sigma^2)$, 여기서 $\mu_i = \beta_0 + \beta_1 x_i$

2. 파라미터: 우리가 추정해야 할 파라미터는 β_0 (절편), β_1 (기울기), σ^2 (오차 분산)입니다.

3. 사전분포 (초기 믿음): 베이지안 접근법은 이 모든 파라미터에 사전분포를 할당합니다. (켤레 사전분포를 사용한다고 가정)

* $\beta_0 \sim N(\mu_0, \sigma_0^2)$ (절편에 대한 사전 믿음) * $\beta_1 \sim N(\mu_1, \sigma_1^2)$ (기울기에 대한 사전 믿음) * $1/\sigma^2 \sim Gamma(a_0, \lambda_0)$ (오차의 정밀도(분산의 역수)에 대한 사전 믿음)

4. 사후분포 (결과): 데이터(X, y)를 관찰하고 베이즈 정리를 적용하면, $\beta_0, \beta_1, \sigma^2$ 각각에 대한 사후 분포를 얻게 됩니다.

핵심 요약

베이지안 회귀의 의미

일반 선형 회귀(빈도주의)는 $\beta_1 = 5.0$ 처럼 하나의 값을 추정합니다.

베이지안 회귀는 β_1 에 대한 하나의 확률 분포를 제공합니다. (예: ” β_1 의 사후 분포는 평균이 5.0이고 표준편차가 0.5인 정규분포와 유사하다.”)

이를 통해 우리는 다음과 같은 강력한 확률적 해석이 가능해집니다:

- ” β_1 (기울기)가 0보다 클 확률은 99.8%이다.”
- ” β_1 의 95% 신용 구간은 [4.02, 5.98]이다.”

예제: 범주형 변수(더미 변수) 해석하기

한 강의의 수강생 그룹(4개)에 따라 숙제 시간을 예측하는 회귀 모델이 있습니다.

$$\hat{y} = 11.0 - 2.0x_{cs1090} + 3.5x_{cscie109} + 5.0x_{stat109}$$

- **기준 그룹 (Reference Group):** 모델 식에 없는 그룹, 즉 ac_{209} 가 기준 그룹(Baseline)입니다.
- **절편 (Intercept = 11.0) 해석:** 모든 x 가 0일 때의 \hat{y} 값입니다. 즉, 기준 그룹(ac_{209}) 학생들의 평균 숙제 시간은 11.0 시간입니다.
- **계수 (Coefficient = 5.0) 해석:** $x_{stat109}$ 의 계수 5.0은 기준 그룹(ac_{209})과의 평균 시간 차이'를 의미합니다. ” $stat_{109}$ 그룹 학생들은 ac_{209} 그룹 학생들보다 평균 5.0시간 더 많이 숙제한다.” ($stat_{109}$ 의 평균 시간 = $11.0 + 5.0 = 16.0$ 시간)
- **예측 (Prediction):** $cscie109$ 학생의 숙제 시간을 예측하려면, $x_{cscie109} = 1$ 이고 나머지는 0을 대입 합니다. $\hat{y} = 11.0 - 2.0(0) + 3.5(1) + 5.0(0) = 14.5$ 시간

8 심화: 릿지(Ridge)와 라쏘(Lasso)의 베이지안 해석

베이지안 모델링은 릿지(Ridge)와 라쏘(Lasso) 회귀가 왜 그렇게 작동하는지에 대한 강력한 직관을 제공합니다.

배경: 손실 함수 (Loss Function)

* **OLS (최소제곱법):** Loss = $\sum(y_i - \hat{y}_i)^2$ (오차의 제곱합, L_2 -loss) * **Ridge (릿지):** Loss = $\sum(y_i - \hat{y}_i)^2 + \lambda \sum \beta_j^2$ (오차의 제곱합 + L_2 -Penalty) * **Lasso (라쏘):** Loss = $\sum(y_i - \hat{y}_i)^2 + \lambda \sum |\beta_j|$ (오차의 제곱합 + L_1 -Penalty)

핵심 연결 고리: MAP 추정

베이지안에서 MAP (사후 최빈값) 추정은 사후 확률 $f(\beta|X)$ 를 최대화하는 β 를 찾는 것입니다.

$$\hat{\beta}_{MAP} = \underset{\beta}{\operatorname{argmax}} f(\beta|X) = \underset{\beta}{\operatorname{argmax}} [f(X|\beta)f(\beta)]$$

로그(log)를 써워도 최대화 지점은 동일합니다. (로그는 단조 증가 함수이므로)

$$\hat{\beta}_{MAP} = \underset{\beta}{\operatorname{argmax}} [\log(f(X|\beta)) + \log(f(\beta))]$$

이를 '최소화' 문제로 바꾸면, 음수(-)를 붙이면 됩니다.

$$\hat{\beta}_{MAP} = \underset{\beta}{\operatorname{argmin}} [-\log(f(X|\beta)) - \log(f(\beta))]$$

이제 이 식을 위 손실 함수들과 비교해 봅시다.

- $f(X|\beta)$ 는 $y \sim N(\beta X, \sigma^2)$ 정규분포(가능도)입니다. $-\log(f(X|\beta))$ 는 $\sum(y_i - \hat{y}_i)^2$ (오차 제곱합) 항에 비례합니다.
- $f(\beta)$ 는 β 계수들에 대한 사전분포(Prior)입니다. $-\log(f(\beta))$ 는 폐널티(Penalty) 항에 비례합니다.

핵심 요약

손실 함수 최소화 ≡ 사후 확률 최대화 (MAP)

(OLS Loss + Penalty) ≡ (로그-가능도 + 로그-사전분포)

8.1 1. 릿지(Ridge) = MAP + 정규(Normal) 사전분포

릿지의 L_2 -폐널티 $\lambda \sum \beta_j^2$ 는 어떤 사전분포 $f(\beta)$ 에서 유래할까요? $-\log(f(\beta)) \propto \beta^2$ 를 만족하는 분포를 찾으면 됩니다.

이는 평균이 0인 정규분포(Gaussian Prior)입니다. $f(\beta) \propto \exp(-\frac{\beta^2}{2\sigma_p^2}) \implies -\log(f(\beta)) \propto \beta^2$

colback=myblue!5!white, colframe=myblue!75!black, title=□ 릿지(Ridge)의 베이지안 해석 릿지 회귀는 β 계수들에 대해 평균 0의 정규(Normal) 사전분포를 가정한 베이지안 회귀의 MAP 추정치와 같습니다.

직관: 이 사전분포는 ” β 계수들은 0 근처에 완만하게(bell-curve) 모여 있을 것이다”라고 믿습니다. 이 믿음이 계수들을 0에 가깝게 ’당기지만(shrinkage)’ 0으로 만들지는 않습니다.

8.2 2. 라쏘(Lasso) = MAP + 라플라스(Laplace) 사전분포

라쏘의 L_1 -페널티 $\lambda \sum |\beta_j|$ 는 어떤 사전분포 $f(\beta)$ 에서 유래할까요? $-\log(f(\beta)) \propto |\beta|$ 를 만족하는 분포를 찾으면 됩니다.

이는 평균이 0인 라플라스(Laplace) 사전분포(이중 지수 분포)입니다. $f(\beta) \propto \exp(-\frac{|\beta|}{b}) \implies -\log(f(\beta)) \propto |\beta|$

colback=mygreen!5!white, colframe=mygreen!75!black, title=□ 라쏘(Lasso)의 베이지안 해석 라쏘 회귀는 β 계수들에 대해 평균 0의 라플라스(Laplace) 사전분포를 가정한 베이지안 회귀의 MAP 추정치와 같습니다.

직관(Sparsity): 라플라스 분포는 정규분포와 달리 $\beta = 0$ 지점에서 ’매우 뾰족한 첨탑’ 모양을 가집니다. 이 사전분포는 ”대부분의 β 계수들은 정확히 0일 것이다”라고 매우 강하게 믿습니다. 이 강한 믿음이 중요하지 않은 계수들을 0으로 만들어 변수 선택(feature selection)을 수행합니다.

colback=mygray, colframe=darkgray, breakable, title=시각적 비교: 정규 사전분포 vs. 라플라스 사전분포

- 정규분포(Normal, for Ridge): $\beta = 0$ 주변이 ’둥근 언덕’ 모양입니다. 0 근처의 값을 선호하지만, 정확히 0이어야 한다고 강하게 주장하지 않습니다.
 - 라플라스분포(Laplace, for Lasso): $\beta = 0$ 지점이 ’뾰족한 첨탑’ 모양입니다. 확률 질량이 0에 훨씬 많이 몰려있어, β 가 0이 될 확률을 훨씬 높게 부여합니다.
- (슬라이드 37의 두 그래프에 대한 텍스트 설명입니다.)

9 계산 방법: MCMC와 갑스 샘플링

9.1 문제: 사후 분포가 너무 복잡할 때

켤레 사전분포를 사용하면 사후 분포를 수학 공식으로 깔끔하게 유도할 수 있습니다. 하지만 모델이 복잡해지거나 켤레가 아닌 사전분포를 사용하면, 사후 분포 $f(\theta|X)$ 의 공식을 해석적으로(analytically) 풀 수 없습니다.

9.2 해결: 시뮬레이션 (MCMC)

공식을 직접 푸는 대신, 그 사후 분포에서 수많은 샘플을 추출(sampling)하여 분포의 모양을 근사합니다. 이를 MCMC (Markov Chain Monte Carlo) 방법이라고 부릅니다.

* 비유 (케이크 레시피): 내가 모르는 '신비한 케이크'(사후 분포)가 있습니다.

해석적 방법 (Conjugate): 케이크의 '레시피'(수학 공식)를 알아내는 것입니다.

MCMC 방법 (Simulation): 레시피는 모르지만, 이 케이크를 수만 조각 '샘플링'하여 맛볼 수 있는 기계가 있습니다. 수만 조각을 맛본(샘플링) 후, 우리는 이 케이크의 평균 당도(사후 평균), 당도의 편차(사후 분산) 등을 경험적으로(empirically) 추정할 수 있습니다.

MCMC 샘플($\theta^{(1)}, \theta^{(2)}, \dots, \theta^{(N)}$)을 얻은 후, 추정은 매우 간단해집니다.

* 사후 평균: 샘플들의 단순 평균 $\frac{1}{N} \sum \theta^{(i)}$ * 95% 신용 구간: 샘플들을 정렬한 뒤, 2.5% 분위수와 97.5% 분위수를 찾습니다.

9.3 갑스 샘플링 (Gibbs Sampling)

MCMC의 대표적인 알고리즘 중 하나로, 다차원 파라미터를 다룰 때 유용합니다. (예: $\theta_1, \theta_2, \theta_3$ 세 개의 파라미터를 동시에 추정해야 할 때)

갑스 샘플링은 결합 사후 분포 $f(\theta_1, \theta_2, \theta_3|X)$ 를 직접 샘플링하기 어려울 때, 대신 '전체 조건부 분포 (Full Conditional Distributions)'를 이용해 번갈아 샘플링합니다.

* $f(\theta_1|\theta_2, \theta_3, X) * f(\theta_2|\theta_1, \theta_3, X) * f(\theta_3|\theta_1, \theta_2, X)$

알고리즘 (3개 파라미터 기준): 1. 초기 값 ($\theta_1^{(0)}, \theta_2^{(0)}, \theta_3^{(0)}$)을 임의로 설정합니다. 2. 반복 (Iteration $t = 1, 2, \dots$): a. $\theta_1^{(t)}$ 를 $f(\theta_1|\theta_2^{(t-1)}, \theta_3^{(t-1)}, X)$ 에서 샘플링. b. $\theta_2^{(t)}$ 를 $f(\theta_2|\theta_1^{(t)}, \theta_3^{(t-1)}, X)$ 에서 샘플링 (방금 뽑은 새 값 사용). c. $\theta_3^{(t)}$ 를 $f(\theta_3|\theta_1^{(t)}, \theta_2^{(t)}, X)$ 에서 샘플링 (방금 뽑은 새 값을 사용). 3. 초기 수천 개의 샘플(예: 1 1000번째)은 초기 값의 영향을 받으므로 'Burn-in' 기간이라 부르며 폐기합니다. 4. Burn-in 이후의 샘플 ($\theta_1^{(k)}, \theta_2^{(k)}, \theta_3^{(k)}$)들이 우리가 원하는 결합 사후 분포의 샘플이 됩니다.

colback=mygray, colframe=darkgray, breakable, title=시각적 비유: 메트로폴리스-헤이스팅스 (Metropolis-Hastings) MCMC의 또 다른 유명한 알고리즘은 '산(사후 분포)을 오르는 맹인 등반가'로 비유할 수 있습니다.

1. 등반가는 현재 위치(θ)에서 임의의 다음 지점(θ^*)을 제안합니다. 2. **UPHILL (오르막):** $f(\theta^*) > f(\theta)$ 이면, 무조건 이동합니다. (더 가능성 높은 곳) 3. **DOWNHILL (내리막):** $f(\theta^*) < f(\theta)$ 이면, 확률적($f(\theta^*)/f(\theta)$)으로 이동합니다. (가파른 내리막(낭떠러지)은 거의 가지 않고, 완만한 내리막은 가끔 갑니다.) 4. 이 과정을 반복하면, 등반가는 결국 산의 정상(MAP) 주변에서 대부분의 시간을 보내게 됩니다.

(슬라이드 48의 그림에 대한 텍스트 설명입니다.)

10 자주 묻는 질문 (FAQ) 및 점검

주의사항

Q: 릿지(Ridge) 모델을 '훈련'할 때는 페널티 항($\lambda \sum \beta^2$)을 쓰는데, 왜 '검증(validation)' 할 때는 페널티 항을 제외한 MSE만 사용하나요?

A: '훈련'과 '검증'의 목적이 다르기 때문입니다.

- **훈련 (Training)의 목적:** 최적의 β 계수를 찾는 것입니다. 이때 페널티 항($\lambda \sum \beta^2$)은 계수가 너무 커지지 않도록 막는 '규제 장치(regularizer)'입니다. 이 장치는 모델을 만드는 과정(process)의 일부입니다.
- **검증 (Validation)의 목적:** 완성된 모델이 새로운 데이터를 얼마나 잘 예측하는지 평가하는 것입니다. 평가를 할 때는 "그래서 예측 값(\hat{y})이 실제 값(y)과 얼마나 차이 나는가?"라는 '자연스러운' 예측 오차(natural error metric)만 측정해야 합니다. 훈련 과정의 도구였던 페널티 항을 평가에까지 포함시키면, 모델의 순수한 예측 성능을 왜곡하게 됩니다.

11 빠르게 훑어보기 (1-Page Summary)

▣ 핵심 요약

1. 베이지안이란?

- 파라미터(θ)를 '고정된 값'이 아닌 '확률 분포(믿음)'로 본다.
- $P(\theta|X) \propto P(X|\theta)P(\theta)$ 공식을 사용한다.
- (최종 믿음) \propto (증거의 힘) \times (초기 믿음)

2. 4가지 핵심 요소

- 사전확률 (Prior $P(\theta)$): 내 초기 믿음. (예: $Uniform(0, 1)$, $N(0, 10)$)
- 가능성 (Likelihood $P(X|\theta)$): 데이터가 말하는 증거. (예: $Binomial(n, \theta)$)
- 사후확률 (Posterior $P(\theta|X)$): 데이터 반영 후 업데이트된 최종 믿음. (우리의 결과물)
- 증거 (Evidence $P(X)$): 정규화 상수. (합을 1로 만듦)

3. 사전분포의 종류

- Informative (정보적): 전문가 지식 반영.
- Uninformative (비정보적): 데이터가 말하게 함. (예: $Uniform$)
- Conjugate (켤레): 계산의 편의성. (예: 이항-베타, 정규-정규)

4. 베이지안 vs. 빈도주의 구간

- 신용 구간 (Bayesian): " θ 가 이 안에 있을 확률 95%" (직관적)
- 신뢰 구간 (Frequentist): "구간 100개 뽑으면 95개가 θ 를 포함" (해석 주의)

5. 릿지/라쏘와의 연결 (MAP 추정)

- Ridge (릿지): β 에 정규(Normal) 사전분포를 가정한 것. (계수를 0 근처로 당김)
- Lasso (라쏘): β 에 라플라스(Laplace) 사전분포를 가정한 것. (뾰족한 분포 \rightarrow 계수를 0으로 만듦
→ 변수 선택)

6. 계산 (MCMC)

- 사후 분포의 공식을 풀기 어려울 때, 대신 수만 개의 샘플을 뽑아 분포를 근사한다.

- **Gibbs Sampling:** 조건부 분포를 이용해 파라미터를 하나씩 번갈아 샘플링.

December 10, 2025

- 강의명: CS109A: 데이터 과학 입문
- 주차: Lecture 12
- 교수명: Pavlos Protopapas, Kevin Rader, Chris Gumb
- 목적: Lecture 12의 핵심 개념 학습

Contents

1	강의 개요	2
2	주요 공지: 중간고사 및 과제	3
2.1	중간고사 (Midterm)	3
2.2	과제 (Homework)	3
3	핵심 용어 정리	4
4	베이즈 추론: 복잡한 모델의 해법, 시뮬레이션	5
4.1	왜 시뮬레이션이 필요한가?	5
4.2	시뮬레이션 샘플의 활용	5
4.3	MCMC 기법 소개: 샘플은 어떻게 뽑는가?	5
5	빅데이터와 고차원성의 문제	7
5.1	빅데이터(Big Data)란 무엇인가?	7
5.2	N이 큰 경우: 많은 관측치	7
5.3	P가 큰 경우: 많은 예측 변수 (고차원성)	7
6	주성분 분석 (Principal Components Analysis, PCA)	9
6.1	PCA의 핵심 아이디어: 정보의 요약	9
6.2	PCA의 수학적 직관: 고유벡터와 고유값	9
7	PCA의 활용: 시각화와 회귀 분석	10
7.1	활용 1: 고차원 데이터의 시각화 (Visualization)	10
7.2	활용 2: 주성분 회귀 (PCA for Regression, PCR)	10
7.3	PCA의 장단점 요약	11
8	중간고사 핵심 개념 복습	12
8.1	가설 검정 (Hypothesis Testing)	12

8.2 순열 검정 (Permutation Test)	12
8.3 부트스트랩 vs. 순열 검정	13
8.4 신뢰 구간 vs. 예측 구간	13
9 중간고사 대비 체크리스트	14
10 초심자를 위한 FAQ	15
11 빠르게 훑어보기 (1-Page Summary)	16

1 강의 개요

▣ 핵심 요약

본 강의는 데이터 과학의 두 가지 주요 주제인 **베이즈 추론**과 **차원 축소**를 다룹니다. 먼저, 복잡한 모델의 결과를 해석하기 위해 시뮬레이션(MCMC 등)을 사용하는 베이즈 계산 방법을 배웁니다. 이후, 예측 변수(P)가 매우 많은 '고차원 데이터'의 문제점을 알아보고, 이를 해결하기 위한 강력한 기법인 **주성분 분석(PCA)**을 집중적으로 학습합니다. 마지막으로 중간고사를 대비하여 **가설 검정**, **p-value**, **순열 검정** 등 핵심 통계 개념을 복습합니다.

이번 주 학습 목표

- 복잡한 후험 분포(Posterior)를 시뮬레이션하는 MCMC 기법의 원리를 이해합니다.
- '고차원성의 저주'가 무엇인지, 왜 'P가 큰' 데이터가 문제가 되는지 설명할 수 있습니다.
- 주성분 분석(PCA)의 핵심 아이디어를 "최대 분산 방향 찾기"로 설명할 수 있습니다.
- PCA를 활용한 두 가지 주요 사례(시각화, 회귀 분석)를 구분하고 적용할 수 있습니다.
- 고전적 t-검정과 컴퓨터 기반의 순열 검정(Permutation Test)의 차이점을 설명할 수 있습니다.

2 주요 공지: 중간고사 및 과제

중간고사에 대한 주요 공지사항입니다. 시험 준비에 참고하세요.

2.1 중간고사 (Midterm)

- **시기:** 다음 주 섹션 시간.
- **형식 (In-Class):**
 - 섹션 시간 전체 (75분) 동안 진행됩니다.
 - 퀴즈보다 약 2.2배 긴 분량입니다.
 - 객관식(multiple-choice) 문제와 단답형/빈칸 채우기(fill-in-the-blank) 문제로 구성됩니다.
 - **오픈북이 아닙니다.** (Closed book)
- **치트 시트 (Cheat Sheets):**
 - 총 2장의 치트 시트 (양면 사용 가능)를 허용합니다. (퀴즈는 1장)
- **별도 코딩 시험 (Take-home Coding Portion):**
 - 수업(섹션)에서 보는 필기시험과 별개로 진행됩니다.
 - 필기시험 이후에 공개되며, **24시간**의 창(window)이 주어집니다.
 - 코딩 시험을 시작하면 **2시간** (또는 3시간, 추후 확정 공지)의 제한 시간 내에 완료해야 합니다.
 - AI (LLM) 사용은 금지되지만, 강의 노트 등은 참고 가능합니다.
 - 예상 소요 시간은 2시간 미만이나, 문제 발생 시를 대비해 2시간을 부여합니다.
- **시험 범위:** 오늘 강의(Lecture 12)까지 다룬 모든 주제. (분류 모델링 등 다음 주 내용은 포함되지 않음)
- **연습 문제:**
 - 지난 퀴즈의 정답 키(answer key)가 게시될 예정입니다.
 - 추가 연습 문제 및 복습 자료가 금요일에 공개될 예정입니다.
 - 코딩 연습 문제 제공은 미정입니다. (No promises)

2.2 과제 (Homework)

- **Homework 3 마감일:** 11월 4일 경으로, 중간고사 이후 약 2주 뒤입니다.
- **중요:** HW 3는 중간고사 범위를 많이 다루고 있습니다.
- 마감일이 멀더라도, 중간고사 준비를 위해 반드시 HW 3를 미리 시작하고 풀어보아야 합니다.
- 시험 전까지 모든 코드를 완벽하게 정리할 필요는 없지만, 내용을 읽고 시도해보는 것이 시험에 유리합니다.

3 핵심 용어 정리

이번 강의에서 다루는 주요 용어들을 정리했습니다.

용어	쉬운 설명	원어	비고
베이즈 추론	데이터(증거)를 바탕으로 기존의 믿음(사전 확률)을 업데이트하는 통계적 방식	Bayesian Inference	믿음 → 증거 → 새로운 믿음
후험 분포	데이터를 관찰한 후 업데이트된 파라미터의 확률 분포	Posterior Distribution	베이즈 추론의 '결과물'
MCMC	복잡한 후험 분포에서 샘플을 추출하는 시뮬레이션 기법의 총칭	Markov Chain Monte Carlo	무작위로 걸어 다니며 샘플 수집
고차원성	데이터의 특성(Feature) 또는 예측 변수(p)의 수가 매우 많은 상태	High Dimensionality	열(Column)이 매우 많은 데이터
차원의 저주	차원이 증가할수록 데이터가 희소(sparse)해지고 분석이 어려워지는 현상	Curse of Dimensionality	"데이터가 외로워진다"
주성분 분석 (PCA)	고차원 데이터의 정보를 최대한 보존하며 저차원으로 축소하는 기법	Principal Components Analysis (PCA)	데이터의 '정보 요약' 기법
주성분	PCA를 통해 새로 생성된, 데이터의 분산을 최대로 설명하는 축(변수)	Principal Component (PC)	원본 변수들의 '선형 조합'
고유값	해당 축(고유벡터)이 설명하는 '분산의 크기' 또는 '중요도'	Eigenvalue	PCA에서는 PC의 중요도를 의미
고유벡터	데이터 공분산 행렬의 '방향'을 나타내는 벡터	Eigenvector	PCA에서는 새 축의 '방향'을 의미
순열 검정	데이터의 라벨을 무작위로 섞어(H_0 가정) 검정 통계량의 분포를 만드는 기법	Permutation Test	t-검정의 비모수적 대안

Table 1: PCA 및 베이즈, 통계 복습 관련 핵심 용어

4 베이즈 추론: 복잡한 모델의 해법, 시뮬레이션

4.1 왜 시뮬레이션이 필요한가?

베이즈 추론의 핵심은 사전 확률(Prior)과 가능도(Likelihood)를 결합하여 **후험 분포(Posterior Distribution)**을 얻는 것입니다.

- **단순한 경우:** 만약 우리가 사용한 사전 분포가 '켤레 사전 분포(Conjugate Prior)'처럼 공식이 잘 맞는 짝이라면, 후험 분포는 "정규 분포"나 "감마 분포"처럼 우리가 잘 아는 깔끔한 형태로 나옵니다. 이 경우 평균, 중위값, 신뢰 구간 등을 수학 공식으로 쉽게 계산할 수 있습니다.
- **복잡한 경우:** 하지만 현실의 모델(예: 다중 선형 회귀)에서는 파라미터가 매우 많습니다(예: $\beta_0, \beta_1, \dots, \beta_p$ 그리고 σ^2). 이 모든 파라미터들의 결합(joint) 후험 분포는 매우 복잡하고 다차원적인 형태를 띠게 됩니다. 이런 분포는 수학 공식 하나로 깔끔하게 표현하거나 적분하기가 거의 불가능합니다.

복잡한 분포를 이해하는 방법: 시뮬레이션 수학 공식으로 풀 수 없다면, 컴퓨터의 힘을 빌려 그 분포에서 수천, 수만 개의 샘플을 직접 뽑아보면 됩니다. 이렇게 뽑힌 샘플들의 분포(히스토그램)를 관찰하면, 원래의 복잡한 후험 분포가 어떻게 생겼는지 근사적으로 파악할 수 있습니다.

이처럼 복잡한 분포에서 샘플을 추출하는 계산(computational) 기법들을 **MCMC(Markov Chain Monte Carlo)**라고 부릅니다.

4.2 시뮬레이션 샘플의 활용

MCMC 등을 통해 후험 분포에서 N_{sims} 개의 샘플(예: 10,000개의 β_1 값)을 얻었다고 가정합시다. 이 샘플들을 어떻게 사용할까요?

- **후험 평균(Posterior Mean):** 매우 쉽습니다. 10,000개 샘플의 표본 평균을 계산하면 됩니다.
- **신뢰 구간(Credible Interval):** 매우 쉽습니다. 10,000개 샘플을 정렬한 뒤, 백분위수(Percentile)를 사용하면 됩니다. (예: 95% 신뢰 구간 = 2.5% 지점 값과 97.5% 지점 값). 이는 부트스트랩(Bootstrap)에서 신뢰 구간을 구하는 방식과 동일합니다.
- **후험 최빈값(Posterior Mode):** 어렵습니다. 샘플 데이터만으로는 분포의 가장 높은 '봉우리(peak)'를 정확히 찾기 어렵습니다. 히스토그램을 그려볼 순 있지만, 구간(bin) 설정에 따라 모양이 바뀝니다. 따라서 '커널 밀도 추정(Kernel Density Estimate, KDE)' 같은 기법으로 부드러운 곡선을 피팅한 후, 그 곡선의 최댓값을 찾는 '蹦프 헌팅(bump-hunting)' 과정이 필요합니다.

4.3 MCMC 기법 소개: 샘플은 어떻게 뽑는가?

MCMC는 복잡한 분포의 정확한 모양(수식)을 몰라도, 특정 지점의 '상대적 높이(확률 밀도)'만 알면 샘플을 뽑을 수 있게 해주는 기법들입니다.

- **적응적 기각 샘플링 (Adaptive Rejection Sampling):** 분포에 '다트'를 던지는 것과 비슷합니다. 분포를 감싸는 단순한 제안 분포(proposal distribution)에서 샘플(x, y 좌표)을 뽑습니다. 만약 뽑힌 샘플이 실제 분포 곡선 '아래'에 떨어지면 '수락(Accept)'하고, 곡선 '위'에 떨어지면 '기각(Reject)'합니다.
- **깁스 샘플링 (Gibbs Sampling):** 다변수 결합 분포를 다룰 때 유용합니다. $f(\theta_1, \theta_2, \theta_3)$ 를 직접 샘플링하는 대신, $\theta_1|\theta_2, \theta_3 / \theta_2|\theta_1, \theta_3 / \theta_3|\theta_1, \theta_2$ 처럼 각 변수의 조건부 분포(conditional distribution)를 번갈아 가며 샘플링합니다.

- **메트로폴리스-헤이스팅스 (Metropolis-Hastings):** 가장 유명한 MCMC 알고리즘 중 하나입니다.

메트로폴리스-헤이스팅스: 안데 쓴 등산가 복잡한 확률 분포를 '산맥'이라고 상상해봅시다. 우리는 이 산맥의 지형도(분포의 전체 형태)는 모르지만, 현재 위치의 '고도(확률 밀도)'는 측정할 수 있습니다.

1. **시작:** 산 중턱 임의의 지점(초기 값 x)에서 시작합니다.
2. **제안:** 다음 발걸음을 내디딜 방향과 거리(새로운 위치 x^*)를 무작위로 제안합니다.

3. **결정:**

- **오르막길 (이동):** 만약 x^* 의 고도가 현재 x 의 고도보다 높다면 (즉, 확률이 더 높다면), 무조건 그쪽으로 이동합니다. ($R = f(x^*)/f(x) > 1$)
- **내리막길 (확률적 이동):** 만약 x^* 의 고도가 더 낮다면, 확률적으로 이동합니다.
 - 완만한 내리막: (예: $R = 0.8$) 80% 확률로 이동합니다.
 - 가파른 내리막(절벽): (예: $R = 0.01$) 1%의 매우 낮은 확률로만 이동합니다.
- **거절:** 만약 이동하지 않기로 결정되면(예: 80% 확률로 이동에 실패), 제자리에 머무릅니다. (그리고 현재 위치 x 를 샘플에 한 번 더 추가합니다.)

4. **반복:** 2 3번 과정을 수만 번 반복합니다.

결과: 이 '등산가'는 높은 고도(확률이 높은 지역)에서 더 많은 시간을 보내고, 낮은 고도(확률이 낮은 지역)에서는 적은 시간을 보내게 됩니다. 이 등산가의 발자취(방문한 위치 목록)를 모으면, 그것이 바로 우리가 원하던 후험 분포의 샘플이 됩니다.

5 빅데이터와 고차원성의 문제

5.1 빅데이터(Big Data)란 무엇인가?

'빅데이터'는 단순히 데이터가 많은 것을 의미하지만, 데이터 과학에서는 두 가지 다른 차원의 '큼'을 구분해야 합니다. 데이터셋을 행렬(Rows \times Columns)로 볼 때:

- **N이 큰 경우 (Large N):** 행(Row)의 수가 매우 많은 경우 (예: 수백만, 수십억 개의 관측치)
- **P가 큰 경우 (Large P):** 열(Column)의 수가 매우 많은 경우 (예: 수천, 수만 개의 예측 변수)

이 두 상황은 서로 다른 문제를 야기합니다.

5.2 N이 큰 경우: 많은 관측치

- **문제점:**
 - **계산 비용:** 알고리즘이 매우 느려집니다. 단순 평균 계산조차 오래 걸리며, 특히 교차 검증(CV)이나 부트스트랩처럼 반복 계산이 필요하면 시간이 기하급수적으로 늘어납니다.
 - **편향(Bias) 문제:** 데이터가 '많이' 편향된 방식(non-representative)으로 수집되었다면, N이 커질 수록 오히려 편향이 심화되어 결과가 나빠질 수 있습니다. ("Garbage in, garbage out")
- **해결책:**
 - **서브샘플링 (Sub-sampling):** 전체 데이터의 10% 또는 1%만 무작위로 추출하여 모델을 학습해도 충분히 좋은 성능을 낼 수 있습니다.
- **특이점:** N이 극도로 커지면(예: 수억 개), 통계적 추론(p-value, 신뢰 구간)의 중요성이 낮아집니다. 왜냐하면 표준 오차가 0에 수렴하여 모든 변수가 '통계적으로 유의미 ($p < 0.05$)'하게 나오기 때문입니다.

5.3 P가 큰 경우: 많은 예측 변수 (고차원성)

N(행)은 적당히 있지만 P(열, 예측 변수)가 N에 가깝거나 N보다 훨씬 많은 경우, 심각한 문제들이 발생합니다.

- **과적합 (Overfitting):** 모델의 자유도가 너무 높아져, 실제 신호(signal)가 아닌 학습 데이터의 노이즈 (noise)까지 완벽하게 외워버립니다. 결과적으로 새로운 데이터에 대한 예측 성능이 급격히 떨어집니다.
- **다중공선성 (Multicollinearity):** 예측 변수 간에 높은 상관관계가 존재할 확률이 높습니다.
- **수학적 문제:** OLS(최소제곱법)에서 $X^T X$ 행렬의 역행렬을 계산할 수 없게 됩니다 (Unidentifiability).
- **차원의 저주 (Curse of Dimensionality):** P가 커질 때 발생하는 근본적인 문제입니다.

주의사항

차원의 저주(Curse of Dimensionality)란? 차원이 증가할수록 (P가 커질수록) 데이터가 존재하는 공간의 부피(Volume)가 기하급수적으로 커집니다. 이로 인해 동일한 N개의 데이터라도, 차원이 높아지면 데이터 포인트들은 서로에게서 엄청나게 멀리 떨어져 희소(sparse)하게 흩어지게 됩니다.

직관적 비유 1: 큐브 속의 구

- **2차원 (정사각형 안의 원):** 한 변이 2인 정사각형(넓이 4) 안에 반지름 1인 원(넓이 $\pi \approx 3.14$)이 차지하는 비율은 $\pi/4 \approx 78.5\%$ 입니다.

- **3차원 (정육면체 안의 구):** 한 변이 2인 정육면체(부피 8) 안에 반지름 1인 구(부피 $\frac{4}{3}\pi \approx 4.19$)가 차지하는 비율은 $\approx 52.3\%$ 입니다.
- **10차원 (10D-큐브 안의 10D-구):** 이 비율은 약 0.25%로 급격히 떨어집니다.

결론: 차원이 높아질수록, 데이터는 대부분 구(중심부)가 아닌 큐브의 '모서리'에 존재하게 됩니다.

직관적 비유 2: 외로운 데이터 포인트

- 1차원에서는 10개의 점이 꽤 촘촘히 모여있습니다.
- 2차원, 3차원으로 갈수록 같은 10개의 점이 서로 멀리 떨어집니다.
- 1000차원에서는 모든 데이터 포인트가 서로 엄청나게 멀리 떨어져 있습니다. "가까운 이웃(neighbor)"이라는 개념 자체가 무의미해집니다. (k-NN 같은 알고리즘이 작동하기 힘든 이유)

6 주성분 분석 (Principal Components Analysis, PCA)

PCA는 'P가 큰' 고차원성 문제를 해결하기 위한 강력한 차원 축소 (Dimensionality Reduction) 기법입니다.

6.1 PCA의 핵심 아이디어: 정보의 요약

- 문제:** 1000개의 예측 변수($P=1000$)가 있지만, 이 중 상당수는 서로 상관관계가 높아 중복된 정보 (redundant)를 담고 있습니다.
- 목표:** 1000개의 변수에 흩어져 있는 '진짜 정보(분산)'를 최대한 보존하면서, 이들을 대표할 수 있는 새로운 축(변수) 몇 개(예: 10개)로 압축하고 싶다.
- 해결책 (PCA):** PCA는 원본 변수들의 선형 조합(linear combination)을 통해, 데이터의 분산(Variance)을 가장 크게 설명하는 새로운 축을 순서대로 찾아냅니다.

PCA: 데이터 구름의 '최적의 축' 찾기 2개의 변수(X_1, X_2)가 있고, 이들의 산점도(scatter plot)가 마치 '길고 얕게 기울어진 타원형 구름'처럼 보인다고 상상해봅시다.

- 기존 축 (X_1, X_2):** X_1 축이나 X_2 축만으로는 이 구름의 흩어짐(분산)을 잘 설명하지 못합니다.
- PCA의 새 축 (Z):**
 - 제1 주성분 ($Z_1, PC1$):** PCA는 이 구름이 가장 길게 뻗어 있는 방향(기울어진 축)을 찾아냅니다. 이 축이 바로 Z_1 입니다. Z_1 은 이 데이터의 분산을 '최대'로 설명합니다(예: 전체 분산의 88% 설명).
 - 제2 주성분 ($Z_2, PC2$):** Z_2 는 Z_1 에 수직(orthogonal)이면서 남은 분산을 최대로 설명하는 축입니다(예: 나머지 12% 설명).

결론: X_1, X_2 대신 Z_1 하나만 사용해도 원본 정보의 88%를 보존할 수 있습니다. 즉, 2차원(X_1, X_2) 데이터를 1차원(Z_1) 데이터로 성공적으로 '차원 축소'한 것입니다.

6.2 PCA의 수학적 직관: 고유벡터와 고유값

(선형대수학을 모른다면 이 부분은 넘어가도 괜찮습니다.)

PCA가 이 '최적의 축'을 찾는 수학적 도구가 바로 고유벡터(Eigenvector)와 고유값(Eigenvalue)입니다. PCA는 원본 예측 변수 X 의 공분산 행렬($X^T X$)에 대해 '고유값 분해(Eigen-decomposition)'를 수행합니다.

- 고유벡터 (Eigenvector):** 공분산 행렬의 '방향'을 나타냅니다.
 - 주성분(PC)의 방향(예: Z_1 축의 방향)이 됩니다.
- 고유값 (Eigenvalue):** 해당 고유벡터 방향으로 데이터가 얼마나 '퍼져있는지(분산)'를 나타내는 '값'입니다.
 - 해당 주성분이 설명하는 분산의 크기 (PC의 '중요도')가 됩니다.

PCA는 가장 큰 고유값을 가진 고유벡터를 제1 주성분(PC1)으로, 두 번째로 큰 것을 제2 주성분(PC2)으로 순서대로 선택합니다.

PCA는 '회전'이다 PCA는 기존의 X_1, X_2 축을 데이터 분산이 최대가 되는 Z_1, Z_2 축으로 회전(rotation)시키는 선형 변환(linear transformation)입니다.

7 PCA의 활용: 시각화와 회귀 분석

7.1 활용 1: 고차원 데이터의 시각화 (Visualization)

- 문제:** 784개의 변수($P=784$)를 가진 Fashion MNIST 이미지 데이터를 어떻게 2D 평면에 시각화할 수 있을까요?
- PCA 해결책:** 1. 784개 변수를 사용해 PCA를 실행합니다. (총 784개의 주성분 Z_1, \dots, Z_{784} 가 나옵니다.) 2. 이 중 가장 중요한 (즉, 분산을 가장 많이 설명하는) 단 2개(Z_1, Z_2)만 선택합니다. 3. 모든 데이터 포인트를 Z_1 축과 Z_2 축으로 구성된 2D 평면에 뿐립니다.
- 결과:** 이 2D 산점도는 784차원 공간에 존재하는 데이터 구름의 '가장 특징이 잘 드러나는 2차원 그림자'라고 할 수 있습니다. 우리는 이 2D 그림을 보고 "아, 데이터가 대략 3개의 뉴어리(cluster)로 나뉘는구나" 하고 파악할 수 있습니다.
- 펭귄 데이터 예시:** 4개의 측정치(bill length, bill depth 등)를 PCA로 2차원(PC1, PC2)으로 축소하여 시각화했더니, 3종류의 펭귄(Adelie, Chinstrap, Gentoo)이 잘 분리되어 보이는 것을 확인할 수 있었습니다.

주의사항

PCA는 Y(라벨)를 모른다 (Unsupervised) PCA는 시각화 시 데이터의 '종류'(예: 펭귄 종류, 옷 종류)를 전혀 고려하지 않습니다. PCA는 오직 X 변수들의 '퍼짐(분산)'만 보고 축을 정합니다.

그럼에도 불구하고 PCA 2D 플롯에서 라벨별로 군집이 잘 분리되었다면, 이는 X 변수들이 Y 를 예측하는 데 유용한 정보를 담고 있다는 강력한 신호입니다.

7.2 활용 2: 주성분 회귀 (PCA for Regression, PCR)

PCA를 회귀 분석의 전처리 단계로 사용하여 과적합을 방지할 수 있습니다.

- 1단계: PCA 수행** P 개의 원본 변수(X_1, \dots, X_P)로 PCA를 수행하여 P 개의 주성분(Z_1, \dots, Z_P)을 만듭니다.
- 2단계: 주성분 선택 (m 개)** P 개의 주성분 중 상위 m 개 (단, $m < P$)만 선택합니다. m 을 결정하는 방법은 3가지가 있습니다.
 - A. 스크리 플롯 (Scree Plot) / 엘보우 방법 (Elbow Method):** 각 주성분(PC1, PC2, ...)이 설명하는 분산의 크기를 막대그래프로 그립니다. 그래프가 급격히 꺾이는 '팔꿈치(elbow)' 지점에서 m 을 결정합니다. (예: 20개 이후로는 설명력이 급감하니 20개만 쓰자)
 - B. 누적 설명 분산 (Cumulative Variance Explained):** "전체 분산의 90%를 설명하는 지점까지" m 을 선택합니다. (예: PC 53개를 더하니 누적 분산이 90%가 되었다면 $m = 53$ 으로 설정)
 - C. 교차 검증 (Cross-Validation):** m 을 모델의 하이퍼파라미터로 취급합니다. $m = 1, 2, 3, \dots$ 일 때의 검증 MSE를 각각 계산하여, MSE가 가장 낮은 최적의 m 을 선택합니다. (가장 성능 지향적인 방법)
- 3단계: 회귀 모델 학습** 선택된 m 개의 주성분을 새로운 예측 변수로 사용하여 선형 회귀 모델을 학습합니다.

$$Y = \beta_0 + \beta_1 Z_1 + \beta_2 Z_2 + \dots + \beta_m Z_m$$

7.3 PCA의 장단점 요약

PCA는 유용하지만 만능은 아닙니다.

장점 (Pros)	단점 (Cons)
1. 차원의 저주 해결: 과적합 위험을 크게 줄여줍니다.	1. 해석력 상실: 주성분은 원본 변수들의 복잡한 조합
2. 다중공선성 제거: 주성분들은 정의상 서로 수직 (직교) 하므로, 변수 간 상관관계가 0이 됩니다.	(예: $Z_1 = 0.5X_1 - 0.2X_2 + \dots$) 이므로, β_1 이 무엇을 의미하는지 직관적으로 해석하기 불가능해집니다.
3. 시각화 가능: 고차원 데이터를 2D/3D로 시각화할 수 있습니다.	2. Y 정보 무시 (Unsupervised): PCA는 Y를 전혀 보지 않습니다. X의 분산을 90% 설명하는 PC1이 Y를 예측하는 데는 전혀 중요하지 않을 수도 있습니다. (최악의 경우 Y 예측에 중요한 정보가 PC100에 있을 수도 있음)
4. 계산 효율성 향상: 변수의 수가 줄어 모델 학습이 빨라집니다.	3. 예측 성능 항상 보장 없음: 성능이 항상 좋아지지는 않습니다.

Table 2: PCA의 장점과 단점

8 중간고사 핵심 개념 복습

8.1 가설 검정 (Hypothesis Testing)

가설 검정은 우리가 데이터에서 관찰한 효과(예: β_1 의 기울기)가 '실제 효과'인지, 아니면 '단순한 우연(random chance)'에 의한 것인지 판단하는 통계적 절차입니다.

가설 검정의 5단계:

1. 가설 설정:

- 귀무가설 (H_0): "효과가 없다." (예: $\beta_1 = 0$. 즉, X 와 Y 는 관련이 없다.)
- 대립가설 (H_A): "효과가 있다." (예: $\beta_1 \neq 0$)

2. 검정 통계량 선택: 가설을 검증할 측도(measure)를 정합니다. (예: t-statistic)

3. 검정 통계량 계산: 수집한 데이터로 해당 통계량을 계산합니다. (예: $\hat{\beta}_1 / SE(\hat{\beta}_1)$)

4. p-value 계산 및 결정: 이 통계량이 H_0 하에서 얼마나 극단적인 값인지 확률(p-value)로 계산합니다. (보통 $\alpha = 0.05$ 와 비교)

5. 결론 도출:

- $p < 0.05$: H_0 을 기각(Reject) 합니다. (즉, $\beta_1 \neq 0$ 일 가능성성이 높다.)
- $p \geq 0.05$: H_0 을 기각하는 데 실패(Fail to Reject) 합니다.

p-value란 무엇인가? p-value란, "만약 귀무가설(H_0)이 사실이라면, 우리가 관찰한 검정 통계량(예: $t = 2.5$)보다 같거나 더 극단적인 값이 나올 확률"을 의미합니다.

- p-value가 낮다(예: 0.01): H_0 이 사실이라는 가정 하에서는 거의 일어나지 않을(1%) 일이 벌어졌다
→ "아무 효과가 없다"는 H_0 가정이 틀린 것 같다 → H_0 을 기각한다.
- "If the p-value is low, H_0 must go!" (p 값이 낮으면, H_0 은 꺼져라!)

8.2 순열 검정 (Permutation Test)

왜 필요한가? 고전적인 t-검정은 데이터가 정규성, 등분산성 등의 가정을 만족해야 한다는 '수학적 짐(baggage)'을 가지고 있습니다. 만약 우리 데이터가 이 가정을 명백히 위반한다면 (예: 분산이 일정하지 않음), t-검정의 p-value를 신뢰할 수 없습니다.

순열 검정의 아이디어 (컴퓨터를 이용한 대안): 순열 검정은 H_0 이 사실이라는 가정(즉, X 와 Y 는 아무 관련이 없다)을 컴퓨터 시뮬레이션으로 구현합니다.

1. 관찰: X 와 Y 사이의 실제 기울기(예: $\hat{\beta}_1 = 0.58$)를 계산하여 저장합니다.

2. 가정 (H_0): X 와 Y 가 관련 없다면, Y 값들(price)을 무작위로 뒤섞어서(shuffle) X (sqft)와 다시 짹지 어도 상관없을 것입니다.

3. 시뮬레이션:

- Y 값을 무작위로 섞은 $Y_{permute}$ 를 만듭니다.
- 이 $Y_{permute}$ 와 X 사이의 기울기 $\hat{\beta}_{permute}$ 를 계산합니다. (이 값은 H_0 이 사실일 때의 기울기 샘플입니다.)
- 이 과정을 1000번 (또는 10000번) 반복합니다.

4. p-value 계산: 1000개의 $\hat{\beta}_{permute}$ 값들(H_0 분포) 중에서, 우리가 처음에 관찰한 실제 기울기(0.58) 보다 더 극단적인(절대값이 큰) 값의 비율을 계산합니다.

5. 결론: 만약 이 비율(p-value)이 0.05보다 작으면, H_0 을 기각합니다.

8.3 부트스트랩 vs. 순열 검정

두 기법 모두 데이터를 재추출(resampling) 하지만, 목적과 방식이 완전히 다릅니다.

특징	부트스트랩(Bootstrap)	순열 검정(Permutation Test)
목표	추정(Estimation)	가설 검정(Hypothesis Testing)
질문	“내 통계량($\hat{\beta}_1$)이 얼마나 불확실한가?”	“ H_0 이 사실이라는 가정 하에 내 $\hat{\beta}_1$ 이 흔한 값인가?”
결과물	신뢰 구간(Confidence Interval)	p-value
가정	H_A (대립가설)을 가정. (관찰된 데이터가 모집단을 대표한다고 믿음)	H_0 (귀무가설)을 가정. (X, Y 는 관련 없음)
방법	복원 추출(Sampling with replacement) (데이터셋에서 (x_i, y_i) 쌍을 그대로 뽑음)	비복원 샘플링(Sampling without replacement) (Y 라벨만 뒤섞음)

Table 3: 부트스트랩과 순열 검정의 비교

8.4 신뢰 구간 vs. 예측 구간

모델의 불확실성을 표현하는 두 가지 다른 '구간'입니다.

특징	신뢰 구간(Confidence Interval, CI)	예측 구간(Prediction Interval, PI)
질문	“특정 x_0 에서 평균 반응값 $E[Y x_0]$ 이 어디쯤 있을까?”	“특정 x_0 에서 새로운 데이터 1개 y_{new} 가 어디쯤 있을까?”
의미	모델(회귀선) 자체의 불확실성 (데이터를 다시 뽑으면 선이 얼마나 바뀔까?)	모델 불확실성 + 데이터 고유의 노이즈(ϵ) (같은 x_0 라도 Y 는 원래 흩어져 있음)
폭	좁다(N이 커지면 0에 수렴)	항상 더 넓다(N이 커져도 ϵ 의 불확실성은 남음)

Table 4: 신뢰 구간(CI)과 예측 구간(PI)의 비교

9 중간고사 대비 체크리스트

중간고사 준비: 자가 점검표

중간고사 시험 범위(오늘 강의까지)를 정확히 알고 있는가?

중간고사가 '필기 시험(In-Class)'과 '코딩 시험(Take-home)'으로 나뉘는 것을 이해했는가?

치트 시트 2장(양면)을 준비하기 시작했는가?

Homework 3를 (마감일과 상관없이) 시험 공부 목적으로 미리 풀어보고 있는가?

베이즈 MCMC의 '목적' (왜 공식을 안 쓰고 시뮬레이션 하는지)을 설명할 수 있는가?

'N이 큰' 문제(계산 속도)와 'P가 큰' 문제(과적합, 차원의 저주)를 구분할 수 있는가?

'차원의 저주'를 "데이터가 희소해지고(sparse) 이웃이 멀어진다"고 설명할 수 있는가?

PCA의 핵심 아이디어가 "데이터 분산이 최대가 되는 새 축을 찾는 것"임을 아는가?

PCA가 수학적으로 '공분산 행렬의 고유벡터'를 찾는 것과 같음을 이해하는가?

PCA를 언제 사용하는가? (1. 시각화, 2. 회귀 분석(PCR))

PCR에서 사용할 주성분의 개수(m)를 정하는 3가지 방법(Elbow, Variance, CV)을 아는가?

PCA의 가장 큰 단점이 '해석력 상실'과 'Y를 무시'하는 것임을 아는가?

가설 검정의 5단계를 말할 수 있는가? (H_0 설정, 통계량, 계산, p-value, 결론)

p-value를 " H_0 이 사실일 때, 관찰값보다 극단적인 값이 나올 확률"이라고 정의할 수 있는가?

t-검정의 가정이 깨졌을 때 '순열 검정'을 사용할 수 있음을 아는가?

부트스트랩(복원추출, 추정)과 순열 검정(셔플링, 검정)의 차이를 설명 할 수 있는가?

신뢰 구간(평균의 불확실성)과 예측 구간(새 데이터 1개의 불확실성)을 구분할 수 있는가?

10 초심자를 위한 FAQ

Q: PCA는 지도 학습인가요, 비지도 학습인가요? **A:** 완벽한 비지도 학습(Unsupervised Learning)입니다. PCA는 차원을 축소하기 위해 오직 예측 변수(X)의 정보(분산, 공분산)만을 사용합니다. 반응 변수(Y , 라벨)는 PCA 계산 과정에서 전혀 고려되지 않습니다.

Q: 주성분(PC)은 원본 변수와 다른가요? **A:** 완전히 다릅니다. PC1(제1 주성분)은 $Z_1 = w_{11}X_1 + w_{12}X_2 + \dots + w_{1p}X_p$ 처럼 모든 원본 변수(X_1 부터 X_p 까지)가 조금씩 섞인 새로운 변수입니다. 원본 변수 중 하나(예: X_1)를 선택하는 것(Feature Selection)과는 근본적으로 다릅니다.

Q: PC1이 항상 Y를 가장 잘 예측하나요? **A:** 절대 아닙니다. PC1은 X 의 '분산'을 가장 잘 설명할 뿐입니다. Y 를 예측하는 데 가장 중요한 정보가 X 분산의 1%만 설명하는 PC10에 들어 있을 수도 있습니다. 이것이 PCA의 한계입니다. (이와 달리 Y 정보까지 고려하여 축을 찾는 것을 '부분 최소 제곱, PLS'라고 부릅니다.)

Q: t-검정과 순열 검정 중 무엇을 써야 하나요? **A:** 데이터의 가정을 먼저 확인해야 합니다.

- **t-검정:** 데이터가 (특히 잔차가) 정규성을 따르고 등분산성을 만족하는 등, 고전적 통계 가정을 잘 만족할 때 사용합니다. 더 적은 계산으로 강력한 결과를 줍니다.
- **순열 검정:** 데이터가 정규성/등분산성 가정을 만족하지 않을 때 사용하는 '비모수적(non-parametric)' 대안입니다. 수학적 가정 대신 컴퓨터의 계산 능력에 의존합니다.

11 빠르게 훑어보기 (1-Page Summary)

베이즈 시뮬레이션 (MCMC)

- **Why?** 모델이 복잡해지면 '후험 분포'를 수학 공식으로 풀 수 없다.
- **What?** 공식 대신, 분포에서 수천 개의 '샘플'을 뽑아 분포의 모양을 근사한다.
- **How?** 메트로폴리스-헤이스팅스 (안대 쓴 등산가 비유: 높은 곳(확률)에서 더 많은 시간을 보냄), 김스 샘플링 등
- **Use?** 샘플의 평균 (\rightarrow 후험 평균), 샘플의 백분위수 (\rightarrow 신뢰 구간)를 계산한다.

고차원성의 문제 (P is Big)

- **Problem?** 예측 변수(P)가 관측치(N)만큼 많거나 더 많은 상황.
- **Curse of Dimensionality:** 차원이 높아질수록 공간의 부피가 커져 데이터가 '희소(sparse)'해지고 모든 점이 서로 '멀어지는' 현상.
- **Result:** 과적합(Overfitting), 다중공선성, 모델 불안정.

주성분 분석 (PCA)

- **Goal:** 고차원(P 가 큼) 데이터의 '정보(분산)'를 최대한 보존하며 저차원으로 '압축'.
- **Idea:** 데이터의 분산이 '최대'가 되는 새 축(방향)을 찾는다 ($= Z_1, \text{PC1}$).
- **Math:** 공분산 행렬의 '고유벡터(Eigenvector)'가 새 축의 방향, '고유값(Eigenvalue)'이 그 축의 중요도(설명 분산)가 된다.
- **Usage 1 (Viz):** 고차원 데이터를 PC1, PC2의 2D 평면에 시각화 (Unsupervised).
- **Usage 2 (PCR):** 상위 m 개의 PC (Z_1, \dots, Z_m)를 회귀 모델의 예측 변수로 사용.
- **Trade-off:** 과적합은 막지만, 모델의 '해석력'을 잃는다.

가설 검정 복습

- **p-value:** H_0 (효과 없음)이 사실일 때, 내 관찰값보다 더 극단적인 값이 나올 확률. (낮으면 H_0 기각)
- **Permutation Test:** H_0 을 시뮬레이션(Y 샘플링)하여 p-value를 계산. (t-검정 가정이 깨졌을 때 사용)
- **Bootstrap vs. Permutation:** 부트스트랩(복원추출)은 '추정'(CI)-용, 순열검정(샘플링)은 '검정'(p-value)-용.
- **CI vs. PI:** CI(신뢰 구간)는 '평균'의 불확실성(좁음). PI(예측 구간)는 '새 데이터 1개'의 불확실성(넓음).

December 10, 2025

- 강의명: CS109A: 데이터 과학 입문
- 주차: Lecture 13
- 교수명: Pavlos Protopapas, Kevin Rader, Chris Gumb
- 목적: Lecture 13의 핵심 개념 학습

Contents

1	복습: 선형 회귀와 통계적 추론 (Review)	4
1.1	가설 검정과 p-value	4
1.2	순열 검정 (Permutation Test)	4
1.3	순열 검정 vs. 부트스트랩 (Permutation vs. Bootstrap)	5
1.4	상호작용 항 (Interaction Terms) 해석	5
1.5	신뢰 구간 vs. 예측 구간	5
2	분류 (Classification) 란 무엇인가?	6
2.1	회귀 vs. 분류 (Regression vs. Classification)	6
2.2	왜 선형 회귀를 분류 문제에 쓰면 안 되는가?	6
2.2.1	문제 1: 다중 클래스의 잘못된 순서 (False Ordering)	6
2.2.2	문제 2: 확률 범위를 벗어남 (Probability Bounds Violation)	6
3	로지스틱 회귀 (Logistic Regression)	8
3.1	핵심 아이디어: S-커브 (The S-Curve)	8
3.2	계수(Coefficient)의 해석: 오즈(Odds)와 로그-오즈(Log-Odds)	8
3.3	모델 추정: 최대가능도추정 (MLE)	9
4	다중 로지스틱 회귀와 결정 경계	11
4.1	다중 로지스틱 회귀 (Multiple Logistic Regression)	11
4.2	결정 경계 (Decision Boundaries)	11
4.3	결정 경계의 형태: 선형과 비선형	11
5	실습 코드 예제 (Python)	13
5.1	순열 검정 (Permutation Test) 예제 (NumPy)	13
5.2	상호작용 모델 (Statsmodels)	13
5.3	단순 로지스틱 회귀 (Scikit-learn)	14
5.4	비선형 결정을 위한 다항 로지스틱 회귀 (Scikit-learn)	14

개요 (Overview)

▣ 핵심 요약

이 문서는 데이터 사이언스의 핵심 주제인 '분류(Classification)'를 다룹니다.

지금까지 다룬 '회귀(Regression)'가 숫자(예: 주택 가격)를 예측하는 문제였다면, '분류'는 범주(예: 심장병 유무, 전공)를 예측하는 문제입니다.

이를 위해, 선형 회귀를 분류 문제에 바로 적용할 때 발생하는 문제점들을 살펴보고, 분류를 위한 핵심적인 파라메트릭 모델인 로지스틱 회귀(Logistic Regression)를 배웁니다.

주요 학습 목표:

- 회귀와 분류의 근본적인 차이를 설명할 수 있습니다.
- 왜 선형 회귀를 분류 문제에 사용하면 안 되는지 이해합니다.
- 로지스틱 회귀가 '확률'을 모델링하기 위해 시그모이드(Sigmoid) 함수를 사용하는 원리를 배웁니다.
- 로지스틱 회귀의 계수가 로그-오즈(Log-Odds) 관점에서 어떻게 해석되는지 설명할 수 있습니다.
- 로지스틱 회귀가 최대가능도추정(MLE)과 이진 교차 엔트로피(BCE)를 통해 어떻게 학습되는지 이해합니다.
- 모델의 결정 경계(Decision Boundary)가 어떻게 형성되는지 이해합니다.

또한, 이 주제에 앞서 중간고사에 포함될 수 있는 가설 검정, 순열 검정, 상호작용 항, 예측 구간 등에 대한 핵심 내용을 복습합니다.

주요 용어 정리 (Terminology)

본격적인 학습에 앞서, 오늘 다룰 핵심 용어들을 정리합니다.

Table 1: 분류 및 로지스틱 회귀 핵심 용어

용어 (Korean)	원어 (English)	쉬운 설명	비고
가설 검정	Hypothesis Testing	데이터가 특정 가설(주장)을 지지하는지 통계적으로 판단하는 과정.	예: $\beta_1 = 0$ 인가? 낮을수록(보통 < 0.05) 관계가 있다고 봄.
p-value	p-value	'귀무가설(예: 관계가 없다)'이 맞다고 할 때, 현재 데이터만큼 극단적인 결과가 우연히 나올 확률.	
순열 검정	Permutation Test	데이터의 라벨(Y)을 무작위로 섞어(순열), 우연만으로 원본 결과가 나오기 힘든 일인지 검증하는 기법.	
부트스트랩	Bootstrap	원본 데이터에서 중복을 허용하여(복원추출) 여러 번 샘플링하는 기법.	t -test의 가정(정규성 등)이 깨졌을 때 유용.
상호작용 항	Interaction Term	한 변수(X1)의 효과가 다른 변수(X2)의 수준에 따라 달라지는 효과를 나타내는 항.	주로 신뢰구간 '추정(Estimation)'에 사용.
예측 구간	Prediction Interval	새로운 개별 관측치(single point)가 존재할 것으로 예상되는 범위.	예: $soft: type$ 신뢰 구간(평균)보다 항상 넓음.
분류	Classification	데이터가 어떤 '범주(Category)'에 속하는지 예측하는 문제.	예: 스펙(I) vs. 정상(I)
회귀	Regression	데이터로부터 '연속적인 숫자(Number)'를 예측하는 문제.	예: 주택 가격 예측
시그모이드	Sigmoid Function	모든 입력을 0과 1 사이의 S자 곡선으로 매핑하는 함수.	로지스틱 함수의 별명.
로지스틱 회귀	Logistic Regression	시그모이드 함수를 사용해 데이터가 특정 범주(예: 1)에 속할 확률을 모델링하는 기법.	이름은 '회귀'지만 '분류' 모델임.
오즈	Odds	성공 확률(p)을 실패 확률(1-p)로 나눈 값. $(\frac{p}{1-p})$	$p=0.8$ 이면 Odds = 4 (성공이 4배)
로그-오즈	Log-Odds (Logit)	오즈에 자연로그(ln)를 취한 값. $\ln(\frac{p}{1-p})$	로지스틱 회귀는 로그-오즈를 선형 모델링함.
최대 가능도추정	MLE (Max Likelihood)	주어진 데이터가 관측될 '가능성(Likelihood)'을 최대로 만드는 모델 파라미터를 찾는 방법.	로지스틱 회귀의 핵심 원리.
이진 교차 엔트로피	BCE (Binary Cross-Entropy)	로지스틱 회귀의 손실 함수(Loss Function). (음의 로그 가능도)	BCE 를 최소화 = 가능도를 최대화.
결정 경계	Decision Boundary	모델이 클래스 0과 1을 구분하는 경계선. (즉, $P(Y=1) = 0.5$ 가 되는 지점)	기본은 선형, 다항식 항 추가 시 곡선 가능.

1 복습: 선형 회귀와 통계적 추론 (Review)

분류 모델을 배우기에 앞서, 선형 회귀 모델의 통계적 추론 방식을 복습합니다.

1.1 가설 검정과 p-value

- **가설 검정(Hypothesis Testing)**이란, 우리가 모델에서 발견한 관계(예: 주택 크기와 가격의 관계)가 '진짜'인지, 아니면 '단순한 우연'인지 통계적으로 판단하는 공식적인 절차입니다.
- **귀무가설(H_0)**: "관계가 없다." (즉, $\beta_1 = 0$ 이다.)
- **대립가설(H_A)**: "관계가 있다." (즉, $\beta_1 \neq 0$ 이다.)

이때 사용되는 핵심 도구가 **t-통계량(t-statistic)**과 **p-value**입니다.

- **t-통계량**: 우리가 추정한 계수($\hat{\beta}_1$)가 표준 오차(SE)에 비해 얼마나 큰지 나타내는 값입니다. (즉, 0에서 얼마나 멀리 떨어져 있는가?)

$$t = \frac{\hat{\beta}_1 - 0}{\text{SE}(\hat{\beta}_1)}$$

- **p-value**: 만약 귀무가설(H_0)이 사실(관계가 없음)이라면, 우리가 관찰한 t-통계량만큼 극단적인 값이 순전히 '우연'에 의해 관찰될 확률입니다.

□ 예제: title

p-value가 0.001이라는 것은, "만약 주택 크기와 가격이 아무 관계가 없다면, 우리가 현재 데이터에서 본 것과 같은 강한 관계가 우연히 나타날 확률이 0.1%밖에 되지 않는다"는 의미입니다. 이 확률이 매우 낮기(보통 0.05 미만), 우리는 "이건 우연이 아니다"라고 결론 내리고 귀무가설을 기각합니다. 즉, "주택 크기와 가격 사이에는 통계적으로 유의미한 관계가 있다"고 말합니다.

1.2 순열 검정(Permutation Test)

t-test는 데이터가 정규분포를 따르고, 분산이 동일하다(등분산성)는 가정이 필요합니다. 만약 데이터가 이 가정을 만족하지 못하면(예: 데이터가 한쪽으로 몰려있음), t-test의 p-value를 신뢰할 수 없습니다.

순열 검정은 이러한 가정 없이 p-value를 계산하는 강력한 재표본추출(Resampling) 기법입니다.

- **핵심 아이디어**: 귀무가설(H_0)이 "X와 Y는 관계가 없다"는 것이므로, 이 가설을 시뮬레이션하기 위해 Y값(예: 주택 가격)을 무작위로 뒤섞어버립니다(shuffling).
- 이렇게 하면 X와 Y 사이의 실제 관계가 인위적으로 파괴됩니다.
- **절차**:
 1. 원본 데이터에서 t-통계량(또는 $\hat{\beta}_1$ 값)을 계산합니다. (예: $\hat{\beta}_1 = 0.5898$)
 2. Y 값을 무작위로 섞은 후, X와 다시 짹지어 모델을 적합하고 $\hat{\beta}_1^*$ 값을 계산합니다. (당연히 0에 가까운 값이 나올 것입니다.)
 3. 이 과정을 1,000번 (또는 10,000번) 반복하여, '관계가 없을 때' 나올 수 있는 $\hat{\beta}_1^*$ 값들의 분포(귀무 분포)를 만듭니다.
 4. 원본 값(0.5898)이 이 귀무 분포(대부분 0 근처)에서 얼마나 극단적인 위치에 있는지 확인하여 p-value를 계산합니다.

1.3 순열 검정 vs. 부트스트랩 (Permutation vs. Bootstrap)

두 기법 모두 데이터를 재표본추출하지만, 목적과 방식이 다릅니다.

Table 2: 순열 검정과 부트스트랩 비교

특징	부트스트랩 (Bootstrap)	순열 검정 (Permutation Test)
목적	추정 (Estimation)	가설 검정 (Hypothesis Testing)
주요 산출물	신뢰 구간 (Confidence Interval)	p-value
샘플링 방식	복원 추출 (With Replacement)	비복원 추출 (Shuffling)
기본 가정	원본 데이터가 모집단을 잘 대표함	귀무가설 (H_0)이 참 (X-Y 관계 없음)

1.4 상호작용 항 (Interaction Terms) 해석

상호작용 항은 ” X_1 의 효과가 X_2 의 수준에 따라 달라지는” 효과를 모델링합니다. 예를 들어, $\text{price} \sim \text{sqft} + \text{type} + \text{sqft} : \text{type}$ 모델을 살펴봅니다. (여기서 type 의 기준(reference) 범주는 ’Condo’입니다.)

$$\text{price} = \beta_0 + \beta_1 \cdot \text{sqft} + \beta_2 \cdot \text{type[multifamily]} + \beta_3 \cdot (\text{sqft} \times \text{type[multifamily]}) + \dots$$

- $\hat{\beta}_1$ (예: 0.6659): 기준 범주(Condo)의 1 평방 피트당 가격 효과입니다.
- $\hat{\beta}_3$ (예: -0.2863): Multifamily의 1 평방 피트당 가격 효과가 Condo에 비해 얼마나 다른지(차이)를 나타냅니다.
- Multifamily의 실제 평방 피트당 가격 효과는 $\hat{\beta}_1 + \hat{\beta}_3$ (즉, $0.6659 - 0.2863$)입니다.
- 이 상호작용 항의 p-value가 유의미하다면(예: < 0.05), ”평방 피트에 따른 가격 변화율이 주택 유형 (Condo vs. Multifamily)에 따라 통계적으로 유의미하게 다르다”고 결론 내릴 수 있습니다.

1.5 신뢰 구간 vs. 예측 구간

- **신뢰 구간 (Confidence Interval):** (더 좁은 구간)
 - ”특정 X 값에 대한 평균 \bar{Y} 값 (\hat{Y})이 존재할 범위”에 대한 구간입니다.
 - 즉, ”우리의 회귀선 자체가 얼마나 정확한가”를 보여줍니다.
 - 데이터가 많아질수록 0에 가깝게 좁아질 수 있습니다.
- **예측 구간 (Prediction Interval):** (더 넓은 구간)
 - ”특정 X 값에 대한 새로운 개별 Y 값 (single new observation)이 존재할 범위”에 대한 구간입니다.
 - 이는 회귀선의 불확실성뿐만 아니라, 데이터 고유의 노이즈(줄일 수 없는 오차, ϵ)까지 포함합니다.
 - 데이터가 무한히 많아져도, 이 고유의 노이즈 때문에 일정 수준 이하로 좁아지지 않습니다.

2 분류 (Classification) 란 무엇인가?

2.1 회귀 vs. 분류 (Regression vs. Classification)

지금까지 우리가 다룬 문제는 대부분 회귀(Regression)였습니다.

- **회귀 (Regression):** 예측하려는 값(Y)이 연속적인 숫자(quantitative)입니다.
 - 예: 내일의 기온(25.5도), 주택 가격(\$500,000), 광고비 대비 매출액(\$18.5)
- **분류 (Classification):** 예측하려는 값(Y)이 범주형(qualitative, categorical)입니다.
 - 예: 내일 날씨(맑음, 흐림, 비), 환자의 심장병 유무(Yes, No), 학생의 전공(CS, Stats, Other)

□ 예제: title

- 회귀 질문: ”이 학생의 최고 심박수는 몇입니다?” (예측: 150 bpm)
- 분류 질문: ”이 학생은 심장병이 있습니까, 없습니까?” (예측: Yes)

2.2 왜 선형 회귀를 분류 문제에 쓰면 안 되는가?

Y 값이 범주형일 때, 선형 회귀($Y = \beta_0 + \beta_1 X$)를 그냥 사용하면 두 가지 심각한 문제가 발생합니다.

2.2.1 문제 1: 다중 클래스의 잘못된 순서 (False Ordering)

Y 값이 3개 이상의 범주(multi-class)를 가질 때를 생각해봅시다. (예: 전공) 우리가 이 범주를 숫자로 강제 인코딩했다고 가정합니다. ($Y = 1$ if CS, $Y = 2$ if Statistics, $Y = 3$ if Otherwise)

주의사항

선형 회귀는 이 숫자들 사이에 수학적인 관계가 있다고 가정합니다.

- 모델은 'CS(1)에서 Stats(2)로의 변화' (+1)과 'Stats(2)에서 Other(3)로의 변화' (+1)를 동일한 크기의 변화로 취급합니다.
- 이는 완전히 무의미한 가정입니다. 만약 'CS=3, Stats=1'로 순서를 바꾸면 모델의 결과가 완전히 달라집니다.

범주형 변수에는 자연스러운 순서나 등간격이 없습니다. (이러한 변수를 *nominal*하다고 합니다.)

2.2.2 문제 2: 확률 범위를 벗어남 (Probability Bounds Violation)

Y 값이 2개의 범주(binary)만 가질 때(예: 심장병 Yes=1, No=0)는 순서 문제는 없지만, 더 심각한 문제가 발생합니다.

이때 선형 회귀는 $P(Y = 1)$ (즉, 심장병에 걸릴 '확률')을 예측하도록 학습될 수 있습니다. 하지만 확률은 반드시 0과 1 사이의 값이어야 합니다.

주의사항

선형 회귀의 예측값(\hat{Y})은 직선이므로, 범위의 제한이 없습니다.

- X 가 매우 작으면(예: MaxHR이 매우 낮음), 모델이 $P(Y = 1) = 1.1$ (110%)과 같이 1보다 큰 값을 예측할 수 있습니다.

- X가 매우 크면(예: MaxHR이 매우 높음), 모델이 $P(Y = 1) = -0.1$ (-10%) 과 같이 0보다 작은 값을 예측할 수 있습니다.
이는 수학적으로, 논리적으로 완전히 잘못된 예측입니다.

(참고: 강의 슬라이드 30페이지의 그림은 선형 회귀선이 $Y=0$ 과 $Y=1$ 데이터를 벗어나 각각 0 미만, 1 초과의 확률을 예측하는 문제점을 시각적으로 보여줍니다.)

3 로지스틱 회귀 (Logistic Regression)

3.1 핵심 아이디어: S-커브 (The S-Curve)

선형 회귀의 문제(0~1 범위를 벗어남)를 해결하기 위해, 우리는 예측값이 항상 0과 1 사이에 머무르도록 하는 새로운 함수가 필요합니다.

- (1) 한 줄 핵심 요약: 모든 입력을 0과 1 사이의 S자 곡선으로 '압축' 시키는 시그모이드(Sigmoid) 함수를 사용해 확률을 모델링합니다.
- (2) 직관적 예시: 선형 회귀의 무한한 직선(h)을 가져와서, 이 직선을 양쪽 끝에서 눌러 0이라는 '바닥'과 1이라는 '천장'에 당도록 찌그러뜨린 모양을 상상하면 됩니다.
- (3) 기술적 설명:
 - 먼저, 선형 회귀와 똑같은 부분(h)을 계산합니다: $h = \beta_0 + \beta_1 X$
 - 이 h 값을 시그모이드(로지스틱) 함수에 통과시켜 확률 p 를 얻습니다.

$$p = P(Y = 1) = \frac{1}{1 + e^{-h}} = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X)}}$$

이 함수는 h 값에 따라 항상 0과 1 사이의 값을 반환합니다.

- $h \rightarrow +\infty$ (아주 큰 양수) 이면, $e^{-h} \rightarrow 0$, 따라서 $p \rightarrow \frac{1}{1+0} = 1$
- $h \rightarrow -\infty$ (아주 큰 음수) 이면, $e^{-h} \rightarrow \infty$, 따라서 $p \rightarrow \frac{1}{1+\infty} = 0$
- $h = 0$ 이면, $e^0 = 1$, 따라서 $p \rightarrow \frac{1}{1+1} = 0.5$

로지스틱 회귀의 이름

이름은 '회귀(Regression)'이지만, 하는 일은 '분류(Classification)'입니다. 이는 모델이 확률이라는 '연속적인 숫자'를 예측한 뒤, 그 확률을 기준으로 '범주'를 결정하기 때문입니다.

3.2 계수(Coefficient)의 해석: 오즈(Odds)와 로그-오즈(Log-Odds)

$p = \frac{1}{1+e^{-h}}$ 공식은 β_1 을 해석하기 매우 어렵습니다. "X가 1 증가할 때, p 는 $\frac{1}{1+e^{-(\beta_0 + \beta_1(X+1))}}$... 만큼 변한다"는 식은 직관적이지 않습니다.

대신, 위 공식을 h 에 대해 정리하면(즉, 역함수를 구하면) 훨씬 강력한 해석이 가능해집니다.

$$\ln \left(\frac{p}{1-p} \right) = \beta_0 + \beta_1 X$$

이 방정식의 좌변을 이해하기 위해 두 가지 개념을 도입합니다.

- **오즈 (Odds):** (성공 확률) / (실패 확률)

$$\text{Odds} = \frac{p}{1-p}$$

- $p = 0.5$ (확률 50%) \Rightarrow Odds = 1 (1:1)
- $p = 0.8$ (확률 80%) \Rightarrow Odds = 4 (실패보다 성공이 4배 높음)
- $p = 0.2$ (확률 20%) \Rightarrow Odds = 0.25 (성공보다 실패가 4배 높음)

- 로그-오즈 (Log-Odds) 또는 로짓(Logit): 오즈에 자연로그(ln)를 취한 값.

$$\text{Logit}(p) = \ln(\text{Odds}) = \ln\left(\frac{p}{1-p}\right)$$

▣ 핵심 요약

로지스틱 회귀의 핵심 해석: 로지스틱 회귀는 로그-오즈(Log-Odds)를 선형 회귀로 모델링하는 것입니다!

"X가 1단위 증가할 때, 로그-오즈가 β_1 만큼 더하기(additive)로 변한다."

이것은 "X가 1단위 증가할 때, 오즈(Odds)가 e^{β_1} 만큼 곱하기(multiplicative)로 변한다."는 의미와 같습니다.

▣ 예제: title

심장병 예측 모델에서 $X = \text{MaxHR}$ (최대 심박수)에 대한 계수가 $\hat{\beta}_1 = -0.0434$ 라고 가정합니다.

- 로그-오즈 해석 (어려움): 최대 심박수가 1 증가할 때마다, 심장병에 걸릴 로그-오즈가 -0.0434 만큼 감소합니다.
- 오즈 해석 (쉬움): $e^{\beta_1} = e^{-0.0434} \approx 0.957$
이는 최대 심박수가 1 증가할 때마다, 심장병에 걸릴 오즈가 약 0.957배가 된다는 의미입니다. (즉, 약 4.3%씩 감소합니다.)
- 만약 $\hat{\beta}_1 = 0$ 이었다면? $e^0 = 1$ 이므로, 오즈가 1배 (변화 없음)가 됩니다. 즉, X와 Y는 관계가 없습니다.
- 만약 $\hat{\beta}_1 = 0.7$ 이었다면? $e^{0.7} \approx 2.01$ 이므로, 오즈가 약 2배 증가합니다.

3.3 모델 추정: 최대가능도추정 (MLE)

최적의 S-커브 (즉, 최적의 β_0, β_1)는 어떻게 찾을까요?

- 선형 회귀의 경우: 손실 함수인 MSE(평균 제곱 오차)를 최소화하는 β 값을 찾았습니다. (이는 Y가 정규분포를 따른다고 가정한 것과 같습니다.)
- 로지스틱 회귀의 경우: Y가 0 또는 1이므로, 베르누이 분포(Bernoulli Distribution) (동전 던지기)를 따른다고 가정합니다.

이때 사용하는 학습 원리가 최대가능도추정 (MLE, Maximum Likelihood Estimation)입니다.

- 가능도 (Likelihood): "현재 우리가 가정한 S-커브(모델)가, 지금 우리가 가진 데이터 ($Y=0$ 또는 1)를 만들어 냈을 총 확률"입니다.
- 모델이 예측한 확률이 p_i 일 때:
 - 실제 값이 $y_i = 1$ (성공)이면: 이 관측치의 가능도는 p_i
 - 실제 값이 $y_i = 0$ (실패)이면: 이 관측치의 가능도는 $1 - p_i$
- 이를 하나의 수식으로 표현하면 $L_i = p_i^{y_i} \cdot (1 - p_i)^{1-y_i}$ 입니다.
- 전체 가능도 (Total Likelihood): 모든 데이터가 독립이라고 가정하므로, 모든 관측치의 가능도를 곱합니다.

$$L(\beta) = \prod_{i=1}^n L_i = \prod_{i=1}^n p_i^{y_i} \cdot (1 - p_i)^{1-y_i}$$

- MLE의 목표: 이 $L(\beta)$ 값을 최대로 만드는 β (즉, β_0, β_1)를 찾는 것입니다.

손실 함수: 이진 교차 엔트로피 (BCE)

곱셈(\prod)은 미분하기 매우 어렵습니다. 따라서 계산을 쉽게 하기 위해 양변에 로그(log)를 취합니다. 이를 로그 가능도(Log-Likelihood)라고 합니다. (로그를 취해도 최대가 되는 지점은 변하지 않습니다.)

$$l(\beta) = \log L(\beta) = \sum_{i=1}^n [y_i \log(p_i) + (1 - y_i) \log(1 - p_i)]$$

컴퓨터는 보통 '최소화' 문제를 풁니다. 따라서 위 값에 마이너스(-)를 붙인 음의 로그 가능도 (Negative Log-Likelihood)를 최소화합니다.

$$\text{Loss} = -l(\beta) = -\sum_{i=1}^n [y_i \log(p_i) + (1 - y_i) \log(1 - p_i)]$$

이 손실 함수를 이진 교차 엔트로피 (Binary Cross-Entropy, BCE)라고 부릅니다. 선형 회귀가 MSE를 최소화하듯, 로지스틱 회귀는 BCE를 최소화합니다.

4 다중 로지스틱 회귀와 결정 경계

4.1 다중 로지스틱 회귀 (Multiple Logistic Regression)

선형 회귀를 다중 선형 회귀로 확장했듯이, 로지스틱 회귀도 여러 개의 예측 변수(X)를 사용하도록 쉽게 확장할 수 있습니다.

단순히 로그-오즈에 대한 선형 방정식을 확장하면 됩니다.

$$\ln \left(\frac{p}{1-p} \right) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p$$

- 해석: β_j 의 해석은 ”다른 모든 변수(X_k)가 일정하다고 가정할 때”라는 조건이 추가됩니다.
- 즉, e^{β_j} 는 다른 변수들이 고정된 상태에서 X_j 가 1단위 증가할 때 오즈(Odds)의 곱셈 변화량입니다.
- 다중 공선성, 과적합 등 다중 선형 회귀에서 발생했던 문제들이 여기서도 동일하게 발생하며, 정규화 (Ridge, Lasso) 등이 필요할 수 있습니다.

4.2 결정 경계 (Decision Boundaries)

로지스틱 회귀는 확률(p)을 반환합니다. 이를 ’분류’ (Yes/No)로 바꾸려면 결정 임계값(Threshold)이 필요합니다.

- 기본 임계값: $p \geq 0.5$ 이면 $Y = 1$ (Yes)로 분류, $p < 0.5$ 이면 $Y = 0$ (No)로 분류합니다.
- 결정 경계 (Decision Boundary): 모델이 Yes와 No를 구분하는 경계선, 즉 $p = 0.5$ 가 되는 지점입니다.

$p = 0.5$ 는 Odds = $\frac{0.5}{1-0.5} = 1$ 을 의미하고, Log-Odds = $\ln(1) = 0$ 을 의미합니다. 따라서 결정 경계는 로지스틱 회귀의 선형 방정식 부분이 0이 되는 지점입니다.

$$\text{결정 경계: } \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p = 0$$

4.3 결정 경계의 형태: 선형과 비선형

- 선형 경계 (Linear Boundary): 기본적인 다중 로지스틱 회귀 모델($\beta_0 + \beta_1 X_1 + \beta_2 X_2 = 0$)은 X_1 과 X_2 에 대한 1차 방정식이므로, 결정 경계는 항상 직선 (또는 3D에서는 평면)이 됩니다.

(참고: 강의 슬라이드 83페이지는 *MaxHR*과 *Chol* 변수만 사용했을 때, 두 클래스를 나누는 경계가 직선으로 나타나는 것을 보여줍니다.)

- 비선형 경계 (Non-linear Boundary): 하지만 데이터가 직선으로 잘 나뉘지 않는 경우가 많습니다.

(참고: 강의 슬라이드 84페이지는 두 클래스가 곡선 형태로 섞여 있어 직선 경계로는 잘 나눌 수 없는 예시를 보여줍니다.)

해결책: 선형 회귀에서 다항 회귀를 사용했듯이, 로지스틱 회귀에도 변수들을 변형하여 추가합니다.

1. 상호작용 항 추가: $X_3 = X_1 \cdot X_2$

2. 다항 항 추가: $X_4 = X_1^2, X_5 = X_2^2$

이렇게 변형된 변수들로 모델을 만들면,

$$\ln \left(\frac{p}{1-p} \right) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 (X_1 \cdot X_2) + \beta_4 X_1^2 + \beta_5 X_2^2$$

결정 경계 ($= 0$)는 X_1, X_2 에 대한 2차 방정식이 되므로, 곡선, 원, 타원 형태의 비선형 경계를 만들어낼 수 있습니다.

모델은 여전히 '선형'입니다

비선형 경계를 만들었음에도 불구하고, 이 모델은 여전히 '선형 모델'로 분류됩니다.

왜냐하면 모델은 계수(β)에 대해 선형이기 때문입니다. (X_1^2 를 그냥 Z_4 라는 새로운 변수로 보면, 모델은 $\beta_0 + \beta_1 Z_1 + \dots + \beta_5 Z_5$ 형태의 선형 결합입니다.)

우리는 입력 변수(X)를 비선형으로 변환(feature engineering)하여, 선형 모델로 비선형 경계를 찾도록 한 것입니다.

5 실습 코드 예제 (Python)

강의에서 사용된 statsmodels 및 scikit-learn 코드 예제입니다.

5.1 순열 검정 (Permutation Test) 예제 (Numpy)

```

1 import numpy as np
2 import sklearn.linear_model
3
4 nsims = 1000
5 X = homes[['sqft']]
6 y = homes[['price']]
7
8 indices = np.arange(0, len(homes))
9 beta1_permute = []
10
11 for i in np.arange(0, nsims):
12     # 1. 귀무가설을 시뮬레이션하기 위해의 Y 인덱스를 섞습니다 .
13     np.random.shuffle(indices)
14     y_permute = y.iloc[indices]
15
16     # 2. 섞인와 Y 원본로 X 모델을 적합합니다 .
17     permute_ols = sk.linear_model.LinearRegression().fit(X, y_permute)
18
19     # 3. 귀무가설하의기울기 (beta1)를 저장합니다.
20     beta1_permute.append(permute_ols.coef_[0][0])
21
22 # 의 beta1_permute 분포귀무 (분포)와
23 # 원본데이터의기울기 (beta1_observed)를 비교하여 p-를 value 계산합니다.

```

Listing 1: Numpy를 이용한 순열 검정 구현

5.2 상호작용 모델 (Statsmodels)

```

1 import statsmodels.formula.api as smf
2
3 # price ~ sqft + type + sqft:type 모델을 적합합니다 .
4 # 'type'은 범주형 변수로 자동 인식됩니다 .
5 interaction_ols = smf.ols(formula="price ~ sqft * type",
6                           data=homes).fit()
7
8 # 결과 요약 출력
9 print(interaction_ols.summary())
10
11 # coef
12 # -----
13 # Intercept           170.5182
14 # type[T.multipfamily] 142.0626
15 # type[T.singlefamily] -708.8103

```

```

16 # sqft                      0.6659 <-- Condo기준()의 sqft 기울기
17 # sqft:type[T.multipamily]   -0.2863 <-- Condo 대비의 multipamily 기울기차이 ''
18 # sqft:type[T.singlefamily]  0.4769 <-- Condo 대비의 singlefamily 기울기차
19   0| ''
# ...

```

Listing 2: Statsmodels를 이용한 상호작용 모델 적합

5.3 단순 로지스틱 회귀 (Scikit-learn)

```

1 from sklearn.linear_model import LogisticRegression
2
3 # X 예측(변수, 2D 배열이어야함)
4 X_hr = df_heart[['MaxHR']]
5 # Y 반응(변수, 1D 배열)
6 y_ahd = df_heart['AHD']
7
8 # penalty='none' : 정규화(Ridge/Lasso)를 사용하지않음
9 logreg = LogisticRegression(penalty='none')
10 logreg.fit(X_hr, y_ahd)
11
12 # beta_1 기울기()
13 print('Estimated beta1: \n', logreg.coef_)
14 # [[-0.04341112]]
15
16 # beta_0 절편()
17 print('Estimated beta0: \n', logreg.intercept_)
18 # [6.3249492]
19
20 # 모델: log(odds) = 6.325 - 0.0434 * MaxHR

```

Listing 3: Scikit-learn을 이용한 단순 로지스틱 회귀

5.4 비선형 결정을 위한 다항 로지스틱 회귀 (Scikit-learn)

```

1 # 1. 비선형특성생성
2 df_heart['Interaction'] = df_heart.MaxHR * df_heart.Chol
3 df_heart['MaxHR_sq'] = df_heart.MaxHR**2
4 df_heart['Chol_sq'] = df_heart.Chol**2
5
6 # 사용할변수리스트
7 features = ['MaxHR', 'Chol', 'Interaction', 'MaxHR_sq', 'Chol_sq']
8
9 data_x = df_heart[features]
10 data_y = df_heart['AHD']
11
12 # 2. 모델적합
13 logreg_poly = LogisticRegression(penalty='none', fit_intercept=True, max_iter=1000)
14 logreg_poly.fit(data_x, data_y)

```

```
15  
16 # 3. 계수확인  
17 print('Estimated betas: \n', logreg_poly.coef_)  
18 print('Estimated beta0: \n', logreg_poly.intercept_)  
19  
20 # 이모델의결정경계 (log_odds = 0)는 X1, 에 X2 대한차식이 2 되어  
21 # 원형또는타원형의곡선경계를생성합니다 .
```

Listing 4: 다항 및 상호작용 항을 사용한 로지스틱 회귀

빠르게 훑어보기 (1-Page Summary)

분류(Classification)와 로지스틱 회귀 핵심 요약

1. 문제 정의: 회귀 vs. 분류

- 회귀 (Regression): 숫자 예측 (예: 가격, 온도). Tool: 선형 회귀
- 분류 (Classification): 범주 예측 (예: Yes/No, 스팸/정상). Tool: 로지스틱 회귀

2. 왜 선형 회귀는 분류에 실패하는가?

- 이유 1 (다중 클래스): 1=CS, 2=Stats 처럼 강제 인코딩 시, 무의미한 순서와 간격을 가정하게 됨.
- 이유 2 (이진 클래스): 예측 값이 확률의 범위 [0, 1]을 벗어남 (예: 110% 또는 -10%).

3. 해결책: 로지스틱 회귀와 시그모이드

- 선형 회귀의 결과($h = \beta_0 + \beta_1 X$)를 시그모이드(Sigmoid) 함수에 넣어 0 1 사이의 확률 값(p)으로 '압축' 시킴.

$$p = P(Y = 1) = \frac{1}{1 + e^{-h}}$$

4. 핵심 해석: 로그-오즈 (Log-Odds)

- 모델의 공식을 변형하면 로그-오즈(Logit)가 X 에 대한 선형 함수임을 알 수 있음.

$$\ln\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 X$$

- β_1 의 해석: X 가 1 증가할 때, 오즈(Odds)가 e^{β_1} 배가 된다.
- $e^{\beta_1} > 1$ (긍정적 관계), $e^{\beta_1} = 1$ (관계 없음), $e^{\beta_1} < 1$ (부정적 관계)

5. 학습 원리: MLE와 BCE

- 가정: Y 는 베르누이 분포를 따름 (동전 던지기).
- 목표: 관측된 데이터가 나타날 가능성(Likelihood)을 최대로 만드는 β 를 찾음 (MLE).
- 손실 함수: 이진 교차 엔트로피(BCE) (음의 로그 가능성)를 최소화함.

6. 결정 경계 (Decision Boundary)

- 모델이 0과 1을 나누는 경계선. (즉, $p = 0.5$ 또는 Log-Odds = 0이 되는 지점)
- 기본 모델: $\beta_0 + \beta_1 X_1 + \beta_2 X_2 = 0 \implies$ 직선 경계.
- 다항/상호작용 모델: $\beta_0 + \dots + \beta_4 X_1^2 + \beta_5 X_2^2 = 0 \implies$ 곡선 경계.

December 10, 2025

- 강의명: CS109A: 데이터 과학 입문
- 주차: Lecture 14
- 교수명: Pavlos Protopapas, Kevin Rader, Chris Gumb
- 목적: Lecture 14의 핵심 개념 학습

▣ 핵심 요약

본 문서는 로지스틱 회귀의 심화 주제를 다룹니다. 단순 모델을 넘어 다중 로지스틱 회귀, 상호작용 항의 해석, 정규화(Ridge)를 통한 과적합 방지 방법을 배웁니다. 또한, 로지스틱 회귀를 분류(Classification) 문제에 활용하는 방법, 즉 결정 경계(Decision Boundary)의 개념과 다중 클래스($K > 2$) 분류를 위한 OvR, 다항 로지스틱 회귀(Softmax)를 학습합니다. 마지막으로, 분류 모델의 성능을 평가하는 핵심 지표인 혼동 행렬(Confusion Matrix), 민감도, 특이도, ROC 커브, AUC의 개념을 상세히 설명합니다.

Contents

1	로지스틱 회귀 복습 (Review)	2
1.1	왜 로지스틱 회귀인가?	2
1.2	핵심 아이디어: 확률을 직접 모델링하지 않는다	2
1.3	추정: 최대가능도 추정법 (MLE)	2
2	로지스틱 회귀의 추론 (Inference)	4
2.1	선형 회귀(t-분포) vs 로지스틱 회귀(Z-분포)	4
2.2	계수(Coefficient) 해석하기	4
3	다중 로지스틱 회귀와 상호작용	6
3.1	다중 로지스틱 회귀 (Multiple Logistic Regression)	6
3.2	상호작용 (Interactions)	6
4	분류와 결정 경계 (Classification & Decision Boundary)	7
4.1	확률에서 분류로: 임계값 (Threshold)	7
4.2	결정 경계 (Decision Boundary)	7
5	정규화 (Regularization)	8

5.1	손실 함수 + L2 (Ridge) 페널티	8
6	다중 클래스 로지스틱 회귀 (Multiclass)	9
6.1	접근법 1: One-vs-Rest (OvR)	9
6.2	접근법 2: 다항 로지스틱 회귀 (Multinomial)	9
6.3	Softmax: 점수를 확률로 변환하기	9
6.4	다중 클래스 분류 ($K > 2$)	9
7	분류 모델 평가 (Evaluation)	10
7.1	혼동 행렬 (Confusion Matrix)	10
7.2	핵심 평가지표	10
7.3	임계 값(Threshold)의 트레이드오프	12
7.4	ROC 커브와 AUC	12
8	핵심 용어 정리	13
9	학습 체크리스트	14
10	1페이지 요약 (1-Page Summary)	15

1 로지스틱 회귀 복습 (Review)

1.1 왜 로지스틱 회귀인가?

선형 회귀(Linear Regression)는 예측 값이 연속적인 숫자(예: 집값, 온도)일 때 사용합니다. 하지만 우리가 예측하려는 대상(Y)이 '성공/실패', '합격/불합격', '생존/사망'처럼 두 가지 범주 중 하나라면 (Binary) 선형 회귀는 적합하지 않습니다.

로지스틱 회귀(Logistic Regression)는 Y 가 범주형일 때, 특히 이진(binary) 분류 문제에서 사용됩니다.

주의사항

오해 피하기: 선형 회귀를 이진 분류에 쓰면 안 되는 이유

선형 회귀 모델($Y = \beta_0 + \beta_1 X$)을 그대로 사용하면 두 가지 큰 문제가 발생합니다.

- **범위 초과:** Y 는 0 또는 1이어야 하지만, 선형 회귀의 예측 값은 1을 넘거나 0보다 작아질 수 있습니다. 이는 '확률'로 해석할 수 없게 만듭니다.
- **관계 왜곡:** 0과 1 사이의 관계가 직선적(linear)이라고 가정하지만, 실제로는 특정 지점에서 급격히 변하는 S자 형태(비선형)일 가능성이 높습니다.

1.2 핵심 아이디어: 확률을 직접 모델링하지 않는다

로지스틱 회귀는 $P(Y = 1|X)$ (성공 확률)을 직접 모델링하는 대신, '확률'을 변형한 **로그-오즈(Log-Odds)**를 선형 회귀 형태로 모델링합니다.

1. **확률 (Probability, P):** 0과 1 사이의 값. (예: $P = 0.8$)
2. **오즈 (Odds):** 성공 확률 / 실패 확률. (예: $Odds = 0.8/(1 - 0.8) = 4$). 0부터 무한대(∞)까지의 값을 가집니다. "실패보다 성공할 확률이 4배 높다"는 의미입니다.

$$\text{Odds} = \frac{P(Y = 1)}{1 - P(Y = 1)} = \frac{P}{1 - P}$$

3. **로그-오즈 (Log-Odds) 또는 로짓(Logit):** 오즈에 자연로그(ln)를 취한 값. (예: $\ln(4) \approx 1.386$). 음의 무한대($-\infty$)부터 양의 무한대($+\infty$)까지 모든 값을 가질 수 있습니다.

$$\text{Logit}(P) = \ln(\text{Odds}) = \ln\left(\frac{P}{1 - P}\right)$$

로그-오즈는 선형 회귀의 예측 값처럼 범위에 제한이 없으므로, 이를 X 에 대한 선형 결합으로 모델링 할 수 있습니다.

$$\underbrace{\ln\left(\frac{P}{1 - P}\right)}_{\text{Log-Odds}} = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p$$

1.3 추정: 최대가능도 추정법 (MLE)

선형 회귀는 β 를 찾기 위해 오차제곱합(SSE)을 최소화했습니다 (최소제곱법, OLS).

로지스틱 회귀는 **최대가능도 추정법 (Maximum Likelihood Estimation, MLE)**을 사용합니다. 직관

적으로, ”우리가 가진 데이터(Y 값들)가 관찰될 확률을 가장 높게 만드는 β 값을 찾자”는 의미입니다. 이는 수학적으로 **음의 로그-가능도(Negative Log-Likelihood)**를 최소화하는 것과 같으며, 이 손실 함수(Loss Function)를 **이진 교차 엔트로피(Binary Cross-Entropy)**라고 부릅니다.

$$\text{Loss (Binary Cross-Entropy)} = - \sum_{i=1}^n [y_i \ln(p_i) + (1 - y_i) \ln(1 - p_i)]$$

* y_i : 실제 값 (0 또는 1) * p_i : 모델이 예측한 $P(Y_i = 1)$ 확률

선형 회귀와 달리 한 번에 풀리는 해(Closed-form solution)가 없으며, **경사 하강법(Gradient Descent)**과 같은 수치 최적화(Numerical Optimization) 기법을 사용해 β 값을 반복적으로 찾아나갑니다.

2 로지스틱 회귀의 추론 (Inference)

추론(Inference)의 목적은 모델의 계수(β)가 통계적으로 유의미한지, 그리고 그 값의 신뢰구간(Confidence Interval)이 어느 정도인지 파악하는 것입니다.

2.1 선형 회귀(t-분포) vs 로지스틱 회귀(Z-분포)

- **선형 회귀:** 오차항의 분산(σ^2)을 별도로 '추정'해야 합니다. 이 불확실성 때문에 β 의 분포는 t -분포를 따릅니다.
- **로지스틱 회귀:** Y 가 베르누이 분포($Y \sim \text{Bernoulli}(P)$)를 따릅니다. 베르누이 분포의 분산은 $P(1 - P)$ 로, 평균(P)이 정해지면 분산이 '공짜로' 결정됩니다. 별도로 추정할 분산이 없으므로, (샘플이 충분히 크다면) β 는 정규분포(Z-분포)를 따른다고 가정합니다.

□ 예제:

Z-분포 vs t-분포의 실제적 차이

95% 신뢰구간을 계산할 때,

- **Z-분포 (로지스틱):** $\hat{\beta} \pm 1.96 \times (\text{표준오차})$
- **t-분포 (선형):** $\hat{\beta} \pm t_{\alpha/2, df} \times (\text{표준오차})$ (보통 2에 가까운 값)

실제 계산에서는 큰 차이가 없으나, 통계적 근거가 다릅니다. 'statsmodels' 라이브러리는 이러한 Z-통계량, p-value, 신뢰구간을 제공해줍니다.

2.2 계수(Coefficient) 해석하기

β 값 자체보다 e^β (지수 변환) 값이 훨씬 직관적입니다.

- β_j : X_j 가 1단위 증가할 때, 로그-오즈(Log-Odds)가 β_j 만큼 증가(덧셈)합니다.
- e^{β_j} : X_j 가 1단위 증가할 때, 오즈(Odds)가 e^{β_j} 만큼 곱해집니다(배수).

이를 **오즈비 (Odds Ratio, OR)**라고 부릅니다. * $e^{\beta_j} > 1$: X_j 가 증가하면 성공 오즈가 증가합니다. (긍정적 관계) * $e^{\beta_j} = 1$: X_j 는 성공 오즈와 관계 없습니다. ($\beta_j = 0$) * $e^{\beta_j} < 1$: X_j 가 증가하면 성공 오즈가 감소합니다. (부정적 관계)

□ 예제:

예: 이진 예측변수 (Binary Predictor) 해석 (성별과 심장병)

심장병 발병($Y = 1$) 여부를 성별로 예측하는 모델을 가정합니다. (기준 그룹: 남성)

$$\ln(\text{Odds}) = \beta_0 + \beta_1 \cdot \text{Female} \quad (\text{Female} = 1, \text{Male} = 0)$$

여기서 $\beta_0 = 0.214$ 이고 $\beta_1 = -1.272$ 라고 가정합시다.

1. β_0 의 해석 (기준 그룹):

- $\beta_0 = 0.214$ 는 남성(기준 그룹)의 로그-오즈입니다.
- $e^{\beta_0} = e^{0.214} \approx 1.24$ 는 남성의 오즈입니다.
- (심장병에 걸릴 오즈가 안 걸릴 오즈보다 1.24배 높다.)

2. β_1 의 해석 (차이):

- $\beta_1 = -1.272$ 는 여성의 로그-오즈가 남성보다 1.272만큼 낮다는 의미입니다.
- $e^{\beta_1} = e^{-1.272} \approx 0.28$ 은 오즈비(Odds Ratio)입니다.
- 해석: ”다른 조건이 같다면, 여성의 심장병 발병 오즈는 남성의 0.28배 (즉, 72% 낮다).”

주의사항

오즈(Odds)와 확률(Probability)을 혼동하지 마세요!

계수(β) 해석은 항상 오즈(Odds) 관점에서 이루어집니다. ”여성의 심장병 발병 확률이 72% 낮다”라고 해석하면 틀립니다. 확률로 변환하려면 $P = \text{Odds}/(1 + \text{Odds})$ 공식을 사용해야 하며, 이는 기준이 되는 확률(baseline probability)에 따라 그 변화량이 달라지는 비선형(non-linear) 관계입니다.

3 다중 로지스틱 회귀와 상호작용

3.1 다중 로지스틱 회귀 (Multiple Logistic Regression)

선형 회귀와 마찬가지로, 여러 개의 예측 변수(X_1, \dots, X_p)를 사용하여 모델을 구성할 수 있습니다.

$$\ln\left(\frac{P}{1-P}\right) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_p X_p$$

해석: β_j (또는 e^{β_j})의 의미는, ”다른 모든 예측 변수(X_k)를 통제(일정하게 유지)했을 때” X_j 가 1 단위 변할 때의 로그-오즈 (또는 오즈비) 변화량입니다.

주의점: 선형 회귀와 동일한 문제들, 즉 **다중공선성(Multicollinearity)**과 **과적합(Overfitting)**이 여기서도 동일하게 발생합니다.

3.2 상호작용 (Interactions)

상호작용 항은 ”한 변수(X_1)가 결과(Y)에 미치는 영향이 다른 변수(X_2)의 값에 따라 달라질 때” 사용됩니다.

□ 예제:

예: 상호작용 해석 (Age × Female)

심장병 모델에 나이(Age)와 성별(Female), 그리고 둘의 상호작용 항을 추가합니다.

$$\ln(\text{Odds}) = \beta_0 + \beta_1 \text{Age} + \beta_2 \text{Female} + \beta_3 (\text{Age} \times \text{Female})$$

이 모델은 성별에 따라 두 개의 다른 모델로 분리됩니다.

1. 남성 (Male, Female=0)의 모델: Female=0을 대입하면 β_2, β_3 항이 사라집니다.

$$\ln(\text{Odds})_{\text{Male}} = \beta_0 + \beta_1 \text{Age}$$

(절편: β_0 , 나이의 기울기: β_1)

2. 여성 (Female, Female=1)의 모델: Female=1을 대입합니다.

$$\ln(\text{Odds})_{\text{Female}} = \beta_0 + \beta_1 \text{Age} + \beta_2 (1) + \beta_3 (\text{Age} \times 1)$$

$$\ln(\text{Odds})_{\text{Female}} = (\beta_0 + \beta_2) + (\beta_1 + \beta_3) \text{Age}$$

(절편: $\beta_0 + \beta_2$, 나이의 기울기: $\beta_1 + \beta_3$)

계수 해석:

- β_1 : 남성(기준 그룹)의 나이 1살 증가에 따른 로그-오즈 변화량.
- β_3 : 여성의 나이 1살 증가에 따른 로그-오즈 변화량이 남성 대비 얼마나 다른지 (그 차이)
만약 β_3 가 0이라면 (상호작용이 없다면), 두 그룹의 나이 기울기는 β_1 로 동일할 것입니다.

4 분류와 결정 경계 (Classification & Decision Boundary)

4.1 확률에서 분류로: 임계값 (Threshold)

로지스틱 회귀는 $P(Y = 1|X)$ 확률을 예측합니다. 이를 0 또는 1의 분류로 바꾸려면 **'임계값(Threshold)''** (보통 0.5)을 정해야 합니다.

- $P(Y = 1) \geq 0.5$ 이면, $\hat{Y} = 1$ (성공)으로 분류한다.
- $P(Y = 1) < 0.5$ 이면, $\hat{Y} = 0$ (실패)로 분류한다.

4.2 결정 경계 (Decision Boundary)

결정 경계는 모델의 예측이 $\hat{Y} = 0$ 에서 $\hat{Y} = 1$ 로 바뀌는 지점, 즉 $P(Y = 1) = 0.5$ 가 되는 지점의 선 또는 면을 의미합니다.

$P = 0.5$ 라는 것은 어떤 의미일까요?

- $P = 0.5$
- Odds = $P/(1 - P) = 0.5/0.5 = 1$
- $\ln(\text{Odds}) = \ln(1) = 0$

즉, 결정 경계는 **로그-오즈가 0이 되는 지점**입니다.

$$\ln(\text{Odds}) = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p = 0$$

주의사항

오해 피하기: 결정 경계는 항상 선형인가?

결정 경계가 선형(직선, 평면)일 수도 있고, 비선형(곡선, 곡면)일 수도 있습니다. 이는 모델에 어떤 항을 포함했는지에 달려있습니다.

- **선형 경계:** 모델이 $\beta_0 + \beta_1 X_1 + \beta_2 X_2 = 0$ 처럼 X 의 1차항만 포함하면, 결정 경계는 X_1 과 X_2 공간에서 직선이 됩니다.
- **비선형 경계:** 모델이 $X_1^2, X_2^2, X_1 X_2$ 같은 다항식(Polynomial) 항이나 상호작용 항을 포함하면, (예: $\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1^2 + \beta_4 X_2^2 = 0$) 결정 경계는 X_1 과 X_2 공간에서 곡선(원, 타원 등)이 됩니다.

비선형 항을 추가함으로써 로지스틱 회귀는 복잡한 데이터 패턴도 분류할 수 있게 됩니다.

[이미지 삽입: 왼쪽은 직선으로 두 클래스를 나누는 결정 경계, 오른쪽은 곡선(원형)으로 두 클래스를 나누는 결정 경계를 보여줌]

5 정규화 (Regularization)

모델에 다항식 항이나 상호작용 항을 많이 추가하면 결정 경계가 매우 복잡해지면서 훈련 데이터에만 꼭 맞는 과적합(Overfitting)이 발생할 수 있습니다.

정규화(Regularization)는 모델의 복잡도에 폐널티를 부과하여 과적합을 방지하는 기법입니다. 계수 (β)의 크기가 너무 커지지 않도록 손실 함수(Loss Function)에 **폐널티 항(Penalty Term)**을 추가합니다.

5.1 손실 함수 + L2 (Ridge) 폐널티

로지스틱 회귀의 손실 함수(Binary Cross-Entropy)에 L2 폐널티(계수 제곱의 합, Ridge)를 더합니다.

$$\text{Loss}_{\text{Regularized}} = \underbrace{\text{Loss} (\text{Binary Cross-Entropy})}_{\text{모델이 데이터에 얼마나 잘 맞는지}} + \underbrace{\lambda \sum_{j=1}^p \beta_j^2}_{\text{모델이 얼마나 복잡한지 (폐널티)}}$$

- λ (람다): 정규화의 강도를 조절하는 하이퍼파라미터입니다.
 - $\lambda = 0$: 폐널티 없음 (표준 로지스틱 회귀).
 - $\lambda \rightarrow \infty$: 폐널티가 매우 강해져 모든 β 가 0에 가까워집니다 (모델이 매우 단순해짐).
- 폐널티는 보통 절편(β_0)을 제외하고 적용됩니다.

주의사항

sklearn의 ‘C’ 파라미터 이해하기

‘sklearn.linear_model.LogisticRegression’ λ 대신 C 라는 파라미터를 사용합니다. C 는 λ 의 역수 ($C = 1/\lambda$) 개념입니다.

- 높은 ‘C’ (예: $C = 100$) \Rightarrow 낮은 λ : 정규화(폐널티)가 약합니다. 모델이 복잡해지고 훈련 데이터에 더 강하게 맞춰집니다 (과적합 위험).
- 낮은 ‘C’ (예: $C = 0.01$) \Rightarrow 높은 λ : 정규화(폐널티)가 강합니다. 모델이 단순해지고 β 계수들이 0에 가까워집니다 (과소적합 위험).

최 적의 C 값은 교차 검증(Cross-Validation)을 통해 찾아야 합니다.

```

1 from sklearn.linear_model import LogisticRegression
2 from sklearn.model_selection import GridSearchCV
3
4 # C 값이 낮을수록 정규화가 강해짐
5 logreg = LogisticRegression(penalty='l2', C=0.1)
6
7 # 교차검증으로 최적의 C 값을 찾을 수도 있음
8 params = {'C': [0.01, 0.1, 1, 10, 100]}
9 grid_search = GridSearchCV(LogisticRegression(penalty='l2'), params, cv=5)
10 # grid_search.fit(X_train, y_train)
11 # print(grid_search.best_params_)

```

Listing 1: sklearn에서 Ridge 정규화를 적용한 로지스틱 회귀

6 다중 클래스 로지스틱 회귀 (Multiclass)

Y 의 범주가 3개 이상일 때 (예: 'CS', 'Stat', 'Other') 사용하는 방법입니다. 여기서는 순서가 없는 **명목형(Nominal)** 범주를 가정합니다.

6.1 접근법 1: One-vs-Rest (OvR)

가장 간단하고 직관적인 방법입니다. 범주가 K 개일 때, K 개의 독립적인 이진 분류기를 만듭니다.

- **분류기 1:** 'CS' vs 'Not CS' (즉, 'Stat' + 'Other')
- **분류기 2:** 'Stat' vs 'Not Stat' (즉, 'CS' + 'Other')
- **분류기 3:** 'Other' vs 'Not Other' (즉, 'CS' + 'Stat')

새로운 데이터가 들어오면, 3개의 분류기를 모두 돌려서 각각의 확률(또는 점수)을 계산한 뒤, 가장 높은 확률(점수)을 보인 클래스로 예측합니다. ('sklearn'의 'multiclass = 'ovr'')

6.2 접근법 2: 다항 로지스틱 회귀 (Multinomial)

OvR과 달리, K 개의 클래스를 한 번에 처리하는 단일 모델을 만듭니다. 하나의 클래스(예: 'Other')를 **기준(Reference) 클래스**로 정합니다. 그리고 $K - 1$ 개의 로그-오즈 모델을 만듭니다.

- **모델 1:** $\ln\left(\frac{P(\text{CS})}{P(\text{Other})}\right) = \beta_0^{(1)} + \beta_1^{(1)} X + \dots$
 - **모델 2:** $\ln\left(\frac{P(\text{Stat})}{P(\text{Other})}\right) = \beta_0^{(2)} + \beta_1^{(2)} X + \dots$
- ('sklearn'의 'multiclass = 'multinomial'')

6.3 Softmax: 점수를 확률로 변환하기

OvR이든 다항 로지스틱이든, 각 클래스 k 에 대한 '점수(Score)' 또는 '로짓(Logit)' s_k 가 나옵니다. 이 점수들은 합쳐도 1이 되지 않기 때문에 확률로 사용하기 어렵습니다.

소프트맥스(Softmax) 함수는 이 점수(s_k)들을 0과 1 사이의 값으로 변환하고, 모든 클래스의 확률 총합이 1이 되도록 정규화해줍니다.

$$P(Y = k|X) = \frac{e^{s_k}}{\sum_{j=1}^K e^{s_j}}$$

6.4 다중 클래스 분류 ($K > 2$)

임계값 0.5는 더 이상 의미가 없습니다. 대신 **'다수결 원칙(Plurality Wins)'**을 사용합니다. Softmax를 통해 계산된 K 개의 확률 중, **가장 높은 확률을 가진 클래스**로 예측합니다.

예: $P(\text{CS}) = 0.2, P(\text{Stat}) = 0.4, P(\text{Other}) = 0.4 \implies$ 가장 높은 확률이 0.4로 두 개이므로, 둘 중 하나를 선택(혹은 추가 규칙 적용). 만약 $P(\text{Stat}) = 0.41, P(\text{Other}) = 0.39$ 였다면 $\hat{Y} = \text{Stat}$ 로 예측.

7 분류 모델 평가 (Evaluation)

모델을 만들었다면, 이 모델이 얼마나 '좋은' 분류기인지 평가해야 합니다. 단순 정확도(Accuracy)는 특히 데이터가 불균형 할 때(예: 99%가 'No', 1%가 'Yes') 성능을 오해하게 만듭니다.

7.1 혼동 행렬 (Confusion Matrix)

분류 결과(예측)와 실제 값을 2×2 표로 정리한 것입니다.

		모델의 예측 (Predicted)	
		Negative (0)	Positive (1)
실제 값 (Actual)	Negative (0)	True Negative (TN)	False Positive (FP)
	Positive (1)	False Negative (FN)	True Positive (TP)

- **True Positive (TP):** 정답. 실제 Positive, 예측 Positive (예: 암환자를 암으로 진단)
- **True Negative (TN):** 정답. 실제 Negative, 예측 Negative (예: 건강한 사람을 건강하다고 진단)
- **False Positive (FP):** 1종 오류. 실제 Negative, 예측 Positive (예: 건강한 사람을 암으로 오진)
- **False Negative (FN):** 2종 오류. 실제 Positive, 예측 Negative (예: 암환자를 건강하다고 오진) ← 치명적 오류!

7.2 핵심 평가지표

1. 민감도 (Sensitivity) = 재현율 (Recall) = True Positive Rate (TPR)

$$\text{Sensitivity} = \frac{TP}{TP + FN} \quad (\text{실제 Positive 중 맞춘 비율})$$

의미: 실제 암환자 중 몇 %를 '암'이라고 잡아냈는가? (FN을 줄이는 데 초점) 의료 진단에서 매우 중요합니다. (놓치면 안 됨)

2. 특이도 (Specificity) = True Negative Rate (TNR)

$$\text{Specificity} = \frac{TN}{TN + FP} \quad (\text{실제 Negative 중 맞춘 비율})$$

의미: 실제 건강한 사람 중 몇 %를 '건강'이라고 판단했는가? (FP를 줄이는 데 초점) FP의 비용이 클 때 (예: 스팸 필터가 중요한 메일을 스팸 처리) 중요합니다.

3. 정밀도 (Precision) = Positive Predictive Value (PPV)

$$\text{Precision} = \frac{TP}{TP + FP} \quad (\text{예측 Positive 중 맞춘 비율})$$

의미: 모델이 '암'이라고 예측한 사람들 중, 실제 암환자는 몇 %인가?

4. False Positive Rate (FPR)

$$\text{FPR} = 1 - \text{Specificity} = \frac{FP}{TN + FP} \quad (\text{실제 Negative 중 틀린 비율})$$

의미: 건강한 사람 중 몇 %를 '암'이라고 잘못 예측했는가?

주의사항

베이즈 정리와 낮은 유병률(Prevalence) 문제

베이즈 정리에 따르면, 아무리 테스트기(모델)의 민감도(99%)와 특이도(99%)가 높아도, 질병 자체가 매우 희귀하다면(예: 유병률 0.1%), 테스트 결과가 '양성(Positive)'이 나왔더라도 실제 환자일 확률(PPV/정밀도)은 매우 낮을 수 있습니다. 대부분이 False Positive이기 때문입니다.

7.3 임계값(Threshold)의 트레이드오프

분류 임계값 0.5는 절대적인 기준이 아닙니다. 임계값을 조절하면 민감도(TPR)와 특이도(1-FPR)가 반비례 관계(Trade-off)로 움직입니다.

- **임계값을 낮추면 (예: 0.5 → 0.3):** 모델이 'Positive'라고 더 쉽게 예측합니다. $TP \uparrow$ (좋음), $FN \downarrow$ (좋음) \Rightarrow 민감도(TPR) 상승 $FP \uparrow$ (나쁨), $TN \downarrow$ (나쁨) \Rightarrow FPR 상승 (특이도 하락) (예: "일단 암일 가능성이 조금만 있어도 양성으로 판정" → FN은 줄지만 FP가 늘어남)
- **임계값을 높이면 (예: 0.5 → 0.7):** 모델이 'Positive'라고 더 보수적으로 예측합니다. $TP \downarrow$ (나쁨), $FN \uparrow$ (나쁨) \Rightarrow 민감도(TPR) 하락 $FP \downarrow$ (좋음), $TN \uparrow$ (좋음) \Rightarrow FPR 하락 (특이도 상승) (예: "확실히 암일 때만 양성으로 판정" → FP는 줄지만 FN이 늘어남)

7.4 ROC 커브와 AUC

ROC 커브 (Receiver Operating Characteristic Curve)는 이 트레이드오프를 시각화한 그래프입니다.

- **X축:** False Positive Rate (FPR) ($1 -$ 특이도)
- **Y축:** True Positive Rate (TPR) (민감도)

모든 가능한 임계값(0에서 1까지)에 대해 (FPR, TPR) 좌표를 찍어서 연결한 선입니다.

[이미지 삽입: ROC 커브 그래프. $(0,0)$ 에서 $(1,1)$ 을 잇는 점선(Random Classifier), $(0,0)$ 에서 $(0,1)$ 을 거쳐 $(1,1)$ 로 가는 실선(Perfect Classifier), 그리고 그 사이를 지나는 실제 모델의 ROC 커브를 보여줌]

- **완벽한 모델 (Perfect Classifier):** $(0, 1)$ 지점을 통과합니다 ($FPR=0$, $TPR=1$).
- **랜덤 모델 (Random Classifier):** $y = x$ 대각선. (FPR과 TPR이 같음)
- **좋은 모델:** 커브가 왼쪽 위 $(0, 1)$ 에 최대한 가까이 붙습니다.

AUC (Area Under the Curve)는 이 ROC 커브 아래의 면적입니다. 0부터 1 사이의 값을 가지며, 모델의 전체적인 성능을 하나의 숫자로 요약해줍니다.

- **AUC = 1.0:** 완벽한 분류기
- **AUC = 0.5:** 쓸모없는 분류기 (랜덤 추측)
- **$AUC \approx 0.8\sim0.9$:** 매우 좋은 분류기

AUC는 임계값에 상관없이 모델이 'Positive' 샘플을 'Negative' 샘플보다 얼마나 더 높은 확률로 예측하는지(순서)를 나타내는 지표입니다.

8 핵심 용어 정리

Table 1: 로지스틱 회귀 및 분류 평가 핵심 용어

용어	원어	쉬운 설명
오즈	Odds	성공 확률 / 실패 확률. ($P/(1 - P)$)
로그-오즈	Log-Odds	오즈에 자연로그를 취한 값. $\ln(P/(1 - P))$. 로지스틱 회귀의 Y 값.
오즈비	Odds Ratio (OR)	X 가 1단위 증가할 때, 오즈가 몇 '배' 변하는지. (e^β)
MLE	Max Likelihood Estimation	데이터가 관찰될 확률을 최대화하는 β 를 찾는 추정 방식.
이진 교차 엔트로피	Binary Cross-Entropy	로지스틱 회귀의 손실 함수(Loss Function). 음의 로그-가능도.
결정 경계	Decision Boundary	예측 클래스가 0에서 1로 바뀌는 경계선. $P = 0.5$ (즉, Log-Odds=0) 인 지점.
정규화	Regularization	모델 복잡도에 페널티를 주어 과적합을 막는 기법. (예: L2/Ridge)
C 파라미터	C (in sklearn)	$1/\lambda$. 정규화 강도의 역수. (C가 낮을수록 정규화가 강함)
OvR	One-vs-Rest	K개 클래스 분류 시, K개의 이진 분류기('A' vs 'Not A')를 만듦.
다항 회귀	Multinomial Regression	K개 클래스 분류 시, 1개의 기준 클래스 대비 K-1개 모델을 만듦.
소프트맥스	Softmax	K개의 클래스 점수(Logit)를 총합 1인 확률로 변환하는 함수.
혼동 행렬	Confusion Matrix	모델의 예측(TP, FP, FN, TN)과 실제 값을 비교한 표.
민감도 (재현율)	Sensitivity (Recall)	실제 '성공' 중 모델이 '성공'으로 맞춘 비율. ($TP/(TP + FN)$)
특이도	Specificity	실제 '실패' 중 모델이 '실패'로 맞춘 비율. ($TN/(TN + FP)$)
정밀도	Precision	모델이 '성공' 예측 중 실제 '성공' 인 비율. ($TP/(TP + FP)$)
ROC 커브	ROC Curve	모든 임계값에 대해 (FPR, TPR)을 그린 그래프.
AUC	Area Under the Curve	ROC 커브 아래 면적. 1에 가까울수록 좋은 모델.

9 학습 체크리스트

title=최종 점검 체크리스트

- 왜 이진 분류 문제에 선형 회귀 대신 로지스틱 회귀를 써야 하는지 설명할 수 있는가?
- 확률(P), 오즈(Odds), 로그-오즈(Log-Odds)의 관계를 설명할 수 있는가?
- β_1 계수와 e^{β_1} (오즈비)의 해석상 차이를 설명할 수 있는가?
- 다중 로지스틱 회귀에서 β_j 를 해석할 때 ”다른 변수를 통제할 때”라는 조건이 왜 붙는지 아는가?
- 상호작용 항($X_1 \times X_2$)이 모델의 절편과 기울기에 각각 어떤 영향을 미치는지 설명할 수 있는가?
- 결정 경계(Decision Boundary)가 $P = 0.5$ 지점, 즉 $X\beta = 0$ 지점과 같다는 것을 수학적으로 유도할 수 있는가?
- 모델에 다항식 항을 추가하면 결정 경계가 어떻게 변하는지 아는가?
- 정규화(Regularization)가 필요한 이유(과적합 방지)를 설명할 수 있는가?
- sklearn의 ‘C’ 파라미터가 낮을수록 정규화가 강해진다는 것을 아는가? ($C \approx 1/\lambda$)
- 다중 클래스 분류의 2가지 접근법 (OvR, Multinomial)을 비교할 수 있는가?
- Softmax 함수의 역할(점수 \rightarrow 확률 정규화)을 아는가?
- 혼동 행렬의 TP, FP, FN, TN이 각각 무엇을 의미하는지 아는가?
- 민감도(재현율), 특이도, 정밀도의 차이를 (공식과 의미) 설명할 수 있는가?
- 분류 임계값(Threshold)을 조절하면 민감도와 특이도가 어떻게 변하는지(Trade-off) 아는가?
- ROC 커브의 X축(FPR)과 Y축(TPR)이 무엇이며, AUC가 왜 0.5면 ‘랜덤’인지 설명할 수 있는가?

10 1페이지 요약 (1-Page Summary)

title=1. 로지스틱 회귀 모델 Y 가 0 또는 1일 때, 성공 확률 P 를 직접 모델링하지 않고, **로그-오즈**를 X 에 대한 선형식으로 모델링합니다.

$$\ln \left(\frac{P}{1 - P} \right) = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p$$

β 는 MLE (최대가능도 추정법) 또는 경사하강법으로 추정합니다.

title=2. 계수 해석 β_j 는 X_j 가 1단위 증가할 때 **로그-오즈의 덧셈 변화량**입니다. e^{β_j} (**오즈비**)는 X_j 가 1단위 증가할 때 **오즈의 곱셈 변화량** (배수)입니다.

title=3. 결정 경계 (Decision Boundary) 분류 임계값을 $P = 0.5$ 로 두면, 결정 경계는 $\ln(\text{Odds}) = 0$ 이 되는 지점, 즉 $\beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p = 0$ 이 됩니다. X 의 1차항만 있으면 직선, 다항식/상호작용 항이 있으면 곡선이 됩니다.

title=4. 정규화 (Regularization) 과적합을 막기 위해 손실 함수(이진 교차 엔트로피)에 페널티 항을 추가합니다. (L2/Ridge) $\text{Loss}_{\text{Reg}} = \text{Loss} + \lambda \sum \beta_j^2$. ‘sklearn’에서는 $C \approx 1/\lambda$ 를 사용하며, C 가 낮을수록 정규화가 강합니다. 최적의 C 는 교차 검증(Cross-Validation)으로 찾습니다.

title=5. 다중 클래스 (Multiclass) $K > 2$

- **OvR (One-vs-Rest):** K 개의 이진 분류기('A' vs 'Not A')를 만듭니다.
- **Multinomial:** 1개의 기준 클래스 대비 $K-1$ 개 모델을 만듭니다.
- **Softmax:** K 개의 점수(Logit)를 총합 1인 확률로 변환.
- **분류:** 가장 높은 확률을 가진 클래스로 예측 (Plurality Wins).

title=6. 분류 평가 (Evaluation)

- **혼동 행렬:** TP, FP, FN, TN
- **민감도(TPR):** $\frac{TP}{TP+FN}$ (실제 P 중 예측 P)
- **특이도(TNR):** $\frac{TN}{TN+FP}$ (실제 N 중 예측 N)
- **정밀도(PPV):** $\frac{TP}{TP+FP}$ (예측 P 중 실제 P)
- **ROC 커브:** X축=FPR (1-특이도), Y축=TPR (민감도). 모든 임계값에서의 성능 시각화.
- **AUC:** ROC 커브 아래 면적. 1에 가까울수록 좋은 분류기.

December 10, 2025

- 강의명: CS109A: 데이터 과학 입문
- 주차: Lecture 15
- 교수명: Pavlos Protopapas, Kevin Rader, Chris Gumb
- 목적: Lecture 15의 핵심 개념 학습

Contents

1	핵심 용어 정리	3
2	다중 클래스 로지스틱 회귀 (Multiclass Logistic Regression)	4
2.1	왜 다중 클래스 분류가 필요한가?	4
2.2	방법 1: 다항 로지스틱 회귀 (Multinomial Logistic Regression)	4
2.2.1	Q: 2개의 모델로 어떻게 3개의 확률을 얻나요?	4
2.3	방법 2: One-vs-Rest (OvR) 로지스틱 회귀	5
2.3.1	Q: 이 3개의 확률은 합이 1이 되나요?	5
2.3.2	해결책: 소프트맥스 (Softmax) 함수	5
2.4	Multinomial vs. OvR: 무엇을 써야 할까?	6
2.5	다중 클래스에서의 예측과 순실 함수	6
3	분류 모델 평가 (Evaluating Classifiers)	7
3.1	혼동 행렬 (The Confusion Matrix)	7
3.2	임계값(Threshold)과 성능의 트레이드오프	7
3.3	ROC 곡선 (Receiver Operating Characteristic Curve)	8
3.4	AUC (Area Under the Curve)	8
4	베이즈 추론 (Bayesian Inference)	9
4.1	베이즈 정리 (Bayes' Theorem)	9
5	베타-이항 모델 (The Beta-Binomial Model)	9
5.1	1단계: 가능성 (Likelihood) - 데이터가 말하는 것	9
5.2	2단계: 사전 확률 (Prior) - 우리의 초기 믿음	10
5.3	3단계: 사후 확률 (Posterior) - 업데이트된 믿음	10
6	베이즈 로지스틱 회귀와 계층 모델	12
6.1	베이즈 로지스틱 회귀 (Bayesian Logistic Regression)	12
6.2	계층 모델 (Hierarchical Modeling) 미리보기	12

Abstract

개요 (Overview)

본 문서는 로지스틱 회귀를 3개 이상의 클래스를 분류하는 다중 클래스(Multiclass) 문제로 확장하는 방법을 다룹니다. 기준이 되는 하나의 클래스와 나머지를 비교하는 다항(Multinomial) 로지스틱 회귀와, 각 클래스와 그 외 모든 클래스를 비교하는 OvR(One-vs-Rest) 접근법을 배웁니다. 분류 모델의 성능을 평가하기 위한 혼동 행렬(Confusion Matrix), ROC 곡선, AUC 개념을 학습합니다. 마지막으로, 파라미터를 확률 분포로 간주하는 베이즈(Bayesian) 추론의 기본 개념과, 이항 분포의 결합 사전 확률인 베타 분포(Beta Distribution)를 활용한 베타-이항 모델을 살펴보고, 계층 모델(Hierarchical Model)의 필요성을 소개합니다.

1 핵심 용어 정리

본 강의에서 다루는 주요 용어들을 미리 살펴봅니다.

[title=주요 용어]

- **다중 클래스 분류 (Multiclass Classification):** 결과 변수(Y)가 3개 이상의 범주를 가지는 분류 문제입니다. (예: 학생의 전공을 'CS', '통계', '기타'로 예측)
- **다항 로지스틱 회귀 (Multinomial Logistic Regression):** 다중 클래스 분류 기법 중 하나. 하나의 클래스를 '기준(reference)'(예: K번째 클래스)로 설정하고, 다른 모든 클래스(1, 2, ..., K-1)를 이 기준 클래스와 비교하는 $K - 1$ 개의 이진 로지스틱 모델을 적합시킵니다.
- **One-vs-Rest (OvR) 로지스틱 회귀:** 다중 클래스 분류 기법 중 하나. 총 K 개의 클래스가 있다면, K 개의 이진 로지스틱 모델을 각각 적합시킵니다. 각 모델은 '특정 클래스 k' vs 'k를 제외한 나머지 모든 클래스'를 분류합니다.
- **소프트맥스 함수 (Softmax Function):** 여러 개의 점수(score)를 입력 받아, 총 합이 1이 되는 확률 값들의 집합으로 변환하는 함수입니다. OvR 모델 등에서 각 클래스에 속할 최종 확률을 계산하는데 사용됩니다.
- **혼동 행렬 (Confusion Matrix):** 분류 모델의 예측 결과를 실제 값과 비교하여 표로 나타낸 것입니다. TP, FP, TN, FN 값을 포함합니다.
- **ROC 곡선 (Receiver Operating Characteristic Curve):** 분류 모델의 임계값(threshold)이 변함에 따라 True Positive Rate (TPR, Y축)와 False Positive Rate (FPR, X축)가 어떻게 변하는지를 그린 그래프입니다.
- **AUC (Area Under the Curve):** ROC 곡선의 아래쪽 면적. 1에 가까울수록 모델의 성능이 좋다고 평가하며, 0.5는 무작위 추측과 같은 수준임을 의미합니다.
- **베이즈 추론 (Bayesian Inference):** 모델의 파라미터를 고정된 값이 아닌 확률 분포로 간주하는 통계적 접근 방식입니다. 사전 확률(Prior)에 가능도(Likelihood)를 곱하여(데이터를 반영하여) 사후 확률(Posterior)을 계산합니다.
- **베타 분포 (Beta Distribution):** $[0, 1]$ 사이의 값을 가지는 연속 확률 분포. 확률 값(p) 자체의 불확실성을 모델링하는 데 사용되며, 이항 분포의 커티지 사전 확률(Conjugate Prior)입니다.

2 다중 클래스 로지스틱 회귀 (Multiclass Logistic Regression)

2.1 왜 다중 클래스 분류가 필요한가?

기존의 로지스틱 회귀는 반응 변수 Y 가 0 또는 1 (예: 실패/성공, 스팸/아님)인 이진 분류(Binary Classification) 문제에 사용되었습니다.

하지만 현실의 많은 문제는 3개 이상의 범주를 가집니다.

- 학생의 전공 예측: {컴퓨터 과학, 통계학, 기타}
- 미식축구 플레이 예측: {패스, 런, 스페셜 팀}
- 상품 카테고리 분류: {의류, 가전, 식품, 도서}

이러한 문제를 **다중 클래스 분류(Multiclass Classification)**라고 부릅니다. 다중 클래스 문제는 범주의 순서 유무에 따라 두 가지로 나뉩니다.

- **명목형 (Nominal):** 범주 간에 순서가 없습니다. (예: 눈동자 색 - 파랑, 갈색, 초록)
- **순서형 (Ordinal):** 범주 간에 명확한 순서가 있습니다. (예: 평점 - 1점, 2점, 3점, 4점, 5점)

이번 강의에서는 **명목형** 다중 클래스 문제를 다루는 두 가지 주요 방법을 배웁니다.

2.2 방법 1: 다항 로지스틱 회귀 (Multinomial Logistic Regression)

이 방법은 '기준 그룹'을 하나 정하고, 다른 모든 그룹을 이 기준 그룹과 비교하는 방식입니다.

□ 예제: title

$K = 3$ 개의 클래스 {CS(1), Stat(2), Other(3)} 가 있다고 가정합니다. 만약 **Other(3)**를 기준(reference) 그룹으로 삼는다면, 우리는 $K - 1 = 2$ 개의 이진 로지스틱 모델을 만듭니다.

- **모델 1:** CS(1) vs Other(3) 분류

$$\ln \left(\frac{P(Y=1)}{P(Y=3)} \right) = \beta_{0,1} + \beta_{1,1}X_1 + \cdots + \beta_{p,1}X_p$$

- **모델 2:** Stat(2) vs Other(3) 분류

$$\ln \left(\frac{P(Y=2)}{P(Y=3)} \right) = \beta_{0,2} + \beta_{1,2}X_1 + \cdots + \beta_{p,2}X_p$$

2.2.1 Q: 2개의 모델로 어떻게 3개의 확률을 얻나요?

좋은 질문입니다. 우리는 $P(Y=1)$, $P(Y=2)$, $P(Y=3)$ 세 가지를 알고 싶습니다. 위의 두 모델은 두 개의 방정식을 제공합니다. 하지만 미지수는 3개입니다. 이때, 확률의 기본 속성인 "모든 확률의 합은 1이다"라는 세 번째 방정식을 사용합니다.

1. $\frac{P(Y=1)}{P(Y=3)} = e^{\beta_1 X}$ ($\beta_1 X$ 는 모델 1의 선형 결합)
2. $\frac{P(Y=2)}{P(Y=3)} = e^{\beta_2 X}$ ($\beta_2 X$ 는 모델 2의 선형 결합)
3. $P(Y=1) + P(Y=2) + P(Y=3) = 1$

이 3개의 방정식을 연립하여 $P(Y=1)$, $P(Y=2)$, $P(Y=3)$ 을 모두 구할 수 있습니다. (예: 1번과 2번

식을 $P(Y = 1)$ 과 $P(Y = 2)$ 에 대해 정리하여 3번 식에 대입하면 $P(Y = 3)$ 를 구할 수 있습니다.)

sklearn 라이브러리 사용 시 참고

이론적으로는 $K - 1$ 개의 모델을 적합하지만, sklearn의 LogisticRegression(multi_class='multinomial')은 K 개의 계수 세트(β)를 반환합니다.

이는 sklearn이 내부적으로 계산을 정규화(renormalize)하여, 각 클래스 k 에 대해 $P(Y = k)$ vs $P(Y \neq k)$ (k vs k 가 아닌 것)에 대한 해석이 가능하도록 변환해주기 때문입니다. 처음에는 혼동될 수 있지만, K 개의 확률을 직접 다루는 것이 더 직관적일 수 있습니다.

2.3 방법 2: One-vs-Rest (OvR) 로지스틱 회귀

이 방법은 '기준 그룹' 없이, 각 클래스가 돌아가면서 주인공이 되는 방식입니다. K 개의 클래스가 있다면 K 개의 모델을 만듭니다.

□ 예제: title

$K = 3$ 개의 클래스 {CS, Stat, Other}가 있다면, 3개의 이진 로지스틱 모델을 만듭니다.

- 모델 1: CS vs (Stat + Other) 분류

$$\ln \left(\frac{P(Y = \text{CS})}{P(Y \neq \text{CS})} \right) = \beta_{\text{CS}} X$$

- 모델 2: Stat vs (CS + Other) 분류

$$\ln \left(\frac{P(Y = \text{Stat})}{P(Y \neq \text{Stat})} \right) = \beta_{\text{Stat}} X$$

- 모델 3: Other vs (CS + Stat) 분류

$$\ln \left(\frac{P(Y = \text{Other})}{P(Y \neq \text{Other})} \right) = \beta_{\text{Other}} X$$

2.3.1 Q: 이 3개의 확률은 합이 1이 되나요?

아니요, 보장되지 않습니다. 이 3개의 모델은 독립적으로 학습됩니다. 모델 1은 "이 학생이 CS일 확률" (p_{CS})을, 모델 2는 "Stat일 확률" (p_{Stat})을, 모델 3은 "Other일 확률" (p_{Other})을 계산합니다. 이 3개의 확률($p_{\text{CS}}, p_{\text{Stat}}, p_{\text{Other}}$)을 단순히 더하면 1이 되지 않을 수 있습니다.

2.3.2 해결책: 소프트맥스 (Softmax) 함수

이 문제를 해결하기 위해, 각 모델에서 나온 "점수"(score, βX)를 총합이 1이 되는 확률로 변환하는 소프트맥스 함수를 사용합니다.

[title=소프트맥스 함수 (Softmax Function)] K 개의 클래스에 대한 점수(logits) $\vec{s} = (s_1, s_2, \dots, s_K)$ 가 있을 때, k 번째 클래스에 속할 확률 P_k 는 다음과 같이 계산됩니다.

$$P_k = \frac{e^{s_k}}{\sum_{j=1}^K e^{s_j}}$$

직관적 해석: 1. e^{s_k} (지수 함수): 모든 점수를 양수로 만들고, 큰 점수와 작은 점수의 차이를 더욱 증폭시킵니다. (Winner-takes-most) 2. $\sum e^{s_j}$ (총합): 모든 클래스의 증폭된 점수 총합입니다. 3. 나누기: 각 클래스의 증폭된 점수를 총합으로 나누어, 전체에서 차지하는 ”비율”을 계산합니다. 이렇게 하면 모든 확률(P_k)의 합은 항상 1이 됩니다.

2.4 Multinomial vs. OvR: 무엇을 써야 할까?

두 방법은 종종 매우 유사한 예측 결과를 제공합니다. 미식축구(NFL) 플레이 타입을 예측하는 예제(패스, 런, 기타)에서도 두 모델의 예측 확률 그래프는 거의 동일한 경향을 보였습니다.

특징	다항 (Multinomial)	OvR (One-vs-Rest)
모델 개수	$K - 1$ 개	K 개
개념	기준 클래스(K) vs. 나머지(k)	클래스(k) vs. 나머지($\neq k$)
효율성	약간 더 효율적 (모델 적음)	개념이 단순함
적합	추론/계수 비교에 유리	순수 분류(prediction)에 선호됨
결과	대부분의 경우 매우 유사한 성능을 보임	

어떤 모델이 더 나은지는 교차 검증(Cross-validation)을 통해 ’테스트 데이터’에 대한 성능(예: 손실 함수 값)을 비교하여 결정할 수 있습니다.

2.5 다중 클래스에서의 예측과 손실 함수

예측 방법: 이진 분류에서는 $P(Y = 1) > 0.5$ 이면 1로 예측했습니다. 다중 클래스에서는 어떤 클래스의 확률도 0.5를 넘지 않을 수 있습니다. (예: $P(A) = 0.4, P(B) = 0.3, P(C) = 0.3$) 따라서 가장 큰 예측 확률을 가진 클래스를 최종 예측값으로 선택합니다.

데이터 불균형 문제: 만약 특정 클래스가 데이터의 대부분을 차지한다면(예: NFL 플레이의 66%가 ’패스’), 모델은 예측 정확도를 높이기 위해 거의 모든 예측을 ’패스’로 할 수 있습니다. (예: ”코카인 사용자 예측” 예제) 이 경우, 모델이 단순히 다수 클래스만 예측하더라도 ’분류 정확도’는 높게 나옵니다. 하지만 이 모델이 소수 클래스에 대한 유의미한 관계를 포착했을 수 있습니다. 따라서 단순 ’분류’ 결과뿐만 아니라 ’확률’ 자체를 보는 것이 중요합니다.

손실 함수: 이진 분류의 손실 함수를 **Binary Cross-Entropy**라고 불렀습니다. 다중 클래스 분류의 손실 함수는 이를 일반화한 **Cross-Entropy** (또는 Multinomial Logistic Loss)라고 부릅니다. 이 손실 함수에 Ridge(L2)나 Lasso(L1) 패널티 항을 추가하여 정규화(Regularization)를 수행할 수 있습니다.

3 분류 모델 평가 (Evaluating Classifiers)

모델을 만들었다면, 이 모델이 얼마나 좋은지 평가해야 합니다. 숫자 예측(회귀)에서 MSE를 쓴 것처럼, 분류 문제에도 전용 평가 지표가 필요합니다.

3.1 혼동 행렬 (The Confusion Matrix)

모든 분류 평가는 혼동 행렬에서 시작합니다. 이는 모델의 예측 값과 실제 값을 비교한 2x2 표입니다. (이진 분류 기준)

		예측된 값 (Predicted)	
		Negative (0)	Positive (1)
실제 값 (Actual)	Negative (0)	True Negative (TN)	False Positive (FP)
	Positive (1)	False Negative (FN)	True Positive (TP)

[title=혼동 행렬의 4가지 요소]

- **True Positive (TP):** 실제 1(Positive) 인 것을 1로 올바르게 예측. (예: 스팸 메일을 스팸으로 분류)
- **True Negative (TN):** 실제 0(Negative) 인 것을 0으로 올바르게 예측. (예: 일반 메일을 일반 메일로 분류)
- **False Positive (FP) / 1종 오류:** 실제 0(Negative) 인 것을 1로 잘못 예측. (예: 일반 메일을 스팸으로 분류)
- **False Negative (FN) / 2종 오류:** 실제 1(Positive) 인 것을 0으로 잘못 예측. (예: 스팸 메일을 일반 메일로 분류)

3.2 임계값(Threshold)과 성능의 트레이드오프

로지스틱 회귀 모델은 '분류'(0 또는 1)를 직접 출력하는 것이 아니라 '확률'(예: 0.7)을 출력합니다. 우리는 이 확률을 임계값(Threshold)과 비교하여 최종 분류를 결정합니다. (보통 0.5 사용)

$$\hat{P}(Y = 1) > \text{threshold} \implies \text{Predict 1}$$

이 임계값을 조절하면 모델의 특성이 바뀝니다.

임계값(Threshold) 조절의 효과

- **임계값을 낮추면 (예: 0.4):** 모델이 '1'로 예측하기 쉬워집니다.
 - **장점:** 실제 1인 것을 놓치지 않습니다. (TP 증가, FN 감소)
 - **단점:** 실제 0인 것을 1로 오인합니다. (FP 증가)
 - **예시:** 암 진단 모델. 환자를 놓치는 것(FN)이 치명적이므로 임계값을 낮춰 민감하게 반응하도록 합니다. (재검사하더라도 일단 잡아냄)
- **임계값을 높이면 (예: 0.6):** 모델이 '1'로 예측하기 어려워집니다. (매우 확신할 때만 1로 예측)
 - **장점:** 실제 0인 것을 1로 오인하지 않습니다. (FP 감소)
 - **단점:** 실제 1인 것을 놓치게 됩니다. (TP 감소, FN 증가)

- 예시: 스팸 메일 필터. 일반 메일을 스팸으로 보내는 것(FP)이 매우 불편하므로 임계값을 높여 확실한 스팸만 걸러내도록 합니다.
- 결론: FN을 줄이면 FP가 늘어나고, FP를 줄이면 FN이 늘어나는 트레이드오프(Trade-off) 관계가 존재합니다.

3.3 ROC 곡선 (Receiver Operating Characteristic Curve)

”그렇다면, 수많은 임계값 중 어떤 것을 선택해야 할까요? 혹시 임계값에 상관없이 모델 자체의 성능을 평가할 수는 없을까요?”

이 질문에 답하는 것이 ROC 곡선입니다. ROC 곡선은 모든 가능한 임계값에 대해 모델의 성능을 그래프로 그린 것입니다.

- Y축: True Positive Rate (TPR) / 민감도 (Sensitivity) / 재현율 (Recall)

$$TPR = \frac{TP}{TP + FN}$$

(실제 Positive 중에서 모델이 Positive라고 예측한 비율. 1에 가까울수록 좋음)

- X축: False Positive Rate (FPR)

$$FPR = \frac{FP}{FP + TN}$$

(실제 Negative 중에서 모델이 Positive라고 잘못 예측한 비율. 0에 가까울수록 좋음)

ROC 곡선 해석하기

- 완벽한 분류기 (Perfect Classifier): (0, 1) 지점을 지나는 곡선. (FPR=0이면서 TPR=1, 즉 모든 것을 완벽하게 분류함)
- 무작위 분류기 (Random Classifier): (0, 0)에서 (1, 1)로 이어지는 대각선 ($y=x$). FPR 50%를 감수해야 TPR 50%를 얻는다는 의미로, 동전 던지기(무작위 추측)와 같습니다.
- 좋은 분류기 (Good Classifier): 대각선보다 위쪽, 즉 원쪽 상단에 최대한 가깝게 (Hugging) 그려지는 곡선입니다. 이는 낮은 FPR(적은 오인)으로도 높은 TPR(많은 정답)을 달성한다는 의미입니다.

3.4 AUC (Area Under the Curve)

ROC 곡선은 모델의 전체적인 성능을 보여주지만, 두 모델의 곡선이 서로 교차하는 등 비교가 어려울 수 있습니다. AUC(Area Under the Curve)는 ROC 곡선 아래의 면적을 계산하여 모델의 성능을 하나의 숫자로 요약합니다.

- AUC = 1.0: 완벽한 분류기 (면적이 1×1 정사각형)
- AUC = 0.5: 무작위 분류기 (면적이 $y=x$ 대각선 아래 삼각형)
- AUC > 0.5: 무작위보다 좋은 모델.

AUC는 임계값에 관계없이 모델이 얼마나 Positive와 Negative 샘플을 잘 구별하는지 나타내는 지표입니다. AUC가 높을수록 좋은 모델입니다.

4 베이즈 추론 (Bayesian Inference)

지금까지 우리는 빈도주의(Frequentist) 관점에서 통계를 다렸습니다. 빈도주의에서는 모델 파라미터(β)가 '고정되어 있지만 알지 못하는 값'이라고 가정하고, 데이터를 사용해 이 값을 '추정'했습니다.

베이즈주의(Bayesian) 관점은 파라미터를 다르게 봅니다.

[title=베이즈 추론의 핵심] 베이즈 관점에서 파라미터(θ)는 고정된 값이 아니라, 불확실성을 가진 확률 변수입니다. 우리는 파라미터에 대한 믿음의 분포(Distribution of Belief)를 가지고 있으며, 데이터를 관찰함으로써 이 믿음을 업데이트합니다.

4.1 베이즈 정리 (Bayes' Theorem)

이 '믿음의 업데이트' 과정은 베이즈 정리를 통해 수학적으로 수행됩니다.

$$\underbrace{f(\theta|X)}_{\text{Posterior}} \propto \underbrace{f(X|\theta)}_{\text{Likelihood}} \cdot \underbrace{f(\theta)}_{\text{Prior}}$$

- **Prior (사전 확률)** $f(\theta)$: 데이터(X)를 보기 전, 파라미터 θ 에 대해 우리가 가진 초기 믿음의 분포입니다. (예: "이 동전은 아마 공정할 거야" $\rightarrow p = 0.5$ 근처에 확률을 높게 부여)
- **Likelihood (가능도)** $f(X|\theta)$: '만약 파라미터가 θ 라면, 우리가 가진 데이터 X 가 관찰될 확률'입니다. (이는 빈도주의의 '가능도 함수'와 동일합니다.)
- **Posterior (사후 확률)** $f(\theta|X)$: 데이터(X)를 관찰한 후, 파라미터 θ 에 대해 업데이트된 믿음의 분포입니다. 이 사후 확률은 우리의 '최종 결과물'입니다.

베이즈 추론의 과정

초기 믿음 (Prior) \times 데이터의 증거 (Likelihood) \implies 업데이트된 믿음 (Posterior)

5 베타-이항 모델 (The Beta-Binomial Model)

베이즈 추론의 가장 고전적인 예시인 '동전 뒤집기' 문제를 통해 베이즈 추론을 이해해 봅니다. 우리의 목표는 동전의 앞면이 나올 확률 p 를 추정하는 것입니다. (단, p 는 0.5가 아닐 수도 있습니다.)

5.1 1단계: 가능도 (Likelihood) - 데이터가 말하는 것

동전을 n 번 던져 앞면(Success)이 $\sum x_i$ 번, 뒷면(Failure)이 $n - \sum x_i$ 번 나왔다고 합시다. 파라미터 p 가 주어졌을 때 이 데이터가 관찰될 확률(가능도)은 이항 분포(Binomial Distribution)를 따릅니다.

$$f(X|p) \propto p^{\sum x_i} (1-p)^{n-\sum x_i}$$

(앞면이 나온 횟수만큼 p 가, 뒷면이 나온 횟수만큼 $(1-p)$ 가 곱해집니다.)

5.2 2단계: 사전 확률 (Prior) - 우리의 초기 믿음

이제 p 에 대한 우리의 초기 믿음을 설정해야 합니다. p 는 확률 값이므로 $[0, 1]$ 사이의 분포여야 합니다. 이때 베타 분포(Beta Distribution)가 사용됩니다.

[title=베타 분포 Beta(α, β)] 베타 분포는 $[0, 1]$ 사이의 값을 가지며, 두 개의 하이퍼파라미터(hyperparameter) α 와 β 에 의해 모양이 결정됩니다.

$$f(p|\alpha, \beta) \propto p^{\alpha-1}(1-p)^{\beta-1}$$

직관적 해석: 베타 분포는 ”과거에 $\alpha - 1$ 번의 성공과 $\beta - 1$ 번의 실패를 본 것과 같은 믿음”을 나타냅니다.

- $E[p] = \frac{\alpha}{\alpha+\beta}$ (분포의 평균)
- $Beta(1, 1) : f(p) \propto p^0(1-p)^0 = 1$. 균등 분포(Uniform Distribution)와 같습니다. ”나는 p 에 대해 아무것도 모르며, 모든 p 값이 똑같이 가능하다”는 의미의 무정보 사전 확률(non-informative prior)입니다.
- $Beta(10, 10) : E[p] = \frac{10}{20} = 0.5$. ” p 는 0.5일 것이라고 강하게 믿는다” (공정한 동전)
- $Beta(2, 5) : E[p] = \frac{2}{7} \approx 0.28$. ”뒷면이 더 잘 나오는 동전 같다”

5.3 3단계: 사후 확률 (Posterior) - 업데이트된 믿음

베이즈 정리에 따라 사전 확률과 가능도를 곱합니다.

$$\text{Posterior} \propto \text{Likelihood} \times \text{Prior}$$

$$f(p|X) \propto [p^{\sum x_i} (1-p)^{n-\sum x_i}] \times [p^{\alpha-1}(1-p)^{\beta-1}]$$

지수 법칙에 따라 같은 밑을 가진 항들을 합칩니다.

$$f(p|X) \propto p^{(\alpha+\sum x_i)-1} \cdot (1-p)^{(\beta+n-\sum x_i)-1}$$

놀라운 결과: 이 사후 확률의 형태는 또 다른 베타 분포입니다! 우리는 $Beta(\alpha, \beta)$ 사전 확률로 시작했는데, 데이터를 반영하니 $Beta(\alpha + \text{성공 횟수}, \beta + \text{실패 횟수})$ 라는 새로운 베타 분포가 되었습니다.

베타-이항 모델 업데이트 규칙

- **Prior:** $p \sim Beta(\alpha, \beta)$
- **Data:** n 번 시도, 성공 $\sum x_i$ 번, 실패 $(n - \sum x_i)$ 번
- **Posterior:** $p|X \sim Beta(\alpha + \sum x_i, \beta + n - \sum x_i)$

이처럼 사전 확률과 사후 확률이 동일한 분포족(distribution family)에 속할 때, 이 사전 확률을 결합 사전 확률(Conjugate Prior)이라고 부릅니다. 베타 분포는 이항/베르누이 분포의 결합 사전 확률입니다.

□ 예제: title

- 1. 사전 믿음 (Prior): 동전에 대한 정보가 전혀 없어 $Beta(1, 1)$ (균등 분포)을 사용합니다. (사전 성공 횟수=0, 사전 실패 횟수=0으로 해석 가능)
- 2. 데이터 (Data): 동전을 10번 던져 앞면(성공)이 7번, 뒷면(실패)이 3번 나왔습니다. ($n = 10, \sum x_i = 7$)
- 3. 사후 믿음 (Posterior): 우리의 믿음은 $Beta(1 + 7, 1 + 3) = Beta(8, 4)$ 로 업데이트됩니다.
 - 데이터 반영 전, p 의 기댓값: $E[p] = \frac{1}{1+1} = 0.5$
 - 데이터 반영 후, p 의 기댓값: $E[p] = \frac{8}{8+4} = \frac{8}{12} \approx 0.67$데이터를 통해 우리의 믿음이 0.5에서 0.67로 이동했습니다.

6 베이즈 로지스틱 회귀와 계층 모델

6.1 베이즈 로지스틱 회귀 (Bayesian Logistic Regression)

”그렇다면 로지스틱 회귀에도 베타 분포를 사전 확률로 쓸 수 있을까요?”

NOPE!

아니요, 쓸 수 없습니다.

- 베타 분포는 $[0, 1]$ 사이의 확률 p 자체에 대한 사전 확률입니다.
- 로지스틱 회귀의 파라미터는 p 가 아니라, β_0, β_1, \dots 계수들입니다.
- β 계수들은 $(-\infty, \infty)$ 범위의 실수 값을 가질 수 있습니다.

따라서 로지스틱 회귀의 β 계수들에 대한 사전 확률로는 $[0, 1]$ 범위의 베타 분포가 아니라, $(-\infty, \infty)$ 범위의 정규 분포(Normal Distribution) (또는 라플라스 분포 등)를 사용합니다.

$$\beta_j \sim N(\mu_0, \sigma^2)$$

만약 우리가 $\mu_0 = 0$ 으로 설정한다면, 이는 ” β_j 계수는 아마 0에 가까울 것이다(즉, X_j 는 Y 에 영향이 없을 것이다)”라는 사전 믿음을 주는 것입니다. 이는 파라미터를 0으로 축소시키는 Ridge (L2) 정규화와 매우 유사한 베이즈적 접근 방식입니다.

6.2 계층 모델 (Hierarchical Modeling) 미리보기

베이즈 추론은 데이터의 구조가 복잡할 때 더욱 강력한 힘을 발휘합니다. 우리는 파라미터 θ 를 모델링하기 위해 하이퍼파라미터 α, β 를 사용했습니다.

$$Y \leftarrow p \leftarrow Beta(\alpha, \beta)$$

만약 α, β 값 자체를 정하는 것이 불확실하다면? α, β 에도 사전 확률을 부여할 수 있습니다. (예: $\alpha \sim Gamma(\dots)$) 이를 하이퍼-사전확률(Hyperprior)이라고 부르며, 이렇게 모델이 여러 층(level)을 가지는 것을 계층 모델(Hierarchical Model)이라고 합니다.

”왜 이렇게 복잡하게 모델링하나요?” 가장 큰 이유는 데이터에 중첩된(nested) 구조가 있기 때문입니다.

□ 예제: title

데이터: Y_{ij} (선수 j 의 i 번째 슛), X_{ij} (슛 거리)

목표: 슛 거리에 따른 성공 확률(p_{ij})을 모델링

$$\log\left(\frac{p_{ij}}{1 - p_{ij}}\right) = \alpha_j + \beta_1 X_{ij}$$

여기서 α_j 는 선수 j 의 고유한 ‘기본 슛 성공 능력’을 나타내는 절편입니다.

접근 1 (모델 없음): 모든 선수가 같다고 가정. ($\alpha_j = \alpha_0 \rightarrow$ 나쁨). 접근 2 (독립 모델): 선수마다 α_j 를 따로 추정. \rightarrow 슛을 적게 쏜 선수의 데이터는 불안정함.

접근 3 (계층 모델):

- **Level 1 (데이터):** 각 선수의 슛은 그 선수의 능력(α_j)에 따라 결정됨.

$$\log \left(\frac{p_{ij}}{1 - p_{ij}} \right) = \alpha_j + \beta_1 X_{ij}$$

- **Level 2 (선수):** 개별 선수의 능력(α_j)은 완전히 제멋대로가 아니라, 'NBA 선수 전체의 능력 분포'에서 샘플링된 값이라고 가정합니다.

$$\alpha_j \sim N(\alpha_{\text{league}}, \sigma_\alpha^2)$$

(모든 α_j 는 리그 평균 α_{league} 을 중심으로 σ_α^2 만큼 흩어져 있다)

장점: 이 모델은 '정보를 공유(Share information)' 합니다. 슛을 많이 쏜 선수(예: 르브론 제임스)는 α_j 가 자신의 데이터에 의해 결정됩니다. 하지만 슛을 10번만 쏜 신인 선수는, 그 10개의 데이터와 '리그 평균'(α_{league}) 사이의 가중 평균으로 α_j 가 추정됩니다. 즉, 데이터가 부족한 관측치(신인 선수)의 추정값을 리그 평균 쪽으로 당겨와(shrink) 더 안정적인 추론을 가능하게 합니다.

December 10, 2025

- 강의명: CS109A: 데이터 과학 입문
- 주차: Lecture 16
- 교수명: Pavlos Protopapas, Kevin Rader, Chris Gumb
- 목적: Lecture 16의 핵심 개념 학습

Contents

1	개요	2
2	용어 정리	3
3	핵심 개념 및 원리	4
3.1	분류(Classification)의 시작: 왜 선형 회귀는 안될까?	4
3.2	로지스틱 회귀 (Logistic Regression)	4
3.3	모델 평가와 정규화	5
3.4	베이지안 추론 (Bayesian Inference)	5
3.5	콜레 사전분포 (Beta-Binomial 모델)	6
3.6	계층 모델 (Hierarchical Models)	6
3.7	MCMC와 사후분포 샘플링	7
4	절차 및 방법 (scikit-learn 파이프라인)	8
5	실습, 코드 및 주요 함정	11
5.1	주요 함정 (Gotchas) 다시보기	11
5.2	Bayesian Logistic Regression (pymc)	11
5.3	MCMC 결과 해석 (Trace Plot)	12
6	학습 체크리스트	13
7	FAQ (자주 묻는 질문)	14
8	빠르게 훑어보기 (핵심 요약)	15

1 개요

이 문서는 분류(Classification) 문제, 특히 로지스틱 회귀(Logistic Regression)의 원리를 다룹니다. 나아가 베이지안(Bayesian) 통계의 관점을 도입하여, 모델을 계층적(Hierarchical)으로 구축하고, 그 해를 MCMC(Markov Chain Monte Carlo) 시뮬레이션을 통해 추정하는 고급 기법까지 탐구합니다.

▣ 핵심 요약

이 노트는 분류 모델의 기초인 로지스틱 회귀부터 시작합니다. 데이터에 비선형성을 추가하고(다항 회귀), 모델의 과적합을 방지하는 정규화(L1, L2)를 배웁니다. 이후 베이지안 관점을 도입해, 사전 확률(Prior)과 사후 확률(Posterior)의 개념을 이해하고, 결례 사전분포(Conjugate Prior)의 편리함(베타-이항)과 한계(로지스틱 회귀)를 배웁니다. 마지막으로, 복잡한 모델(계층 모델)의 해를 구하기 위한 강력한 시뮬레이션 도구인 MCMC와 Metropolis-Hastings 알고리즘의 원리를 학습합니다.

2 용어 정리

핵심 용어들을 미리 숙지하면 뒤따르는 개념들을 이해하기 훨씬 수월합니다.

용어	쉬운 설명 (직관)	원어(영어)	비고
분류 (Classification)	데이터를 정해진 카테고리(예: 'A', 'B')로 나누는 작업.	Classification	회귀 (Regression) 와 대비됨.
로지스틱 회귀	연속적인 값이 아닌, '성공/실패' 같은 확률을 예측하는 회귀.	Logistic Regression	이름은 회귀지만 분류 모델.
로그 오즈 (Log-odds)	확률(p)을 $(-\infty, +\infty)$ 범위로 변환한 값. $\log(\frac{p}{1-p})$	Log-odds	로지스틱 회귀의 예측 대상.
MLE	데이터가 주어졌을 때, 이 데이터를 가장 잘 설명하는 모델 파라미터를 찾는 방법.	Max. Likelihood Est.	'이 데이터가 나올 확률이 최대가 되게 하라'
교차 엔트로피	모델의 예측 확률이 실제 정답과 얼마나 다른지(틀렸는지) 측정하는 손실 함수.	Cross-Entropy Loss	로지스틱 회귀의 손실 함수.
정규화	모델이 너무 복잡해지는 것(과적합)을 방지하기 위해 '벌칙'을 주는 기법.	Regularization	L1(Lasso), L2(Ridge) 가 있음.
사전 확률 (Prior)	데이터를 보기 전, 내가 이미 가지고 있던 믿음(지식)의 분포.	Prior Distribution	"경험상 아마 이럴 것이다."
사후 확률 (Posterior)	사전 믿음(Prior)을 데이터(Likelihood)로 업데이트한 후의 새로운 믿음.	Posterior Distribution	베이지안 추론의 최종 목표.
켤레 사전분포	사전분포와 사후분포가 같은 종류의 분포가 되는 편리한 조합.	Conjugate Prior	예: 베타(Prior) + 이항(Data) = 베타(Posterior)
계층 모델	데이터가 그룹(계층) 구조를 가질 때 (예: 학생-학교), 이를 모델링에 반영하는 기법.	Hierarchical Model	'부분적으로 정보 공유'
MCMC	사후 확률 분포가 복잡해서 수식으로 풀 수 없을 때, 샘플링(무작위 점찍기)으로 분포를 근사하는 방법.	Markov Chain Monte Carlo	베이지안 모델의 핵심 도구.

3 핵심 개념 및 원리

3.1 분류(Classification)의 시작: 왜 선형 회귀는 안될까?

'펭귄의 성별(Male/Female)'이나 '주택 구매 여부(Yes/No)'처럼, 결과가 범주형(Categorical)인 문제를 풀어야 할 때가 있습니다.

- (1) **한 줄 요약:** 선형 회귀는 출력이 $(-\infty, +\infty)$ 로 뻗어 나가기 때문에, 0과 1 사이의 확률을 예측하는 분류 문제에 부적합합니다.
- (2) **직관적 예시:** 펭귄의 부리 길이(x)로 성별(y)을 예측한다고 가정합시다. Male=1, Female=0으로 두고 선형 회귀($y = \beta_0 + \beta_1 x$)를 적용하면, 부리가 아주 긴 펭귄은 y 값이 1.5, 아주 짧은 펭귄은 -0.2가 될 수 있습니다. 하지만 '확률 150%'나 '확률 -20%'는 말이 되지 않습니다.
- (3) **기술적 설명:** 우리는 예측값이 항상 0과 1 사이에 머무르도록 강제할 필요가 있습니다. 이때 등장하는 것이 시그모이드(Sigmoid) 또는 로지스틱(Logistic) 함수입니다.

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

이 함수는 입력 z 가 아무리 큰/작은 값이 들어와도, 출력은 항상 0과 1 사이의 값을 가집니다.

3.2 로지스틱 회귀 (Logistic Regression)

로지스틱 회귀는 선형 회귀의 예측값($z = \beta_0 + \beta_1 x$)을 시그모이드 함수에 통과시켜 확률로 변환하는 모델입니다.

- (1) **한 줄 요약:** 선형 모델의 출력을 0 1 사이의 확률로 압축하여 분류 문제를 푸는 모델입니다.
- (2) **수식의 변환(확률, 오즈, 로그 오즈):**
 1. **확률 (Probability):** 우리가 원하는 것. $P(Y = 1|X) = p$. 범위: $[0, 1]$.

$$p = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X)}}$$

2. **오즈 (Odds):** 성공 확률 / 실패 확률. $Odds = \frac{p}{1-p}$. 범위: $[0, \infty]$. (확률 0.5는 오즈 1, 확률 0.8은 오즈 4)

$$\frac{p}{1-p} = e^{\beta_0 + \beta_1 X}$$

3. **로그 오즈 (Log-odds):** 오즈에 로그를 씌운 것. 범위: $(-\infty, +\infty)$.

$$\ln\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 X$$

- (3) **기술적 설명(계수 해석):** 로지스틱 회귀의 핵심은 로그 오즈를 선형 예측하는 것입니다.
 β_1 의 해석: X 가 1단위 증가할 때, 로그 오즈가 β_1 만큼 증가합니다.
현실적 해석(오즈비, Odds Ratio): X 가 1단위 증가할 때, 오즈(Odds)가 e^{β_1} 배 증가합니다.

□ 예제:

NBA 농구공 예시: 슛 거리(Distance)로 성공 여부(Success=1)를 예측하는 모델을 만들었습니다.

$$\ln\left(\frac{p}{1-p}\right) = 0.796 - 0.0474 \times \text{Distance}$$

- 절편 (0.796): 거리가 0일 때(골대 바로 밑)의 로그 오즈입니다. 이때의 성공 확률 $p = \frac{1}{1+e^{-0.796}} \approx 0.689$ (약 69%).
- 계수 (-0.0474): 거리가 1피트 증가할 때마다, 성공에 대한 로그 오즈가 0.0474만큼 감소합니다.
- 오즈비: $e^{-0.0474} \approx 0.954$. 즉, 거리가 1피트 늘어날 때마다 성공 오즈가 약 0.954배가 됩니다 (약 4.6% 감소). 멀어질수록 슛이 어려워진다는 직관과 일치합니다.

3.3 모델 평가와 정규화

- 모델 학습 (MLE): 로지스틱 회귀는 최대우도추정법 (MLE, Maximum Likelihood Estimation)으로 학습합니다. 이는 주어진 데이터가 나타날 확률을 최대화하는 β 값을 찾는 과정이며, 이는 교차 엔트로피 손실 (Cross-Entropy Loss)을 최소화하는 것과 수학적으로 동일합니다.
- 평가 (ROC-AUC): 정확도(Accuracy)는 데이터가 불균형 할 때(예: 90%가 Male, 10%가 Female) 모델 성능을 제대로 평가하기 어렵습니다. 이때 ROC 커브 (Receiver Operating Characteristic Curve)와 AUC (Area Under the Curve)를 사용합니다.
 - ROC 커브: 모든 가능한 임계값(Threshold)에 대해 '가짜 양성 비율(FPR)' 대비 '진짜 양성 비율(TPR)'을 그린 그래프입니다.
 - AUC: 이 커브의 아래 면적. 1에 가까울수록 모델이 양성/음성을 잘 구별한다는 의미입니다. 0.5는 무작위 추측(동전 던지기)과 같습니다.
- 정규화 (Regularization): 모델이 너무 복잡해져 훈련 데이터에만 과적합(Overfitting)하는 것을 막기 위해, 모델의 β 계수 크기에 벌칙(Penalty)을 부과합니다.

주의사항

Scikit-Learn의 ‘C’ 파라미터 함정

- 수학에서 정규화 강도는 λ (람다)로 표현합니다. λ 가 클수록 강한 정규화입니다.
- scikit-learn의 LogisticRegression(C=...)에서 C 는 $1/\lambda$ 입니다.
- 즉, C 가 작을수록 ($C = 0.01$) → 정규화가 강해지고, C 가 클수록 ($C = 100$) → 정규화가 약해집니다. 이는 직관과 반대이므로 반드시 기억해야 합니다.
- 정규화 사용 시, 모든 변수의 스케일을 맞추기 위해 StandardScaler 사용이 거의 필수적입니다.

3.4 베이지안 추론 (Bayesian Inference)

데이터 과학에서 모델의 파라미터(β)를 추정하는 두 가지 큰 관점이 있습니다.

- 최대우도법 (Frequentist): ”파라미터(β)는 고정된 값이다. 우리가 가진 데이터로 그 값을 가장 잘 설명하는 ‘단 하나의 점’을 찾자.” (우리가 배운 sklearn의 .fit() 이 여기에 해당)
- 베이지안 (Bayesian): ”파라미터(β)도 불확실성을 가진 ‘확률 분포’다. 데이터를 보기 전의 믿음 (Prior)을 데이터 (Likelihood)를 통해 업데이트하여, 더 정교한 믿음 (Posterior)을 얻자.”

$$\underbrace{P(\theta|D)}_{\text{사후 확률}} \propto \underbrace{P(D|\theta)}_{\text{우도 (Likelihood)}} \times \underbrace{P(\theta)}_{\text{사전 확률 (Prior)}}$$

3.5 캘레 사전분포 (Beta-Binomial 모델)

베이지안 계산은 복잡하지만, 특정 조합에서는 수식이 아주 깔끔하게 펼어집니다.

- (1) 한 줄 요약: 사전분포와 사후분포가 같은 종류의 분포가 되는 (계산이 편리한) 마법 같은 조합입니다.
- (2) 직관적 예시 (동전 던지기):
 - 문제: 동전의 앞면이 나올 확률 p 를 추정하고 싶다.
 - 데이터 (Likelihood): p 를 모르는 상태로 10번(n) 던져서 앞면이 7번($\sum y_i$) 나왔다. (이항분포/베르누이분포)
 - 사전 믿음 (Prior): ”나는 이 동전이 공정할 것 같아 ($p \approx 0.5$).” 이 믿음을 베타 분포(Beta Distribution)로 표현합니다. 베타 분포는 0과 1 사이 값의 불확실성을 표현하기 완벽한 분포입니다.
 - $Beta(a_0, b_0)$ 의 의미: a_0 는 ’사전의 가장 앞면 개수’, b_0 는 ’사전의 가장 뒷면 개수’로 해석할 수 있습니다. ”공정할 것 같다”는 $Beta(a_0 = 1, b_0 = 1)$ (모든 값 동일하게 가능) 또는 $Beta(a_0 = 5, b_0 = 5)$ (0.5 근처일 거라 강하게 믿음) 등으로 표현할 수 있습니다.
- (3) 기술적 설명 (Posterior): 놀랍게도, 사전분포 $Beta(a_0, b_0)$ 와 이항분포 데이터를 결합하면, 사후분포는 정확히

$$Posterior \sim Beta(a_0 + \sum y_i, b_0 + (n - \sum y_i))$$

가 됩니다. (사전의 앞면 개수 + 실제 앞면 개수, 사전의 뒷면 개수 + 실제 뒷면 개수)

사후 평균 (Posterior Mean): 이 사후분포의 평균(p 의 점 추정치)은

$$\hat{p}_{PM} = \frac{a_0 + \sum y_i}{a_0 + b_0 + n}$$

이는 사전 믿음의 평균과 데이터의 평균(MLE) 사이의 가중 평균이 됩니다. 데이터(n)가 많아질수록 사전 믿음(a_0, b_0)의 영향력은 줄어들고 데이터의 힘이 강해집니다.

3.6 계층 모델 (Hierarchical Models)

- (1) 한 줄 요약: 데이터가 그룹 구조(예: 선수별 슛)를 가질 때, 각 그룹이 완전히 다르지도(No Pooling), 완전히 같지도(Complete Pooling) 않다고 보고, ’부분적으로 정보를 공유’(Partial Pooling)하는 현명한 모델입니다.
- (2) 직관적 예시 (NBA 선수 슛 예측):
 - 문제: 슛 거리(Distance)와 선수(Player)를 이용해 슛 성공을 예측.
 - 접근 1 (Complete Pooling): ”모든 선수는 같다.” 선수를 무시하고 하나의 모델 $\ln(p/(1-p)) = \beta_0 + \beta_1 \text{Dist}$ 를 씁니다. (Underfitting)
 - 접근 2 (No Pooling / OLS): ”모든 선수는 완전히 다르다.” 선수를 One-Hot 인코딩하여 수백 개의 β 계수를 만듭니다.
 - 문제점: 맥 맥클링(Mac McClung) 선수가 2번 슛해서 2번 다 실패(0%) 했다면, 이 모델은 그의 β

계수를 $-\infty$ 에 가깝게 추정하여 ”그는 0

- 접근 3 (Hierarchical / Partial Pooling): ”선수들은 저마다 다르지만, 결국 모두 ’NBA 선수’라는 큰 그룹에 속한다.”
- (3) 기술적 설명: 각 선수(j)마다 고유의 절편(α_j)을 갖는다고 가정합니다.

$$\ln\left(\frac{p_{ij}}{1-p_{ij}}\right) = \alpha_j + \beta_1 \text{Distance}_{ij}$$

하지만 이 α_j 들이 완전히 따로 노는 게 아니라, 모든 선수의 평균 실력(α_0)과 실력의 편차(σ_α^2)를 따르는 정규분포에서 추출되었다고 가정합니다.

$$\alpha_j \sim N(\alpha_0, \sigma_\alpha^2)$$

결과 (수축, Shrinkage):

- 데이터가 많은 선수(예: 스테판 커리)는 자신의 데이터로 α_j 가 결정됩니다.
- 데이터가 적은 선수(예: 맥 맥클링)는 모델이 ”데이터가 2개뿐이라 못 믿겠다”고 판단하여, α_j 를 전체 평균(α_0) 쪽으로 강하게 끌어당깁니다(수축).
- 즉, 맥 맥클링의 0

3.7 MCMC와 사후분포 샘플링

- 문제 상황: 베이지안 로지스틱 회귀나 계층 모델은 결례성이 깨집니다. (사전분포 $N(\cdot)$ + 우도 $\text{Logistic}(\cdot) \rightarrow ???$). 사후분포(Posterior)가 매우 복잡한 형태가 되어 수식으로 풀 수 없습니다.
- (1) 한 줄 요약: 사후분포라는 복잡한 산맥의 지도를 그릴 수 없을 때, 그 산맥을 무작위로 돌아다니면서 (MCMC) 수천, 수만 개의 발자국(Samples)을 찍어, 그 발자국 밀도로 산맥의 형태(분포)를 역추적하는 기법입니다.
- (2) 직관적 예시 (Rejection Sampling):
 - 목표: 복잡한 쿠키 모양($f(x)$)의 반죽을 찍어내고 싶다.
 - 문제: 쿠키틀이 없지만, 그 쿠키틀을 덮는 네모난 상자($Mg(x)$)는 가지고 있다.
 - 방법: 1. 네모난 상자($g(x)$) 안에서 무작위로 위치(x)를 하나 찍는다. 2. 그 위치(x)에서 쿠키 반죽의 높이($f(x)$)와 상자 높이($Mg(x)$)를 비교한다. 3. $f(x)/(Mg(x))$ 의 확률로 그 위치를 ’채택(Accept)’ 한다. (즉, 반죽이 두꺼운 곳은 채택될 확률이 높음)
 - 결과: 수천 번 반복하면, 채택된 점들의 분포가 정확히 쿠키 모양($f(x)$)을 따르게 됩니다.
- (3) 기술적 설명 (Metropolis-Hastings): Rejection Sampling은 고차원에서 매우 비효율적입니다. (거의 모든 점이 거절됨) Metropolis-Hastings는 ’무작위 산책(Random Walk)’을 통해 더 효율적으로 샘플을 뽑습니다.

알고리즘 (등산가 비유): 1. 현재 위치($\theta^{(t)}$)에 서 있습니다. (현재 위치의 높이는 $f(\theta^{(t)})$) 2. 다음 발걸음을 아무 데나 제안합니다. (θ^*) (제안 위치의 높이는 $f(\theta^*)$) 3. 비교: 제안된 곳(θ^*)이 현재($\theta^{(t)}$) 보다 높으면 (Uphill) \rightarrow 무조건 이동합니다. 4. 비교: 제안된 곳이 현재보다 낮으면 (Downhill) $\rightarrow R = f(\theta^*)/f(\theta^{(t)})$ (높이의 비율) 만큼의 확률로 이동합니다. (낮아도 가끔은 이동해야 골짜기에 갇히지 않음) 5. 이동하지 않으면, 현재 위치에 한 번 더 머무릅니다(발자국 하나 더 찍음).
이 과정을 수천 번 반복하면, 등산가가 머물렀던 위치(발자국)의 분포가 정확히 산맥의 형태(사후분포)와 일치하게 됩니다.

4 절차 및 방법 (scikit-learn 파이프라인)

실제 데이터 분석에서는 scikit-learn을 사용하여 분류 모델을 효율적으로 구축합니다. 다음은 펭귄의 성별을 예측하는 전형적인 머신러닝 작업 흐름입니다.

1단계: 데이터 준비 및 전처리 (EDA)

- **데이터 로드:** 데이터를 불러옵니다. (예: 펭귄 데이터셋)
- **NaN 값 확인:** `df.isna().sum()`으로 결측치를 확인합니다.
- **NaN 값 처리:**
 - 예측 변수(Feature)의 NaN: 행 전체를 삭제(`dropna()`)하거나, 평균/최빈값 등으로 채웁니다 (`fillna()`).
 - 대상 변수(Target)의 NaN: 해당 행은 모델 학습이나 평가에 사용할 수 없으므로, 보통 `dropna(subset=['target'])`를 통해 삭제합니다.
- **대상 변수 인코딩:** 'Male', 'Female'과 같은 문자열은 모델이 이해할 수 없습니다. `LabelEncoder` 등을 사용해 0, 1과 같은 숫자로 변환합니다.

2단계: 훈련/테스트 데이터 분리

- **목적:** 모델이 처음 본 데이터(Test)에서 얼마나 잘 작동하는지 평가하기 위해 데이터를 분리합니다.
- **불균형 데이터 문제:** 만약 90%가 Male이고 10%가 Female인 데이터를 무작위로 분리하면, 테스트셋에 Female이 하나도 포함되지 않을 수 있습니다.
- **해결 (Stratified Split):** `train_test_split`의 `stratify=` 옵션에 대상 변수(y)를 지정합니다. 이는 훈련셋과 테스트셋의 0/1 비율을 원본 데이터의 비율과 동일하게 유지시킵니다.

3단계: 파이프라인 구축 및 하이퍼파라미터 튜닝 (GridSearch)

모델 학습 과정(스케일링, 모델링)을 하나로 묶고, 최적의 파라미터를 찾습니다.

주의사항

GridSearch와 K-Fold 교차 검증

모델의 성능은 데이터를 어떻게 나누었는지에 따라 우연히 좋거나 나쁘게 나올 수 있습니다. **K-Fold 교차 검증 (Cross-Validation)**은 데이터를 K개의 랜덤으로 나눈 뒤, (K-1)개로 학습하고 1개로 검증하는 과정을 K번 반복하여 평균을 내는, 더 안정적인 성능 평가 방법입니다. `GridSearchCV`는 이 K-Fold 방식을 사용하여 각 하이퍼파라미터 조합의 성능을 평가합니다.

```

1 from sklearn.model_selection import train_test_split, KFold, GridSearchCV
2 from sklearn.preprocessing import StandardScaler, LabelEncoder
3 from sklearn.neighbors import KNeighborsClassifier
4 from sklearn.pipeline import Pipeline
5 import numpy as np
6
7 # --- 1. 데이터준비가정 () ---
8 # X, y = load_penguin_data()
9 # y = LabelEncoder().fit_transform(y) # 'Male'/Female' -> 0/1

```

```

10
11 # --- 2. 데이터분리 ---
12 # X_train, X_test, y_train, y_test = train_test_split(
13 #     X, y, test_size=0.2, stratify=y, random_state=42
14 # )
15
16 # --- 3. 파이프라인 및 GridSearch ---
17
18 # 3.1. 수행할 작업정의스케일러 ( + 모델)
19 pipe = Pipeline([
20     ('scaler', StandardScaler()),
21     ('knn', KNeighborsClassifier())
22 ])
23
24 # 3.2. 탐색할 하이퍼파라미터 그리드 정의
25 # 모델이름 '파라미터이름_-' 형식
26 param_grid = {
27     'knn__n_neighbors': [3, 5, 7, 9]
28 }
29
30 # 3.3. 교차검증 (CV) 방법정의
31 kfold = KFold(n_splits=5, shuffle=True, random_state=42)
32
33 # 3.4. GridSearch 객체생성
34 # cv=kfold : 5-fold로 CV 성능평가
35 # scoring='accuracy' : 정확도를 기준으로 최적 모델 선택
36 # n_jobs=-1 : 모든 CPU 코어를 사용해 병렬로 탐색
37 grid_search = GridSearchCV(
38     estimator=pipe,
39     param_grid=param_grid,
40     cv=kfold,
41     scoring='accuracy',
42     n_jobs=-1
43 )
44
45 # 3.5. 훈련데이터로 탐색 시작
46 # grid_search.fit(X_train, y_train)
47
48 # --- 4. 결과확인 ---
49 # print(f"Best Score: {grid_search.best_score_}")
50 # print(f"Best Params: {grid_search.best_params_}")
51 # best_model = grid_search.best_estimator_
52
53 # --- 5. 최종테스트 ---
54 # test_accuracy = best_model.score(X_test, y_test)

```

Listing 1: scikit-learn 파이프라인과 GridSearch 예시

4단계: 결정 경계 (Decision Boundary) 시각화

모델이 2D 공간을 어떻게 나누고 있는지 시각화하면 모델을 직관적으로 이해할 수 있습니다.

- `np.meshgrid`: 2D 평면을 촘촘한 격자(Grid)로 나눕니다. x축 좌표 배열과 y축 좌표 배열을 받아, 모든 (x, y) 좌표 쌍을 생성합니다.
- `model.predict`: 격자의 모든 점에 대해 모델이 예측(0 또는 1)을 수행합니다.
- `plt.contourf`: 예측 결과(0 또는 1)에 따라 격자점을 다른 색으로 칠하여, 모델이 만든 경계선을 시각화합니다.

5 실습, 코드 및 주요 함정

5.1 주요 함정 (Gotchas) 다시보기

주의사항

- Pandas `value_counts()`의 함정: `df['sex'].value_counts()`는 NaN(결측치)을 자동으로 무시하고 개수를 셹니다. 이로 인해 데이터가 누락되는 것을 인지하지 못할 수 있습니다. (e.g., Male 134, Female 132 = 266개. 실제 데이터 273개. 7개의 NaN이 무시됨)
- Scikit-Learn C 파라미터의 함정: $C = 1/\lambda$ 입니다. C가 작을수록 ($C = 0.01$) 규제가 강해집니다.
- 정규화(Regularization)와 스케일링: L1, L2 정규화는 계수의 크기에 벌칙을 줍니다. 만약 A 변수(1000 2000)가 B 변수(0 1)보다 스케일이 훨씬 크다면, A 변수의 계수는 불공평하게 큰 벌칙을 받게 됩니다. 따라서 정규화 전 StandardScaler는 필수입니다.
- 계층 모델의 과적합 방지 (Mac McClung 예시): '2번 슛, 2번 실패(0%)'라는 적은 데이터를 가진 선수에게 일반 OLS 모델은 '성공률 0%'라는 극단적인 결론을 내립니다 (과적합). 계층 모델은 이 선수의 데이터를 '평균 NBA 선수' 데이터와 혼합(Shrinkage)하여, '0%보다는 높지만 평균보다는 낮은' 합리적인 추정치를 제공합니다.

5.2 Bayesian Logistic Regression (pymc)

`scikit-learn`이 아닌 베이지안 프레임워크 `pymc`를 사용하면 모델을 어떻게 구축할까요? Skittles 맷테스트 예제를 통해 MCMC가 어떻게 작동하는지 살펴봅니다.

```

1 import pymc as pm
2 import numpy as np
3 import arviz as az
4
5 # --- 1. 데이터가정 () ---
6 # 가지 8 다른맛 (x)에 대해, 명 n 중명이 y 맛있다고 "" 응답
7 flavoring = np.array([1.69, 1.72, 1.75, 1.78, 1.81, 1.83, 1.86, 1.88])
8 n_testers = np.array([59, 60, 62, 56, 63, 59, 62, 60])
9 n_loved = np.array([6, 13, 18, 28, 52, 52, 61, 60])
10
11 # --- 2. 모델정의 (with pm.Model() as ... ) ---
12 with pm.Model() as skittles_model:
13
14     # --- 2.1. 사전확률 (Priors) 정의 ---
15     # 와 alpha 가 beta 무엇인지모른다는약한믿음매우 (넓은정규분포)
16     alpha = pm.Normal("alpha", mu=0, sigma=100)
17     beta = pm.Normal("beta", mu=0, sigma=100)
18
19     # --- 2.2. 모델로직 (Deterministic Link) ---
20     # 로지스틱회귀의선형부분로그 (오즈)
21     logit_p = alpha + beta * flavoring
22
23     # 시그모이드함수를통과시켜확률로 p 변환
24     p = pm.Deterministic("p", pm.math.invlogit(logit_p))

```

```

25
26 # --- 2.3. 우도 (Likelihood) 정의 ---
27 # 우리의 관측값 (y)이 모델 (p)로부터 어떻게 생성되었는가 ?
28 # 명 n 중의 p 확률로 명이 y 성공 -> 이항분포 !
29 y_obs = pm.Binomial("y_obs", n=n_testers, p=p, observed=n_loved)
30
31
32 # --- 3. MCMC 샘플링 실행 ---
33 # Metropolis-Hastings 또는 (더발전된 NUTS) 알고리즘이
34 # 복잡한 사후분포에서 샘플을 추출합니다 .
35 # tune: 예열 (burn-in) 단계 / draws: 실제 저장할 샘플 수
36 with skittles_model:
37     trace = pm.sample(draws=2000, tune=2000, return_inferencedata=True)
38
39 # --- 4. 결과 분석 ---
40 # az.plot_trace(trace, var_names=["alpha", "beta"])
41 # az.summary(trace, var_names=["alpha", "beta"])

```

Listing 2: pymc를 사용한 베이지안 로지스틱 회귀 (Skittles 예시)

5.3 MCMC 결과 해석 (Trace Plot)

MCMC 샘플링 후, `az.summary(trace)` 는 다음과 같은 요약 통계량을 보여줍니다.

- **mean, sd:** 추정된 사후분포의 평균과 표준편차. (즉, α 는 약 -60.4, β 는 약 34.1)
- **hdi_3%, hdi_97%:** 94% 신뢰구간(Credible Interval). 파라미터가 이 범위 안에 있을 확률이 94%라는 의미입니다.
- **ess_bulk, ess_tail:** 유효 샘플 크기. 2000번 뽑았지만, 샘플 간 상관관계 때문에 실제로는 약 1000개 정도의 독립적인 정보만 얻었다는 의미. (높을수록 좋음)
- **r_hat (\hat{R}):** 가장 중요한 진단 지표. 여러 개의 MCMC 체인(무작위 템색가)들이 모두 같은 결과(분포)로 수렴했는지를 봅니다. 정확히 1.0이거나 1.01 미만이어야만 결과를 신뢰할 수 있습니다. 1.1 이상이면 샘플링이 실패했다는 뜻입니다.

6 학습 체크리스트

- 분류 문제에 왜 선형 회귀 대신 로지스틱 회귀를 사용해야 하는지 설명할 수 있는가?
- 로지스틱 회귀의 계수(β)가 확률이 아닌 '로그 오즈'에 대한 것임을 이해하는가?
- scikit-learn에서 C 파라미터가 작을수록 정규화가 강해진다는 것을 아는가?
- 데이터가 불균형 할 때 왜 정확도(Accuracy) 대신 ROC-AUC를 사용해야 하는지 아는가?
- np.meshgrid와 plt.contourf를 사용해 결정 경계를 그리는 원리를 이해하는가?
- 베이지안 추론의 3요소 (Prior, Likelihood, Posterior)의 관계를 설명할 수 있는가?
- 캘레 사전분포(예: 베타-이항)가 왜 편리한지 설명할 수 있는가?
- 계층 모델이 '부분적 정보공유(Partial Pooling)'를 통해 과적합을 방지하는 원리(Shrinkage)를 이해하는가?
- 사후분포를 수식으로 풀 수 없을 때 MCMC 샘플링을 사용하는 이유를 아는가?
- MCMC 결과에서 \hat{R} (r_hat) 지표가 1.0에 가까워야 하는 이유를 아는가?

7 FAQ (자주 묻는 질문)

Q: 왜 KNN이나 로지스틱 회귀 전에 StandardScaler를 써야 하나요?

A: 이 모델들은 '거리'나 '크기'에 민감하기 때문입니다.

KNN (K-Nearest Neighbors): "가까운 이웃"을 찾을 때 유clidean 거리를 사용합니다. 만약 A 변수(키, 150~190cm)가 B 변수(시험 성적, 0~100점)보다 스케일이 크다면, A 변수가 B 변수보다 거리에 훨씬 큰 영향을 미치게 됩니다. 이는 모델이 사실상 A 변수(키)에만 의존하게 만듭니다.

로지스틱 회귀 (정규화 사용 시): L1/L2 정규화는 계수(β)의 '크기'에 벌칙을 줍니다. 스케일이 큰 변수(A)는 작은 계수($\beta_A \approx 0.01$)를, 스케일이 작은 변수(B)는 큰 계수($\beta_B \approx 10$)를 가질 수 있습니다. 정규화는 β_B 에 훨씬 큰 벌칙을 주게 되어 불공평합니다.

StandardScaler는 모든 변수의 평균을 0, 표준편차를 1로 만들어, 모든 변수가 공평하게 모델에 기여하도록 만듭니다.

Q: 로그 오즈, 오즈, 확률... 너무 헷갈립니다.

A: 변환의 목적은 '범위'를 맞추는 것입니다.

- 선형 회귀의 출력 (z): $-\infty$ 에서 $+\infty$ 까지 모든 값을 가집니다.
- 확률 (p): 0에서 1 사이의 값만 가져야 합니다.

이 둘을 연결하기 위해, 확률 p 를 z 와 같은 범위로 바꾸는 변환이 필요합니다.

- 확률 \rightarrow 오즈: $Odds = p/(1-p)$. 범위가 $[0, 1]$ 에서 $[0, \infty]$ 로 바뀝니다. (0보다 작은 값이 없어짐)
- 오즈 \rightarrow 로그 오즈: $LogOdds = \ln(Odds)$. 범위가 $[0, \infty]$ 에서 $[-\infty, +\infty]$ 로 바뀝니다.

이제 $LogOdds$ 는 선형 회귀의 출력 z 와 범위가 같아졌습니다!

$$\underbrace{\ln\left(\frac{p}{1-p}\right)}_{\text{범위: } (-\infty, \infty)} = \underbrace{\beta_0 + \beta_1 X}_{\text{범위: } (-\infty, \infty)}$$

로지스틱 회귀는 확률 p 가 아닌, 로그 오즈를 선형적으로 예측하는 모델입니다.

Q: 베이지안 모델은 복잡한데 왜 굳이 사용하나요?

A: 베이지안 모델은 '불확실성의 정량화'와 '계층 구조'라는 강력한 무기를 제공합니다.

- 불확실성 정량화:** 최대우도법(MLE)은 " $\beta_1 = 5.0$ "이라는 '점 추정'을 줍니다. 베이지안 모델은 " β_1 은 평균 5.0, 표준편차 0.3인 정규분포를 따른다"처럼 '분포 추정'을 줍니다. 즉, β_1 이 4.5에서 5.5 사이에 있을 확률이 95%라고 말할 수 있습니다. 이는 우리의 추정이 얼마나 확실한지 알려줍니다.
- 계층 모델 (Shrinkage):** 위에서 설명한 NBA 선수 예시처럼, 데이터가 적은 그룹(선수)이 과적합되는 것을 막아주는 매우 강력하고 합리적인 방법입니다. 이는 일반적인 OLS나 MLE로는 구현하기 매우 어렵습니다.

8 빠르게 훑어보기 (핵심 요약)

로지스틱 회귀 (Logistic Regression)

- 용도:** 선형 회귀($y = ax + b$)의 출력을 시그모이드 함수 $\frac{1}{1+e^{-z}}$ 에 넣어 0~1 사이의 확률로 변환, '분류' 문제에 사용.
- 핵심:** X 가 변할 때 '확률'이 선형적으로 변하는 게 아니라, '로그 오즈' $\ln(p/(1-p))$ 가 선형적으로 변한다.
- 해석:** β_1 계수는 X 가 1 증가할 때 '오즈'가 e^{β_1} 배 변한다는 의미.
- 손실 함수:** 교차 엔트로피 (Cross-Entropy Loss)

정규화 (Regularization)

- 용도:** 모델이 훈련 데이터에만 과적합(Overfitting)되는 것을 방지.
- 방법:** 계수(β)의 크기에 벌칙(Penalty)을 부과.
- L1 (Lasso):** $|\beta|$ 에 비례. 중요하지 않은 변수의 β 를 0으로 만들고 (변수 선택).
- L2 (Ridge):** β^2 에 비례. 모든 β 를 0에 가깝게 줄임 (부드러운 모델).
- 주의:** sklearn의 C 는 $1/\lambda$. C 가 작을수록 규제가 강하다.

베이지안 추론 (Bayesian Inference)

- 핵심 공식:** 사후 확률(Posterior) \propto 우도(Likelihood) \times 사전 확률(Prior)
- 의미:** 나의 기존 믿음(Prior)을 데이터(Likelihood)를 통해 업데이트(Posterior) 한다.
- 켤레성 (Conjugacy):** $Beta(Prior) + Binomial(Data) = Beta(Posterior)$ 처럼, 계산이 깔끔하게 떨어지는 조합.
- 사후 평균:** $\hat{p}_{PM} = \frac{a_0 + \sum y_i}{a_0 + b_0 + n}$. 사전 믿음과 데이터의 가중 평균.

계층 모델 (Hierarchical Model)

- 용도:** 데이터가 그룹(예: 학생-학급, 선수-팀)으로 묶여 있을 때 사용.
- 개념:** '완전 독립'(No Pooling)과 '완전 동일'(Complete Pooling) 사이의 합리적 절충안.
- 핵심 (Partial Pooling):** 각 그룹(j)의 파라미터(α_j)가 공통 분포 $N(\alpha_0, \sigma_\alpha^2)$ 에서 나왔다고 가정.
- 효과 (Shrinkage):** 데이터가 적은 그룹(Mac McClung)의 추정치를 전체 평균(α_0) 쪽으로 수축(Shrinkage) 시켜 과적합을 방지.

MCMC (Markov Chain Monte Carlo)

- 용도:** 계층 모델처럼 복잡해서 수식으로 풀 수 없는 사후 확률(Posterior) 분포를 '샘플링'을 통해 근사적으로 알아내는 방법.
- Metropolis-Hastings:** '무작위 산책' 알고리즘.
- 로직:** (1) 다음 위치 제안 \rightarrow (2) 현재보다 높으면(Uphill) 무조건 이동 \rightarrow (3) 현재보다 낮으면(Downhill) 확률적으로 이동.

- **결과 해석:** 수렴 진단 지표 \hat{R} (`r_hat`)이 반드시 1.0에 가까워야 함.

December 10, 2025

- 강의명: CS109A: 데이터 과학 입문
- 주차: Lecture 17
- 교수명: Pavlos Protopapas, Kevin Rader, Chris Gumb
- 목적: Lecture 17의 핵심 개념 학습

▣ 핵심 요약

본 강의는 MCMC (Markov Chain Monte Carlo)와 결측치 처리 방법을 다룹니다. MCMC는 복잡한 확률 분포에서 샘플을 추출하는 강력한 방법이며, 메트로폴리스-헤이스팅스 알고리즘의 원리와 번인(Burn-in), \hat{R} 수렴 진단 방법을 학습합니다. 또한 데이터셋의 결측치를 적절히 처리하는 대치(Imputation) 기법과 그 한계를 이해합니다.

Contents

1	핵심 용어 정리	2
2	MCMC 상세 설명	3
2.1	\hat{R} 통계량	3
3	학습 체크리스트	3
4	FAQ	3
5	결측치 대치 방법	3
5.1	단순하지만 나은 방법	3
5.2	가장 좋은 방법	3

1 핵심 용어 정리

Table 1: Lecture 17 핵심 용어

용어	원어	쉬운 설명	비고 (등장 위치)
MCMC	Markov Chain Monte Carlo	마르코프 체인을 이용해 복잡한 분포에서 샘플을 뽑는 방법.	MCMC
메트로폴리스-헤이스팅스	Metropolis-Hastings	MCMC의 대표적인 알고리즘. "언덕 오르기" 비유.	MCMC
번인 (Burn-in)	Burn-in	MCMC가 안정적인 분포에 도달하기 전까지의 초기 샘플(버려야 함).	MCMC
\hat{R} (R-hat)	R-hat	MCMC가 수렴했는지 판단하는 지표. 1.0에 가까워야 함.	MCMC
결측치	Missing Data	데이터셋에 값이 누락된 상태. (e.g., 'NaN', 'NA')	결측치
대치 (대체)	Imputation	결측치를 추정된 값으로 채우는 것.	결측치

2 MCMC 상세 설명

2.1 R-hat (\hat{R}) 통계량

MCMC는 보통 여러 개의 체인(로봇 여러 대)을 동시에 돌립니다. \hat{R} 은 이 체인들이 모두 같은 산봉우리(안정적인 분포)에 수렴했는지 확인하는 지표입니다.

주의사항

$\hat{R} \approx 1.0$ 이어야 합니다.

만약 $\hat{R} > 1.1$ (혹은 1.05) 이라면, 체인들이 아직 수렴하지 못했거나(e.g., 벤인이 부족), 서로 다른 곳(e.g., 어떤 로봇은 A봉우리, 어떤 로봇은 B봉우리)에 가있다는 뜻입니다.

3 학습 체크리스트

- 결례성(Conjugacy)의 의미와, 결례성이 깨졌을 때(e.g., 로지스틱 회귀) 왜 MCMC가 필요한지 설명할 수 있는가?
- MCMC 추적(Trace) 플롯이 안정적인지(수렴했는지) 시각적으로 판단할 수 있는가?
- \hat{R} (R-hat) 통계량이 1.0에 가까워야 하는 이유는 무엇인가?
- 결측치를 단순히 제거('dropna')하면 안 되는 이유는 무엇인가? (편향 문제)

4 FAQ

Q: \hat{R} (R-hat)이 1.0이 아니고 1.3 처럼 나오면 어떻게 해야 하나요?

A: 이는 MCMC가 ”수렴에 실패했음“을 알리는 심각한 경고입니다. 여러 대의 로봇(체인)이 서로 다른 산봉우리에 가있거나, 아직 산을 다 오르지도 못했다는 뜻입니다.

해결 방법:

1. 벤인(Burn-in) 기간을 늘린다
2. 체인 실행 횟수를 늘린다
3. 더 좋은 초기값(Starting Point)을 설정한다

5 결측치 대치 방법

5.1 단순하지만 나은 방법

- 지시 변수(Indicator): ‘X_imputed’ (0으로 채움) + ‘X_was_missing’ (1/0)

5.2 가장 좋은 방법

- 확률론적 모델 대치 (Stochastic Imputation)
- Imputed Value = Model.Predict() + Random Noise(ϵ)

- (분류 문제의 경우: ‘predict_proba()‘ 결과로 편향된 동전 던지기)

December 10, 2025

- 강의명: CS109A: 데이터 과학 입문
- 주차: Lecture 18
- 교수명: Pavlos Protopapas, Kevin Rader, Chris Gumb
- 목적: Lecture 18의 핵심 개념 학습

▣ 핵심 요약

이 문서는 CS109A 강의의 **결정 트리(Decision Tree)** 모델을 다룹니다. 결정 트리는 데이터를 분류하기 위해 ”예/아니오” 질문을 반복하는, 마치 스무고개와 같은 직관적인 모델입니다.

- **왜 필요한가?:** 로지스틱 회귀 등 선형 모델이 잘 구분하지 못하는 복잡하고 비선형적인 데이터 경계를 쉽게 처리할 수 있습니다.
- **어떻게 작동하는가?:** ”키가 6.5보다 큰가?” 같은 질문(규칙)을 통해 데이터를 반복적으로 분할하여, 각 영역이 하나의 클래스(예: 오렌지, 레몬)로만 구성되도록 합니다.
- **어떻게 학습하는가?:** 각 단계에서 데이터를 가장 ’순수하게’(즉, 한 종류로만) 나누는 최적의 질문(분할 기준)을 탐욕적(Greedy)으로 찾습니다.
- **주요 과제:** 트리가 너무 복잡해져 훈련 데이터에만 과적합(Overfitting)되는 것을 방지하기 위해 중지 조건(예: 트리의 최대 깊이 제한)이 반드시 필요합니다.

Contents

1 개요: 왜 로지스틱 회귀가 아닌 결정 트리인가?	2
2 용어 정리	3
3 핵심 원리 1: 결정 트리는 어떻게 작동하는가?	4
3.1 직관: 공학용 순서도 (Engineering Flowchart)	4
3.2 예측 (Prediction): 트리 순회 (Traversing the Tree)	4
4 핵심 원리 2: 트리는 어떻게 ”학습”하는가?	5
4.1 학습 목표: 영역 순수도 (Region Purity)	5
4.2 불순도 측정 기준 (Splitting Criteria)	5
4.3 기준 1: 분류 오류 (Classification Error)	5
4.4 기준 2: 지니 불순도 (Gini Impurity)	6

4.5	기준 3: 엔트로피 (Entropy)	6
4.6	불순도 지표 비교	7
4.7	학습 알고리즘: 탐욕적 알고리즘 (Greedy Algorithm)	7
5	핵심 원리 3: 과적합 방지 (Overfitting)	8
5.1	문제점: 멈추지 않는 트리	8
5.2	해결책: 중지 조건 (Stopping Conditions)	8
5.3	트리 성장 전략: Level-Order vs. Best-First	8
5.4	편향-분산 트레이드오프 (Bias-Variance Trade-off)	9
5.5	최적의 하이퍼파라미터 찾기	9
6	학습 체크리스트	10
7	FAQ (주요 질문 및 답변)	11
8	1페이지 요약: 결정 트리 핵심	12

1 개요: 왜 로지스틱 회귀가 아닌 결정 트리인가?

우리가 배운 로지스틱 회귀는 강력한 분류 모델이지만, 한계가 명확합니다. 로지스틱 회귀는 기본적으로 데이터 공간을 **하나의 선(또는 초평면)**으로 나누려고 시도합니다.

- **로지스틱 회귀가 잘하는 것:** 데이터가 선형적으로 잘 분리될 때 (예: 위도/경도 데이터에서 '농경지'와 '건조지'가 직선으로 명확히 나뉠 때)
- **로지스틱 회귀가 못하는 것:** 데이터의 경계가 복잡한 비선형(non-linear) 일 때.

예를 들어, 위 (우) 이미지처럼 농경지(초록 점)가 중앙에 모여 있고 건조지(흰 점)가 그 주위를 둘러싸고 있는 경우, 로지스틱 회귀는 직선 하나로 이 두 클래스를 제대로 분리할 수 없습니다.

물론 다항 회귀(Polynomial Regression)를 사용해 $x_1^2 + x_2^2 - 0.25 = 0$ 과 같은 원형 경계를 만들 수도 있지만, 데이터의 경계가 더 복잡해지면 (예: 4사분면에 흩어져 있는 경우) 이를 설명할 방정식을 찾는 것은 거의 불가능에 가깝습니다.

새로운 모델의 요구사항 (Wish List) 우리는 다음과 같은 특징을 가진 새로운 모델이 필요합니다.

1. 복잡한 결정 경계를 만들 수 있어야 합니다.
2. 모델의 결정 과정을 이해하고 해석하기 쉬워야 합니다. (Interpretability)
3. 계산이 효율적이고 빨라야 합니다.

이 모든 것을 만족하는 모델이 바로 결정 트리(Decision Tree)입니다.

2 용어 정리

결정 트리를 이해하기 위해 사용되는 주요 용어들입니다.

Table 1: 결정 트리 핵심 용어

용어	쉬운 설명	원어	비고
결정 트리	데이터를 분류하기 위해 스무고개처럼 질문을 나열한 나무 구조의 모델	Decision Tree	
루트 노드	트리가 시작되는 첫 번째 질문 노드 (나무의 뿌리)	Root Node	트리에 단 하나만 존재
내부 노드	루트와 리프 사이의 모든 중간 질문 노드 (나무의 가지)	Internal Nodes	
리프 노드	트리의 가장 마지막에 위치한 노드로, 최종 결정을 의미 (나뭇잎)	Leaf Nodes	'터미널 노드'라고도 함
분할	하나의 노드(데이터 영역)를 질문을 통해 2개 이상의 자식 노드로 나누는 과정	Split	
순회	새로운 데이터가 주어졌을 때, 루트 부터 리프까지 질문을 따라 내려가는 과정	Traversing the Tree	이 과정을 통해 예측 수행
순수도	특정 노드(영역)에 하나의 클래스만 존재하는 정도. (100% 순수 = 모두 같은 클래스)	Purity	불순도(Impurity)의 반대
불순도	특정 노드에 여러 클래스가 섞여 있 는 정도. (예: 50:50으로 섞인 상태)	Impurity	Gini, Entropy로 측정
과적합	트리가 너무 복잡해져서 훈련 데이터 의 '노이즈'까지 모두 암기한 상태	Overfitting	새 데이터에 대한 성능 저하
중지 조건	과적합을 막기 위해 트리 성장을 멈 추는 규칙 (예: 최대 깊이 제한)	Stopping Conditions	하이퍼파라미터

3 핵심 원리 1: 결정 트리는 어떻게 작동하는가?

3.1 직관: 공학용 순서도 (Engineering Flowchart)

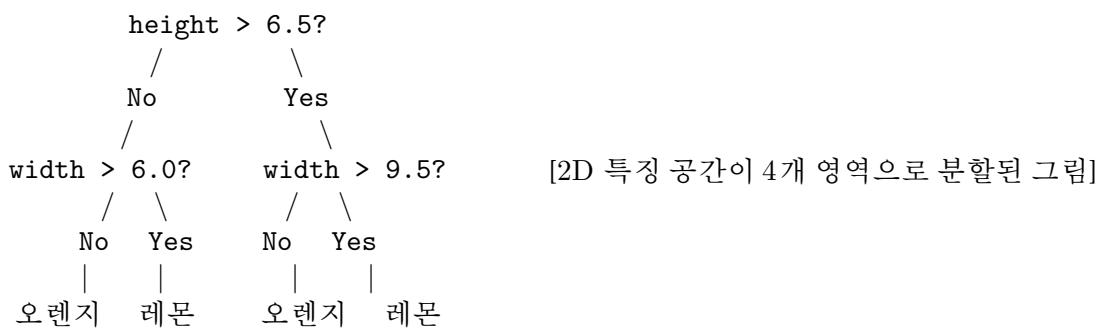
결정 트리는 우리가 일상에서 문제를 해결하는 방식과 매우 유사합니다. ”공학용 순서도” 예시를 살펴봅시다.

이 순서도는 일련의 이진(binary) 질문을 통해 최종 결론(WD-40, 덱트 테이프, 문제 없음)에 도달합니다. 결정 트리는 이 아이디어를 데이터 분류에 그대로 적용합니다.

3.2 예측 (Prediction): 트리 순회 (Traversing the Tree)

결정 트리가 이미 ”학습”되었다고 가정해봅시다. 예를 들어, 과일의 height(높이)와 width(너비)를 보고 ’레몬’과 ’오렌지’를 분류하는 트리입니다.

Figure 1: 결정 트리(좌)와 그로 인한 특징 공간 분할(우)



좌측의 트리는 우측의 2D 특징 공간(Feature Space)을 축에 평행한(axis-parallel) 직선들로 분할하는 것과 같습니다. 각 리프 노드(최종 결정)는 특징 공간의 사각형 영역 하나에 해당합니다.

▣ 예제:

새로운 과일 예측하기

새로운 과일이 들어왔습니다: (height = 5.9, width = 5.8)

이 과일은 레몬일까요, 오렌지일까요?

이 예측 과정을 ”트리 순회(Traversing the Tree)”라고 부릅니다.

1. 루트 노드: height > 6.5인가?
 - 5.9는 6.5보다 크지 않습니다. → No 브랜치로 이동합니다.
2. 내부 노드: width > 6.0인가?
 - 5.8은 6.0보다 크지 않습니다. → No 브랜치로 이동합니다.
3. 리프 노드: 최종 결정 노드에 도달했습니다.
 - 이 노드의 레이블은 ”오렌지”입니다.

결론: 이 과일은 ’오렌지’로 예측됩니다.

4 핵심 원리 2: 트리는 어떻게 ”학습”하는가?

예측은 간단합니다. 하지만 `height > 6.5`나 `width > 6.0` 같은 ”최적의 질문”은 어떻게 찾아내는 것일까요? 이것이 바로 결정 트리의 ”학습” 과정입니다.

4.1 학습 목표: 영역 순수도 (Region Purity)

트리 학습의 목표는 각 분할(Split)을 통해 생성된 자식 노드(영역)가 최대한 순수(Pure)해지도록 하는 것입니다.

- **순수한(Pure) 노드:** 노드 안의 모든 데이터가 동일한 클래스에 속합니다. (예: 100% 오렌지) → 불확실성 낮음
 - **불순한(Impure) 노드:** 여러 클래스가 섞여 있습니다. (예: 오렌지 50%, 레몬 50%) → 불확실성 높음
- 학습 알고리즘은 현재 노드를 분할할 수 있는 모든 가능한 질문(모든 특징, 모든 가능한 분할)을 테스트해보고, 그 결과로 만들어지는 두 자식 노드의 평균 불순도를 가장 많이 낮추는 질문을 선택합니다.

4.2 불순도 측정 기준 (Splitting Criteria)

불순도(Impurity)를 측정하는 방법에는 여러 가지가 있습니다.

4.3 기준 1: 분류 오류 (Classification Error)

가장 직관적인 방법입니다. 해당 노드에서 가장 많은 클래스를 정답으로 택했을 때, 틀리는 데이터의 비율입니다.

$$\text{분류 오류} = 1 - \max_k(\Psi(k|R))$$

여기서 $\Psi(k|R)$ 은 영역 R 에서 k 번째 클래스가 차지하는 비율입니다.

□ 예제:

영역 R_2 에 파란 원 5개, 주황 삼각형 8개가 있습니다. (총 13개)

- 파란 원의 비율: $\Psi(\text{파랑}|R_2) = 5/13$
- 주황 삼각형의 비율: $\Psi(\text{주황}|R_2) = 8/13$

이 영역의 다수 클래스는 ’주황 삼각형’입니다. 이 클래스로 예측하면 5개가 틀립니다.

$$\text{분류 오류} = 1 - \max(5/13, 8/13) = 1 - 8/13 = 5/13 \approx 0.38$$

주의사항

분할 평가는 반드시 ’가중 평균’으로 해야 합니다.

어떤 분할이 영역 R_1 (샘플 1개, 오류 0)과 R_2 (샘플 17개, 오류 0.18)를 만들었다고 가정합시다.

단순히 R_1 의 오류가 0이라고 해서 이 분할이 좋은 것은 아닙니다. 샘플이 1개뿐인 영역은 의미가 없습니다.

따라서, 분할의 좋고 나쁨은 생성된 자식 노드들의 불순도를 샘플 수로 가중 평균하여 평가해야 합니다.

$$\text{가중 평균 오류} = \left(\frac{N_1}{N} \right) \times \text{Error}(R_1) + \left(\frac{N_2}{N} \right) \times \text{Error}(R_2)$$

(여기서 N 은 총 샘플 수, N_1, N_2 는 각 자식 노드의 샘플 수)

4.4 기준 2: 지니 불순도 (Gini Impurity)

결정 트리에서 가장 널리 사용되는 기준 중 하나입니다.

$$\text{Gini} = 1 - \sum_k \Psi(k|R)^2$$

이 식은 해당 영역에서 랜덤하게 2개의 샘플을 뽑았을 때, 두 샘플이 서로 다른 클래스일 확률을 의미합니다.

- 완전 순수 (예: 100% 주황): $1 - (1.0^2 + 0^2) = 0$
- 완전 불순 (예: 50% 파랑, 50% 주황): $1 - (0.5^2 + 0.5^2) = 1 - (0.25 + 0.25) = 0.5$

□ 예제:

위와 동일하게 파란 원 5개, 주황 삼각형 8개 (총 13개)가 있는 영역:

$$\text{Gini} = 1 - [(5/13)^2 + (8/13)^2] = 1 - [25/169 + 64/169] = 1 - 89/169 \approx 0.47$$

4.5 기준 3: 엔트로피 (Entropy)

정보 이론(Information Theory)에서 유래한 개념으로, 데이터의 불확실성(무질서도)을 측정합니다.

$$\text{Entropy} = - \sum_k \Psi(k|R) \log_2(\Psi(k|R))$$

- 완전 순수 (예: 100% 주황): $-[1.0 \log_2(1.0) + 0 \log_2(0)] = 0$
- 완전 불순 (예: 50% 파랑, 50% 주황): $-[0.5 \log_2(0.5) + 0.5 \log_2(0.5)] = 1$

엔트로피를 기준으로 분할을 평가할 때는 정보 획득(Information Gain)이라는 개념을 사용하며, 이는 “분할 전 엔트로피 - 분할 후 가중 평균 엔트로피”입니다. 즉, 엔트로피를 가장 많이 줄이는 분할을 선택합니다.

4.6 불순도 지표 비교

Q: Gini와 Entropy는 왜 '분류 오류'보다 더 선호되나요? **A:** 불순도 변화에 더 민감하기 때문입니다.

'분류 오류'는 약간의 불순도 변화에 둔감합니다. 예를 들어, (50%:50%)인 노드를 (60%:40%)으로 개선해도 분류 오류는 여전히 0.4로, (70%:30%)로 개선해야 비로소 0.3으로 떨어집니다.

반면, **Gini**와 **Entropy**는 곡선 형태(Convex)이기 때문에, (50%:50%)에서 약간만 벗어나도 불순도 값이 민감하게 감소합니다.

결론: Gini와 Entropy는 노드를 "더 순수하게" 만들려는 동기가 '분류 오류'보다 강하기 때문에, 더 좋은 트리를 만드는 경향이 있습니다. 특히 **Entropy**가 불순도에 가장 민감하게(가장 가파르게) 반응합니다. (실무에서는 계산이 조금 더 빠른 Gini가 자주 쓰입니다.)

4.7 학습 알고리즘: 탐욕적 알고리즘 (Greedy Algorithm)

모든 가능한 트리 구조를 탐색하여 "전역 최적(Global Optimum)" 트리를 찾는 것은 NP-complete 문제로, 사실상 계산이 불가능합니다.

대신, 결정 트리는 탐욕적 알고리즘(Greedy Algorithm)을 사용합니다. "미래를 보지 않고, 지금 당장 최선의 선택을 한다"는 의미입니다.

1. **시작:** 모든 데이터가 루트 노드 R 에 있습니다.
2. **최적 분할 탐색:**
 - 모든 특징 p (예: height, width)에 대해,
 - 모든 가능한 분할 값 t (예: 6.0, 6.1, 6.5...)를 시도해봅니다.
 - 이 분할(p, t)이 만드는 두 자식 노드(R_1, R_2)의 가중 평균 불순도를 계산합니다.
3. **분할 수행:** 계산된 가중 평균 불순도를 가장 많이 감소시키는 최적의 특징(p^*)과 분할 값(t^*)을 선택하여 노드를 분할합니다.
4. **재귀 (Recurse):**
 - 생성된 자식 노드 R_1 과 R_2 에 대해 2 3단계를 재귀적으로 반복합니다.
5. **중지:** 특정 중지 조건이 만족되면 분할을 멈추고 해당 노드를 리프 노드로 선언합니다.

5 핵심 원리 3: 과적합 방지 (Overfitting)

5.1 문제점: 멈추지 않는 트리

만약 트리가 멈추지 않고 계속 분할하도록 내버려 두면 어떻게 될까요? 트리는 훈련 데이터의 모든 노이즈 (noise)까지 암기하기 시작합니다.

결국, 각 리프 노드에 단 하나의 훈련 데이터 포인트만 남을 때까지 트리가 성장합니다. 이 트리는 훈련 데이터에 대해서는 100%의 정확도를 보이지만, 실제 데이터(Test Data)에서는 노이즈까지 반영했기 때문에 성능이 매우 나빠집니다. 이를 과적합(Overfitting)이라고 부릅니다.

5.2 해결책: 중지 조건 (Stopping Conditions)

과적합을 방지하기 위해, 우리는 트리의 성장을 의도적으로 제한해야 합니다. 이를 ”사전 가지치기 (Pre-pruning)”라고도 하며, 다양한 중지 조건 (하이퍼파라미터)을 사용합니다.

주요 중지 조건 (Hyperparameters)

- **max_depth (최대 깊이)**: 트리가 몇 단계까지 내려갈 수 있는지 제한합니다. (예: ‘ $\text{max_depth} = 3$ ’)
- **min_samples_leaf (리프 노드 최소 샘플 수)**: 리프 노드가 되기 위해 필요한 최소한의 샘플 수를 지정합니다. (예: ‘ $\text{min_samples_leaf} = 4$ ’)
- **max_leaf_nodes (최대 리프 노드 수)**: 트리 전체의 리프 노드 개수를 제한합니다. (예: ‘ $\text{max_leaf_nodes} = 5$ ’)
- **min_impurity_decrease (최소 불순도 감소량)**: 분할을 통해 얻는 불순도 감소(정보 획득)가 이 임계값보다 커야만 분할을 수행합니다.
- **순수 노드**: 분할하려는 노드가 이미 100% 순수하다면(Gini=0) 더 이상 분할할 이유가 없으므로 중지합니다.

5.3 트리 성장 전략: Level-Order vs. Best-First

트리가 성장하는 방식에는 두 가지가 있습니다.

1. Level-Order (수준별 성장):

- 우리가 흔히 생각하는 방식입니다. 너비 우선 탐색(BFS)과 유사합니다.
- Depth 1의 모든 노드를 분할하고, 그다음 Depth 2의 모든 노드를 분할하는 식으로 트리를 층(level) 별로 완성해 나갑니다.
- **max_depth**와 같은 대부분의 중지 조건에서 기본으로 사용됩니다.

2. Best-First (최선 우선 성장):

- **max_leaf_nodes** 중지 조건이 설정될 때 주로 사용됩니다.
- 이 방식은 ”어떤 노드를 분할하는 것이 지금 당장 불순도를 가장 많이 줄일까?”를 트리의 모든 리프 노드에 대해 비교합니다.
- 즉, Depth 2에 있는 Depth 4에 있는 상관없이, 불순도 감소량이 가장 큰 노드를 ”골라서” 분할합니다.
- 단점: 모든 가능한 다음 분할을 비교해야 하므로 계산 비용이 매우 비쌉니다.

5.4 편향-분산 트레이드오프 (Bias-Variance Trade-off)

트리의 복잡도(예: `max_depth`)는 모델의 편향(Bias)과 분산(Variance)에 직접적인 영향을 줍니다.

Table 2: 트리 깊이에 따른 편향-분산 트레이드오프

특징	얕은 트리 (Shallow Tree) (예: <code>max_depth = 4</code>)	깊은 트리 (Deep Tree) (예: <code>max_depth = 100</code>)
모델 복잡도	낮음 (단순함)	높음 (복잡함)
편향 (Bias)	높음 (High Bias)	낮음 (Low Bias)
(설명)	데이터를 단순하게만 봐서 진짜 패턴을 놓침.	데이터의 미세한 패턴(노이즈 포함)
분산 (Variance)	낮음 (Low Variance)	높음 (High Variance)
(설명)	훈련 데이터가 조금 바뀌어도 트리가 거의 변하지 않음.	훈련 데이터가 조금만 바뀌어도 트리가 많이�
결과	과소적합 (Underfitting)	과적합 (Overfitting)

5.5 최적의 하이퍼파라미터 찾기

그렇다면 `max_depth`는 4, 6, 100 중 무엇으로 정해야 할까요? 정답은 데이터마다 다르다며, 이 최적의 값을 찾는 방법이 바로 교차 검증(Cross-Validation)입니다.

우리는 `max_depth`를 2, 3, 4, ..., 10 등으로 바꿔가며 교차 검증을 수행하고, 검증 세트(Validation Set)에서 가장 성능이 좋았던 `max_depth` 값을 최종 모델의 하이퍼파라미터로 선택합니다.

6 학습 체크리스트

결정 트리 학습 점검표

- 동기 (Motivation)

- 로지스틱 회귀가 어떤 종류의 데이터(경계)에서 실패하는지 설명할 수 있는가?

- 결정 트리가 로지스틱 회귀 대비 가지는 장점(복잡성, 해석력)을 아는가?

- 작동 원리 (Prediction)

- 루트, 내부, 리프 노드의 차이점을 아는가?

- 새로운 데이터가 주어졌을 때 '트리 순회'를 통해 예측하는 과정을 시연할 수 있는가?

- 트리의 분할이 2D 특징 공간에서 어떻게 '사각형 영역'으로 표현되는지 이해하는가?

- 학습 원리 (Training)

- 트리 학습의 목표가 '영역 순수도'를 높이는 것임을 아는가?

- '순수도'와 '불순도'가 무엇을 의미하는지 설명할 수 있는가?

- 3가지 불순도 지표(분류 오류, Gini, Entropy)의 공식을 아는가?

- Gini와 Entropy가 분류 오류보다 선호되는 이유(민감도)를 설명할 수 있는가?

- 분할 평가 시 '가중 평균' 불순도를 사용해야 하는 이유를 아는가?

- 결정 트리 학습이 왜 '탐욕적(Greedy)' 알고리즘인지 설명할 수 있는가?

- 과적합 (Overfitting)

- 트리를 멈추지 않으면 왜 과적합이 발생하는지(노이즈 암기) 설명할 수 있는가?

- 주요 중지 조건(예: max_depth, min_samples_leaf)의 역할을 아는가?

- 'Level-order'와 'Best-first' 성장 전략의 차이점과 계산 비용을 아는가?

- 트리 깊이(복잡도)와 편향-분산 트레이드오프의 관계를 설명할 수 있는가?

- 최적의 중지 조건(하이퍼파라미터)을 어떻게 찾는지 아는가? (정답: 교차 검증)

7 FAQ (주요 질문 및 답변)

Q: Gini 불순도와 Entropy 중 무엇을 사용해야 하나요? A: 대부분의 경우 결과는 비슷합니다. Gini 불순도는 \log_2 계산이 없는 단순 계급 연산이라 계산 속도가 조금 더 빠릅니다. (이것이 Scikit-learn의 기본 값인 이유입니다.) Entropy는 불순도에 이론적으로 더 민감하지만, 실제 성능 차이는 크지 않은 경우가 많습니다.

Q: 왜 '탐욕적(Greedy)' 알고리즘을 사용하나요? 모든 경우를 다 따져보면 더 좋지 않나요? A: 모든 경우의 수(모든 가능한 트리 구조)를 따져보는 것은 "전역 최적해"를 찾는 것을 의미하지만, 이 문제는 조합 폭발(combinatorial explosion)로 인해 계산적으로 불가능(NP-complete) 합니다. 탐욕적 접근법은 비록 전역 최적해를 보장하지는 않지만, 매우 빠르고 현실적인 시간 안에 "충분히 좋은" 트리를 찾아냅니다.

Q: `max_depth`와 `min_samples_leaf` 중 과적합 방지에 무엇이 더 효과적인가요? A: 둘 다 트리의 복잡도를 제어하는 중요한 하이퍼파라미터입니다.

- `max_depth`: 트리의 전체적인 '크기'와 '구조'를 직접적으로 제한합니다. 매우 강력한 규제 수단입니다.

- `min_samples_leaf`: 데이터의 '밀도'에 기반하여 분할을 제어합니다. 노이즈(이상치)가 리프 노드가 되는 것을 방지하여 모델을 더 안정적으로(robust) 만듭니다.

어느 것이 더 낫다고 말할 수 없으며, 두 하이퍼파라미터 모두 교차 검증(Cross-Validation)을 통해 데이터에 맞게 튜닝되어야 합니다.

8 1페이지 요약: 결정 트리 핵심

결정 트리(Decision Tree) 한눈에 보기

1. 모델 정의: 스무고개(직관적)

- 아이디어: ”예/아니오” 질문(예: $\text{height} > 6.5?$)을 반복하여 데이터를 분류하는 나무 구조의 모델.
- 장점:
 1. 해석 가능성 (White Box): 모델의 결정 과정을 사람이 쉽게 이해할 수 있음.
 2. 비선형 분류: 로지스틱 회귀가 못하는 복잡한 결정 경계(원, 사각형 등)를 만들 수 있음.
- 예측: ’트리 순회(Traversing)’ → 새로운 데이터가 루트부터 리프까지 질문을 따라 내려가며 최종 클래스 레이블을 할당받음.

2. 학습 원리: 순수도 찾기(탐욕적)

- 목표: 각 노드(영역)가 최대한 ’순수하게’(한 클래스로만 구성) 되도록 분할.
- 학습 방식: 탐욕적(Greedy) 알고리즘 → 지금 당장 불순도를 가장 많이 줄이는 최적의 (특징, 분할 값)을 찾아 분할하고 재귀 반복.
- 불순도 측정 (Splitting Criteria):
 - Gini Index (기본 값): $1 - \sum p_k^2$ (계산이 빠름)
 - Entropy: $-\sum p_k \log_2(p_k)$ (불순도에 가장 민감함)

3. 주요 과제: 과적합 방지(규제)

- 문제: 중지 조건이 없으면 트리가 무한정 성장하여 훈련 데이터의 ’노이즈’까지 모두 암기함 (과적합: High Variance, Low Bias).
- 해결 (Stopping Conditions): 트리의 성장을 의도적으로 제한(사전 가지치기).
 - max_depth: 트리의 최대 깊이 제한.
 - min_samples_leaf: 리프 노드가 가져야 할 최소 샘플 수.
- 튜닝: 최적의 하이퍼파라미터(예: 최적의 ‘ max_depth ’) 교차 검증(Cross-Validation) .

December 10, 2025

- 강의명: CS109A: 데이터 과학 입문
- 주차: Lecture 19
- 교수명: Pavlos Protopapas, Kevin Rader, Chris Gumb
- 목적: Lecture 19의 핵심 개념 학습

Contents

1	핵심 용어 정리	2
2	회귀 트리 (Regression Trees)	3
2.1	핵심 질문: 어떻게 분할 기준을 정할까? (Splitting Criteria)	3
2.2	회귀 트리의 성장 및 예측 과정	4
3	범주형 변수 처리 (Categorical Attributes)	5
3.1	잘못된 접근: 순서형 인코딩 (Ordinal Encoding)	5
3.2	올바른 접근: 원-핫 인코딩 (One-Hot Encoding, OHE)	5
4	가지치기 (Pruning): 과적합과의 전쟁	6
4.1	과적합 제어: 사전 중단 vs 사후 가지치기	6
4.2	비용 복잡도 가지치기 (Cost Complexity Pruning, CCP)	6
4.3	최적의 α 와 트리 찾는 과정 (Nested Cross-Validation)	7
5	FAQ 및 자가 점검	8

개요: 결정 트리 학습의 두 가지 핵심

이 문서는 결정 트리(Decision Tree) 모델의 두 가지 핵심적인 확장 개념인 **회귀(Regression)**와 **가지치기(Pruning)**에 대해 다룹니다.

데이터 사이언스 초심자도 쉽게 이해할 수 있도록 각 개념의 필요성부터 작동 원리, 그리고 실제 적용 시 주의사항까지 단계별로 설명합니다.

1. 회귀 트리 (Regression Trees): '예/아니오' 같은 분류(Classification) 문제뿐만 아니라, '집값', '매출액' 등 연속적인 숫자(Quantitative outcome)를 예측하는 회귀 문제에 결정 트리를 사용하는 방법입니다. 핵심은 **'불순도(Impurity)'** 대신 'MSE(평균 제곱 오차)'를 기준으로 트리를 분할하는 것입니다.

2. 가지치기 (Pruning): 트리가 훈련 데이터에만 과도하게 최적화되는 **과적합(Overfitting)**을 방지하기 위한 핵심 기술입니다. 트리를 일단 최대로 성장시킨 후, 불필요한 가지들을 체계적으로 잘라내어 모델의 일반화 성능을 높입니다. 핵심은 **'비용 복잡도(Cost Complexity)'**라는 정규화 항을 도입하는 것입니다.

1 핵심 용어 정리

본격적인 학습에 앞서, 자주 등장하는 핵심 용어들을 정리합니다.

용어	원어	쉬운 설명	비고
회귀 트리	Regression Tree	연속적인 숫자(예: 가격)를 예측하는 결정 트리	분류 트리의 반대
분류 트리	Classification Tree	범주형 값(예: 생존/사망)을 예측하는 결정 트리	
분할 기준	Splitting Criterion	트리의 가지를 나눌 때 사용하는 기준	분류: 지니 불순도, 엔트로피 회귀: MSE (평균 제곱 오차)
단말 노드	Leaf Node (Terminal Node)	트리의 가장 마지막에 위치한 노드 (예측값 결정)	
과적합	Overfitting	모델이 훈련 데이터에만 너무 잘 맞아, 새로운 데이터에 대한 예측 성능이 떨어지는 현상	
가지치기	Pruning	과적합을 막기 위해 트리의 복잡도를 줄이는 과정	(사후 가지치기)
비용 복잡도	Cost Complexity	모델의 오류(Error)와 복잡도(Complexity)를	
	Pruning (CCP)	동시에 고려하는 가지치기 공식	
복잡도 매개변수	Complexity Parameter (α)	트리의 복잡도에 얼마나 큰 폐널티를 줄지 정하는 값	하이퍼파라미터

Table 1: 회귀 트리 및 가지치기 관련 핵심 용어

2 회귀 트리 (Regression Trees)

우리는 결정 트리를 '스무고개'와 비슷하다고 배웠습니다. "A인가?", "B인가?" 질문을 통해 대상을 구분하는 것은 **분류(Classification)** 문제입니다.

그렇다면 "이 사람의 나이는 몇 살인가?" 또는 "이 집의 가격은 얼마인가?"처럼 연속적인 숫자(Continuous Variable)를 예측해야 하는 **회귀(Regression)** 문제에는 결정 트리를 어떻게 적용할 수 있을까요?

회귀 트리 (Regression Tree)란? 회귀 트리는 예측 결과값이 범주(Category)가 아닌 연속적인 숫자인 결정 트리 모델입니다.

- **분류 트리:** 각 단말 노드(Leaf Node)는 다수결(Majority Vote)을 통해 가장 흔한 클래스(예: '생존')를 예측합니다.
- **회귀 트리:** 각 단말 노드(Leaf Node)는 해당 노드에 속한 훈련 데이터 샘플들의 평균(Mean) 값을 예측합니다.

타이타닉 생존 예측 (분류 트리)

- 질문: "성별이 여성인가?" → 예
- 질문: "객실 등급이 1등급인가?" → 예
- 예측: 해당 노드에 '생존' 80명, '사망' 5명이 있다면, 예측값은 '생존' (다수결)

보스턴 집값 예측 (회귀 트리)

- 질문: "방 개수가 6개 이상인가?" → 예
- 질문: "범죄율이 1% 미만인가?" → 예
- 예측: 해당 노드에 속한 집들의 가격이 {5억, 6억, 5.5억} 이라면, 예측값은 5.5억 (평균)

2.1 핵심 질문: 어떻게 분할 기준을 정할까? (Splitting Criteria)

분류 트리에서는 '지니 불순도(Gini)'나 '엔트로피(Entropy)'를 사용하여, 분할 후 두 그룹이 얼마나 더 '순수해졌는지(Pure)'를 측정했습니다. "생존/사망이 명확히 갈리는가?"가 중요했습니다.

회귀 트리에서도 비슷한 목표를 가집니다. "분할된 두 그룹의 값들이 얼마나 서로 비슷비슷하게 모여 있는가?"

- **나쁜 분할:** 한 그룹에 {1억, 5억, 10억}이 섞여있음 → 평균 5.3억은 누구도 대표하지 못함 (분산이 큼)
- **좋은 분할:** 한 그룹에 {1억, 1.1억, 1.2억}이 모여있음 → 평균 1.1억은 그룹을 잘 대표함 (분산이 작음)
여기서 "값이 얼마나 흩어져 있는가(분산)"를 측정하는 가장 좋은 지표가 바로 MSE(Mean Squared Error, 평균 제곱 오차)입니다.

회귀 트리의 분할 기준: MSE 최소화 회귀 트리는 분할 후 생성될 두 자식 노드(R_1, R_2)의 MSE의 가중 평균을 최소화하는 지점(변수 p 와 임계값 t_p)을 찾습니다.

각 노드 R 의 MSE는 해당 노드의 분산(Variance)과 같습니다.

$$MSE(R) = \frac{1}{n_R} \sum_{i \in R} (y_i - \bar{y}_R)^2$$

(n_R : 노드 R 의 샘플 수, y_i : 실제 값, \bar{y}_R : 노드 R 의 평균 예측 값)

따라서, 트리가 찾는 최적의 분할 (p, t_p)는 다음 공식을 최소화하는 것입니다.

$$\min_{p, t_p} \left[\frac{N_1}{N} MSE(R_1) + \frac{N_2}{N} MSE(R_2) \right]$$

(N : 부모 노드의 총 샘플 수, N_1, N_2 : 각 자식 노드의 샘플 수)

2.2 회귀 트리의 성장 및 예측 과정

1. **분할 (Greedy Algorithm):** 모든 변수(Predictors)와 모든 가능한 분할 지점(Unique values)을 하나씩 다 시도해봅니다. 그 중 MSE 가중 평균을 가장 많이 낮춰주는 '최적의 질문' 하나를 선택하여 노드를 분할합니다.
2. **성장:** 위 1번 과정을 재귀적으로 반복하며 트리를 키워나갑니다.
3. **중단 (Stopping Conditions):** 미리 정해둔 중단 조건에 도달하면 성장을 멈춥니다.
 - 'max_{depth}' :
 - 'min_{samplesleaf}' :
 - **Accuracy Gain (MSE Reduction):** 분할로 인해 MSE가 줄어드는 양('Gain(R)')이 정해진 임계값(Threshold)보다 작으면 더 이상 분할하지 않습니다.

$$Gain(R) = MSE(R_{\text{parent}}) - \left(\frac{N_1}{N} MSE(R_1) + \frac{N_2}{N} MSE(R_2) \right)$$

4. **예측 (Prediction):** 새로운 데이터(x_i)가 들어오면, 트리의 질문을 따라가며 특정 단말 노드에 도달합니다. 해당 단말 노드에 저장된 훈련 데이터의 평균값(\bar{y}_{leaf})을 최종 예측값(\hat{y}_i)으로 반환합니다.

3 범주형 변수 처리 (Categorical Attributes)

결정 트리는 ”변수 < 임계값” 형태의 비교를 기반으로 작동합니다.

- 수치형(Numerical): ”Sepal width > 4.0?” (가능)
 - 범주형(Categorical): ”Color < Red?” (불가능)
- ’Red’보다 작다는 것은 수학적으로 의미가 없습니다.

3.1 잘못된 접근: 순서형 인코딩 (Ordinal Encoding)

가장 간단하게 숫자를 부여하는 방식입니다. (예: Yellow=0, Red=1, Purple=2)

순서형 인코딩은 절대 사용하면 안 됩니다 (데이터가 실제 순서를 갖지 않는 한).

이 방식은 모델에게 인공적인 순서(Artificial Order)를 강제로 학습시킵니다. (예: ”Purple(2) > Red(1)“)

만약 인코딩 순서를 ”Yellow=2, Red=0, Purple=1”로 바꾸면, 트리의 분할 결과 자체가 완전히 달라집니다. 이는 모델의 성능이 데이터의 본질이 아닌, 프로그래머의 임의적인 인코딩 순서에 의존하게 됨을 의미합니다.

3.2 올바른 접근: 원-핫 인코딩 (One-Hot Encoding, OHE)

범주의 각 값(Category)을 자신만의 새로운 이진(Binary) 변수로 만드는 것입니다.

원-핫 인코딩 (OHE) ’Color’라는 하나의 변수를 ’is_Yellow’, ’is_Red’, ’is_Purple’이라는 여러 개의 0 또는 1 값을 갖는 변수로 변환합니다.

	Sepal width	Color
변환 전 (Original)	3.0 mm	Yellow
	3.5 mm	Red
	3.7 mm	Purple

	Sepal width	Color_Yellow	Color_Red	Color_Purple
변환 후 (One-Hot Encoded)	3.0 mm	1	0	0
	3.5 mm	0	1	0
	3.7 mm	0	0	1

Table 2: 원-핫 인코딩 예시

이렇게 변환하면, 트리는 ”Color_Red >= 1?” (즉, 색상이 빨간색인가?) 과 같은 논리적인 질문을 수치형으로 수행할 수 있게 됩니다.

Scikit-Learn 사용 시 주의사항

‘sklearn.tree.DecisionTreeClassifier’ 나 ‘DecisionTreeRegressor’는 범주형 변수를 자동으로 처리해주지 않습니다.

모델에 데이터를 넣기 전에, 사용자가 직접 Pandas의 ‘get_dummies’ ‘sklearn.preprocessing.OneHotEncoder’ ‘XGBoost, LightGBM, CatBoost’ .)

4 가지치기 (Pruning): 과적합과의 전쟁

결정 트리를 아무런 제한 없이 끝까지 성장시키면(Full Tree), 훈련 데이터의 모든 샘플을 완벽하게 구분(분류)하거나 완벽하게 예측(회귀, $R^2=1$) 하려 합니다.

이는 훈련 데이터의 사소한 노이즈(Noise)까지 모두 학습하는 과적합(Overfitting) 상태로 이어집니다. 이런 모델은 새로운(Unseen) 데이터가 들어왔을 때 형편없는 성능을 보입니다.

과적합의 비유 ($R^2=1$ 임)

”오늘 $R^2=1$ 인 모델을 만들었어요!” ”당장 내 집에서 나가!”

$R^2=1$ 은 모델이 훈련 데이터를 100% 완벽하게 설명한다는 뜻입니다. 이는 현실에서는 불가능하며, 훈련 데이터에만 과적합된 ‘암기 기계’를 만들었다는 의미입니다. 이런 모델은 실제 문제 해결에 아무런 도움이 되지 않기 때문에 데이터 과학자들이 가장 경계하는 상황입니다.

4.1 과적합 제어: 사전 중단 vs 사후 가지치기

1. 사전 중단 (Pre-stopping): 트리가 성장하는 중에 멈추는 방식입니다. (max_{depth} , $\text{min}_{samples,leaf}$)
2. 문제점: 최적의 중단 시점을 미리 알기 어렵습니다. 너무 일찍 멈추면 과소적합(Underfitting)이, 너무 늦게 멈추면 과적합(Overfitting)이 발생합니다.

사후 가지치기 (Post-pruning / Pruning): ”일단 끝까지 키우고, 나중에 잘라낸다.”

- 장점: 트리의 전체 구조를 본 후에 불필요한 부분을 제거하므로, 더 유연하고 정교한 모델 복잡도 제어가 가능합니다.
- 방법: 가장 표준적인 방법이 비용 복잡도 가지치기(Cost Complexity Pruning, CCP)입니다.

4.2 비용 복잡도 가지치기 (Cost Complexity Pruning, CCP)

CCP는 모델의 오류(Error)와 복잡도(Complexity) 사이에 균형점을 찾는 정규화(Regularization) 기법입니다.

비용 복잡도 (Cost Complexity) 공식 CCP는 트리의 ‘비용’ $C(T)$ 를 최소화하는 것을 목표로 합니다.

$$C(T) = \text{Error}(T) + \alpha \cdot |T|$$

- $C(T)$: 트리 T 의 총 비용 복잡도
- $\text{Error}(T)$: 트리 T 가 훈련 데이터에서 발생시키는 총 오류 (예: 분류 오류 수, 또는 총 MSE)
- $|T|$: 트리 T 의 단말 노드(Leaf) 개수. (트리의 복잡도를 나타내는 척도)
- α (알파): 복잡도 매개변수 (Complexity Parameter). 사용자가 정하는 하이퍼파라미터입니다. α 는 ‘복잡도에 대한 폐널티’입니다.
- $\alpha = 0$: 복잡도 폐널티 없음. $C(T) = \text{Error}(T)$ 가 되므로, 훈련 오류가 가장 낮은 가장 큰 트리(Full Tree)가 선택됩니다.
- $\alpha \rightarrow \infty$: 복잡도 폐널티가 매우 큼. 단말 노드가 2개인 것보다 1개인 것을 선호하게 되므로, 결국 가장 단순한 트리(Root Node만 있는 트리)가 선택됩니다.
- $0 < \alpha < \infty$: α 값이 커질수록, 오류(Error)가 조금 늘어나더라도 더 단순한(단말 노드가 적은) 트리를 선호하게 됩니다.

$\alpha = 0.2$ 일 때, 두 트리를 비교해봅시다.

트리 T (Full Tree):

- $\text{Error}(T) = 0.32$

- $|T|$ (단말 노드 수) = 8
- $C(T) = 0.32 + (0.2 \times 8) = 1.92$

트리 T_{small} (가지치기 후):

- $\text{Error}(T_{small}) = 0.33$ (훈련 오류는 약간 증가)
- $|T_{small}|$ (단말 노드 수) = 7
- $C(T_{small}) = 0.33 + (0.2 \times 7) = 1.73$

결론: 비록 T_{small} 이 훈련 오류는 0.01 더 높지만, 전체 비용 복잡도 $C(T)$ 는 더 낮습니다 ($1.73 < 1.92$). 따라서 $\alpha = 0.2$ 일 때는 T_{small} 이 T 보다 '더 좋은 트리'로 간주됩니다.

4.3 최적의 α 와 트리 찾는 과정 (Nested Cross-Validation)

CCP의 알고리즘은 복잡해 보이지만, 본질은 2단계 교차 검증입니다.

1. [Inner Loop] 특정 α 에 대한 최적의 트리 찾기

먼저 α 값을 하나 고정합니다 (예: $\alpha = 0.1$). 알고리즘은 Full Tree(T_0)에서 시작하여, '가장 약한 고리 (Weakest Link)' (즉, 잘라냈을 때 $C(T)$ 가 가장 적게 증가하거나 오히려 감소하는 노드)부터 차례대로 제거합니다.

이 과정을 통해 $\alpha = 0.1$ 일 때 가능한 후보 트리들의 목록을 만듭니다: $\{T_0(\text{Full}), T_1, T_2, \dots, T_L(\text{Root})\}$

이 후보 트리들을 검증 데이터(Validation Set)에 적용하여, $\alpha = 0.1$ 일 때 검증 성능(예: Validation MSE)이 가장 좋은 트리 $T_{best_for_0.1}$ 를 하나 선택합니다.

2. [Outer Loop] 모든 α 중 최적의 α 찾기

이제 다른 α 값 (예: $\alpha = 0.2, \alpha = 0.5, \dots$)에 대해 1번 과정을 반복합니다.

- $\alpha = 0.1$ 일 때 최적 트리: $T_{best_for_0.1}$ (검증 점수: 85점)
- $\alpha = 0.2$ 일 때 최적 트리: $T_{best_for_0.2}$ (검증 점수: 90점)
- $\alpha = 0.5$ 일 때 최적 트리: $T_{best_for_0.5}$ (검증 점수: 88점)

3. 최종 선택

모든 α 후보들 중에서 가장 높은 검증 점수(90점)를 기록한 $\alpha = 0.2$ 와, 그 때의 트리 $T_{best_for_0.2}$ 를 최종 모델로 선택합니다.

5 FAQ 및 자가 점검

강의 중 나온 퀴즈와 자주 묻는 질문들을 통해 이해도를 점검합니다.

Q1: 결정 트리 모델은 '탐욕적 알고리즘(Greedy Algorithm)'을 사용하는데, 왜 이것이 정당화되나요?

A: '탐욕적'이라는 것은 매 분할 시 점마다 당장 MSE(또는 Gini)를 가장 많이 줄여주는 최적의 분할을 찾는다는 의미입니다. 이렇게 찾은 분할이 나중에 전체 트리의 최적(Global Optimum)을 보장하지는 않습니다. 하지만, 모든 가능한 트리 조합을 탐색하는 것은 계산적으로 거의 불가능(NP-hard)합니다. 탐욕적 접근 방식은 계산 비용이 훨씬 저렴하면서도 현실적으로 매우 준수한 성능의 모델을 찾아주기 때문에 널리 사용됩니다.

Q2: 하나의 결정 트리 안에서 동일한 변수(Feature)가 여러 번 사용될 수 있나요?

A: 예, 가능합니다. 예를 들어, 상위 노드에서 "Work experience > 2"로 분할한 후, 그 자식 노드 중 하나에서 "Work experience > 4"로 더 세분화된 분할을 할 수 있습니다. 이는 모델이 데이터의 복잡한 관계를 더 정교하게 학습할 수 있게 해줍니다.

Q3: 훈련된 회귀 트리(Regression Tree)가 새로운 데이터 포인트를 예측할 때, 마지막 단계는 무엇인가요?

A: 새로운 데이터가 트리를 따라 내려가 도달한 단말 노드(Leaf Node)에 저장된 값을 예측 값으로 반환합니다. 이 저장된 값은 바로 해당 단말 노드에 속해있던 훈련(Training) 데이터들의 평균(Average) 값입니다.

Q4: 어떤 회귀 트리가 4개의 단말 노드(Region)를 생성했습니다. 이 트리의 전체 MSE는 어떻게 계산하나요?

- R1: 샘플 90개, MSE = 0.2
- R2: 샘플 5개, MSE = 1.2
- R3: 샘플 3개, MSE = 1.5
- R4: 샘플 2개, MSE = 1.8

A: 트리의 전체 MSE는 각 노드 MSE의 가중 평균(Weighted Average)입니다.

- 총 샘플 수(N) = $90 + 5 + 3 + 2 = 100$
- 전체 MSE = $(\frac{90}{100} \times 0.2) + (\frac{5}{100} \times 1.2) + (\frac{3}{100} \times 1.5) + (\frac{2}{100} \times 1.8)$
- 전체 MSE = $(0.18) + (0.06) + (0.045) + (0.036) = 0.321$

대부분의 샘플(90%)이 R1에 속해 있으므로, 전체 MSE는 R1의 MSE(0.2)에 가장 가깝게 나옵니다.

Q5: ' $\text{max}_{depth} = 1$ ' (DecisionBoundary) (Linear), (Non-linear) ?

A: 선형(Linear)입니다. ' $\text{max}_{depth} = 1$ ' 단 하나의 분할 .(: "x" > 6.5") () , . 2 .

December 10, 2025

- 강의명: CS109A: 데이터 과학 입문
- 주차: Lecture 20
- 교수명: Pavlos Protopapas, Kevin Rader, Chris Gumb
- 목적: Lecture 20의 핵심 개념 학습

Contents

1	용어 정리	3
2	결측치(Missingness)란 무엇인가?	4
2.1	결측치의 정의와 문제점	4
2.2	결측치 처리의 순진한 접근법(과 그 위험성)	4
3	결측의 3가지 유형(MCAR, MAR, MNAR)	5
3.1	1. MCAR (Missing Completely at Random, 완전 임의 결측)	5
3.2	2. MAR (Missing at Random, 임의 결측)	5
3.3	3. MNAR (Missing Not at Random, 비임의 결측)	5
4	결측치 처리(Imputation) 방법론	6
4.1	처리 전 고려사항	6
4.2	방법 1: 결측치 표시자 변수(Missingness Indicator)	6
4.3	방법 2: 모델 기반 대체(Model-based Imputation)	6
4.4	방법 3: 불확실성을 고려한 모델 기반 대체	7
4.5	여러 변수에 결측치가 있는 경우(Iterative Imputation)	7
4.6	Sklearn을 사용한 구현	8
5	블랙박스 모델 해석 및 시각화	9
5.1	시각화의 목표	9
5.2	왜 모델 해석이 필요한가? (Parametric vs. Non-parametric)	9
5.3	부분 의존성 플롯(Partial Dependence Plots, PDP)	9
5.3.1	예제 1: 단일 변수 모델 시각화	9
5.3.2	예제 2: 다중 변수 모델의 PDP(상호작용이 없는 경우)	10
5.3.3	예제 3: PDP를 이용한 상호작용(Interaction) 발견	10

6	효과적인 시각화의 원칙 (부록)	11
6.1	나쁜 시각화의 예	11
6.2	좋은 시각화의 원칙	11
7	빠르게 훑어보기 (1-Page Summary)	12

▣ 핵심 요약

이 문서는 데이터 분석의 두 가지 큰 장애물인 **결측치(Missingness)**와 **블랙박스 모델(Black Box Models)**을 다루는 방법을 설명합니다.

1. **결측치 처리:** 데이터에 구멍(NaN)이 있을 때 발생하는 문제를 이해하고, 단순 삭제나 평균값 대체의 위험성(편향)을 배웁니다. MCAR, MAR, MNAR이라는 세 가지 결측 유형을 구분하고, '결측치 표시자'나 '모델 기반 대체' (특히, 불확실성을 고려한 대체)와 같은 고급 기법들을 살펴봅니다.
2. **모델 시각화:** k-NN이나 의사결정나무처럼 해석이 어려운 '블랙박스' 모델의 작동 방식을 이해하기 위해 **부분 의존성 플롯(Partial Dependence Plots, PDP)**을 사용하는 방법을 배웁니다. PDP를 통해 특정 변수가 모델의 예측에 어떤 영향을 미치는지, 변수 간 상호작용은 없는지 시각적으로 확인할 수 있습니다.

1 용어 정리

주요 용어들을 표로 정리했습니다.

용어	원어	쉬운 설명	비고
결측치	Missingness	데이터셋에 값이 누락된 '구멍'이 있는 상태 또는 그 값.	NaN (Not a Number)로 표시됨.
편향	Bias	모델의 예측이나 추정치가 체계적으로 한쪽으로 치우치는 경향.	예: 저울이 항상 500g 높게 측정하는 것.
대체 (대치)	Imputation	누락된 결측치를 추정된 값으로 채워 넣는 과정.	가장 간단한 예는 '평균값'으로 채우기.
MCAR	Missing Completely at Random	**완전 임의 결측.** 결측이 다른 어떤 변수와도 상관없이 무작위로 발생.	(예: 데이터 입력 중 무작위 오타)
MAR	Missing at Random	**임의 결측.** 결측 여부가 다른 변수와 관련 있음.	(예: 남성이 여성보다 '소득' 문항 응답률이 낮음)
MNAR	Missing Not at Random	**비 임의 결측.** 결측 여부가 누락된 값 자체* 또는 관측되지 않은* 변수와 관련 있음.	(예: 고소득자가 '소득' 문항 응답을 거부함)
블랙박스 모델	Black Box Model	모델의 내부 작동 원리를 이해하기 어려운 복잡한 모델.	예: k-NN, 랜덤 포레스트, 딥러닝
PDP	Partial Dependence Plot	**부분 의존성 플롯.** 다른 변수들을 고정한 채, 특정 변수 하나가 모델 예측에 미치는 영향을 보여주는 그래프.	모델 해석의 핵심 도구.

Table 1: 결측치 및 모델 해석 관련 핵심 용어

2 결측치(Missingness) 란 무엇인가?

2.1 결측치의 정의와 문제점

결측치(Missingness)란 데이터셋에 값이 존재하지 않는 '구멍'이 있는 것을 의미합니다. Pandas에서는 주로 NaN (Not a Number)로 표시됩니다.

데이터에 결측치가 있으면 두 가지 큰 문제가 발생합니다.

1. **모델 학습 불가:** 대부분의 머신러닝 라이브러리(예: sklearn)는 데이터에 NaN 값이 하나라도 포함되어 있으면 오류를 발생시키며 모델 학습을 거부합니다.
2. **심각한 편향(Bias) 발생:** 결측치를 처리하기 위해 순진한 방법을 사용하면, 모델이 현실을 체계적으로 잘못 예측하는 '편향'이 발생할 수 있습니다.

주의사항

편향(Bias)이란?

편향은 모델의 예측이 지속적으로 한쪽으로 치우치는 것입니다.

* **직관적 예시:** 어떤 저울이 항상 실제 무게보다 1kg을 더 높게 보여준다면, 이 저울은 '편향'되어 있습니다. 몇 번을 측정하든 항상 1kg 높은 값이 나옵니다. * **데이터 예시:** 결측치를 단순히 '0'으로 채웠다고 가정해봅시다. 만약 '0'이라는 값이 실제 데이터에서는 매우 드문 값이라면, 모델은 '0'이라는 값을 과도하게 학습하여 현실과 동떨어진 예측을 하게 될 수 있습니다.

2.2 결측치 처리의 순진한 접근법 (과 그 위험성)

결측치를 만났을 때 가장 먼저 떠올리는 간단한 두 가지 방법과 그 위험성은 다음과 같습니다.

1. **관측치(행) 삭제:** 결측치가 하나라도 있는 행(row)을 모두 삭제합니다. * **위험성:** 만약 결측치가 무작위로 발생한 것이 아니라 특정 그룹(예: 특정 연령대)에서만 집중적으로 발생했다면, 해당 그룹의 데이터가 통째로 사라집니다. 이는 모델이 해당 그룹을 아예 학습하지 못하게 만들어 심각한 편향을 유발합니다.
2. **단순 값 대체 (Mean/Median/Mode):** * **수치형 변수:** 전체 데이터의 '평균(mean)'이나 '중앙값(median)'으로 모든 NaN을 채웁니다. * **범주형 변수:** 가장 빈번하게 등장한 '최빈값(mode)'으로 모든 NaN을 채웁니다. * **위험성:** 모든 결측치에 똑같은 값을 넣으면, 해당 값 주변에 데이터가 비정상적으로 몰리게 됩니다. 이는 데이터의 실제 분포를 왜곡하며, 변수 간의 관계를 약화시키거나 왜곡시킵니다.

이러한 순진한 방법들은 데이터의 귀중한 정보를 손실시키고 편향을 유발하므로, 왜 결측치가 발생했는지 먼저 파악하는 것이 중요합니다.

3 결측의 3가지 유형 (MCAR, MAR, MNAR)

결측치를 제대로 처리하기 위해서는 결측치가 ”왜” 발생했는지 그 원인을 파악해야 합니다. 통계학에서는 이 원인을 세 가지 유형으로 분류합니다.

3.1 1. MCAR (Missing Completely at Random, 완전 임의 결측)

***”결측이 완전히 무작위로 발생했다”**는 의미입니다.

이는 결측 여부가 데이터셋의 그 어떤 변수(관측된 변수, 누락된 변수)와도 아무런 관련이 없는 경우입니다.

* **비유:** 설문지 데이터를 엑셀에 입력하다가, 직원이 피곤해서 아무 데나 랜덤하게 몇 개를 빠뜨리고 입력한 상황입니다. * **특징:** 가장 이상적이고 다루기 쉬운 케이스입니다. * **처리:** 데이터가 충분히 많다면, MCAR인 경우는 결측치가 있는 행을 삭제해도 편향이 발생하지 않습니다. (단, 데이터 손실은 감수해야 함)

3.2 2. MAR (Missing at Random, 임의 결측)

***”결측 여부가 관측된 다른 변수와 관련이 있다”**는 의미입니다.

결측이 발생한 변수(X_1) 자체와는 관련이 없지만, 우리가 관측할 수 있는 다른 변수(X_2)와는 관련이 있는 경우입니다.

* **예시:** 직장 내 괴롭힘에 대한 설문조사에서 ’괴롭힘 경험’ 문항에 결측치가 많습니다. 이 결측 여부가 ’괴롭힘 경험’ 자체와는 관련이 없을 수 있지만, ’성별’ 변수와는 관련이 있을 수 있습니다. (예: 남성이 여성보다 해당 문항 응답을 더 꺼림) * **특징:** ’성별’이라는 관측된 변수를 활용하면 결측의 패턴을 설명할 수 있습니다. * **처리:** 모델링을 통해 결측치를 잘 처리할 수 있습니다. (예: 성별을 예측 변수로 사용하여 결측치 대체 모델 생성)

3.3 3. MNAR (Missing Not at Random, 비임의 결측)

***”결측 여부가 누락된 값 자체 또는 관측되지 않은 변수와 관련이 있다”**는 의미입니다.

* **예시 1 (누락된 값 자체):** ’소득’ 설문에서, 고소득자일수록 자신의 소득을 밝히기 꺼려 해서 응답을 하지 않는 경우. 즉, ’소득’이라는 값 자체가 높을수록 결측이 될 확률이 높습니다. * **예시 2 (관측되지 않은 변수):** 임상시험에서 부작용이 심한 환자들이(관측되지 않은 ’부작용’ 변수) 시험을 중도 포기하여 (결과값 결측) 데이터에서 누락되는 경우. * **특징:** 가장 다루기 어렵고 심각한 편향을 유발합니다. 우리가 그 이유를 알 수 없기 때문입니다. * **처리:** 통계적으로 완벽하게 해결하기 매우 어렵습니다.

주의사항

Q: 내 데이터가 어떤 유형인지 어떻게 알 수 있나요?

A: 알 수 없습니다.

안타깝게도 우리는 데이터만 보고 이 결측치가 MCAR, MAR, MNAR 중 무엇인지 통계적으로 완벽하게 증명할 수 없습니다. MNAR은 ”관측되지 않은” 변수에 의해 발생할 수 있기 때문입니다.

따라서 데이터 분석가들은 ***”우리의 데이터가 최소한 MAR이라고 가정하고, 관측된 다른 변수들을 최대한 활용하여 결측치를 모델링하자”**라는 실용적인 접근 방식을 취합니다.

특징	MCAR (완전 임의 결측)	MAR (임의 결측)	MNAR (비임의 결측)
결측 원인	완전 무작위	관측된 다른 변수	누락된 값 자체 또는 숨겨진 변수
예시	데이터 입력 실수	성별에 따라 소득 응답률 다른	고소득자가 소득 응답 거부
해결 난이도	쉬움 (삭제 가능)	중간 (모델링으로 해결 가능)	매우 어려움 (편향 피하기 힘듦)

Table 2: 결측 3유형 비교

4 결측치 처리(Imputation) 방법론

결측치를 단순히 삭제하거나 평균값으로 채우는 대신, 더 정교한 방법들을 사용해야 합니다.

4.1 처리 전 고려사항

1. **결측치가 어디에 있는가? (Y vs X)** * **예측 변수(X)에 결측:** 대부분의 처리 기법이 여기에 초점을 맞춥니다. * **반응 변수(Y)에 결측:** 매우 다루기 어렵습니다. Y 값을 예측하는 것이 모델의 최종 목표인데, 그 Y 값이 없기 때문입니다. 이 경우 해당 행을 삭제하는 것이 일반적입니다. 2. **변수 유형 (수치형 vs 범주형):** 처리 방법이 달라집니다. (예: 수치형은 k-NN, 범주형은 로지스틱 회귀) 3. **결측량 (Amount):** 만약 특정 변수가 60% 이상 결측치라면, 이 변수를 대체하는 것은 오히려 새로운 노이즈를 만드는 것일 수 있습니다. 이 경우 해당 변수(열)를 삭제하는 것을 고려할 수 있습니다.

4.2 방법 1: 결측치 표시자 변수 (Missingness Indicator)

이 방법은 결측치가 발생했다는 '사실 자체'가 중요한 정보를 담고 있을 수 있다고 가정합니다. (특히 MNAR의 경우에 유용)

* **아이디어:** "응답 거부"라는 제3의 그룹을 만듭니다. * **방법:** 1. 결측치가 있는 변수 X_1 을 복사하여 두 개의 변수를 만듭니다. 2. X_1^* (대체 변수): X_1 의 결측치를 0이나 평균 등 특정 값으로 모두 대체합니다. 3. $X_{1,miss}$ (표시자 변수): X_1 에서 값이 누락되었으면 1, 아니면 0을 갖는 이진(binary) 변수를 만듭니다. 4. 모델을 학습시킬 때, 원래 변수인 X_1 대신 이 두 변수(X_1^* 와 $X_{1,miss}$)를 함께 사용합니다.

□ 예제:

예제 예시: 결측치 표시자 변수 생성

아래 표는 X_1 과 X_2 의 결측치를 0으로 대체하고, 결측 여부를 $X_{1,miss}$, $X_{2,miss}$ 로 표시한 예입니다.
이제 모델은 X_1^* 과 $X_{1,miss}$ 를 보고, ' $X_{1,miss}$ 가 1일 때(즉, X_1 이 누락되었을 때)'는 X_1^* 의 0을 '관측된 0'과 다르게 취급할 수 있게 됩니다.

4.3 방법 2: 모델 기반 대체 (Model-based Imputation)

결측치를 '예측' 문제로 접근하는 방식입니다.

* **아이디어:** X_1 변수에 결측치가 있다면, 나머지 관측된 변수들(X_2, X_3, \dots)을 독립 변수로, X_1 을 종속 변수로 하는 예측 모델을 만듭니다. * **절차:** 1. 데이터를 두 그룹으로 나눕니다. * **학습용(Train):** X_1 이 관측된 모든 행. * **예측용(Test):** X_1 이 누락된 모든 행. 2. 학습용 데이터로 $X_2, X_3, \dots \rightarrow X_1$ 을 예측하는 모델(예: k-NN, 선형 회귀, 의사결정나무)을 학습시킵니다. 3. 학습된 모델을 예측용 데이터에 적용하여, 누락된 X_1 값을 예측하고 그 값으로 채워 넣습니다.

□ 예제:

예제 예시: k-NN을 사용한 대체

색깔(X, 관측됨)을 이용해 Y값(결측 존재)을 대체해봅시다. ($k=2$, 즉 가장 가까운 2개 사용)

[참고 이미지: k -NN 대체 시각화 - 색깔 기반으로 가장 가까운 2개 이웃의 평균값 사용] * **첫 번째 물음표 (?):** 색깔이 '중간 빨강'입니다. 가장 가까운 2개는 '진한 빨강'(Y=1)과 '밝은 빨강'(Y=0.5)입니다. * **대체:** 두 값의 평균인 $(1 + 0.5)/2 = 0.75$ 를 채워 넣습니다. * **두 번째 물음표 (?):** 색깔이 '노랑'입니다. 가장 가까운 2개는 '주황'(Y=0.1)과 '연두'(Y=10)입니다. * **대체:** 두 값의 평균인 $(0.1 + 10)/2 = 5.05$ 를 채워 넣습니다.

4.4 방법 3: 불확실성을 고려한 모델 기반 대체

위의 '모델 기반 대체'는 한 가지 큰 문제점을 가집니다. 바로 **"너무 완벽한"** 값을 채워 넣는다는 것입니다.

주의사항

결정론적 대체(Deterministic Imputation)의 함정

선형 회귀 모델로 결측치를 대체한다고 상상해봅시다. 예측된 값들은 모두 회귀선 위에 완벽하게 놓이게 됩니다. (아래 그림의 왼쪽)

하지만 실제 데이터는 어떻습니까? 항상 회귀선 주변에 흩어져 있습니다. (아래 그림의 오른쪽, 회색 점)

만약 우리가 모든 결측치를 회귀선 위의 완벽한 값으로만 채운다면, 데이터의 실제 '불확실성(분산)'이 사라지고 매우 인위적으로 좁은 분포를 갖게 됩니다. 이는 모델이 현실을 과도하게 확신하게 만듭니다.

[참고 이미지: 결정론적 vs. 확률적 예측 시각화 - 확률 모델은 예측에 불확실성을 반영]

해결책: 예측에 무작위성(불확실성)을 더하자!

* ** k -NN 대체 시:** k 개의 이웃을 찾은 뒤, 그 값들을 '평균'내는 대신 k 개 중 하나를 **무작위로 샘플링**하여 채워 넣습니다. * (위의 '노랑' 예시: 5.05를 채우는 대신, 50% 확률로 0.1, 50% 확률로 10을 뽑아 넣습니다.) * **선형 회귀 대체 시:** 예측값 \hat{y} 을 구한 뒤, 학습 데이터의 실제 잔차(residual, ϵ) 중 하나를 **무작위로 샘플링**하여 $\hat{y} + \epsilon$ 을 채워 넣습니다. * ** 의사결정나무 대체 시:** 예측값이 속한 최종 노드(leaf)에 여러 개의 학습 데이터가 있다면, 그 값을 '평균'내는 대신 그중 하나를 **무작위로 샘플링**하여 채워 넣습니다.

이러한 '불확실성을 고려한 대체'는 데이터의 원래 분포와 분산을 보존하는 데 훨씬 효과적입니다.

4.5 여러 변수에 결측치가 있는 경우 (Iterative Imputation)

만약 X_1, X_2, X_3 모두에 결측치가 있다면 어떻게 해야 할까요? 이는 "닭과 달걀의 문제"와 같습니다. X_1 을 예측하려면 X_2, X_3 가 필요한데, X_2, X_3 에도 결측치가 있습니다.

해결책: 반복적(Iterative)으로 예측합니다.

1. **초기화:** X_1, X_2, X_3 의 모든 결측치를 일단 '평균값'으로 채웁니다.
2. **1라운드 (X_1 예측):** (평균으로 채워진) X_2, X_3 를 이용해 X_1 의 결측치를 예측하고 업데이트합니다.
3. **1라운드 (X_2 예측):** (방금 업데이트된) X_1 과 (평균으로 채워진) X_3 를 이용해 X_2 의 결측치를 예측하고 업데이트합니다.
- 4.

1 라운드 (X3 예측): (업데이트된) X_1, X_2 를 이용해 X_3 의 결측치를 예측하고 업데이트합니다. 5. **2 라운드 이후:** 1 4의 과정을 X_1, X_2, X_3 의 대체값들이 더 이상 크게 변하지 않을 때까지 (수렴, converge) 여러 번 반복합니다.

4.6 Sklearn을 사용한 구현

이러한 복잡한 과정들은 sklearn.impute 모듈에 구현되어 있습니다.

```

1 from sklearn.impute import SimpleImputer
2 from sklearn.impute import IterativeImputer
3 from sklearn.impute import KNNImputer
4 from sklearn.impute import MissingIndicator
5
6 # 1. 단순대체평균 (, 중앙값, 최빈값등)
7 imputer_simple = SimpleImputer(strategy='mean')
8
9 # 2. 결측치표시자
10 indicator = MissingIndicator()
11
12 # 3. k-NN 기반대체모델 (기반)
13 imputer_knn = KNNImputer(n_neighbors=5)
14
15 # 4. 반복적대체가장 (정교한방법)
16 # 다른모든피처를사용하여각피처의결측치를예측
17 imputer_iterative = IterativeImputer(max_iter=10, random_state=0)

```

Listing 1: sklearn의 주요 Imputer

5 블랙박스 모델 해석 및 시각화

5.1 시각화의 목표

데이터 시각화는 여러 목표를 갖습니다.

* (모델링 전) 데이터 탐색 및 가설 수립 (EDA) * (모델링 후) **모델 결과 커뮤니케이션**

모델이 복잡해질수록 ”모델이 왜 이런 예측을 했는가?”를 설명하기 어려워집니다. 이 섹션은 모델링 후의 커뮤니케이션에 초점을 맞춥니다.

5.2 왜 모델 해석이 필요한가? (Parametric vs. Non-parametric)

1. **파라메트릭 모델 (Parametric Models):** * 예: 선형 회귀, 로지스틱 회귀 * 모델이 $Y = \beta_0 + \beta_1 X_1 + \dots$ 처럼 간단한 수식으로 정의됩니다. * 우리는 계수(coefficient) β_1 을 보고 ” X_1 이 1단위 증가할 때 Y 는 β_1 만큼 증가(또는 감소)한다”라고 명확하게 해석할 수 있습니다. (화이트박스 모델) 2. **비파라메트릭 모델 (Non-parametric Models):** * 예: k-NN, 의사결정나무, 랜덤 포레스트 * 모델이 복잡한 규칙(Tree)이나 거리 계산(k-NN)의 조합으로 이루어집니다. * ” β_1 ”처럼 해석할 수 있는 간단한 계수가 없습니다. * 이처럼 내부 작동 원리를 직관적으로 파악하기 어려운 모델을 **블랙박스(Black Box)** 모델이라고 부릅니다.

블랙박스 모델이라도, 우리는 모델이 ’어떻게’ 예측하는지 이해해야 합니다. 이때 사용하는 기법이 **부분 의존성 플롯(PDP)**입니다.

5.3 부분 의존성 플롯 (Partial Dependence Plots, PDP)

PDP는 ”다른 모든 변수들을 평균(또는 특정 값)으로 고정했을 때, 내가 관심 있는 변수 하나(X_1)를 변화시키면 모델의 예측값이 어떻게 변하는가?”를 보여주는 그래프입니다.

□ 예제:

예제 PDP의 직관적 비유: 오디오 믹서

PDP는 오디오 믹서(Mixer)와 같습니다. 훌륭한 음악(모델 예측)은 보컬, 드럼, 베이스(여러 변수)가 조합된 결과입니다.

PDP는 이 음악에서 **’베이스(X_1)’가 전체 사운드에 어떤 영향을 주는지** 알고 싶을 때, ’보컬’과 ’드럼’(다른 변수)의 볼륨을 ’중간’으로 고정해 놓고, ’베이스’의 볼륨만 0에서 100까지 쭉 돌려보면서 소리의 변화(Y 예측값)를 녹음하는 것과 같습니다.

5.3.1 예제 1: 단일 변수 모델 시각화

먼저 간단한 모델로 시작합니다. 심장병(AHD, 0 or 1)을 최대 심박수(MaxHR) 하나만으로 예측하는 k-NN($k=50$) 모델을 만들었습니다.

* **모델:** AHD MaxHR * **시각화:** ‘MaxHR’ 값을 70부터 200까지 촘촘하게 만들고(synthetic X), 각 값에 대해 모델이 예측하는 ’심장병 확률’을 계산하여 점을 찍어 연결합니다.

[참고 이미지: PDP 시각화 - MaxHR vs. 심장병 확률의 부정적 관계]

* **해석:** 이 그래프(PDP)는 ‘MaxHR’이 낮을수록 심장병 확률이 70% 이상으로 높고, ‘MaxHR’이 높을수록(즉, 건강할수록) 확률이 20% 근처로 낮아지는 **부정적(negative) 관계**를 모델이 학습했음을 보여줍니다.

5.3.2 예제 2: 다중 변수 모델의 PDP (상호작용이 없는 경우)

이제 더 복잡한 모델을 만듭니다. ‘MaxHR’뿐만 아니라 ‘Age’, ‘Sex’, ‘RestBP’ 등 10개의 변수를 사용했습니다.

* **모델:** AHD = $\text{MaxHR} + \text{Age} + \text{Sex} + \dots$ * **시각화 (PDP):** ‘MaxHR’의 영향을 보기 위해, 다른 9개 변수(‘Age’, ‘Sex’ 등)를 모두 **”전체 데이터의 중앙값(median)”**으로 고정합니다. (즉, ’평균적인 환자’를 가정) * 그런 다음, 이 ’평균적인 환자’의 ‘MaxHR’만 70에서 200으로 바꾸면서 심장병 확률을 예측합니다.

[참고 이미지: 다중 변수 PDP - 평균 환자에 대한 MaxHR 영향력 시각화]

* **해석:** 여러 변수를 추가했지만, ’평균적인 환자’에 대한 ‘MaxHR’의 영향력은 예제 1과 거의 유사한 부정적 관계를 보입니다.

5.3.3 예제 3: PDP를 이용한 상호작용(Interaction) 발견

PDP의 진정한 힘은 ’평균적인 환자’가 아닌 ’특정 그룹’에 대한 예측을 비교할 때 나옵니다.

* **가설:** ‘MaxHR’이 심장병에 미치는 영향은 ‘Age’(나이)에 따라 다르지 않을까? * **시각화 (PDP 비교):** 3개의 PDP를 한꺼번에 그립니다. 1. (주황색) ‘Age’ = **중앙값** (약 55세), 다른 변수도 중앙값 2. (초록색) ‘Age’ = **최대값** (약 77세), 다른 변수는 중앙값 3. (빨간색) ‘Age’ = **최소값** (약 29세), 다른 변수는 중앙값

[참고 이미지: 상호작용 PDP - 다양한 Age 값에 따른 MaxHR 효과 변화]

* **해석:** * 그래프가 세 그룹(다른 나이)에 대해 다르게 그려집니다! 이는 모델이 **‘MaxHR’와 ‘Age’ 간의 상호작용**을 학습했다는 뜻입니다. * ‘MaxHR’이 낮을 때는 (왼쪽, < 110) 나이와 상관없이 모두 확률이 높습니다. * ‘MaxHR’이 높을 때는 (오른쪽, > 160) 나이의 영향이 커집니다. 젊은 환자(빨간색, 최소값)는 확률이 15%까지 떨어지지만, 나이 많은 환자(초록색, 최대값)는 확률이 25% 정도로 상대적으로 높게 유지됩니다. * **결론:** 이 모델에 따르면, 최대 심박수가 높은 것(운동)은 젊은 사람에게 더 큰 심장병 예방 효과가 있습니다.

6 효과적인 시각화의 원칙 (부록)

6.1 나쁜 시각화의 예

모델 해석뿐만 아니라 모든 데이터 시각화에서 '나쁜 시각화'는 의미를 왜곡합니다. [참고 이미지: 잘못된 시각화 예시 - 하버드 GPA 인플레이션을 왜곡한 차트]

위 차트는 하버드의 GPA 인플레이션을 보여주려 했으나, 최악의 시각화 중 하나입니다.

* **문제점:** 2005년부터 2017년까지의 '3.67'과 2017년 이후의 '4.00'은 완전히 다른 척도(하나는 숫자, 하나는 등급)를 의미할 수 있으며, 막대그래프로 수치를 표현했지만 모든 막대의 높이가 동일하여 시각적 정보를 전혀 주지 못합니다. 이는 데이터를 조작하고 청중을 속이는 행위입니다.

6.2 좋은 시각화의 원칙

역사적으로 가장 위대한 데이터 시각화들은 다음과 같은 원칙을 따릅니다. (예: 존 스노우의 콜레라지도, 나이팅게일의 로즈 차트, 미나르의 나폴레옹 행군도)

1. **그래픽 무결성 (Graphical Integrity):** 데이터를 왜곡하거나 속이지 않아야 합니다. (예: Y축을 0에서 시작하기, 척도 통일하기)
2. **단순함 (Keep it simple):** 불필요한 장식(3D 효과, 그림자, 과도한 색상)을 제거하여 '데이터 잉크 비율'을 높여야 합니다.
3. **올바른 디스플레이 사용 (Use the right display):** * (Good) **위치 (Position), 길이 (Length):** 사람이 가장 정확하게 인지. (예: 막대 차트, 산점도) * (Bad) **각도 (Angle), 면적 (Area):** 사람이 크기를 과소/과대 평가하기 쉬움. (예: 파이 차트, 도넛 차트)
4. **전략적인 색상 사용 (Use color strategically):** * **범주형 (Qualitative):** 서로 구분이 명확한 색상 사용 (예: 정당, 과일 종류) * **순차형 (Sequential):** 하나의 색상을 연한 색 진한색으로 표현 (예: 인구 밀도 $0 \rightarrow 100$) * * * (Diverging) : * * (0) (: $-100 \rightarrow 0 \rightarrow +100$) * / .5. * * (Know your audience) : * * (Annotations, CallOutBoxes) .
5. * * (Tell a story) :

7 빠르게 훑어보기 (1-Page Summary)

1. 결측치 (Missingness)란?

데이터에 값이 누락된 '구멍' (NaN). 그냥 삭제하거나 평균값으로 채우면 **편향(Bias)**이 발생하여 모델이 현실을 잘못 예측하게 됨.

2. 결측의 3유형 (원인)

- **MCAR (완전 임의):** 완전 무작위 (예: 오타). 삭제해도 편향 없음 (데이터는 손실됨).
- **MAR (임의):** 관측된 변수(예: 성별)와 관련됨. 모델링으로 해결 가능.
- **MNAR (비임의):** 값 자체(예: 고소득) 또는 숨겨진 변수(예: 부작용)와 관련됨. 해결 매우 어려움.

현실: 어떤 유형인지 증명 불가. 보통 MAR로 가정하고 모델링.

3. 결측치 처리 (Imputation) 전략

- **표시자 변수 (Indicator):** "결측됨" 자체를 정보로 활용. (결측=1, 아닌=0인 새 변수 추가)
- **모델 기반 대체 (k-NN, Regression):** 다른 변수들로 결측치를 '예측'하여 채움.
- **불확실성 고려 대체 (Best):** 예측 값(\hat{y})에 무작위 성(예: 잔차 ϵ)을 더해 $\hat{y} + \epsilon$ 로 채움. (데이터의 실제 분포를 보존하기 위해)
- **반복적 대체 (Iterative):** 여러 변수에 결측치가 있을 때, 수렴할 때까지 서로를 예측하며 반복적으로 채움 (IterativeImputer).

4. 블랙박스 모델 해석 (PDP)

PDP (부분 의존성 플롯): 복잡한 모델(k-NN, 트리 등)을 해석하는 도구.

"다른 변수들은 '평균'으로 고정하고, 관심 변수 X 하나만 바꿨을 때 모델의 예측 Y가 어떻게 변하는지" 보여주는 그래프.

만약 '나이(A) 그룹' 별로 그린 PDP와 '나이(B) 그룹' 별로 그린 PDP의 모양이 다르다면, 모델이 '나이'와 '관심 변수 X' 간의 **상호작용(Interaction)**을 학습했다는 의미.

- 강의명: CS109A: 데이터 과학 입문
- 주차: Lecture 22
- 교수명: Pavlos Protopapas, Kevin Rader, Chris Gumb
- 목적: Lecture 22의 핵심 개념 학습

Contents

1	용어 정리	3
2	핵심 개념: 랜덤 포레스트의 원리	4
2.1	배깅(<i>Bagging</i>)의 한계: 트리의 상관관계	4
2.2	랜덤 포레스트: 비상관화(<i>Decorrelation</i>)의 도입	4
2.3	하이퍼파라미터 튜닝	4
3	변수 중요도(<i>Variable Importance</i>) 평가	6
3.1	1. 불순도 감소량 평균(<i>Mean Decrease in Impurity</i> , MDI)	6
3.2	2. 순열 중요도(<i>Permutation Importance</i>)	6
3.3	랜덤 포레스트 vs. 배깅 변수 중요도 비교	7
4	불균형 데이터(<i>Class Imbalance</i>) 처리	8
4.1	처리 방법	8
5	결측값(<i>Missing Data</i>) 처리: 서러게이트 분할	9
5.1	서로게이트 분할의 개념	9
5.2	서러게이트 분할의 이점	9
6	FAQ 및 초심자 체크리스트	10
6.1	자주 묻는 질문(FAQ)	10
6.2	학습 및 구현 체크리스트	10

개요: 랜덤 포레스트와 변수 중요도

문서 핵심 요약

랜덤 포레스트(Random Forest)는 배깅(Bagging)의 변형으로, 다수의 결정 트리(Decision Tree)를 훈련하고 결과를 합계하여 예측의 분산(Variance)을 줄이는 양상을 학습 기법입니다. 핵심은 각 트리의 분할 시마다 전체 특징(Feature, 변수)의 부분집합을 무작위로 선택하여, 트리 간의 상관관계를 낮춰 성능을 극대화하는 것입니다. 트리 간의 낮은 상관관계는 모델의 안정성(Robustness)을 높이고 과적합(Overfitting) 위험을 줄입니다. 변수 중요도(Variable Importance)는 모델의 해석력을 높이는 수단이며, 불순도 감소량 평균(Mean Decrease in Impurity, MDI)과 순열 중요도(Permutation Importance) 두 가지 방법으로 계산됩니다. 또한, 불균형 데이터(Imbalanced Data) 문제와 결측값(Missing Data) 처리 방법도 함께 다룹니다.

1 용어 정리

주요 용어 및 직관적 설명

2 핵심 개념: 랜덤 포레스트의 원리

2.1 배깅(Bagging)의 한계: 트리의 상관관계

배깅은 부트스트랩 샘플(Bootstrap Sample)을 사용하여 다수의 결정 트리를 학습하고 그 결과를 집계(Aggregate)하여 분산을 줄이는 양상을 기법입니다. 그러나 배깅에서 생성된 트리는 서로 상관관계(Correlation)가 높게 나타나는 경향이 있습니다.

상관관계가 높은 이유에 대한 비유

강력한 예측 변수(Strong Predictor)가 있는 경우, 모든 부트스트랩 샘플에서도 이 예측 변수가 가장 높은 불순도 감소 효과를 보일 것입니다. 마치 모든 의사(Estimator)가 동일한 환자 데이터를 기반으로 훈련받고, '휴식, 수분 섭취'와 같은 가장 확실한 조언을 첫 번째 조치로 내리는 것과 같습니다. 따라서, 대부분의 트리는 최상위 노드(Root Node)에서 동일한 특징으로 분할을 시작하게 되어, 트리가 서로 비슷한 구조와 예측을 하게 됩니다. 이러한 상관관계는 양상들이 기대하는 분산 감소 효과($1/\sqrt{n}$)를 약화시킵니다.

2.2 랜덤 포레스트: 비상관화(Decorrelation)의 도입

랜덤 포레스트는 배깅을 개선하여 트리 간의 상관관계를 낮추는 방법입니다. 트리의 상관관계를 낮추는 것이 분산(Variance)을 효과적으로 줄이는 핵심입니다.

랜덤 포레스트의 핵심 아이디어

각 분할 노드(Split Node)마다, 전체 특징 J 개 중 무작위로 선택된 $J' < J$ 개의 특징 부분집합만 고려합니다. 즉, 트리가 자라날 때마다 최고의 특징을 선택할 때 전체 특징이 아닌 일부 특징 중에서만 선택하게 강제합니다. 이로 인해 강력한 특징이 매번 선택될 확률이 낮아지고, 결과적으로 트리가 다양해지고 서로 덜 유사해집니다.

랜덤 포레스트 요약 단계

1. B 개의 부트스트랩 데이터셋을 생성합니다 (배깅과 동일).
2. B 개의 결정 트리를 초기화합니다.
3. 각 트리 내의 각 분할(Split)마다:
 - (a) 전체 특징 J 개 중 무작위로 J' 개의 특징 부분집합을 선택합니다 ($J' < J$).
 - (b) 이 J' 개의 특징 중에서 최적의 특징과 최적의 임계값(Threshold)를 선택하여 분할합니다.
4. 최종적으로 모든 트리의 예측을 합계합니다 (분류는 다수결, 회귀는 평균).

이때 J' 는 각 분할마다 새롭게 무작위로 선택됩니다.

2.3 하이퍼파라미터 튜닝

랜덤 포레스트는 여러 개의 하이퍼파라미터를 가지며, 이는 모델의 성능과 훈련 속도에 영향을 미칩니다.

주요 하이퍼파라미터

- 분할 시 무작위 선택할 특징의 개수 J' : 트리 간의 상관관계를 조절하는 가장 중요한 파라미터입니다.
- 양상을 내 트리의 총 개수 B : 분산 감소량을 결정합니다. 트리가 많을수록 분산은 줄어들지만 계산 시간이 늘어납니다.
- 트리 정지 조건: 최대 깊이(*Maximum Depth*), 최소 리프 노드 크기(*Minimum Leaf Node Size*) 등.
- 분할 기준: 지니 불순도(*Gini Impurity*) 또는 엔트로피(*Entropy*).

하이퍼파라미터 튜닝과 전문가의 권장 사항

1. 최적의 J' 선택: 교차 검증(*Cross-Validation*)을 사용해야 하지만, 일반적인 경험 법칙(*Rule of Thumb*)이 있습니다.
 - 분류(*Classification*): $\sqrt{N_j}$ (전체 특징 개수의 제곱근)
 - 회귀(*Regression*): $N_j/3$
2. 트리의 개수 B : OOB 오류가 더 이상 감소하지 않고 안정화되는 지점까지 늘립니다. 트리의 개수는 과적합(*Overfitting*)을 유발하지 않습니다. 단지 분산만 감소시킬 뿐입니다.
3. 정지 조건 및 기준: 보통 최대 깊이(*Maximum Depth*)나 지니 불순도(*Gini*)를 기본 값으로 사용하고, 모델이 작동한 후에 더 세밀한 튜닝을 시도할 수 있습니다.

OOB 오류를 활용한 검증

랜덤 포레스트는 OOB(Out-of-Bag) 샘플, 즉 부트스트랩 과정에서 특정 트리 훈련에 사용되지 않은 데이터 포인트들을 사용하여 별도의 검증 세트 없이 모델을 평가할 수 있습니다. 이것이 교차 검증을 대체할 수 있는 효율적인 방법입니다.

3 변수 중요도(*Variable Importance*) 평가

양상블 모델은 단일 결정 트리처럼 쉬운 규칙(Rule) 형태로 해석하기 어렵기 때문에, 어떤 특징이 예측에 가장 큰 영향을 미쳤는지 평가하여 모델의 해석력(*Interpretability*)을 높여야 합니다.

3.1 1. 불순도 감소량 평균(*Mean Decrease in Impurity*, MDI)

MDI는 특정 특징이 트리의 불순도(예: 지니 불순도)를 평균적으로 얼마나 감소시켰는지를 측정하여 중요도를 산출하는 방법입니다.

MDI 계산 절차

1. 노드별 불순도 감소량 계산: 단일 트리 내의 각 노드 q 에서 분할로 인한 불순도 감소량 ΔI_q 를 계산합니다.

$$\Delta I_q = \left(\frac{n}{N} \right) \left[Gini_n - \sum_{m \in \text{children}} \left(\frac{m}{n} \right) Gini_m \right]$$

(n : 노드의 샘플 수, N : 전체 데이터 샘플 수, m : 자식 노드의 샘플 수)

2. 특징별 중요도 합산: 특정 특징 j 가 사용된 모든 노드 n 의 불순도 감소량을 합하여 해당 특징의 중요도 $F_j^{(t)}$ 를 계산합니다.
3. 정규화: 중요도 합계를 모든 특징의 중요도 합계로 나누어 0에서 1 사이 값으로 정규화합니다.
4. 양상블 평균: 모든 트리 T 에 대해 정규화된 특징 중요도 $\hat{F}_j^{(t)}$ 를 평균하여 최종 MDI 중요도를 구합니다.

$$\mathcal{F}_j = \frac{\sum_t \hat{F}_j^{(t)}}{T}$$

MDI의 장단점

- 장점: 계산 속도가 빠릅니다. 모든 필요한 값은 랜덤 포레스트 훈련 중에 계산됩니다.
- 단점: 수치형 특징(Numerical Feature)이나 범주형 특징 중 고유값이 많은 특징(High Cardinality Categorical Feature)에 편향(Bias)되어 중요도를 과대평가하는 경향이 있습니다. 이들은 분할 지점(Split Point)이 많기 때문에 우연히 불순도를 크게 감소시키는 분할을 찾을 가능성이 높습니다.

3.2 2. 순열 중요도(*Permutation Importance*)

순열 중요도는 특징의 값을 무작위로 섞어(Permute) 해당 특징과 결과 변수 간의 관계를 끊었을 때, 모델의 검증/OOB 성능이 얼마나 하락하는지를 측정하여 중요도를 산출하는 방법입니다.

순열 중요도 계산 절차

1. 기준 성능 기록: 원본 데이터셋에서의 OOB 또는 검증 정확도 s 를 기록합니다.
2. 특징 순열: 관심 있는 특징 j 의 데이터 열을 무작위로 섞습니다.
3. 순열 성능 측정: 섞인 데이터셋으로 모델을 실행하여 새로운 OOB/검증 정확도 $s_{k,j}$ 를 기록합니다.

4. 반복 및 평균: 2, 3단계를 K 번 반복하고 평균 순열 정확도 s_j 를 계산합니다.

$$s_j = \frac{1}{K} \sum_{k=1}^K s_{k,j}$$

5. 중요도 산출: 기준 성능과 평균 순열 성능의 차이를 계산하여 중요도를 산출합니다.

$$\text{특징 중요도} = s - s_j$$

순열 중요도의 장단점

- 장점: MDI의 편향이 없으며, 실제 모델 성능에 미치는 영향을 직접적으로 측정하므로 더 직관적이고 신뢰할 만한 중요도를 제공합니다.
- 단점: MDI보다 계산 비용이 더 많이 듭니다. (각 특징마다 K 번의 예측이 필요)

3.3 랜덤 포레스트 vs. 배깅 변수 중요도 비교

트리 간 상관관계가 높은 배깅(*Bagging*)은 소수의 강력한 예측 변수(예: ChestPain, Ca)에 중요도가 집중되는 경향을 보입니다. 반면, 비상관화 과정을 거친 랜덤 포레스트(*Random Forest*)의 중요도는 여러 특징에 걸쳐 더 부드럽고 분산된 분포를 보여줍니다. 이는 트리들이 다양한 특징을 고려하도록 강제되었기 때문입니다.

랜덤 포레스트가 잘 작동하지 않는 경우

전체 특징의 수가 매우 많지만, 실제로 중요한 특징의 수가 적을 때 (예: 1000개의 특징 중 10개만 중요), 랜덤 포레스트는 오히려 성능이 떨어질 수 있습니다. 각 분할에서 무작위로 선택된 특징 부분집합에 중요한 특징이 포함되지 않을 확률이 높아지기 때문에, 생성된 트리들이 대부분 '약한 모델'이 될 수 있습니다. 이 경우 PCA나 다른 특징 선택 기법을 사용해 불필요한 특징을 미리 제거하는 것이 좋습니다.

4 불균형 데이터(*Class Imbalance*) 처리

데이터셋에서 특정 클래스(소수 클래스)의 샘플 수가 다른 클래스(다수 클래스)에 비해 현저히 적을 때 발생하는 문제입니다. 이 경우 모델이 다수 클래스만 예측해도 높은 '정확도'를 얻을 수 있어, 실제로는 소수 클래스 예측 능력이 매우 낮아집니다.

불균형 데이터 문제 진단

정확도(*Accuracy*)는 좋은 평가 지표가 아닙니다. 99% 대 1%의 클래스 불균형에서 모델이 항상 99%의 다수 클래스만 예측해도 정확도는 99%가 나옵니다. 따라서, 불균형 데이터셋에서는 **F1-** 점수나 **ROC 곡선 하 면적(AUC)**과 같은 지표를 사용해야 합니다.

4.1 처리 방법

1. 언더샘플링(*Under-sampling*): 다수 클래스 샘플 수 감소

- 무작위 언더샘플링: 다수 클래스에서 무작위로 샘플을 제거하여 클래스 균형을 맞춥니다. (단점: 중요한 정보를 포함하는 샘플이 손실될 수 있습니다.)
- 니어 미스(*Near Miss*): 결정 경계(*Decision Boundary*)에서 멀리 떨어진, 즉 분류에 덜 유익한 다수 클래스 샘플을 제거하여 정보 손실을 최소화합니다. (경계 근처의 샘플은 보존)

2. 오버샘플링(*Over-sampling*): 소수 클래스 샘플 수 증가

- 무작위 오버샘플링: 소수 클래스의 샘플을 복원 추출(*with replacement*)하여 수를 늘립니다. (단점: 단순 복제이므로 과적합을 유발하거나 데이터의 다양성을 해칠 수 있습니다.)
- SMOTE (*Synthetic Minority Oversampling Technique*): 소수 클래스 샘플 주변에 새로운 합성 데이터(*Synthetic Data*)를 생성하여 수를 늘립니다. 데이터 영역을 확장하여 모델의 일반화 성능을 높입니다.

3. 클래스 가중치(*Class Weighting*)

- 모델의 손실 함수(*Loss Function*)에서 소수 클래스의 오류에 더 높은 가중치를 부여하여 학습 시 소수 클래스에 더 많은 관심을 기울이도록 합니다.
- 사이킷런(*scikit-learn*)에서는 `class_weight='balanced'` 옵션을 통해 자동으로 클래스 빈도에 반비례하여 가중치를 조정할 수 있습니다.

$$W_K = \frac{N}{K \times N_K}$$

(N : 전체 샘플 수, K : 클래스 수, N_K : 클래스 K 의 샘플 수)

실제 사용 전략

데이터 균형을 맞출 때에는 세 가지 방법(언더샘플링, 오버샘플링, 가중치) 중 하나 또는 그 조합을 사용합니다. 예를 들어, 오버샘플링과 언더샘플링을 결합하거나, 데이터 재조정 후 클래스 가중치를 적용할 수 있습니다. 중요한 것은 항상 데이터를 균형 있게 처리하는 것입니다 (40%/60% 불균형에서도 적용 권장).

5 결측값(*Missing Data*) 처리: 서러게이트 분할

결정 트리 기반 모델은 결측값을 처리하는 독특한 방법인 서로게이트 분할(*Surrogate Split*)을 사용할 수 있습니다.

5.1 서로게이트 분할의 개념

서로게이트 분할은 트리를 훈련하는 과정에서 최적의 분할 특징(*Optimal Splitter*)의 결측값(*Missing Value*)을 대신할 수 있는 차선책의 특징(*Alternative Feature*)을 미리 찾아 순위를 매겨 놓는 방법입니다.

작동 원리

- 최적 분할 특징 선택:** 노드 N 에서 불순도를 가장 크게 줄이는 최적의 분할 특징 P_{opt} 를 찾습니다.
- 분할 분포 기록:** P_{opt} 를 사용해 노드를 분할했을 때, 각 자식 노드로 이동하는 반응 변수(Response Variable)의 분포(예: Yes/No 개수)를 기록합니다.
- 대리 특징 찾기:** 나머지 모든 특징 P_i 에 대해 P_{opt} 와 동일한 분할을 시도했을 때, 반응 변수의 분포가 P_{opt} 의 분포와 가장 유사한 특징을 찾습니다.
- 유사도 측정:** 두 특징의 분할 분포가 얼마나 유사한지(즉, 분포를 일치시키기 위해 얼마나 많은 '플립'이 필요한지)를 측정하여 유사도를 산출하고 순위를 매깁니다.
- 예측 시 사용:** 실제 예측 시점에 입력 데이터의 P_{opt} 값이 결측이면, 미리 정의된 순위에 따라 가장 유사한 서러게이트 분할 특징을 대신 사용하여 데이터를 분할합니다.

서러게이트 분할 예시

주요 특징이 'Arteries Blocked'라고 가정합니다. 이 특징으로 분할했을 때, 환자들의 '심장병 유무' 분포가 나옵니다.

- **최적 분할 특징 (Arteries Blocked):** FALSE → [3 No, 1 Yes], TRUE → [0 No, 2 Yes]
- **대리 특징 후보 (Chest Congested):** FALSE → [3 No, 2 Yes], TRUE → [0 No, 1 Yes]
- 두 분포를 비교하여 유사도를 측정합니다. 이 예시에서는 'Chest Congested'가 'Arteries Blocked'와 가장 유사한 분할 분포를 보였으므로 최적의 서러게이트가 됩니다.

따라서, 예측할 데이터에서 'Arteries Blocked' 값이 결측이면, 대신 'Chest Congested' 특징을 사용하여 트리를 따라 내려갑니다.

5.2 서러게이트 분할의 이점

- **해석력 향상:** 서러게이트는 최적 분할 특징과 비슷한 역할을 하는 보조 특징을 보여주므로, 주 분할기의 작동 방식을 이해하는 데 도움이 됩니다.
- **다중 공선성(Multi-collinearity) 활용:** 다중 공선성이 있는 경우(특징들이 서로 높은 상관관계를 가지는 경우), 대체할 서러게이트를 찾을 가능성성이 높고 성능도 좋습니다.
- **명시적 대체 불필요:** 데이터 분석가가 별도의 결측값 대체(*Imputation*) 방법을 고민할 필요 없이, 트리가 자체적으로 결측값을 처리할 수 있습니다.

6 FAQ 및 초심자 체크리스트

6.1 자주 묻는 질문 (FAQ)

1. Q: 랜덤 포레스트에서 트리의 개수(*Number of Trees*)가 많아지면 과적합되나요?

A: 아닙니다. 트리의 개수를 늘리는 것은 분산(*Variance*)만 줄이는 역할을 합니다. 이는 앙상블의 예측을 안정화할 뿐이며, 다른 하이퍼파라미터처럼 모델의 복잡도(*Complexity*)를 제어하지 않으므로 과적합 위험을 증가시키지 않습니다.

2. Q: 랜덤 포레스트는 어떻게 클래스 예측 확률(*Class Probabilities*)을 반환하나요?

A: 각 개별 트리는 최종적으로 클래스 예측 결과를 내놓습니다. 랜덤 포레스트 분류기(*Classifier*)는 모든 트리가 예측한 클래스 예측의 평균을 계산하여 이를 확률의 근사치(*Proxy*)로 사용합니다. 이 값을 임계값(*Threshold*)과 결합하여 ROC 곡선 등을 그릴 수 있습니다.

3. Q: MDI와 순열 중요도 중 어떤 것을 사용해야 하나요?

A: 순열 중요도(*Permutation Importance*)가 더 신뢰할 수 있습니다. MDI는 계산이 빠르다는 장점이 있지만, 수치형이나 고유값이 많은 범주형 특징에 편향되어 중요도를 과대평가하는 경향이 있습니다. 순열 중요도는 모델의 실제 성능 변화를 측정하므로 더 직관적이고 정확합니다.

4. Q: 결정 트리는 단일 모델로 불안정한데, 왜 배깅이나 랜덤 포레스트에서는 안정적인가요?

A: 단일 결정 트리는 훈련 데이터의 작은 변화에도 민감하게 반응하여 매우 다른 구조로 학습될 수 있습니다(고분산). 앙상블은 부트스트랩을 통해 생성된 다양한 불안정한 트리를 평균하여, 그 '노이즈'로 인한 예측의 변동성(*Variability*)을 상쇄하고 평균적인 안정적인 예측을 얻습니다.

6.2 학습 및 구현 체크리스트

랜덤 포레스트 구현 전 점검 사항

- **핵심 원리 이해:** 랜덤 포레스트가 배깅과의 차이점(특정 무작위 부분집합 선택)과 분산 감소 메커니즘을 명확히 설명할 수 있는가?
- **하이퍼파라미터 튜닝:** 가장 중요한 하이퍼파라미터 J' 와 B 의 역할 및 권장 기본값($\sqrt{N_j}$, $N_j/3$)을 아는가?
- **평가 지표 선택:** 데이터 불균형 여부를 확인하고, 불균형 시 F1-점수 또는 AUC를 주 평가 지표로 사용하는가?
- **불균형 처리:** 언더샘플링(Near Miss)이나 오버샘플링(SMOTE) 또는 클래스 가중치 중 최소 한 가지 방법을 적용하여 데이터를 균형 있게 조정했는가?
- **변수 중요도 평가:** MDI의 편향성을 인지하고, 가능하면 순열 중요도를 사용하여 특징의 기여도를 평가했는가?
- **성능 비교:** 단일 결정 트리, 배깅, 랜덤 포레스트의 RMSE 또는 정확도를 비교하여 분산 감소 효과를 확인했는가?

- 강의명: CS109A: 데이터 과학 입문
- 주차: Lecture 23
- 교수명: Pavlos Protopapas, Kevin Rader, Chris Gumb
- 목적: Lecture 23의 핵심 개념 학습

Contents

1	필수 용어 정리	2
2	부스팅(Boosting)이란 무엇인가?	3
2.1	왜 필요한가? (배경)	3
2.2	부스팅의 핵심 메커니즘	3
3	그레디언트 부스팅 (Gradient Boosting)	4
3.1	핵심 원리: 잔차(Residual) 학습하기	4
3.2	동작 알고리즘 (단계별 설명)	4
4	시각적 예시와 흐름	5
4.1	데이터 상황	5
4.2	학습 과정 시각화	5
4.3	수식 요약	5
5	심화: 왜 '그레디언트(Gradient)' 부스팅인가?	6
5.1	경사 하강법 (Gradient Descent) 복습	6
5.2	함수 공간(Function Space)에서의 하강	6
6	실전 가이드: 모델 튜닝하기	7
6.1	1. 학습률 (Learning Rate, λ)	7
6.2	2. 반복 횟수 (Number of Iterations)	7
6.3	튜닝 전략	7
7	자주 묻는 질문 (FAQ) 및 오해 바로잡기	7
8	학습 마무리 체크리스트	8

부스팅(Boosting)과 그레디언트 부스팅(Gradient Boosting)

약한 모델들을 모아 강력한 모델을 만드는 앙상블 기법 완전 정복

▣ 핵심 요약

문서 핵심 요약 이 문서는 머신러닝의 강력한 앙상블 기법인 '부스팅(Boosting)'의 개념부터 '그레디언트 부스팅(Gradient Boosting)'의 수학적 원리까지 다룹니다.

- 핵심 아이디어:** 단순한 모델(Weak Learner)을 순차적으로 연결하여, 앞선 모델의 실수를 뒤따르는 모델이 바로잡습니다.
- 목표:** 편향(Bias)을 줄여 예측 성능을 극대화합니다. (랜덤 포레스트가 분산(Variance)을 줄이는 것과 대조적)
- 원리:** 잔차(Residual)를 새로운 모델의 목표값(Target)으로 학습하며, 이는 수학적으로 경사 하강법(Gradient Descent)과 동일합니다.
- 실전:** 학습률(Learning Rate) 조정을 통해 과적합(Overfitting)을 방지하는 것이 중요합니다.

1 필수 용어 정리

본격적인 학습에 앞서, 아래 용어들을 먼저 숙지하면 이해가 훨씬 수월합니다.

용어 (한글)	원어 (English)	쉬운 설명	비고
약한 학습기	Weak Learner	성능이 턱 걸이 수준인 단순한 모델 (예: 깊이가 1인 트리)	부스팅의 재료
강한 학습기	Strong Learner	약한 학습기를 모아 만든 고성능 모델	최종 결과물
잔차	Residual	정답과 내 예측값의 차이 ($y - \hat{y}$), 즉 '오차'	다음 모델의 목표
앙상블	Ensemble	여러 모델을 조합하여 성능을 높이는 기법	협동 작업
스텀프	Stump	가지가 딱 한 번만 뻗은 아주 얕은 결정 트리	깊이 (Depth)=1
학습률	Learning Rate	모델을 업데이트할 때 얼마나 반영할지 결정하는 보폭 (λ)	과적합 방지용

Table 1: 부스팅 학습을 위한 필수 용어 사전

2 부스팅(Boosting)이란 무엇인가?

2.1 왜 필요한가? (배경)

우리는 이전에 **랜덤 포레스트(Random Forest)**와 **배깅(Bagging)**을 배웠습니다. 이들은 깊고 복잡한 트리(Overfitting 되기 쉬움)를 여러 개 만들어 평균을 냄으로써 **분산(Variance)**을 줄이는 전략을 썼습니다.

하지만 **부스팅**은 정반대의 접근을 취합니다.

- 아주 단순하고 명청한 모델(High Bias, Low Variance)에서 시작합니다.
- 이 모델의 **실수(편향, Bias)**를 줄이는 방향으로 모델을 계속 추가합니다.
- 결과적으로 **편향을 획기적으로 줄여** 강력한 예측 모델을 만듭니다.

직관적 비유: 어려운 시험 통과하기 여러분이 통과하기 매우 어려운 시험을 앞두고 있다고 상상해 봅시다. 한 명의 천재가 문제를 다 푸는 것이 아니라, 평범한 학생들의 지혜를 모으는 방식입니다.

1. **학생 1 (단순한 규칙):** ”일단 기출문제를 보니, 4번은 정답이 아니야.” → 정확도 60%
2. **학생 2 (실수 보완):** ”학생 1이 틀린 문제들을 보니, 보기에 ‘과적합’이라는 단어가 있으면 그게 정답 이더라.” → 정확도 65%
3. **학생 3 (추가 보완):** ”학생 1, 2가 틀린 걸 보니, ‘교차 검증’이 답인 경우가 많아.” → 정확도 70%

이 세 학생의 의견을 적절히 합치면(가중치 부여), 혼자서는 풀 수 없던 문제를 90점 이상으로 통과할 수 있습니다. 이것이 바로 **부스팅**입니다.

2.2 부스팅의 핵심 메커니즘

부스팅은 **순차적(Iterative)**이고 **덧셈적(Additive)**인 과정입니다.

$$T_{final}(x) = \sum_{h \in H} \lambda_h T_h(x)$$

- T_h : 약한 학습기 (개별 학생의 의견)
- λ_h : 가중치 (그 학생의 의견을 얼마나 신뢰할 것인가)
- T_{final} : 최종 모델 (강한 학습기)

중요한 점: 한 번에 모든 모델을 만드는 것이 아니라, **앞선 모델이 저지른 실수를 다음 모델이 해결**하도록 순서대로 만듭니다.

3 그레디언트 부스팅 (Gradient Boosting)

부스팅 중에서도 가장 널리 쓰이고 강력한 방법이 **그레디언트 부스팅**입니다. 이름이 어렵게 들리지만, 원리는 간단합니다. **”이전 모델의 오차(잔차)를 새로운 모델이 학습한다”**는 것입니다.

3.1 핵심 원리: 잔차(Residual) 학습하기

비유: 골프 퍼팅 목표 지점(홀컵)까지 공을 보내야 합니다.

1. 첫 번째 샷 (T_0): 공을 쳤는데 홀컵까지 10m가 남았습니다. (잔차 = 10m)
2. 두 번째 샷 (T_1): 이제 목표는 홀컵이 아니라, '남은 거리 10m'을 보내는 것입니다. 쳤는데 2m가 남았습니다. (잔차 = 2m)
3. 세 번째 샷 (T_2): 이제 목표는 '남은 거리 2m'입니다.

이렇게 계속 남은 거리(잔차)를 메우는 샷을 더해나가는 것이 그레디언트 부스팅입니다.

3.2 동작 알고리즘 (단계별 설명)

데이터가 입력 값 x 와 정답 y 로 구성되어 있다고 합시다.

Step 1: 아주 단순한 첫 번째 모델(T_0)을 만듭니다.

- 예: 그냥 전체 데이터의 평균값으로 예측합니다. 당연히 많이 틀립니다.

Step 2: 잔차(Residual)를 계산합니다.

- $r_0 = y - T_0(x)$
- 즉, (실제 정답) - (첫 번째 모델의 예측값) = (아직 맞추지 못한 남은 오차)입니다.

Step 3: 잔차를 예측하는 두 번째 모델(T_1)을 만듭니다.

- 이번에는 y 를 맞추는 게 아니라, r_0 (오차)를 맞추도록 학습합니다.
- 즉, ”얼마나 더 더해야 정답에 가까워지는가?”를 배웁니다.

Step 4: 모델을 업데이트합니다.

- $T_{new} = T_0 + \lambda \times T_1$
- 여기서 λ (람다)는 **학습률(Learning Rate)**입니다. 두 번째 모델의 의견을 100% 반영하지 않고, 조금만 반영하여 과적합을 막습니다.

Step 5: 반복합니다.

- 다시 새로운 잔차를 계산하고, 그 잔차를 맞추는 모델 T_2, T_3, \dots 를 계속 더해 나갑니다.

주의: 실제로는 거대한 나무 하나를 만드는 게 아니다 ”트리를 합친다”고 해서 실제로 노드와 가지를 물리적으로 합쳐서 거대한 트리 하나를 만드는 것이 아닙니다. 추론(예측) 시에는 T_0 의 출력값, T_1 의 출력값, ... T_n 의 출력값을 모두 계산한 뒤 **숫자들을 더해서** 최종 값을냅니다.

4 시각적 예시와 흐름

그레디언트 부스팅이 데이터를 어떻게 학습하는지 1차원 데이터 예시로 살펴봅니다.

4.1 데이터 상황

- x : 입력 변수 (0~8 사이의 값)
- y : 출력 변수 (구불구불한 곡선 형태의 데이터)

4.2 학습 과정 시각화

단계	모델의 행동	결과 해석
Round 1	단순한 스텁프(Stump) 하나로 전체 평균을 예측 (T_0)	데이터의 복잡한 패턴을 전혀 못 잡음. 잔차(오차)가 매우 큼.
Round 2	Round 1의 잔차(실제값 - 예측값)를 목표로 하는 새 스텁프 학습 (T_1)	큰 오차가 있던 부분을 보정함. 전체 모델 모양이 데이터에 약간 가까워짐.
Round 3	현재까지 모델 ($T_0 + \lambda T_1$)의 잔차를 다시 계산해 T_2 학습	세밀한 굴곡을 맞추기 시작함. 오차가 점점 0에 가까워짐.
...	반복 (Iteration)	점점 정밀한 모델 완성

Table 2: 그레디언트 부스팅의 단계별 학습 흐름

4.3 수식 요약

$$\text{최종 예측 } \hat{y} = T_0(x) + \lambda T_1(x) + \lambda T_2(x) + \cdots + \lambda T_N(x)$$

- 각 T_i 는 이전 단계까지 해결하지 못한 '나머지 오차'를 해결하는 전문가입니다.
- λ (학습률)가 작을수록, 더 많은 트리가 필요하지만 더 정교하고 안정적인 모델이 됩니다.

5 심화: 왜 '그레디언트(Gradient)' 부스팅인가?

"잔차를 학습하는 것"이 왜 "경사 하강법(Gradient Descent)"과 같은지 이해하면, 이 알고리즘의 본질을 깨닫게 됩니다.

5.1 경사 하강법 (Gradient Descent) 복습

우리가 산 정상에서 가장 낮은 계곡(손실 함수 L 의 최소값)으로 내려가려 합니다.

- 눈을 가리고 있다면, 발끝으로 경사를 느낍니다.
- 경사가 가장 가파르게 올라가는 방향(Gradient, ∇L)의 **반대 방향**으로 발을 내디뎌야 내려갈 수 있습니다.
- 공식: $w_{new} = w_{old} - \lambda \times \nabla L$ (파라미터 업데이트)

5.2 함수 공간(Function Space)에서의 하강

일반적인 머신러닝은 파라미터(w , 가중치)를 수정합니다. 하지만 부스팅은 **함수(모델의 예측값 \hat{y}) 자체**를 수정하여 정답에 다가갑니다.

우리가 최소화하고 싶은 손실 함수가 평균 제곱 오차(MSE)라고 가정해 봅시다.

$$L(y, \hat{y}) = \frac{1}{2}(y - \hat{y})^2$$

이 손실 함수를 예측값 \hat{y} 에 대해 미분해 봅니다(경사를 구합니다).

$$\frac{\partial L}{\partial \hat{y}} = -(y - \hat{y})$$

어라? 결과가 익숙합니다.

- $y - \hat{y}$ 는 바로 **잔차(Residual)**입니다.
- 즉, **-(잔차)**가 곧 **기울기(Gradient)**입니다.
- 경사 하강법은 기울기의 '반대 방향'으로 가는 것이므로, **-(-잔차) = 잔차 방향**으로 이동하면 됩니다.

▣ 핵심 요약

핵심 결론 "잔차를 더해주는 것"은 수학적으로 "손실 함수의 기울기(Gradient) 반대 방향으로 이동하여 에러를 줄이는 것"과 완벽하게 동일합니다. 따라서 이 방법을 그레디언트 부스팅이라고 부릅니다.

6 실전 가이드: 모델 튜닝하기

그레디언트 부스팅을 실제로 사용할 때 가장 중요한 두 가지 설정(Hyperparameter)이 있습니다.

6.1 1. 학습률 (Learning Rate, λ)

- **정의:** 새로 추가되는 트리의 의견을 얼마나 반영할지 결정하는 비율(보통 0.01 ~ 0.1 사이).
- ** λ 가 너무 클 때:** 학습이 빠르지만, 최적점을 지나치거나(Overshooting) 과적합될 위험이 큽니다. (성큼성큼 걸다가 구덩이에 빠짐)
- ** λ 가 너무 작을 때:** 학습이 매우 느리고 많은 트리가 필요하지만, 일반화 성능이 좋습니다. (아주 조심스럽게 발을 내딛음)

6.2 2. 반복 횟수 (Number of Iterations)

- 트리를 몇 개나 더 할 것인가를 결정합니다.
- 너무 많으면 훈련 데이터에 과도하게 맞춰져(Overfitting) 테스트 성능이 떨어질 수 있습니다.
- **조기 종료(Early Stopping):** 검증 데이터(Validation Set)의 성능이 더 이상 좋아지지 않으면 학습을 멈추는 기법을 주로 사용합니다.

6.3 튜닝 전략

보통 **학습률을 낮게($\lambda \downarrow$)** 설정하고, **트리 개수를 늘리는($N \uparrow$)** 방식이 성능이 가장 좋습니다. 다만 계산 시간이 오래 걸린다는 단점이 있습니다.

7 자주 묻는 질문 (FAQ) 및 오해 바로잡기

Q1. 랜덤 포레스트와 부스팅 중 뭐가 더 좋나요?

A. 정답은 ”데이터에 따라 다르다“입니다.

- 랜덤 포레스트:** 병렬 학습이 가능해 빠르고, 튜닝 없이도 무난하게 좋은 성능을냅니다. (Overfitting에 강함)
 - 그레디언트 부스팅:** 순차 학습이라 느리고 튜닝이 까다롭지만, 잘 튜닝하면 일반적으로 랜덤 포레스트보다 더 높은 정확도를냅니다. (Kaggle 대회 우승 알고리즘의 주역)
- 테이블 형태(Tabular) 데이터에서는 두 모델 모두 딥러닝보다 더 좋은 성능을 낼 때가 많습니다.

Q2. 'Inference'란 무슨 뜻인가요?

A. 문맥에 따라 다릅니다.

- 통계학:** 데이터로부터 모집단의 특성을 추론하고 가설을 검정하는 것(예: p-value 구하기).
- 머신러닝/공학:** 학습된 모델에 새로운 데이터를 넣어 예측(Prediction) 값을 뽑아내는 과정. 부스팅 강의나 자료에서 ”Inference가 느리다“고 하면 ”예측값을 계산하는 데 시간이 걸린다“는 뜻입니다.

8 학습 마무리 체크리스트

이 문서를 다 읽은 후, 아래 질문에 스스로 답할 수 있다면 완벽하게 이해한 것입니다.

- **개념:** 부스팅이 배깅(랜덤 포레스트)과 근본적으로 어떻게 다른지 설명할 수 있는가? (힌트: 병렬 vs 순차, 분산 감소 vs 편향 감소)
- **직관:** ”약한 학습기”와 ”강한 학습기”의 관계를 비유를 들어 설명할 수 있는가?
- **알고리즘:** 그레디언트 부스팅이 다음 트리를 만들 때 사용하는 ’타겟 값’이 무엇인가? (힌트: 잔차)
- **수학:** 잔차(Residual)가 왜 손실 함수의 기울기(Gradient)와 관련이 있는지 설명할 수 있는가?
- **실전:** 학습률(λ)이 높을 때와 낮을 때의 장단점을 알고 있는가?

”여러 개의 명청한 모델이 힘을 합치면, 하나의 천재 모델보다 똑똑할 수 있다.”

– 부스팅의 철학

- 강의명: CS109A: 데이터 과학 입문
- 주차: Lecture 24
- 교수명: Pavlos Protopapas, Kevin Rader, Chris Gumb
- 목적: Lecture 24의 핵심 개념 학습

Contents

1 개요	2
2 필수 용어 정리	2
3 핵심 개념과 원리	3
3.1 1) 기본 아이디어: 오답 노트 만들기	3
3.2 2) 약한 학습기(Weak Learner)로서의 스텁프(Stump)	3
3.3 3) 레이블의 변경: 0과 1 대신 -1과 +1	3
4 AdaBoost의 동작 절차(Step-by-Step)	3
5 직관적 예제: 오렌지색 원과 파란색 삼각형	5
6 비교: AdaBoost vs Gradient Boosting	5
7 주의사항: 과적합(Overfitting)	6
7.1 1) Boosting은 과적합될 수 있다	6
7.2 2) 주요 하이퍼파라미터 (Hyperparameters)	6
8 실전 가이드 및 자주 묻는 질문(FAQ)	6
8.1 1) 언제 어떤 라이브러리를 써야 하나요?	6
8.2 2) FAQ: 초심자가 자주 하는 오해	6
9 학습 체크리스트 (시험 및 프로젝트 대비)	8
10 한 페이지 요약 (Cheat Sheet)	9

AdaBoost (Adaptive Boosting)

약한 모델을 모아 강한 모델을 만드는 양상을 기법

1 개요

▣ 핵심 요약

핵심 요약 AdaBoost는 '실수로부터 배운다'는 철학을 가진 머신러닝 알고리즘입니다. 아주 간단한 모델(Weak Learner)을 여러 번 순차적으로 학습시키되, 이전 단계에서 틀린 문제(오분류 데이터)에 더 큰 가중치를 부여하여 다음 모델이 그 문제를 집중적으로 공부하게 만듭니다. 최종적으로 이들을 합쳐 강력한 예측 성능을 냅니다.

학습 목표:

- AdaBoost가 샘플의 가중치(Weight)를 업데이트하는 원리를 이해합니다.
- 왜 AdaBoost가 지수 손실(Exponential Loss) 함수를 최소화하는지 파악합니다.
- Gradient Boosting과 AdaBoost의 수학적, 절차적 차이를 구분합니다.
- 실제 데이터 분석 프로젝트(과제 등)에서 과적합(Overfitting)을 피하는 방법을 익힙니다.

2 필수 용어 정리

처음 접하는 분들을 위해 핵심 용어를 정리했습니다.

용어	원어	쉬운 설명	비고
약한 학습기	Weak Learner	무작위 추측보다 조금 더 나은 수준의 아주 단순한 모델.	주로 Stump 사용
스텀프	Stump	뿌리만 있는 나무. 질문(분기)이 딱 하나인 결정 트리.	Node 1개, Leaf 2개
양상블	Ensemble	여러 모델의 의견을 종합하여 결론을 내리는 기법.	집단 지성
가중치	Weight (w)	해당 데이터가 얼마나 중요한지를 나타내는 점수.	틀리면 높아짐
학습률	Learning Rate (λ)	한 모델(스텀프)의 발언권 세기.	신뢰도에 비례
지수 손실	Exponential Loss	정답과 예측이 다를 때 폐널티를 급격히(지수적으로) 주는 함수. $e^{-y\hat{y}}$	

Table 1: AdaBoost 핵심 용어 비교표

3 핵심 개념과 원리

3.1 1) 기본 아이디어: 오답 노트 만들기

AdaBoost의 작동 방식은 **수험생이 오답 노트를 공부하는 과정**과 매우 비슷합니다.

- 1단계: 모의고사를 봅니다. 쉬운 문제는 맞히고 어려운 문제는 틀립니다.
- 2단계: 틀린 문제(오분류 데이터)를 별표(*) 쳐서 강조합니다. (가중치 증가)
- 3단계: 다음 공부 때는 별표 친 문제 위주로 공부합니다.
- 4단계: 이 과정을 반복한 뒤, 여러 번의 시험 결과를 종합해 최종 실력을 냅니다.

3.2 2) 약한 학습기(Weak Learner)로서의 스텁프(Stump)

AdaBoost는 복잡한 트리를 쓰지 않고, 아주 단순한 **스텀프(Stump)**를 사용합니다.

- 구조: 질문 하나(Node), 결과 두 갈래(Leaves). (예: " x_1 이 4.6보다 큰가?")
- 이유: 모델이 너무 복잡하면 과적합되기 쉽습니다. 아주 단순한 모델을 여러 개 쌓아 올리는 것이 더 효율적입니다.

3.3 3) 레이블의 변경: 0과 1 대신 -1과 +1

보통 분류 문제에서 클래스를 0과 1로 두지만, AdaBoost는 수학적 편의를 위해 **-1과 +1**을 사용합니다.

- 이점: 정답(y)과 예측(\hat{y})을 곱했을 때의 부호로 정답 여부를 바로 알 수 있습니다.
 - 정답($y = +1$), 예측($\hat{y} = +1$) $\rightarrow y \cdot \hat{y} = 1$ (양수, 정답)
 - 정답($y = -1$), 예측($\hat{y} = -1$) $\rightarrow y \cdot \hat{y} = 1$ (양수, 정답)
 - 틀린 경우 (하나가 +1, 하나가 -1) $\rightarrow y \cdot \hat{y} = -1$ (음수, 오답)
- 이 속성은 가중치 업데이트 수식에서 매우 중요하게 사용됩니다.

4 AdaBoost의 동작 절차 (Step-by-Step)

데이터 포인트가 N 개 있을 때, AdaBoost는 다음 과정을 반복합니다.

1. **초기화:** 모든 데이터의 가중치(w_n)를 똑같이 $\frac{1}{N}$ 로 설정합니다.
2. **반복 (Iteration):** 멈춤 조건(예: 50번 반복)이 될 때까지 다음을 수행합니다.
 - (a) **스텀프 학습:** 현재 가중치(w_n)를 고려하여, 데이터를 가장 잘 분류하는 스텁프(S)를 찾습니다. 가중치가 높은 데이터를 틀리면 폐널티가 크므로, 모델은 가중치가 높은 데이터를 맞히려고 노력합니다.
 - (b) **에러 계산 (ϵ):** 스텁프가 틀린 데이터들의 가중치 합을 구합니다.
 - (c) **스텀프의 중요도(λ) 산출:**
$$\lambda = \frac{1}{2} \ln \left(\frac{1 - \epsilon}{\epsilon} \right)$$
 - 에러(ϵ)가 작을수록(잘 맞힐수록) λ 는 커집니다. (발언권이 세짐)

- 예러가 0.5(무작위)면 λ 는 0이 됩니다. (의견 무시)

(d) 데이터 가중치 업데이트:

$$w_{new} \leftarrow w_{old} \times \exp(-\lambda \cdot y \cdot S(x))$$

- 맞쳤을 때 ($y \cdot S(x) > 0$): $e^{-\lambda}$ (1보다 작음)를 곱해 가중치 감소.
- 틀렸을 때 ($y \cdot S(x) < 0$): e^{λ} (1보다 큼)를 곱해 가중치 증가.

(e) 양상블 업데이트: 지금까지 만든 모델에 현재 스텁프를 더합니다.

$$T_{new} = T_{old} + \lambda \cdot S$$

3. 최종 예측: 모든 스텁프의 예측값에 각자의 중요도(λ)를 곱해 더한 뒤, 부호(Sign)를 봅니다. (양수면 +1, 음수면 -1)

5 직관적 예제: 오렌지색 원과 파란색 삼각형

가상의 2차원 데이터를 분류하는 상황을 상상해 봅시다.

1. **Step 1:** 첫 번째 스텁프가 $x_1 > 4.6$ 이라는 기준을 세워 데이터를 나눕니다. 몇몇 파란색 삼각형은 맞히지만, 일부 오렌지색 원을 틀립니다.
2. **Weight Update:** 틀린 오렌지색 원들의 크기(가중치)를 키웁니다. 이제 이 원들은 아주 '중요한' 데이터가 되었습니다.
3. **Step 2:** 두 번째 스텁프는 커진 오렌지색 원들을 맞히기 위해 $x_2 > 8$ 이라는 새로운 기준을 가져옵니다. 이번에는 이전에 맞았던 파란색 삼각형 몇 개를 틀릴 수도 있습니다.
4. **Weight Update:** 이번에 틀린 데이터들의 크기를 다시 키웁니다.
5. **Step 3:** 세 번째 스텁프가 또 다른 기준($x_1 > 5.7$)으로 남은 어려운 문제들을 해결하려 합니다.
6. **결과:** 이 세 개의 단순한 직선(스텀프)들이 합쳐져서, 복잡한 곡선 형태의 경계면을 만들어냅니다.

팁: 스텁프는 어떻게 가중치를 반영하나요? 결정 트리 알고리즘(Gini Index 등)을 계산할 때, 단순히 데이터 개수를 세는 것이 아니라 **가중치의 합(Weighted Sum)**을 사용하면 됩니다. 또는, 가중치가 높은 데이터를 가중치만큼 복사(Resampling)하여 데이터 셋을 늘린 뒤 학습하는 방법도 있습니다.

6 비교: AdaBoost vs Gradient Boosting

두 알고리즘 모두 '이전 모델의 실수를 바로잡는다'는 점은 같지만, 접근 방식이 다릅니다.

비교 항목	Gradient Boosting (GBM)	AdaBoost
목표 함수 (Loss)	주로 MSE (회귀), Log Loss (분류)	Exponential Loss ($e^{-y\hat{y}}$)
실수 처리 방식	잔차(Residual)를 직접 학습	오분류 데이터의 가중치(Weight)를 증가
학습률 (λ)	사용자가 지정 (하이퍼파라미터)	수식으로 자동 계산 (Optimal λ)
최적화 방식	손실 함수의 기울기(Gradient)를 따라감	Exponential Loss를 최소화하는 해를 분석적으로 찾음

Table 2: Gradient Boosting과 AdaBoost의 차이점

- **AdaBoost의 특징:** 학습률(λ)을 튜닝할 필요 없이, 매 단계마다 수학적으로 최적의 λ 를 찾아냅니다. (물론 구현체에 따라 추가적인 학습률 조정 옵션이 있을 수 있습니다.)
- **Gradient 관점:** AdaBoost도 넓은 의미에서 보면 Exponential Loss에 대한 Gradient Descent를 수행하는 것으로 해석할 수 있습니다.

7 주의사항: 과적합(Overfitting)

7.1 1) Boosting은 과적합될 수 있다

Random Forest와 같은 배깅(Bagging) 기법은 나무를 많이 심을수록 에러가 줄어들고 안정화되는 경향이 있습니다. 반면, **Boosting 계열(AdaBoost 포함)은 너무 많이 반복하면 과적합됩니다.**

- **이유:** 계속해서 틀린 문제(노이즈나 이상치일 수도 있음)에 집착하여 가중치를 높이기 때문에, 나중에는 데이터의 노이즈까지 외워버리게 됩니다.
- **해결:** 검증 데이터(Validation Set)의 에러가 다시 증가하기 시작하는 시점에서 학습을 멈추는 **조기 종료(Early Stopping)**가 필수적입니다.

7.2 2) 주요 하이퍼파라미터 (Hyperparameters)

프로젝트나 과제 진행 시 조정해야 할 값들입니다. (Scikit-Learn 기준)

- **n_estimators:** 스텁프의 개수(반복 횟수). 너무 크면 과적합.
- **learning_rate:** 각 스텁프의 영향력을 전체적으로 줄이는 비율. (기본값 1.0). 값을 줄이면 **n_estimators** 를 늘려야 합니다.
- **base_estimator:** 약한 학습기의 종류. (기본값: Depth 1짜리 Decision Tree). 깊이를 늘리면 모델이 복잡해져 과적합 위험이 커집니다.

8 실전 가이드 및 자주 묻는 질문(FAQ)

8.1 1) 언제 어떤 라이브러리를 써야 하나요?

- **Scikit-Learn (AdaBoostClassifier):** 학습용, 데이터가 작을 때, 기본 원리 이해용.
- **XGBoost / LightGBM / CatBoost:** 실무 프로젝트, 대용량 데이터, 높은 성능이 필요할 때. (속도 최적화, 결측치 처리, 정규화 기능이 포함됨)

8.2 2) FAQ: 초심자가 자주 하는 오해

Q1. 왜 하필 지수 손실(Exponential Loss)을 쓰나요?

A1. 지수 손실은 미분 가능하며, 0-1 손실(맞으면 0, 틀리면 1)의 상한선(Upper Bound) 역할을 합니다. 즉, 지수 손실을 줄이면 분류 에러도 자연스럽게 줄어듭니다. 또한, 잘못된 예측에 대해 벌점을 아주 크게 주기 때문에 학습 방향을 잡기 좋습니다.

Q2. 스텁프(Stump)는 너무 단순하지 않나요?

A2. 바로 그 단순함이 핵심입니다! 모델 하나하나는 약하지만(무작위보다 조금 나은 수준), 이것들이 수십, 수백 개 모여 서로의 단점을 보완하면 매우 복잡하고 정교한 경계면을 만들어냅니다.

Q3. 결정 트리는 파라메트릭인가요, 논파라메트릭인가요?

A3. 면접이나 시험에 나올 수 있는 질문입니다. 정답은 ”둘 다 볼 수 있다”입니다.

- **Parametric 관점:** 분기 기준(Threshold)과 같은 파라미터를 학습하므로.
- **Non-parametric 관점:** 데이터가 많아질수록 트리 구조가 계속 커지고 복잡해지므로(파라미터 수가

고정되어 있지 않음). 통계학적으로는 주로 Non-parametric으로 분류합니다.

9 학습 체크리스트 (시험 및 프로젝트 대비)

주의사항

과제/프로젝트 주의사항 XGBoost와 같은 최신 라이브러리는 과제(Homework)에서는 사용이 제한될 수 있으니(직접 구현 요구 등), 공지사항을 꼭 확인하세요. 하지만 기말 프로젝트에서는 성능을 위해 적극 권장됩니다.

- **개념:** Weak Learner가 모여 Strong Learner가 되는 과정을 문장으로 설명할 수 있는가?
- **수식:** 가중치 업데이트 식에서 $y \cdot \hat{y}$ 의 부호에 따라 가중치가 어떻게 변하는지 설명할 수 있는가? (맞으면 감소, 틀리면 증가)
- **계산:** λ 값을 구하는 공식 $\frac{1}{2} \ln\left(\frac{1-\epsilon}{\epsilon}\right)$ 을 알고 있는가?
- **비교:** Random Forest(배깅)과 AdaBoost(부스팅)의 과적합 성향 차이를 아는가?
- **구현:** Scikit-Learn을 사용하여 모델을 학습시키고, 하이퍼파라미터(`n_estimators`)를 튜닝할 수 있는가?
- **평가:** Validation Curve를 그려보고, 언제 학습을 멈춰야 할지(Early Stopping) 판단할 수 있는가?

10 한 페이지 요약 (Cheat Sheet)

AdaBoost Quick Review

1. 정의: 약한 학습기(Stump)를 순차적으로 연결하여 강한 예측기를 만드는 양상분 기법.
2. 핵심 메커니즘:
 - 이전 모델이 틀린 데이터 \rightarrow 가중치(w) 증가 \rightarrow 다음 모델이 집중 학습.
 - 잘 맞히는 모델 \rightarrow 중요도(λ) 증가 \rightarrow 최종 투표에서 큰 목소리.
3. 손실 함수: Exponential Loss ($e^{-y\hat{y}}$)를 최소화.
4. 장점: 구현이 쉽고, 과적합에 강한 편(하지만 영원히 강하진 않음), 파라미터 튜닝이 비교적 적음.
5. 단점: 노이즈 데이터나 이상치(Outlier)에 민감함 (계속 가중치를 키우려다 망가질 수 있음).
6. 코드 예시 (Python/sklearn):

```

1 from sklearn.ensemble import AdaBoostClassifier
2 from sklearn.tree import DecisionTreeClassifier
3
4 # 깊이가인 1 약한학습기 (Stump) 생성
5 stump = DecisionTreeClassifier(max_depth=1)
6
7 # 아다부스트모델생성번 (50 반복)
8 ada = AdaBoostClassifier(
9     base_estimator=stump,
10    n_estimators=50,
11    learning_rate=1.0
12 )
13
14 # 학습및예측
15 ada.fit(X_train, y_train)
16 y_pred = ada.predict(X_test)

```

- 강의명: CS109A: 데이터 과학 입문
- 주차: Lecture 25
- 교수명: Pavlos Protopapas, Kevin Rader, Chris Gumb
- 목적: Lecture 25의 핵심 개념 학습

Contents

1	개요: 왜 더 복잡한 앙상블이 필요한가?	2
2	필수 용어 정리	2
3	핵심 개념: 앙상블의 진화	3
4	Blending: 단순하고 직관적인 결합	3
4.1	기본 원리	3
4.2	수행 절차 (Step-by-Step)	3
5	Stacking: 데이터 낭비 없는 정교한 결합	4
5.1	Blending과의 차이점	4
5.2	수행 절차 (Step-by-Step)	4
5.3	Passthrough (패스스루)	4
6	Mixture of Experts (MoE)	5
6.1	개념: 협력이 아닌 분업	5
6.2	구조: 전문가와 게이트	5
6.3	주요 이슈: 전문가 붕괴 (Expert Collapse)	5
7	구현 및 활용 가이드 (Python/Scikit-Learn)	6
7.1	언제 어떤 모델을 써야 할까? (의사결정)	6
8	학습 점검 체크리스트 & FAQ	7
8.1	자주 묻는 질문 (FAQ)	7

CS109A 수업 노트: 양상블 학습의 확장

(Blending, Stacking, and Mixture of Experts)

1 개요: 왜 더 복잡한 양상블이 필요한가?

이 문서는 기존의 랜덤 포레스트나 부스팅(Bagging/Boosting)을 넘어, 서로 다른 종류의 모델들을 결합하여 성능을 극대화하는 고급 양상블 기법을 다룹니다.

▣ 핵심 요약

핵심 목표 및 요약

- 한계 극복:** 단일 모델(Decision Tree 등)이나 동일한 모델의 집합(Random Forest)만으로는 해결하기 어려운 복잡한 데이터 패턴을 학습합니다.
- 이종 결합:** 선형 회귀, KNN, SVM 등 성격이 완전히 다른 모델들을 하나로 합쳐 각 모델의 장점만 취합니다.
- 메타 학습:** 모델들의 예측값(Output)을 다시 학습 데이터로 사용하여 최종 결론을 내리는 '관리자 모델(Meta-model)'을 도입합니다.

2 필수 용어 정리

본격적인 학습 전에 아래 용어를 숙지하면 이해가 훨씬 빠릅니다.

용어 (한글/영문)	설명	비고
기반 모델 (Base Model)	1차적으로 데이터를 학습하여 예측을 수행하는 개별 모델들	전문가 팀원
메타 모델 (Meta Model)	기반 모델들의 예측값을 입력받아 최종 판단을 내리는 모델	팀장/관리자
동질성 (Homogeneous)	같은 종류의 알고리즘(예: 트리)만 모아서 양상블 하는 것	Random Forest 등
이질성 (Heterogeneous)	서로 다른 알고리즘(예: 선형회귀+KNN)을 섞어서 사용하는 것	Blending/Stacking
홀드아웃 세트 (Holdout Set)	메타 모델을 학습시키기 위해 따로 떼어둔 검증용 데이터	커닝 방지용
게이팅 (Gating Network)	데이터의 특성에 따라 '어떤 전문가 모델'을 쓸지 결정하는 망	상담원/분류기

Table 1: 양상블 심화 학습을 위한 핵심 용어

3 핵심 개념: 앙상블의 진화

우리가 이전에 배운 Bagging과 Boosting은 강력하지만, 주로 ** 같은 종류의 모델(주로 결정 트리)** 을 사용하는 한계가 있습니다. Blending과 Stacking은 ** 서로 다른 모델** 을 결합한다는 점에서 차이가 있습니다.

어벤져스 팀 구성하기

- **Bagging (Random Forest):** 100명의 의사가 모여서 진단하고 다수결로 결정합니다. (모두가 의사라는 점에서 동질적임)
- **Blending/Stacking:** 의사, 변호사, 회계사가 모입니다. 그리고 이들의 의견을 종합해서 최종 결정을 내리는 '판사(Meta Model)' 가 있습니다. (서로 다른 직업군이 모여 이질적임)

구분	구성 모델	결합 방식	특징
Bagging	동질적 (주로 트리)	병렬 학습 → 평균/투표	분산(Variance) 감소, 과적합 방지
Boosting	동질적 (약한 모델)	순차 학습 → 가중치 합	편향(Bias) 감소, 오답 집중 학습
Blending	이질적 (다양함)	데이터 분할 → 메타 모델 학습	구현이 쉬우나 데이터 낭비 발생
Stacking	이질적 (다양함)	교차 검증(CV) → 메타 모델 학습	데이터 효율 높음, 연산 비용 큼

Table 2: 다양한 앙상블 기법 비교

4 Blending: 단순하고 직관적인 결합

4.1 기본 원리

Blending은 데이터를 여러 조각으로 나누어, 일부는 기반 모델(Base Model)을 학습시키고, 나머지는 그 모델들을 평가(예측)하여 메타 모델(Meta Model)을 학습시키는 방식입니다.

4.2 수행 절차 (Step-by-Step)

1. **데이터 분할:** 전체 데이터를 Train / Validation / Holdout / Test 세트로 나눕니다.
2. **기반 모델 학습:** Train 세트로 여러 모델(선형회귀, 결정트리, KNN 등)을 학습시킵니다.
3. **예측 생성 (1차):** 학습된 기반 모델들로 Validation 세트를 예측합니다.
4. **메타 모델 학습:**
 - **입력(X):** 기반 모델들이 내놓은 예측값들 (필요시 원본 데이터도 포함 가능)
 - **정답(Y):** Validation 세트의 실제 정답
 - 위 데이터를 사용해 메타 모델(보통 간단한 선형 모델)을 학습시킵니다.
5. **최종 평가:** Test 세트로 최종 성능을 확인합니다.

주의사항

Blending의 단점: 데이터 효율성 Blending은 Validation과 Holdout 세트를 따로 떼어놓아야 하므로, 실제로 모델 학습에 사용할 수 있는 데이터의 양이 줄어듭니다. 데이터가 아주 많다면 괜찮지만, 데이터가 적다면 성능 저하가 올 수 있습니다.

5 Stacking: 데이터 낭비 없는 정교한 결합

5.1 Blending과의 차이점

Stacking은 Blending과 비슷하지만, **교차 검증(Cross-Validation)**을 사용하여 데이터를 낭비하지 않고 모든 데이터를 학습에 활용합니다. 조금 더 복잡하지만 성능은 일반적으로 더 좋습니다.

5.2 수행 절차 (Step-by-Step)

1. 데이터 분할: 전체 데이터를 Train과 Test로 나눕니다.
2. 교차 검증 (CV) 수행: Train 데이터를 K 개의 폴드(Fold)로 나눕니다 (예: 3-Fold).
3. 예측값 생성 (Out-of-Fold Prediction):
 - Fold 1, 2로 학습하고 → Fold 3을 예측합니다.
 - Fold 1, 3으로 학습하고 → Fold 2를 예측합니다.
 - Fold 2, 3으로 학습하고 → Fold 1을 예측합니다.
 - 이렇게 하면 모든 Train 데이터에 대한 '예측값'을 얻을 수 있습니다.
4. 메타 데이터 생성: 위에서 얻은 예측값들을 모아서 새로운 입력 데이터셋(X_{meta})을 만듭니다.
5. 메타 모델 학습: X_{meta} 와 실제 정답(Y)을 이용해 메타 모델을 학습시킵니다.
6. 최종 추론: Test 데이터를 입력하면, 기반 모델들이 예측값을 내놓고, 이를 메타 모델이 받아 최종 결과를 출력합니다.

Stacking 과정 비유 팀원들이 돌아가며 모의고사를 봅니다.

- A가 시험 볼 때 B, C가 공부한 걸로 채점해주고,
- B가 시험 볼 때 A, C가 도와주는 식입니다.
- 결과적으로 팀원 모두의 모의고사 성적표(예측값)가 모이게 되고, 선생님(메타 모델)은 "철수는 수학을 잘하고 영희는 영어를 잘하네"라는 패턴을 학습하여 실전 수능(Test Set)에 대비합니다.

5.3 Passthrough (패스스루)

메타 모델을 학습시킬 때, 단순히 기반 모델의 예측값(\hat{y})만 쓰는 것이 아니라, **원본 데이터(X)도 함께 넣어주는 기법**입니다.

- 장점: 메타 모델이 "어떤 데이터 상황에서 어떤 모델을 믿어야 할지" 더 잘 판단할 수 있습니다.
- 주의: 데이터 차원이 커지고 복잡해질 수 있습니다.

6 Mixture of Experts (MoE)

6.1 개념: 협력이 아닌 분업

지금까지의 양상들은 모든 모델이 힘을 합쳐(평균, 투표) 결과를 냈습니다. 하지만 MoE는 **”이 문제는 네가 전문가니까 네가 처리해”**라고 맡기는 방식입니다. 최근 대형 언어 모델(LLM)에서도 효율성을 위해 많이 사용되는 핵심 개념입니다.

6.2 구조: 전문가와 게이트

- **전문가 (Experts):** 특정 데이터 영역이나 패턴에 강한 모델들입니다. (예: $X < 0$ 일 때는 선형회귀 A, $X \geq 0$ 일 때는 선형회귀 B)
- **게이팅 네트워크 (Gating Network):** 입력 데이터(X)를 보고 어떤 전문가에게 일을 맡길지 결정하는 ‘분류기’입니다. 주로 Softmax 함수를 사용하여 각 전문가에 대한 가중치(확률)를 출력합니다.

$$\text{최종 예측 } \hat{y} = \sum_{i=1}^K g_i(x) \cdot f_i(x) \quad (1)$$

- $g_i(x)$: 게이팅 네트워크가 부여한 i 번째 전문가의 가중치 (합은 1)
- $f_i(x)$: i 번째 전문가의 예측값

병원 진료 시스템 환자(데이터)가 병원에 오면, 접수처 직원(Gating Network)이 증상을 듣고 정형외과(Expert 1)로 보낼지, 내과(Expert 2)로 보낼지 결정합니다. 모든 의사가 다 같이 진료하는 것이 아니라, 가장 적합한 의사가 주도적으로 치료합니다.

6.3 주요 이슈: 전문가 붕괴 (Expert Collapse)

학습 과정에서 특정 전문가 한 명만 너무 똑똑해지거나, 게이팅 네트워크가 한쪽으로만 데이터를 몰아주는 현상이 발생할 수 있습니다.

- **증상:** 모든 데이터가 Expert 1에게만 가고, 나머지 Expert들은 놉니다.
- **원인:** 초기에 성능이 조금이라도 좋은 쪽에 몰아주다 보니 빈익빈 부익부 현상이 발생함.
- **해결:** 학습 시 손실 함수(Loss Function)를 조정하여 전문가들이 골고루 학습되도록 유도해야 합니다.

7 구현 및 활용 가이드 (Python/Scikit-Learn)

Stacking은 ‘sklearn’ 라이브러리를 통해 쉽게 구현할 수 있습니다. 아래는 개념적인 의사 코드(Pseudo-code) 흐름입니다.

Stacking Regressor 구현 흐름

```
from sklearn.ensemble import StackingRegressor
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.neighbors import KNeighborsRegressor

# 1. 기반 모델(Experts) 정의
estimators = [
    ('rf', RandomForestRegressor(n_estimators=10)),
    ('knn', KNeighborsRegressor(n_neighbors=5)),
    ('lr', LinearRegression())
]

# 2. 메타 모델(Manager) 정의 - 보통 단순한 모델 사용
meta_model = LinearRegression()

# 3. Stacking 모델 구성 (cv=5는 5-Fold 교차검증 의미)
# passthrough=True 옵션: 원본 데이터도 메타 모델에 전달
clf = StackingRegressor(
    estimators=estimators,
    final_estimator=meta_model,
    cv=5,
    passthrough=True
)

# 4. 학습 및 예측
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
```

7.1 언제 어떤 모델을 써야 할까? (의사결정)

- 데이터가 적고 빠른 결과가 필요하다: → 단일 모델 또는 Random Forest
- 성능을 0.1%라도 더 올리고 싶다 (Kaggle 등): → Stacking (다양한 모델 결합)
- 데이터의 특성이 영역별로 확연히 다르다: → Mixture of Experts (또는 Tree 모델)
- 해석(Interpretation)이 중요하다: → Stacking을 쓰되, 메타 모델로 선형 회귀를 써서 각 기반 모델의 가중치(계수)를 확인한다.

8 학습 점검 체크리스트 & FAQ

- **기반 모델의 다양성 확보:** 서로 비슷한 모델(예: 트리 3개)만 섞지 않았는가? (KNN, 선형, 트리를 섞는 것이 좋음)
- **데이터 누수(Leakage) 주의:** Stacking 시 Test 데이터를 학습 과정에 실수로 포함시키지 않았는가?
- **메타 모델의 단순성:** 메타 모델을 너무 복잡하게(예: 딥러닝) 만들어서 과적합되지 않았는가? (보통 선형 모델 권장)
- **전처리 일관성:** KNN 같은 거리 기반 모델을 섞을 때는 스케일링(StandardScaler)을 했는가? (트리는 필요 없지만 KNN은 필수)

8.1 자주 묻는 질문 (FAQ)

Q1. Stacking이 Random Forest보다 무조건 좋은가요?

A. 아닙니다. Stacking은 연산량이 훨씬 많고 복잡합니다. 데이터에 따라 Random Forest 같은 단일 양상을 기법이 더 효율적일 수 있습니다. Stacking은 '마지막 성능 쥐어짜기' 용도로 주로 쓰입니다.

Q2. Passthrough는 언제 써야 하나요?

A. 기반 모델들이 데이터의 모든 정보를 다 캡처하지 못했다고 판단될 때 씁니다. 하지만 원본 데이터의 차원(Feature 수)이 너무 많으면 메타 모델이 과적합될 수 있으니 주의해야 합니다.

Q3. Mixture of Experts에서 '전문가'는 사람이 정해주나요?

A. 아닙니다. 데이터 학습 과정(Gradient Descent)을 통해 모델이 스스로 "이 데이터 영역은 내가 처리 할게"라고 학습하게 됩니다. 하지만 초기 설정이나 손실 함수 설계가 잘못되면 한 명의 전문가만 일하는 콜림 현상이 발생할 수 있습니다.

학습 팁: 처음에는 Random Forest나 Gradient Boosting 같은 검증된 단일 양상을 기법을 먼저 마스터하세요. 그 후 성능의 한계를 느낄 때, 서로 다른 모델들을 섞는 Stacking을 시도해보는 것이 가장 효율적인 학습 순서입니다.