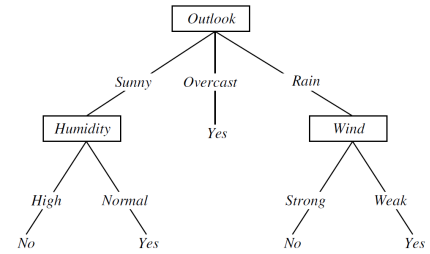
KU KONKUK UNIVERSITY

Decision Tree

자연어처리연구실

Decision Tree

- 결정 트리
  - 자질들의 정보 획득량(information gain)에 따라 트리 형태의 규칙을 자동 생성하는 기계학습 모델
- 예제: Play Tennis
  - 테니스 시합을 할 수 있을지 판단하는 문제



Edited by Harksoo Kim

자질 (Feature)

- 자질 (feature)
  - 문제 해결에 영향을 미치는, 판단 근거가 되는 요소
  - 관측 및 측정 가능한 요소
- 예제: Play Tennis
  - 날씨 (outlook)
    - Sunny, Overcast, Rain
  - 온도 (temperature)
    - Hot, Mild, Cool
  - 습도 (humidity)
    - High, Normal, Low
  - 바람 (wind)
    - Strong, Weak

Edited by Harksoo Kim

정보 획득량 (Information Gain)

- 정보 획득량
  - 자질의 값을 알게 됨으로써 얻어지는 문제 복잡도(전체 엔트로피)에 대한 감소 기대치
- 엔트로피 (entropy)
  - 문제의 복잡도를 측정하는 척도
  - 정보 이론 (information theory): 특정 확률 p를 가진 메시지를 상대방에게 전달하는데 필요한 비트 수에 대한 기댓값
  - 예제
    - 상자의 상단에 별이 위치한다는 정보를 상대방에게 전달하려면 몇 비트가 필요할까? 우측 상단에 별이 있다면?

$-log_2(1/2)$
$= -log_2(2^{-1})$
$= 1$

$-log_2(1/4)$
$= -log_2(2^{-2})$
$= 2$

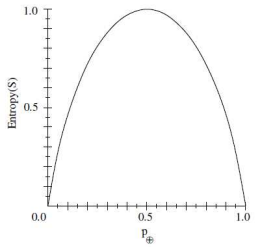
Edited by Harksoo Kim

엔트로피 (Entropy)

- 엔트로피의 최대값
  - 가장 애매한 확률(=1/2)을 가질 때 판단을 내리기 가장 힘듦 → 문제가 복잡함 → 최대 엔트로피를 가짐
- 기댓값
  - 각 사건이 벌어졌을 때의 이득과 그 사건이 벌어질 확률을 곱한 것을 전체 사건에 대해 합한 값

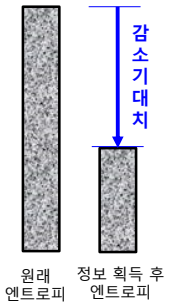
- S is a sample of training examples
- $p_{\oplus}$ is the proportion of positive examples in S
- $p_{\ominus}$ is the proportion of negative examples in S
- Entropy measures the impurity of S

$$Entropy(S) \equiv -p_{\oplus} \log_2 p_{\oplus} - p_{\ominus} \log_2 p_{\ominus}$$



정보 획득량 (Information Gain)

- 정보 획득량
  - 자질의 값을 알게 됨으로써 얻어지는 문제 복잡도(전체 엔트로피)에 대한 감소 기대치



$$Gain(S, A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

Training Examples for Play Tennis

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

S = [9+, 5-]
S_{Weak} = [6+, 2-]
S_{Strong} = [3+, 3-]

IG(S,Wind) = Entropy(S) - 8/14Entropy(S_{Weak})
- 6/14Entropy(S_{Strong})
= 0.940 - 8/14*0.811
- 6/14*1.00
= 0.048

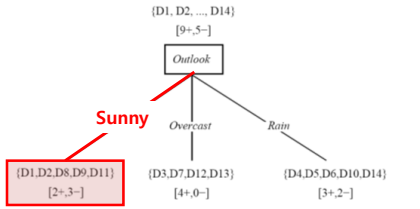
IG(S,Outlook) = 0.246
IG(S,Humidity) = 0.151
IG(S,Temperature) = 0.029

∴ 정보획득량이 가장 큰 'Outlook'을 DT의 최우선 노드로 결정하는 것이 최적임

Values(Wind) = { Weak, Strong }

Training Examples for Play Tennis

Day		Temperature	Humidity	Wind	PlayTennis
D1		Hot	High	Weak	No
D2		Hot	High	Strong	No
D3		Hot	High	Weak	Yes
D4		Mild	High	Weak	Yes
D5		Cool	Normal	Weak	Yes
D6		Cool	Normal	Strong	No
D7		Cool	Normal	Strong	Yes
D8		Mild	High	Weak	No
D9		Cool	Normal	Weak	Yes
D10		Mild	Normal	Weak	Yes
D11		Mild	Normal	Strong	Yes
D12		Mild	High	Strong	Yes
D13		Hot	Normal	Weak	Yes
D14		Mild	High	Strong	No



S = [2+, 3-]
S_{Hot} = [0+, 2-]
S_{Mild} = [1+, 1-]
S_{Cool} = [1+, 0-]

S = [2+, 3-]
S_{High} = [0+, 3-]
S_{Normal} = [2+, 0-]

S = [2+, 3-]
S_{Strong} = [1+, 1-]
S_{Weak} = [1+, 2-]

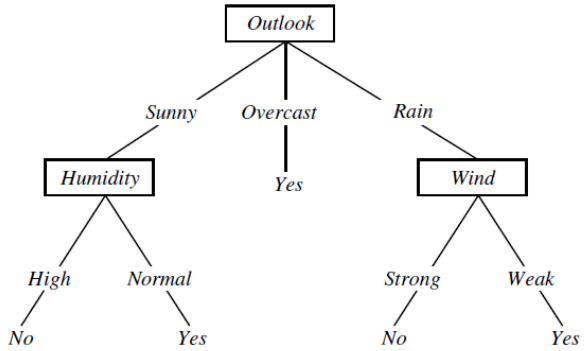
IG(S,Temp)=Entropy(S)-2/5Entropy(S_{Hot})-2/5Entropy(S_{Mild})-1/5Entropy(S_{Cool})=0.971-2/5*0-2/5*1-1/5*0=**0.571**

IG(S,Hum)=

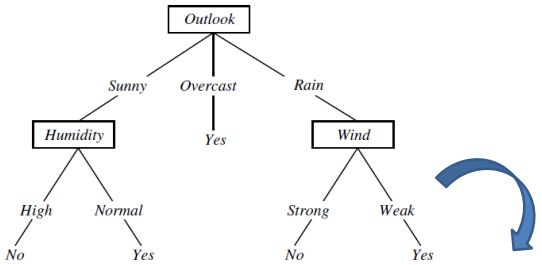
IG(S,Wind)=

?

Decision Tree for Play Tennis



Converting a Tree to Rules



IF $(Outlook = Sunny) \wedge (Humidity = High)$
THEN $PlayTennis = No$

IF $(Outlook = Sunny) \wedge (Humidity = Normal)$
THEN $PlayTennis = Yes$

...

자질 구성 시 고려사항

- 연속 자질 (continuous feature)
  - 엔트로피 측정을 위해서는 이산 자질(discrete feature)로 변환
  - 예제
    - 기온<15 → Cool, 15<=기온<25 → Mild, 기온>=25 → Hot
- 복잡한 자질
  - 빈도가 낮아서 엔트로피가 높게 측정 됨
  - 문제 해결에 독립적으로 작용 가능하다면 분할하는 것이 좋음
  - 예제
    - Date: 2021년-5월-10일 → Year: 2021년, Month: 5월, Day: 10일
- 비싼 자질
  - 측정 및 관측 비용이 많이 드는 자질은 비용을 엔트로피에 반영하는 것을 고려할 수 있음

Advantages and Disadvantages

- Advantages
  - Does feature selection
  - Handles features of different types
  - Very fast prediction
  - Interpretable decision rules
- Disadvantages
  - Does not combine feature values, difficulty with dependent features
  - Finding optimal trees is intractable
- Implementations
  - C4.5: Free
  - CART: Commercial

실습

실습 코드 다운로드:
https://github.com/KUNLP/Lecture

- PlayTennis 데이터를 입력으로 하여 Decision Tree를 생성하는 프로그램을 작성하시오.
  - PlayTennis 데이터 형식
    - CSV (Comma-Separated Value)
    - 입력: outlook, temp, humidity, windy
    - 출력: play 여부 (yes/no)

	A	B	C	D	E
1	outlook	temp	humidity	windy	play
2	sunny	hot	high	FALSE	no
3	sunny	hot	high	TRUE	no
4	overcast	hot	high	FALSE	yes
5	rainy	mild	high	FALSE	yes
6	rainy	cool	normal	FALSE	yes
7	rainy	cool	normal	TRUE	no
8	overcast	cool	normal	TRUE	yes
9	sunny	mild	high	FALSE	no
10	sunny	cool	normal	FALSE	yes
11	rainy	mild	normal	FALSE	yes
12	sunny	mild	normal	TRUE	yes
13	overcast	mild	high	TRUE	yes
14	overcast	hot	normal	FALSE	yes
15	rainy	mild	high	TRUE	no


Edited by Harksoo Kim

실습

구글 colab 연결

```
from google.colab import drive
drive.mount("/gdrive", force_remount=True)
```

데이터 읽기

Data Frame을 손쉽게 다루게 해주는 라이브러리

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn import tree
import graphviz

# 데이터 경로
file_path = "/gdrive/MyDrive/colab/dt/PlayTennis.csv"
# 데이터 경로로부터 파일을 읽음 (pandas 라이브러리 사용)
datas = pd.read_csv(file_path)

# 데이터 출력 (데이터 형태 확인)
print(datas)
```

	outlook	temp	humidity	windy	play
0	sunny	hot	high	False	no
1	sunny	hot	high	True	no
2	overcast	hot	high	False	yes
3	rainy	mild	high	False	yes
4	rainy	cool	normal	False	yes
5	rainy	cool	normal	True	no
6	overcast	cool	normal	True	yes
7	sunny	mild	high	False	no
8	sunny	cool	normal	False	yes
9	rainy	mild	normal	False	yes
10	sunny	mild	normal	True	yes
11	overcast	mild	high	True	yes
12	overcast	hot	normal	False	yes
13	rainy	mild	high	True	no


Edited by Harksoo Kim

실습

데이터 변환 (문자열 → 숫자)

```
# 범주형 데이터를 수치형 데이터로 자동 변환해주는 라이브러리
label_encoder = LabelEncoder()

# 정답 클래스 이름 (yes, no)
target_names = label_encoder.fit(datas['play']).classes_
print("target_names : {}".format(target_names))

datas['outlook'] = label_encoder.fit_transform(datas['outlook'])
datas['temp'] = label_encoder.fit_transform(datas['temp'])
datas['humidity'] = label_encoder.fit_transform(datas['humidity'])
datas['windy'] = label_encoder.fit_transform(datas['windy'])
datas['play'] = label_encoder.fit_transform(datas['play'])

# 데이터 출력 (데이터 포맷 변환 후 결과 확인)
print(datas)
```

```
target_names : ['no' 'yes']
    outlook  temp  humidity  windy  play
0         2     1         0      0     0
1         2     1         0      1     0
2         0     1         0      0     1
3         1     2         0      0     1
4         1     0         1      0     1
5         1     0         1      1     0
6         0     0         1      1     1
7         2     2         0      0     0
8         2     0         1      0     1
9         1     2         1      0     1
10        2     2         1      1     1
11        0     2         0      1     1
12        0     1         1      0     1
13        1     2         0      1     0
```

```
# 입력 데이터와 정답 데이터로 분리
x_data, y_data = datas.drop(['play'], axis=1), datas['play']

# 분리 결과 확인
print(x_data)
print()
print(y_data)
```

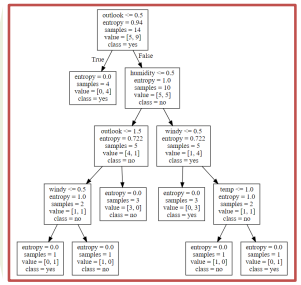
```
    outlook  temp  humidity  windy
0         2     1         0      0
1         2     1         0      1
2         0     1         0      0
3         1     2         0      0
4         1     0         1      0
5         1     0         1      1
6         0     0         1      1
7         2     2         0      0
8         2     0         1      0
9         1     2         1      0
10        2     2         1      1
11        0     2         0      1
12        0     1         1      0
13        1     2         0      1

0     0
1     0
2     1
3     1
4     1
5     0
6     1
7     0
8     1
9     1
10    1
11    1
12    1
13    0
Name: play, dtype: int64
```


Edited by Harksoo Kim

실습

```
# Decision tree 모델 학습 ()
decision_tree = tree.DecisionTreeClassifier(criterion = 'entropy')
train_result = decision_tree.fit(x_data, y_data)
```

```
# 학습 결과 확인 (graphviz 라이브러리 사용)
graph = graphviz.Source(tree.export_graphviz(train_result, out_file=None,
                                             feature_names=x_data.columns,
                                             class_names=target_names))
graph
```




```
# 학습한 모델을 사용하여 예측
predict_result = decision_tree.predict(x_data)

# 예측 결과 출력 (실제 정답을 맞춘 경우 True로 표시됨)
print(predict_result == y_data)
```

```
0     True
1     True
2     True
3     True
4     True
5     True
6     True
7     True
8     True
9     True
10    True
11    True
12    True
13    True
Name: play, dtype: bool
```


Edited by Harksoo Kim

질의응답

Q&A

Homepage: http://nlp.konkuk.ac.kr
E-mail: nlpdrkim@konkuk.ac.kr

Edited by Harksoo Kim