# 기계학습 앙상블 학습과 랜덤포레스트 정리노트

4조-(이희구, 유제우)

## ※ 4조 의견

**질문) 앙상블이 뭘까?**

- 앙상블이란 여러 개의 결과를 종합하여 예측값을 지정하는 학습이다. 추가로 앙상블 학습이란 예측기 여러 개의 결과를 종합하여 예측값을 지정하는 학습이다. 이런 앙상블을 사용하기 위한 방법으로는 앙상블 학습을 지원하는 앙상블 학습 알고리즘이 있다.

**질문) 투표식 분류기의 특징에는 무엇이 있을까?**

- 앙상블에 포함된 분류기들 사이의 독립성이 전재되는 경우 개별 분류기 보다 정확한 예측이 가능하다.
- 독립성이 보장되지 못한다면 투표식 분류기의 성능이 더 낮아질 수 있다.

**질문) 큰 수의 법칙이란게 뭐야?**

- 반복 시행하는 횟수가 많거나 표본이 커질수록 일정한 수준으로 수렴 되고 비교적 정확한 예측이 가능하다는 의미이다.

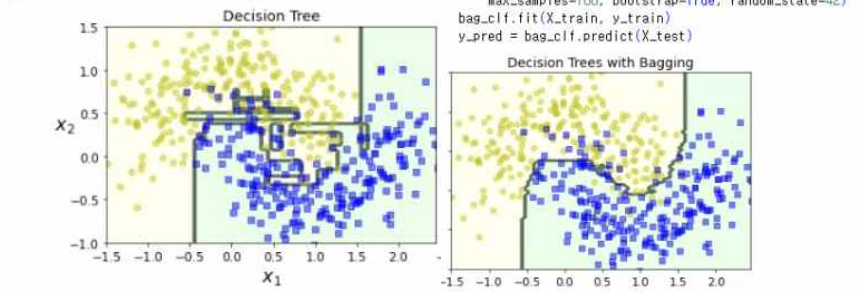
**질문) 배깅을 할 때 주의해야할 점이 뭐가 있을까?**

- 통계 분야에서 부트스트래핑, 즉, 중복허용 리샘플링으로 불린다.
- 배깅 방식은 동일 훈련 샘플을 여러번 샘플링할 수 있다.

**질문) 이 코드 결과 더 좋은 방식은 뭘까?**

```
tree_clf = DecisionTreeClassifier(random_state=42)
tree_clf.fit(X_train, y_train)
y_pred_tree = tree_clf.predict(X_test)
print(accuracy_score(y_test, y_pred_tree))
```

```
from sklearn.ensemble import BaggingClassifier
from sklearn.tree import DecisionTreeClassifier

bag_clf = BaggingClassifier(
    DecisionTreeClassifier(), n_estimators=500,
    max_samples=100, bootstrap=True, random_state=42)
bag_clf.fit(X_train, y_train)
y_pred = bag_clf.predict(X_test)
```




- 일반적으로 배깅 방식이 편향은 키우고, 분산은 줄임. 앙상블은 비슷한 편향에서 더 작은 분산을 만든다. 전반적으로 배깅 방식이 좀 더 나은 모델을 생성한다.

**질문) obb 샘플? 그게 뭐야?**

- obb 샘플이란 out-of-bag의 약어로 선택되지 않은 훈련 샘플을 말한다. 이 샘플을 활용하여 앙상블 학습에 사용된 개별 예측기의 성능을 평가할 수 있다. 앙상블의 평가는 각 예측기의 oob 평가를 평균하여 얻는다.

**질문) 랜덤 패치가 뭐야?**

- 훈련 샘플과 훈련 특성 모두를 대상으로 중복을 허용하며, 임의의 샘플 수와 임의의 특성 수만큼을 샘플링해서 학습하는 기법이다.

**질문) 랜덤 서브스페이스는 뭐야?**

- 전체 훈련 세트를 학습 대상으로 삼지만 훈련 특성은 임의의 특성 수만큼 샘플링해서 학습하는 기법이다.

**질문) 랜덤 서브스페이스는 뭐야?**

- 전체 훈련 세트를 학습 대상으로 삼지만 훈련 특성은 임의의 특성 수만큼 샘플링해서 학습하는 기법이다.

**질문) 랜덤포레스트가 학습시키는 모델인건 아는데 정확히 정의가 뭐야?**

- 랜덤포레스트는 배깅/페이스팅 방법을 적용한 결정트리의 앙상블을 최적화한 모델이다. 분류 용도와 회귀 용도로 사용될 때 다르게 사용된다.

**질문) 랜덤포레스트는 어떻게 쓰는거야?**

- 트리의 노드를 분할할 때 전체 특성 중에서 최선의 특성을 찾는 대신 무작위성을 더 주입하며, 트리를 더욱 다양하게 만들고, 편향을 손해 보는 대신 분산을 낮춘다.

**질문) 엑스트라 트리는 또 뭐야?**

- 랜덤포레스트의 노드 분할 방식이다. 특성과 특성 임곗값 모두 무작위 선택을 하고 일반적인 램덤포레스트보다 속도가 훨씬 빠르다. 이 방식을 사용하면 편향은 늘고, 분산은 줄어든다. 엑스트라 트리를 만들려면 사이킷런의 ExtraTreesClassifier를 사용한다.

**질문) 특성 중요도?**

- 해당 특성을 사용한 노드가 평균적으로 불순도를 얼마나 감소시키는지를 측정한다.

**질문) 부스팅이란?**

- 성능이 약한 학습기를 여러 개 연결하여 강한 성능의 학습기를 만드는 앙상블 기법이다. 앞의 모델을 보완해나가면서 순차적으로 예측기를 학습시키고, 부스팅 방법에는 에이다부스트, 그레이디언트 부스팅이 있다.

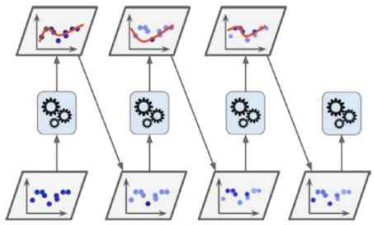
**질문) 에이다부스트?**

- 좀 더 나은 예측기를 생성하기 위해 잘못 적용된 가중치를 조정하여 새로운 예측기를 추가하는 앙상블 기법이다.

**질문) 에이다부스트는 정확히 뭘 하는거야?**

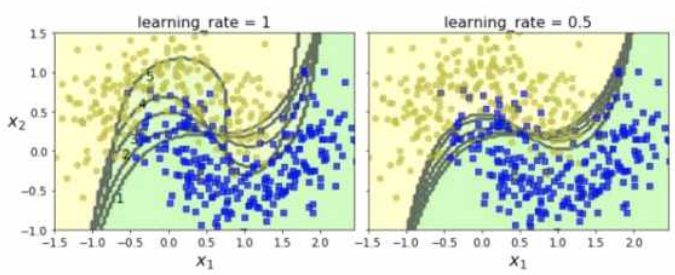
- 이전 모델이 제대로 학습하지 못한, 즉 과소적합했던 훈련 샘플들에 대한 가중치를 더 높이는 방식으로 새로운 모델을 생성한다.
- 새로운 예측기는 학습하기 어려운 샘플에 조금씩 더 잘 적응하는 모델이 연속적으로 만들어져 간다.

에이다 부스트 동작원리



▲ 그림7-7 샘플의 가중치를 업데이트하면서 순차적으로 학습하는 에이다부스트 동작

**질문) 연속된 예측기의 결정 경계를 설명해줘**

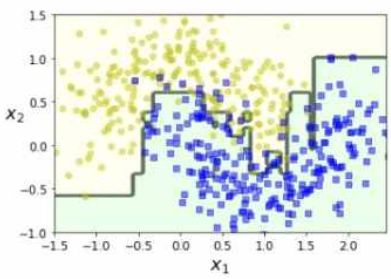



▲ 그림7-8 연속된 예측기의 결정 경계

- 경사 하강법의 모델 파라미터 조정은 비용 최소화, 예측이 더 좋아지도록 앙상블에 예측기를 추가한다.
- 학습률을 반으로 낮추면 잘못 분류된 샘플의 가중치는 반복마다 절반 정도만 높아진다.

**질문) 에이다부스트 실습은 어떻게 시켜?**

```
from sklearn.ensemble import AdaBoostClassifier

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

plot_decision_boundary(ada_clf, X, y)
```




- 사이킷런의 AdaBoostClassifier를 사용한다. 200개의 결정 트리를 기반으로 하는 에이다부스트 분류기를 훈련한다.

**질문) 그레이디언트 부스팅 설명해줘**

- DecisionTreeRegressor를 훈련세트에 학습

```
from sklearn.tree import DecisionTreeRegressor

tree_reg1 = DecisionTreeRegressor(max_depth=2, random_state=42)
tree_reg1.fit(X, y)

y2 = y - tree_reg1.predict(X)
tree_reg2 = DecisionTreeRegressor(max_depth=2, random_state=42)
tree_reg2.fit(X, y2)
```

- 두 번째 예측기가 반드 잔여 오차에 세번째 회귀 모델을 훈련

```
y3 = y2 - tree_reg2.predict(X)
tree_reg3 = DecisionTreeRegressor(max_depth=2, random_state=42)
tree_reg3.fit(X, y3)
```

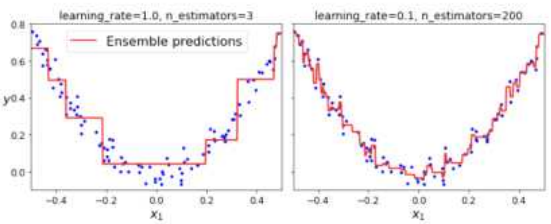
- 새로운 샘플에 대한 예측을 만드려면 모든 트리의 예측을 더함.

```
X_new = np.array([[0.8]])

y_pred = sum(tree.predict(X_new) for tree in (tree_reg1, tree_reg2, tree_reg3))
```

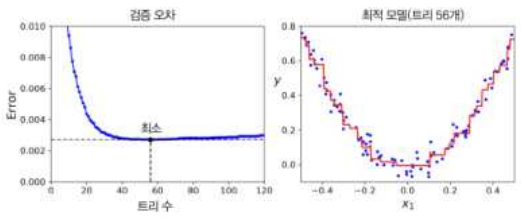
- 이전 학습기에 의한 오차를 보정하도록 새로운 예측기를 순차적으로 추가하는 아이디어는 에이다 부스트와 동일하다. 샘플의 가중치를 수정하는 대신 이전 예측기가 만든 잔여 오차(residual error)에 대해 새로운 예측기를 학습시킨다. 잔여오차는 예측값과 실제값 사이의 오차를 의미한다. 잔여 오차를 줄이는 방향으로 모델을 학습시키는데, 이 때 경사 하강법(GradientDescent)을 사용하여 최적화한다.

**질문) 성능 좋아지게 어떻게 해?**

```
from sklearn.ensemble import GradientBoostingRegressor

gbrt = GradientBoostingRegressor(max_depth=2, n_estimators=3, learning_rate=1.0, random_state=42)
gbrt.fit(X, y)
```




▲ 예측기가 부족한 경우와 너무 많은 경우의 GBRT 앙상블




▲ 조기 종료를 사용하여 트리 수 튜닝

- GradientBoostingRegressor 를 사용하여, GBRT 앙상블을 훈련한다.
- learning_rate 매개변수가 각 트리의 기여 정도를 조절
- 축소 규제를 시킨다. 학습률을 낮게 정하면 많은 수의 의사결정나무 필요하지만 성능이 좋아진다.