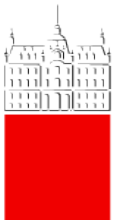
Univerza *v Ljubljani*
Fakulteta za *matematiko in fiziko*



Strojno učenje

12. naloga pri Matematično-fizikalnem praktikumu

Avtor: Marko Urbanč (28191096)
Predavatelj: prof. dr. Borut Paul Kerševan

8.9.2023

Kazalo

1 Uvod

Medtem ko se je še nedavno zdelo strojno učenje prava temna magija (vsaj meni), je pravzaprav dandanes uporaba različnih algoritmov strojnega učenja popolnoma vsakdanja in že rutinska. Ljudje se pravzaprav stalno srečujemo z različnimi algoritmi strojnega učenja, ki nam pomagajo pri različnih stvareh, kot so npr. priporočila na Netflixu, Googlovi iskalni algoritmi, različni algoritmi za prepoznavanje objektov na slikah, itd. Vse to so algoritmi, ki so zasnovani na strojnem učenju.

Prvotno sem želel pustiti prejšnji stavek brez pojasnila, vendar sem se odločil, da bom vseeno napisal nekaj besed o kar mislim. Poglejmo prvo Netflix. Netflix je spletna storitev za ogled filmov in serij. Ker je na Netflixu ogromno filmov in serij, je težko najti tisto, kar bi si želeli ogledati. Zato Netflix uporablja algoritem, ki na podlagi vaših prejšnjih ogledov in ocen priporoča filme in serije, ki bi vam lahko bili všeč [1]. Ampak ne samo to, glede na to iz katere naprave gledate, ob kakšnem času dneva in drugih parametrih, ki jim Netflix pravi Kontekst [2], pravzaprav so pa to *contextual bandits*, ki se uporabljajo v enem okusu strojnega učenja. Več o tem kasneje. V kolikor sem uspel razumeti ta članek zgleda, da še več kot to, želijo praktično *in real time* prilagajati priporočila, glede na to kakšen je vaš t.i. *intent* [**?**]. Oh in vsi, ki si med seboj delite Netflix račune, nikakor ne skrbite, tudi to vas zna nekoč tepst kakšen algoritem strojnega učenja [3].

Nadalje, Google. Google je spletni iskalnik, ki ga praktično vsi uporabljamo. Google uporablja algoritme strojnega že precej dolgo časa. Leta 2015 so predstavili deep learning sistem RankBrain, ki je bil namenjen izboljšanju rezultatov iskanja. RankBrain je bil namenjen predvsem iskanju poizvedb, ki jih Google še nikoli ni videl. RankBrain je bil zasnovan tako, da je na podlagi preteklih iskanj in rezultatov iskanj, ki so jih uporabniki izbrali, izboljševal rezultate iskanja. Od 2018 naprej so v Google Search v rabi nevronske mreže, ki so namenjene predvsem izboljšanju rezultatov iskanja. Od 2019 naprej pa Google uporablja tudi BERT [4], ki je bil ogromen korak naprej v razumevanju naravnega jezika.

Na hitro o tipih strojnega učenja oz. osnovnih vrstah algoritmov strojnega učenja. Poznamo

- Nadzorovano učenje (supervised learning)
  - Klasifikacija (classification): Sortiranje podatkov v razrede. Primer: Razpoznavanje številk na sliki.
  - Regresija (regression): Modeliranje odvisnosti med podatki. Primer: Napovedovanje cene nepremičnine.
- Nenadzorovano učenje (unsupervised learning)
- Stimulirano učenje (reinforcement learning)

Poglejmo si zdaj uporabo strojnega učenja v fizikalnem kontekstu. V fiziki se strojno učenje uporablja za različne namene. Največkrat uporabljamo algoritme prvega tipa, torej jih nadzorovano učimo. V fiziki visokih energij se

strojno učenje uporablja za razpoznavanje delcev v detektorjih, za razpoznavanje različnih pojavov, za izboljšanje različnih meritev, itd. To je nekaj kar bomo tudi sami počeli v tej nalogi. Iskali bomo Higgsov bozon v podatkih, ki jih je zbrala ATLAS kolaboracija.

1.1 Kako delujejo algoritmi strojnega učenja?

Pred tem, pa še osnovno o tem, kako sploh delujejo algoritmi strojnega učenja. Imamo nabor parametrov $\mathcal{D} = \{(x_k, y_k)\}_{k=1}^{N}$, kjer je $x_k = (x_k^1, ...x_k^M)$ naključno izbrani vektor z M lastnostmi oz. *karakteristikami*, $y_k = (y_k^1, ..., y_k^Q)$ pa je vektor Q ciljnih vrednosti, ki so lahko diskretne ali realna števila ali kaj drugega (npr. barve, ki jim tudi lahko priredimo številske vrednosti). Vrednosti (x_k, y_k) so neodvisne in porazdeljene po neznani porazdelitvi $\mathcal{P}(\cdot, \cdot)$. Cilj strojnega učenja je poiskati oz. priučiti preslikavo $h : \mathbb{R}^Q \to \mathbb{R}$, ki bo minimizirala pričakovano vrednost funkcije izgube (angl. loss function) $\mathcal{L}(h)$

$$\mathcal{L}(h) = \mathbb{E}\mathrm{L}(\vec{y}, \vec{h}(\vec{x})) = \frac{1}{N}\sum_{k=1}^{N} \mathrm{L}(y_k, \vec{h}(x_k)) \,, \tag{1}$$

kjer je $\mathrm{L}(\cdot, \cdot)$ gladka funkcija, ki opisuje oceno za kvaliteto napovedi, pri čemer so $(\vec{x}, \vec{y})$ neodvisno vzorčene iz nabora $\mathcal{D}$ po porazdelitvi $\mathcal{P}(\cdot, \cdot)$. Po koncu učenja imamo torej na voljo funkcijo $h(\vec{x})$, ki nam za nek vhodni nabor vrednosti $\hat{x}$ poda napoved $\hat{y} = \vec{h}(\hat{x})$., ki ustrezno kategorizira ta nabor vrednosti.

Funkcije $\vec{h}$ so v praksi sestavljene iz množice preprostih funkcij z prostimi parametri, kar na koncu seveda pomeni velik skupni nabor neznanih parametrov in zahteven postopek postopek minimizacije funkcije izgube.

1.2 Odločitvena drevesa (Decision trees)

Odločitvena drevesa so ena izmed najbolj preprostih metod strojnega učenja. Odločitvena drevesa so sestavljena iz vozlišč in povezav. Vozlišča so povezana z povezavami, ki so lahko usmerjene ali neusmerjene. Osnovni gradnik odločitvenega drevesa je kar stopničasta funkcija $\mathrm{H}(x_i, -t_i) = 0, 1$, ki je enaka ena za $x_i > t_i$ in nič za $x_i < t_i$, kjer je x_i ena izmed karakteristik in t_i neznani parameter. Iz skupine takšnih funkcij, ki predstavljajo binarne odločitve lahko skonstruiramo končno uteženo funkcijo

$$\vec{h} = \sum_{i=1}^{J} \vec{w}_i \mathrm{H}(x_i, -t_i) \,, \tag{2}$$

kjer so $\vec{w}_i$ vektorji neznanih uteži. Tako t_i kot $\vec{w}_i$ lahko določimo, v procesu učenja. Nadgradnjo odločitvenih dreves predstavljajo pospešena odločitvena drevesa (angl. boosted decision trees), ki uporabljajo več odločitvenih dreves, ki so med seboj povezana. Temelijo pa na Gradient Boosting algoritmu [5], ki je bil razvit leta 1999 in je bil namenjen izboljšanju napovedi. Gradient Boosting algoritem je bil razvit za regresijske probleme, vendar se ga da uporabiti tudi za klasifikacijske probleme in je v praksi zelo uspešen.

Literatura

[1] Sudarshan Lamkhede and Christoph Kofler. Recommendations and results organization in netflix search, 2021.

[2] Harald Steck, Linas Baltrunas, Ehtsham Elahi, Dawen Liang, Yves Raimond, and Justin Basilico. Deep learning for recommender systems: A netflix case study. *AI Magazine*, 42(3):7–18, Nov. 2021.

[3] Soheil Esmaeilzadeh, Negin Salajegheh, Amir Ziai, and Jeff Boote. Abuse and fraud detection in streaming services using heuristic-aware machine learning, 2022.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina N. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. 2018.

[5] Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5), 2001.