


ML_03. 분류

🕒 생성일	@2022년 6월 8일 오전 1:10
📁 유형	머신러닝/딥러닝
👤 작성자	 동훈 오

목차

- 혼공 머신 : 로지스틱 회귀 이진 분류 / 다중 분류
 - 데이터 로드 및 로지스틱 회귀 모델 훈련 및 예측.
- ISLR : Chapter 4. Classification
- 핸즈온 : Chapter 3. 코드 진행

혼공머신 : 로지스틱 회귀 이진 분류 / 다중 분류

‘혼자 공부하는 머신러닝, 딥러닝’ 도서에서 제시된 로지스틱 회귀 이진 분류 코드를 소개하겠다. 간단한 분류 문제 이기에 전체적인 코드 흐름이 이렇다 정도로만 생각하면 된다.

코드 아래 ‘>>>’ 표시는 출력 결과를 의미하며, 제시된 코드는 전반적으로 데이터의 내용을 미리 알고 있다는 것을 가정으로 한다. (그래서 특정 열의 이름을 어떻게 알았는지 설명없이, 특정 열의 이름을 검색하거나, 삭제하는 코드를 바로 보여준다.)

데이터 준비

```
# 데이터 로드
import pandas as pd
fish = pd.read_csv('https://bit.ly/fish_csv_data')

# unique() 함수를 사용해서 어떤 종류의 생선이 있는지 확인
print(pd.unique(fish['Species']))
>>> ['Bream' 'Roach' 'Whitefish' 'Parkki' 'Perch' 'Pike' 'Smelt']

# Species 열을 타깃, 나머지 열들을 입력 데이터로 사용.
# input data 설정
fish_input = fish[['Weight', 'Length', 'Diagonal', 'Height', 'Width']].to_numpy()

# target data 설정
fish_target = fish['Species'].to_numpy()

# 훈련 데이터, 테스트 데이터 분리
from sklearn.model_selection import train_test_split
train_input, test_input, train_target, test_target = train_test_split(
    fish_input, fish_target, random_state=42)

'''
'print(fish_input[:5])' 코드를 따로 실행시켜보면 fish_input
데이터에서 각 열에 따른 데이터의 scale 이 차이가 큰 것을 알 수 있다.
학습에 앞서 데이터를 정제해야 하는 작업이 필요하고 사이킷런 라이브러리에서는
표준 전처리 클래스인 'StandardScaler' 를 제공한다.
주의할 점은 훈련 세트의 통계값으로 테스트 세트를 변환해야 한다는 점이다.
'''

from sklearn.preprocessing import StandardScaler
ss = StandardScaler()
ss.fit(train_input)
train_scaled = ss.transform(train_input)
test_scaled = ss.transform(test_input)
```

로지스틱 회귀 모델은 일반적으로 이진 분류만 가능하다. 여러 개의 클래스를 분류할 때는 소프트맥스 회귀 모델을 사용하면 된다.

특이하게도, 사이킷런에서 제공하는 로지스틱 회귀 기반 분류기 ‘LogisticRegression’ 은 들어가는 데이터의 타깃 값이 2개일 때는 이진 분류, 여러 개일 때는 다중 분류를 별다른 제약 없이 수행해준다. 소프트맥스 함수 입장에서 클래스가 2일 때 로지스틱과 동일하기 때문에 가능한 수행 능력이라고 생각된다.

여기서는 타깃으로 설정된 ‘Species’ 에서 ‘Bream’ 과 ‘Smelt’ 에 관련된 데이터만 따로 빼내서 그 데이터를 가지고 훈련 → 테스트 데이터 입력 시, 그 데이터가 ‘Bream’ 인지 아니면 ‘Smelt’인지 분류하는 모델을 구성하겠다.

```
# ‘Bream’ 과 ‘Smelt’ 데이터만 전체에서 빼낸다.
# bream_smelt_indexes 배열은 도미와 빙어일 경우 True,
# 그 외는 모두 False 값이 들어 있다.

bream_smelt_indexes = (train_target == 'Bream') | (train_target == 'Smelt')
train_bream_smelt = train_scaled[bream_smelt_indexes]
target_bream_smelt = train_target[bream_smelt_indexes]

# 로지스틱 회귀 모델 훈련
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
lr.fit(train_bream_smelt, target_bream_smelt)

# 예측
# 현재 테스트 세트는 'Bream', 'Smelt'과 다른 클래스 데이터도 같이 있다.
lr.predict(test_input)

# 선형 방정식의 계수 확인.
print(lr.coef_, lr.intercept_)
>>> [[-0.4037798  -0.57620209 -0.66280298 -1.01290277 -0.73168947]] [-2.16155132]
```

다중 분류를 하기 위해서는 원래 계획한 데이터, (train_scaled, train_target) 을 사용하면 된다. 이번에는 학습의 반복 횟수를 충분하게 주어 1000 정도로 설정해서 학습을 진행시키겠다.

```
lr = LogisticRegression(C=20, max_iter=1000)
lr.fit(train_scaled, train_target)
print(lr.score(train_scaled, train_target))
print(lr.score(test_scaled, test_target))

>>> 0.9327731092436975
>>> 0.925

# 우선 테스트 세트에 있는 처음 5 개의 데이터만 예측해 보겠다.
print(lr.predict(test_scaled[:5]))
>>> ['Perch' 'Smelt' 'Pike' 'Roach' 'Perch']

# 위 샘플들에 대한 확률 출력
proba = lr.predict_proba(test_scaled[:5])
print(np.round(proba, decimals=3))

>>> [[0.    0.014 0.841 0.    0.136 0.007 0.003]
      [0.    0.003 0.044 0.    0.007 0.946 0.    ]
      [0.    0.    0.034 0.935 0.015 0.016 0.    ]
      [0.011 0.034 0.306 0.007 0.567 0.    0.076]
      [0.    0.    0.904 0.002 0.089 0.002 0.001]]
```

ISLR : 4. Classification

회귀 모델에서는 타깃값 Y 에 대해 수치로 표현하려고 했었다. 하지만, 출력에 대해서 ‘qualitative’ 하게 나타내야 하는 경우가 있다. 대표적으로 눈동자의 색깔은 ‘qualitative’ 하다. 파랑색, 갈색, 녹색의 조합으로 이루어져 있기 때문이다. qualitative 는 categorical 의 표현으로 대체할

수 있다. 그리고 이러한 특성을 가진 출력을 다루는 과정을 분류(classification) 이라고 한다.

몇몇 방식들에선 , 분류를 위한 첫 번째 과정으로 'qualitative variables' 의 클래스를 확률로서 계산한다. 분류를 위해 회귀 방식을 사용하는 것이다. 이러한 점에선 분류와 회귀 간의 경계가 모호하다.

분류기를 만들기 위한 다양한 접근법들이 있다. 그 중 대표적으로 활용되는 로지스틱 회귀(logistic regression), 선형판별분석(linear discriminant analysis), k-최근접 이웃 분류(knn classification) 을 소개하겠다.

분류에 쓰이는 다소 computational 한 방식들에는 generalized additive model, 결정트리, 랜덤 포레스트, 부스팅, 서포트 벡터 머신(SVM) 등이 있으며 이것은 다른 챕터에서 설명하겠다.

왜 선형 회귀는 분류에 적합하지 않는가?

선형 회귀가 'qualitative variables' 를 다루는데 적합하지 않은 이유를 다음의 그래프를 통해 설명한다.

If we use linear regression, some of our estimates might be outside the [0,1] interval (see Figure 4.2), making them hard to interpret as probabilities!

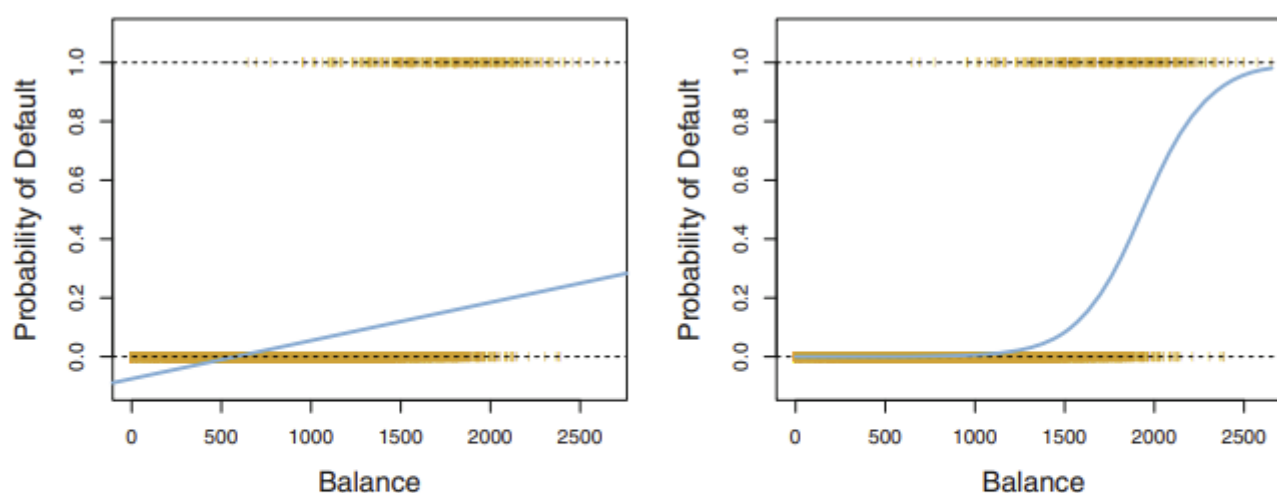


FIGURE 4.2. Classification using the **Default** data. Left: Estimated probability of **default** using linear regression. Some estimated probabilities are negative! The orange ticks indicate the 0/1 values coded for **default** (No or Yes). Right: Predicted probabilities of **default** using logistic regression. All probabilities lie between 0 and 1.

Curiously, it turns out that the classifications that we get if we use linear regression to predict a binary response will be the same as for the linear discriminate analysis (LDA) procedure we discuss in later.

Logistic Regression

위의 그래프에서 나타나듯이(오른쪽), 로지스틱 회귀에서 출력 Y는 특정 클래스에 데이터가 속할 확률을 보여준다. - “확률”이란 표현을 유심히 봐야한다.

데이터 X 가 우리가 확인하고자 하는 클래스 (양성 클래스 = 1) 가 될 확률은 다음과 같이 나타낸다.

$$p(X) = \Pr(Y = 1|X)$$

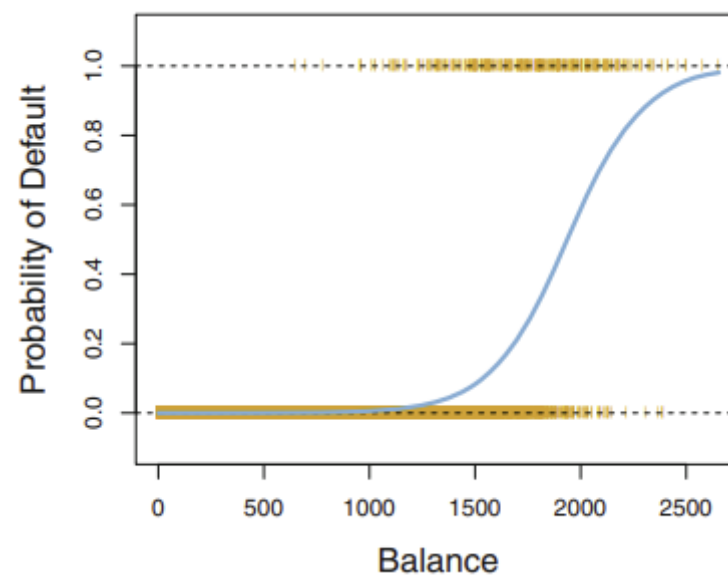
선형 회귀에서는 이 확률을 다음과 같이 표현한다.

$$p(X) = \beta_0 + \beta_1 X. \quad (4.1)$$

하지만 이렇게 직선으로 나타내는 선형 회귀 모델은 위 그래프에서 보듯이 그 결과가 0-1 사이를 벗어나는 경우가 생긴다. 출력이 0-1 사이의 값으로 제한되는 함수 $p(X)$ 가 필요하며 로지스틱 함수는 만족스럽다.

$$p(X) = \frac{e^{\beta_0 + \beta_1 X}}{1 + e^{\beta_0 + \beta_1 X}}. \quad (4.2)$$

왜 만족스럽냐하면, 로지스틱 함수의 개형을 통해 확인할 수 있다.



위 그래프에서 Y 가 1.0에 가까울 수록 우리가 찾는 클래스 (=양성 클래스) 가 될 가능성이 커지게 된다. 위 그래프에서는 balance(신용카드 사용 잔액)에 따른 파산 가능성을 나타내었는데, 사용잔액이 적을수록 확실히 파산 가능성이 낮으며(그렇다고 0은 아니다.), 사용잔액이 많을수록 확실히 파산 가능성이 높게 된다. (그렇다고 확률이 1.0 은 아니다.)

로지스틱 함수의 첫 번째 장점은 양성 클래스, 음성 클래스에 대한 가능성이 확연히 대비 된다는 점이다.

로지스틱 함수의 두 번째 장점은 언제나 S-shape 을 가지고 있어서 X 에 관계없이 항상 0-1 사이의 출력값을 가진다는 점이다.

로지스틱 함수는 어떻게 도출되었는지를 살펴보면 왜 “로지스틱 회귀” 라고 표현되는지 알 수 있다. 식 변형 통해 선형 회귀와 로지스틱 간의 관계를 나타내면 다음과 같다.

$$\frac{p(X)}{1 - p(X)} = e^{\beta_0 + \beta_1 X}. \quad (4.3)$$

$$\log \left(\frac{p(X)}{1 - p(X)} \right) = \beta_0 + \beta_1 X. \quad (4.4)$$

$p(X) / [1-p(X)]$ 는 odds (= 결과의 승산은 결과가 발생하지 않을 확률에 대한 결과가 발생할 확률의 비율) 라고 한다. 수식 (4.4) 에서 좌변을 log-odds 또는 logit 이라고 부른다. 결국 X 에 대해 linear 한 logit 을 가지며, 이것이 로지스틱 회귀라고 불리는 이유다.

계수는 어떻게 추정하는가?

회귀 모델에서 least squares 방식으로 미지수인 계수들을 추정했다. least squares을 로지스틱에 적용할 수 있지만, 더 일반적인 방식은 maximum likelihood(최대우도 추정) 방법이며 통계학적 특성의 관점에서 더 적합하다.

원문에서 최대 우도 추정을 사용하는 이유를 단순한 직관으로 설명한다.

We try to find β_0 and β_1 such that plugging these estimates into the model for $p(X)$, given in (4.2), yields a number close to one for all individuals who defaulted, and a number close to zero for all individuals who did not.

즉, X 값이 양 극단으로 갈수록 양성 클래스 1이 될 가능성이 극단적으로 대비되도록 만들기 위해서이다. likelihood 는 ‘우도’ 또는 ‘가능도’ 라고도 표현된다는 점을 고려해보자.

가능도함수(likelihood function) 는 다음과 같다.

$$\ell(\beta_0, \beta_1) = \prod_{i:y_i=1} p(x_i) \prod_{i':y_{i'}=0} (1 - p(x_{i'})). \quad (4.5)$$

위 함수값을 최대화하는 β_0, β_1 을 찾으면 된다.

maximum likelihood 방법은 비선형적 모델을 fit 하는데 일반적인 접근방식이다. maximum likelihood 를 통해 로지스틱을 fit 하는 과정은 이 책의 범위를 벗어나지만, R, 파이썬 패키지를 통해서도 간편하게 구현 가능하다.

Multiple Logistic Regression

단순 선형 회귀에서 다항 선형 회귀로 확장하는 것과 같이, 로지스틱 회귀에도 multiple 상황을 고려할 수 있다. (혼동하지 않아야 할 것은, 타깃 클래스는 여전히 2개이며, 클래스에 영향을 미치는 특성만 늘어난 것이다.)

$$\log \left(\frac{p(X)}{1 - p(X)} \right) = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p, \quad (4.6)$$

$$p(X) = \frac{e^{\beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p}}{1 + e^{\beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p}}. \quad (4.7)$$

Linear Discriminant Analysis (LDA)

로지스틱 회귀에서는 $Pr(Y = k|X = x)$ 을 로지스틱 함수를 이용해 direct 하게 모델링했다. 이번에는 좀 더 통계적 접근법으로, Y 의 conditional 한 분포를 모델링할 것이다. 타깃 클래스 Y 를 추정하는데 있어, 간접적이고 alternative 한 방식이라고 생각하면 된다. 이러한 접근법에 대해 필요성은 다음과 같이 설명 된다.

When the classes are well-separated, the parameter estimates for the logistic regression model are surprisingly unstable. Linear discriminant analysis does not suffer from this problem.

If n is small and the distribution of the predictors X is approximately normal in each of the classes, the linear discriminant model is again more stable than the logistic regression model.

Linear discriminant analysis is popular when we have more than two response classes.