

교육 제목	데이터 기반 인공지능 시스템 엔지니어 양성 과정_ 머신러닝
교육 일시	2021년 10월 19일
교육 장소	YGL C-6 학과장 & 자택(디스코드 이용한 온라인)
교육 내용	
오전	<p>지난 시간 Review & 기계학습(Machine Learning ML) 혼공머(혼자 공부하는 머신러닝) 03-3. 특성 공학과 규제</p> <pre>from sklearn.model_selection import train_test_split train_input, test_input, train_target, test_target = train_test_split(perch_full, perch_weight, random_state=42) """## 사이킷런의 변환기""" from sklearn.preprocessing import PolynomialFeatures poly = PolynomialFeatures() poly.fit([[2, 3]]) print(poly.transform([[2, 3]])) poly = PolynomialFeatures(include_bias=False) poly.fit(train_input) train_poly = poly.transform(train_input) print(train_poly.shape) poly.get_feature_names() test_poly = poly.transform(test_input)</pre>

```
"""## 다중 회귀 모델 훈련하기"""

from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(train_poly, train_target)
print(lr.score(train_poly, train_target))

print(lr.score(test_poly, test_target))

poly = PolynomialFeatures(degree=5, include_bias=False)
poly.fit(train_input)
train_poly = poly.transform(train_input)
test_poly = poly.transform(test_input)
print(train_poly.shape)

lr.fit(train_poly, train_target)
print(lr.score(train_poly, train_target))

print(lr.score(test_poly, test_target))
```

```
"""## 규제"""

from sklearn.preprocessing import StandardScaler
ss = StandardScaler()
ss.fit(train_poly)
train_scaled = ss.transform(train_poly)
test_scaled = ss.transform(test_poly)
```

```
"""## 릿지 회귀"""

from sklearn.linear_model import Ridge
ridge = Ridge()
ridge.fit(train_scaled, train_target)
print(ridge.score(train_scaled, train_target))

print(ridge.score(test_scaled, test_target))

import matplotlib.pyplot as plt
train_score = []
test_score = []

alpha_list = [0.001, 0.01, 0.1, 1, 10, 100]
for alpha in alpha_list:
    # 릿지 모델을 만듭니다
    ridge = Ridge(alpha=alpha)
    # 릿지 모델을 훈련합니다
    ridge.fit(train_scaled, train_target)
    # 훈련 점수와 테스트 점수를 저장합니다
    train_score.append(ridge.score(train_scaled, train_target))
    test_score.append(ridge.score(test_scaled, test_target))

plt.plot(np.log10(alpha_list), train_score)
plt.plot(np.log10(alpha_list), test_score)
plt.show()

ridge = Ridge(alpha=0.1)
ridge.fit(train_scaled, train_target)
print(ridge.score(train_scaled, train_target))
print(ridge.score(test_scaled, test_target))
```

```
"""## 라쏘 회귀"""

from sklearn.linear_model import Lasso
lasso = Lasso()
lasso.fit(train_scaled, train_target)
print(lasso.score(train_scaled, train_target))

print(lasso.score(test_scaled, test_target))

train_score = []
test_score = []
alpha_list = [0.001, 0.01, 0.1, 1, 10, 100]
for alpha in alpha_list:
    # 라쏘 모델을 만듭니다
    lasso = Lasso(alpha=alpha, max_iter=10000)
    # 라쏘 모델을 훈련합니다
    lasso.fit(train_scaled, train_target)
    # 훈련 점수와 테스트 점수를 저장합니다
    train_score.append(lasso.score(train_scaled, train_target))
    test_score.append(lasso.score(test_scaled, test_target))

plt.plot(np.log10(alpha_list), train_score)
plt.plot(np.log10(alpha_list), test_score)
plt.show()

lasso = Lasso(alpha=10)
lasso.fit(train_scaled, train_target)
print(lasso.score(train_scaled, train_target))
print(lasso.score(test_scaled, test_target))

print(np.sum(lasso.coef_ == 0))
```

혼공머(혼자 공부하는 머신러닝)

04-1. 로지스틱 회귀

- **로지스틱 회귀**는 선형 방정식을 사용한 분류 알고리즘입니다. 선형 회귀와 달리 시그모이드 함수나 소프트맥스 함수를 사용하여 클래스 확률을 출력할 수 있습니다.
- **다중 분류**는 타깃 클래스가 2개 이상인 분류 문제입니다. 로지스틱 회귀는 다중 분류를 위해 소프트맥스 함수를 사용하여 클래스를 예측합니다.
- **시그모이드 함수**는 선형 방정식의 출력을 0과 1 사이의 값으로 압축하며 이진 분류를 위해 사용합니다.
- **소프트맥스 함수**는 다중 분류에서 여러 선형 방정식의 출력 결과를 정규화하여 합이 1이 되도록 만듭니다.

scikit-learn

- **LogisticRegression**은 선형 분류 알고리즘인 로지스틱 회귀를 위한 클래스입니다.
- solver 매개변수에서 사용할 알고리즘을 선택할 수 있습니다. 기본값은 'lbfgs'입니다. 사이킷 런 0.17 버전에 추가된 'sag'는 확률적 평균 경사 하강법 알고리즘으로 특성과 샘플 수가 많을 때 성능은 빠르고 좋습니다. 사이킷런 0.19 버전에는 'sag'의 개선 버전인 'saga'가 추가되었습니다.
- penalty 매개변수에서 L2 규제(릿지 방식)와 L1 규제(라쏘 방식)를 선택할 수 있습니다. 기본값은 L2 규제를 의미하는 'l2'입니다.
- C 매개변수에서 규제의 강도를 제어합니다. 기본값은 1.0이며 값이 작을수록 규제가 강해집니다.

- **predict_proba()** 메서드는 예측 확률을 반환합니다.

이진 분류의 경우에는 샘플마다 음성 클래스와 양성 클래스에 대한 확률을 반환합니다.
다중 분류의 경우에는 샘플마다 모든 클래스에 대한 확률을 반환합니다.

- **decision_function()**은 모델이 학습한 선형 방정식의 출력을 반환합니다.

이진 분류의 경우 양성 클래스의 확률이 반환됩니다. 이 값이 0보다 크면 양성 클래스, 작거나 같으면 음성 클래스로 예측합니다.
다중 분류의 경우 각 클래스마다 선형 방정식을 계산합니다. 가장 큰 값의 클래스가 예측 클래스가 됩니다.

04-2. 확률적 경사 하강법

- **확률적 경사 하강법**은 훈련 세트에서 샘플 하나씩 꺼내 손실 함수의 경사를 따라 최적의 모델을 찾는 알고리즘입니다. 샘플을 하나씩 사용하지 않고 여러 개를 사용하면 미니배치 경사 하강법이 됩니다. 한 번에 전체 샘플을 사용하면 배치 경사 하강법이 됩니다.
- **손실 함수**는 확률적 경사 하강법이 최적화할 대상입니다. 대부분의 문제에 잘 맞는 손실 함수가 이미 정의되어 있습니다. 이진 분류에는 로지스틱 회귀(또는 이진 크로스엔트로피) 손실 함수를 사용합니다. 다중 분류에는 크로스엔트로피 손실 함수를 사용합니다. 회귀 문제에는 평균 제곱 오차 손실 함수를 사용합니다.
- **에포크**는 확률적 경사 하강법에서 전체 샘플을 모두 사용하는 한 번 반복을 의미합니다. 일반적으로 경사 하강법 알고리즘은 수십에서 수백 번의 에포크를 반복합니다.

scikit-learn

- **SGDClassifier**는 확률적 경사 하강법을 사용한 분류 모델을 만듭니다.

loss 매개변수는 확률적 경사 하강법으로 최적화할 손실 함수를 지정합니다. 기본값은 서포트 벡터 머신을 위한 'hinge' 손실 함수입니다. 로지스틱 회귀를 위해서는 'log'로 지정합니다.

penalty 매개변수에서 규제의 종류를 지정할 수 있습니다. 기본값은 L2 규제를 위한 'l2'입니다. L1 규제를 적용하려면 'l1'로 지정합니다. 규제 강도는 alpha 매개변수에서 지정합니다. 기본값은 0.0001입니다.

max_iter 매개변수는 에포크 횟수를 지정합니다. 기본값은 1000입니다.

tol 매개변수는 반복을 멈출 조건입니다. n_iter_no_change 매개변수에서 지정한 에포크 동안 손실이 tol 만큼 줄어들지 않으면 알고리즘이 중단됩니다. tol 매개변수의 기본값은 0.001이고 n_iter_no_change 매개변수의 기본값은 5입니다.

- **SGDRegressor**는 확률적 경사 하강법을 사용한 회귀 모델을 만듭니다.

loss 매개변수에서 손실 함수를 지정합니다. 기본값은 제곱 오차를 나타내는 'squared_loss'입니다.

앞의 SGDClassifier에서 설명한 매개변수는 모두 SGDRegressor에서 동일하게 사용됩니다.

05-1. 결정 트리

- **결정 트리**는 예 / 아니오에 대한 질문을 이어나가면서 정답을 찾아 학습하는 알고리즘입니다. 비교적 예측 과정을 이해하기 쉽고 성능도 뛰어납니다.
- **불순도**는 결정 트리가 최적의 질문을 찾기 위한 기준입니다. 사이킷런은 지니 불순도와 엔트로피 불순도를 제공합니다.
- **정보 이득**은 부모 노드와 자식 노드의 불순도 차이입니다. 결정 트리 알고리즘은 정보 이득이 최대화되도록 학습합니다.
- 결정 트리는 제한 없이 성장하면 훈련 세트에 과대적합되기 쉽습니다. **가지치기**는 결정 트리의 성장을 제한하는 방법입니다. 사이킷런의 결정 트리 알고리즘은 여러 가지 가지치기 매개변수를 제공합니다.
- **특성 중요도**는 결정 트리에 사용된 특성이 불순도를 감소하는데 기여한 정도를 나타내는 값입니다. 특성 중요도를 계산할 수 있는 것이 결정 트리의 또 다른 큰 장점입니다.

pandas

- **info()**는 데이터프레임의 요약된 정보를 출력합니다. 인덱스와 컬럼 타입을 출력하고 널(null)이 아닌 값의 개수, 메모리 사용량을 제공합니다. verbose 매개변수의 기본값 True를 False로 바꾸면 각 열에 대한 정보를 출력하지 않습니다.
- **describe()**는 데이터프레임 열의 통계 값을 제공합니다. 수치형일 경우 최소, 최대, 평균, 표준편차와 사분위값 등이 출력됩니다.
문자열 같은 객체 타입의 열은 가장 자주 등장하는 값과 횟수 등이 출력됩니다.
percentiles 매개변수에서 백분위수를 지정합니다. 기본값은 [0.25, 0.5, 0.75]입니다.

scikit-learn

- **DecisionTreeClassifier**는 결정 트리 분류 클래스입니다.
criterion 매개변수는 불순도를 지정하며 기본값은 지니 불순도를 의미하는 'gini'이고 'entropy'를 선택하여 엔트로피 불순도를 사용할 수 있습니다.
splitter 매개변수는 노드를 분할하는 전략을 선택합니다. 기본값은 'best'로 정보 이득이 최대가 되도록 분할합니다. 'random'이면 임의로 노드를 분할합니다.
max_depth는 트리가 성장할 최대 깊이를 지정합니다. 기본값은 None으로 리프 노드가 순수하거나 min_samples_split보다 샘플 개수가 적을 때까지 성장합니다.
min_samples_split은 노드를 나누기 위한 최소 샘플 개수입니다. 기본값은 2입니다.
max_features 매개변수는 최적의 분할을 위해 탐색할 특성의 개수를 지정합니다. 기본값은 None으로 모든 특성을 사용합니다.
- **plot_tree()**는 결정 트리 모델을 시각화합니다. 첫 번째 매개변수로 결정 트리 모델 객체를 전달합니다.
max_depth 매개변수로 나타낼 트리의 깊이를 지정합니다. 기본값은 None으로 모든 노드를 출력합니다.
feature_names 매개변수로 특성의 이름을 지정할 수 있습니다.
filled 매개변수를 True로 지정하면 타깃값에 따라 노드 안에 색을 채웁니다.
- **plot_tree()**는 결정 트리 모델을 시각화합니다. 첫 번째 매개변수로 결정 트리 모델 객체를 전달합니다.
max_depth 매개변수로 나타낼 트리의 깊이를 지정합니다. 기본값은 None으로 모든 노드를 출력합니다.
feature_names 매개변수로 특성의 이름을 지정할 수 있습니다.
filled 매개변수를 True로 지정하면 타깃값에 따라 노드 안에 색을 채웁니다.