

머신러닝(Machine Learning)

앙상블(ensemble) 기법

목 차

01 앙상블(ensemble) 기법

02 앙상블(ensemble) 기법 - 랜덤 포레스트(Random Forest)

03 앙상블(ensemble) 기법 - Gradient Boosting 기법

04 앙상블(ensemble) 기법 - 두 모델의 비교

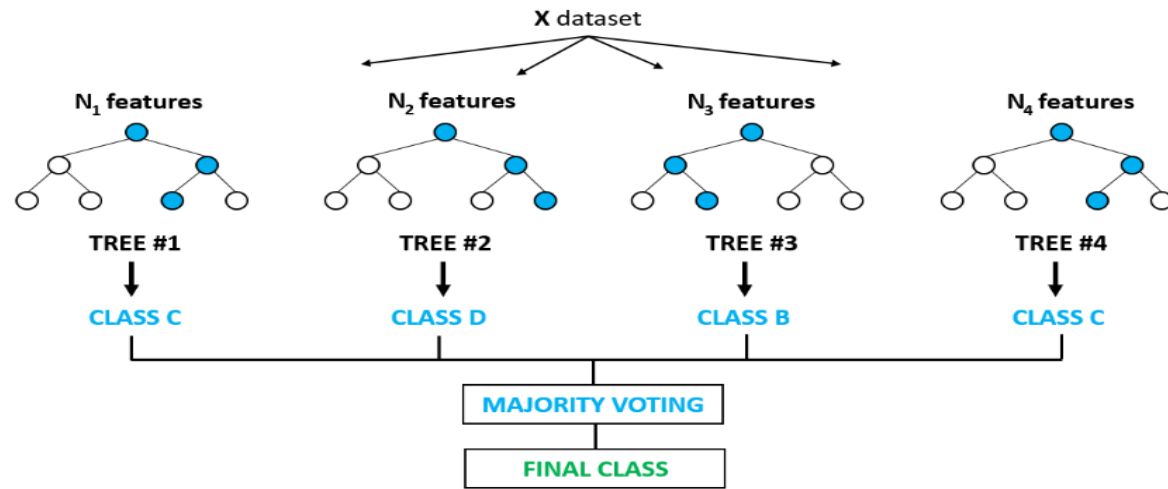
05 여러가지 모델

01 앙상블 기법

- ▶ 앙상블(ensemble)는 여러 머신러닝 모델을 연결하여 더 강력한 모델을 만드는 기법
- ▶ 랜덤 포레스트(Random Forest)와 그래디언트 부스팅(gradient boosting)
 - => 둘 다 모델을 구성하는 기본 요소로 결정 트리를 사용.

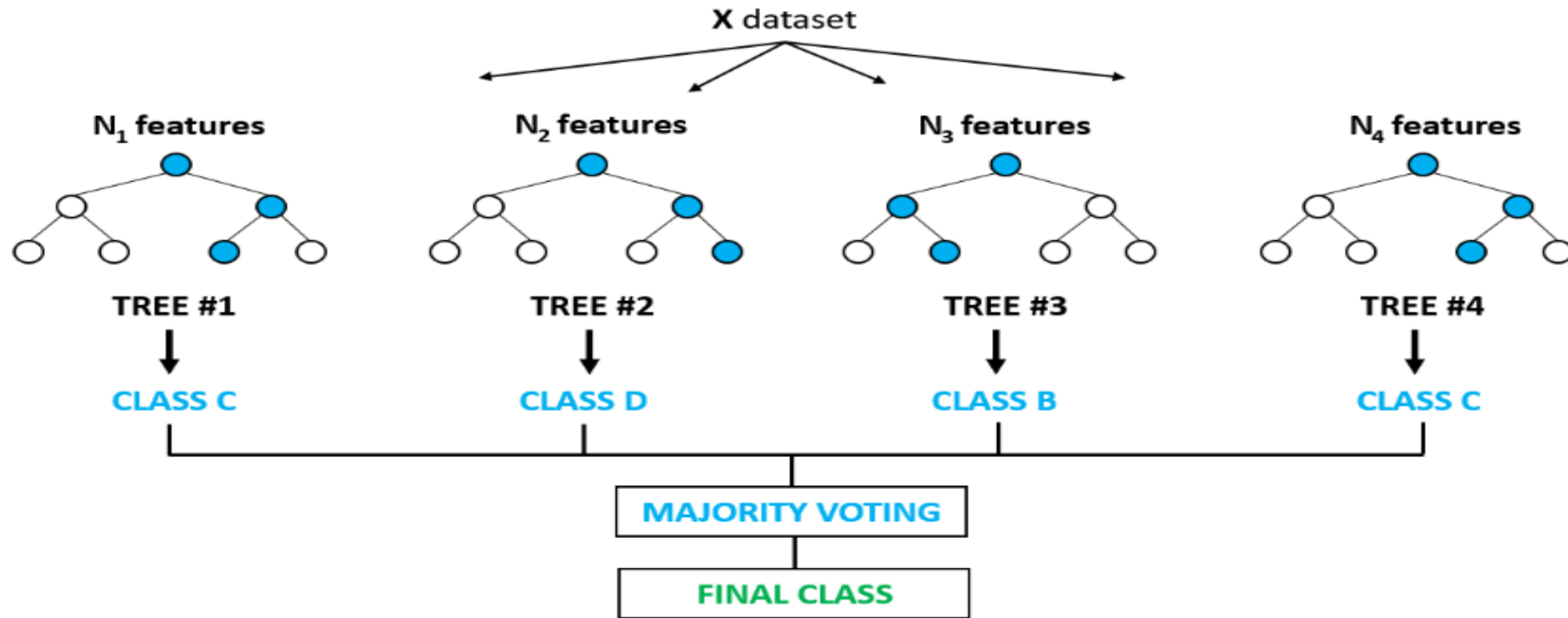
02 앙상블 기법-랜덤 포레스트

- ▶ 결정 트리의 주요 단점 - 학습용 데이터에 **과대 적합**되는 경향이 있음.
 - A. 랜덤 포레스트(Random Forest)는 이 문제를 회피할 수 있는 방법.
- ▶ 아이디어 : 조금씩 다른 여러 결정 트리의 묶음.



하나 하나의 트리는 데이터의 일부에 과대적합을 하는 경향이 있다.

02 앙상블 기법-랜덤 포레스트



<http://bitly.kr/s2GOAgZm0> 참조

- 같은 데이터 셋을 활용하여 다수의 트리를 만든다.
- 각 트리별로 예측값을 예측한다. 이 중에 가장 많이 예측된 값으로 최종 예측으로 한다.(분류)
- 학습된 트리별로 예측값을 예측한다. 트리별로 예측의 평균값을 최종 예측으로 한다.(회귀)

02 앙상블 기법-랜덤 포레스트

▶ 결정 트리의 주요 원리

(A) 약간씩 서로 다른 데이터에 대해서 과대 적합되는 트리를 많이 만들어 평균을 내면 과대적합을 줄어든다.

(B) 이에 대한 결과가 수학적으로 증명되었다.

(1) 타깃 예측을 잘 해야 함.

(2) 과대적합이 아닌 일반화된 모델을 위해 다른 트리와 조금씩 달라야 한다.

=> A. 데이터 포인트를 무작위로 선택

=> B. 분할 테스트에서 feature(특성)을 무작위로 선택

02 앙상블 기법-랜덤 포레스트

▶ 랜덤 포레스트 모델 구축하는 단계

(A) 생성할 트리의 개수를 정한다.

(B) 샘플을 생성한다.(부트스트랩 샘플-bootstrap sample)

* n개의 데이터 포인트 중에서 n번의 횟수만큼 반복 추출

* 어떤 데이터 포인트는 누락될 수 있고, 어떤 데이터포인트는 **중복**될 수 있다.

=> 부트스트랩 샘플링은 랜덤 포레스트의 모든 트리가 서로 달라지도록 만든다.

(C) 각 노드에서 무작위로 특성(feature)을 선택 후, 이 후보들 중에서 최선의 테스트를 찾는다.

⇒ 관련 매개변수 : max_features

⇒ max_features 값을 크게 하면 랜덤 포레스트의 트리들은 매우 비슷해지고 가장 두드러진 특성을 이용해 데이터에 잘 맞춰질 것이다. 작게 하면 랜덤 포레스트 트리들은 많이 달라지게 된다.

02 앙상블 기법-랜덤 포레스트

▶ 회귀의 랜덤 포레스트 만들기

(A) 각각의 트리가 예측한 값들의 평균을 최종 예측으로 한다.

▶ 분류의 랜덤 포레스트 만들기

(A) 각 알고리즘이 가능성 있는 출력 레이블의 확률 또는 출력 레이블을 제공. 예측한 레이블의 확률을 평균을 내어 가장 높은 확률 또는 가장 많이 예측 값을 갖는 클래스가 예측 값이 된다.

02 앙상블 기법-랜덤 포레스트

▶ 장점

- (A) 단일 모델의 하나의 트리보다 덜 과대 적합 되며, 훨씬 좋은 결정 경계를 만들어준다.
- (B) 성능이 매우 뛰어나고 매개변수 튜닝을 많이 하지 않아도 잘 작동한다.
- (C) 데이터의 스케일(값의 범위)을 맞출 필요가 없다.

▶ 단점

- (A) 매우 차원이 높고 희소한 데이터에서 잘 작동하지 않음.
- (B) 선형 모델보다 많은 메모리를 사용하며 학습과 예측이 오래 걸림.

02 앙상블 기법-랜덤 포레스트 - 매개변수

▶ n_estimators

- 랜덤 포레스트가 생성할 트리의 개수

▶ max_features

- 각 트리가 사용하는 특징(변수)의 개수.
- 각 트리가 얼마나 무작위가 될지를 결정. 일반적으로 기본값이 좋은 결정이 될 수 있다.
- 분류는 $\text{max_features}=\text{sqrt}(\text{n_features})$ 이고, 회귀는 $\text{max_features}=\text{n_features}$

▶ n_jobs

- 컴퓨터에서 사용할 코어 수의 지정이 가능. -1일 경우 컴퓨터의 모든 코어를 사용.

02 앙상블 기법-랜덤 포레스트 - 매개변수

▶ random_state

- 데이터를 뽑을 때 사용하는 패턴 지정. 이를 통해 동일한 데이터 샘플을 만들 수 있다.

03 앙상블 기법 - Gradient Boosting 기법

- ▶ 여러 개의 결정 트리를 묶어 강력한 모델을 만든다.
- ▶ 분류(Classification)과 회귀(Regression)에 모두 사용 가능.
- ▶ 랜덤 포레스트(random forest)와 달리 **이전 트리의 오차 보완하는 방식**
- ▶ 그래디언트 부스팅 회귀 트리는 무작위성이 없다. 대신 강력한 사전 가지치기 사용.
- ▶ 트리가 많을 수록 성능이 좋아짐.
- ▶ 랜덤 포레스트보다 매개 변수 설정에 더 민감하며, 잘 조정하면 높은 정확도를 얻음.

03 앙상블 기법 - Gradient Boosting 기법

(A) 트리의 깊이가 5정도 깊지 않은 트리를 사용하며 메모리 사용이 적고 예측이 빠름.

이런 얇은 트리와 같은 간단한 모델을 약한 학습기(weak learner)이라 한다.

(B) 메모리 사용이 적고 예측이 빠르다.

(C) 머신러닝 데이터 분석 대회에서 우승을 많이 차지하였고, 업계에서도 널리 사용.

(D) 이전 트리 오차를 얼마나 강하게 보정할 것인가를 제어하는 파라미터(learning_rate)

* 학습률이 크면 클수록 트리는 오차 보정을 강하게 하므로 복잡한 모델이 만들어진다.

(E) 손실함수를 정의하고, 경사 하강법(gradient descent)를 사용하여 다음 추가될 트리의 예측값을 보정해 간다.

03 앙상블 기법 - Gradient Boosting 기법

▶ 장점

- (A) feature 의 스케일을 조정하지 않아도 된다.
- (B) 이진 특성이 연속적인 특성에서도 잘 동작한다.

▶ 단점

- (A) 매개 변수를 잘 조정해야 한다.
- (B) 학습 시간이 길다.
- (C) 희소한 고차원 데이터에서 잘 작동하지 않는다.

03 앙상블 기법 - Gradient Boosting 기법

- ▶ learning_rate : 이전트리의 오차 보정 정도
- ▶ n_estimator : 트리의 모델 수
 - (A) n_estimator : 크면 클수록 좋음(랜덤 포레스트)
 - (B) n_estimator : 과적합의 가능성(그래디언트 부스팅)
- ▶ max_depth : 트리 모델의 복잡도. 트리의 최대 깊이를 지정.
 - (A) max_depth를 매우 작게 설정하며 트리의 깊이가 5보다 깊어지지 않도록 한다.
- ▶ 매개변수 튜닝 : n_estimators을 맞춘 이후에 learning_rate를 찾음.

04 앙상블 기법 - 두 모델의 비교

▶ 그래디언트 부스팅 vs 랜덤 포레스트

A. 보통은 일반적으로 매개변수 설정을 하지 않아도 되는 랜덤 포레스트가 안정적이다.

B. 예측 시간이 중요하거나 머신러닝 모델에서 마지막 성능까지 쥐어짜야 할 때 그래디언트 부스팅을 사용하면 도움이 된다.

05 여러가지 모델

▶ KNN(최근접 이웃)

작은 데이터 셋, 기본 모델로서 좋고 설명하기 쉽다.

▶ 선형 모델

대용량 데이터 셋 가능. 대용량 데이터 셋 가능. 고차원 데이터에 가능

▶ 나이브 베이즈(naive bayes)

분류만 가능. 선형모델보다 훨씬 빠름. 선형 모델보다 덜 정확함.

▶ 결정 트리

매우 빠르고, 데이터 스케일 조정이 필요 없음. 시각화하기 좋고, 설명하기 쉬움.

05 여러가지 모델

▶ 랜덤 포레스트

- A. 결정 트리 하나보다 거의 항상 좋은 성능을 냄. 매우 안정적이고 강력.
- B. 데이터 스케일 조정 필요 없음. **고차원 희소 데이터에 잘 안 맞음.**

▶ 그래디언트 부스팅 결정 트리

- A. 랜덤 포레스트보다 조금 더 성능이 좋음.
- B. 랜덤 포레스트보다 학습은 느리나 예측은 빠름. 메모리를 조금 사용.
- C. 매개변수 튜닝이 많이 필요.

▶ 신경망

- 특별히 대용량 데이터셋에서 매우 복잡한 모델을 만들 수 있음. 매개변수 선택과 데이터 스케일에 민감.
- 큰 모델 학습이 오래 걸림