3-2-4 앙상블

앙상블 정의

주어진 자료로부터 여러 개의 예측모형들을 조합하여 하나의 최종 예측 모형을 만드는 방법

- 다중 모델 조합, 분류기 조합이 있다.

학습방법의 불안전성

학습집단의 작은 변화에 의해 예측모형이 크게 변하는 경우 그 '학습 방법은 불안정하다'

- 가장 안정적인 방법으로는 1-near neighbor(가장 가까운 자료만 변하지 않으면 예측 모형이 변하지 않음), 선형회귀모형(최소제곱법으로 추정해 모형 결정)이 존재한다
- 가장 불안정한 방법으로는 의사결정나무가 있다.

배깅(Bagging)의 어원적 의미와 개념

용어 구성: Bootstrap + Aggregating

- **Bootstrap**: 통계학에서 '부트스트래핑'은 데이터에서 복원추출(sampling with replacement)로 여러 샘플을 만들어내는 기법
  - 원래는 작은 데이터셋으로 신뢰구간이나 분산을 추정할 때 사용하던 방법
- **Aggregating**: 여러 모델의 예측 결과를 평균내거나 다수결로 종합하는 방법
  - 이게 바로 예측의 안정성을 높이는 핵심 기법

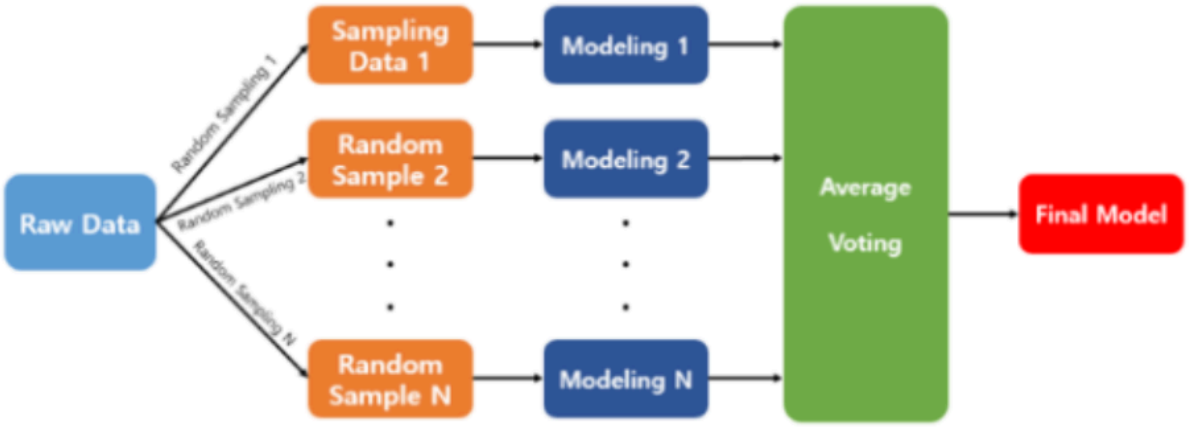
'Bagging'의 탄생 배경

- 1994년 통계학자 **Leo Breiman**이 이 개념을 제안
  - 머신러닝 모델, 특히 불안정한 모델(예: 결정 트리)이 데이터 변화에 민감하다는 점에 주목
- 그래서 그는 무작위로 추출된 여러 데이터 샘플로 각각 모델을 학습시키고,
  - 그 예측 결과를 모아서 성능을 향상시키자는 아이디어

배깅

주어진 자료에 여러개의 부스트랩 자료를 생성하고 각 부스트랩 자료에 예측모형을 만든 후 결합하여 최종 예측모형을 만드는 방법

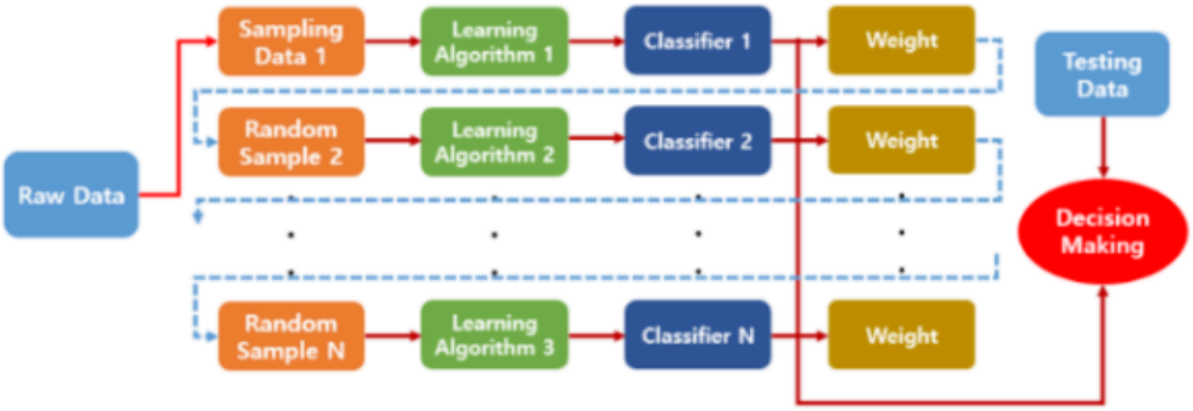
- 부스트랩(bootstrap): 주어진 자료에서 동일한 크기의 표본을 랜덤 복원추출로 뽑은 자료를 의미
- 보팅(voting): 여러 개의 모형으로부터 산출된 결과를 다수결에 의해서 최종결과를 선정하는 과정
- 최적의 의사결정나무를 구축할 때 가장 어려운 부분이 가지치기이지만 배깅에서는 가지치기 하지 않고 최대로 성장한 의사결정나무를 활용
- 훈련자료로 모집단으로 생각하고 평균예측모형을 구하여 분산을 줄이고 예측력을 향상



출처 https://blog.naver.com/boundgreen/223106895369

부스팅

- 예측력이 약한 모형들을 결합하여 강한 예측모형을 만드는 방법
  - 부스팅(Boosting)은 여러 약한 학습기(weak learner)를 **순차적으로 학습시키면서** 각 모델이 앞선 모델의 오류를 보완하도록 만드는 **앙상블 학습 기법**
  - 단순한 모델들을 조합해 강한 모델(strong learner)을 만드는 방법
  - ‘boost’는 말 그대로 **성능을 끌어올린다**는 의미에서 나온 말



부스팅의 정의와 어원

Boosting의 어원

- *영어 "boost"**는 '밀어 올리다, 향상시키다'는 뜻
- 그래서 boosting은 **약한 예측 성능을 가진 모델의 성능을 끌어올리는 과정**을 의미
- 통계학/기계학습에서 약한 학습기(예: 작은 결정 트리)를 여러 개 조합해 강한 학습기를 만드는 기법

핵심 개념

- 단일 모델이 잘 못하는 부분을 **다음 모델이 보완하도록 설계**
- 이 과정을 반복하면서 점점 **더 정교한 예측 모델**을 구성
- 최종 예측은 여러 모델의 예측을 **가중합(weighted sum)** 또는 **가중 투표(weighted vote)** 방식으로 결합

부스팅의 동작 방식

단계적 학습(Sequential Learning)

- 첫 번째 약한 학습기는 전체 데이터를 학습
- 이후 모델들은 이전 모델이 틀린 샘플에 **더 많은 가중치**를 부여하여 학습
- 반복하면서 **에러가 줄어드는 방향으로 학습**을 강화

최종 예측

- 각 모델의 예측에 가중치(weight)를 부여해서 **가중합 또는 투표로 결정**
- 예: 에이다부스트(AdaBoost)는 모델 성능에 따라 가중치를 조절해 결합

주요 부스팅 알고리즘

AdaBoost (Adaptive Boosting)

- 가장 대표적이고 고전적인 방식.
- 잘못 예측한 샘플의 가중치를 높여 다음 모델이 집중하도록 함

이진분류 문제에서 랜덤분류기보다 조금 더 좋은 분류기 n개에 각각 가중치를 설정하고 n개의 분류기를 결합하여 최종 분류기를 만드는 방법(단, 가중치의 합은 1)

- 훈련 오차가 빨리 그리고 쉽게 줄일 수 있다
- 배깅에 비해 많은 경우 예측오차가 향상되어 Adaboost의 성능 배깅보다 뛰어난 경우가 많다.

Gradient Boosting

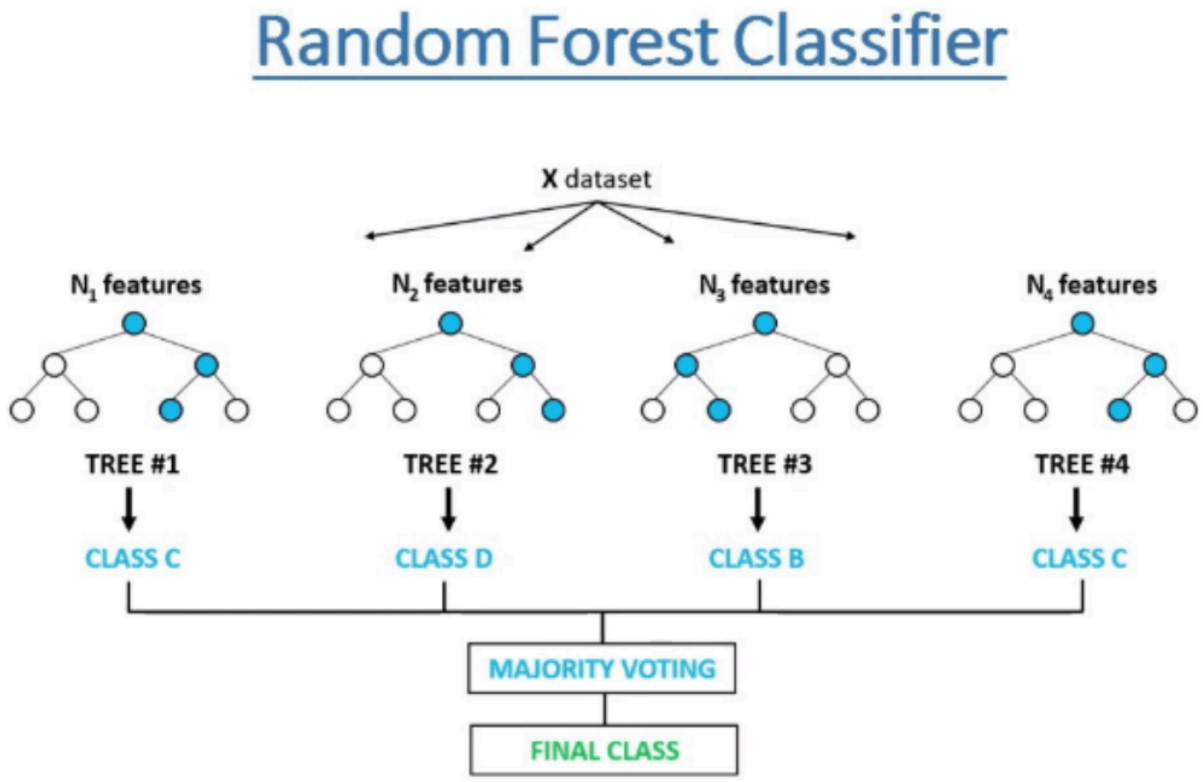
- 잔차(residual)를 줄이도록 모델을 계속 학습시킴
- **XGBoost**, **LightGBM**, **CatBoost** 같은 실전 알고리즘이 여기에서 파생
- 성능 좋고 Kaggle 대회 1등 먹은 방법이 이 계열

랜덤포레스트(random forest)

랜덤포레스트는 배깅에 랜덤 과정을 추가한 방법

- 원 자료로부터 부트스트랩 샘플을 추출하고, 각 부트스트랩 샘플에 대해 트리를 형성해 나가는 과정은 배깅과 유사하나,
- 각 노드마다 모든 예측변수 안에서 최적의 분할(split)을 선택하는 방법 대신
    - 예측 변수들을 임의로 추출하고, 추출된 변수 내에서 최적의 분할을 만들어 나가는 방법을 사용
- 새로운 자료에 대한 예측은 분류(classification)의 경우는 다수결(majority votes)로,

- 회귀(regression)의 경우에는 평균을 취하는 방법을 사용하며, 이는 다른 앙상블 모형과 동일하다.



랜덤 포레스트 특징

- 의사결정나무의 특징인 분산이 크다는 점을 고려하여 배깅과 부스팅보다 더 많은 무작위성을 주어 약한 학습기들을 생성한 후 이를 선형결합하여 최종 학습기를 만드는 방법
- randomForest 패키지는 random input에 따른 forest of tree를 이용한 분류방법
- 랜덤한 forest에 많은 트리들이 생성됨
- 수천개의 변수를 통해 변수끼리 없이 상관없이 실행하므로 정확도 측면에서 좋은 성과를 보임
- 이론적 설명이나 최종 결과에 대한 해석이 어렵다는 단점이 있지만 예측력이 매우 높음
- 특히 입력변수가 많은 경우 배깅과 부스팅과 비슷하거나 좋은 예측력을 보인다.

정리

✅ 요약: 앙상블 모형(Ensemble Model)은 여러 개의 모델을 조합하여 예측 성능을 높이는 기법이야.

기본 아이디어는 "여러 모델을 합치면 더 강력한 하나가 된다"는 거고, 대표적인 방법으로는 배깅(Bagging), 부스팅(Boosting), 스태킹(Stacking)이 있음

일반적으로 과적합을 줄이면서 성능을 높이는 데 효과적

그래서 머신러닝 대회나 실제 산업 응용에서도 자주 써먹히는 비장의 무기

앙상블 모형의 개요

앙상블의 핵심 개념

- 여러 개의 모델을 독립적으로 혹은 계층적으로 훈련시켜 하나의 최종 결과를 도출
- "다수결", "평균", "가중 평균" 등의 방법으로 예측을 통합
- 개별 모델의 약점을 보완하고, 분산과 편향 사이의 균형을 맞추는 데 유리

왜 앙상블을 쓸까?

- **일관된 예측 성능**: 하나의 모델이 잘못 판단하더라도 다른 모델이 이를 보완
- **과적합 방지**: 특히 결정트리 기반 모델에서 효과적
- **성능 향상**: 개별 모델보다 일반화 능력이 뛰어남

대표적인 앙상블 기법

1. 배깅 (Bagging)

- Bootstrap Aggregating의 줄임말.
- 여러 모델을 서로 다른 데이터 샘플(중복 허용된 샘플)로 학습시켜 평균 내거나 다수결 투표
- 예: 랜덤 포레스트(Random Forest).
- 장점: 분산 감소, 과적합 감소.

2. 부스팅 (Boosting)

- 이전 모델이 틀린 부분에 가중치를 부여해 다음 모델이 더 잘 학습하도록 만듦.
- 약한 모델을 모아 강한 모델로 만드는 방식.
- 예: AdaBoost, Gradient Boosting, XGBoost, LightGBM.
- 장점: 편향 감소, 성능 우수하지만 과적합 주의 필요.

3. 스태킹 (Stacking)

- 서로 다른 유형의 모델들의 예측 결과를 또 다른 메타 모델이 입력으로 사용.
- 예: 로지스틱 회귀, 선형 회귀, 딥러닝 등을 메타 모델로 사용 가능.
- 장점: 다양한 모델의 장점을 조합함.

스태킹은 **2단계 구조**로 이루어져 있습니다:

1. **기본 모델들** (Base Models):
    - 여러 개의 **다양한 모델**을 사용하여 예측을 수행
        - 예를 들어, 결정 트리, 로지스틱 회귀, 서포트 벡터 머신(SVM), KNN 등의 모델을 사용
    - 각 모델은 독립적으로 데이터를 학습하고 예측
2. **메타 모델(Meta-Model)**:
    - 기본 모델들이 생성한 예측값을 **입력값**으로 받아, 최종 예측을 **조합하는 모델**
    - 메타 모델은 기본 모델들의 예측을 바탕으로 **최종 예측을** 수행
        - 보통 **로지스틱 회귀**나 **선형 회귀** 등의 간단한 모델을 사용

실습

배깅

```
> # 배깅 모델 생성
> iris.bagging ← bagging(Species ~ ., data = iris, mfinal = 10)
> # 10번째 트리 시각화
```

```
> plot(iris.bagging$trees[[10]])
> text(iris.bagging$trees[[10]])
> pred ← predict(iris.bagging, newdata=iris)
> # pred
> tb ← table(pred$class, iris[,5])
> tb

           setosa versicolor virginica
setosa         50         0         0
versicolor      0        49         5
virginica       0         1        45
```

요약:

위 코드는 `R`의 `adabag` 패키지를 이용하여 **배깅(Bagging)** 모델을 학습하고, `iris` 데이터셋 전체에 대해 예측을 수행한 뒤 결과를 분석하는 과정을 보여줘. 결과적으로는 **versicolor와 virginica 사이에 약간의 혼동**이 있었고, `setosa`는 완벽하게 분류됨. 각 약한 학습기는 결정 트리이며, 마지막(10번째) 트리를 시각화한 것도 포함되어 있어.

배깅 모델 학습 및 트리 시각화

모델 학습

```
iris.bagging ← bagging(Species ~ ., data = iris, mfinal = 10)
```

- `Species ~ .`: 종속 변수는 Species, 독립 변수는 나머지 전부.
- `mfinal = 10`: 총 10개의 결정 트리(약한 학습기)를 학습해서 앙상블 구성.
- 이때 각 트리는 데이터셋을 복원 추출(bootstrap sampling)해서 학습함.

트리 시각화

```
plot(iris.bagging$trees[[10]])
text(iris.bagging$trees[[10]])
```

- 10번째 학습된 결정 트리를 시각적으로 보여줌.
- `text()`는 각 분기점의 조건과 리프 노드의 클래스를 라벨링해서 표시함.

- 이를 통해 각 트리가 어떤 변수와 조건으로 분류하는지 파악 가능함.

예측 및 혼동 행렬 확인

예측 수행

```
pred ← predict(iris.bagging, newdata=iris)
```

- 훈련 데이터 `iris`에 대해 다시 예측.
- 예측 결과는 `pred$class`에 저장됨.

혼동 행렬 확인

```
tb ← table(pred$class, iris[,5])
```

결과:

```
           setosa versicolor virginica
setosa         50          0         0
versicolor      0         49         5
virginica       0          1        45
```

해석:

- **setosa**는 완벽하게 50개 모두 올바르게 예측됨.
- **versicolor** 중 50개 중 49개만 맞췄고, 1개는 virginica로 잘못 예측됨.
- **virginica** 중 50개 중 45개만 맞췄고, 5개는 versicolor로 오분류됨.
- 가장 큰 혼동은 virginica ↔ versicolor 사이에서 발생했음.

모델 성능 요약

정확도 및 혼동 분석

- 총 샘플 수: 150개

- 올바른 예측: 50(setosa) + 49(versicolor) + 45(virginica) = **144**
- 예측 오류 수: 6개
- 오류율:{Error Rate} = {6}/{150} = 0.04 (4%)

특징

- 배깅 모델은 기본적으로 분산을 줄이는 데 효과적임.
- 다만, **클래스 간 경계가 모호한 경우**(versicolor vs virginica)에서는 혼동 가능.
- 단일 결정 트리보다는 성능이 좋아도, boosting만큼의 미세 조정은 부족할 수 있음.

부스팅

```
> # 부스팅
> # boosting 모델 생성
> boo.adabag ← boosting(Species ~ ., data = iris, boos = TRUE, mfinal = 1
0)
> boo.adabag$importance
Petal.Length  Petal.Width Sepal.Length  Sepal.Width
  65.508511    19.942490     4.773342     9.775657
> pred ← predict(boo.adabag, newdata=iris)
> tb ← table(pred$class, iris[,5])
> tb

            setosa versicolor virginica
  setosa         50          0         0
  versicolor      0         50         0
  virginica       0          0        50
> error.rpart ← 1 - ( sum(diag(tb)) / sum(tb) )
> error.rpart
[1] 0
```

요약:

위 코드는 `R`의 `adabag` 패키지를 사용하여 **AdaBoost 기반 분류기**를 학습하고, **iris 데이터셋 전체를 다시 예측**해 정확도를 확인하는 과정이야. 결과적으로 완벽한 분류(오류율 0)를 달성

- 중요 변수는 `Petal.Length`가 가장 높게 나왔고
- `boosting()` 함수로 만든 모델의 예측 성능이 매우 우수함을 보여주는 예제

부스팅 모델 학습 및 중요 변수 추출

`boosting()` 함수로 AdaBoost 모델 생성

```
boo.adabag ← boosting(Species ~ ., data = iris, boos = TRUE, mfinal = 10)
```

- `Species ~ .` : 종속 변수는 Species, 설명 변수는 나머지 전부.
- `boos = TRUE` : 샘플의 가중치(weight)를 부여하는 boosting 수행.
- `mfinal = 10` : 최종 모델은 약한 학습기 10개로 구성.

변수 중요도 확인

```
boo.adabag$importance
```

결과:

```
Petal.Length  65.5
Petal.Width   19.9
Sepal.Width    9.8
Sepal.Length   4.8
```

- `Petal.Length`가 **가장 높은 중요도**를 가지며 모델에 가장 큰 기여를 함.
- 이건 실제로 iris 데이터셋에서 꽃 품종을 구분하는 핵심 속성

모델 예측 및 정확도 평가

예측 수행

```
pred ← predict(boo.adabag, newdata=iris)
```

- 훈련 데이터(iris)를 그대로 사용하여 다시 예측함.
- **일종의 self-test**라 과적합 가능성 있음.

혼동 행렬(Confusion Matrix)

```
tb ← table(pred$class, iris[,5])
```

결과:

```
            setosa versicolor virginica
setosa         50         0         0
versicolor      0        50         0
virginica       0         0        50
```

- 각 클래스별로 **정확하게 50개씩** 예측함.
- 따라서 모델이 완벽하게 훈련셋을 분류했다는 뜻이야.

정확도 계산

```
error.rpart ← 1 - ( sum(diag(tb)) / sum(tb) )
```

- `diag(tb)`는 정답인 셀(대각선)의 합 = 150
- 전체 데이터도 150이므로 오류율은 `0`

종료