Decision Tree

의사결정 나무 기본 개념

> 정의
>
> : 의사결정 규칙을 나무 구조로 나타내어 전체 자료를 몇개의 소집단으로 분류하거나 예측하는 분석 방법

> 종류

- 1. ID3(Iterative Dichotomiser) : 반복적 이분법 --> 고전적
  - entropy : p(x)log_2P(x)
    - 사건이 일어나냐, 일어나지 않느냐의 확률값으로 이야기함
    - 이산형(카데고리성) 데이터에 적합
- 2. CART(Class and Regression Tree)
  - jini index : 1-p(x)^2
    - overfitting --> 정지기준 (depth, max_leaf_nodes) 설정하기

> 분류 방법

- 주로 분류에서 사용하지만 회귀에서도 지원
- 분류 : 들어와있는 데이터들 중 개수가 많은 쪽으로 구분
- 회귀 : 들어와있는 데이터의 평균값으로 데이터를 구분

> 관련 용어

- 가지의 끝 = leaf node
- 깊이 = depth --> depth 당 피쳐 하나의 분기점이된다
  - 데이터를 완벽히 거르기위해 depth 끝까지 가게되면 overfitting --> 완벽히 분리되지 않더라도 depth 중간까지만 분류

> 구성요소

- root node : 시작 노드 , 전체 자료 포함
- parent node : 주어진 마디의 상위에 있는 노드
- child node : 하나의 마디로부터 분리되어 나간 2개 이상의 노드들

- leaf node(terminal node) : 끝 노드, 자식 마디가 없는 노드
- internal node : 부모 노드와 자식 노드가 모두 있는 마디
- depth : root부터 leaf 노드까지의 중간 노드들의 수
  - depth 당 피쳐 하나의 분기점이된다
  - 데이터를 완벽히 거르기위해 depth 끝까지 가게되면 overfitting
  - 완벽히 분리되지 않더라도 depth 중간까지만 분류하는 것이 좋다

> 활용

- root node는 어떤 피쳐로 잡아야하는가?
  - 지니, entropy 사용
- 가지치기는 어느정도가 적당한가?
  - 하이퍼파라미터 튜닝

사용 분류 기준

Jini(지니) index

> 정의 : 순수한 정도(pure) --> 회귀에서 점수가 높게 나옴

$$Gini = 1 - \sum_{i=1}^{C} (p_i)^2$$

> 원리

- 지니계수가 높다 = 순도 낮다(무엇인가 섞여있다)
  - 지니계수가 높다(불순하다) = 분류의 대상이 된다 --> 지니계수를 낮추기 위해 가지치기
  - 부모 노드의 순도에 비해 자식 노드들의 순도가 증가하도록 가지치기
- 지니계수가 0 = 순도 높다 --> 지니가 낮을수록 순도 높다 --> 더이상 가지치기 하지 않음

Entropy(엔트로피)

> 정의
>
> : 무질서한 정도 , 정보량(information gain) --> 카테고리성(이산형) 변수에서 점수 높게 나옴

$$H(X) = -\sum_{i=1}^{n} p_i \log_2 p_i$$

> 원리

- 매번 일어나는 사건에 대한 정보량이 크다
  - 정보량이 많다 = 엔트로피가 높다 --> 무질서한 방향쪽으로 가지치기
- 부모 노드와의 Gap 차이가 크다 = 순도 높다
  - Gap 차이가 작다는 것은 가치치기 할 것이 많다는 것
- 엔트로피가 큰 쪽으로 가지치기
  - 엔트로피가 크다 = 분류대상이 된다 --> 엔트로피가 0이 되도록 가지치기
  - 엔트로피가 0 = 분류할 것이 없다(=pure)
- 엔트로피 최대치 = 50% 확률 = 정보량 가장 많다 --> 분류하기 가장 어려움

카이제곱 통계량(빅분기 문제집에만)

> 정의
>
> : 데이터의 분포와 사용자가 선택한 기대 또는 가정된 분포 사이의 차이를 나타내는 측정값
>
> : (기대도수) = (열의 합계) * (합의 합계) / (전체합계)

$$\chi^2 = \sum \frac{(O_i - E_i)^2}{E_i}$$

Hyper-parameter Tunning

max_depth

- 최대 트리의 깊이 (기본 값 : None, 제한 없음)

```
tree = DecisionTreeClassifier(max_depth=3, random_state=SEED)
tree.fit(X_train, y_train)
show_trees(tree)
```

min_sample_split

- 노드 내에서 분할이 필요한 최소의 샘플 수 (기본값 : 2)

```
tree = DecisionTreeClassifier(max_depth=6, min_samples_split=20,  random_state=SEED)
                                        # 조건이 상충되는경우 하나의 조건에만 해당해도 멈춤(or 조건)
tree.fit(X_train, y_train)
show_trees(tree)
```

min_samples_leaf

- 말단 노드의 최소 샘플 수 (기본값 : 1)

max_leaf_nodes

- 말단 노드의 최대 갯수, 과대 적합 방지용 (기본 값: None, 제한 없음)

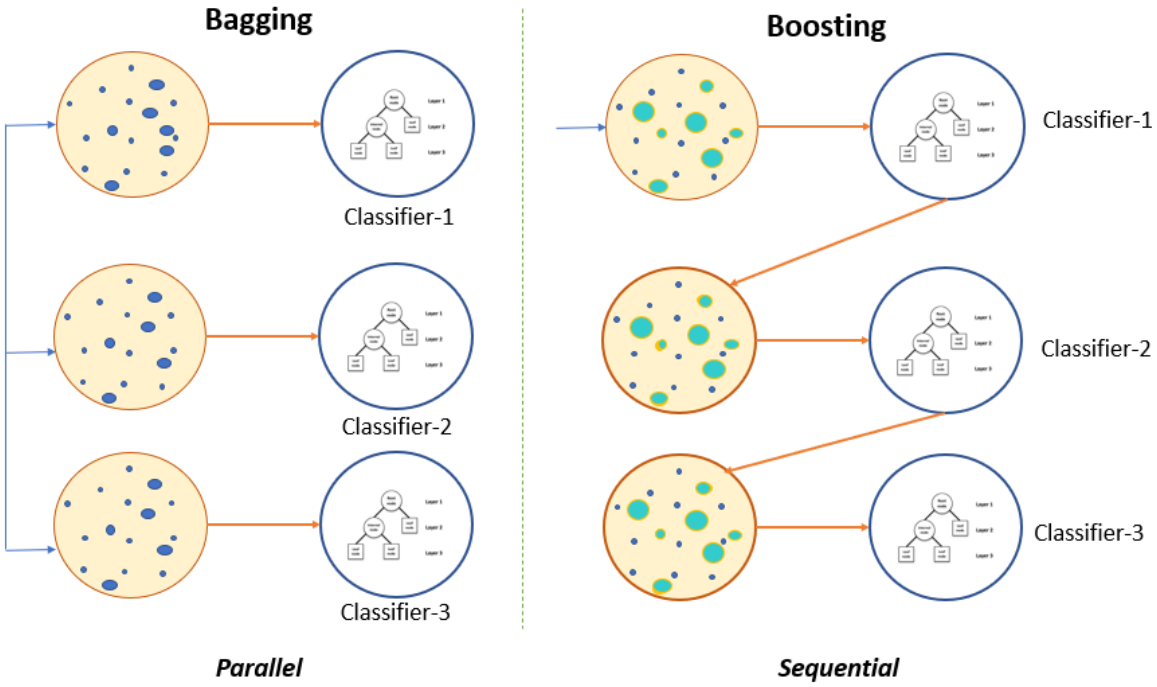
```
tree = DecisionTreeClassifier(max_depth=7, max_leaf_nodes=10, random_state=SEED)
tree.fit(X_train, y_train)
show_trees(tree)
```

Ensemble

> Ensemble 개념

- ensemble : 여러가지 모평들의 예측/분류 결과를 종합하여 최종적인 의사결정에 활용하는 기법

Bagging & Boosting



Bagging

> 정의
>
> : 하나의 모델로 다양한 데이터를 학습한다(병렬)

- bootstrap자료를 생성하여 각 자료를 모델링 한 후 결합하여 최종 예측모형을 만드는 알고리즘
  - bootstrap : 주어진 자료에서 동일한 크기의 표본을 랜덤 복원추출로 뽑은 자료

> 분석과정

1. 동시적 학습이 가능하다(parallel, 병렬적) : 속도 빠름
2. voting(투표)한 결과대로 최종 결과 산정
   - Hard Voting : 다수결
   - Soft Voting : 가중치 부여, 각 레이블 예측 확률의 평균으로 최종 분류(확률값을 w로 사용)
     - 머신러닝 default(특별한 경우 하드로 설정 가능)

> 대표적 모델 : RandomForest

Boosting

> 정의
>
> : 가지치기 후 w(가중치)를 보정하여 다시 가치치기(모델 업데이트)

> 종류

- XGBoost
- LGBoost
- catboost : 카테고리성 변수일 경우 점수 매우 잘나옴

> 분석 과정

1. 다양한 트리모델(트리 업데이트)로 데이터 하나를 학습
   - 잘못 분류된 쪽으로 강한 가중치(ex.원 가중치 * 2) 부여하여 모델 업데이트
     - 패널티가 큰 가지만 뽑아 다시 가지 형성(잘못 분류된 것들을 다시 학습시킴)
   - 올바르게 분류된 관측치에는 (ex.원래 가중치* 1/2)가중치 부여
2. overfitting의 문제 --> 모든 boosting 모델에는 규제가 있음
3. 학습이 한번 끝날 때 마다 가지치기의 모양이 달라짐
   - 순차적으로 움직임 : 하나의 학습이 끝나야 다음 학습 : 속도 느림

Lidge & Lasso

규제

규제의 목적

- 다차원의 오버피팅을 막기 위해 주요피쳐만을 다룸
  - 모델이 복잡해질수록 오버피팅이 남
  - 필요없는 피쳐 학습에 약하게 가담(주요 피쳐만 학습에 가담)시켜 올바른 학습을 가능하게 함
  - 일반화 한다

규제의 필요성

- cost 비용 조절 --> 정상적 학습 가능하게 하기 위함
  - 회기계수(w)가 커질 경우 머신러닝은 편향되게 학습한다(고편향) --> 규제 상수 값을 작게 만들면 최소 비용을 유지가능
  - 회기계수가 너무 작을 경우 학습이 제대로 되지 않음 --> 규제 상수 값을 크게
- 규제가 있는 모델은 고급 모델 : 가중치가 지나치게 커지거나 편향된 학습을 하지 않게 함 --> 비용 최소화
- W가 크면 규제상수를 낮춰야 오버피팅을 피할 수 있다.+

L1과 L2

필요 개념

- 1. 벡터(Vector) :크기와 방향을 가지고 있는 기하학적 수치
- 2. 노름(norm) : 백터의 크기를 측정하는 방법
  - 백터의 크기(magnitude) : 1차(scaler), 2차(평면), 3차(공간)....
  - 백터의 길이(length) : 백터와 백터 사이의 길이
    - 유클리디안 거리(피타고라스 정의,삼각함수: a^2+b^2=c^2)-->머신러닝 default
    - 맨하튼 거리(taxicab)

L1(Lasso)

> 원리

- 변별력 최소화, 필요없는 피쳐 삭제
- 맨하튼 거리(taxicab)로 백터 길이 측정
  - w 절대값 합 --> loss 값 그대로 사용 --> outlier에 민감하지 않음, outlier가 심한 경우 사용
  -

$$J(\theta) = \mathrm{MSE}(\theta) + \alpha \sum_{i=1}^{n} | \theta_i |$$

L2(Lidge)

> 원리

- 중요하지 않은 피쳐 약하게,가중치(기울기) 0에 가깝게 (y값에 주는 영향 최소화)
  - 기울기가 크면 클수록 주요 피쳐로 여김
- 유클리디안 거리(루트)로 백터 길이 측정
  - w의 제곱합 : 실제값과 예측값이 차이가 커지면 가중치가 커짐 ---> outlier가 심한 경우 사용 불가(일반적인 경우에 사용)
  -

$$J(\theta) = \text{MSE}(\theta) + \alpha \frac{1}{2} \sum_{i=1}^{n} \theta_i^2$$

Elastic-Net

- L1+L2
-

$$J(\theta) = \text{MSE}(\theta) + r\alpha \sum_{i=1}^{n} |\theta_i| + \frac{1-r}{2} \alpha \sum_{i=1}^{n} \theta_i^2$$

PCA

> 정의 : 차원축소, 주성분 분석, 행렬 내적, (고유값)분해

- 대표적 비지도 학습 알고리즘
- 피쳐가 다차원일 경우 차원을 줄여 거리좌표를 계산

> 원리

- 고유값 분해 : 정방형의 백터선(eigenvector)을 만들어 고유값(eigenvalue)을 만들어냄
  - pc1 : 분산이 최대화 되게 백터선을 만들어야함 --> 공간의 각 점들을 대부분 투영(projection)할 수 있도록
  - pc2 : pc1에 투영되지 못한 나머지 점들을 투영하기위해 pc1의 직교선을 그음
  - pc3 : pc2에 투영되지 못한 나머지 점들을 투영하기위해 pc2의 직교선을 그음 --> 일반적으로 pc3까지 그음
  - 백터선을 긋기 위해 pc(n)의 n+1의 피쳐가 필요함

> 장/단점

- 장점
  - 좌표상의 거리와 loss를 확실히 계산가능
  - overfitting ↓, 모델의 복잡도를 떨어뜨림
  - 일반화 시킬 수 있음(loss ↓)
- 단점
  - 차원을 축소하며 피쳐의 특징 일부가 사라짐(모델의 설명력이 떨어지게 됨)
    - but, 모델의 복잡도를 떨어뜨리기 때문에 점수를 높일 수 도 있다