7장 앙상블 학습과 랜덤 포레스트 2부

감사의 글

자료를 공개한 저자 오렐리앙 제롱과 강의자료를 지원한 한빛아카데미에게 진심어린 감사를 전합니다.

주요 내용

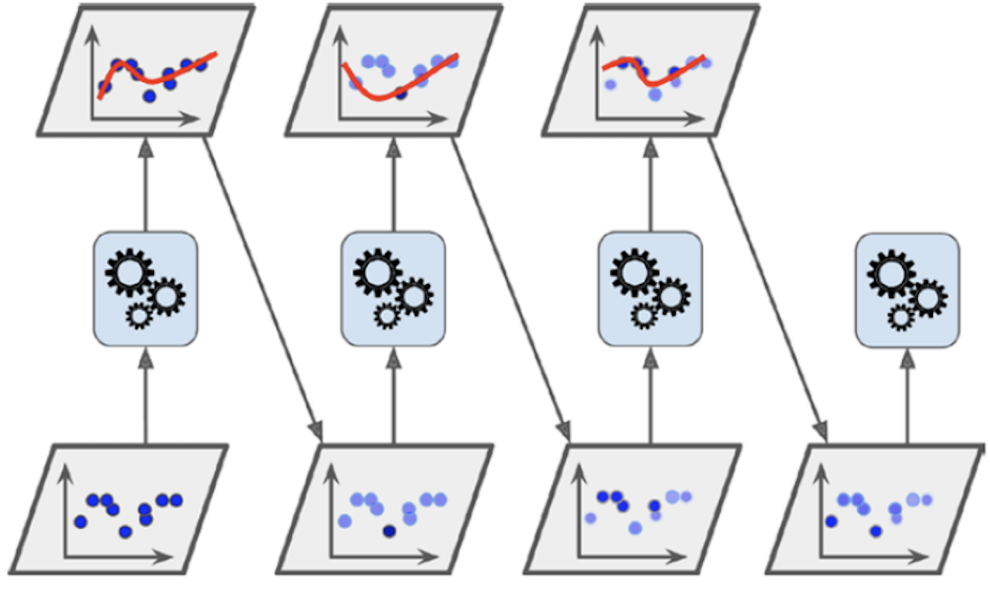
- 1부
    - 앙상블 학습이란?
    - 배깅
        - 페이스팅
        - 램덤 패치 / 랜덤 서브스페이스
        - 램덤포레스트
- 2부
    - 부스팅
    - 스태킹

7.5 부스팅

- 부스팅(boosting): 성능이 약한 학습기의 여러 개를 선형으로 연결하여 강한 성능의 학습기를 만드는 앙상블 기법. 대표적 알고리즘은 다음과 같음.
    - 에이다부스트(AdaBoost)
    - 그레이디언트 부스팅(Gradient Boosting)
- 순차적으로 이전 학습기의 결과를 바탕으로 성능을 조금씩 높혀감. 즉, 편향을 줄여나감.
- 성능이 약한 예측기의 단점을 보완하여 좋은 성능의 예측기를 훈련해 나가는 것이 부스팅의 기본 아이디어
- 순차적으로 학습하기에 배깅/페이스팅에 비해 확장성이 떨어짐

에이다부스트(AdaBoost)

- 좀 더 나은 예측기를 생성하기 위해 잘못 적용된 가중치를 조정하여 새로운 예측기를 추가하는 앙상블 기법. 이전 모델이 제대로 학습하지 못한, 즉 과소적합했던 샘플들에 대한 가중치를 더 높이는 방식으로 새로운 모델 생성.
- 새로운 예측기는 학습하기 어려운 샘플에 조금씩 더 잘 적응하는 모델이 연속적으로 만들어져 감.

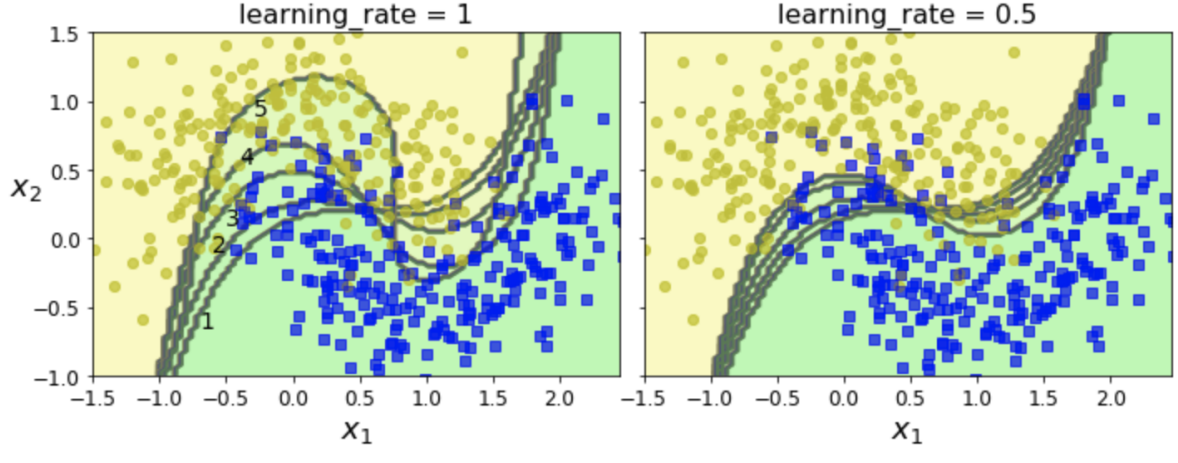


샘플 가중치

- 훈련중에 특정 샘플을 보다 강조하도록 유도하는 가중치를 가리킴.
- 사이킷런 모델의 `fit()` 메서드는 `sample_weight` 옵션인자를 추가로 사용하여 훈련 세트의 각 샘플에 대한 가중치를 지정할 수 있음.
- 샘플 가중치는 모든 샘플에 대해 주어지며, `sampe_weight` 옵션인자를 이용하여 각 샘플에 대한 가중치를 결정함.
- `sample_weight` 옵션인자를 지정하지 않으면 모든 샘플의 가중치를 동일하게 간주함.
- 참고: SVC의 `fit()` 메서드 정의

에이다부스트 알고리즘 작동 과정

- moons 데이터셋에 rbf 커널을 사용하는 SVC 모델을 5번 연속 새로 생성하는 방식으로 학습한 결과를 보여줌.
- 새로운 예측기의 `fit()` 메서드는 이전 예측기의 경우와 다른 `sample_weight` 옵션값을 사용함.
- 새로운 예측기는 이전의 예측기의 예측값이 틀린 샘플을 보다 강조하도록 유도됨.
- 왼편과 오른편은 학습률만 다름.
    - **주의사항:** `learnign_rate`는 기존에 설명한 학습률과 다른 의미이며, 각 예측기의 기여도 조절에 사용됨.



사이키런의 에이다부스트

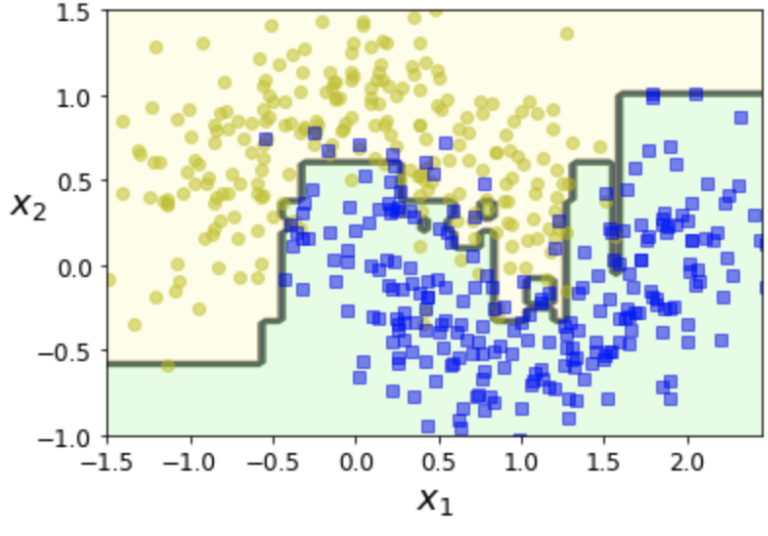
- 분류 모델: `AdaBoostClassifier`
- 회귀 모델: `AdaBoostRegressor`

예제: 에이다부스트 + 결정트리

- `AdaBoostClassifier` 의 기본 모델임.

```
ada_clf = AdaBoostClassifier(DecisionTreeClassifier(max_depth=1),
                             n_estimators=200, algorithm="SAMME.R",
                             learning_rate=0.5, random_state=42)
```

- 훈련 세트: moons 데이터셋



그레이디언트 부스팅

- 이전 학습기에 의한 오차를 보정하도록 새로운 예측기를 순차적으로 추가하는 아이디어는 에이다부스트와 동일
- 샘플의 가중치를 수정하는 대신 이전 예측기가 만든 **잔차**(residual error)에 대해 새로운 예측기를 학습시킴
- 잔차(residual error): 예측값과 실제값 사이의 오차

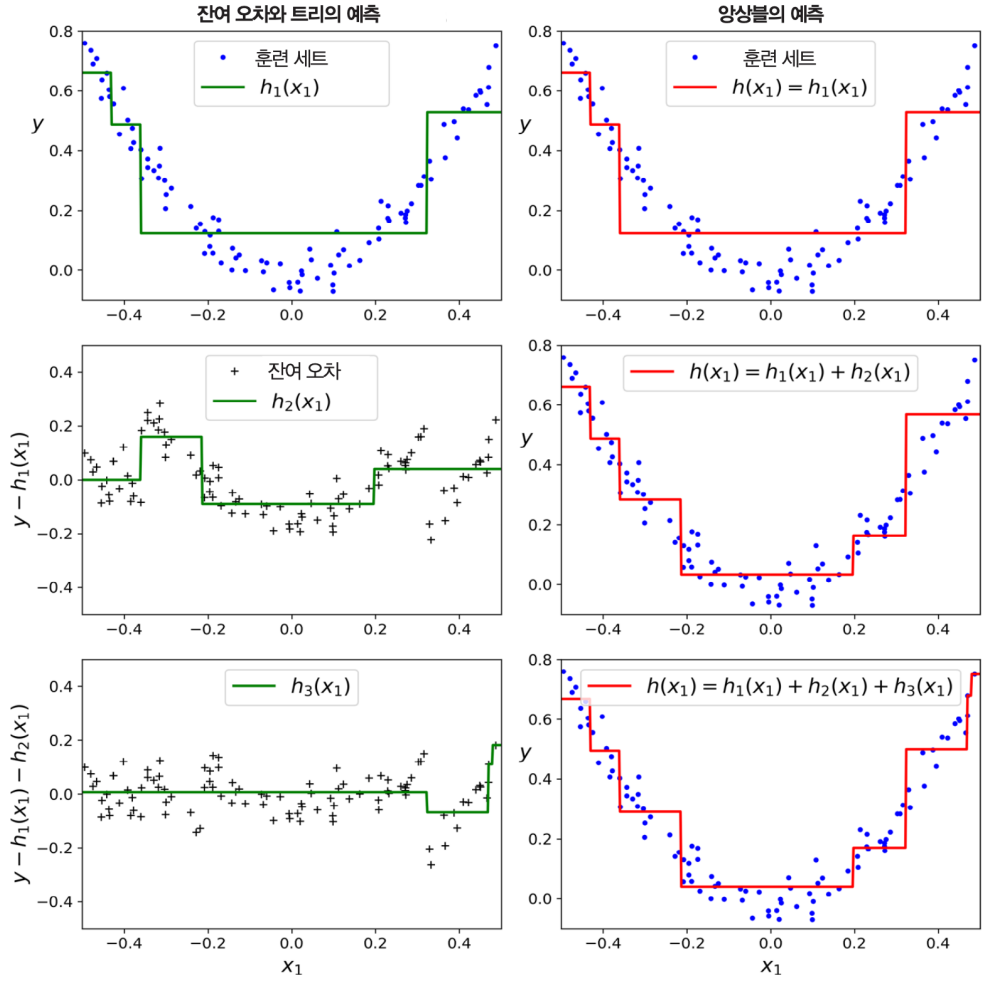
사이킷런 그레이디언트 부스팅 모델

- 분류 모델: `GradientBoostingClassifier`
    - `RandomForestClassifier`와 비슷한 하이퍼파라미터를 제공
- 회귀 모델: `GradientBoostingRegressor`
    - `RandomForestRegressor`와 비슷한 하이퍼파라미터를 제공

그레이디언트 부스티드 회귀 나무(GBRT) 예제: 그레이디언트 부스팅 (회귀)+ 결정트리

- 2차 다항식 데이터셋에 결정트리 3개를 적용한 효과와 동일하게 작동

```
gbrt = GradientBoostingRegressor(max_depth=2,
                                 n_estimators=3,
                                 learning_rate=1.0)
```

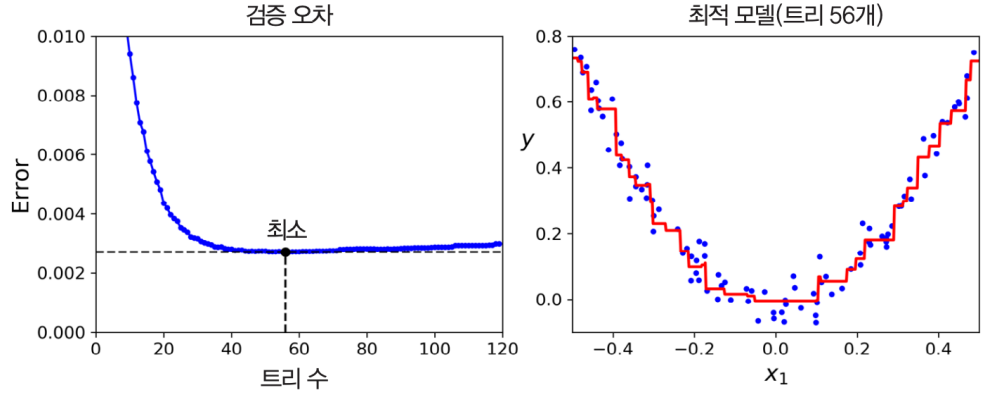

잔여 오차와 트리의 예측
앙상블의 예측
훈련 세트
$h_1(x_1)$
$h(x_1) = h_1(x_1)$
잔여 오차
$h_2(x_1)$
$h(x_1) = h_1(x_1) + h_2(x_1)$
$h_3(x_1)$
$h(x_1) = h_1(x_1) + h_2(x_1) + h_3(x_1)$
y
$y - h_1(x_1)$
$y - h_1(x_1) - h_2(x_1)$
x_1

`learning_rate`(학습률)

- `learnign_rate`는 기존에 설명한 학습률과 다른 의미의 학습률.
    - 각 결정트리의 기여도 조절에 사용
- 수축(shrinkage) 규제: 학습률을 낮게 정하면 많은 수의 결정트리 필요하지만 성능 좋아짐.
- 이전 결정트리에서 학습된 값을 전달할 때 사용되는 비율
    - 1.0이면 그대로 전달
    - 1.0보다 작으면 해당 비율 만큼 조금만 전달

최적의 결정트리 수 확인법

- 조기종료 기법 활용



확률적 그레이디언트 부스팅

- 각 결정트리가 훈련에 사용할 훈련 샘플의 비율을 지정하여 학습: `subsample=0.25` 등 비율 지정
- 훈련 속도 빨라짐.
- 편향 높아지지만, 분산 낮아짐.

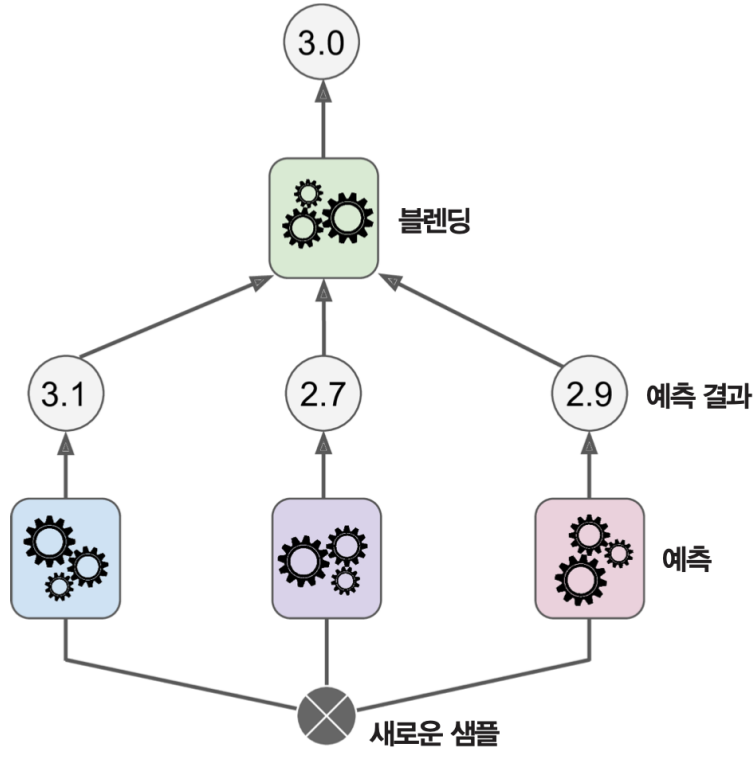
XGBoost

- Extreme Gradient Boosting의 줄임말.
- 빠른 속도, 확장성, 이식성 뛰어남.
- 조기종료 등 다양한 기능 제공.

```
import xgboost
  xgb_reg = xgboost.XGBRegressor(random_state=42)
  xgb_reg.fit(X_train, y_train,
              eval_set=[(X_val, y_val)],
              early_stopping_rounds=2)
```

7.6 스태킹

- 배깅방식의 응용으로 볼 수 있는 기법
- 다수결을 이용하는 대신 여러 예측값을 훈련 데이터로 활용하는 예측기를 훈련시키는 기법

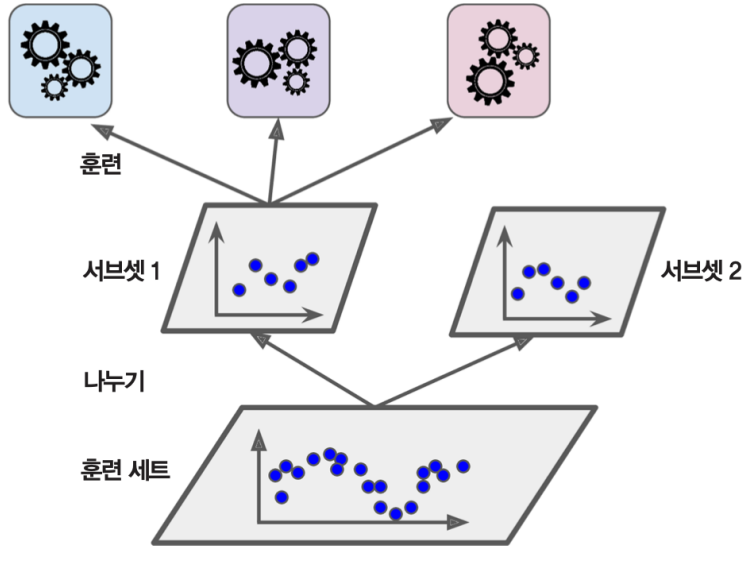


스태킹 모델 훈련법

- 책에서는 스태킹 기법을 소개만하고 코드 구현은 연습문제 9번에서 설명한다.
- 여기서는 사이킷런 0.22부터 지원하는 스태킹 모델을 활용하여 코드구현을 설명한다.
- 참조: Stacked generalization

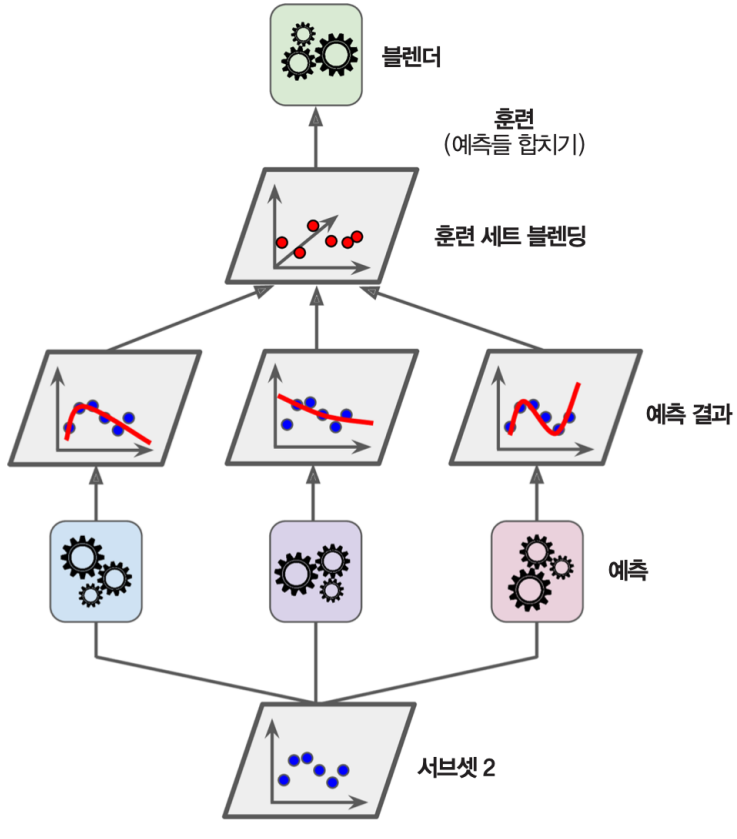
1층 훈련

- 먼저 훈련 세트를 훈련세트1과 훈련세트2로 이등분한다.
- 하나의 훈련세트1의 전체 샘플을 이용하여 주어진 예측기들을 각자 독립적으로 훈련시킨다.



2층 훈련

- 훈련세트2의 모든 샘플에 대해 훈련된 예측기별로 예측값을 생성한다.
- 예측값들로 이루어진 훈련세트를 이용하여 믹서기 모델(블렌더)을 훈련시킨다.
    - 2층 훈련에 사용되는 샘플의 차원은 1층에 사용되는 예측기 개수이다.



사이킷런의 `StackingRegressor` 모델 활용법

```
estimators = [('ridge', RidgeCV()),
              ('lasso', LassoCV(random_state=42)),
              ('knr', KNeighborsRegressor(n_neighbors=20,
                                          metric='euclidean'))]

final_estimator = GradientBoostingRegressor(n_estimators=25, subsample=0.5,
                        min_samples_leaf=25, max_features=1, random_state=42)

reg = StackingRegressor(estimators=estimators,
                        final_estimator=final_estimator)
```

사이킷런의 `StackingClassifier` 모델 활용법

```
estimators = [('rf', RandomForestClassifier(n_estimators=10, random_state=42)),
              ('svr', make_pipeline(StandardScaler(),
                                    LinearSVC(random_state=42)))]

clf = StackingClassifier(estimators=estimators,
                         final_estimator=LogisticRegression())
```

스태킹 모델의 예측값

- 레이어를 차례대로 실행해서 믹서기(블렌더)가 예측한 값을 예측값으로 지정한다.
- 훈련된 스태킹 모델의 편향과 분산이 훈련에 사용된 모델들에 비해 모두 감소한다.

다층 스태킹

- 2층에서 여러 개의 믹서기(블렌더)를 사용하고, 그위 3층에 새로운 믹서기를 추가하는 방식으로 다층 스태킹을 훈련시킬 수 있다.
- 다층 스태킹의 훈련 방식은 2층 스태킹의 훈련 방식을 반복하면 된다.

예제: 3층 스태킹 모델 훈련과정

- 훈련세트를 세 개의 부분 훈련세트로 나눈다.
- 훈련세트1은 1층 예측기 훈련에 사용한다.
- 훈련세트2은 2층 믹서기 훈련에 사용한다.
- 훈련세트3은 3층 믹서기 훈련에 사용한다.