인공지능

- 사람이 의식적으로 하는 행동을 컴퓨터가 할 수 있도록 하는 것

* 강한 인공지능

- 사람과 같은 지능
- 마음을 가지고 사람처럼 느끼면서 지능적으로 행동하는 기계
- 추론, 문제 해결, 판단, 계획, 의사소통, 자아 의식, 감정, 지혜, 양심

* 약한 인공지능

- 특정 문제를 해결하는 지능적 행동
- 사람의 지능적 행동을 흉내 낼 수 있는 수준
- 대부분의 인공지능 접근 방향
- 중국인 방 사고 실험

* Symbolic AI

- 프로그래머가 이해하는 데이터 구조에 맞게 지능을 모델화 함
- 게임, 이론 증명
- 기호 및 이산형 변수
- 장점 : 더 유용한 로직, 쉬운 디버그, 설명, 크지않은 데이터에 유용, 추상문제에 적합

* Subsymbolic AI

- 뉴런과 유사한 수준의 모델링
- 장점 : 소음에 강함, 사전지식 없어도 됨, 빅데이터에 유용, 지각문제에 유용

논리표 *

¬ : 부정 ∨ : 논리합 ∧ : 논리곱 → : 함의 ≡ : 논치

P	Q	¬P	P ∨ Q	P ∧ Q	P → Q	P ≡ Q
F	F	T	F	F	T	T
F	T	T	T	F	T	F
T	F	F	T	F	F	F
T	T	F	T	T	T	T

불확실성의 원인

1) 약한 인과성이나 애매한 인과관계 (베이즈 정리, 확신도 표현)
2) 부정확한 언어 사용 (퍼지 이론 사용표현)
3) 불완전하거나 결손된 데이터에 기반한 지식 (근사적추론, unknown으로 간주)
4) 상충되는 지식의 통합 (지식 소스별로 가중치 부여)

베이즈 정리

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}$$

- $P(B|A)$: 가능도
- $P(A)$: 사전확률
- $P(A|B)$: 사후확률
- $P(B)$: 증거

질병A의 발병률 = 0.1% , 질병이 있을때 질병이 있다고 검진할 확률 99% , 질병이 없을때 질병이 없다고 검진할 확률 = 98% , 만약 질병이 있다고 검진 받았을때 실제 질병에 걸릴확률?

$P(A)$ = 질병에 걸릴확률 = 0.001 $P(B)$ = 질병이 있다고 검진할 확률

$P(B|A) = 0.99$ $P(B^c|A^c) = 0.98$ $P(A|B) = ?$: $\frac{P(A \cap B)}{P(B)} = \frac{0.00099}{0.02097} = 0.0472 = 4.72\%$

$P(B) = P(B|A)\,P(A) + P(B|A^c)\,P(A^c)$

$= 0.99 \times 0.001 + 0.02 \times 0.999 = 0.02097$

0.00099 0.01998

DATE. NO.

건빵 2봉지를 샀다 첫번째 봉지의 박사탕에서 흰색 10개 분홍색 30개가 나왔고 두번째 봉지 박사탕에서는 흰색 20개 분홍색 20개가 나왔다 총 80개의 박사탕을 모아놓고 눈 감고 하나 뽑았을때 분홍색이 나왔다. 이 박사탕이 첫 번째 봉지에서 나왔을 확률은?

$P(A)=$ 박사탕이 분홍색 $=\frac{50}{80}=\frac{5}{8}$ $P(B)=$ 첫번째 봉지 $=\frac{40}{80}=\frac{1}{2}$

$P(B|A)=?$ $P(A|B)=\frac{30}{40}=\frac{3}{4}$ $P(A\cap B)=\frac{3}{8}$ $P(A\cap B^c)=\frac{2}{8}$.

$=\frac{P(B\cap A)}{P(A)}=\frac{3}{8}\times\frac{8}{5}=\frac{3}{5}=60\%$

-오류가 생기는 이유

실제 베이즈 접근법을 사용하려면 어떤 가설과 부정 하에서 증거의 조건부 독립을 포함하여 많은 가정을 충족해야 함

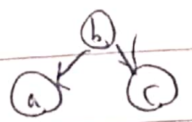
보세미 문제는 이런 가정을 제대로 만족 못함

베이즈 정리 적용시 - 증거의 조건부 독립성, 믿을 만한 통계 데이터, 각 가설에 대한 사전 확률이 필요

Probabilistic Graphical model

선형

분기

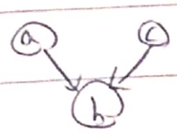


b를 모르면 a와 c는 독립이 아니고

b를 알면 a와 c는 독립임

b를 알면 a←b→c는 체인이 폐쇄 된다

합류



b를 모르면 a와 c는 독립 (분기와 반대)

b를 알면 a와 c는 독립이 아님이 성립됨

b를 알면 a→b←c 체인은 열린다

체인의 폐쇄

a와 c를 잇는 체인 a→c 에서 노드집합 W를 주어질때 다음을 하나라도 성립하면 폐쇄됨

선형: 머리~꼬리 노드에 속할때

분기: 꼬리-꼬리노드에 속할때

합류: 체인에 머리-머리 노드가 없을때, 이 노드와 노드의 자손 모두 W에 안속할때

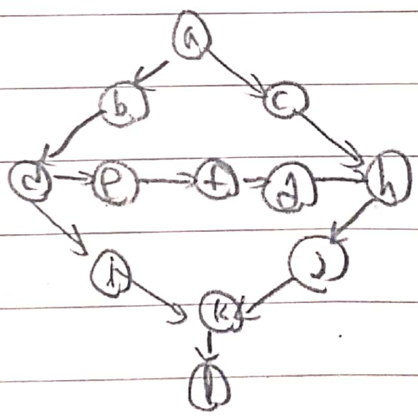
d→e→f→g→h 는 W={f,a}에 의해 폐쇄됨 (선형)

d→b←a→c→h 는 W={f,a}에 의해 폐쇄됨 (분기)

d→i→k←j←h 는 W={f,a} 에 의해 폐쇄됨 (합류) 모르기때문에 (k,l)은

d→i→k←j←h 는 W={f,a,k}에 의해 폐쇄안됨

d→i→k←j←h는 W={f,a,k,j}에 의해 폐쇄됨 (분기 j가 있어서)

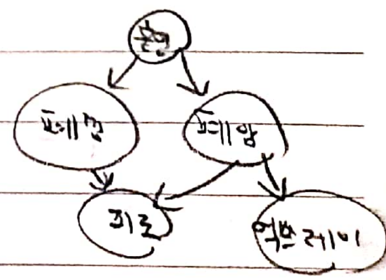


d-분리

두노드 a와 c사이에 있는 모든 체인이 노드집합 W에서 폐쇄되면 두노드는 d-분리 되었다고 함

d-sep(a,c|W)로 표시 이 정의는 집합으로 확장하여 d-sep(A,C|W)도 가능

d-분리면 조건부 독립



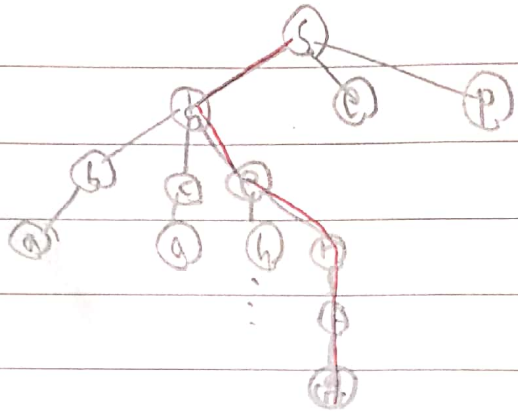
d-sep(흡연, 피로|폐암) 아니다. 흡연→폐염→피로가 열림

d-sep(흡연, 피로|{폐렴,폐암})이다

d-sep(폐렴, 엑스레이|피로) 아니다 폐렴→피로←폐암→엑스레이 가 열림

1 DFS (Depth-First-Search)

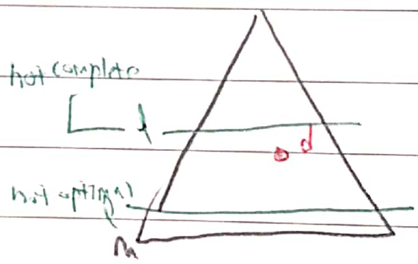
깊이 우선탐색
후입선출 (stack) 사용
가장 먼저방문된 child를 따라서 쭉
가다 정답이아니면 상위노드로 바뀜



complete = (m이 유한하면) complete 함
optimal = X
Time = $O(b^m)$
space = $O(bm)$

2. DLS (Depth-Limited Search)

깊이 제한 탐색
깊이 L을 제한하여 탐색
DFS 방식에 Limit을 줌



Complete = No, if $l < d$
Optimal = No, if $l > d$
Time = $O(b^l)$
space = $O(bl)$

3 Iterative deepening Search (Iterative Deepening Depth-First-Search) (IDS)

DLS 에서 L값을 하나씩 늘려감
DFS 에서 L=1 일때   노드수 = $(l+1)b^0 + lb^1 + \cdots 2b^{l-1} + 1b^l$
DFS 에서 L=2 일때

complete : Yes
optimal : Yes (cost가 같으면)
Time = $O(b^d)$: $(d+1)b^0 + db^1 + \cdots 2b^{d-1} + b^d = O(b^d)$
Space = $O(bl)$

DATE. NO.

4 BFS (Breadth-First-Search)

너비우선 탐색

FIFO 선입선출 (Queue) 사용

Complete = Yes (유한트리)

Optimal = Yes (Cost가 모두 같을때)

Time = $O(b^{d+1}) = 1+b+b^2+\cdots b^{d-1}+b^d = O(b^{d+1})$

Space = $O(b^d)$

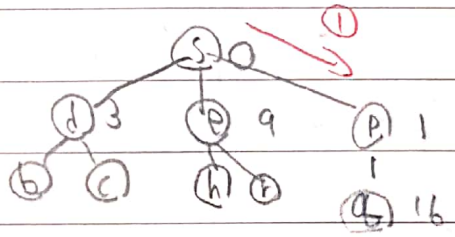
5 UCS (Uniform Cost search)

: 가장 값이 적은 노드로 확장

priority queue 이용 (우선순위 큐 이용)

BFS에서 sort하는 느낌 step이 같으면 BFS임

: 모든 / 최소 step값 (C^*/ϵ)



Complete = Yes, (유한트리)

Optimal = Yes

Time = $O(b^{1+[C^*/\epsilon]})$

Space = $O(b^{1+[C^*/\epsilon]})$

	BFS	UCS	DFS	DLS	IDS
Complete	Yes	Yes	Yes (Finite)	No	Yes
Optimal	Yes	Yes	No	No	Yes
Time	$O(b^{d+1})$	$O(b^{[C^*/\epsilon]+1})$	$O(b^m)$	$O(b^l)$	$O(b^d)$
Space	$O(b^d)$	$O(b^{[C^*/\epsilon]+1})$	$O(bm)$	$O(bl)$	$O(bd)$

Informed search

Greedy Search

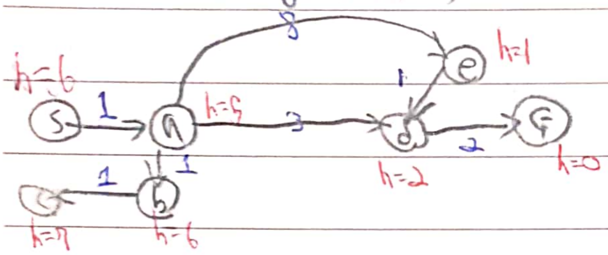
목적지와 가장 가깝다고 생각하는 곳으로 확장

Heuristic: 목적지와 현재거리가 가까운 정도

A* Search (★★★)

UCS + Greedy   Cost와 Heuristic의 합친값

$A^* = f(n) = g(n) + h(n)$

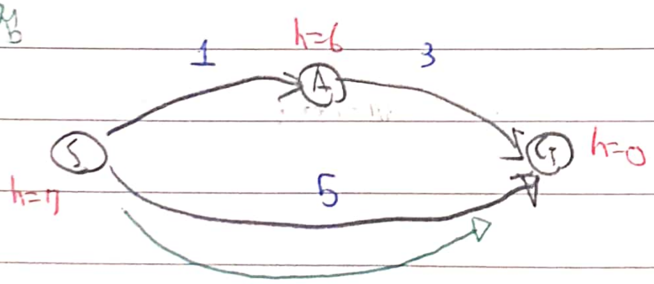


UCS = S → a → b → c

Greedy = S → a → e → d → G

A* = S → a → d → G

문제점



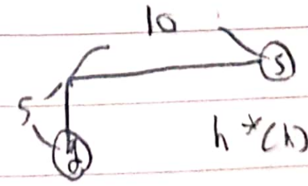
A*   not optimal   ※ h(n) 값이 실제 cost 보다 낮은값을 가져야함

Admissible Heuristic

$0 \le h(n) \le h^*(n)$

$h^*(n)$ = 목적지로 가는 실제값

ex)



$h^*(n) \geq 15$

Optimality of A* search

증명: B가 목적지에 있고 A도 B만의 목적지에 있다고 가정

Claim: n은 B 이전에 확장될 것이다.

1. $f(n) \le f(A)$ → $f(n) = g(A) + h(n)$ $f(n) \le g(A)$ $g(A) = f(A)$ ($h=0$)
2. $f(A) < f(B)$ → $g(A) < g(B)$ → $f(A) < f(B)$
3. n은 B전에 확장된다 → $f(n) \le f(A) < f(B)$

모든 노드는 B이전에 A를 먼저 확장한다

A 확장 후 B를 확장한다.

A*는 optimal 하다 (admissible Heuristic 일때)

A* properties

Complete: Yes

Optimal: Yes (if Heuristic is admissible)

Time: $O(b^d)$ (but 좋은 Heuristic이 주어지면 많은 시간 단축)

Space: 각각의 모든 노드를 모두 메모리에 저장해야 함 (A*의 단점)

그래프 서치에서 A*

① Admissible = heuristic cost ≤ actual cost to goal

$h(A) \le$ A to G

Consistency: $h(A) - h(C) \le$ cost (A to C)

↳ $h(A) \le$ cost(A to C) + $h(C)$ 하면 A* graph search는 optimal

Optimality

Tree search:

A*: heuristic이 admissible 할때 optimal

UCS: $h=0$ 일때 optimal

Graph search:

A*: heuristic이 consistent 할때 optimal

UCS: optimal

morning glory

CamScanner로 스캔하기

Adversarial Search (Mini-Max)

나의 차례에서 Max 상대 차례에 Min을 고르는 것

Complete = Yes

optimal = Yes (If optimal 한 적이면)

Time = $O(b^m)$ 철저한 DFS

Space = $O(bm)$

Alpha-Beta Pruning (Improving Mini-Max)

Mini-Max에서 필요없는 트리 제거

Time = 최악 $O(b^m)$

최선 $O(b^{m/2})$

Local Search Algorithms

장점: 메모리를 거의 안씀

효과적인 방법 제공, 빠르고, 효과적 (but Incomplete and suboptimal)

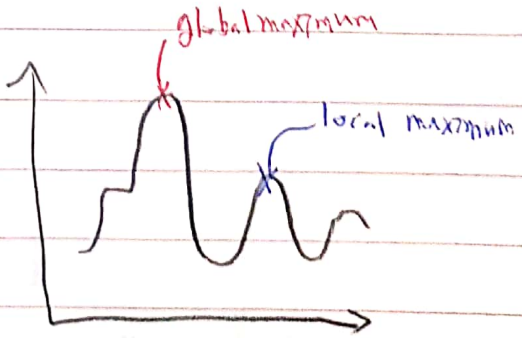
Complete X optimal X

Hill-climbing Search

주변 state를 보고 좋은 쪽으로 감

문제점: local maxima에 빠질 수 있음

주변값은 evaluation 함수 비교



DATE. NO.

Optimization (최적화기법)

- 파라미터 찾기 (search space)

- Function optimization (continuous optimization)
  - gradient descent

- combinatorial optimization (discrete optimization)
  - 유전 알고리즘 for (TSP)

Gradient Descent Method

경사하강법

함수의 미분방향으로 결정하여감

Goal: $\text{minimize}_x f(x)$

반복: $x_{k+1} = x_k - \epsilon_k \nabla f(x_k)$

☆ Stepsize matters (학습률) : 너무세면 못찾고 적으면 오래걸림

CamScanner로 스캔하기