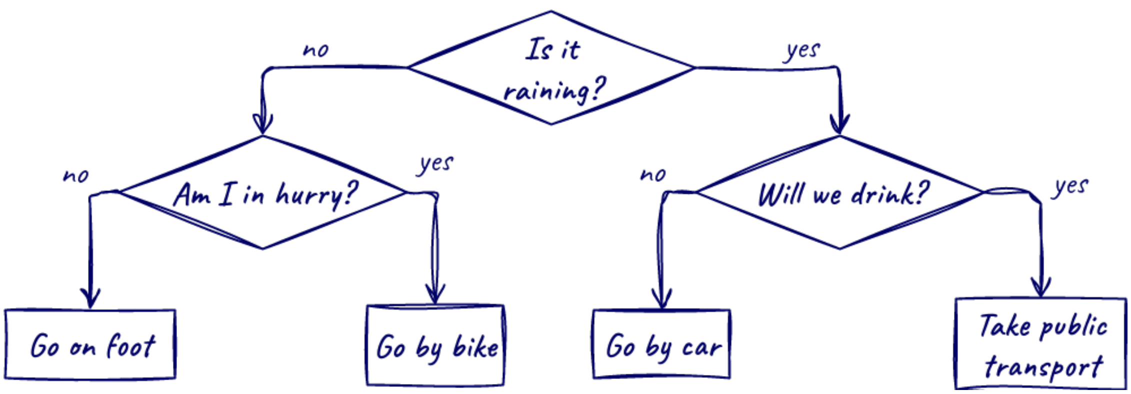
3. házi feladat

November 11, 2022

1 Elméleti háttér

A döntési fák (decision tree) talán az emberi gondolkodáshoz legközelebb álló, intuitív osztályozók (classifier). Általában relatíve kevés adattal is hatékonyan taníthatóak, és a struktúrájuk *emberi szemmel* is érthető. Egyszerűségük ellenére is igen jó előrejelző teljesítményt nyújtanak, így sikeresen alkalmazhatóak különböző területeken az orvosi döntéstámogatástól kezdve, az üzleti életen át az ajánló rendszerekig (recommender systems).

Mivel a döntési fák tanítása igen egyszerűen implementálható, a házi feladat tárgya egy döntési fa létrehozása és tanítása lesz.



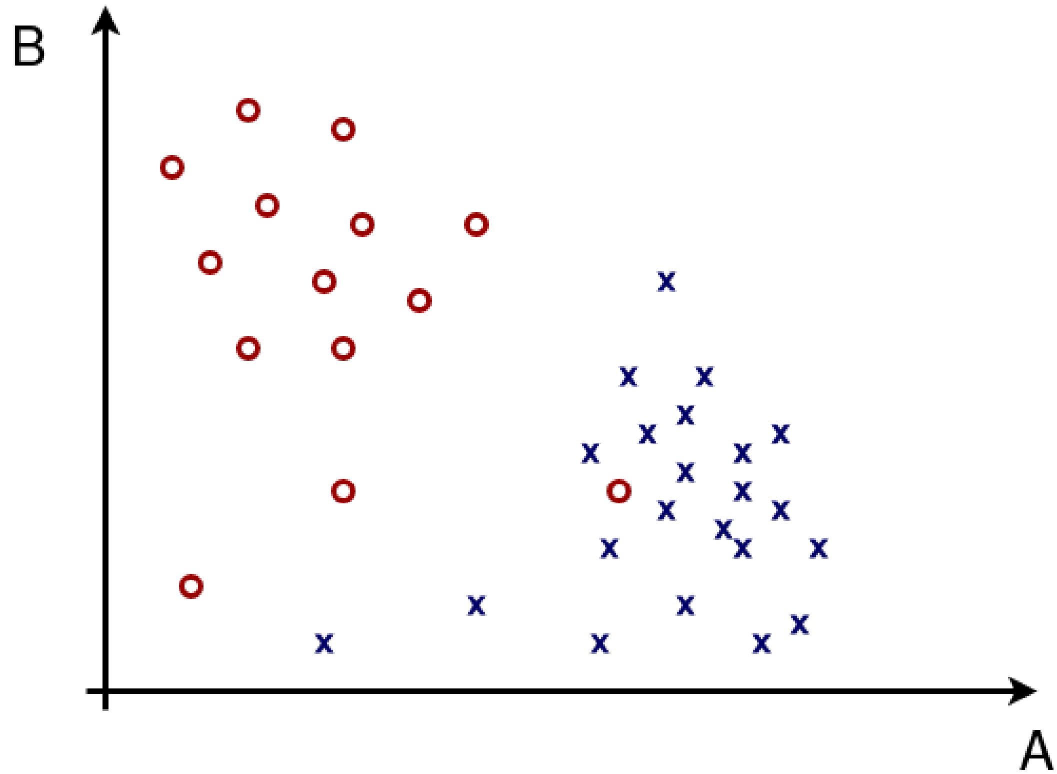
1.1 Áttekintés

Osztályozási feladatoknál az adatunk megadható táblázatos formában, például így:

Tulajdonság_A	Tulajdonság_B	Címke
1	2	o
2	7	o
12	4	x
...	...	...

Ebben a példában a Tulajdonság_A és Tulajdonság_B oszlopokban valamilyen ismert tulajdonságok (feature) szerepelnek, míg a Címke (label) oszlop a döntés kimenetelét szimbolizálja. Például, ha $A = 2$ és $B = 7$, akkor a célváltozónk az o kategóriába fog esni, azonban $A = 12$ és $B = 4$ esetén az x-be.

Ugyanez természetesen az $(A \times B)$ térbe is felrajzolható[1], valahogy így:



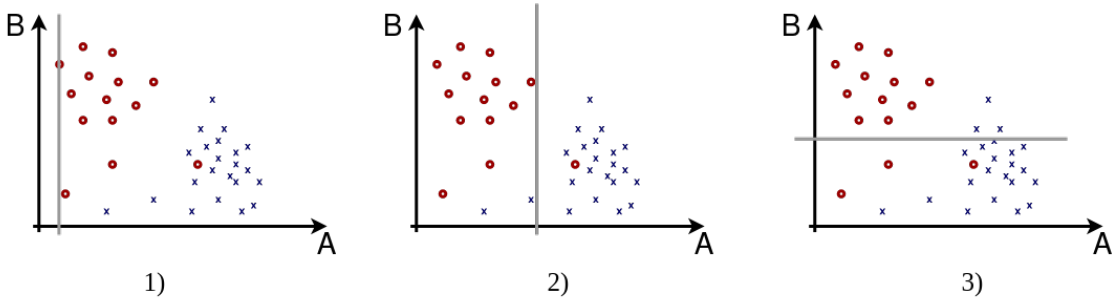
Klasszifikációs probléma esetén a feladatunk tulajdonképpen az, hogy megadjuk azt az alakzatot, amely elvágja egymástól (szeparálja) a kék x-eket a piros köröktől.

1.2 Szeparáció és a döntési fa struktúrája

Körvonalazódni látszik tehát egy algoritmus, mely a tulajdonságtér felosztásával megoldja a klasszifikációs feladatunkat. A célunk tehát az, hogy olyan rekurzív felosztásokat hozzunk létre, melyek végeredményként a lehető legjobban elválasztják a különbözően címkézett adatpontokat. Ezt úgy is megfogalmazhatjuk, hogy a felosztások két oldalán kialakuló adatpontok rendezettlensége legyen a lehető legkisebb.

Első közelítésben (és nem mellesleg ahhoz, hogy döntési fákat kapjunk) próbáljuk meg a tulajdonságok terét a tulajdonságok értékeinél egy-egy egyenessel (grafikusan: a tengelyekkel párhuzamos egyenesekkel) két részre bontani.

[1] Két tulajdonság esetén a probléma könnyen ábrázolható a síkon, viszont n tulajdonság esetén n-dimenziós terekkel kell dolgoznunk. Ezt persze már nehéz elképzelni grafikusan, de egy konkrét változó lefixálása mellett a bemutatott módszer általánosítható $n-1$ dimenziós hipersíkokkal.



Figyeljük meg a különböző felosztáspéldákat! Az 1) esetben nem nyertünk sokat, legfeljebb annyit, hogy egyetlen piros o-ról tudjuk, hogy ettől az egyenestől balra található, azonban tőle jobbra teljesen vegyesen vannak o és x címkéjű elemek.

A 2) és 3) felosztási lehetőséget jobban megvizsgálva láthatjuk, hogy a 2)-es bizonyos értelemben jobb, mint a 3)-as: a 2)-es egyenestől balra 2 darab x címkéjű elem található, míg tőle jobbra egyetlen o. Eközben a 3)-as esetében az elválasztó egyenes alatt 3 o található, felette pedig 2 x.

Ahhoz, hogy ezt a minőségi eltérést számszerűsíteni tudjuk, meg kell mérnünk a rendezetlenséget, hiszen a rendezetlenség kapcsolatba hozható a helyesen és helytelenül osztályozott példákkal.

1.2.1 Entrópia

Kódolástechnikából vagy információelméletből ismerős fogalom lehet a rendezetlenség mértékeként a (Shannon-féle) entrópia (H):

$H = -\sum_{x \in X} p(x) \cdot \log_2 p(x)$,

ahol X a *címkeértékek* halmaza.

Ha egy halmaz teljesen rendezett, azaz csak egyféle címkét tartalmaz, akkor az entrópiája $H = 0$ lesz.

1.2.2 A döntési fa struktúratanulása

Specifikáljunk most egy igen egyszerű, mohó módszert döntési fák tanulására!

Jelöljük $H(L)$-lel a kiindulási entrópiát, és végezzünk el egy x tulajdonság menti felosztást $x \leq a$ határnál. Vizsgáljuk meg a felosztási határ alatt maradó pontok entrópiáját, és jelöljük ezt $H(L|x \leq a)$-val. A felosztás hatékonysága így nem más, mint $H(L) - H(L|x \leq a)$, amit *információnyereségnek* hívunk.

Ebből talán már következik is a mohó algoritmusunk: vegyük szisztematikusan az *összes lehetséges* szeparációt, és válasszuk ezek közül azt, aminél az információnyereség maximális. Jegyezzük fel a döntési tulajdonságot, és a döntési határt, ugyanis ez lesz a döntési fánk *döntési csomópontja.* A szeparáció által létrejött két kisebb halmazunk (melyek nem feltételnül rendezettek). Amennyiben ezek egyike teljesen rendezett, azaz csak egyféle címkét tartalmaz, így entrópiája 0.0, akkor az a döntési fa egy levele, azaz egy adekvát *döntés* lesz. Ellenkező esetben még mindig egy rendezetlen halmazzal van dolgunk, így az algoritmust ezen a részhalmazon rekurzíve folytatjuk tovább, és a korábbi döntési csomópont egyik gyerekeként fogjuk szerepeltetni az általa kapott eredményeket (részfát).

Alapvetően ez az algoritmus persze nem lesz optimális, hiszen a kapott döntési fa komplexitása túlságosan nagy lesz, mely ront a fa általánosító képességén. Azonban a házi feladat megoldásához még így is elegendő teljesítményt nyújt. A kialakult fát egyébként optimálissá lehet tenni csomópontok összevonásával, a döntési fa nyírásával (pruning).

2 Feladatleírás

Az Ön feladata az lesz, hogy implementáljon Java programozási nyelven egy döntési fát. A megoldását egyetlen, `Solution.java` állományként várjuk a Moodle rendszerében.

A döntési fa célja az lesz, hogy egy publikus adathalmazon, a California Housing Prices[2] adathalmazon végzett tanulás után eldöntse, hogy egy adott tulajdonságokkal rendelkező házat megvenne-e egy vásárló vagy sem. (A vásárló döntését véletlenszerűen sorsoljuk a házi feladat ellenőrzése közben.[3]) Minden adatot egész számokra kerekítettünk, így az előzőekben leírt algoritmussal létre lehet hozni egy döntési fát.

A feladat megoldásához a Moodle rendszerben elérhet néhány állományt:

1. `Solution.java`: a megoldás minimális vázát tartalmazó állomány. A házi feladat megoldása során célszerű ezt az állományt bővíteni.
2. `train.csv`: Tanító adathalmaz. A szokásos `csv` formátumnak megfelelő állomány (táblázatos adatokat tárolunk olymódon, hogy egy-egy rekordot az állomány egy-egy sora reprezentál, az adattagok pedig `,` (vessző) karakterrel vannak elválasztva). A feldolgozást segítendő az állományban nem szerepelnek fejlécek[4], így az első kiolvasható sor az első rekordot tartalmazza. Az utolsó oszlopban szerepel a döntés értéke, mely 0, ha a döntési hamis, azaz nem vásárolja meg a képzeletbeli vásárló az adott paraméterekkel bíró házat, illetve a döntés értéke 1, ha a megvásárlás mellett dönt.
3. `test.csv`: A tanító adathalmazhoz hasonló felépítésű adatsor azzal a különbséggel, hogy ezen állomány nem tartalmaz döntési oszlopot.

2.1 Entrópiaszámítás (1p)

Készítsen entrópiaszámító függvényt *két osztály* megkülönböztetésére. A függvény szignatúrája *legyen* a következő:

```java
public static double getEntropy(int nCat1, int nCat2);
```

A függvény paraméterei a két kategóriába eső rekordok számát jelentik.

2.2 Optimális szeparáció (2p)

Az entrópiaszámító függvény és az információnyereség felhasználásával készítsen egy optimális szeparációt megadó függvényt[5] a következő szignatúrával:

```java
public static int[] getBestSeparation(int[][] features, boolean[] labels);
```

[2] Legutolsó elérés: 2022.10.20.
[3] A vásárló döntési szabályát sorsoljuk, majd a kapott döntési szabály alapján felcímkézzük az adathalmazt.
[4] Az attribútumok sorrendje megegyezik a California Housing Prices oldalon felsorolt attribútumsorrenddel.
[5] Amennyiben több optimális szeparáció is létezik, elegendő az egyik megadása.

A függvény paraméterei a következők: - `features`: az adatsor címkéi. Az első index a rekordokat (sorokat) címzi, a második index pedig az adattagokat (oszlopokat). - `labels`: a döntések tömbje, melyek soronként indexelődnek.

A függvény kimenete egy 2-elemű tömb legyen. Az első elem adja meg azt, hogy mely tulajdonság mentén talált optimális szeparáció, azaz, hogy hanyadik oszlop szerint lehet optimálisan szétválasztani. A második elem pedig legyen az értékhatár, amely mentén az elválasztást el kell végezni!

2.3 Döntési fa (9p)

Az optimális elválasztásokat kereső függvény rekurzív/iteratív alkalmazásával implementáljon egy döntési fát!

2.3.1 Tanítás

A főprogram olvassa be a vele megegyező mappában található `train.csv` állományt! Az állomány segítségével, valamint az elválasztó függvényével tanítsa a döntési fát!

Jelen feladatban nem cél az optimális méretű, komplexitású fa kialakítása; így nyugodtan építhet akkora fát, melynek minden levelében az entrópia 0.0 lesz, mélységkorlátozás nélkül.

2.3.2 Előrejelzés

Ha elkészült a döntési fa tanításával, a főprogram olvassa be a vele megegyező mappában található `test.csv` állományt, és végezze el a következtetést!

A következtetés eredményét írja ki egy `results.csv`. Az kimeneti formátum az, hogy a nem megvásárolandó házakat 0-val, a megvásárolandó házakat 1-gyel jelöljük. A kimenet minden sorában pontosan egy 1-es vagy 0 szerepeljen!

3 Értékelés

Az Ön által beadott megoldást a Moodle automatikusan kiértékeli.

Ehhez le fogja fordítani az Ön által beküldött `Solution.java` állományt, és véletlenszerű bemenetekkel leteszteli a `getEntropy` és `getBestSeparation` függvényeket, így nyomatékosan kérjük, hogy tartsák be azok fent említett szignatúráját. A `getEntropy` függvény esetén 10^{-5}-nyi toleranciával ellenőrzünk, míg a `getBestSeparation` függvénynél azt ellenőrizzük, hogy valóban a lehető legjobb szeparációk közül találták-e meg egyet a függvény.

Ezután a kiértékelő le fogja futtatni az Ön által beküldött megoldást 5 különböző, véletlenszerűen címkézett adatsoron. A létrejövő `results.csv` állomány ellenőrzésével f2-score-t számítunk az Ön megoldása és a véletlenszerű osztályozás között. Beadását az 5 lefutás során legmagasabb f2-score teljesítményt (f_{2max}) nyújtó megoldása alapján pontozzuk az alábbi képlet szerint:

$\max(9, \lceil 10 \cdot f_{2max} \rceil)$.