

비즈니스를 위한 데이터마이닝

가톨릭대학교 경영학과

이홍주

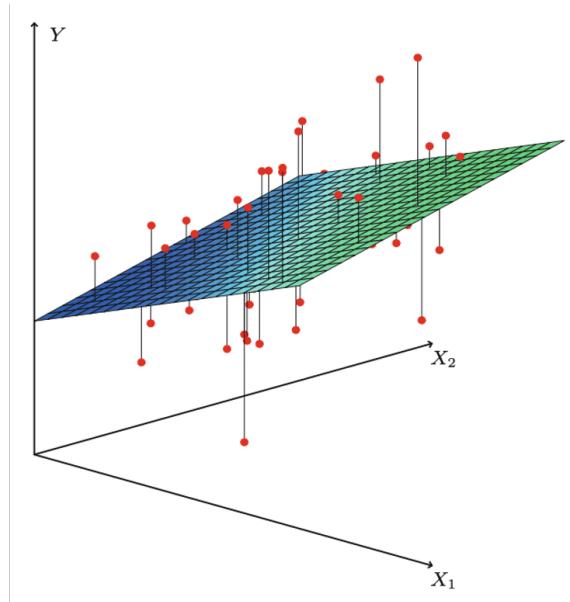
회귀분석 Regression Analysis

- 지도 학습 방안 중 하나이며, 수치를 예측하는 것에 활용됨
- 회귀분석은 관찰된 연속형 변수들인 독립변수(Independent variable)와 종속변수(Dependent variable) 사이의 상관관계에 따른 수학적 모델인 선형 관계식을 구하여 어떤 독립변수가 주어졌을 때 이에 따른 종속변수의 값을 예측함

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p + \epsilon,$$

outcome
constant
coefficients
predictors
error (noise)

회귀분석 Regression Analysis



(출처: An Introduction to Statistical Learning with applications in R)

회귀분석: 설명 vs 예측

- 설명 모델
 - 독립변수와 종속변수 간의 관계 설명
 - 사회과학에서 많이 활용되는 가설 검정 방안
 - 데이터를 모형에 적합시켜서 독립변수의 모형에 대한 기여를 이해
 - R^2 , p values 의 지표 활용

회귀분석: 설명 vs 예측

- 예측 모델
 - 새로운 사례를 정확하게 예측하는 모델
 - 예측 정확도를 최적화하는 것에 목적을 둠
 - 학습 데이터를 통해 모형 생성하며, 테스트 데이터로 성과측정
 - 독립변수의 영향력을 설명하는 것이 주요 목적은 아니나 유용함

회귀분석

- 입력 값이 x_1, x_2, \dots, x_p 인 레코드의 출력 변수 값 예측

$$\hat{Y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + \dots + \hat{\beta}_p x_p \quad (6.2)$$

- 노이즈 ϵ (또는 Y)는 표준 정규 분포를 따른다.
- 선택된 독립 변수가 적절하고 선형성을 따른다.
- 각 변수들이 서로 독립적이다.

예제: 도요타 코롤라 가격 예측

- 중고차 속성에 기반한 도요타 코롤라 중고차 가격 예측
- 중고 도요타 코롤라 1000대의 정보

표 6-1 도요타 코롤라 예제의 변수 설명

Age_08_04	Price	판매 가격(단위: 유로)
	Age	2004년 8월 당시 자동차의 내용 연수 (단위: 개월)
Fuel_Type	Kilometer	연비 누적 평균 주행 거리
	Fuel Type	연료 유형(휘발유, 경유, 천연가스)
Quarterly_Tax	HP	마력 ^{horse power}
	Metallic	금속 색상(yes = 1, no = 0)
	Automatic	자동 변속(yes = 1, no = 0)
	CC	실린더 부피
	Doors	자동차 문의 개수
	QuartTax	분기별 도로사용세(단위: 유로)
	Weight	무게(단위: 킬로그램)

예제: 도요타 코롤라 가격 예측

- 데이터 샘플

Price	Age	KM	Fuel_Type	HP	Metallic	Automatic	cc	Doors	Quarterly_Tax	Weight
13500	23	46986	Diesel	90	1	0	2000	3	210	1165
13750	23	72937	Diesel	90	1	0	2000	3	210	1165
13950	24	41711	Diesel	90	1	0	2000	3	210	1165
14950	26	48000	Diesel	90	0	0	2000	3	210	1165
13750	30	38500	Diesel	90	0	0	2000	3	210	1170
12950	32	61000	Diesel	90	0	0	2000	3	210	1170
16900	27	94612	Diesel	90	1	0	2000	3	210	1245
18600	30	75889	Diesel	90	1	0	2000	3	210	1245
21500	27	19700	Petrol	192	0	0	1800	3	100	1185
12950	23	71138	Diesel	69	0	0	1900	3	185	1105
20950	25	31461	Petrol	192	0	0	1800	3	100	1185

예제: 도요타 코롤라 가격 예측

- 전처리
 - 회귀분석은 수치형 변수가 독립변수로 활용 가능
 - Fuel_Type은 Diesel, Petrol, CNG 범주형 변수
 - 더미 변수로 변환. 회귀분석은 모든 범주를 더미 변수로 표현하면 다중공
선성 오류 발생 (독립변수들이 높은 상관관계를 가질 때 발생)
only
 - get_dummies(,drop_first=True) 로 첫번째 범주는 더미변수에서 제
외하면 됨
 - 다른 기계학습 모형은 모든 범주의 더미변수 입력 가능

예제: 도요타 코롤라 가격 예측

표 6-3 코롤라 중고차 속성을 이용한 가격 선형 회귀 모델

가격 vs. 선형 회귀 모델 자동차 속성

```
# reduce data frame to the top 1000 rows and select columns for regression analysis
car_df = pd.read_csv('ToyotaCorolla.csv')
car_df = car_df.iloc[0:1000]

predictors = ['Age_08_04', 'KM', 'Fuel_Type', 'HP', 'Met_Color', 'Automatic', 'CC',
              'Doors', 'Quarterly_Tax', 'Weight']
outcome = 'Price'

# partition data
X = pd.get_dummies(car_df[predictors], drop_first=True)
y = car_df[outcome]
train_X, valid_X, train_y, valid_y = train_test_split(X, y, test_size=0.4, random_state=1)

car_lm = LinearRegression() .. class를 하려면 개선해야 한다.
car_lm.fit(train_X, train_y)

# print coefficients
print(pd.DataFrame({'Predictor': X.columns, 'coefficient': car_lm.coef_}))

# print performance measures (training data)
regressionSummary(train_y, car_lm.predict(train_X))
```

실행 결과

Predictor	coefficient
Age_08_04	-140.748761
KM	-0.017840
HP	36.103419
Met_Color	84.281830
Automatic	416.781954
CC	0.017737
Doors	-50.657863
Quarterly_Tax	13.625325
Weight	13.038711
Fuel_Type_Diesel	1066.464681
Fuel_Type_Petrol	2310.249543

Regression statistics

Mean Error (ME) : 0.0000
Root Mean Squared Error (RMSE) : 1400.5823
Mean Absolute Error (MAE) : 1046.9072
Mean Percentage Error (MPE) : -1.0223
Mean Absolute Percentage Error (MAPE) : 9.2994

예제: 도요타 코롤라 가격 예측

표 6-4 검증 데이터셋 20개의 예측값 및 검증 데이터셋에 대한 성능 요약

회귀 모델을 학습 데이터셋에 맞추고 검증에서 가격을 예측하기 위한 파이썬 코드

Use predict() to make predictions on a new set

car_lm_pred = car_lm.predict(valid_X) ... 예측도 코드 한 줄이면 됨
... 목표변수는 딸 'X'

```
result = pd.DataFrame({'Predicted': car_lm_pred, 'Actual': valid_y,  
                      'Residual': valid_y - car_lm_pred})  
  
print(result.head(20))  
  
# print performance measures (validation data)  
regressionSummary(valid_y, car_lm_pred)
```

	Predicted	Actual	Residual
507	10607.333940	11500	892.666060
818	9272.705792	8950	-322.705792
452	10617.947808	11450	832.052192
368	13600.396275	11450	-2150.396275
242	12396.694660	11950	-446.694660
929	9496.498212	9995	498.501788
262	12480.063217	13500	1019.936783
810	8834.146068	7950	-884.146068
318	12183.361282	9900	-2283.361282
49	19206.965683	21950	2743.034317
446	10987.498309	11950	962.501691
142	18501.527375	19950	1448.472625
968	9914.690947	9950	35.309053
345	13827.299932	14950	1122.700068
971	7966.732543	10495	2528.267457
133	17185.242041	15950	-1235.242041
104	19952.658062	19450	-502.658062
6	16570.609280	16900	329.390720
600	13739.409113	11250	-2489.409113
496	11267.513740	11750	482.486260

Regression statistics

Mean Error (ME) : 103.6803
Root Mean Squared Error (RMSE) : 1312.8523
Mean Absolute Error (MAE) : 1017.5972
Mean Percentage Error (MPE) : -0.2633
Mean Absolute Percentage Error (MAPE) : 9.0111

회귀 분석의 변수 선택

모든 변수에는
관계가 있음!

- 모델에 변수 활용 시 주의 사항
 - 예측 변수들을 전부 수집하는 것이 실행 가능하지 않거나 비용이 너무 비쌀 수 있다.
 - 예측 변수가 많을수록 데이터에 결측치가 존재할 위험성이 높아진다.
 - 변수가 많은 모델에서는 다중 공선성으로 인해 회귀 계수의 추정치들이 불안정할 수 있다. 회귀 계수는 간결한 모델에서 더욱 안정적이다.
 - 종속 변수와 상관관계가 없는 예측 변수를 사용하면 예측의 분산이 증가할 수 있다.
 - 종속 변수와 실제 상관관계가 있는 예측 변수를 누락시키면 예측의 평균 오차 혹은 바이어스가 증가할 수 있다.

회귀 분석의 변수 선택

- 충분히 잘 작동하는 가장 단순한 모형을 찾는 것이 목적

방법 ① 도메인 지식 활용을 통해 변수 선택

② 전역 탐색과 부분 집합 선택 알고리즘 활용

- 전역 탐색: 독립 변수의 모든 조합으로 모형을 생성하여 성과 측정 (수정 결정 계수, adjusted R²). 가장 성과가 좋은 변수 부분 집합 선택.

- 기본 R square는
독립변수를 더해도 줄어들지 않음!
→ 전역탐색 시 양수↑

- 독립변수 n'이면 성과가
줄어들지만 수정결정계수는 증가함

$$AIC = n \ln(SSE/n) + n(1 + \ln(2\pi)) + 2(p+1)$$

(6.3)

내가 만든
모형의 품질을
알고주는
(상대적임)
더 작은게
是最好的
제일 좋은거!
(범위도 안수X)

$$BIC = n \ln(SSE/n) + n(1 + \ln(2\pi)) + \ln(n)(p+1)$$

(6.4)

전역 탐색

표 6-5 도요타 코롤라 예제의 예측 변수 감소를 위한 전역 탐색 결과

전체 검색을 위한 코드: 부록에서 소스코드를 제공하는 유ти리티 함수 exhaustive_search()를 사용한다. 이 함수는 모든 기능의 목록, 모델을 생성하는 함수, 주어진 기능과 모델에 점수를 매기는 함수로 인수가 3개다.

```
def train_model(variables):
    model = LinearRegression()
    model.fit(train_X[list(variables)], train_y)
    return model

def score_model(model, variables):
    pred_y = model.predict(train_X[list(variables)])
    # we negate as score is optimized to be as low as possible
    return -adjusted_r2_score(train_y, pred_y, model)
```

```
allVariables = train_X.columns
results = exhaustive_search(allVariables, train_model, score_model)
모든변수들이

data = []
for result in results:
    model = result['model']
    variables = list(result['variables'])
    AIC = AIC_score(train_y, model.predict(train_X[variables]), model)

    d = {'n': result['n'], 'r2adj': -result['score'], 'AIC': AIC}
    d.update({var: var in result['variables'] for var in allVariables})
    data.append(d)
pd.DataFrame(data, columns=('n', 'r2adj', 'AIC') + tuple(sorted(allVariables)))
```

수정된 점수는 1에 가까운 수로 정렬

전역 탐색이란?
모든 가능한 것들 도메인에서 최적화 찾기

n	r2adj	AIC	Age_00_04	Automatic	CC	Doors	Fuel_Type_Diesel	Weight
0	1	0.767901	10689.712094	True	False	False	False	False
1	2	0.801160	10597.910645	True	False	False	False	False
2	3	0.829659	10506.084235	True	False	False	False	False
3	4	0.846357	10445.174820	True	False	False	False	False
4	5	0.849044	10435.578836	True	False	False	False	False
5	6	0.853172	10419.932278	True	False	False	False	False
6	7	0.853860	10418.104925	True	False	False	False	True
7	8	0.854297	10417.290103	Good	True	True	False	False
8	9	0.854172	10418.789079	True	True	True	True	True
9	10	0.854036	10420.330800	True	True	False	True	True
10	11	0.853796	10422.298278	True	True	True	True	True

	Fuel_Type_Petrol	HP	KM	Met_Color	Quarterly_Tax	Weight
0	False	False	False	False	False	False
1	False	True	False	False	False	False
2	False	True	False	False	False	True
3	False	True	True	False	False	True
4	False	True	True	False	True	True
5	True	True	True	False	True	True
6	True	True	True	False	True	True
7	True	True	True	False	True	True
8	True	True	True	False	True	True
9	True	True	True	True	True	True
10	True	True	True	True	True	True

EX01 6주차 예제 허기 분석하기

exhaustive-search 우연한 바로
증거 봐줄

부분 집합 선택 알고리즘

$X_n + X_{n+1} + \dots$

- 전방 선택, 후방 소거법, 단계적 선택 방법
- 후방 소거법 Backward Elimination
 - 처음에는 모든 예측 변수를 사용하는 데에서 시작해 단계별로 가장 유용하지 않은 예측 변수들을 (통계적 유의성에 의해) 제거해 나가는 방법
 - 제거되지 않고 남아 있는 예측 변수들의 기여도가 모두 유의하다고 판단될 때 중단
 - 모든 예측 변수를 포함하는 초기 모델을 계산하는 데 시간이 많이 소요되고 불안정

하나씩 빼보고 훈련

후방 소거법

표 6-6 도요타 코롤라 예제의 예측 변수 감소를 위한 후방 소거법 결과



후방 소거법을 실행하는 파이썬 코드

```
def train_model(variables):
    model = LinearRegression()
    model.fit(train_X[variables], train_y)
    return model

def score_model(model, variables):
    return AIC_score(train_y, model.predict(train_X[variables]), model)

allVariables = train_X.columns
best_model, best_variables = backward_elimination(allVariables, train_model,
                                                    score_model, verbose=True)

print(best_variables)

regressionSummary(valid_y, best_model.predict(valid_X[best_variables]))
```

Variables: Age_08_04, KM, HP, Met_Color, Automatic, CC, Doors, Quarterly_Tax, Weight, Fuel_Type_Diesel, Fuel_Type_Petrol
Start: score=10422.30
Step: score=10420.33, remove CC .. 짧아지는 게 좋은 것!
Step: score=10418.79, remove Met_Color
Step: score=10417.29, remove Doors
Step: score=10417.29, remove None .. 단위가 무언 봐도 좋은 것 X
['Age_08_04', 'KM', 'HP', 'Automatic', 'Quarterly_Tax', 'Weight', 'Fuel_Type_Diesel', 'Fuel_Type_Petrol']

부분 집합 선택 알고리즘

- **전방 선택 방법**

- 예측 변수가 없는 상태에서 시작
- 가장 기여가 큰 예측 변수를 하나씩 추가해 감
- 추가되는 예측 변수의 기여도가 통계적으로 유의하지 않을 때 중단

전방 선택

표 6-7 도요타 코롤라 예제의 예측 변수 감소를 위한 전방 선택 방법 결과

```
# The initial model is the constant model - this requires special handling
# in train_model and score_model
def train_model(variables):
    if len(variables) == 0:
        return None
    model = LinearRegression()
    model.fit(train_X[variables], train_y)
    return model

def score_model(model, variables):
    if len(variables) == 0:
        return AIC_score(train_y, [train_y.mean()] * len(train_y), model, df=1)
    return AIC_score(train_y, model.predict(train_X[variables]), model)

best_model, best_variables = forward_selection(train_X.columns, train_model, score_model,
                                                verbose=True)
print(best_variables)

Variables: Age_08_04, KM, HP, Met_Color, Automatic, CC, Doors, Quarterly_Tax, Weight, Fuel_Type_Diesel, Fuel_Type_Petrol
Start: score=11565.07, constant
Step: score=10689.71, add Age_08_04
Step: score=10597.91, add HP
Step: score=10506.08, add Weight
Step: score=10445.17, add KM
Step: score=10435.58, add Quarterly_Tax
Step: score=10419.93, add Fuel_Type_Petrol
Step: score=10418.10, add Fuel_Type_Diesel
Step: score=10417.29, add Automatic
Step: score=10417.29, add None
['Age_08_04', 'HP', 'Weight', 'KM', 'Quarterly_Tax', 'Fuel_Type_Petrol', 'Fuel_Type_Diesel', 'Automatic']
```

단계적 선택 방법

$\emptyset \rightarrow + \dots + +$

- 전방 선택과 유사
- 각 단계마다 추가만 하는 것이 아니라 유의하지 않은 독립변수를 제외하는 것도 검토

```
best_model, best_variables = stepwise_selection(train_X.columns, train_model, score_model, verbose=True)

print(best_variables)

Variables: Age_08_04, KM, HP, Met_Color, Automatic, CC, Doors, Quarterly_Tax, Weight, Fuel_Type_Diesel, Fuel_Type_Petrol
Start: score=11565.07, constant
Step: score=10689.71, add Age_08_04
Step: score=10597.91, add HP
Step: score=10506.08, add Weight
Step: score=10445.17, add KM
Step: score=10435.58, add Quarterly_Tax
Step: score=10419.93, add Fuel_Type_Petrol
Step: score=10418.10, add Fuel_Type_Diesel
Step: score=10417.29, add Automatic
Step: score=10417.29, unchanged None
['Age_08_04', 'HP', 'Weight', 'KM', 'Quarterly_Tax', 'Fuel_Type_Petrol', 'Fuel_Type_Diesel', 'Automatic']
```

부분 집합 선택 알고리즘

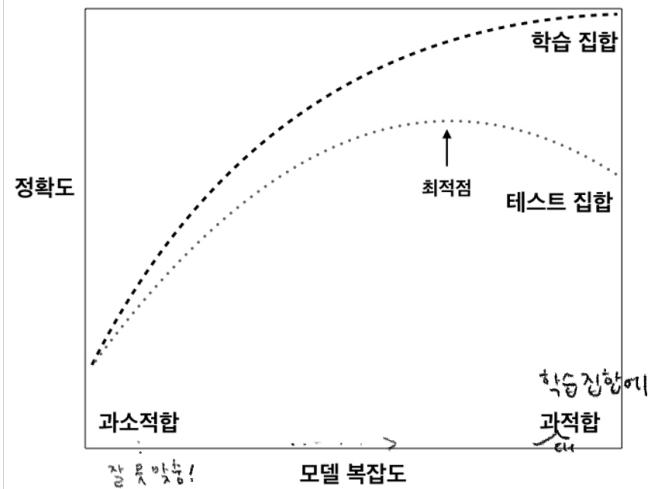
어떤 걸 시도 등일!

Variable	Forward	Backward	Stepwise	Exhaustive
Age_08_04	✓	✓	✓	✓
KM	✓	✓	✓	✓
HP	✓	✓	✓	✓
Met_Color				
Automatic	✓	✓	✓	✓
CC				
Doors				
Quarterly_Tax	✓	✓	✓	✓
Weight	✓	✓	✓	✓
Fuel_TypeDiesel	✓	✓	✓	✓
Fuel_TypePetrol	✓	✓	✓	✓

모형 적합도와 과적합

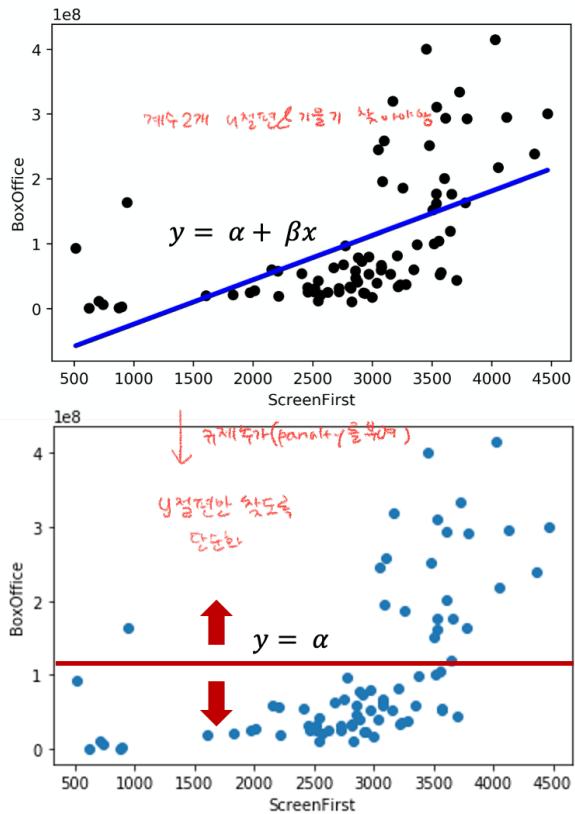
을 줄이는 것이 목표

- 모형 복잡도와 정확도 간의 관계
 - 모형이 복잡하다고 해서 언제나 테스트 집합을 더 잘 분류하지는 못함
 - 모형의 학습에 대해서는 과소적합(underfitting)과 과적합(overfitting)의 문제가 있음
 - 충분히 학습되지 않은 경우에는 학습 집합과 테스트 집합 모두에서 잘 분류하지 못하는 경우가 있으며, 이를 과소적합이라고 함
 - 반면 모형이 복잡해지면 해질수록 학습 집합은 잘 분류하지만, 테스트 집합은 오히려 일정 복잡도를 넘어서게 되면 정확도가 낮아지는 경우가 있으며 이런 상태를 모형이 학습 집합에 과적합되었다고 함



모형 적합도와 과적합

- 모형의 복잡도를 단순화하여 과적합 문제를 줄이기도 하지만, 모형 학습에 규제(Regularization)를 추가하여 계수값이 큰 변수에 페널티를 추가하는 방식으로 모형을 단순화하기도 함
- 규제(Regularization): 모형을 단순하게 하고 과적합의 위험을 감소시키기 위해 모형에 제약을 가하는 것
- 옆의 그림에서 모형은 절편과 기울기 두 개의 모형 파라미터를 가지고 있음. 두 개의 자유도(degree of freedom)를 학습 알고리즘에 부여함
- 기울기 = 0 이라고 강제하면 알고리즘은 절편 파라미터 값만 조절할 수 있기에 한 개의 자유도만 남음



규제

- 변수를 사용할 지 말 지의 이진 선택이 아닌 변수의 계수값에 페널티를 부여하여 계수값을 조절함
- 계수의 크기를 줄일 수 있으며 이를 통해 예측결과의 분산을 줄일 수 있으며, 이를 통해 성과 개선 가능
- 계수의 크기가 0까지 줄여야 한다면 제거되는 효과가 있음

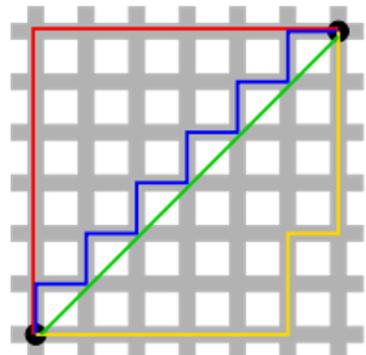
규제

- 규제로 많이 사용되는 노름(Norm)
• 규제로 많이 사용되는 노름(Norm) $\|\beta\|_p = \left(\sum_i |\beta_i|^p \right)^{1/p}$
• $p - \text{Norm}$
• 1 – Norm (L1 Norm) : 거리개념으로
는 Manhattan 거리
 $\|\beta\|_1 = (\sum_i |\beta_i|) = |\beta_1| + |\beta_2| + |\beta_3| + \dots + |\beta_i|$
• 2 – Norm (L2 Norm) : 거리개념으로
는 Euclidean 거리

$$\|\beta\|_2 = \sqrt{\left(\left(\sum_i \beta_i^2 \right) \right)} = \sqrt{\beta_1^2 + \beta_2^2 + \dots + \beta_i^2}$$

β 의 차공이 훨씬 많음

오차 $|y - \hat{y}|$ 최소화하는 계수
 B_0, B_1 찾는것이 회귀분석 목적



(출처: Taxicab geometry, https://en.wikipedia.org/wiki/Taxicab_geometry)

릿지 회귀 Ridge

모든 기계학습 함수는
cost function 有

- 릿지(Ridge) 회귀는 규제가 추가된 선형 회귀 모형

- 선형 회귀 모형이 최소화하고자 하는 값 (Sum of Squared Error, 비용함수)

$$RSS = \sum_{i=1}^n (y_i - \hat{y})^2 = \sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2$$

- L2 노름을 규제로 추가

$$\sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2 + \alpha \sum_{j=1}^p \beta_j^2$$

규제항을 조절
 $\frac{1}{2}, \frac{1}{4}, \dots$
0이면 일반 회귀분석과 ==

- 규제는 $(1/2)(L2 \text{ 노름})^2$ 이 됨

- L2 노름의 계수부분을 λ 라고 표기하기도 함

- sklearn은 alpha로 사용

- alpha = 0 이면 선형회귀 모형이며, alpha 값이 커질수록 규제가 커져 계수 추정치의 값이 작아지게 됨

- β_0 는 규제되지 않음

릿지 회귀 Ridge

```
ridge = Ridge(normalize=True, alpha=1)
ridge.fit(train_X, train_y)
regressionSummary(valid_y, ridge.predict(valid_X))

Regression statistics
    Mean Error (ME) : 154.3286
    Root Mean Squared Error (RMSE) : 1879.7426
    Mean Absolute Error (MAE) : 1353.2735
    Mean Percentage Error (MPE) : -2.3897
    Mean Absolute Percentage Error (MAPE) : 11.1309

bayesianRidge = BayesianRidge(normalize=True)
bayesianRidge.fit(train_X, train_y)
regressionSummary(valid_y, bayesianRidge.predict(valid_X))
alpha = bayesianRidge.lambda_ / bayesianRidge.alpha_
print('Bayesian ridge chosen regularization: ', alpha)

Regression statistics
    Mean Error (ME) : 105.5382
    Root Mean Squared Error (RMSE) : 1313.0217
    Mean Absolute Error (MAE) : 1017.2356
    Mean Percentage Error (MPE) : -0.2703
    Mean Absolute Percentage Error (MAPE) : 9.0012
    Bayesian ridge chosen regularization : 0.004623
```

Ridge 회귀분석의
Alpha 값을 결정해줌

Normalize = True는

계수 표준화

↳ 라쏘합

라쏘 회귀 Lasso

- Lasso: Least absolute shrinkage and selection

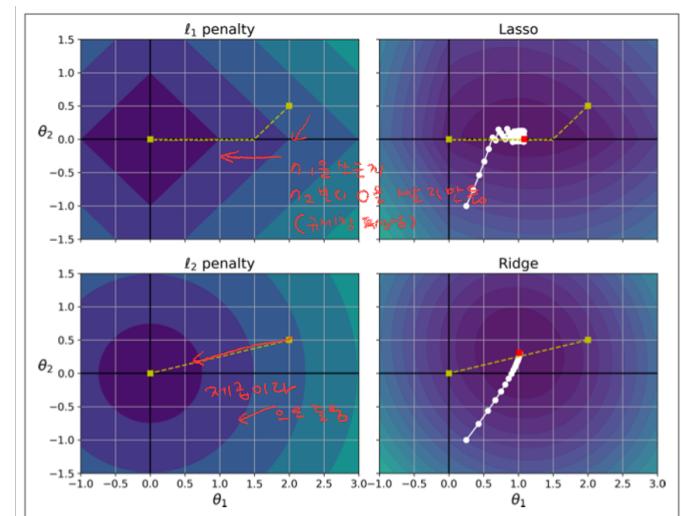
을가하이

- 라쏘(Lasso) 회귀는 L1 규제가 추가된 선형 회귀 모형

$$\sum_{i=1}^n (y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij})^2 + \alpha \sum_{j=1}^p |\beta_j|$$

ℓ_1 가 작을 때
 ℓ_1 을 늘기

- 라쏘 회귀의 중요한 특징은 덜 중요한 변수의 가중치가 0이 됨
- 즉, 라쏘 회귀는 중요한 변수만 자동으로 선택함



(출처: Hands on Machine Learning with Scikit Learn Keras and Tensorflow, 2nd Edition, Aurelien Geron, O'reilly)

라쏘 회귀 Lasso

표 6-9 도요타 코롤라 데이터에 적용된 라쏘(Lasso)와 릿지 회귀(Ridge regression)



정규화된 선형 회귀를 위한 파이썬 코드와 출력(일부)

```
lasso = Lasso(normalize=True, alpha=1)
lasso.fit(train_X, train_y)
regressionSummary(valid_y, lasso.predict(valid_X))
```

Regression statistics

```
    Mean Error (ME) : 120.6311
    Root Mean Squared Error (RMSE) : 1332.2752
    Mean Absolute Error (MAE) : 1021.5286
    Mean Percentage Error (MPE) : -0.2364
    Mean Absolute Percentage Error (MAPE) : 9.0115
```

```
lasso_cv = LassoCV(normalize=True, cv=5)
lasso_cv.fit(train_X, train_y)
regressionSummary(valid_y, lasso_cv.predict(valid_X))
print('Lasso-CV chosen regularization: ', lasso_cv.alpha_)
print(lasso_cv.coef_) 각 특성변수의 계수
```

Regression statistics

```
    Mean Error (ME) : 145.1571
    Root Mean Squared Error (RMSE) : 1397.9428
    Mean Absolute Error (MAE) : 1052.4649
    Mean Percentage Error (MPE) : -0.2966
    Mean Absolute Percentage Error (MAPE) : 9.2918
```

```
Lasso-CV chosen regularization : 3.5138
[-140 -0.018 33.9 0.0 69.4 0.0 0.0 2.71 12.4 0.0 0.0]
```

6개 변수는 0
1개 변수는 0
나머지 변수는 0

RidgeCV 도 있음

Cross Validation

교차 검증 Cross Validation

- K겹 교차검증 (K-fold cross validation)
 - 전체 데이터 집합을 학습 집합과 테스트 집합으로 나누어서 모형 학습에 학습 데이터 집합을 사용하고, 학습된 모형의 성과를 평가하는 테스트 집합을 활용함
 - 무작위로 학습 집합과 테스트 집합을 분류하지만 모든 데이터가 테스트 집합으로 활용되지는 않음
 - 우연에 의해 나누어진 학습 집합과 테스트 집합 조합이 좋거나 나쁜 성과를 보일 수 있음
 - 교차검증은 데이터 집합을 체계적으로 바꿔가면서 모든 데이터에 대해 모형의 성과를 측정하는 방식임

교차 검증 Cross Validation

- K겹 교차검증 (K-fold cross validation)
 - 전체 데이터의 수가 N개인 데이터 집합이 있다면 이를 k개의 부분 집합으로 균등하게 분할함
 - 교차검증은 학습과 테스트를 k번 시행하며 모든 데이터는 테스트 데이터에 1번, 학습에 k-1번 사용됨
 - 예를 들어 k 가 5라면 그림처럼, 5개의 학습 집합과 테스트 집합으로 나누어짐
 - 각각의 학습 데이터와 테스트 데이터를 가지고 모형 학습과 성과를 측정하고 k번 수행된 성과의 평균을 내어 모형의 성과를 측정함
 - 일반적으로 k는 5 또는 10이 활용됨

[5 fold cross validation
10 fold cross validation]

cf. $R = RSS + \delta L_2$
 $L = RSS + \delta L_1$
 $E = RSS + \delta L_1 + (\text{---}) L_2$
p 뒤지 + 뒷고
ElasticNet

