Ch. 6 질적 분류

Pattern Recognition & Machine Learning Laboratory
Tae-jin Woo
Jul. 08, 2021

Introduction

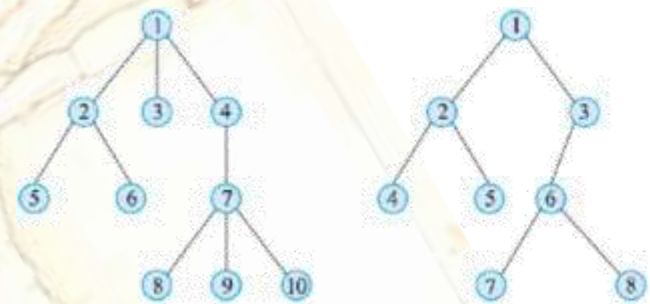
- 데이터 정의
  - 계량 데이터
    - 양적으로 표현 가능하며, 정수 또는 실수로 표현 가능
      - ex) 키, 나이, GDP 등
  - 비계량 데이터
    - 양적으로 표현 불가능하며, 편의상 정수 또는 실수로 표현 가능
      - ex) 직업, 행정 구역, 혈액형 등
    - 비계량 데이터 간 비교
      - 단순 연산을 통한 데이터 간 거리 측정 불가
      - 비계량 값 x_i를 갖는 d차원 특징 벡터

        $\mathbf{x} = (x_1, x_2, \dots, x_d)^T$로 표현하여 연산 필요
- 질적 분류기
  - 비계량 데이터를 다루는 분류기
    - 베이시언 분류기, 신경망, SVM은 모두 양적 분류기
      - 샘플은 d차원 공간에서 점으로 표시되며, 점 사이 거리 개념 존재
      - 비계량 데이터는 계량형으로 변환 후 적용 가능
    - 결정 트리, 스트링 인식기는 질적 분류기
      - 결정 트리는 계량 데이터도 적용 가능

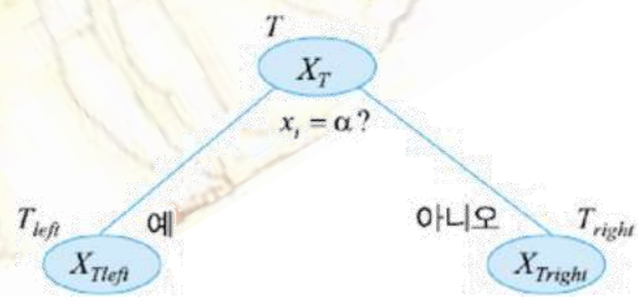
양적 분류기와 질적 분류기

결정 트리 (1/4)

- 생성 원리
  - 기본 구조
    - 계층 구조를 나타낼 수 있는 트리형 자료 구조 적용
    - 예/아니오 형 질문을 통해 트리 구조 생성
- 고려 사항
  - 노드에서 나눌 가지의 개수 및 질문 내용
  - 멈춤 조건
  - 잎 노드의 할당 부류
    - 잎 노드에 가장 높은 확률로 존재하는 ω_i 부류로 지정
- 분기
  - 분기 방법
    - 노드 T에서 노드 T_{left}와 노드 T_{right}로 분기
      - 각 노드의 질문 내용에 따라 분기
  - X_T는 초기 훈련 집합 X의 부분 집합이며, 루트 노드에서 T까지의 경로에서 살아남은 샘플 집합
    - 가능한 한 X_{Tleft}와 X_{Tright}에 각각 동일 부류 샘플 필요
  - 샘플 집합
    - $X_{Tleft} \cup X_{Tright} = X_T$ 및 $X_{Tleft} \cap X_{Tright} = \emptyset$ 만족 필요



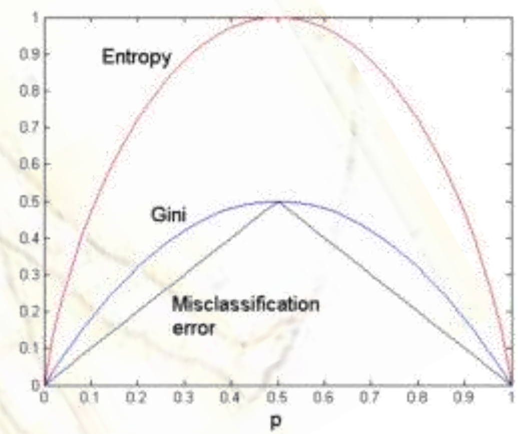
트리와 이진 트리



노드의 분기

결정 트리 (2/4)

- 분기에 대한 평가
  - ➢ 동질 부류 판단 기준의 필요성
    - 질문을 만드는 방법
      - – d개의 특징을 가지며, 각 특징이 평균 n개의 값을 가지면 가능한 후보 질문은 dn개
      - – 후보 질문이 상수이므로 완전 탐색 사용
    - 후보 질문 평가를 위한 기준 함수
      - – 각 질문에 의한 분기가 잘 되었는지에 기준 필요
  - ➢ 불순도
    - 엔트로피 불순도 : $im(T) = -\sum_{i=1}^{M} P(\omega_i|T)\log_2 P(\omega_i|T)$
      - – M개의 부류에서 얻을 수 있는 정보의 기댓값
    - 지니 불순도 : $im(T) = \sum_{i \neq j} P(\omega_i|T)P(\omega_j|T)$
      - – 임의의 두 샘플을 선택했을 때, 서로 다른 부류일 확률
    - 오분류 불순도 : $im(T) = 1 - \max_i P(\omega_i|T)$
      - – 샘플을 가장 높은 확률의 부류로 분류할 때 발생하는 오분류 비율



불순도 함수 그래프

  - ➢ 불순도 감소량
    - 분기로 생성되는 새 노드가 상위 노드 대비 낮은 불순도를 가지는 것이 유리
    - 기본 함수 : $\Delta im(T) = im(T) - \frac{|X_{Tleft}|}{|X_T|} im(T_{left}) - \frac{|X_{Tright}|}{|X_T|} im(T_{right})$
    - 투잉 기준 : $\Delta im(T) = \frac{|X_{Tleft}|}{|X_T|}\frac{|X_{Tright}|}{|X_T|}\left(\sum_{i=1}^{M} |p(\omega_i|T_{left}) - p(\omega_i|T_{right})|\right)^2$

결정 트리 (3/4)

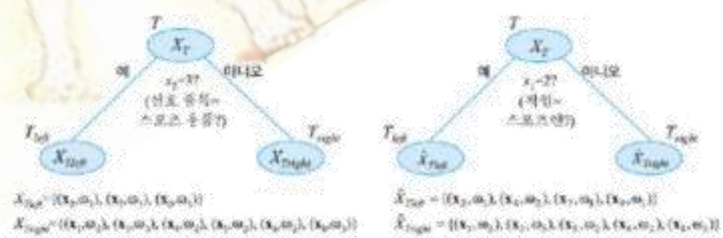
- 학습
  - 학습 알고리즘
    - 재귀 함수를 통한 반복 학습
    - 후보 질문 생성 및 선정
    - 멈춤 조건에 따른 새로운 자식 노드 생성 여부 확인
      - 불순도가 0인 경우
      - X_T의 샘플 개수 또는 불순도 감소량이 임계값 이하인 경우
  - 멈춤 조건
    - 불순도만 고려하면 쉽게 과적합 발생 가능
    - 임계값이 너무 크면 설익은 수렴 도달 가능
- 특징
  - 장점
    - 계량 및 비계량 데이터 모두 처리 가능
    - 정규화 등의 데이터 전처리 불필요
    - 화이트 박스 모델이므로 분류 결과 해석 가능
    - 손실 특징을 다루기 용이
      - 대리 분기 사용 가능
  - 단점
    - 불안정성을 보일 수 있음

```
입력: 훈련 집합 X = {(x1, t1), ..., (xN, tN)}
출력: 결정 트리 R
알고리즘:

노드 하나를 생성하고 그것을 R이라 한다.  // 이것이 루트 노드이다.
T = R;
XT = X;
split_node(T, XT);    // 루트 노드를 시작점으로 하여 순환 함수를 호출한다.

split_node(T, XT) {  // 순환 함수
    노드 T에서 후보 질문을 생성한다.
    모든 후보 질문의 불순도 감소량을 측정한다.  // (6.6) 또는 (6.7) 이용
    불순도 감소량이 최대인 질문 q를 선택한다.
    if (T가 멈춤 조건을 만족) {
        T에 부류를 할당한다.
        return;
    }
    else {
        q로 XT를 XTleft와 XTright로 나눈다.
        새로운 노드 Tleft와 Tright를 생성한다.
        split_node(Tleft, XTleft);
        split_node(Tright, XTright);
    }
}
```

학습 알고리즘 의사코드



주 분기와 대리 분기

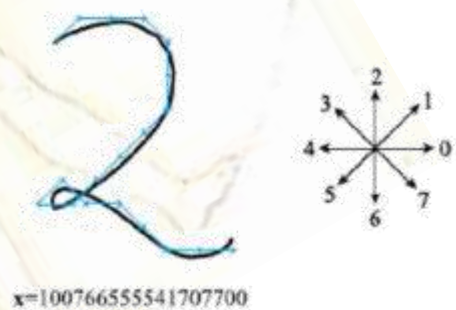
결정 트리 (4/4) & 스트링 인식기 (1/3)

- **대표 시스템**
  - 시스템 종류
    - **CART, ID3, C4.5 등**
      - 기본 트리 원리 채택, 연구 그룹 상이
  - 특성 비교
    - 트리 형태, 가지치기, 회귀 지원 등 차이 존재
    - 각 시스템마다 절대적 우열 관계 없음
      - 특정 상황 및 데이터에 따라 다를 수 있음

특성	CART	ID3	C4.5
실수 데이터	부등호 질문	등식 질문	부등호 질문
트리 형태	이진 트리	트리	←
가지치기	잎 노드 병합	X	규칙 집합
분류	지원	←	←
회귀	지원	X	X
손실 특징	대리 분기	X	샘플 무시
다중 변수 질문	지원	X	X

CART, ID3, C4.5 특성 비교

- **교정 거리**
  - 목적
    - 응용 분야에서 패턴은 종종 가변 길이의 스트링으로 표현됨
      - **ex)** 음성 인식, 필기 인식 등
      - 길이가 가변적이므로 유클리디언 거리 적용 불가
    - 가변 길이의 스트링 간 거리 또는 유사도 측정 방법 필요
      - 기존 해밍 거리 적용 또한 부적절
      - 삽입, 삭제 등의 교정을 통한 거리 계산 가능
  - 거리 계산 방법
    - 레벤슈타인 거리
    - 다메라우-레벤슈타인 거리



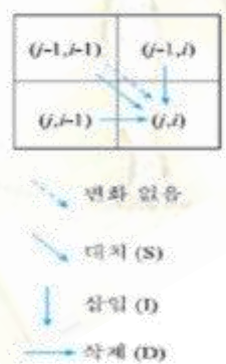
$$\mathbf{x}_1 = aabbacb$$
$$\mathbf{x}_2 = abbacb*$$
$$\mathbf{x}_3 = aaccabb$$
$$\mathbf{x}_4 = a*bbacb$$

체인 코드 표현

스트링 인식기 (2/3)

- 레벤슈타인 거리
  - 샘플 구분
    - 테스트 샘플 $\mathbf{x} = x_1x_2 \dots x_c$ 및 기준 샘플 $\mathbf{y} = y_1y_2 \dots y_r$
      - **x**를 **y**로 변환하는 비용 ≠ **y**를 **x**로 변환하는 비용
      - 두 변환 비용이 일치할 경우 구분 불필요
  - 연산
    - 삽입, 삭제, 대치 사용
      - 삽입 : x = **abc** → x = **abcd** (x_3뒤에 **d** 삽입)
      - 삭제 : x = **abcd** → x = **abd** (x_3 삭제)
      - 대치 : x = **abcd** → x = **abce** (x_4를 **e**로 대치)
  - 2차원 D배열

		0	1	2	3	4	5	6	7	8	9	10
		x=	r	e	v	g	n	i	a	t	o	n
0	y=	0	1	2	3	4	5	6	7	8	9	10
1	r	1	0	1	2	3	4	5	6	7	8	9
2	e	2	1	0	1	2	3	4	5	6	7	8
3	c	3	2	1	1	2	3	4	5	6	7	8
4	o	4	3	2	2	2	3	4	5	6	6	7
5	g	5	4	3	3	2	3	4	5	6	7	7
6	n	6	5	4	4	3	2	3	4	5	6	7
7	i	7	6	5	5	4	3	2	3	4	5	6
8	t	8	7	6	6	5	4	3	3	3	4	5
9	i	9	8	7	7	6	5	4	4	4	4	5
10	o	10	9	8	8	7	6	5	5	5	4	5
11	n	11	10	9	9	8	7	6	6	6	5	4




```
입력: 테스트 스트링 x와 기준 스트링 y
    // 삽입과 삭제의 비용이 같으면 대칭이므로 테스트와 기준 스트링의 구별
       이 불필요
출력: x와 y 사이의 거리 d, 교정 연산 목록 edit_list[]
알고리즘:

1. x의 길이를 c라 하고 y의 길이를 r이라 한다.
2. (r+1)*(c+1)의 배열 D[0…r][0…c]를 생성한다.
        // 전방 계산 (라인 3-12)
3. for (i=0 to c) D[0][i] = i;  // 행 0의 초기화
4. for (j=0 to r) D[j][0] = j;  // 열 0의 초기화
5. for (j=1 to r)
6.    for (i=1 to c) {
7.       if (xi = yj) scost=0; else scost=1;  // 대치 비용
8.       D[j][i] = minimum(D[j-1][i]+1, D[j][i-1]+1, D[j-1][i-1]+scost);
                           (삽입)       (삭제)       (대치)
9.       act = 라인 8에서 최소가 되었던 연산;  // 역 추적에 사용할 것임
10.      if (act = '대치' and scost = 0) act = '변화 없음';
11.      action[j][i] = act;
12.   }
```

알고리즘 의사코드

스트링 인식기 (3/3)

- 거리 계산
  - 최소 비용의 변환 방법을 찾는 최적화 문제
    - 완전 탐색의 경우 계산 시간 과다 소요
  - 크기가 $(r+1)(c+1)$인 2차원 배열 $D[0:r][0:c]$ 적용
    - 삽입, 삭제, 대치 연산 중 최소 비용의 연산 선택
    - 대치 비용은 x_i에 따라 다르게 적용 ex) $s_{cost_{u \to v}} < s_{cost_{u \to q}}$
    - $D[j][i]$ 반복 계산 및 최적 원리에 따라 2차원 배열 완성 (동적 프로그래밍 적용)
- 특성
  - 대치만 사용하면 해밍 거리와 같음
  - 삽입과 삭제만 사용하면 최장 공통 부분 스트링 (LCS)과 같음
    - LCS는 삽입과 삭제만 허용
  - 2차원 D배열을 채워야 하므로, 계산 복잡도 $\Theta(rc)$
    - 연산이 상수개의 덧셈, 뺄셈 뿐이므로 상수 시간 필요

▪ 다메라우-레벤슈타인 거리

- 레벤슈타인 거리와의 비교
  - 기존 레벤슈타인 거리 연산에 교환 추가
    - ex) pattren → pattern (x_5와 x_6 교환)
    - 실생활에서 인접한 두 글자가 서로 바뀌는 경우가 높은 확률로 존재
  - 교환 연산에 대한 알고리즘
    - 교환이 성립하는 경우, 교환 비용과 기존 비용 비교 후 최저 비용의 연산 선택