Kratek pregled vsebine izbirnega tekmovanja iz umetne inteligence

Programski odbor izbirnega tekmovanja iz umetne inteligence

September 2026

1 Priprava podatkov

Priprava podatkov (data preprocessing) je faza strojnega učenja, v kateri surove podatke spremenimo v obliko, primerno za učenje modelov. Tudi najbolj sofisticiran algoritem je nemočen, če so podatki nepopolni, napačno strukturirani ali z zavajajočimi vzorci. V praksi priprava podatkov pogosto zavzame velik del projekta: od razumevanja **tipov podatkov** (numerični, kategorični, ordinalni, časovne vrste, slike, besedilo), preko obravnave **manjkajočih vrednosti** in **šuma**, do pravilne **delitve na učne in testne množice**, normalizacije in pretvorb.

Intuitivno je priprava podatkov podobna pripravi sestavin pred kuhanjem zahtevne jedi. Tudi če imamo vrhunski recept (algoritem), iz pokvarjenih ali napačno narezanih sestavin ne bo nastala dobra jed. Najprej odstranimo pokvarjene dele (očiten šum, očitne napake), nato operemo, narežemo in stehtamo sestavine (standardizacija, oblikovanje značilk, normalizacija), in na koncu vse razporedimo po posodah (učna, validacijska, testna množica), da jih lahko uporabljamo na pravi način.

Eden pomembnih korakov je obravnava **manjkajočih vrednosti**. Preprost pristop je **izločitev vrstic ali stolpcev** z veliko manjkajočimi podatki, a s tem lahko izgubimo pomembne informacije. Pogostejši je pristop **imputacije** (imputation): manjkajoče numerične vrednosti nadomestimo s **povprečjem**, **mediano** ali naprednejšimi metodami (npr. napoved manjkajoče vrednosti s kopiranjem najbolj podobnega primera), kategorične pa z **najpogostejšim razredom** ali posebnim razredom oz. oznako. Pri tem moramo paziti, da imputacijo izvajamo samo na učni množici in enako pravilo nato uporabimo na validacijski/testni množici, da se izognemo **podatkovnemu puščanju**, ki lahko prikaže rezultate kot preveč optimistične.

Ključni korak je pravilna **delitev podatkov** na:

- **Učno množico** (train): na njej se model uči.
- Validacijsko množico (validation): služi za izbiro hiperparametrov, arhitektur in odločitev, kdaj ustaviti učenje (zgodnje ustavljanje), ampak ni vedno potrebna.
- **Testno množico** (test): uporabi se samo na koncu za pošteno oceno končnega modela na prej nevidenih podatkih.

Tipično razmerje je npr. 60–80% učna, 10–20% validacijska, 10–20% testna množica, odvisno od velikosti podatkov. Pri **klasifikaciji** pogosto uporabimo **stratificirano vzorčenje**, da ohranimo podobno razmerje razredov v vseh množicah. Pri **časovnih vrstah** pa delitve ne smemo delati naključno, temveč po času (učna preteklost, test prihodnost), da ne pride do uhajanja informacij iz prihodnosti v preteklost ali delitve sosednjih primerov v učno in testno.

Pomemben del priprave je tudi rokovanje z različnimi **tipi podatkov**. **Numerične spremenljivke** pogosto **standardiziramo** (odštejemo povprečje in delimo z odklonom) ali **normaliziramo** (npr. skaliranje v interval $[0, 1]$), da pospešimo učenje in preprečimo, da bi spremenljivke z velikimi vrednostmi dominirale nad ostalimi. **Kategorične spremenljivke** (npr. barva = rdeča/zelena/modra) lahko pretvorimo ali kodiramo

z **eničnim kodiranjem** (one-hot encoding). **Ordinalne spremenljivke** (npr. ocena 1–5) moramo posebej obravnavati: običajno ohranimo naravni vrstni red ($1 < 2 < 3 < 4 < 5$), a pazimo, da ne pretiravamo z interpretacijo razdalj.

Kritično je tudi **odkrivanje in obravnava izstopajočih vrednosti oz. osamelcev** (outliers) ter **neuravnoteženih razredov**. Osamelci lahko izkrivijo porazdelitev in zmedejo model, zato jih včasih omejimo ali pa celo odstranimo. Pri neuravnoteženih razredih (npr. 99% negativnih in 1% pozitivnih primerov) je nujno, da uporabimo ustrezne **metrike ocenjevanja** (npr. F1-natančnost) in se ne zanašamo le na najbolj poznane ter intuitivne (npr. točnost).

V kontekstu AI olimpijade je priprava podatkov ena ključnih veščin, ki loči dobre tekmovalce od odličnih. Prepoznavanje potreb po ustreznih metodah navedenih zgoraj je ključni predpogoj za treniranje dobrih modelov, ki so tako uspešni v napovedih, kot splošni ter robustni.

Predlogi za nadaljnje branje:

1. Jason Brownlee, *Data Preparation for Machine Learning: Data Cleaning, Feature Selection, and Data Transforms in Python*.

2 Nadzorovano učenje

Nadzorovano učenje (supervised learning, SL) je veja strojnega učenja, kjer se model uči iz *označenih primerov*. Vsak učni primer je par (x, y), kjer je x vhod (npr. slika, besedilo, vektor meritev), y pa izhod (npr. razred mačka/pes ali realno število kot je cena stanovanja). Cilj je poiskati tak model (funkcijo), ki se nauči preslikave iz vhodov v izhode tako, da za nove, še nevidene primere napoveduje čim bolj pravilno. Tipični podnalogi sta **klasifikacija** (napovedovanje omejenega števila diskretnih razredov) in **regresija** (napovedovanje v principu neskončno možnih realnih vrednosti).

Intuitivno je nadzorovano učenje zelo podobno klasičnemu šolskemu pouku s primeri in rešitvami. Učitelj učencu pokaže nalogo (vhod) in pravilno rešitev (izhod). Učenec najprej rešuje podobne naloge s "plonkanjem" ali napakami, nato pa skozi veliko število vaj, primerov in povratne informacije prepozna vzorce: kaj v besedilu naloge je pomembno, kako uporabiti formulo, kako prepoznati tip naloge. Sčasoma učenec zna reševati tudi nove naloge, ki jih ni še nikoli videl, a so dovolj podobne učnim primerom.

Osrednji izziv nadzorovanega učenja je **posploševanje** (generalization): model se ne sme zgolj "na pamet" naučiti učnih primerov, temveč mora pravilno napovedovati tudi za nove podatke iz enake porazdelitve. Pri tem naletimo na dilemo intenzitete **prileganja** (underfitting vs. overfitting). Model s *prenizko* kompleksnostjo ne zmore ujeti strukture podatkov (underfitting), model s *previsoko* kompleksnostjo pa si lahko "zapomni šum" v učnih podatkih (overfitting) in na testu s povsem novimi podatki odpove. Tipične rešitve tega problema vključujejo **regularizacijo** (dodajanje kazni za prevelike uteži), ločene validacijske množice za izbiro hiperparametrov, prečno preverjanje ter skrbno pripravo in razumevanje podatkov.

Eden najbolj znanih primerov nadzorovanega učenja je klasifikacija na podatkovni množici Iris, ki vsebuje 150 primerov cvetov rož s štirimi značilkami (dolžina in širina čašnih ter venčnih listov) ter pripadajočo vrsto rože (trije razredi). Naloga modela je, da se iz označenih primerov nauči ločiti med vrstami na podlagi danih značilk in nato pravilno razvrsti nove, še nevidene primere. Zaradi majhnosti, dobro razumljene strukture in delne linearne ločljivosti razredov je Iris postal klasičen primer za demonstracijo in primerjavo različnih klasifikacijskih algoritmov.

V kontekstu AI olimpijade nadzorovano učenje predstavlja hrbtenico potrebnega znanja. Klasifikacija in regresija sta stebra strojnega učenja tako v resničnem svetu kot na tekmovanjih, zato je dobro razumevanje ključno za razvoj dobrih (ne nujno najbolj kompleksnih) napovednih modelov.

Predlogi za nadaljnje branje:

1. Christopher M. Bishop, *Pattern Recognition and Machine Learning*, Springer.
2. G. James, D. Witten, T. Hastie, & R. Tibshirani, *An Introduction to Statistical Learning*, Springer.

3 Spodbujevano učenje

Spodbujevano učenje oziroma okrepitveno učenje (reinforcment learning, RL) je veja umetne inteligence, kjer se agent (model) nauči uspešnega (v idealnem primeru pa optimalnega) vedenja v določenem okolju skozi izvajanje akcij. Izvedene akcije so retrospektivno nagrajene v primeru, če akcije vodijo do zmage oz. uspeha ter kaznovane, če vodijo do poraza oz. neuspeha. Vsaka akcija v določenem stanju okolja ima tako svojo oceno kvalitete, naučeni agenti pa pri svojem vedenju izbirajo akcije z boljše oceno.

Pristop temelji na podobnih konceptih kot učenje oz. dresiranje domačih živali in malčkov. Na primer, če naš pes v fazi dresure nekaj naredi prav, ga nagradimo s prigrizkom, če nekaj naredi narobe, nagrade ne dobi ali pa ga nahrulimo. Skozi čas se tako pes nauči "kaj je prav in kaj narobe" ter vzpostavi primerno vedenje za okolje v katerem živi.

Matematični okvir za večino RL problemov je **Markov proces odločanja** (MDP). MDP določa strukturo učenja s štirimi ključnimi komponentami: **Stanja** (trenutno stanje agenta, npr. postavitev figur na igralni površini), **Akcije** (poteze, ki jih agent lahko izvede, npr. premik kraljice za 2 polji desno), **Nagrade** (numerični signal, običajno apliciran po končani partiji, npr. +10 za vse akcije v primeru zmage, -10 v primeru poraza), / in **Strategija** (logika, ki agentu pove, katero akcijo izbrati v danem stanju).

Glavni izziv je najti najboljšo dolgoročno strategijo. Pri tem pomagajo **Bellmanove enačbe**, ki izračunajo vrednost trenutnega stanja tako, da ga povežejo s takojšnjo nagrado in pričakovano vrednostjo prihodnjih stanj. Na teh enačbah temelji algoritem **Q-učenja**, ki ustvari tabelo (Q-tabela), ki služi kot "plonk listek" in shranjuje največjo pričakovano nagrado za določeno akcijo v določenem stanju.

Ključen problem pri RL je dilema **raziskovanja** (Exploration) proti **izkoriščanju** (Exploitation): agent mora najti ravnovesje med tem, koliko poskušati nove, neznane poti (raziskovanje), in tem, da uporabi tisto, kar že ve, da je najbolje (izkoriščanje). Raziskovanje je potrebno, da se agent lahko nauči novih dobrih strategij.

Pogost preprost primer problema, ki ga RL reši zelo kvalitetno, je igra križci in krožci. Igra ima končno omejeno stanj ter majhen nabor možnih potez v vsakem koraku. Pri vsaki igri RL sistem beleži celotno verigo potez agenta, če je agent zmagal se potezam, oz. stanjem na polju, ki ga povzročijo poteze dodeli pozitivna ocena, če je izgubil pa negativna. Sprva agent dela naključne poteze in je v igri slab, skozi čas pa postaja vedno boljši. Njegove odločitve namreč temeljijo na tabeli vseh stanj, po dovoljšnjem številu iger je v tabeli jasno določeno katere poteze vodijo do zmage, katere pa do poraza. Če naredimo sistem v katerem dva RL agenta tekmujeta eden proti drugem, se oba kaj hitro naučita dobro napdati in braniti, tako da se vse igre končajo z remijem, kar je tudi pričakovano pri dobrih igralcih te igre. Problem opisanega pristopa je v tem, da se ocena vseh akcij oz. stanj posodobi šele po vsaki končani igri. Pri križcih in krožcih so igre kratke, tako da to ni tako problematično, pri bolj kompleksnih problemih pa ta rešitev ni idealna.

Pomembna nadgradnja RL je tako učenje s časovno razliko (time difference learning, TD learning), pri katerem agent posodablja svoje znanje po vsakem koraku (namesto da bi čakal do konca naloge), tako da primerja svojo staro napoved z novo, bolj informirano.

Predlogi za nadaljnje branje:

1. Richard S. Sutton & Andrew G. Barto, *Reinforcement Learning: An Introduction*, Bradford Books.
2. Miguel Morales, *Grokking Deep Reinforcement Learning*, Manning.

4 Evalvacija modelov strojnega učenja

Evaluacija modelov strojnega učenja (Evaluation of ML Models) je ključen korak v procesu razvoja in terniranja, saj le-ta oceni **uspešnost**, **zanesljivost** ter **posplošljivost** naučenega modela. Brez ustrezne in nepristranske evaluacije ne moremo z gotovostjo trditi, ali bo model uspešno deloval na novih, nevidnih podatkih, kar je njegov glavni namen. Večina modelov se namreč uči oz. trenira na omejeni učni množici ter potem aplicira v "realnem svetu" na novih podatkih, ki v času treniranja modela še niso znani.

Temeljni pristop k evalvaciji zahteva, da celoten nabor podatkov razdelimo na vsaj dva ločena sklopa: **učni nabor** (Train set) in **testni nabor** (Test set). Učni nabor se uporablja za prilagajanje notranjih parametrov modela, torej za njegovo dejansko učenje oz. treniranje. Testni nabor se uporabi šele na koncu, za končno, nepristransko oceno posploščljivosti modela, saj testni nabor vsebuje podatke, ki jih model med učenje ni videl. Na ta način dobimo objektivno oceno kvalitete modela nad novimi podatki ter hkrati preverimo ali je med treniranjem prišlo do **prevelikega prilagajanja učnim podatkov** (overfitting).

Včasih pa delitev na le dva dela ni dovolj, še posebej, ko je treba optimizirati hiperparametre modela. Hiperparametri so dodatne nastavitve modela, ki se jih model ne nauči sam skozi traniranje, ampak jih nastavi inženir pred učenjem (npr. hitrost učenja ali globina dreves). V teh primerih dodamo še tretji, **validacijski nabor** (Validation set). Model se uči na učnem naboru, medtem ko validacijski nabor sproti služi za preskus različnih nastavitev hiperparametrov in s tem za njihovo optimizacijo. Z uporabo validacijskega nabora lahko tudi zgodaj ustavimo učenje, ko opazimo, da se uspešnost na tem naboru začne slabšati (običajno je to znak prevelikega prilagajanja učnim podatkom), čeprav se uspešnost na učnem naboru še izboljšuje. Testni nabor še vedno ohrani svojo funkcijo končne, nepristranske in edine resnične ocene, ki modela ni videl v nobeni fazi prilagajanja.

Za kvantifikacijo uspešnosti pri klasifikacijskih problemih lahko uporabljamo različne metrike. Nekatere najbolj pogosote so izpeljane iz **matrike zamenjav** (Confusion matrix). Ključne med njimi so **natančnost** (Precision), ki meri delež pravilno pozitivnih napovedi, **priklic** (Recall), ki meri delež dejanskih pozitivnih primerov, ki jih je model zaznal, ter **mera F1** (F1-score), ki ponuja uravnoteženo kombinacijo obeh. Za globljo analizo se uporabljajo tudi ROC-AUC krivulje, ki grafično ponazarjajo kompromis med **občutljivostjo** (Sensitivity) in **specifičnostjo** (Specificity). Občutljivost, znana tudi kot stopnja resnično pozitivnih primerov (True Positive Rate), je delež pravilno identificiranih pozitivnih primerov od vseh dejansko pozitivnih primerov. Specifičnost, znana tudi kot stopnja resnično negativnih pirmerov (True Negative Rate), je delež pravilno identificiranih negativnih primerov od vseh dejansko negativnih primerov.

Ključen problem, ki ga skuša evalvacija modelov strojnega učenja nasloviti, je kompromis med pristranskostjo in varianco (Bias-Variance Tradeoff), ki pomaga pri odkrivanju premajhnega ali pretiranega prileganja (overfitting). Do pretiranega prileganja pride, ko se model preveč natančno nauči vzorcev v učnih podatkov, vključno s šumom, ker ti vzorci ne obstajajo potem v realnih podatkih, model svojega znanja ne uspe posplošiti na nove podatke. To prepoznamo, ko je učna uspešnost visoka, uspešnost na validacijskem (ali testnem) naboru pa znatno nižja. Za zagotavljanje bolj robustne in zanesljive ocene zmogljivosti se uporablja metoda **navzkrižne validacije** (K-fold Cross-Validation), kjer se učni nabor razdeli na K delov in model se ponovno uči in validira K-krat, kar zmanjša odvisnost ocene od specifične delitve podatkov ter nam hkrati poda negovotovost za oceno kvalitete modela.

Predlogi za nadaljnje branje:

1. G. James, D. Witten, T. Hastie, & R. Tibshirani, *An Introduction to Statistical Learning*, Springer.
2. Géron, Aurélien, *Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow*, O'Reilly.