1 Strojno ucenje

1.1 Evalviranje hipotez

Pomembni kriteriji:

- **konsistentnost** hipotez z primeri (ucnimi)
- **splosnost** (tocnost za nevidene primere)
- **razumljivost** hipotez

TP=true positive, TN-true negative, FP-false positive (napaka 1. tipa), FN-false negative (napaka 2. tipa)

Klasifikacijska tocnost $= \frac{TP+TN}{TP+TN+FP+FN} = \frac{TP+TN}{N}$

Obcutljivost/senzitivnost $= TPR = \frac{TP}{TP+FN}$

1.1.1 Binarizacija atributov

Aleternativa za resevanje problematike z vecvrednostnimi atributi: Strategije (za primer B = {Y, G, R, B}):

- $[\{Y\},\{R,G,B\}]$ (one-vs-all)
- $[\{Y,R\},\{G,B\}]$
- vpeljava bianrnih atributov za vsako barvo

Primer B = {Y, G, R}, konstruiramo 3 nove binarne atribute:

barva		Y	G	R
Y	→	1	0	0
G		0	1	0
R		0	0	1

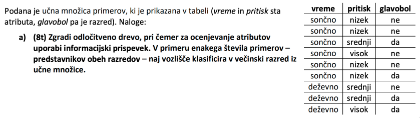
Prednost: manjse vejanje drevesa.

1.2 Gradnja odlocitvenih dreves

Drevo gradimo tako da v vsakem koraku izberemo atribut, ki najvec zmanjsa entropijo (ima najvecji informacijski prispevek).
Rezidualna entropija (najbolsi atribut -> najnizja)

$H_{rez}(A) = -\sum_{a_i \in A} p(A=a_i) \sum_{c_i \in C} p(C=c_i|A=a_i)\log_2 p(C=c_i|A=a_i)$

Informacijski prispevek (najboljsi atribut maksimizira)
Gain(A) $= H(\text{Class}) - H_{rez}(A)$
Razmerje inofrmacijskega prispevka atributa A:
IGR(A) $= \frac{\text{Gain(A)}}{\text{H(A)}}$


Podana je učna množica primerov, ki je prikazana v tabeli (*vreme* in *pritisk* sta atributa, *glavobol* pa je razred). Naloge:

a) **(8t) Zgradi odločitveno drevo, pri čemer za ocenjevanje atributov uporabi informacijski prispevek. V primeru enakega števila primerov – predstavnikov obeh razredov – naj vozlišče klasificira v večinski razred iz učne množice.**

vreme	pritisk	glavobol
sončno	nizek	ne
sončno	nizek	ne
sončno	srednji	da
sončno	visok	ne
sončno	nizek	ne
sončno	nizek	da
deževno	srednji	ne
deževno	srednji	da
deževno	visok	da

$H(G) = -\frac{5}{9}\log_2\frac{5}{9} - \frac{4}{9}\log_2\frac{4}{9} = -0.991$

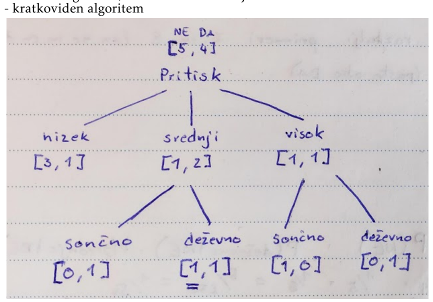
$H_{rez}(V) = \frac{3}{9}(H(1/3,2/3)) + \frac{6}{9}(H(4/6,2/6)) = -0.918$

$H_{rez}(P) = \frac{4}{9}(H(1/4,3/4)) + \frac{3}{9}(H(2/3,1/3)) + \frac{2}{9}(H(1/2,1/2)) = 0.89$

$H_{rez}(P) < H_{rez}(V) \rightarrow$ Pritisk je najboljsi atribut z njim naprej delimo...

1.2.1 TDIDT (Top down induction decision tree) algoritem

Pozresen algoritem, ki **lokalno** izbira najbolsi atribut.
- kratkoviden algoritem



Nizek pritisk nismo delili naprej, saj nimamo nobenega primera ki bi imel nizek pritisk in dezevno vreme (samo soncno). Ce bi delili bi dobili razreda soncno: [3, 1] in dezevno: [0, 0] kar nebi imelo smisla.

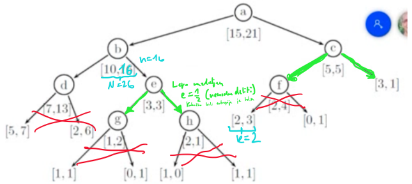
1.3 Ucenje iz sumnih podatkov (rezanje)

Ucna mnozico razbijemo na: 70% za gradnjo, 30% za rezanje. Z rezanjem odstranimo poddrevesa, ki niso kriticna in so redundantna tako **zmansamo velikost drevesa**.

tocnost t...verjetnost pravilnosti klasifikacije
napaka e ... $1-t$
relativna frekvenca $p = \frac{n}{N}$
m-ocena $p = \frac{n+p_a*m}{N+m}$
m... koliko zaupam apriorni verjetnosti
p_a apriorna verjetnost (pove domenski ekspert) (v nasem primeru relativna frekvenca)
Laplacova ocena verjetnosti $p = \frac{n+1}{N+k}$
k...stevilo vseh moznih razredov

1.3.1 MEP (Minimal Error Prunning)

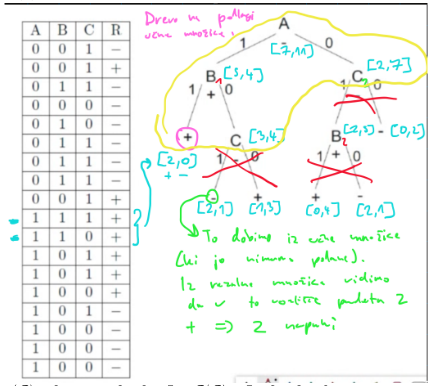
e=staticna napaka, E=vzvratna napaka, $e \leq E \rightarrow$ rezemo poddrevo



Primer z (Laplace)
$e_L(d) = 1 - t = 1 - \frac{13+1}{20+2} = 0.363$
$E_L(d) = 12/20 \cdot e_L(d_l) + 8/20 \cdot e_L(d_d) = \frac{12}{20}\cdot(1-\frac{7+1}{12+2}) + \frac{8}{20}(1-\frac{13+1}{20+2})$

1.3.2 REP (reduced error prunning)

G(v)=st. napacnih klasifikacij v poddrevesu - st. napacnih klasifikacij v korenu poddrevesa
$G(v) \geq 0 \Rightarrow$ rezemo podrevo

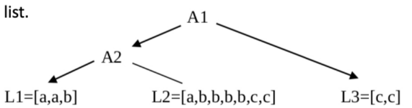


A	B	C	R
0	0	1	−
0	0	1	+
0	1	1	−
0	0	0	−
0	1	0	−
0	1	1	−
0	1	1	−
0	1	1	−
0	0	1	+
1	1	1	+
1	1	0	+
1	0	1	+
1	0	1	+
1	0	0	+
1	0	1	−
1	0	0	−
1	0	0	−
1	0	0	−

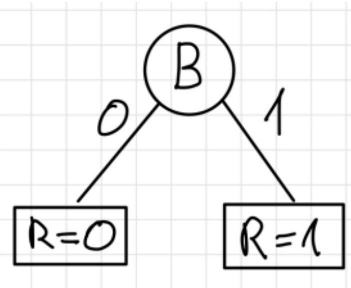
$e(C) = 3, \quad e_T = 2+3=5, \quad G(C) = 5-3 = 2 \geq 0 \rightarrow$ rezemo

1.4 Ocenjevanje uspesnosti modelov

tocnost t ... verjetnost pravilnosti klasifikacije
Laplacova ocena verjetnosti $p = \frac{n+1}{N+k}$
k...stevilo vseh moznih razredov



$t_{L1} = \frac{2+1}{3+3} = 0.5, t_{L2} = \frac{4+1}{7+3} = 0.5, t_{L3} = \frac{2+1}{2+3} = 0.6$
tocnost drevesa: $t_D = 3/12 \cdot 0.5 + 7/12 \cdot 0.5 + 2/12 \cdot 0.6 = 0.5167$



$e = 1 - (P(B=0)P(R=0|B=0) + P(B=1)P(R=1|B=1))$

1.5 Obravnava mankajocih atributov, naivni Bayesov klasifikator

1.5.1 Naivni Bayes

Ce poznamo razred, kam klasificiramo ce nepoznamo atributov:

Klasifikator: $\text{argmax}_{c\in C} P(c) \prod_{i=1}^{n} P(x_i|c)$

c...razred, x_i...atributi
Verjetnost::

$P(C=c|x_1,\dots,x_n) = \frac{P(C=c)P(X_1=x_i|C=c)P(X_2=x_j|C=c)\dots}{P(X_1=x_i)P(X_2=x_j)\dots}$

Primer pritisk, vreme, razred glavobol

X\Y	ne	da
P(G)	P(G=ne) = 5/9	P(G=da) = 4/9
P(P=srednji)	P(P=sre|G=ne)=1/5	P(P=sre|G=da)=2/4
P(V=dezevno)	P(V=dez|G=ne)=1/5	P(V=dez|G=da)=2/4
$P(y)\prod_{i=1}^{n} P(x_i\|y)$	$5/9\cdot 1/5 \cdot 1/5$	$4/9\cdot 2/4\cdot 2/4$

1.5.2 Nomogrami

Ciljni razred $C = c_T$

$X_{X_i=x_j} = \ln\left(\frac{\frac{P(C=c_T|X_i=x_j)}{P(C=\overline{c_T}|X_i=x_j)}}{\frac{P(C=c_T)}{P(C=\overline{c_T})}}\right)$

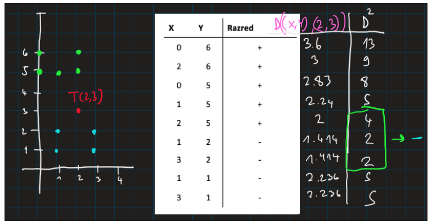
PRIMER: Napisi nomogram za verjetnostno razlago modela za klasificiranje v razred **bolezen=da**

		bolezen	
		DA	NE
družina	da	200	150
	ne	120	110
spol	moški	140	160
	ženska	180	100

$X_{D=da} = \ln\left(\frac{\frac{200}{150}}{\frac{320}{260}}\right) = 0.08$
$X_{D=ne} = -0.121$
$X_{S=m} = -0.341$
$X_{S=z} = 0.380$

Da ima oseba **bolezen** najbolj pripomoreta, da je oseba **zenska** in ima **druzino**.

1.6 K-najblizjih sosedov



X	Y	Razred	D((x,y),(2,3))	D²
0	6	+	3.6	13
2	6	+	3	9
0	5	+	2.83	8
1	5	+	2.24	5
2	5	+	2	4
1	2	-	1.414	2
3	2	-	1.414	2
1	1	-	2.236	5
3	1	-	2.236	5

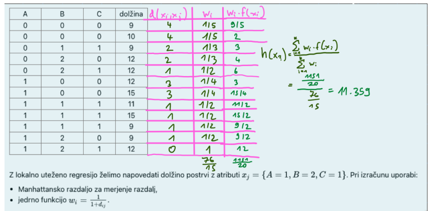
1.7 Regresija

Ucni primeri so podani/oznaceni kot **vrednosti vhodov in izhodov**.
$(\vec{x}_1,\vec{y}_1),(\vec{x}_2,\vec{y}_2),\dots,(\vec{x}_N,\vec{y}_N)$
$\vec{x}_i$ ... atributi, $\vec{y}_i$ ... ciljna spremenljivka
Locimo dve vrsti problemov:

1. **Klasifikacijski problemi** - y_j diskretna
2. **Regresijski problemi** - y_j zvezna

1.7.1 Lokalno utezena regresija

$h(\vec{x}_?) = \frac{\sum_{i=1}^{k} w_i \cdot f(\vec{x}_i)}{\sum_{i=1}^{k} w_i}$, $w_i(d)$...utez



A	B	C	dolžina	$d(x_i,x_j)$	w_i	$w_i \cdot f(x_i)$
0	0	0	9	4	1/5	9/5
0	0	0	10	4	1/5	2
0	1	1	9	2	1/3	3
0	2	0	12	2	1/3	4
0	2	1	12	1	1/2	6
1	0	0	12	3	1/4	3
1	0	0	15	3	1/4	15/4
1	1	1	11	1	1/2	11/2
1	1	1	15	1	1/2	15/2
1	1	1	9	1	1/2	9/2
1	2	0	9	1	1/2	9/2
1	2	1	12	0	1	12
					76/15	1151/20

$h(x_?) = \frac{\sum_{i=1}^{k} w_i \cdot f(x_i)}{\sum_{i=1}^{k} w_i} = \frac{\frac{1151}{20}}{\frac{76}{15}} = 11.359$

Z lokalno uteženo regresijo želimo napovedati dolžino postrvi z atributi $x_j = \{A=1, B=2, C=1\}$. Pri izračunu uporabi:

- Manhattansko razdaljo za merjenje razdalj,
- jedrno funkcijo $w_i = \frac{1}{1+d_{ij}}$.

1.7.2 Regresijska drevesa

Linearna regresija je poseben primer regresijskega drevesa.
V listih regresijskega drevesa vcasih napovemo kar povprecno vrednost.

1.8 Unsupervised learning

Ucni primeri niso oznaceni (nimajo ciljne spremenljivke), **ucimo se vzorcev v podatkih**, (npr. grucenje)

1.8.1 Hierarhicno grucenje

Poveze po podobnosti med primeri, primer zacne kot samostojna gruca, na koncu vsi primeri pripadajo eni gruci
Dendrogram: drevo, ki predstavlja grucenje.
Single-linkage: povezava med grucami je najkrajsa razdalje med primeroma iz razlicnih gruc.
Complete-linkage: najdaljsa razdalja med primeroma iz razlicnih gruc.
Average-linkage: povprecna razdalja med primeroma iz razlicnih gruc.
Tocke A(3,1),B(1,2),C(3,4),D(5,2),E(1,1), manhattan, complete linkage:

	A	B	C	D	E
A	0	3	3	3	2
B		0	4	4	$\underline{1}$
C			0	4	5
D				0	5
E					0

→

	A	BE	C	D
A	0	$\underline{3}$	3	3
BE		0	5	5
C			0	4
D				0

→

	ABE	C	D
ABE	0	5	5
C		0	$\underline{4}$
D			0

→

	ABE	CD
ABE	0	5
CD		0

1.8.2 K-means

1. V prostor dodamo k centroidov, ki predstavljajo gruce.
2. Izracunamo ketri centroid je najblizji vsakemu primeru.
3. Izracunamo nove centre gruc $= \frac{1}{|G|}\sum_{i\in G} x_i$
4. Ponovimo korake 2 in 3 dokler se centri ne premaknejo.

V mnozici tock A(3,1),B(1,2),C(3,4),D(5,2),E(1,1), manhattanska razdalja, zacetni vrednosti centroidov C1(4,4) in C2(5,4).

Tocka	d(X,C1)	d(X,C2)	Gruca
A	4	5	C1
B	5	6	C1
C	1	2	C1
D	3	2	C2
E	6	7	C1

V naslednji iteraciji sta koordinati centroidov:
$C1 = (\frac{3+1+3+1}{4}, \frac{1+2+4+1}{4}) = (2,2)$ in $C2 = D = (5,2)$ ...

1.9 Spodbujevalno ucenje - reinforcement learning

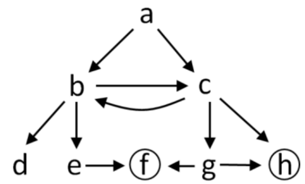
Inteligentni agent se uci iz zaporedja **nagrad** in **kazni**

1.10 Ocenjevanje ucenja

k-fold, celo ucno mnozico razbij na k disjunktnih podmnozic za vsako od k podmnozic uporabi mnozico kot testno mnozico, preostalih k-1 mnozic kot ucno mnozico.

2 Preiskovanje

2.1 Neinformirani preiskovalni algoritmi



2.1.1 Iskanje v globino

Cikle crtamo...

Razvijano	Generirano	DFS list
/	a	a
a	b c	b c
b	c' d e	c' d e c
c'	(b') g h	g h c' d e c
g	f h'	f h' g h c' d e c
f	koncno	koncno

Izboljsave (**Iskanje s sestopanjem, iterativno poglabljanje**)

2.1.2 iskanje v sirino

Razvijano	Generirana	Vrsta
a	b c	[b,c]
b	c' d e	[c,c',d,e]
c	b' g h	[c',d,e,b',g,h]
c'	(b") g' h'	[d,e,b',g,h,g',h']
d	/	[e,b',g,h,g',h']
...		
h	koncno	koncno

2.1.3 iterativno poglabljanje

problem gobinsko omejenega iskanja -> nastavitev meje l Mejo l postopoma povecujemo za 1, dokler ne najdemo resitve.

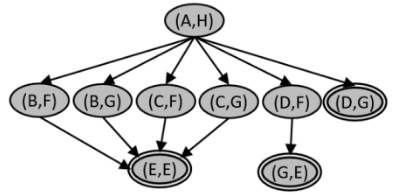
- **popolnost**: Da
- **optimalnost**: Da
- **casovna zahtevnost** $O(b^d)$
- **prostorska zahtevnost** $O(bd)$

Boljse od iskanja v globino/sirino

2.1.4 dvosmerno iskanje

Pozenemo vzporedni iskanji od zacetka do cilja in od cilja do zacetka.
Implementacija dvosmernega iskanja:

- ciljno vozlisce mora biti znano
- originalni problemski prostor preslikamo v dvosmerni prostro stanj E1, E2 dosegljiv iz E in S1,S2,S3 dosegljiv iz S (S,E) -> (S1, E1), (S1,E2), (S2, E1), (S2, E2)... Vozlisce (Si, Ei) je v dvosmernem prostur ciljo vozlisce ce velja E=S (soda dolzina na isto mesto pridemo iz obeh strani) ali S->E (liha pot sosednja)



2.1.5 Primerjave algoritmov

Kriterij	Iskanje v širino	Iskanje v globino	Iskanje z omejitvijo globine	Iterativno poglabljanje	Dvosmerno iskanje
Popolnost	Da (če je b končen)	Ne	Ne	Da (če je b končen)	Da
Optimalnost	Da (če so cene enake)	Ne	Ne	Da (če so cene enake)	Da
Čas. zahtevnost	$O(b^d)$	$O(b^m)$	$O(b^l)$	$O(b^d)$	$O(b^{d/2})$
Prost. zahtevnost	$O(b^d)$	$O(bm)$	$O(bl)$	$O(bd)$	$O(b^{d/2})$

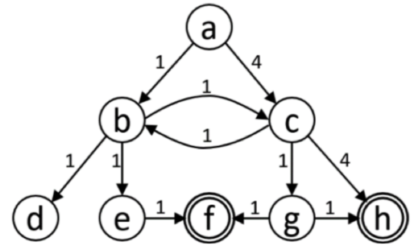
2.2 Informirani preiskovalni algoritmi

Ideja: preiskovanje usmerjamo z dodatnim znanjem **hevristiko** (ocenitvena funkcija za obetavnost vozlisca)
- optimisticna/dopustna: $\forall n: \boldsymbol{h(n) \leq h^*(n)}$ (h^* je optimalna ocena)
- optimalna: $h(n) = h^*(n)$
- pesimisticna: $h(n) \geq h^*(n)$

2.2.1 A*

A* is informed version of **dijkstra** (uses heuristics and pq), ce h(dopustna)=**popolna in optimalna**

Casovna zahtevnost odvisna od hevristike: $E = (h^* - h)/h^*$, $O(b^{E\cdot d})$, b-stopnja vejanja, d-globina optimalne resitve
Prostorska zahtevnost problem (hrani vsa vozlisca v spominu)



X	a	b	c	d	e	f	g	h
h(X)	4	6	2	6	3	0	7	0

$f(n) = g(n) + h(n)$, g(n) cena do vozlisca, h(n) hevristika
Razvijamo dokler ne pridemo do ciljnega vozlisca

Razvijano	Generirana	Priority Queue
/	a(4)	[]
a	b(7) c(6)	$[c(6), b(7)]$
c	b'(11) g(12) h(8)	[b(7),h(8),b'(11),g(12)]
b	c'(4) d(8) e(5)	[c'(4),e(5),h(8),d(8),b'(11),g(12)]
...	...	...
f		

2.2.2 IDA* (Iterative deepening A*)

f(n) = g(n) + h(n), g(n)=cena poti do n

Meja	Razvijano	Generirana	DFS (list)
0	/	s(7)	/
7	/	s(7)	s
	s	a(8) b(7) c(7)	b, c
	b	f(6) h(5)	f h c
	f	g(7) h(9) i(11)	g h c
	g		

2.2.3 Kakovost hevristicnih funkcij

Kakovost h ocenimo z **stevilom generiranih vozlisc** ter **efektivnim faktorjem vejanja** (N vozlisc je algoritem generiral da je na globini d nasel resitev)
Hocemo imeti dopustne hevristike s **cim visjimi vrednostmi** in **spre-jelmjivo ceno** (casom izracuna)
Ce $h_2(n) \geq h_1(n), \forall n$ potem h_2 **dominira** h_1

2.3 Nedeterministicno okolje

Pomagajo resevati probleme z **dekompozicijo na manjse probleme**
Uporabnost:

- princip deli in vladaj
- iskanje v nedeterministicnih okoljih
- igre med dvema nasprotnikoma s popolno informacijo (sah, dama)
- ekspertno resevanje problem

2.3.1 AO*

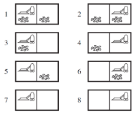
- posplositev A* na grafe AND/OR
- **popoln in optimalen** ⟺ h(n) ne precenjuje dejanske cene do cilja

$F(N)$ ocena za usmerjanje preiskovanja, $H(N)$ dinamicna hevristicna ocena
Postopek:

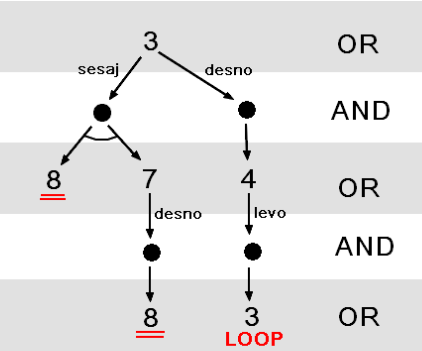
1. Razvij najcenejse vozlisce
   - ce list in koncno (oznaci), preveri 3. korak, nadaljuj v 1.
   - ce list in ni koncno (oznaci) vrednost vozlisca = ∞
2. Posodobi vse predhodnike
   - v AND starsih, cena starsa = $\sum$ sinov + povezava v
   - v OR starsih, cena starsa = min(sinovi) + povezava v
3. Koncaj ko obstaja pot od zacetnega vozlisca, po kateri v AND vozliscih po vseh sinovih prides do cilja, v OR vozliscih v vsaj enem

Podana je problem sesalca, ki smo ga obravnavali na predavanjih, pri čemer so stanja v problemskem prostoru zaradi lažjega sklicevanja oštevilčena, kot je prikazano na desni. Na začetku je sesalec v stanju 3 (nesnaga samo v prvi sobi, v isti sobi se nahaja sesalec). Njegove možne akcije so naslednje:

- *desno*: izvede premik v desno sobo (če se že nahaja v desni sobi, ta akcija ni možna), akcija ne vpliva na stanje smeti;
- *levo*: izvede premik v levo sobo (samo če se nahaja v desni sobi, sicer akcija ni možna), akcija ne vpliva na stanje smeti;
- *sesaj*: posesa smeti (akcija je možna samo, če so v sobi prisotne smeti); zaradi pokvarjenega vezja v sesalcu ta akcija včasih povzroči samo sesanje smeti, včasih pa sesanje smeti in hkraten premik v desno sobo.



Sesalec zaključi nalogo, ko pride v stanje 8 (obe sobi čisti, sesalec v desni sobi). Opomba: Nekatera od zgornjih stanj v tem problemu niso dosegljiva, s čimer si ne bomo belili glave.

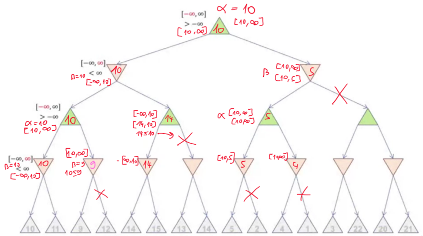


2.3.2 algoritem MINIMAX

$O(n^d)$

2.3.3 Rezanje alfa-beta

V najbolsem primeru zmansa iz $O(b^{m \cdot d})$ na $O(b^{m \cdot d/2})$
V max vozliscih popravljamo **alfa**
V min vozliscih popravljamo **beta**
Iz vozlisca na otroke prenesamo par [alfa, beta]

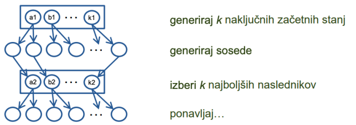


2.4 Lokalno preiskovalni algoritmi

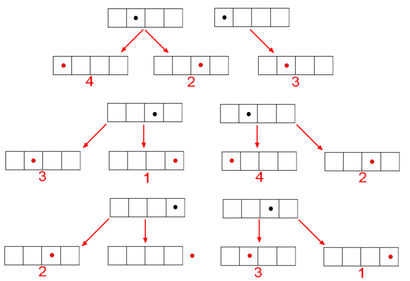
plezanje na hrib, simulirano ohlajanje, gen.algoritmi...

2.4.1 Lokalno iskanje v snopu

generiraj k nakljucnih zacetnih stanj
iz vsakegea generiraj sosede
izberi k najboljsih naslednikov
ponavljaj (iz maksimum stohasticno iskanje -> 1-verjetnost/sumvseh)



PRIMER: resujemo problem iskanja izhoda iz hodnika, ki ga sestavljajo 4 sobe. V eni izmed sob se nahaja robot, ki se lahko na vsakem koraku premakne za 1 sobo v (L) ali v (D). Problem resujemo s preiskovanjem v snopu, ki na vsakem koraku hrani 2 aktualni stanji.



2.4.2 Resevanje iz lokalnih maksimumov

Koraki vstran: ce ima naslednje stanje isto vrednost kriterijske funkcije, dovolimo premik v to stanje
Stohasticno plezanje na hrib; iz mnozice boljsih stanj, verjetnostno izberemo naslednje stanje (pri cemer upostevamo, da imajo boljsa stanja vecjo verjetnost izbora)
Nakljucni ponovni zagon: veckrat pozeni plezanje na hrib

3 Planiranje

plan zaporedje akcij, ki pripelje od zacetnega do koncnega stanja

3.1 Planiranje s sredstvi in cilji (STRIPS)

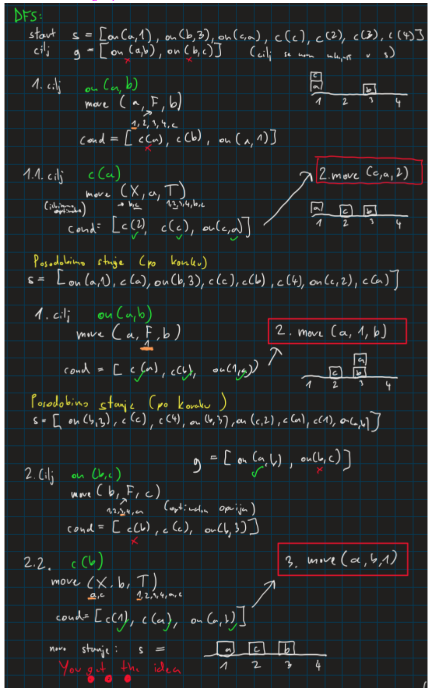
Agentu opisemo svet in postavimo fizikalne omejitve.
Ne zagotovalja optimalne resitve, obravnavamo le en cilj naenkrat (ko ga dosezemo, se lahko ostali izgubijo) = Sussmanova anomalija
Akcija move(X, From, To)

- pogoj: cond=[clr(X), on(X,F), clr(T)] ⟶ pogoji za izvajanje akcije,
- poz. ucinki: adds=[on(X, T), clr(F)] ⟶ nova stanja,
- neg. ucinki: dels=[on(X, F), clr(T)] ⟶ izbrisana stanja,
- omejitve: constr=[F ≠ T, X≠ F, X≠ T, block(X)] ⟶ omejitve akcij (fizikalne omejitve),

Algoritem:

1. Izberi se neresen cilj iz mnozice CILJEV
2. Izberi akcijo, ki izbrani cilj doda v stanje
3. Omogoci izbrano akcijo (izpolni pogoje)
4. Izvedi akcijo (ki izpolni najvec pogojev)
5. Ce obstajajo nereseni cilji ⟹ 1.

Primer dfs, zlaganje kock



3.2 Planiranje z regresiranjem ciljev (STRIPS)

Resitev za sussmanovo anomalijo
Zacnemo v ciljih, regresiramo do zacetka ($G_i \subset S_0$):

1. $G_{i+1} = G_i \cup \text{cond}(A) - \text{adds}(A)$
2. POGOJ: $G_i \cap \text{dels}(A) = \emptyset$
3. Preveri da ni protislovja (npr. $G_{i+1} = [on(b,c), \ldots, c(c) \ldots]$)

⟶ zactno_stanje = [on(a,1), on(b,a), c(b), on(c,3), c(c)]
⟶ hocemo da zacetno_stanje ⊂ G_i

1. G_0 = [on(a,b), on(b,c)]
   - on(a,b): $A_0 = move(a, From, b)$
   - From = 1
   - POGOJ: $G_0 \cap \text{dels}(A_0) = \emptyset$✓
   - G_1 = [on(a,b), on(b,c), c(a), c(b), on(a,1)]-[c(1), on(a,b)] ✓
2. G_1 =[on(b,c),c(a),c(b),on(a,1)]
   - c(a): $A_1 = move(X, a, To)$
   - X = c, To = 2
   - POGOJ: $G_1 \cap \text{dels}(A_1) = \emptyset$✓
   - G_2 = [on(b,c),c(a),c(b),on(a,1),c(c),c(2),on(c,a)] -[c(a), on(c,2)] ✗(protislovje)
   - on(b,c): $A_2 = move(b, From, c)$
   - From = 3
   - POGOJ: $G_2 \cap \text{dels}(A_2) = \emptyset$✓
   - G_2 = [on(b,c),c(a),c(b),on(a,1),c(c),c(b),on(b,3)] ✓
3. G_2 = ...

3.3 RAZPOREJANJE OPRAVIL PDDL

Razsirimo lahko notacijo (PDDL):
Akcija1 ≺ Akcija2: Akcija1 se mora zgoditi pred Akcijo2
Resources podajo stevila razpolozljivih resursov
DURATION opredejljuje trajanje posamezne akcije
CONSUME opredeljuje (trajno) porabo dolocene kolicine resursov
USE opredeljuje (zacasno) zasedenost kolicine resursov med izvajanjem akcije

```
Jobs (AddEngine1 ≺ AddWheels1 ≺ Inspect1,
      AddEngine2 ≺ AddWheels2 ≺ Inspect2 )
Resources (EngineHoists(1), WheelStations(1), Inspectors(2), LugNuts(500))

Action (AddEngine1 , DURATION:30,
        USE:EngineHoists(1))
Action (AddEngine2 , DURATION:60,
        USE:EngineHoists(1))
Action (AddWheels1 , DURATION:30,
        CONSUME:LugNuts(20), USE:WheelStations(1))
Action (AddWheels2 , DURATION:15,
        CONSUME:LugNuts(20), USE:WheelStations(1))
Action (Inspect i, DURATION:10,
        USE:Inspectors (1))
```

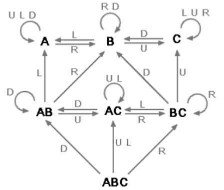
Metoda kriticne poti
kriticna pot: pot, ki je **najdaljsa** in doloca dolzino trajanja celotnega plana vsaki akciji priredimo par **[ES, LS]**

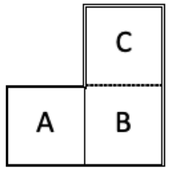
- **ES**: najbolj zgodnji mozen zacetek (Earliest Start)
  - $ES(start) = 0, \quad ES(B) = \max_{A \prec B} [ES(A) + Duration(A)]$
- **LS**: najbolj pozen mozen zacetek (Latest Start)
  - LS(Finish) = ES(Finish), $LS(A) = \min_{A \prec B} [LS(B) - Duration(A)]$

rezerva(slack)=LS-ES (**casovna rezerva**) Algoritem po **hevristiki minimum slack** ⟶ na vsaki iteraciji ima prednost akcija ki ima izpolnjene predhodnike in najnizji slack, nato posodobi [ES in LS] za celotni graf in ponovi.

3.4 Preiskovanje brez informacije o stanju

belief states - graf, zacetna stanja = potencna mnozica vseh stanj, konec samo koncno stanje
Podan je majhen labirin iz treh sob (A,B,C). V labirintu se nahaja robot, ki mora priti v sobo **C**. Vendar pa robot nima senzorja za zaznavo v kateri sobi se nahaja zato mora brez informacije o stanju poiskati akcije ki ga bodo zagotovo pripeljale na cilj. Izvede lahko akcije U, D, L in R



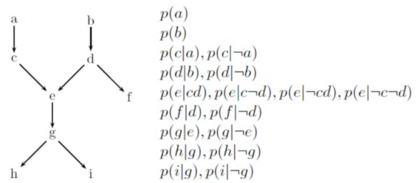


4 Sklepanje

4.1 Bayesovske mreze

Baye. mreza = Usmerjen graf, kjer so podane zahtevane verjetnosti:

- Za vozlisca **brez starsev** verjetnosti $P(v_i)$
- Za vozlisca z **starsi** pogojne verjetnosti vseh kombinacij starsev



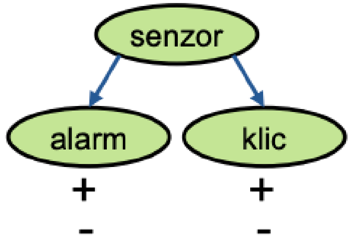
Pravila verjetnostnega sklepanja:

1. **Konjunkcija**: $P(X_1 X_2 \mid C) = P(X_1 \mid C) P(X_2 \mid X_1 C)$
   - $P(A_1 \cap \cdots \cap A_n) = \prod_{k=1}^{n} P\left(A_k \mid \bigcap_{j=1}^{k-1} A_j\right)$
2. **Gotov dogodek**: $P(X \mid \ldots X \ldots) = 1$
3. **Nemogoc dogodek**: $P(X \mid \ldots \overline{X} \ldots) = 0$
4. **Negacija**: $P(\overline{X} \mid C) = 1 - P(X \mid C)$
5. Ce je Y naslednik od X in je Y vsebovan v pogojnem delu: $P(X \mid YC) = P(X \mid C) \cdot \frac{P(Y|XC)}{P(Y|C)}$
6. Ce pogojni del ne vsebuje naslednika od X:
   (a) ce X **nima** starsev: $P(X \mid C) = P(X)$, P(X) je podan
   (b) ce **ima** X starse S: $P(X \mid C) = \sum_{S \in P_X} P(X \mid S) P(S \mid C)$

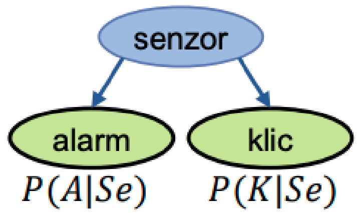
4.2 Pravila sklepanja

4.2.1 Divergentno vozlisce

Alarm in klic sta **odvisna** P(A|K) ≠ P(A), P(K|A) ≠ P(K)

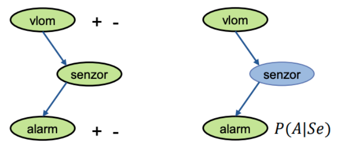


Vendar poznavanje resnicnosti prednika **neodvisna** P(A|Se K) = P(A|Se), P(K|Se A) = P(K|Se)

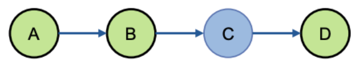


4.2.2 Veriga

Ce vemo da je resnicen tudi **senzor** postaneta vlom in alarm **neodvisna**



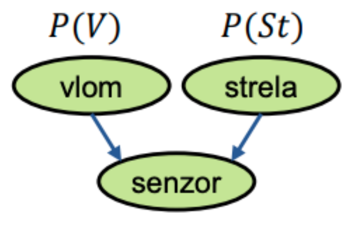
Pravilo posploseno na daljse verige:



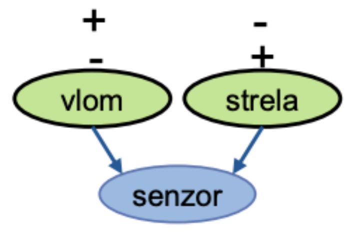
$P(D|ABC) = P(D|C)$

4.2.3 Konvergentno vozlisce

Vlom in strela sta medseboj **neodvisna** P(V|St)=P(V), P(St|V)=P(St)

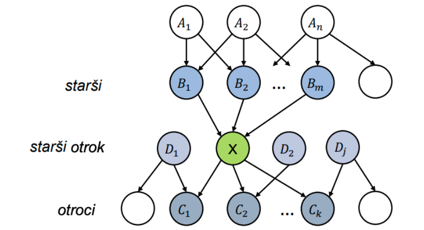


Vendar poznavanje da se je sprozil senzor, dogodka postaneta **odvisna** P(V|St Se) ≠ P(V|Se), P(St|V Se) ≠ P(St|Se)



4.2.4 Ovojnica Markova

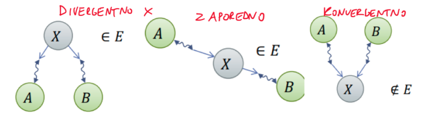
X je **neodvisno** od vseh ostalih ⟺ podani **starsi**, **otroci** in **starsi otrok**



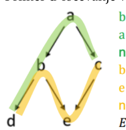
4.3 D-locevanje

A in B v mrezi sta **neodvisni** ⟺ obstaja mnozica vozlisc E, ki d-locuje A in B, potem sledi: ($P(A|EB) = P(A|E)$)

```
za vsako neusmerjeno pot P med A in B v bayesovski mreži:
  za vsako vozlišče X na poti P:
    analiziraj pogoj za pripadnost X množici E glede na tip:
      divergentno ali zaporedno vozlišče: X ∈ E
      konvergentno vozlišče: X in nasledniki ∉ E
    Sx = množice vozlišč, ki ustrezajo pogoju za X
  Sp = ∪X(SX)     // množice, ki d-ločujejo samo na poti P
                     (unija množic za vozlišča na poti)
E = ∩P SP       // množice, ki d-ločujejo v celi mreži
                     (presek množic za vse možne poti)
```



! pri konvergentnem izlocimo tudi vse naslednike X
Primer d-locevanje vozlisc c in d



b ∈ E: $\{\{b\},\{a,b\},\{b,e\},\{a,b,e\}\}$
a ∈ E: $\{\{a\},\{a,b\},\{a,e\},\{a,b,e\}\}$
na celi zeleni poti: $\{\{a\},\{b\},\{a,b\},\{a,e\},\{b,e\},\{a,b,e\}\}$
b ∈ E: $\{\{b\},\{a,b\},\{b,e\},\{a,b,e\}\}$
e ∉ E: $\{\{\ \},\{a\},\{b\},\{a,b\}\}$
na celi oranžni poti: $\{\{\ \},\{a\},\{b\},\{a,b\},\{b,e\},\{a,b,e\}\}$
$E = \{\{a\},\{b\},\{a,b\},\{b,e\},\{a,b,e\}\}$

⟶ P(d|ca)=P(d|a), P(d|cb)=P(d|b), P(d|cab)=P(d|ab), P(d|cbe)=P(d|be),P(d|cabe)=P(d|abe)

4.4 I-ekvivalentnost

Mrezi sta I-ekvivalentni ce imate **enako strukturo** (ob ignoriranju usmerjenosti povezav) in **ista konvergentna vozlisca**: