참고자료: Machine Learning p.66 (issues in decision tree learning)

**Chapter 3. < C4.5 알고리즘 >**

|  |
| --- |
| ID3 알고리즘의 단점을 보완하고 새로운 기능을 추가한 알고리즘이다. 수치형속성 처리, 결측치  처리, 속성선택 시 Branch의 수에 대한 가중치 적용, 가지치기(Pruning) 등의 기능을 추가하였다. 기본적인 개념은 ID3와 동일하다. 따라서 C4.5를 이해하려면 ID3의 동작원리를 자세히 이해하고 그 후에 보완된 기능들을 학습해야 한다. |

**1. C4.5 알고리즘 소개**

   이번 장에서는 C4.5 라는 분류 알고리즘에 대해서 설명하고자 한다. C4.5 알고리즘은 1993년 Quinlan 에 의해 제안되었다. C4.5 알고리즘은 ID3 알고리즘과 크게 다르지 않다. ID3 알고리즘과 다른 어떤 더 좋은 알고리즘이라고 하기 보다는 ID3 알고리즘의 몇 가지 단점(한계)들을 보완한 알고리즘이라고 이해해는 것이 정확하다.

  앞에서 설명한 ID3 알고리즘은 트리 기반의 분류 알고리즘에 있어 대표적인 알고리즘이다. 이 알고리즘은 몇 가지 점에서 단점을 가지고 있다. ID3가 갖고 있는 한계점을 들을 보완한 알고리즘이 C4.5 알고리즘이다.

\* C4.5 알고리즘들이 보완하고자 한 ID3 알고리즘의 문제들은 아래와 같다.

**(a) 수치형 속성 취급 (handling continuous attributes)**   ID3 알고리즘은 범주형 속성에 대해서만 트리를 생성하는 방법을 제시하고 있다. 따라서 수치형 속성은 모델 생성에 활용할 수 없는 한계가 있다. C4.5에서는 수치형 속성까지 사용하는 방법에 대해서 제안한다.

**(b) 무의미한 속성을 제외하는 문제**   학번, 생일, 번지수(세부주소)등과 같이 모든 레코드를 잘게 분할하는 (극단적으로는 한 개의 속성으로 구분하는) 속성이 선택되지 않도록 하는 것이 필요하다.

**(b) 나무의 깊이 문제 (how deeply to grow the decision tree)**  
   ID3 알고리즘으로 나무 모델을 생성할 경우 나무의 깊이가 너무 깊게 들어가는 문제가 있다. C4.5 알고리즘에서는 이 문제를 해결하기 위해 깊이를 어느 정도까지 진행할 것인가에 대하여 제안한다. 이것은 “과적합(overfitting) 문제” 및 “가지치기(pruning)”와 관련이 있다.

**(c) 결측치 처리 (Handling missing attributes values)**   데이터 중 특정 속성의 값이 부분적으로 입력되어 있지 않는 데이터에 대한 처리 문제

**(d) 비용고려 (handling attributes with different costs)**  
   속성들을 서로 다른 가중치를 반영하여 사용하는 방법

**(e) 효율성 (Improving computational efficiency)**  
   시행되는 시간적인 효율성을 높이는 방법

**2. C4.5 에서 수치형 속성을 처리하는 방법**

참고1: Data Mining (IAN H.WITTEN, EIBE FRANK) p.159-161 (6.1절 Numeric Attr.)  
참고2: Machine Learning p.72 중반 p.73

   전에 설명했던 ID3 알고리즘은 모든 속성이 명목형(Nominal) 속성일 경우에만 처리가 가능하였다. (명목형 속성은 범주형 속성<Categorical Attribute)과 같은 의미로 볼 수 있다.) 그러나 실세계의 데이터들은 대부분 수치형 속성들을 포함하고 있다. ID3 알고리즘에 수치형 속성을 처리할 수 있도록 확장하는 것이 필요하다. 이 작업은 그렇게 어려운 것은 아니다. 수치형 속성에 대하여 우리는 binary 방식과 Split 방식의 두 가지 방식을 고려할 수 있다. Table1.3과 같이 수치형 속성을 갖고 있는 Weather 데이터에 대하여 분석을 하는 경우를 고려해보자. 먼저 온도(Temperature)에 대한 분할을 고려해보자. 온도는 아래와 같이 표현될 수 있다. (클래스 값과 같이 표현하였다.) (위 그림에서 72, 75의 경우 클래스 값이 두 개가 기록된 것은 같은 온도 값에 대한 레코드가 2개로 중복된 경우이다.)

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 64 | 65 | 68 | 69 | 70 | 71 | 72 | 75 | 80 | 81 | 83 | 85 |
| yes | no | yes | yes | yes | no | no yes | yes yes | no | yes | yes | no |

   위 표에서 보면 분할할 수 있는 Breakpoint는 11개가 존재한다. 만약 같은 클래스에 대해서 분할을 허용하지 않는다면 8개의 가능한 분할지점(Breakpoint)이 있다. 각 분할지점에 대한 Information Gain 값을 이전에 배운 Entropy 식에 의해 계산될 수 있다.

    예를 들어, [온도 < 71.5] 를 기준으로 하면 4개의 yes와 2개의 no가 포함된다. 반면 [온도 > 71.5] 에 대해서는 5개의 yes와 3개의 no가 포함된다. 따라서 Information value는 아래와 같이 계산된다. (아래와 같이 71.5도를 분할점으로 한 온도라는 속성을 기준으로 분할하면 0.939의 Info Gain 값을 얻는다.)

**Info( [4,2], [5,3] ) = (6/14)\*Info([4,2]) + (8/14)\*(info[5,3]) = 0.939 bits.**

   더 복잡한 아이디어를 사용하면 더 좋은 info gain을 얻을 수도 있겠지만, (위와 같은 방식으로 Info가 가장 낮은 분할지점을 찾아서) 특정 값을 찾아서 그 값을 기준으로 하여 분할하는 단순한 방법을 일반적으로 많이 사용한다. 예를 들어, 인스턴스기반 학습에서 개념의 중간지점을 분할점으로 선택하는 단순한 방식이 ---  
   
   분할정복 방식을 사용하여 의사결정트리를 생성할 때, 분할을 위한 첫 번째 속성이 선택되면 그 속성에 의하여 하위노드들로 분리하고 각 하위 자식노드들에 대하여 재귀적으로 알고리즘을 호출한다. 수치형 속성에 대해서는 각 노드의 레코드들을 수치형 속성의 값에 따라 정렬하여야 한다. 실제로 의사결정트리를 생성하는 프로그램들이 이러한 방식을 사용한다. 그러나 사실은 각 노드별로 다시 정렬해야할 필요가 없다. 왜냐하면 상위노드에서 정렬된 순서는 하위 노드에서도 그대로 사용될 수 있기 때문이다. 빠른 프로그램의 구현을 위해서는 상위 노드의 정렬 정보를 사용하면 된다. Weather 데이터 분석에서 Temperature(온도) 속성에 대하여 고려해보자. 온도 속성의 정렬 순서는 아래와 같다. (이번에는 중복된 경우를 모두 포함하고 있다.)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 64 | 65 | 68 | 69 | 70 | 71 | 72 | 72 | 75 | 75 | 80 | 81 | 83 | 85 |
| 7 | 6 | 5 | 9 | 4 | 14 | 8 | 12 | 10 | 11 | 2 | 13 | 3 | 1 |

   위 그림에서 아래의 기울임체로 기록된 숫자는 각 인스턴스들의 번호를 의미한다. 즉, 인스턴스 7의 온도는 64도이고, 인스턴스 6의 온도는 65도 이다. (앞에서부터 순서대로 설명한 것임) 우리가 최 상위노드에서 outlook 이라는 속성을 분할을 위하여 선택했다고 가정해보자. 자식 노드들 중에서 outlook=sunny 인 경우에 해당하는 노드를 고려해보자. outlook이 sunny 값을 갖는 인스턴스들의 번호는 1, 2, 8, 9 그리고 11이다. (즉 5개의 인스턴스가 노드에 포함됨)

상위노드에서 생성된 순서를 사용하여 하위 노드의 순서를 찾을 수 있다.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **9** | **8** | **11** | **2** | **1** |

   따라서, 각 수치형 속성에 대하여 반복적인 정렬은 하지 않아도 된다. 각 수치형 속성의 순서는 처음의 상위 노드(루트노드)에서 결정되므로 하위 노드에서 더 이상의 정렬을 필요하지 않다.

   4.3절에서 설명한 것 같이 의사결정트리는 명목형(범주형) 속성을 테스트 할 때, 각 분할되는 가지(branch)는 속성이 갖는 값들의 개수만큼 펼쳐지게 된다. 그러나 수치형 속성에 대해서는 하나의 breakpoint 값을 기준으로하여  2개의 branch를 펼치게 된다. 이러한 방식(수치형 속성에 대해서 2개의 가지를 평치는 방식)은 명목형 속성과 수치형 속성을 처리하는 면에서 중요한 차이점을 발생시키게 된다. 명목형 속성에 대해서는 상위노드에서 선택되었다면 하위 노드에서는 다시 고려할 필요가 없다. (즉, 다시 선택되지 않는다.)  반면에 수치형 속성에 대해서는 같은 속성에 대해서도 계속적인 분할이 발생할 수 있다. 명목형 속성은 루트노드로부터 해서 단 한번만 선택될 수 있지만, 수치형 속성의 경우 여러 차례 (하위노드에서) 테스트될 수 있다. 이러한 방식으로 생성된 트리는 복잡하고 이해하기 어려울 수 있다. 왜냐하면 하나의 수치형 속성에 대한 기준이 한꺼번에 나타나지 않고, path에 걸쳐서 분산되어 나타나기 때문이다. 이러한 문제를 개선한 만들기는 어렵지만 이해하기 좋은 트리를 생성하기 위해서는 상위노드로부터 여러 개의 기준값을 선정하여 다중 분할하는 multiway ( <-> binary-way) 테스트를 사용한다. 이 외에, 성능은 조금 떨어지지만 단순한 방식인 이산화 방식을 7.2절에서 설명한다.

|  |
| --- |
| **정리** \* 수치형 데이터에 대해서는 binary 분할 방식을 사용한다. (다중 분할 보다는)  \* binary 분할에서는 하나의 breakpoint를 찾아야 한다.  \* 가능한 모든 값에 대하여 breakpoint를 설정하고 Info 값을 계산해 본다. (시간이 많이 걸림)    가장 좋은 (info가 가장 작은 분할 값을 선택한다.)  \* 범주형 속성의 경우는 하나의 path에서 한번만 사용되지만, 반면에 수치형 속성은 여러차례    나타날 수 있다. 흩어져서 속성이 나타나므로 트리의 모양이 복잡하고 이해하기 어려울 수 있다.  \* 따라서, 이에 대한 보완책으로 다중 분할 방식을 사용할 수 도 있다.  \* 미리 수치형 값을 이산화 처리하여 사용하는 방식도 있다.  \* (정렬하는 방법에 대해서 많은 부분을 할애하고 있으나 많이 중요한 부분은 아닌 것 같다. 구현시    시간을 줄여주는 방법이며 핵심적인 아이디어는 아니다.) |

|  |  |
| --- | --- |
| http://cfile28.uf.tistory.com/image/1202FB164AFC2027017C86 | 그림1 |

|  |  |
| --- | --- |
| http://cfile2.uf.tistory.com/image/111CDC184AFC2038A2054F | 그림2 |

   위 데이터를 ID 알고리즘에 적용하면 어떤 결과가 나올까? 우리는 이 데이터를 통해 어떤 날씨가 경기하기에 좋고, 또 어떤 날씨가 경기하기에 안 좋은 날씨인가에 대한 지식을 얻고 싶다. ID3 알고리즘을 지원하는 다양한 데이터마이닝 프로그램들이 있지만, 무료로 공개되어 있는 weka 프로그램을 작성하여 수행한 결과는 아래와 같다.

|  |  |
| --- | --- |
| http://cfile24.uf.tistory.com/image/122E0A184AFC204B991538 | 그림3 |

   위 그림이 갖는 의미를 해석해보자. 왼쪽부터 순서로 해석하면 아래와 같은 규칙으로 된 지식들을 얻을 수 있다. 해석하는 방법은 단순하므로 3개 정보만 기록하였다. 물론, 해석결과가 일반적이지 않을 수 있지만 주어진 데이터에서는 적합한 규칙들이다. (아마도 실내 경기장이 아닐까?)   
   
 ▪ 날씨가 맑고(outlook=sunny) 습도가 낮으면(humidity=low) 운동을 한다. (play=yes)  
 ▪ 날씨가 맑고(outlook=sunny) 습도가 낮으면(humidity=high) 운동을 한다. (play=no)  
 ▪ 날씨가 흐리면(outlook=overcast) 운동을 한다. (play=yes)

**2. Entropy 계산 방법**

참고 : (1) Data Mining (IAN.H) p.93-94. (2) Concept & Tech(한글판) p.356.

   ID3 알고리즘의 자세한 동작 방법, 절차는 본 장의 마지막에서 설명하기로 하고, 먼저 필요한 수식들을 학습하도록 하자. 본 절에서는 Entropy 개념 및 계산 방법을 학습한다.

   엔트로피(entropy)는 주어진 데이터 집합의 혼잡도를 의미한다. 즉, 주어진 데이터 집합에 레코드들이 서로 다른 종류(클래스)들이 많이 섞여있으면 엔트로피가 높고, 같은 종류(클래스)의 레코드들이 많이 있으면 엔트로피가 낮다. 곧 소개할 엔트로피 계산식에서는 엔트로피 값은 0에서 1사이의 값을 갖는다. 가장 혼잡도가 높은 상태의 값이 1이며, 하나의 클래스로만 구성된 상태의 값이 0이다. 의사결정트리 분류 알고리즘에서는 엔트로피가 높은 상태에서 낮은 상태가 되도록 데이터를 특정 조건을 찾아 나무 모양으로 구분해 나간다.

|  |  |
| --- | --- |
| http://cfile2.uf.tistory.com/image/2065E4174AFC205D6FE2CD | 그림4 |

   바로 전 절에서 소개한 그림을 예로 들어 설명해보자. 매 위의 노드(outlook이라고 써있는 노드)의 엔트로피 값이 가장 크며 그 하위의 3개의 노드(outook에 의해 구분된 하위 노드)들의 엔트로피가 더 낮을 것이다. 또 그 하위에 있는 노드들은 부모노드 보다 엔트로피 값이 작을 것이다.

   엔트로피 값은 아래의 식에 의하여 계산된다. 이 식은 주어진 데이터셋 S에 대한 엔트로피 값을 구하는 계산식이다. 엔트로피 값은  즉 각 클래스 값의 포함 비율에 로그를 적용하고 다시 값을 가중치로 곱한 값을 모두 더하는 식이다. log2 함수 적용을 통해 마이너스(-) 값이 나타나므로 전체 수식 값에 - 를 붙여주어 0 에서 1 사이의 값을 갖도록 한다.

|  |
| --- |
| http://cfile221.uf.daum.net/image/20620E154A4242132A9845 |

  위 식에서 는 i번째 클래스 값이 대하여 해당 데이터집합 중에서 차지하는 비율(Probability)을 의미한다. 각 변수들이 갖는 의미는 아래와 같다.

|  |
| --- |
| **S**: 주어진 데이터들의 집합  **C = {C1, C2, ... , Ck}** : 클래스 값들의 집합  **freq(Ci,S)** : S에서 class Ci에 속하는 레코드의 수  **|S|** : 주어진 데이터들의 집합의 데이터 개수 |

    그러면, 간단한 예를 가지고 엔트로피 값을 계산해보자. 10명의 회원이 있는데 5명이 우수회원, 5명이 일반회원이라고 하자. 그러면, 데이터가 완전히 혼합되어 있으므로 엔트로피 값은 1이 된다. 반면, 10명 모두 일반회원이라면 전혀 혼합되어 있지 않으므로 엔트로피 값은 0 이 된다.

  10명의 회원이 있는데 5명이 우수회원, 5명이 일반회원인 경우 엔트로피 계산식에 적용하면 아래와 같다.

|  |  |
| --- | --- |
| http://cfile213.uf.daum.net/image/1654B5194A4243AF4BB611 | 식1 |

    
   10명의 회원이 있는데 10명이 우수회원, 0명이 일반회원인 경우 엔트로피 계산식에 적용하면 아래와 같다.

|  |  |
| --- | --- |
| http://cfile233.uf.daum.net/image/165827164A42447E78444E | 식2 |

  이해를 위하여 다른 예를 들어보자. 노드에 데이터가 총 14개 있는데 클래스가 Yes인 레코드가 9개, No인 클래스가 5개 있다고 가정하자. 그러면 아래와 같이 계산할 수 있다.

|  |  |
| --- | --- |
| http://cfile233.uf.daum.net/image/155927164A4245586E1071 | 식3 |

   물론, 클래스가 2개 이상일 경우에도 같은 방식으로 적용하면 된다. 클래스의 값이 a 또는 b 또는 c의 3가지인 경우는 아래와 같이 계산될 수 있다.

|  |  |
| --- | --- |
| http://cfile233.uf.daum.net/image/204D6E174A4245E34B1B4E | 식? |

**3. Information Gain 계산 방법**

  “정보 이득(information gain)”이란 어떤 속성을 선택함으로 인해서 데이터를 더 잘 구분하게 되는 것을 의미한다. 예를 들어, 학생 데이터에서 수능등급을 구분하는데에 있어 수학 점수가 체육 점수보다 변별력이 더 높다고 하자. 그러면 수학점수 속성이 체육 점수 속성보도 정보이득이 높다라고 말할 수 있다. 정보이득을 계산하기 위해서는 먼저 엔트로피 개념과 계산방법을 이해해야 한다. 따라서, 정보이득 계산방법을 설명하기 전에 엔트로피 계산식을 살펴보기로 하자.

정보 이득의 계산식은 아래와 같다.

|  |  |
| --- | --- |
| http://cfile219.uf.daum.net/image/116A33194A4248AC355503 | 식 5 |

   여기서,       , I(s1,s2,...,sm) 은 원래 상위 노드의 엔트로피를 의미한다. 즉, 상위노드의 엔트로피에서 하위노드의 엔트로피를 뺀 것이다. 그리고 E(A)는 A라는 속성을 선택했을 때 하위로 작은 m개의 노드를 나누어진다고 하면 하위 각 노드의 엔트로피를 계산한 후 노드의 속한 레코드의 개수를 가중치로 하여 엔트로피를 평균한 값이다.

  식 Gain(A) 의 의미는 ... 속성 A를 선택했을 때의 정보이득 양을 계산하는 수식으로써 ... 원래 노드의 엔트로피를 구하고 A라고 하고, 속성 A를 선택한 후의 m개의 하위 노드로 나누어진 것에 대한 전체적인 엔트로피를 구한 후 그것을 B라고 하면, A-B를 의미한다. 이 값이 클수록 정보 이득이 큰 것이고 해당 속성 A가 변별력이 좋다는 것을 의미한다.

|  |  |
| --- | --- |
| http://cfile232.uf.daum.net/image/16667D184A424D876AB435 | 그림 5 |

각 노드의 Entropy를 구해보자.

|  |  |  |
| --- | --- | --- |
| |  | | --- | | http://cfile5.uf.tistory.com/image/163EA3114B0286623FBD12 | | 그림 6 |

 각 노드의 엔트로피를 계산하였으면, 속성 A에 의한 Information 값을 구할 수 있다.

|  |  |
| --- | --- |
| http://cfile235.uf.daum.net/image/144679174A424B56656C11 |  |

**\* 참고자료 (References)**