ITAP: druga domača naloga

Rok oddaje: 18. maj 2020

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 Drevesa

1.1 Učinkovit izračun variance

Na predavanjih ste omenili, da je časovna zahtevnost gradnje dreves enaka $\mathcal{O}(mn \log n)$, pri čemer je n število primerov, m pa število vhodnih spremenljivk. Gornje (med drugim) velja ob predpostavki, da lahko v danem notranjem vozlišču, ki ga doseže $\tilde{n}$ primerov, izračunamo kakovosti vseh kandidatov za test v skupnem času $\mathcal{O}(m\tilde{n})$.

Pri izbrani vhodni spremenljivki x_i moramo biti torej sposobni posamezne kakovosti

$$Var_{\mathcal{D}}(y) - \left(\frac{|\mathcal{D}_1|}{|\mathcal{D}|} Var_{\mathcal{D}_1}(y) + \frac{|\mathcal{D}_2|}{|\mathcal{D}|} Var_{\mathcal{D}_2}(y) \right).$$

v povprečju izračunati v konstantnem času. Zgoraj smo z $\mathcal{D}$ označili vse podatke, ki dosežejo dano vozlišče, $\mathcal{D}_{1,2}$ pa sta podmnožici, na kateri razpade $\mathcal{D}$ pri uporabi testa, katerega kakovost računamo. Da ne bomo imeli težav s podmnožicami velikosti ena, bomo varianco vektorja vrednosti y (dolžine $\tilde{n}$), ki pripada dani (pod)množici podatkov $\mathcal{D}$ (velikosti $\tilde{n}$) ocenili kot

$$Var_{\mathcal{D}}(y) = \frac{1}{\tilde{n}} \sum_{i=1}^{\tilde{n}} (y_i - \bar{y})^2,$$

pri čemer smo z $\bar{y}$ označili povprečje vektorja y (deljenje z $\tilde{n} - 1$ bi lahko povzročilo težave).

Implementirajte funkcijo `vseKakovosti(y, x)`, ki sprejme vektorja (oba dolžine $\tilde{n}$) vrednosti ciljne in neke vhodne spremenljivke primerov, ki dospejo

v vozlišče. Predpostaviš lahko, da je vektor `x` urejen. Funkcija naj izračuna kakovosti vseh možnih testov, in sicer v času $\mathcal{O}(\tilde{n})$. Naj vrne največjo.

Namig: $\mathbb{E}((Y - \mathbb{E}(Y))^2) = \mathbb{E}(Y^2) - \mathbb{E}^2(Y)$, prav tako pa predpostavke o urejenosti x nismo dali brez potrebe.

Prijazno opozorilo: časovne zahtevnosti na kvizu ne moremo preveriti, lahko pa zagotovimo, da bo $\tilde{n}$ dovolj velik :)

1.2 Globina drevesa

Zapiši funkcijo `globina(model)`, ki sprejme model, ki ga dobimo s klicem `model = train(..., method="rpart", ...)`, in izračuna globino dobljenega drevesa. Drevo, ki vsebuje le koren, naj ima globino 1.

1.3 Sekajmo drevesa

Zapiši funkcijo `najdiPrvo(model)`, ki sprejme model, ki ga dobimo s klicem `model = train(..., method="rpart", ...)`, in iz njega izlušči pravilo, ki ga dobimo, če sledimo testom od korena do najbolj levega lista.

Če klic `print(model$finalModel)` prikaže model

```
1) root 1000 320 + (0.32000000 0.68000000)
  2) x2< 0.49 483 186 - (0.61490683 0.38509317)
    4) x1< 0.5 297  14 - (0.95286195 0.04713805) *
    5) x1>=0.5 186  14 + (0.07526882 0.92473118) *
  3) x2>=0.49 517  23 + (0.04448743 0.95551257) *
```

je najbolj levi list tisti s statistikami `297 14 - (0.95286195 0.04713805)`, na poti do njega pa sta pogoja $x_2 < 0{,}49$ in $x_1 < 0{,}5$. Pravilo se tako glasi

$$x_2 < 0{,}49 \wedge x_1 < 0{,}5 \Rightarrow \texttt{-}.$$

Dobljeno pravilo zapiši kot funkcijo `pravilo(x)`, ki vrne napoved pripadajočega lista za tiste vektorje `x`, ki zadoščajo pogojem, in `NULL` sicer. V pomoč so ti lahko skripta `pravila.R`.

Dodatni izziv: eden od možnih načinov iskanja kakovostnih pravil je, da najprej zgradimo drevesa, jih popolnoma razsekamo na pravila (za vsak list dobimo eno) in nato optimiziramo. Če te veseli programiranje, si lahko ogledaš, kako deluje sestopanje (oz. iskanje v globino), ter napišeš funkcijo, ki drevo popolnoma razseka.

2 Podporni vektorji

2.1 Radialno jedro

Implementiraj funkcijo `radialnoJedro(x1, x2, sigma=1)`, ki vrne vrednost Gaussovega jedra, tj. funkcijo $x_1, x_2, \sigma \mapsto \exp(-\|x_1 - x_2\|_2^2 / (2\sigma^2))$, kjer je $x_{1,2} \in \mathbb{R}^d$ za neki $d \in \mathbb{N}$.

2.2 Ločimo kroglo

Z $0 \in \mathbb{R}^5$ označimo ničelni vektor ter s $K(s, r)$ kroglo s središčem v s in polmerom r. Z radialnim jedrom bi radi ločili $K(0, 1)$ in $K(0, 2)^C$. Želimo, da velja

$$\texttt{radialnoJedro}(0, x, \sigma) \begin{cases} > b & ; x \in K(0,1) \\ < b & ; x \in K(0,2)^C \end{cases}.$$

Zapiši funkcijo `najmanjsiSigma(b)`, ki izračuna natančno spodnjo mejo intervala primernih vrednosti σ.

2.3 Jedrni trik

Oglejte si podatke, ki so podani v datoteki `podatki23.csv`. Gre za klasifikacijo, kjer smo možna razreda označili z -1 in 1. Najdite

- taka podporna vektorja p_1 in p_2,
- taka σ_1 in σ_2 ter
- tak b,

da bomo s funkcijo

$$f(x) = \texttt{radialnoJedro}(p_1, x, \sigma) + \texttt{radialnoJedro}(p_2, x, \sigma) - b$$

lahko popolnoma ločili oba razreda podatkov. Za poljuben primer (x, y) iz podatkov naj torej velja $yf(x) > 0$.

3 Nevronske mreže

3.1 Enostaven perceptron

V datoteki `podatki31.csv` se skriva klasifikacijski problem (tri vhodne spremenljivke, dvojiški izhod). Zanj bomo uporabili enostavni perceptron, zato

odločanje deluje tako:

$$(x_1, x_2, x_3) \mapsto v = w_0 + \sum_i w_i x_i \mapsto \hat{y} = \text{predznak}(v)$$

Uteži w_0, w_1 in w_2 fiksiramo, radi pa bi našli primeren w_3, tako da bo perceptron popolnoma ločil naša razreda. Zapiši funkcijo, ki ob danih vhodnih podatkih najde natančno spodnjo mejo za primerne w_3.

3.2 Kako velike so mreže?

Vsak spodoben seznam slavnih nevronskih mrež bi moral vsebovati mrežo VGG16. Vprašanje pri tej nalogi je preprosto: koliko uteži ima mreža VGG16?

Stvar si bomo poenostavili do te mere, da

- bomo privzeli črnobele vhode (tj. en kanal na vhodnem sloju),
- ne bomo uporabljali dodatnih obrob: mislimo si, da konvolucijo ali akumulacijo začnemo levo zgoraj in se nato premikamo z danimi koraki, dokler se lahko. Na sliko dimenzije 3×3 torej filter dimenzije 2×2 postavimo na
  - $4 = 2 \cdot 2$ načine, če se premikamo s korakom 1 (v obeh dimenzijah),
  - en sam način, če se premikamo s korakom, večjim od 1 (v obeh dimenzijah).

Arhitekturo mreže VGG 16 opisuje tabela 1. Stolpec oblika (na izhodu iz sloja) je podan, kjer je treba (vhod, polnopovezani sloji), obliko izhoda konvolucije ali zbiranja pa opišimo tu.

Če ima trenutni vhod obliko *vrstice* $\times$ *stolpci* $\times$ *kanali* (kar je relevantno že po prvih konvolucijah, ko število kanalov naraste na 64), potem že vemo, kako do prvih dveh dimenzij izhoda, saj sta določeni z obliko filtra in korakom (ter vhodom). S tretjo dimenzijo je pa tako:

- Po zbiralnem sloju se tretja dimenzija ne spremeni, saj vsak kanal obravnavamo ločeno.
- Po konvolucijskem sloju je tretja dimenzija enaka številu filtrov, saj ima filter poleg podanih širine in višine tudi globino, ki je enaka številu kanalov.

Pri računanju pazite na sledeče:

- zbiralnih slojev ne optimiziramo in ne vsebujejo uteži,
- ne pozabite na proste člene (npr. filter velikosti 3×3 določa $9 \cdot kanali + 1$ uteži).

Tabela 1: (Malo poenostavljena) arhitektura mreže VGG16. Število 16 se nanaša na 16 lokacij, kjer so uteži. Zadnji sloj je tudi izhodni. (Povzeto po `https://towardsdatascience.com/step-by-step-vgg16-implementation-in-keras-for-beginners-a833c686ae6c`)

vrsta sloja	oblika	število filtrov	oblika filtra	korak
vhodni sloj	$(224, 224, 1)$			
konvolucija		64	$(3, 3)$	$(1, 1)$
konvolucija		64	$(3, 3)$	$(1, 1)$
zbiranje			$(2, 2)$	$(2, 2)$
konvolucija		128	$(3, 3)$	$(1, 1)$
konvolucija		128	$(3, 3)$	$(1, 1)$
zbiranje			$(2, 2)$	$(2, 2)$
konvolucija		256	$(3, 3)$	$(1, 1)$
konvolucija		256	$(3, 3)$	$(1, 1)$
konvolucija		256	$(3, 3)$	$(1, 1)$
zbiranje			$(2, 2)$	$(2, 2)$
konvolucija		512	$(3, 3)$	$(1, 1)$
konvolucija		512	$(3, 3)$	$(1, 1)$
konvolucija		512	$(3, 3)$	$(1, 1)$
zbiranje			$(2, 2)$	$(2, 2)$
konvolucija		512	$(3, 3)$	$(1, 1)$
konvolucija		512	$(3, 3)$	$(1, 1)$
konvolucija		512	$(3, 3)$	$(1, 1)$
zbiranje			$(2, 2)$	$(2, 2)$
polnopovezan	4096			
polnopovezan	4096			
polnopovezan	2			

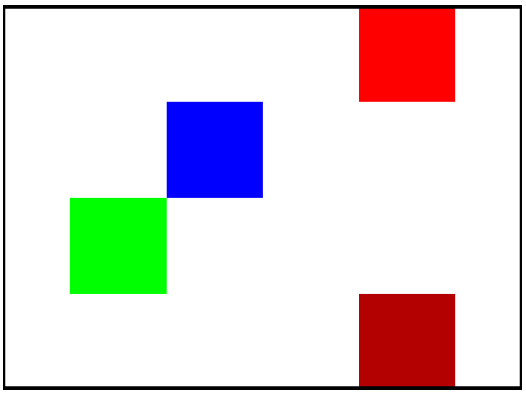
3.3 Odkrijmo moder kvadrat

Barvno sliko predstavimo s tenzorjem $X \in [0, 1]^{vrstice \times stolpci \times 3}$, pri čemer kanali po vrsti pripadajo rdečemu, zelenemu in modremu delu slike: $X[, , 3]$ torej opisuje, kako modra je slika. Zapiši funkcijo, ki sprejme barvno sliko

in odkrije položaj (kot par (x, y)) središča modrega kvadrata velikosti 5×5 pikslov. Predpostavite lahko, da

- je ozadje slike belo, tj. za vse (i, j) na območju, kjer ni kvadratov, velja $X[i, j, 1] = X[i, j, 2] = X[i, j, 3] = 1$;
- je na sliki natanko en moder kvadrat;
- so na sliki zgolj kvadrati in da imajo vsi stranico 5;
- se kvadrati ne prekrivajo;
- je vsak kvadrat ali povsem zelen ali povsem moder ali povsem rdeč, in sicer cel istega odtenka. Primer: za vse (i, j) na območju, na katerem je moder kvadrat, torej velja $X[i, j, 1] = X[i, j, 2] = 0 < X[i, j, 3] = c \in (0, 1]$, za neko konstanto c.

Primer slike (s tokrat izjemoma dodano črno obrobo) najdemo na sliki 1.



Slika 1: Primer slike s kvadrati v odtenku ene od treh barv na beli podlagi (na sliki sta rdeča kvadrata v različnih odtenkih). Črna obroba je dodana izjemoma.