Racket: Construcția arborilor de decizie

Descriere

Scopul temei este implementarea funcționalității algoritmului **ID3** de creare a unui arbore de decizie.

Reprezentarea internă a arborelui de decizie va fi la alegere (nu este impusă) și arborele va fi 'citit' prin intermediul unui set de funcții de acces.

Astfel, fazele de realizare a temei vor fi
- decizia asupra reprezentării interne a arborelui de decizie;
- implementarea setului de funcții de acces pentru reprezentarea aleasă. De ajutor în alegerea reprezentării pot fi testele acestor funcții;
- implementarea funcțiilor de calcul pentru entropia informațională și pentru câștigul informațional (în raport cu un atribut) a/al unei mulțimi de exemple;
- construcția unui arbore de decizie;
- BONUS: gestionarea unor cazuri speciale – atribute insuficiente pentru a separa exemplele, exemple listă pentru anumite combinații de valori ale atributelor.

Arbori de decizie

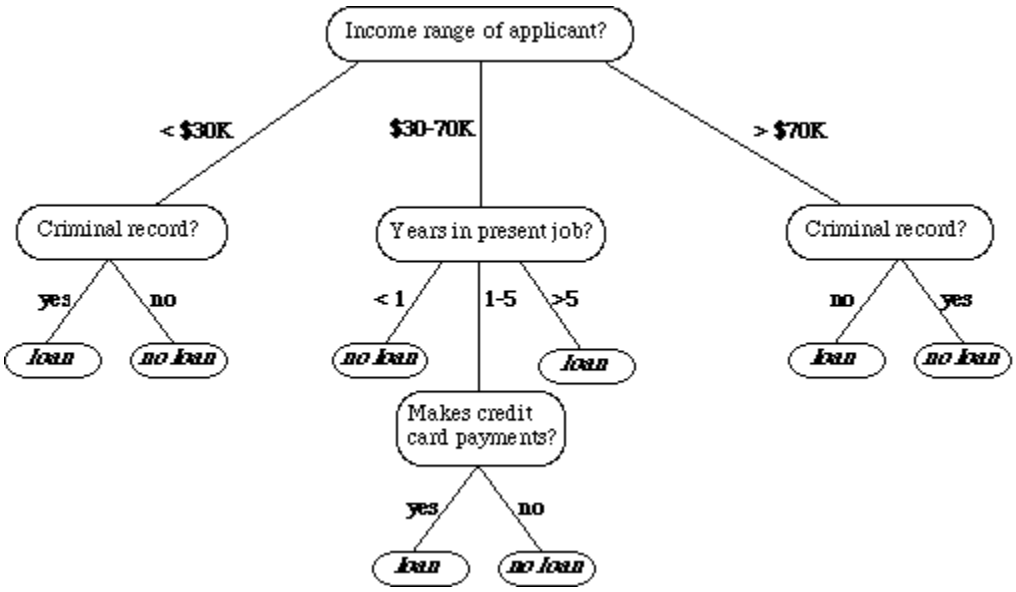
Arborii de decizie sunt o metodă de clasificare care aplică învățarea supervizată.

Învățarea și testarea se fac prin **exemple**, fiecare dintre exemple fiind caracterizat prin **valorile** pentru diverse **atribute**. De exemplu, fiecare dintre obiectele dintr-o mulțime pot fi caracterizate prin dimensiunea și forma lor, e.g. forma unui obiect poate fi 'rotund', 'pătrat' sau 'neregulat'. Fiecare exemplu are de asemenea o **clasă**.

Inițial, se dau un număr de exemple *de învățare*, pentru care clasa de care aparțin este dată. Pe baza acestora se construiește un arbore de decizie.

Un arbore de decizie este un arbore în care nodurile reprezintă atribute care separă exemplele (nodurile reprezintă *decizii*), ramurile care pleacă dintr-un nod corespund valorilor atributului din nod, iar frunzele reprezintă clase. Atunci când se dă un exemplu nou (*de test*), pentru care trebuie să determinăm clasa, este suficient să se parcurgă arborele, pornind de la rădăcină, și la fiecare nod întâlnit să se meargă pe ramura corespunzătoare valorii care caracterizează exemplul. Când se ajunge la o frunză, clasa din frunză este clasa în care se încadrează exemplul.

De exemplu, o aplicație simplă pentru decizia acordării unui credit poate folosi următorul arbore de decizie:



Astfel, un nou aplicant, cu un venit de $50K anual, angajat de 6 ani pe postul curent, va corespunde cu ramura din mijloc a rădăcinii, și ramura din stânga a nodului „Years in present job“. Această cale ajunge la o frunză care clasifică aplicantul ca un bun candidat pentru primirea unui credit.

Construcție

Construcția unui arbore de decizie se face în etape, la fiecare etapă alegând un atribut care separă exemplele și reiterând pentru fiecare din sub-mulțimile de exemple care rezultă din separare.

Să presupunem că avem mulțimea de exemple formată din 3 obiecte: o sferă roșie, un cub albastru, și un cub roșu. Facem teste pentru a vedea care obiecte atrag atenția mai ușor, pentru ca apoi să putem prezice această capabilitate pentru alte obiecte (spoiler: cele roșii). Vom avea astfel două atribute: „formă“ și „culoare“, și clasa „atractiv“. Exemplele de învățare vor fi:

```
sferă roșu da
cub albastru nu
cub roșu da
```

Ignoranți în fața evidenței, vom porni arborele de decizie cu atributul formă, rezultând o partiționare în {sferă roșie da} și {cub albastru nu, cub roșu da}. Pentru sfere avem un singur exemplu, care are clasa „da“, deci vom avea o frunză corespunzătoare. Pentru cuburi, vom separa și după culoare pentru a putea separa clasa. Avem un arbore de înălțime 3 (în număr de niveluri).

Dar am fi putut avea un arbore de înălțime 2 dacă am fi ales culoarea ca prim atribut.

Pentru a alege optim atributele putem folosi câștigul informațional, care se calculează folosind entropia informațională. În cazul nostru:

Entropia setului inițial de exemple S este:

```
H(S) = - (p("da")*log2(p("da")) + p("nu")*log2(p("nu"))) =
     = -(-0.38 + -0.52) = 0.91
```

Câștigul pentru „formă“ este calculat în funcție de dimensiunea și entropia sub-mulțimilor de exemple corespunzătoare valorilor atributului S-sferă (1 element) și S-cub (2 elemente, clase diferite):

```
IG(S, "formă") = H(S)- p("sferă")*H(S-sferă) - p("cub")*H(S-cub)
               = 0.91 - 1/3*0 - 2/3*1 = 0.25 (aprox)
```

`H(S-sferă)` este 0 pentru că toate exemplele au aceeași clasă → `p(„sferă“/„da“)` este 1 (deci logaritmul său este 0) și `p(„sferă“)/„nu“` este 0 (deci nu mai calculăm logaritmul și produsul este 0).

`H(S-cub)` este 1 din `-(1/2*log2(1/2) + 1/2*log2(1/2)) = -log2(1/2) = 1`.

Câștigul pentru „culoare“ este calculat în funcție de dimensiunea și entropia mulțimilor S-roșu (2 elemente, aceeași clasă) și S-albastru (1 element):

```
IG(S, "culoare") = 0.91 - 2/3*0 - 1/3*0 = 0.91.
```

Deci atributul culoare are un câștig mai mare și va fi ales pentru prima separare.

Bonus

Există situații în care la separarea setului de exemple după un atribut, pentru anumite valori nu există exemple (mulțimea din nodul corespunzător valorii este vidă). În acest caz, nodul va deveni o frunză „specială“ cu tipul 'default' și având drept clasă clasa majoritară a exemplelor din nodul părinte.

Există de asemenea situația în care într-un nod avem o mulțime de exemple care nu au toate aceeași clasă, dar nu mai există atribute după care să împărțim exemplele. În acest caz, nodul devine frunză specială, cu tipul 'majority', și clasa majoritară a exemplelor din nodul curent.

Bonusul constă din implementarea funcționalității corespunzătoare acestor cazuri speciale.

Cerințe

Nu uitați să inspectați fișierul `decision-tree-test.rkt` pentru informații despre formatelor datelor primite și pentru funcții potențial utile. Testele folosite pot fi inspectate apelând funcția `get-test` având ca argument una dintre valorile `'food-small`, `'food-big`, `'objects`, `'weather` sau `'bonus` (pentru bonus). Valoarea întoarsă de `get-test` conține informațiile despre exemple de învățare, atribut clasă, set de atribute și exemple de test (fiecare este o listă din care primul element – numele listei – va fi eliminat).

Reprezentarea arborelui

Reprezentarea internă a structurii arborelui de decizie este la alegere. Trebuie totuși să fie posibilă recunoașterea câtorva structuri (folosind funcțiile de mai jos):

- un nod este un nod intern al arborelui, în care se face separația după un anumit atribut și care are un număr de ramuri descendente egal cu numărul de valori al atributului.
- o frunză este un nod care nu are copii, și este caracterizată de o clasă.
- o frunză „specială“ este o frunză care, pe lângă clasă, are și un tip (majority sau default).
- o ramură este caracterizată de o valoare (una dintre valorile atributului verificat în nodul părinte) și un nod copil (care poate fi nod intern sau frunză).

Testarea se va face folosind o listă de 6 (opțional 7) funcții de acces, care întorc:

- dacă un nod este frunză.
- care este clasa unei frunze (va fi apelată numai după verificare folosind funcția de mai sus).
- (pentru bonus) dacă un nod este frunză „specială“ (cu tip); o frunză specială este și frunză.
- (pentru bonus) care este tipul unei frunze speciale (va fi apelată numai după verificare folosind funcția de mai sus).
- care este atributul verificat într-un nod (va fi apelată numai după ce s-a verificat că nodul nu este frunză).
- care este nodul copil corespunzător unei valori a atributului verificat în nod (la fel).
- opțional, se poate implementa o funcție care confirmă dacă un argument oarecare este un nod valid în arbore (intern sau frunză).

Calculul entropiei și al câștigului informațional

Se cere implementarea celor două funcții, conform cu formulele prezentate (disponibile atât în sursă cât și pe wikipedia.

Construcția arborelui

Se cere construcția arborelui de decizie 9folosind algoritmul ID3), pe baza unei mulțimi de exemple (adnotate cu clasa lor), a unei liste de atribute (cu valori) și a mulțimii de valori pentru atributul clasă.

Algoritmul de construcție este unul recursiv:

- dacă exemplele sunt toate în aceeași clasă, se întoarce o frunză cu clasa respectivă
- (pentru bonus) dacă mulțimea de exemple este vidă, se întoarce o frunză de tip 'default'
- (pentru bonus) dacă nu mai sunt atribute de verificat, se întoarce o frunză de tip 'majority'
- altfel
  - se alege din lista de atribute atributul A cu câștig informațional maxim pe mulțimea de exemple dată
  - se scoate atributul A din lista de atribute
  - exemplele se împart în funcție de valorile lor pentru atributul A
  - pentru fiecare valoare v a atributului A, se apelează funcția recursiv, transmițând ca argumente mulțimea exemplelor care au valoare v pentru atributul A, lista de atribute din care atributul A a fost eliminat, și informațiile despre valorile clasei
  - se creează un nou nod, având drept copii nodurile corespunzătoare diverselor valori pentru atributul A

Primele teste nu verifică alegerea atributelor în funcție de câștigul lor informațional, deci în primă fază arborele poate fi construit alegând atributele în orice ordine. Este doar necesar ca decizia să se facă în mod corect.