Stanford CS229 Machine Learning I Supervised learning setup, LMS I 2022 I Lecture 2

1. 강의 개요 및 목표

- **강의 구조**:
  - 머신러닝의 기본 모델을 이해하기 위한 세 가지 주요 강의로 구성됨:
    1. 선형 회귀(linear regression): 데이터를 선형 모델로 학습.
    2. 분류(classification): 레이블이 이산형(discrete)인 경우를 다룸.
    3. 일반화된 지수 가족 모델(exponential family models): 다양한 모델을 통합적으로 다루는 접근법.
  - 이번 강의에서는 선형 회귀를 중심으로 모델 구성과 학습 방법을 소개하며, 다음 강의로 분류와 고급 모델로 확장함.

목표

- 학습 데이터 외 새로운 데이터를 예측(generalization)할 수 있는 모델 구축.
- 모델의 매개변수를 추정하고 최적화하는 방법 이해.
- 머신러닝의 실제 응용과 현대적 알고리즘의 기초를 학습.

---

2. 머신러닝의 기본 정의와 목표

2.1 머신러닝의 목표

- 데이터(X)와 레이블(Y)의 관계를 학습하여 새로운 데이터에 대한 예측 모델 구축

예측 함수: X → Y

image cat

text is hate speech?

house data price

- 지도 학습(Supervised Learning) 중심으로 설명.

2.2 회귀(Regression)와 분류(Classification)의 차이점

- 회귀: 레이블이 연속형 값을 예측하는 문제. (집 가격을 예측하는 문제)
- 분류: 이산형 레이블(예: 고양이/개)을 예측하는 문제.

2.3 학습 데이터와 일반화

- 학습 데이터는 X와 Y의 쌍으로 구성된 훈련 데이터로 제공됨.
- 일반화란 훈련 데이터에 없는 새로운 데이터에 대해 잘 작동하도록 모델을 만드는 것.

---

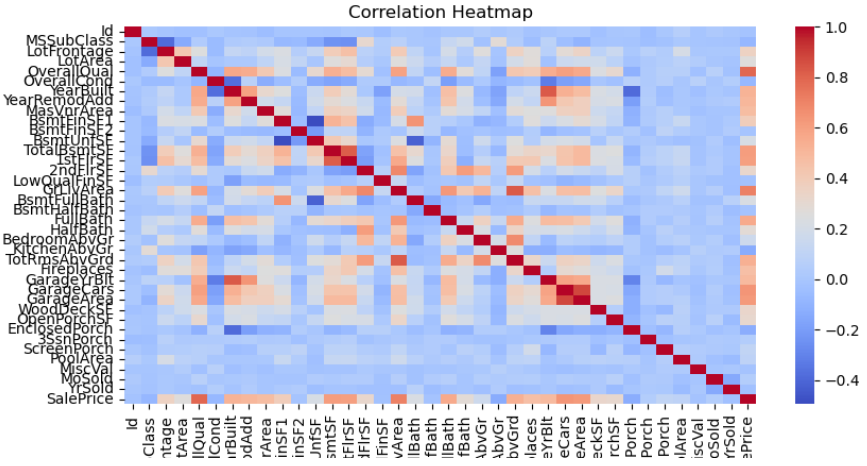
3. 선형 회귀(linear regression)의 기본 개념

선형 회귀란?
: 입력 데이터
X와 레이블 Y 간의 관계를 학습하여 새로운 X에 대해 Y를 예측하는 모델. = 데이터의 특징(feature)을 기반으로 직선을 맞추는 모델.

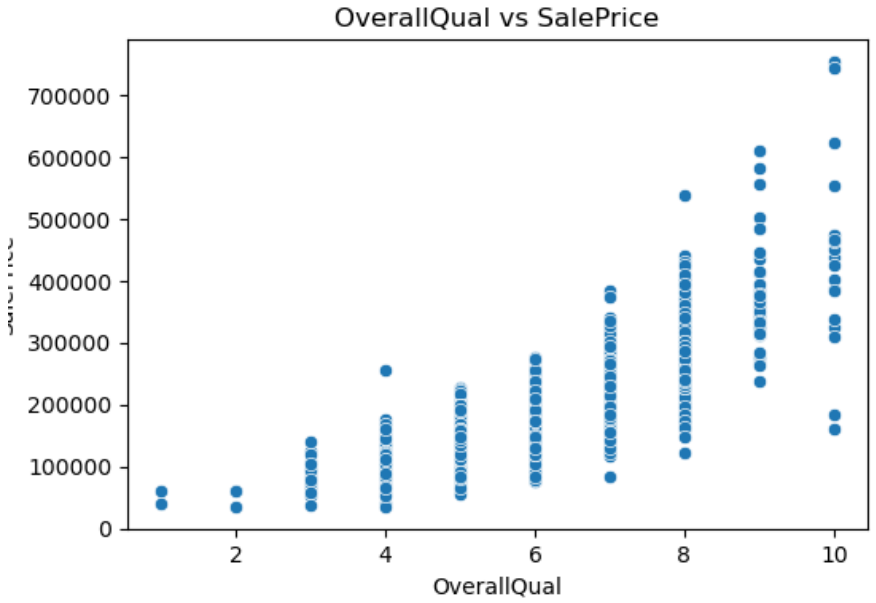
데이터는 벡터 공간에서 다루어지며, 각 레이블은 특정 집합에 속합니다. 이러한 데이터는 보통 일련의 특성(feature)들로 표현

예시:

- 집 크기(입력)와 가격(레이블) 사이의 관계 학습.
- 목표는 새로운 집 크기에 대해 가격을 예측.



상관관계 히트맵 : 색이 진할 수록 상관관계 강함



`OverallQual vs SalePrice`: `OverallQual`이 높을수록 `SalePrice`가 높아지는 경향을 보여줌.

▼ 코드

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import os

# 1. 데이터 로드
# 파일 경로 지정
dir_path = r"D:\time_series\house-prices-advanced-regression-techniques"
file_name = "train.csv"  # 파일명이 다를 경우 수정 필요
file_path = os.path.join(dir_path, file_name)

# CSV 파일 로드
data = pd.read_csv(file_path)

# 데이터 확인
print("데이터 샘플:")
print(data.head())
print("\n데이터 정보:")
print(data.info())

# 2. 데이터 상관관계 분석
# SalePrice와 상관관계 높은 변수 확인
correlation = data.corr()
top_correlated_features = correlation['SalePrice'].sort_values(ascending=F
print("\nSalePrice와 상관관계 높은 변수:")
print(top_correlated_features)
```

```
# 상관관계 히트맵
plt.figure(figsize=(12, 10))
sns.heatmap(correlation, cmap='coolwarm', annot=False)
plt.title("Correlation Heatmap")
plt.show()

# 3. 주요 변수와 SalePrice의 관계 시각화
# 상관관계 상위 변수 중 SalePrice 제외
key_features = top_correlated_features.index.tolist()[1:]  # SalePrice 제외

# 산점도 시각화
for feature in key_features:
    plt.figure(figsize=(6, 4))
    sns.scatterplot(x=data[feature], y=data['SalePrice'])
    plt.title(f"{feature} vs SalePrice")
    plt.xlabel(feature)
    plt.ylabel("SalePrice")
    plt.show()

# 4. SalePrice 분포 시각화
plt.figure(figsize=(8, 6))
sns.histplot(data['SalePrice'], kde=True, bins=30)
plt.title("Distribution of SalePrice")
plt.xlabel("SalePrice")
plt.ylabel("Frequency")
plt.show()

# 5. 주요 변수 간 Pairplot
# SalePrice 포함한 상위 변수 시각화
sns.pairplot(data[top_correlated_features.index.tolist()])
plt.show()
```

모델 수식

- $h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_d x_d$
- θ: 매개변수(parameter).
- x: 입력 특징(feature).

3.3 고차원 문제

- 고차원 데이터의 특징을 벡터 공간에 맵핑하는 과정 설명.
- 다양한 특징(feature)을 조합하여 복잡한 문제 해결 가능.

손실 함수(loss function)

큰 오차에 대해서 더 큰 패널티를 부여. 손실 함수는 모델의 예측 오류를 측정하며, 이를 최소화하는 매개변수를 찾는 것이 최종 목표

- 잔차(residual) 제곱합(Sum of Squares)을 최소화하여 최적의 매개변수를 찾음.
- 선형회귀에서는 평균 제곱 오차(MSE)를 주요 손실 함수
- 손실 함수 수식:
  - $J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$
  - m: 데이터 샘플의 수.
- 손실을 최소화하면 예측이 실제 값에 가까워짐.

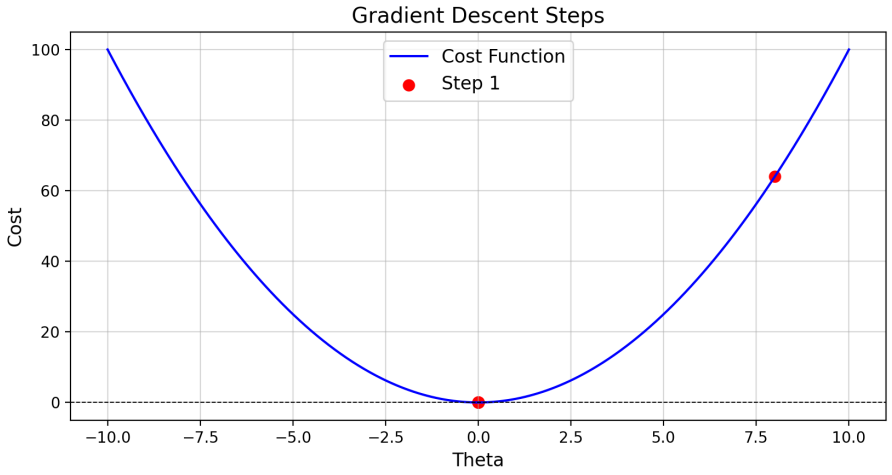
- 선형 회귀 문제를 해결하기 위한 주요 도구로 정규 방정식(normal equation)과 배치 경사 하강법(gradient descent)을 소개
  - **손실 함수(loss function)**: 잔차(residual)의 제곱합(Sum of Squares)을 최소화하여 오류를 줄임.
  - **정규 방정식(normal equation)**: 선형 회귀 문제를 수학적으로 해결하는 기본적인 방법으로 간단하지만, 데이터셋이 클 경우 계산 비용이 높음.

4.
최적화 기법: 경사 하강법(Gradient Descent)

- **경사 하강법이란?**

  손실 함수의 기울기를 계산하여 매개변수를 점진적으로 조정, 최적의 매개변수(θ)를 찾는 알고리즘.

  = 큰 데이터셋에 대해 효율적으로 손실 함수를 최소화하는 방법.
  - 경사 하강법은 머신러닝의 핵심 최적화 기법으로 제시됨.
  - 60년대에는 증분적 경사하강법으로 불림
  - 모델의 학습 초기에는 작은 배치를 사용하여 효율적인 학습을 도모하며, 학습 속도를 개선하기 위한 여러 최적화 전략이 언급됨.



▼ 코드

```
import numpy as np
import matplotlib.pyplot as plt

# 데이터 생성
def cost_function(theta):
    return theta**2  # 간단한 비용 함수

def gradient(theta):
    return 2 * theta  # 비용 함수의 기울기

# Gradient Descent 설정
theta_values = np.linspace(-10, 10, 500)
cost_values = cost_function(theta_values)
theta = 8.0  # 초기값
learning_rate = 0.5
iterations = 10

# 그래프 그리기
plt.figure(figsize=(10, 6))
plt.plot(theta_values, cost_values, label="Cost Function", color="blue")

# Gradient Descent 반복
for _ in range(iterations):
    plt.scatter(theta, cost_function(theta), color="red", s=50, label=f"St
    theta -= learning_rate * gradient(theta)

# 그래프 꾸미기
plt.title("Gradient Descent Steps", fontsize=14)
plt.xlabel("Theta", fontsize=12)
plt.ylabel("Cost", fontsize=12)
plt.axhline(0, color="black", linestyle="--", linewidth=0.7)
plt.legend(fontsize=12)
```

```
plt.grid(alpha=0.5)

# 그래프 출력
plt.show()
```

과정

1. 초기값 θ_0 설정 (무작위, 0 등).
2. 손실 함수의 기울기($\nabla J(\theta)$)를 계산.
3. 학습률(learning rate, α)에 따라 매개변수를 업데이트:
   - $\theta_j := \theta_j - \alpha \frac{\partial J(\theta)}{\partial \theta_j}$
4. 손실 함수가 더 이상 줄어들지 않으면 종료.

학습률(learning rate)

- 학습률이 너무 크면 발산하고, 너무 작으면 학습 속도가 느려짐.
- 적절한 값을 선택하거나, 적응형 학습률(adaptive optimizers)을 사용함.

배치 크기 선택:

- 전체 데이터를 사용하는 배치(batch)는 정확도가 높지만 느림. →경사하강법
- 일부 데이터를 사용하는 미니배치(mini-batch)는 속도와 정확도의 균형을 맞춤. → 스톡캐스틱 경사 하강법 (Stochastic Gradient Descent)

- 작은 배치는 빠르지만 정확도 낮음.
- 큰 배치는 느리지만 정확도가 높음.
- 일반적으로 미니배치(mini-batch)를 사용해 속도와 정확도 균형을 맞춤.

---

5. 분류(Classification)와 일반화(Generalization)

- **분류로 확장**:
  - 선형 회귀에서 분류로 전환하면서 지수 가족 모델(exponential family models)을 소개.
  - 분류는 특정 레이블(예: 고양이, 개)로 데이터를 구분하며, 모델의 출력은 이산형 값(discrete values)을 가짐.
- **일반화**:
  - 학습 데이터셋 외의 새로운 데이터에 대해 정확히 예측하는 것이 목표.
  - 모델이 과적합(overfitting)되지 않도록 일반화 기법을 활용.
  - 학습의 목표는 훈련 데이터 외의 새로운 데이터에 대해 일반화된 예측을 수행하는 것임.

    - 이를 위해 학습 데이터셋의 특징을 추출하고, 적절한 손실 함수(loss function)를 설정하여 최적의 매개변수를 찾음.
- **분류로 확장**:
    - 선형 회귀에서 분류로 전환하면서 지수 가족 모델(exponential family models)을 소개.
    - 분류는 특정 레이블(예: 고양이, 개)로 데이터를 구분하며, 모델의 출력은 이산형 값(discrete values)을 가짐.
- **일반화**:
    - 학습 데이터셋 외의 새로운 데이터에 대해 정확히 예측하는 것이 목표.
    - 모델이 과적합(overfitting)되지 않도록 일반화 기법을 활용.

---

6. 현대 머신러닝의 응용 및 실제 데이터 활용:

- 머신러닝 모델이 다양한 데이터 유형(예: 이미지, 텍스트)에 어떻게 적용되는지 설명.
- 현대적인 응용 사례:
    - 이미지 분류, 자연어 처리, 번역 시스템 등.
    - 알고리즘의 기본 구조(예: 경사 하강법)는 이러한 응용에서도 핵심적인 역할을 함.
- 데이터 전처리와 시각화의 중요성 강조:
    - 데이터셋을 시각화하여 패턴과 트렌드를 파악.
    - 예시: 집 크기(입력)와 가격(레이블)의 관계를 그래프로 표현하여 선형적 경향성을 확인.
    - 데이터의 특징(feature)을 추출하고 고차원 공간에서 모델링하는 기법을 소개.

활

5.1 지수족 모델(Exponential Family Models)

- 분류, 회귀 등 다양한 모델을 하나의 일반화된 프레임워크로 통합.
- 복잡한 데이터에도 일관된 방식으로 매개변수를 추정하고 예측 가능.

5.2 현대적 응용

- 이미지 처리, 자연어 처리, 번역 등 다양한 응용 분야에서 SGD와 같은 알고리즘 사용.
- 복잡한 모델도 선형 회귀의 확장으로 이해 가능.

---

7. 미래 강의 계획:

다음 강의 내용

- 분류(classification)와 지수 가족 모델에 대한 심층 분석.
- 다양한 산업적 응용 사례 소개.

- 앞으로의 강의에서는 신경망(neural networks), 커널 기법(kernels), 비지도 학습(unsupervised learning), 그래프 모델(graphical models) 등을 다룰 예정.
- 모든 기법은 동일한 추상적 구조를 통해 이해할 수 있도록 구성됨.