Pràctica 2:

Arbres de decisió

Coneixement, raonament i incertesa

Curs 2020-2021

Biel Castaño Segade, 1419234
Sergi Masip Cabeza, 1533031
Jordi Xhafa Daci, 1419987

16 de novembre de 2020

Universitat Autònoma
de Barcelona

Resum

En aquest informe es presenta el plantejament i la solució implementada per un problema d'arbres de decisió. A partir d'un conjunt de dades, s'han aplicat diferents arbres de decisió per generar prediccions fiables. S'han utilitzat els algorismes ID3 i C4.5, s'han tractat els missing values del conjunt de dades

Abstract

In this essay report we will present the design and implementation for the resolution of a problem – the crossword puzzle – using the backtracking algorithm, the forward checking technique and the minimum remaining values heuristic. The program has been implemented using python. We have successfully completed the proposed problems, including the execution of the program for a large crossword puzzle with a dictionary of 580.000 words.

Índex

1 Introducció **1**

2 Base de dades **2**
2.1 Preprocessing 2

3 Solució proposada **3**
3.1 Arbres de decisió bàsics 3
3.2 Tractament de missing values 4
3.3 Tractament d'atributs continus 4
3.4 Random forests 4
3.5 Validació 4

4 Problemes trobats i solucions proposades **5**

5 Resultats **6**

6 Conclusió i possibles ampliacions **7**

1 Introducció

El problema presentat consisteix en la implementació d'arbres de decisió sobre un conjunt de dades. En particular, es tracta de l'algorisme ID3 i el criteri de separació C4.5 per a atributs categòrics, que ens permeti determinar el tipus d'entitat d'una nova mostra. També es demana representar visualment l'arbre de decisió resultant.

L'algorisme ID3 és un algorisme inventat per Ross Quinlan l'any 1975 emprat per a generar un arbre de decisió a partir d'un conjunt de dades. El criteri de divisió utilitzat és el guany d'informació basat en la entropia. És una estratègia greedy de selecció, i no garanteix una solució òptima. A més, pot produir overfitting. Això s'intenta arreglar a la última part de la pràctica.

L'algorisme C4.5 és una extensió de l'algorisme ID3 que presenta vàries millores respecte aquest. Principalment, l'avantatge més gran és que pot gestionar atributs tant discrets com continus. Pels atributs continus, ho fa creant un threshold i dividint les dades en dos sub-conjunts - els que estan per sobre del threshold i els que estan per sota. Un altre avantatge en comparació amb l'ID3 és que pot gestionar missing data sense haver de manipular el conjunt de dades.

El conjunt de dades a tractar és l'Adult Data Set, que conté vàries categories d'informació obtingudes a partir d'un conjunt d'adults. Els atributs són tant de tipus discrets (nominals) com continus. Per poder analitzar les dades amb l'algorisme ID3 s'ha d'aplicar un tractament de dades per tal d'eliminar qualsevol irregularitat, com poden ser valors buits o atributs que no ens aporten informació rellevant. A més a més, els atributs continus s'han de discretitzar. Això es pot fer amb llibreries python.

S'ha validat el conjunt de dades amb el mètode de cross-validation i les mètriques de *accuracy*, *precision*, *recall*, *specificity*, i *F1-score*.

Implementació de missing values avançat

Tractament d'arguments continus

Reducció de overfitting

2 Base de dades

El conjunt de dades a tractar és l'*Adult Data Set*. Aquest conjunt conté un seguit de categories d'informació sobre adults que es van recol·lectar a l'any 1994. A partir d'aquestes dades es vol predir si un adult guanya més de $50.000 anuals.

Els atributs del conjunt són:

- age: continu
- workclass: Private, Self-emp-not-inc, Federal-gov, etc.
- fnlwgt: continu
- education: Bachelors, Some-college, 11th, etc.
- education-num: continu
- marital-status: Divorced, Never-married, Separated, etc.
- occupation: Tech-support, Craft-repair, Sales, etc.
- relationship: Wife, Own-child, Husband, etc.
- race: White, Amer-Indian-Eskimo, Black, etc.
- sex: Female, Male
- capital-gain: continu
- capital-loss: continu
- hours-per-week: continu
- native-country: United-States, Cambodia, England, etc.

2.1 Preprocessing

Com s'utilitzarà l'algorisme ID3, és necessari fer un tractament inicial per optimitzar les dades que li entrarà al programa.
Per això, s'han tret alguns dels atributs ja que no eren útils per l'anàlisi del problema. Aquests atributs són:

- *fnlwgt*, ja que no afecta a la classificació.
- *education-num*, ja que és una representació numèrica d'education i per tant és redundant.
- *capital-gain*, degut al gran nombre de 0s que té.
- *capital-loss*, també degut al gran nombre de 0s que té.

La resta de canvis i optimitzacions tenen a veure amb els *missing values*, i s'explicaràn a l'apartat 3.2. [1]

[1] Tractament de missing values

3 Solució proposada

La solució del problema es basarà en la implementació de les funcions i classes que permeten definir i entrenar un arbre de decisió des d'un punt de vista general (és a dir, sense fer específica la implementació per al conjunt de dades donat), així com per definir mètodes de validació i random forests.

En el desenvolupament d'aquest projecte s'ha seguit una estratègia TDD. Per tant, s'han desenvolupat un conjunt de tests *unittest* de python que comprovessin que els resultats de les funcions i objectes desenvolupats fossin els esperats en un conjunt de casos simples pels quals ja es coneix la solució, ja que s'ha calculat de forma manual.

Com ja s'ha comentat a la introducció, la implementació dels arbres de decisió s'ha dividit en quatre apartats. A més també s'han implementat els mètodes de validació.

3.1 Arbres de decisió bàsics

En aquest apartat s'explicarà l'estructura de dades que s'ha construït per a suportat els arbres de decisió. S'han implementat dues classes:

1. **DecisionTree**: Aquesta classe funciona com a interfície de l'arbre. Proporciona un conjunt de mètodes bàsics que seran cridats per a entrenar i utilitzar un arbre de decisió:

   - **Constructor:** Per a construïr l'arbre de decisió se li hauran de passar tres paràmetres:
     - **attr_headers:** Consisteix en una llista que contidrà el nom dels atributs amb els que haurà de treballar l'arbre de decisió.
     - **continuous_attr_header:** Aquest paràmetre serà una llista que especifiqui els noms dels atributs[2].
     - **criterion**: Constant que indicarà quin criteri de divisió haurà de seguir l'arbre de decisió per a decidir amb quin atribut etiquetar cada node de decisió. S'ha implementat l'ID3, que triarà l'atribut que maximitzi el guany d'entropia, que per a un conjunt de dades S i un atribut A valdrà

       $$gain(S, A) = entropy(S) - \sum_{v \in A} \frac{|S_v|}{|S|} entropy(S_v)$$

       i el CART, basat en el guany d'Índex de Gini, que per a un conjunt de dades S, un conjunt de classes y i un atribut A valdrà

       $$giniGain(S, A) = Gini(y, S) - \sum_{v \in A} \frac{S_v}{S} Gini(y, S_v).$$

   - **fit(X, Y):** Entrena l'arbre de decisió a partir d'un conjunt de vectors de característiques X, i un conjunt de variables objectiu Y, que indiquen la classe de cada vector de característiques en X.
   - **predict(X):** Amb l'arbre prèviament entrenat, retornarà la predicció per al vector de característiques X

2. **SubsTree**: Es tractarà de la classe fonamental dels arbres de decisió. Cada objecte d'aquesta classe representarà un node de decisió en l'arbre. Cada objecte d'aquesta classe guardarà per quin atribut està ramificant. A més, contindrà un diccionari que permetrà accedir als seus nodes fills, que poden ser nodes de decisió representats per altres objectes de la classe SubTree, o bé fulles de l'arbre, que a la practica seran strings que contindran el nom de la classe corresponent.

[2] Els atributs indicats en el paràmetre *continuous_attr_header* també han d'aparèixer en el paràmetre *attr_headers*.

Les claus en aquest diccionari vindran donades pels diferents valors que pot prendre l'atribut associat al node de decisió. Per tant, a nivell conceptual, les claus del diccionari de nodes fills seran les arestes de l'arbre de decisió.

D'aquesta manera, la classe DecisionTree contindrà un objecte de la classe SubTree, que correspondrà a l'arrel de l'arbre de decisió, i a partir del qual es pot accedir a la resta de l'arbre.

Aquesta classe implementarà dues funcionalitats bàsiques:

(a) **Creació de l'arbre de decisió:** Inicialment, un objecte de la classe SubTree haurà de decidir quin és l'atribut pel que haurà de ramificar. Amb aquest propòsit es defineix el mètode *select_attribute*, que a partir de les dades que se li han donat a l'objecte de la classe SubTree, calcularà quin és l'atribut que maximitza el guany (d'entropia o índex de gini, segons el criteri seleccionat).

Un cop trobat l'atribut pel que ramificar, executarà el mètode *develop_child_nodes*. Aquest mètode, per cada valor possible de l'atribut seleccionat, s'encarregarà de decidir quin haurà de ser el node fill corresponent en l'arbre, i de crear-lo i emmagatzemar-lo en el diccionari de nodes fills. Es diferencien 4 casos posibles:

- En cas que l'atribut pel que s'està ramificant sigui l'últim atribut disponible per a ramificar, tots els nodes fills seran fulles de l'arbre. La fulla obtinguda a partir del valor v de l'atribut contindrà la classe més probable entre les dades amb aquest valor.
- Si totes les dades amb un valor concret de l'atribut pel que s'està ramificant pertanyen a la mateixa classe, llavors el node fill corresponent a aquest valor haurà de ser una fulla etiquetada amb aquesta calsse.
- Si per a un valor de l'atribut pel que s'està ramificant no es té cap dada, el node fill corresponent a aquest valor haurà de ser una fulla etiquetada amb la classe més comú en el node de decisió actual.
- Si s'està ramificant per un atribut que no compleix el primer cas, i es té un valor que no encaixa en la resta de situacions, llavors el node fill corresponent a aquest valor serà un nou node de decisió, és a dir un objecte de la classe tree. A aquest node se li proporcionen les dades corresponents a l'atribut i valor considerats.

D'aquesta manera, es van creant els diferents nodes de decisió recursivament, fins arribar a les fulles de l'arbre.

Cal esmentar que tot i que els punts anteriors reflecteixen el criteri de parada[3] a la pràctica hi haurà un punt afegit que s'ha hagut d'implementar explícitament per raons tècniques, tal i com s'explicarà en l'apartat de dificultats trobades.

3.2 Tractament de missing values

3.3 Tractament d'atributs continus

3.4 Random forests

3.5 Validació

[3] Criteri en un arbre de decisió per decidir quan parar la expansió dels nodes.

4 Problemes trobats i solucions proposades

La primera dificultat trobada ha estat la gran dimensió de la base de dades. En tenir un volum de dades tan gran, ha obligat a invertir molts esforços en l'optimització de la implementació dels diferents algorismes per tal que el temps d'execució sigui acceptable, tant pere l'entrenament com a l'hora de fer prediccions. Les optimitzacions han consistit en fer el millor ús possible de la llibreria numpy i estalviar tantes operacions com fos possible.

Una altra dificultat ha estat el fet que existien casos en què no queden atributs per dividir i, a més, tampoc hi queden dades per classificar. Tot i haver implementat els criteris de parada de cadascun dels casos per separat, no ha estat suficient i ha estat necessària la implementació d'un nou criteri de parada que contemplés aquests casos alhora.

Finalment, i relacionada amb l'anterior dificultat, ha estat la generació d'arbres en què existien nodes amb branques fulla en què el resultat de les seves classificacions era per totes el mateix. Si bé es pot considerar un problema no tan important, s'ha decidit solucionar-li implementant un nou criteri de parada per a comprovar si, en una branca on resta un atribut per dividir, cal desenvolupar node o, directament, posar-hi una fulla.

5 Resultats

6 Conclusió i possibles ampliacions

L'objectiu principal de la pràctica era aconseguir implementar un model de classificació d'arbres de decisió capaç de predir si una persona cobra més de certa quantitat o no. Aquest objectiu s'ha assolit satisfactòriament mitjançant la implementació d'un algorisme basat en els arbres de decisió ID3 i una altra en la qual se segueix el criteri de GINI. Tot això mantenint un temps de còmput acceptable i assumible per un volum de dades realista i gran, la qual cosa era una fita igualment important.

Si bé el desenvolupament del projecte no ha estat senzill, sinó tediós, gràcies a aquest s'han assimilat els conceptes, algorismes i criteris dels arbres de presa de decisió mitjançant una programació a més baix nivell del que acostumen a presentar les llibreries d'intel·ligència artificial com sklearn. Així, s'ha assolit i fonamentat un coneixement que permetrà entendre i treballar millor, de fet, amb aquestes llibreries i a l'hora de configurar els hiperparàmetres que treguin el màxim rendiment de cada model.

Tenint tot això en consideració, una ampliació i millora en un treball futur podria ser la implementació de diferents tècniques de prunning per a poder incrementar la generalització de les dades per part de l'arbre. D'aquesta manera, es podrien comparar els rendiments dels models dels arbres sense podar, els podats i els implementats mitjançant els random forests.

També es podrien desenvolupar altres tipus d'ensemble methods com l'adaboost o el bagging, els quals podrien arribar a rivalitzar amb el rendiment dels random forests, tot i seguir estràtegies diferents. Tot això tunejant els respectius hiperparàmetres i dibuixant diferents gràfiques de rendiment com les corbes ROC i/o PR.

Anant més enllà es podria provar de programar altres tipus de models de classficació com podrien ser el KNN o, inclús, d'altres probabilístics com el naïve bayes. D'aquesta manera, hi hauria més probabilitats de trobar un model que s'ajusti millor a la base de dades treballada.

No cal oblidar, però, que cada base de dades és diferent i no existeix un model únic i universal que sigui capaç de generalitzar el coneixement a totes les dades existens. És per aquesta raó que té sentit provar nous tipus de model com els proposats anteriorment.