머신러닝응용 제11강

Ensemble Learning 2.

첨단공학부 김동하교수

제11강 Ensemble Learning 2.

1	부스팅의 개념에 대해 학습한다.
2	대표적인 부스팅 알고리즘인 AdaBoost에 대해 학습한다.
3	AdaBoost 알고리즘을 어떻게 해석할 수 있는지 학습한다.

핵심 단어

- 부스팅
- AdaBoost
- 경험위험함수
- 가파른 강하 알고리즘

11강. Ensemble Learning 2.

01. 부스팅 개요

1) 부스팅의 기본 아이디어

- 예측력이 약한 모형 (weak learner) 들을 결합하여 강한 예측모형을 만드는 것.

- Weak learner
  - 랜덤하게 예측하는 것보다 약간 좋은 예측력을 지닌 모형.
  - 반면, 강한 예측모형은 예측력이 최적에 가까운 예측모형을 뜻함.

1) 부스팅의 기본 아이디어

◆ 예제

◆ 경마에서 우승하는 말을 맞추기 위한 전략

- 여러 전문가들로부터 기본적인 전략을 들을 수 있음.
- 최근에 가장 많이 우승한 말에게 배팅
- 주변 사람이 가장 선호하는 말에 배팅
- 등등.

1) 부스팅의 기본 아이디어

- ◆이들은 임의로 예측하는 것보다는 좋은 전략이지만 정확성이 높지는 않음.
- ◆이들의 전략을 모두 모아서 결합한다면 높은 예측력을 발휘할 수 있을 것.
- ◆각각의 전략을 합칠 때, 가중치를 어떻게 부여할 것인가?

11강. Ensemble Learning 2.

02. AdaBoost

1) AdaBoost

- Freund and Schapire (1997)
- 최초의 부스팅 알고리즘
- 이진 분류 문제를 푸는 부스팅 알고리즘.
  - $y \in \{-1, 1\}$

2) AdaBoost의 알고리즘

1. n 개의 가중치를 $w_i = \frac{1}{n}, i = 1, \ldots, n$ 로 초기화

2. $\mathrm{m} = 1, \ldots, M$ 에 대해서 다음의 과정을 반복.

   ① 가중치 w_i 를 이용하여 이진 분류기 $f_m(x)$ 를 적합.

   ② $\mathrm{err_m}$ 를 다음과 같이 계산.

$$err_m = \frac{\sum_{i=1}^{n} w_i \cdot I(y_i \neq f_m(x_i))}{\sum_{i=1}^{n} w_i}$$

   ③ $\mathrm{c_m} = \log\big((1 - err_m)/err_m\big)$ 로 설정.

   ④ 가중치 w_i 를 다음과 같이 업데이트.

$$w_i \leftarrow w_i \cdot \exp(c_m I(y_i \neq f_m(x_i)))$$

3. 단계 2에서 얻은 M개의 분류기를 결합하여 최종 분류기 $sign(\sum_{m=1}^{M} c_m f_m(x))$ 를 얻는다.

3) AdaBoost의 해석

◆단계 2의 ③, ④가 매우 중요.

◆예측모형 $f_m(x)$ 가 랜덤한 추측보다 조금 더 좋은 예측력을 갖는다고 가정

- $f_m(x)$의 (가중) 오분류율인 err_m은 0.5보다 작게 됨.
- 따라서 c_m은 0보다 큰 값을 가짐.

3) AdaBoost의 해석

◆④에서 다음 단계에 사용될 관측치가 다음과 같은 방식으로 할당됨.

- 정분류된 관측치의 가중치는 기존의 값과 같음.
- 오분류된 관측치의 가중치는 증가.

3) AdaBoost의 해석

◆ 다시 말해서 AdaBoost 알고리즘은

- 매 반복마다 오분류된 관측치의 가중치는 증가시키고,
- 정분류된 가중치는 감소시키면서 해당 시점의 예측모형을 만들어감.

4) AdaBoost의 weak learner

- AdaBoost는 의사결정나무를 weak learner로 사용

- Bagging이나 Random Forest와는 달리 그루터기(stump)를 많이 사용한다.

  - 그루터기: 한 번의 분리만을 사용한 의사결정나무

5) AdaBoost의 해석 2-가파른 강하 알고리즘

- AdaBoost 는 지수손실함수에 대한 경험 위험 최소 추정량을 가파른 강하 알고리즘을 사용해 추정한 결과라는 사실이 증명됨.

- 지수손실함수:

$$L(y, f(x)) = \exp(-y \cdot f(x))$$

5) AdaBoost의 해석 2-가파른 강하 알고리즘

- 지수손실함수를 이용한 경험위험함수:

$$\frac{1}{n}\sum_{i=1}^{n}\exp(-y_i \cdot f(x_i))$$

- 가파른 강하 알고리즘: 다음과 같은 형태로 업데이트되는 알고리즘.

$$u^{(t+1)} = u^{(t)} + \alpha^{(t+1)} \cdot \Delta u^{(t+1)}$$

  - 예: Gradient descent algorithm, Newton-Raphson algorithm

5) AdaBoost의 해석 2-가파른 강하 알고리즘

- 다음의 경험 위험 함수를 최소화하는 함수를 찾고자 함.

$$\frac{1}{n}\sum_{i=1}^{n}\exp(-y_i \cdot f(x_i))$$

- m번 반복했을 때의 함수를 $F_m(x)$라 하면, m+1번째 반복을 통해 얻는 함수는 다음의 결과가 될 것임.

$$F_m(x) + \alpha_{m+1} \cdot f_{m+1}(x)$$

5) AdaBoost의 해석 2-가파른 강하 알고리즘

- 주어진 $F_m(x)$ 하에서 경험 위험 함수를 최소화하는 $\alpha_{m+1}, f_{m+1}(x)$ 를 찾는 것이 목표.
- 즉,

$$\alpha_{m+1}, f_{m+1}(x) = \underset{\alpha, f}{\operatorname{argmin}} \sum_{i=1}^{n} \exp(-y_i \cdot [F_m(x_i) + \alpha \cdot f(x_i)])$$

- 고려하는 α는 0 이상의 실수, f는 이진 분류기.
  - 대부분 f는 의사결정나무를 사용.

5) AdaBoost의 해석 2-가파른 강하 알고리즘

- **최적의 $f_{m+1}(x)$는 $\alpha = 1$를 가정해고 계산해도 상관 없음.**

$$f_{m+1}(x) = \underset{f}{\operatorname{argmin}} \sum_{i=1}^{n} \exp(-y_i \cdot [F_m(x_i) + f(x_i)])$$

$$= \underset{f}{\operatorname{argmin}} \sum_{i=1}^{n} w_i \cdot \exp(-y_i \cdot f(x_i))$$

- **여기서,** $w_i = \exp\left(-y_i \cdot F_m(x_i)\right), i = 1, \ldots, n.$

5) AdaBoost의 해석 2-가파른 강하 알고리즘

- $f_{m+1}(x)$가 주어졌을 때 경험 위험 함수를 최소화하는 α는 다음과 같음:

$$\alpha_{m+1} = \underset{\alpha}{\operatorname{argmin}} \sum_{i=1}^{n} \exp(-y_i \cdot [F_m(x_i) + \alpha \cdot f_{m+1}(x_i)])$$

$$= \underset{\alpha}{\operatorname{argmin}} \sum_{i=1}^{n} w_i \cdot \exp(-y_i \cdot \alpha \cdot f_{m+1}(x_i))$$

$$= \underset{\alpha}{\operatorname{argmin}} \sum_{i=1}^{n} w_i \cdot \exp(-\alpha) \cdot I\left(y_i = f_m(x_i)\right) + \sum_{i=1}^{n} w_i \cdot \exp(\alpha) \cdot I\left(y_i \neq f_m(x_i)\right)$$

5) AdaBoost의 해석 2-가파른 강하 알고리즘

◆ 앞의 식을 최소화하는 α를 찾으면 다음과 같음.

$$\alpha_{m+1} = \frac{1}{2}\log\left(\frac{1 - err_{m+1}}{err_{m+1}}\right)$$

✓가파른 강하 알고리즘으로 구한 최종 함수와 AdaBoost 알고리즘으로 구한 함수는 왜 같은가?

- ½를 곱해도 최종 함수값의 부호는 같기 때문!

6) AdaBoost의 추가 설명

- AdaBoost 알고리즘의 본래 목적은 훈련오차를 빨리 그리고 쉽게 줄이는 것.

- 그러나 AdaBoost 알고리즘이 제안된 이후 본래의 목적과는 상관없이 예측오차도 감소한다는 것이 경험적으로 증명.

- AdaBoost 알고리즘은 배깅과는 다르게 간단한 의사결정나무인 그루터기 (stump)를 많이 사용.

7) 다양한 부스팅 알고리즘

◆ RealBoost

- Schapire and Singer (1999)
- AdaBoost 를 변형하여 회귀 문제에 응용한 알고리즘.

7) 다양한 부스팅 알고리즘

Gradient Boosting

- Friedman (2001)
- 부스팅 알고리즘을 기울기 강하 알고리즘으로 해석하여 이를 확장한 알고리즘.
- 분류, 회귀 등 모든 문제에 사용 가능한 알고리즘.
- 가장 많이 사용되는 부스팅 알고리즘.

11강. Ensemble Learning 2.

03. Python을 이용한 실습

1) 데이터 설명

- 보스턴 주택 가격 데이터
  - 1978년에 발표된 데이터로 미국 보스턴 지역의 주택 가격에 영향을 미치는 요소들을 정리
  - 총 13가지의 요소들과 주택 가격으로 이루어져 있음.

- 부스팅 기법을 이용하여 주택 가격을 예측하는 회귀 모형을 만들자.

2) 환경설정

◆필요한 패키지 불러오기

```
import pandas as pd
import numpy as np
import os
from sklearn.datasets import load_boston
from sklearn.model_selection import train_test_split

from sklearn.metrics import confusion_matrix
from sklearn.ensemble import GradientBoostingRegressor
#from sklearn.ensemble import GradientBoostingClassifier

import matplotlib.pyplot as plt
```

◆데이터 불러오기

```
boston = load_boston()

X_boston = pd.DataFrame(boston.data, columns=boston.feature_names)
y_boston = pd.DataFrame(boston.target, columns=['MEDV']).iloc[:,0]
```

◆데이터 분할하기

```
X_train, X_test, y_train, y_test = \
train_test_split(X_boston, y_boston,
                 test_size = 0.3, random_state=123)
```

4) 부스팅 적합하기

- 1000개의 weak learner 사용.
- 각각의 weak learner는 최대 깊이 3의 의사결정나무

```
gbm_model = GradientBoostingRegressor(n_estimators = 1000,
                                      max_depth = 3)

gbm_model.fit(X_train, y_train)
```

4) 부스팅 적합하기

◆시험 데이터에서의 성능 확인하기

```
gbm_pred = gbm_model.predict(X_test)
print((y_test-gbm_pred).pow(2).mean())
```

```
13.372383277786511
```

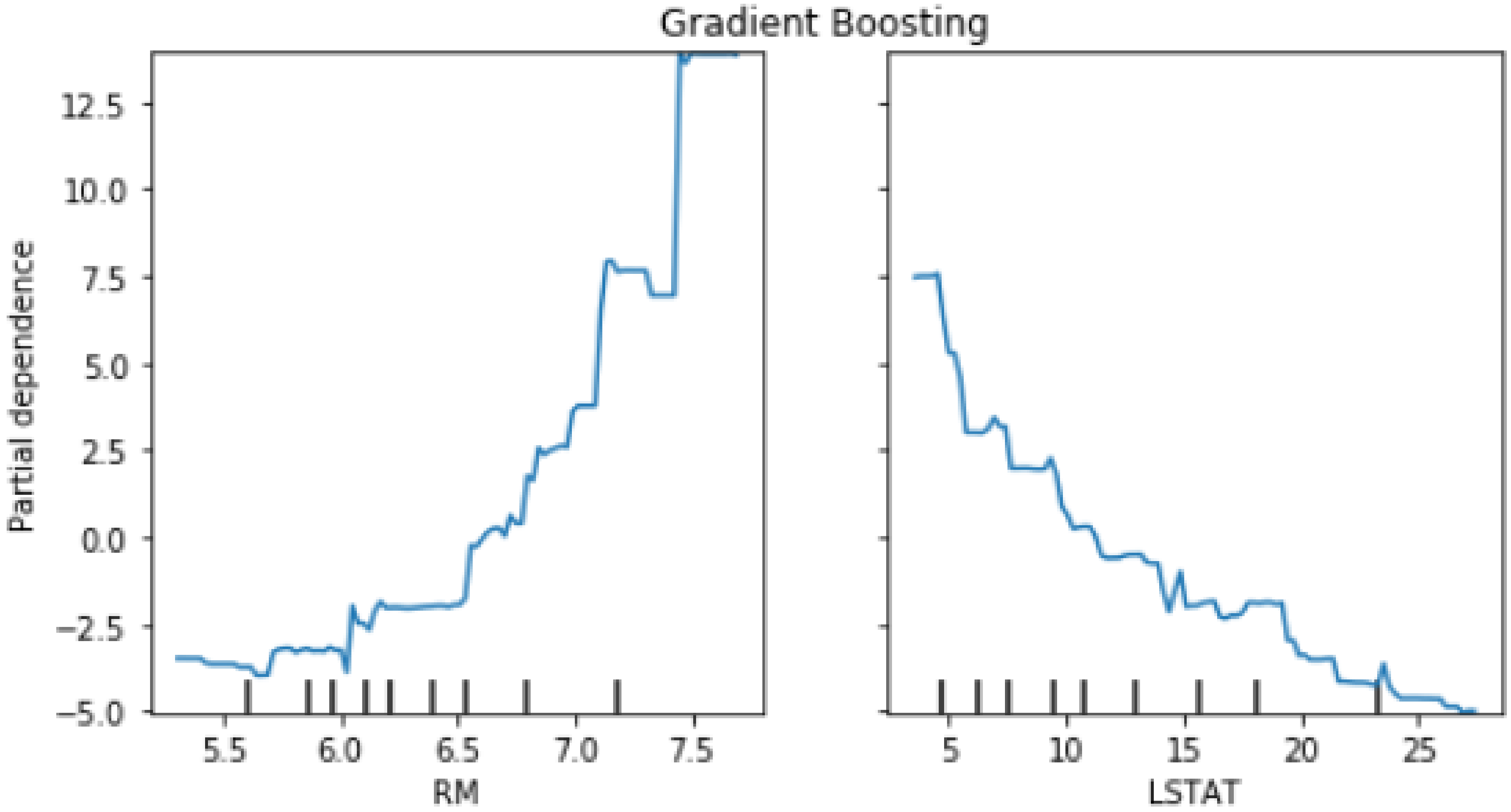
5) 모형 결과 시각화하기

◆ 부분 의존성 그림 그리기.

```
from sklearn.inspection import plot_partial_dependence
fig, ax = plt.subplots(figsize=(8, 4))
ax.set_title("Gradient Boosting", fontsize=12)
tree_disp = plot_partial_dependence(gbm_model, X_train, ["RM","LSTAT"], ax=ax)
```

5) 모형 결과 시각화하기

◆ 부분 의존성 그림 그리기.



5) 모형 결과 시각화하기

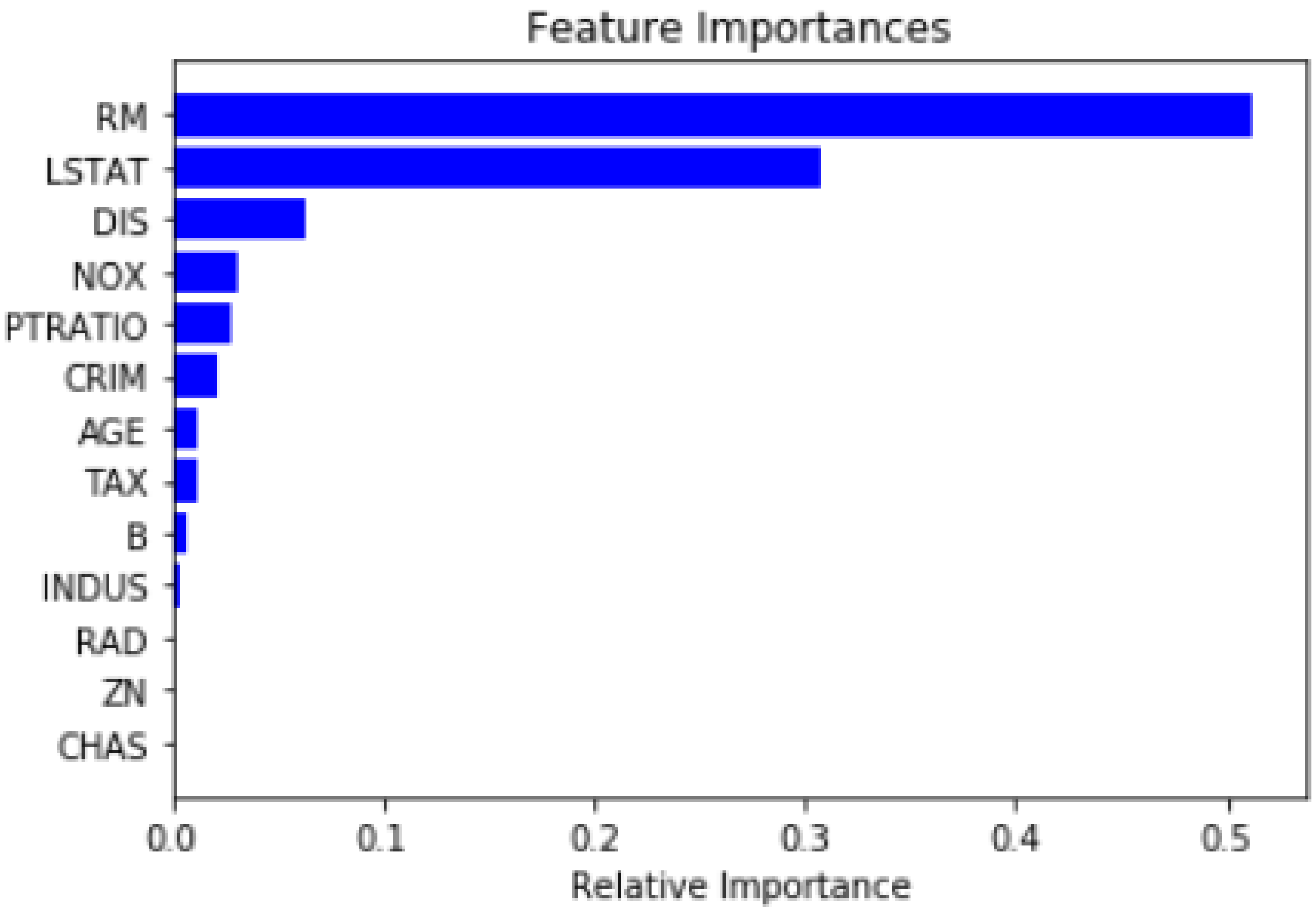
◆ 변수 중요도 그림 그리기.

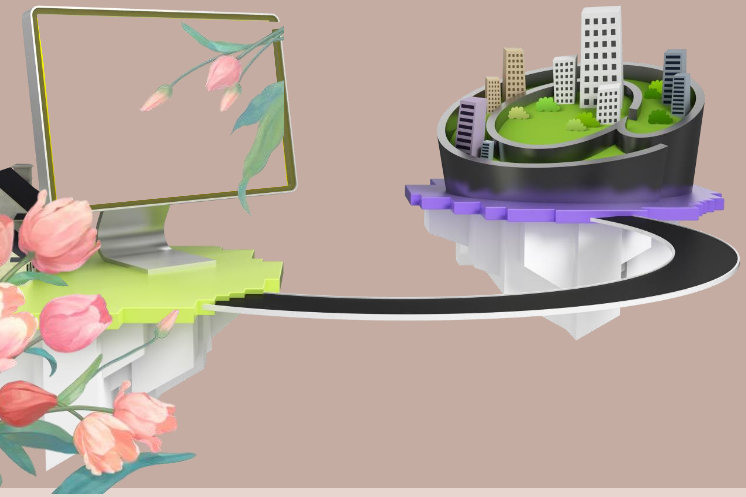
```
importances = gbm_model.feature_importances_
indices = np.argsort(importances)

plt.title('Feature Importances')
plt.barh(range(len(indices)), importances[indices], color='b', align='center')
plt.yticks(range(len(indices)), [boston.feature_names[i] for i in indices])
plt.xlabel('Relative Importance')
plt.show()
```

5) 모형 결과 시각화하기

◆ 변수 중요도 그림 그리기.





다음시간안내

제12강

Support Vector Machine