ITAP: drugi del druge domače naloge

Rok oddaje: dan pred kvizom

Rešitve stisnite v ZIP datoteko z imenom `ime-priimek.zip`, in jih oddajte preko spletne učilnice. Poročilo, s katerim opišete postopek reševanja, ni potrebno, saj bo ocenjevanje temeljilo na kvizu.

Priložite programe, s katerimi ste naloge rešili. Programi za vsako nalogo naj bodo v svoji mapi z imenom `naloga<zaporedna številka>`. Naloge naj bodo rešene v R, saj bodo kvizi pripravljeni v tem jeziku.

Če uporabite kodo, ki ste jo našli, navedite vir. Če imate kakšno vprašanje o nalogah, se obrnite na asistenta ali profesorja, lahko pa objavite vprašanje kar na forumu.

1 Ansambli

V datoteki `podatki1.csv` se nahaja $m \times (n+1)$ tabela, v kateri so

- prave vrednosti napovedane ciljne spremenljivke y (prvi stolpec),
- napovedi $\hat{y}_{ij}$ posameznih dreves iz naključnega gozda: vrednosti $\hat{y}_{ij}$ je napoved j-tega drevesa za i-ti primer, $1 \leq i \leq m$, $1 \leq j \leq n$.

V datoteki `koristnosti1.csv` se nahaja $p \times n$ tabela, ki podaja koristnosti vhodnih spremenljivk, kot jih izračunamo za vsako drevo: (i, j)-ti element tabele podaja koristnost i-te vhodne spremenljivke za j-to drevo.

1.1 Kako podobna so si drevesa?

Ocenimo korelacijo med napovedmi dreves. Za vsak par dreves i in j izračunamo korelacijo ρ_{ij} med napovedmi ter na koncu dobljene vrednosti povprečimo. Koliko znaša povprečna korelacija? Spomnimo se: korelacijo med vektorjema vrednosti a in b dolžine m ocenimo kot

$$\rho(a, b) = \frac{\sum_{k=1}^{m}(a_k - \bar{a})(b_k - \bar{b})}{\sqrt{\sum_{k=1}^{m}(a_k - \bar{a})^2}\sqrt{\sum_{k=1}^{m}(b_k - \bar{b})^2}},$$

pri čemer sta $\bar{a}$ in $\bar{b}$ povprečji vektorjev a in b.

1.2 Optimizacija uteži

Datoteko z napovedmi razdelimo na dva dela. Pretvarjajmo se, da je prvih $m/2$ primerov validacijska množica, zadnjih $m/2$ primerov pa testna množica. Napoved gozda $\hat{y}_i$ za i-ti primer dobimo iz napovedi $\hat{y}_{ij}$ posameznih dreves preko formule

$$\hat{y}_i = \sum_{j=1}^{n} w_j \hat{y}_{ij}$$

za primerno izbrane uteži w_j. Pri prejšnji nalogi smo npr. imeli $w_j = 1/n$, za vse j. Sedaj bomo našli optimalne, tj. take, pri katerih je napaka gozda (MSE) na validacijski množici najmanjša. Da poenostavimo nalogo, ne bomo podali omejitev, kot je npr. $\sum_{j=1}^{n} w_j = 1$ ali $w_j \geq 0$.

Napako gozda na **testni** množici, ko uporabljamo optimalne uteži, označimo z e_g^{OPT}, napako, ki pripada standardnim utežem $w_j = 1/n$ pa z e_g^{STD}. Koliko znaša razlika $e_g^{\text{STD}} - e_g^{\text{OPT}}$?

1.3 Koliko dreves zadošča?

Najprej se spomnimo, kako po navadi izračunamo koristnost spremenljivke za ansambel dreves (glej algoritem 1). Tipično si najprej izberemo primerno mero koristnosti (npr. povprečno zmanjšanje nečistosti), nato izračunamo koristnost spremenljivke x za vsako drevo, končni rezultat pa je povprečje teh *drevesnih* koristnosti. Če dojemamo koristnost spremenljivke kot slučajno

Algoritem 1 *koristnostSpremenljivke*$(meraKoristnosti, x, d_1, \ldots, d_n)$

```
K = 0                                   # koristnost spremenljivke x
for i = 1, 2, ..., n do
    K = K + meraKoristnosti(d_i, x)     # izračun za vsako drevo d_i
vrni K/n
```

spremenljivko, potem vidimo, da z večanjem števila dreves njena varianca pada proti nič, torej se mora (skoraj gotovo) vrstni red spremenljivk glede na njihovo koristnost čez nekaj časa ustaliti. Ogledali si bomo, koliko dreves moramo zgraditi, da se to zgodi. Nalogo rešimo, kot je zapisano v psevdokodi algoritma 2: ta vrne najmanjši $\tilde{n}$, po katerem se vrstni red spremenljivk ne spreminja več.

Algoritem 2 *ustalitev*$(drevesneKakovosti, d_1, \ldots, d_n)$

V = vrstni red spremenljivk za ansambel vseh n dreves
for $\tilde{n} = n-1, \ldots, 1$ **do**
  $\tilde{V}$ = vrstni red spremenljivk za ansambel prvih $\tilde{n}$ dreves
  if $\tilde{V} \neq V$ **then**
    vrni $\tilde{n}+1$
vrni 1

2 Nenadzorovano učenje

2.1 Vinogradniški vodje

Če merimo oddaljenost med primeri kot kvadrat evklidske razdalje, potem je posodobitveno pravilo za središča skupin v metodi k-voditeljev znano: novi centroid je kar povprečje skupine primerov. Kakšno je posodobitveno pravilo v primeru, da za merjenje razdalje uporabljamo vinogradniško metriko?

Preden se spomnimo definicije metrike, podajmo motivacijo zanjo in si oglejmo sliko 1: če se želimo sprehoditi po vinogradu po isti vrsti (konstanten y), je vse tako kot po navadi, medtem ko moramo v primeru, da želimo vrsto zamenjati, najprej do poti, ki povezuje vse vrste (os y), iti vzdolž nje in potem iti v izbrano vrsto. Točna definicija vinogradniške metrike je torej

$$d((x_1, y_1), (x_2, y_2)) = \begin{cases} |x_1 - x_2| & ;\ y_1 = y_2 \\ |x_1| + |y_1 - y_2| + |x_2| & ;\ y_1 \neq y_2 \end{cases}.$$

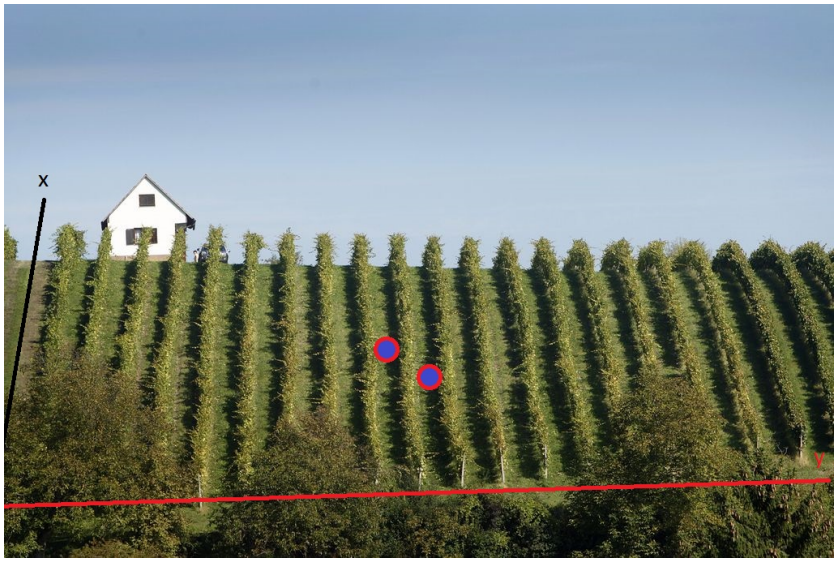
Implementirajte funkcijo `centroid(primeri)`, ki sprejme matriko primerov in vrne centroid c_0, v katerem je dosežena vrednost

$$\min_c \sum_{x \in \texttt{primeri}} d(c, x).$$

Predpostavite lahko, da je `primeri` $\in \mathbb{R}^{m \times 2}$ za neki lih $m \in \mathbb{N}$ in da bodo primeri v splošni legi (vse vrednosti v matriki bodo različne). Lihost števila m zagotavlja enoličnost rešitve.

Nekaj namigov:

- Preden se lotite te naloge, lahko poskusite izpeljati posodobitveno pravilo za Manhattan metriko, saj je podobna vinogradniški (in boste to pot v vsakem primeru morali prehoditi).
- Pri iskanju optimalnega centroida je treba obravnavati dva primera (saj je metrika d tako definirana), a preden se lotite programiranja,



Slika 1: Vinogradniška metrika. Najkrajša pot med označenima točkama (x_1, y_1) in (x_2, y_2) je pot $(x_1, y_1) - (0, y_1) - (0, y_2) - (x_2, y_2)$. Slika: `https://www.delo.si/prosti-cas/potovanja/bog-zivi-gospodarja-ki-daje-piti-brez-denarja.html`

lahko izkoristite predpostavko o splošni legi ter dokažete, da enega od primerov ni treba obravnavati - to precej poenostavi kodo.

Dodatni izzivi:

- Kako je s številom rešitev, če je velikost skupine m soda?
- Kako je z rešitvijo, ko dodatne predpostavke o splošni legi ni?
- Kako bi vinogradniško metriko razširili na podatke višjih dimenzij?

2.2 Kakovost razvrščanja v skupine

Z uporabo Wardove razdalje podatke iz datoteke `podatki22.R` hierarhično razvrstimo v skupine. Glavno vprašanje je: kakšno je najbolj smiselno število skupin (tj. kje zarezati v dendrogram)? Nanj odgovorimo tako, da z neko hevristiko izračunamo kvaliteto razbitij, ki jih porodi hierarhična razvrstitev, in izberemo najboljše (glede na to hevristiko).

Mi bomo kakovost razbitij ocenili s silhueto (*ang.* *silhouette* `https://en.wikipedia.org/wiki/Silhouette_(clustering)`). Za dane primere x_i, $1 \leq i \leq n$, in njih razbitje na k skupin S_j jo izračunamo na sledeč način.

Silhueto s najprej izračunamo za posamezne primere x_i, zato naj bo $S(i)$ skupina, v katero spada primer x_i. Če je $|S(i)| = 1$, potem definiramo $s(i) = 0$. Sicer najprej izračunamo

$$\begin{aligned} a(i) &= \frac{1}{|S(i)| - 1} \sum_{x \in S(i) \setminus \{x_i\}} d(x_i, x_j), \\ b(i) &= \min_{S \neq S(i)} \frac{1}{|S|} \sum_{x \in S} d(x_i, x). \end{aligned}$$

Število $a(i)$ pove, kako blizu primerom iz svoje skupine je x_i (želimo si majhnih vrednosti), medtem ko $b(i)$ pove, koliko je najmanjša povprečna oddaljenost x_i od preostalih skupin (želimo si velikih vrednosti). Končno dobimo

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} \in [-1, 1],$$

ki pove, kako dobro je bil v skupino razvrščen x_i: višji kot je $s(i)$, bolje je. Imenovalec ulomka je prisoten zaradi normalizacije. Kakovost samega razbitja $\{S_1, \ldots, S_k\}$ ocenimo kot povprečno vrednost $s(i)$, tj.

$$s(\{S_1, \ldots, S_k\}) = \frac{1}{n} \sum_{i=1}^{n} s(i).$$

Kateri je optimalni k (glede na silhueto) za naše podatke? Za d vzemite kar evklidsko razdaljo.

Opomba: Funkcija za izračun hierarhičnega razvrščanja je sicer že napisana (najdemo jo v `ward.R`), a si jo vseeno dobro oglejte, saj jo bo treba morda na kvizu malo spremeniti ...

2.3 Pekarstvo Edinburg

Pekarna *The Bread Basket* (iz starega mestnega jedra Edinburga) je obelodanila del podatkov o tem, kaj prodajajo (`https://github.com/viktree/curly-octo-chainsaw/blob/master/Bakery%20Transactions.ipynb`). V datoteki `podatki23.csv` se nahajajo transakcije: vsaka je v svoji vrstici in je opisana kot niz kupljenih stvari, ločenih s podpičji.

Za pogoste bomo proglasili množice stvari, ki se med transakcijami pojavijo vsaj 20-krat. Kolikšna je vsota pojavitev vseh pogostih množic velikosti vsaj tri?

Namig: Je res treba uporabiti APRIORI?