Data Science Assignment2 : Decision Tree

2016025078 강덕영

- Decision Tree

데이터의 attribute 값을 비교해 나가며 레이블을 예측하는 트리 구조의 모델. Assignment2 는 제공된 훈련 데이터 셋을 바탕으로 decision tree 모델을 생성한 후, 테스트 데이터의 레이블을 예측하는 것.

- Source code

assignment2 directory structure:

2022_ITE4005_2016025078/assignment2/----data/dt_test.txt,,,,

----result/dt_result.txt,,,,

----test-program/dt_test.exe,,,,

dt.py

report.py

data file directory: assignment2/data/

train data: dt_train.txt, dt_train1.txt

test data: dt_test.txt, dt_test1.txt

result file directory: assignment2/result/

result file: dt_result.txt, dt_result1.txt

test program directory: assignment2/test-program/

test program: dt_test.exe

answers: dt_answer.txt, dt_answer1.txt (contains label data for each dt_test.txt and dt_test1.txt)

main code: dt.py

- Summary of algorithm

dt.py 는 main 함수에서 시작되며, main 함수는 훈련 데이터셋으로 decision tree 를 생성하는 부분과, 이 decision tree 를 바탕으로 테스트 데이터 셋의 정답을 예측하는 부분으로 나뉘어집니다.

첫 번째 파트는 generate_decision_tree 함수를 통해 decision tree 를 생성하는 것입니다. 이 함수는 재귀적으로 호출되면서 decision tree 를 생성합니다. 함수가 호출되면 먼저 새로운 트리 노드를 생성합니다. 그리고 이 노드의 자식 노드를 만들기 위해 현재 노드의 데이터를 가장 잘 나누는 attribute 를 구합니다. 이때, information gain 이 가장 커지도록 노드를 split 하는 attribute 가 선정됩니다. 이렇게 기준 attribute 를 구한 후엔 해당 attribute 를 기준으로 실제로 노드를 split 하고, 마지막으로 이렇게 나뉘어진 노드들에 대해 generate_decision_tree 함수를 재귀 호출 함으로써, 계속 split 이 진행되어 트리가 만들어지도록 하였습니다. 특정 recursive call path 에서 더 이상 재귀 호출을 하지 않고 노드를 leaf 로 만들고 리턴하는 조건은 두 가지 입니다. 첫째로, split 된 노드에 속한 모든 데이터가 같은 레이블을 가지고 있을 때입니다. 둘째로, 더 이상 데이터를 나눌 attribute 가 존재하지 않을 경우입니다.

두 번째 파트에서 생성된 decision tree 를 사용해서 테스트 데이터의 정답을 예측합니다. predict_label 함수를 통해 모든 테스트 데이터에 대해 레이블 값을 얻은 후, 이를 데이터 프레임의 새로운 행(레이블 행)에 추가합니다. 마지막으로 데이터 프레임을 텍스트 파일로 저장합니다.

- **Implementation detail**

트리 노드는 클래스로 정의하였습니다. 노드 클래스의 인스턴스 메소드 중 중요한 것은 set_splitting_attribute 와 check 메소드 입니다.

set_splitting_attribute 메소드는 노드의 best-splitting attribute 가 결정되었을 때 사용됩니다. 즉, information gain 이 가장 커지도록 split 하는 attribute 를 구한 후에 호출되며, 해당 attribute 가 가진 값의 종류만큼 branch 를 생성합니다. 만약, 기준 attribute 가 'age' 이고, 데이터에 존재하는 'age' 의 고유한 값이 ['<10', '10..20', '>20'] 이렇게 세 가지라고 가정했을 때, set_splitting_attribute 를 호출하면 이 노드는 세개의 자식 노드를 가리키는 branch 를 만들게 됩니다. 따라서 이후에 테스트 데이터를 가지고 트리를 따라 내려오다가 이 노드에 도착하면, 나이 값에 따라 branch 를 선택하여 자식 노드로 내려갈 수 있습니다.

그리고 check 은 트리가 빌드되고 난 후에 데이터를 라우팅 하기 위한 메소드 입니다. check 메소드는 데이터의 특정 attribute 의 값에 해당하는 자식 노드를 리턴합니다. 만약 테스트 데이터의 'age' attribute 의 값이 '10..20' 이면, 두 번째 자식노드를 리턴하게 됩니다. 즉, 일종의 라우팅 혹은 이정표 역할을 하는 것이고, 최종적으로 리프 노드에 도달했을 땐 해당 리프 노드의 레이블

값을 리턴합니다.

트리를 생성하는 generate_decision_tree 함수의 핵심 로직인, 데이터를 가장 잘 나누는 attribute 을 구하는 과정은 find_the_best_splitting_attribute 함수가 담당합니다. 이 함수의 마지막 파라미터인 attribute_selection_method 는 ID3, C4.5 와 같은 데이터 분류 함수이며, 그 중 ID3 방식을 채택하여 구현하였습니다. 가장 좋은 attribute 를 찾기 위해, 현재 가능한 모든 attribute 에 대해 ID3 를 적용하여 가장 작은 값을 가진 attribute 를 기준 attribute 로 사용하도록 하였습니다.

best attribute 를 얻은 후엔, 이 attribute 를 가지고 노드를 실제로 split 합니다. 다시 말해, attribute 에 속한 값들을 기준으로 노드를 split 하는 것입니다. 이때, 특정 값을 기준으로 split 한 노드가 텅 비였다면 majority voting 을 통해 레이블 을 결정하고 리프 노드로 마무리합니다. 반면, 빈 노드가 아닌 노드들에 대해선 다시 generate_decision_tree 함수를 호출하여 리프 노드가 될 때까지 split 이 계속되도록 하였습니다. 예를 들면, 현재 데이터를 나누는 기준 attribute 가 'age' 이고, ['<10', '10..20', '>20'] 값으로 노드를 split 한다고 가정합시다. 이때, 나이가 20세가 넘는 데이터가 하나도 없는 경우가 존재할 수 있고, 그러면 '>20' 에 해당하는 자식 노드는 빈 노드이기 때문에, 레이블을 정할 수 없습니다. 따라서 현재 데이터에 속한 레이블 중 가장 많은 레이블로 대체하는 majority voting 을 실시합니다. 즉, 부모 노드의 데이터에서 수적으로 가장 우세한 레이블을 선택하는 것입니다. 이렇게 majority voting 을 통해 레이블을 정하고 해당 노드를 리프 노드로 설정함으로써, 다양한 테스트 데이터에 대해서도 레이블을 예측할 수 있도록 하였습니다.

decision tree 가 생성되고 나면, 테스트 데이터셋의 각 데이터에 대해 predict_label 함수를 실행하여 레이블을 예측합니다. predict_label 함수는 테스트 데이터의 attribute 를 check 메소드를 통해 비교해 나가면서 리프 노드를 향해 내려가도록 하고, 최종적으로 레이블을 리턴합니다.

- Instruction

운영체제: macOS

코드 실행 가상환경: anaconda (which includes numpy and pandas module)

기존 데이터셋에 대해 코드 실행 방법:

2022_ITE4005_2016025078/assignment2/ 에서,

```
python (or python3) dt.py dt_train.txt dt_test.txt dt_result.txt
```

python (or python3) dt.py dt_train1.txt dt_tes1t.txt dt_result1.txt

테스트 프로그램 실행방법:

2022_ITE4005_2016025078/assignment2/ 에서,

mono ./test-program/dt_test.exe ./test-program/dt_answer.txt ./result/dt_result.txt

mono ./test-program/dt_test.exe ./test-program/dt_answer1.txt ./result/dt_result1.txt

새로운 데이터셋에 대해 코드 및 테스트 프로그램 실행방법:

1. 2022_ITE4005_2016025078/assignment2/data/ 에 train.txt, test.txt 파일 저장,
   2022_ITE4005_2016025078/assignment2/test-program/ 에 answer.txt 파일 저장.
2. dt.py 실행:

2022_ITE4005_2016025078/assignment2/ 에서,

python (or python3) dt.py train.txt test.txt result.txt

3. test program 실행:

2022_ITE4005_2016025078/assignment2/ 에서,

mono ./test-program/dt_test.exe ./test-program/answer.txt ./result/result.txt