Decision Trees

* 1. Definition(정의)
* 의사결정나무(Decision Tree)는 입력 벡터 X가 주어졌을 때, 그게 어떤 입력 영역의 k개의 구역 {R₁, R₂, ..., Rₖ}에 속하는지를 보고, 그 구역에 대응되는 예측값 {w₁, w₂, ..., wₖ}을 반환하는 단계적 매핑(mapping)을 수행하는 것으로 생각할 수 있다. 이 구역은 입력 전체 공간을 겹치지 않게 나누는 분할(partition)이어야 하며, 이는 어떤 두 구역도 겹치지 않고, 모든 구역의 합집합이 전체 입력 영역을 완전히 덮는다는 것을 의미한다.

= 쉽게 말해서 의사결정나무는 입력 공간을 겹치지 않게 k개로 나누고(R₁~Rₖ), 각 구역에 고정된 예측값 wₖ을 대응시키는 함수임

+ 분할된 구역을 합치면 전체 공간을 다시 복원 가능

+ 각 구역에 속한 입력은 항상 동일한 예측값을 가짐

* 가장 많은 정보를 얻을 수 있는 쪽으로 질문을 던짐

1. 결정 트리를 회귀(regression)에 사용할 때 :

폰트, 텍스트, 화이트, 타이포그래피이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

* 구역 Rⱼ에 속한 데이터들의 예측값 -> 그 구역에 속한 데이터(label)들의 평균값을 예측값으로 사용함

1. 결정 트리를 분류(classification)로 사용하는 경우

* 예측값 wⱼ는 해당 구역에 속한 학습 데이터 중 가장 많이 등장한 클래스 값으로 설정함
* 분류를 정답 클래스가 뭐인지 맞히는 문제니까 그 구역 안에서 가장 많이 나온 클래스를 고르는게 최소 오차 전략이 됨
* 이런 매핑 구조는 트리 형태로 쉽게 표현 가능함

1. DT의 특징

* DT는 질문을 계속하면서 데이터를 split 해나가는 구조임
* 이때 질문을 하는 지점을 node(노드)라고 함

\*Node

- internal node (내부 노드) : 조건을 걸고 split하는 지점

- leaf node (리프 노드) : DT에서 예측값이 나오는 마지막 지점 = 더 이상 split X

-> Tree를 따라 내려가다보면 더 이상 쪼갤 필요가 없거나, 샘플 수가 적거나, 깊이 제한에 도달하면 거기서 멈추고 예측값을 지정함 = leaf = tree의 끝

* 1. Model fitting (모델 학습)
* 모델을 학습하기 위해서는 loss function을 최소화하려고 함

폰트, 텍스트, 화이트, 라인이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

* Loss function을 나타낸 것으로 tree가 데이터를 나눈 구역 R1, R2,…각각에 대해 해당 구역에 속한 모든 데이터에 대해 그 구역에 할당된 예측값 w와의 loss(차이)를 구해서 구역별로 합치고, 다시 전체 구역에 대해 더한 것을 표현한 것임
* Tree model은 입력 x가 정확히 하나의 구역 R에 속하고, 이 구역에는 고정된 예측값 w가 있기 때문에 모델의 예측값인 f(x)를 w로 수식을 바꿔쓸 수 있는거임
* DT는 입력 공간을 불연속적으로 쪼개기 때문에 예측값 또한 부드럽게 변하지 않고 손실함수도 불연속이 되버림 -> 미분을 이용한 최적화 방법을 사용할 수 없음
* Greedy 알고리즘 사용

= 한번에 가장 좋은 선택을 하면서 쭉 진행하는 방식으로 지금 당장 가장 좋은 split을 반복하면서 tree를 만듬

* 1. Regularization(정규화)
* Greedy 알고리즘을 무한히 실행시켜 매우 깊은 tree는 내부적으로 일관된 데이터셋에 대해서도 완벽하게 fit할 수 있음 = loss가 0이 되는 tree를 만들 수 있음
* 하지만 이렇게 깊은 트리를 사용하면 여러 문제가 발생할 수 있음

1. 학습 데이터에 대한 overfitting(과적합)
2. 지나치게 복잡한 모델 (너무 많은 node와 분할)

* 정규화를 위해 여러 제한을 둠

1. 트리의 최대 깊이 제한
2. 전체 leaf node 수를 제한
3. 각 노드의 샘플 수가 일정 이하일 경우 split 금지
4. Split grain(분할 이득?) 이 임계값 이하일 경우 split 금지

* 또 다른 정규화 방법은 loss = 0인 tree를 먼저 완성한 다음 pruning(가지치기)를 하는 것이다.

1.5.1 DT의 장점

* outlier(이상치)에 강함
* 해석 가능함
* 조건문 형태로 설명이 가능하니까 모델이 왜 이 예측을 했는지 사람이 따라갈 수 있음
* 연속형 및 이산형 특성 모두에 잘 작동함
* 입력 특성에 대한 monotonic transformation(단조 변환)에 강함

: 입력값의 순서는 유지되지만, 값 자체는 변하는 함수

* DT는 숫자 값 자체보단 비교 결과(크거나 작음)를 중요하게 보기 때문에 영향 X
* 대규모 데이터 셋에서도 비교적 빠르게 적절한 성능을 낼 수 있음

1.5.2 DT의 단점

* 정규화를 하더라도 일반화 성능이 약할 수 있음
* Split 경계 근처에서 모델이 매우 불안정함
* 작은 변화에도 민감

2. 앙상블 학습을 통한 개선 (Ensemble Learning)

* 앙상블 학습은 여러 모델의 예측을 평균 또는 집계하여 모델의 불안정성을 줄이는 방법
* 일반적으로 분산을 줄이면서 bias를 거의 증가시키지 않음
* 전체 성능 향상에 효과적임
* 더 많은 모델을 학습하고 예측해야하므로 계산 비용이 증가하는 단점이 有

2.1 Bagging

* Bootstrap Aggregating의 줄임말
* Bootstrapping으로 서로 다른 학습용 데이터셋을 만든 후, 각 데이터 셋에 대해 개별 모델을 학습시키고, 최종적으로 모든 모델의 예측을 통합해서(평균내거나-> 회귀, 다수결을 통해-> 분류) 앙상블 예측을 생성함
* 학습 알고리즘이 결정적일 경우, 모든 모델이 동일한 결과를 내게 되어 앙상블의 효과가 X
* bootstrapping같은 복원 샘플링을 사용하면 다양성이 확보되며 이 다양성은 모델의 분산을 낮추는데 도움이 된다.
* Low-variance(고편향)이면 효과가 작고, DT처럼 high-variance(고분산) 모델일수록 bagging 의 효과가 크다

\*\*결정적 (deterministic)

- 입력이 같으면 항상 결과가 같은 것

- ex) 전통적인 DT, KNN, 선형 회귀

\*\*비결정적 (stochastic)

* 학습 과정에 randomness가 포함되어 있어서 같은 입력이라도 결과가 달라질 수 있는
* Ex) RF, neural network, SGD

2.1.1 Random Forests (랜덤 포레스트)

* DT를 기반으로 한 앙상블 모델이며, bagging보다 더 많은 randomness를 도입하여 개별 모델 간의 상관성을 줄임
* RF는 Bagging과 동일하게 각 tree를 서로 다른 bootstrap sample로 학습시키지만, tree split 시 고려할 feature(특성=factor)의 일부만 무작위로 선택함 (feature의 제곱근 수만큼)
* 이러한 추가적인 randomness로 인해 트리들간의 다양성이 증가하고, 과적합을 줄이면서 성능을 개선시킬 수 있음

2.2 Boosting (부스팅)

* Boosting은 Bagging과 달리 모델을 순차적으로 학습시키는 방식
* 다음 모델은 이전 모델들의 예측 성능을 보완하도록 학습됨
* 원래 부스팅은 이진 분류를 위해 개발됨
* 부스팅은 기본적으로 약한 learner를 결합하여 강한 learner를 만드는 접근법임
* 다음 모델로 갈 때마다 약한 학습기에 가중치를 부여하여 예측
* 부스팅은 개별 모델이 복잡하지 않더라도, 이들을 순차적으로 조합하면서 복잡한 예측이 가능
* Bias(편향)을 줄이는 데 효과적이며 bagging이나 RF처럼 분산을 줄이는 방식과는 다름

\*Forward stagewise additive modeling (전방 단계적 가산 모델링?)

* Gradient Boosting의 기반이 됨

2.2.2 Gradient Boosting

* Gradient Boosting은 부스팅 개념을 nonparametric(비매개변수적)모델, 특히 DT에 확장할 수 있도록 한 기법임
* 매 단계마다 이전 앙상블 모델이 잘못 예측한 부분에 집중하는 새로운 모델을 추가하는 것이 포인트 -> 새로운 학습기는 loss function의 gradient를 학습하여 약간 모델이 할당되어야 할 방향을 알려줌

= 현재 모델에 새로운 함수 F를 더했을 때 loss를 최소화하도록 F를 학습시킴

* 이 최적화는 일반적으로 functional gradient descent로 근사함

\*Gradient Boosting에서 사용되는 함수 F는 일반적으로 깊이가 얕고 강하게 정규화된 DT이며, 이렇게 빠르게 학습할 수 있는 약한 모델을 순차적으로 조합함으로써 강력한 앙상블을 구성하게 됨

2.2.3 XGBoost

* Gradient Tree Boosting을 기반으로 함,
* Gradient Boosting 에서 몇가지 개선점을 추가함

1. Loss function에 정규화 항 포함
2. 무작위 특성 선택

* 각 노드 분할 시, bagging 이나 RF처럼 전체 특성 중 일부만 선택해 사용
* 트리 간 다양성을 높여 일반화 성능 향상

1. 2차 근사 사용

* Gadient Boosting은 1차 미분만 사용하지만, XGBoost는 2차 미분까지 사용하여 더 정교하게 최적화를 수행
* 빠르면서도 정확한 성능을 보이며, 대규모 데이터와 복잡한 문제에서도 자주 사용됨