1 Uvod v Umetno inteligenco

1.1 Turingov test

Opazovalec po pogovru ne more lociti racunalnika od cloveka.

2 Strojno ucenje

2.1 Problemski prostor, ocenjevanje znanja

2.2 Evalviranje hipotez

Pomembni kriteriji:

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

Ocenjevanje uspesnosti pri klasifikaciji na podlagi njihove **tocnosti**:

TP - true positive, FP - false positive, FN - false negative, TN - true negative

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

Napaka 1. tipa = FP, napaka 2. tipa = FN

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

2.3 Gradnja odlocitvenih dreves

Informacijski prispevek Gain(A) = $I - I_{res}(A)$, I=H(C)

$$I_{\text{res}} = -\sum_{v_i \in A} p_{v_i} \sum_{c} p(c|v_i)\log_2 p(c|v_i)$$

Za koliko se entropija zmanjsa po delitvi z Atributom A.

Razmerje inofrmacijskega prispevka atributa A:

IGR(A) = $\frac{\text{Gain(A)}}{\text{H(A)}}$

2.3.1 TDIDT (Top down induction decision tree) algoritem

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

- kratkoviden algoritem

2.3.2 Binarizacija atributov

Aleternativa za resevanje problematike z vecvrednostnimi atributi:

Primer: barve ∈ rdeca, rumena, zelena, modra

Strategije, razbijemo v dve mnozici:

- rdeca, rumena, zelena, modra
- rdeca, rumena, zelena, modra

Prednost: manjse vejanje drevesa.

2.4 Ucenje iz sumnih podatkov (rezanje)

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 (domenski ekspert lahko pove)

Laplacova ocena verjetnosti $p = \frac{n+1}{N+k}$

k...stevilo vseh moznih razredov

2.4.1 REP (reduced error prunning)

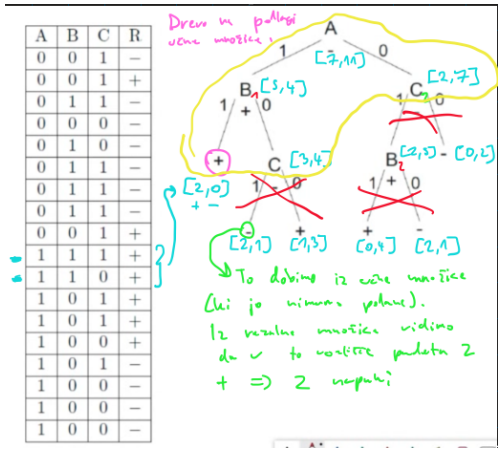
Dela dobro ce imamo veliko rezalno mnozico.

Obicajno uporabljamo relativno frekvenco za ocenjevanje verjetnosti.

$G(v) = \#\text{napak}_T - \#\text{napak}_v$

$G(v) \geq 0 \Rightarrow$ 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	−



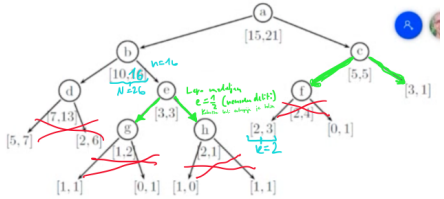
$e(C) = 3$

$e_T = 2+3 = 5$

$G(C) = 5-3 = 2 \geq 0 \rightarrow$ rezemo

2.4.2 MEP (Minimal Error Prunning)

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



(Laplace)

$e_L(d) = 1 - t = 1 - \frac{13+1}{20+2} = 0.363$

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

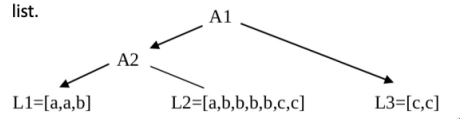
2.5 Ocenjevanje uspesnosti modelov

tocnost t … verjetnost pravilnosti klasifikacije

Laplacova ocena verjetnosti $p = \frac{n+1}{N+k}$

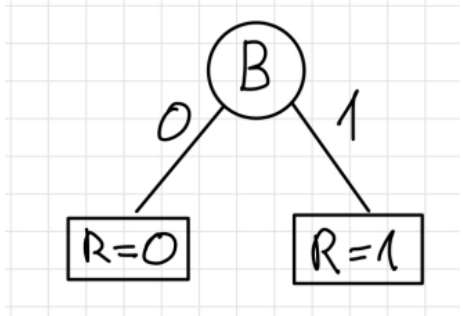
k...stevilo vseh moznih razredov

list.



$t_{L1} = \frac{2+1}{3+3} = 0.5$, $t_{L2} = \frac{4+1}{7+3} = 0.5$, $t_{L3} = \frac{2+1}{2+3} = 0.6$

tocnost drevesa: $t_D = 3/12 \cdot 0.5 + 7/12 \cdot 0.5 + 2/12 \cdot 0.6 = 0.5167$



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

2.6 Obravnanva mankajocih atributov, navini Bayesov klasifikator

2.6.1 Naivni bayes

Ce poznamo razred, kam klasificiramo ce nepoznamo atributov:

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

c…razred, x_i…atributi

Verjetnost::

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

Primer moski: visina ≥ 175, teza ≥ 65, spol = M

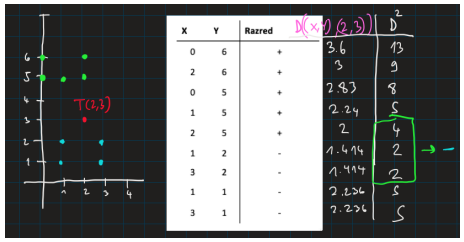
$X \backslash Y$	Razred A	Razred B
p_a	$P(A) = \frac{2}{3}$	$P(B) = \frac{1}{3}$
spol	$P(M\|A)$	$P(M\|B)$
visina	$P(V \geq 175\|A)$	$P(V \geq 175\|B)$
teza	$P(T \geq 65\|A)$	$P(T \geq 65\|B)$
$P(y)\prod_{i=1}^{n} P(x_i\|y)$	…	…

2.6.2 Nomogragmi

Ciljni razred $C = c_T$

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

2.7 K-najblizjih sosedov



3 Vrste ucenja

3.1 Nadzorovano ucenje (supervised learning)

Ucni primeri so podani/oznaceni kot **vrednosti vhodov in izhodov**.

$(\vec{x}_1,\vec{y}_1),(\vec{x}_2,\vec{y}_2),\ldots,(\vec{x}_N,\vec{y}_N)$

$\vec{x}_i$… atributi, $\vec{y}_i$… ciljna spremenljivka

Locimo dve vrsti problemov:

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

3.1.1 Lokalno utezena regresija

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

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	11/4
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	3/2
1	2	1	12	0	1	12
					26/5	144/20

$h(x_q) = \frac{\sum_{i=1}^{k} w_i \cdot f(x_i)}{\sum_{i=1}^{k} w_i} = \frac{144/20}{26/5} = \frac{72}{65} = 11.359$

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 $w_i = \frac{1}{1+d_{ij}}$.

3.1.2 Regresijska drevesa

Linearna regresija je poseben primer regresijskega drevesa.

V listih regresijskega drevesa vcasih napovemo kar povprecno vrednost.

3.2 Nenadzorovano ucenje (unsupervised learning)

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

3.2.1 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 najkrajse razdalje med primeroma iz razlicnih gruc.

Complete-linkage: povezava med grucami je najdaljsa razdalja med primeroma iz razlicnih gruc.

Average-linkage: povezava med grucami je povprecna razdalja med primeroma iz razlicnih gruc.

3.2.2 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.

3.3 Spodbujevalno ucenje - reinforcement learning

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

3.4 Ocenjevanje ucenja

3.4.1 Precno preverjanje

Poseben primer **veckratnega ucenja** in testiranja

k-kratno precno preverjanje

- celo ucno mnozico razbij na k disjunktnih podmnozic
- za vsako od k podmnozic:
  - uporabi **mnozico** kot **testno mnozico**
  - uporabi **preostalih k-1** mnozic kot **ucno mnozico**
- povpreci dobljenih k ocen tocnosti v koncno oceno

Pri precnem preverjanju uporabimo vse podatke za testiranje in vse za ucenje

Metoda **leave one out** je poseben primer precnega preverjanja

Imamo dve hipotezi A in B. Izkase se, da A bolje napoveduje na ucnih podatkih B pa na testnih. Potem je B verjetno boljsa hipoteza.

4 Preiskovanje

4.1 Neinformirani preiskovalni algoritmi

4.1.1 Iskanje v sirino

4.1.2 iskanje v globino

Izboljsave:

- **Iskanje s sestopanjem**

- **depth-limited-search** (vnapej definiramo globino l (dolocimo preko domenskega znanja))

4.1.3 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

4.1.4 dvosmerno iskanje

Ideja: pognati vzporedni iskanji od zacetka do cilja in od cilja do zacetka.
Motivacija:

Implemenatcija dvosmernega iskanja

- ciljno vozlisce mora biti znano
- originalni problemski prostor preslikamo v dvosmerni prosto 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)

4.1.5 cenovno - optimalno iskanje

- posplositev iskanja v sirino (iskanje v sirino je optimalno, ce so cene vseh povezav enake 1)

- dijkstra basically (sam do zadnga noda)
- https://stackoverflow.com/a/14587449

4.1.6 Primerjava algoritmov

Kriterij	sirino	globino	omejitvijo globine	iterativno poglabljanje	dvosmerno iskanje

4.2 Informirani preiskovalni algoritmi

4.2.1 hevristicno preiskovanje

ideja: preiskovanje usmerjamo z dodatnim znanjem (ocenitven funcija za obetavnost vozlisca)
hevristika je ocenitvena funkcija za obetavnost vozlisca
Niso optimalni algoritmi (ne zagotavljajo optimalnih resitev)

Primeri hev. algoritmov:

- A*
- IDA*
- RBFS (recursive best-first search)
- iskanje v snopu (beam search)
- plezanje na hrib (hill climbing)

4.2.2 pozresno preiskovanje/ greedy best-first search

$h(n)$ hevristicna ocena
vrednotenje vozlisca $f(n) = h(n)$ hevristicna ocena ... npr manhattan distance (zracna razdalja)

- **popolnost** (ali najde vedno resitev): Ne
- **optimalnost**: Ne
- **casovna zahtevnost** $O(b^m)$, kjer je m najvecja globina drevesa

4.2.3 A*

Ideja: boljsa hevrisiticna fun
Vozlisca vrednotimo glede na ceno najboljse poti skozi vozlisce n $f(n) = g(n) + h(n)$, $g(n)$ cena poti do n (znano), $h(n)$ cena od n do najblizjega cilja (ocena)
prioritetna vrsta (max glede na f(n)) Basically dijkstra + h(n) (A* is basically an informed variation of Dijkstra.)

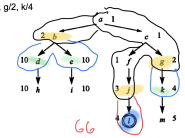
- **popolnost**: Da (ce ustreza pogoju dopustnosti)
- **optimalnost**: Da (ce ustreza pogoju dopustnosti)
- **casovna zahtevnost** $O(b^m)$, kjer je m najvecja globina drevesa

h(n) je **dopustna** ce nikoli **ne precenjuje cene do cilja**
- hevristika optimisticna predvideva da je do cilja manj kot dejansko je (zracna razdalja, igra 8 ploscic)
- $\forall h(n) \leq h*(n)$, kjer je $h*(n)$ dejanska cena optimalne poti do cilja za vozlisce n - h(n) = 0 (A* = dijkstra), preiskovanje s cenami, velja da je sevedno optimalen

4.2.4 IDA* (Iterative deepening A*)

Deluje kot **dfs**, vendar za mejo uporabljamo **f(n)** namest globine - za mejo na zacetku izberemo vrednost f(n) zacetnega v - na vsaki iteraciji razvijemo vsa vozlisca z f(n) <= mejni vrednosti - za naslednjo iteracijo izberemo mejo, ki je najmanjsi f(n) od se nerazvitih vozlisc

- primer:
  - podane so vrednost **f(n)** ($= g(n) + h(n)$) vozlišč
  - simuliraj preiskovanje z IDA*
- generirana vozlišča
  - 1. iteracija, meja=1: a/1, b/2, c/1, f/1, j/3, g/2
  - 2. iteracija, meja=2: a/1, b/2, d/10, e/10, c/1, f/1, j/3, g/2, k/4
  - 3. iteracija, meja=3: a/1, b/2, d/10, e/10, c/1, f/1, j/3, l/4, g/2, k/4
  - 4. iteracija, meja=4: a/1, b/2, d/10, e/10, c/1, f/1, j/3, **l/4**



Ucinkovitost

- neucinkovit ce vozlisca raznolika f(n)
- prednost: ne hrani vec vseh vozlisc kot A*
- optimalen: ce razvija v prioritetnem vrsntem redu, h(n) mora biti **monotona|konsistentna** (h(n) skos pada) (posledicno tudi dopustna)

$$h(n) \leq c(n, n') + h(n')$$

(h naslednjega vozlisca manjsi ker je blizji cilja)
- monotona → dopustna (proti primer h(n) = 0)

4.2.5 Kakovost hevristicnih funkcij

7	2	4
5		6
8	3	1

Primer igra 8 ploscic
-h_1: stevilo ploscic ki niso na pravem mestu (8)
-h_2: vsota manhattanskih razdalj ploscic do pravega mesta(3+1+2+2+2+3+3+2=18)
Kakovost h ocenimo z:
- stevilom generirarnih vozlisc
- z efektivnim faktorjem vejanja (koliko vozlisc N je algoritem generiral da je na globini d nasel resitev)

	število generiranih vozlišč			*efektivni faktor vejanja*		
Globina	**IDS**	**A*(h_1)**	**A*(h_2)**	**IDS**	**A*(h_1)**	**A*(h_2)**
2	10	6	3	2,45	1,79	1,79
4	112	13	12	2,87	1,48	1,45
6	680	20	18	2,73	1,34	1,30
8	6384	39	25	2,80	1,33	1,24
10	47127	93	39	2,79	1,38	1,22
12	3644035	227	73	2,78	1,42	1,24
14	?	539	113	?	1,44	1,23
16	?	1301	211	?	1,45	1,25
18	?	3056	363	?	1,46	1,26
20	?	7276	676	?	1,47	1,27
22	?	18094	1219	?	1,48	1,28
24	?	39135	1641	?	1,48	1,26

Vidimo $h_2(n) \geq h_1(n) \forall n$ pravimo h_2 **dominira** h_1

4.3 Lokalno preiskovalni algoritmi

4.3.1 Plezanje na hrib

Ne pomnemo poti do cilja, ampak samo trenutno stanje
Koristni v primerih:
- ce nas zanima samo kakovost resitve (in ne pot do cilja)
- resevanje optimizacijskih problemov (kjer je podana **kriterijska funkcija** za oceno kakovosti resitve)

Prednosti:
- majhna poraba prostora

Primer 4 kraljice na sahovnici - kriterijska funkcija: maksimiziramo - (minus) stevilo kraljic, ki se medsebojno napadajo

Tezave:
- lokalni maksimumi
- "rame, plaote" (kriterijska funkcija konstantna vrednost)
- grebeni (za plezanje navzgor je potreben sestop po pobocju grebena)

Resevanje iz lokalnih maksimumov:
- **koraki vstran**: ce ima naslednje stanje isto vrednost kriterijske funkcie, 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 iz nakljucnih stanj dokler ne najdes resitve

4.3.2 Simulirano ohlajanje

algoritem ki izvira iz metalurgije (ko je jeklo tekoce, so molekule v njem bolj gibljive; ko se ohlaja se strjuje in molekuele se umirjajo)
Analogija:
- generiramo nakljucne sosede trenutnega stanja
- ce najdemo **boljse stanje ga izberemo**
- ce najdemo **slabse stanje, ga izberemo z doloceno verjetnostjo**
- verjetnost izbire neoptimalnega stanja s casom pada (nizanje temperature)

4.3.3 Lokalno iskanje v snopu

Algoritem:
- v spominu hrani k aktualnih stanj namesto enega
- izberi k optimalnih stanj od sosedov aktualnih stanj
- ponavaljaj do ustavitnega pogoja

4.4 Grafi AND/OR, nedeterministicno okolje