SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

IMPLEMENTACIJA NEURONSKIH MREŽA ZA SIMBOLIČKU REGRESIJU

Mihael Pristav

1 SADRŽAJ

2 TEORIJSKI UVOD

2.1 SIMBOLIČKA REGRESIJA

Simbolička regresija je oblik regresijske analize u kojem je cilj pronaći matematičku formulu koja najbolje opisuje vezu između ulaznih varijabli i izlaza. Za razliku od klasične regresije koja pretpostavlja unaprijed određeni oblik modela (npr. linearna, polinomska), simbolička regresija traži strukturu modela i njegove parametre istovremeno. Rješenje simboličke regresije je eksplicitna, interpretativna formula. Do nedavno, simbolička regresija se gotovo isključivo temeljila na evolucijskim algoritmima, posebice genetskom programiranju. Ovi algoritmi koriste populaciju kandidata (matematičkih izraza) koji se postupno poboljšavaju kroz procese poput mutacije, križanja i selekcije.

2.2 NEURONSKE MREŽE

Neuronske mreže su računalni modeli inspirirani strukturom i funkcijom bioloških neuronskih sustava. One se sastoje od velikog broja međusobno povezanih čvorova, organiziranih u slojeve: ulazni sloj, jedan ili više skrivenih slojeva, te izlazni sloj. Svaki neuron prima ulazne podatke, primjenjuje na njih težinske koeficijente, prosljeđuje ih kroz nelinearnu aktivacijsku funkciju i šalje rezultat sljedećem sloju. Ova sposobnost oblikovanja nelinearnih odnosa između ulaza i izlaza čini neuronske mreže izuzetno fleksibilnim i moćnim modelima za aproksimaciju funkcija. Neuronske mreže posjeduju niz svojstava zbog kojih su postale temelj suvremenog strojnog učenja.

Neuronske mreže su izuzetno snažan alat za modeliranje složenih odnosa u podacima zahvaljujući svojoj sposobnosti univerzalne aproksimacije, što znači da teoretski mogu naučiti bilo koju kontinuiranu funkciju, pod uvjetom da raspolažu dovoljnom količinom podataka i računalnih resursa. Njihova fleksibilnost očituje se u primjenjivosti na različite oblike podataka, poput slika, vremenskih nizova ili tabličnih zapisa, bez potrebe za ručnim inženjeringom značajki. Umjesto toga, mreže samostalno uče korisne reprezentacije tijekom procesa treniranja. U modernim aplikacijama, neuronske mreže postižu visoke performanse i lako se skaliraju na velike skupove podataka, što ih čini nezaobilaznim komponentama u mnogim područjima strojnog učenja i umjetne inteligencije.

Učenje neuronskih mreža odvija se kroz proces optimizacije težinskih koeficijenata koji određuju ponašanje svakog neurona. Taj se proces temelji na minimiziranju funkcije pogreške između predikcija mreže i stvarnih vrijednosti, najčešće korištenjem gradijentnog spusta ili njegovih naprednijih varijanti poput Adam algoritma. Podaci prolaze kroz mrežu u tzv. fazi *forward* propagacije, pri čemu se izračunava izlaz modela. Razlika između tog izlaza i stvarne vrijednosti koristi se za izračun pogreške, koja se zatim širi unatrag kroz mrežu u fazi poznatoj kao *backpropagation*. Tijekom tog procesa izračunavaju se gradijenti, odnosno derivacije funkcije pogreške s obzirom na svaki parametar, a ti gradijenti se koriste za ažuriranje težina u mreži s ciljem postepenog poboljšavanja performansi modela.

3 EQL MREŽA

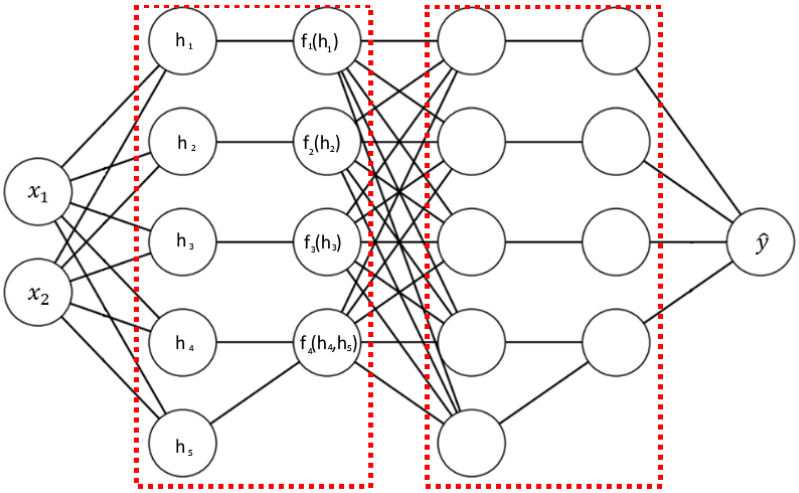
3.1 IDEJA

Temeljna ideja istražena u radu *Integration of Neural Network-Based Symbolic Regression in Deep Learning for Scientific Discovery*[1] je mogućnost rješavanja problema simboličke regresije s pomoću neuronskih mreža koje koriste specijalizirane prijenosne funkcije. Klasične neuronske mreže koriste univerzalne, ali teško interpretativne nelinearne funkcije poput ReLU ili tanh, koje omogućuju lagano računanje gradijenta. U ovom radu predlaže se zamjena tih funkcija, jednostavnijim funkcijama kao što su sinus, eksponencijalna, kvadratna i druge osnovne matematičke operacije.

Ovaj pristup omogućuje modelima da tijekom treniranja ne samo aproksimiraju odnose među varijablama, već i otkriju eksplicitne simboličke izraze koji opisuju te odnose. Uz to, korištenje standardnih mehanizama treniranja poput propagacije unatrag omogućuje učinkovitu optimizaciju mreže. ovaj pristup teži povezivanju efektivnosti učenja neuronskih mreža s jasnoćom simboličkih metoda, stvarajući modele koji su i precizni i razumljivi.

3.2 EQL MREŽA

Predložena programska implementacija ovakve neuronske mreže nazvana je EQL(*Equation Learner*) mreža (Slika 3.1). Ovakva mreža sastoji se od ponavljajućih slojeva u parovima(*na slici crveno*): jedan potpuno povezani sloj bez zajedničke aktivacijske funkcije i jedan sloj koji sadrži veći broj različitih aktivacijskih funkcija, koje kao ulaz primaju jedan ili dva izlaza prethodnog sloja.



Slika 3.1 Skica EQL mreže

$$L_{0.5}^{*} = \begin{cases} |w|^{0.5} & |w| > a \\ \left(-\frac{w^4}{8a^3} + \frac{3w^2}{4a} + \frac{3a}{8}\right)^{0.5} & |w| < a \end{cases}$$

Jednadžba 1 Modificirana L˘0.5 regularizacija

3.3 UČENJE I INTERPRETACIJA

Prednost korištenja neuronskih mreža je mogućnost učenja propagacijom unatrag. Zbog korištenja više različitih funkcija aktivacije umjesto jedne koja se primjenjuje na cijeli izlaz prijašnjeg sloja, složenost izračuna gradijenta je veća, ali I dalje zanemariva u usporedbi s izračunom potrebnim za parametre unutar potpuno povezanog sloja. Kako bi se potaknula jednostavnija rješenja predložena je modificirana L1/2 regularizacija(Jednadžba 1). Također predložen je pristup učenju koji epohe dijeli na rane epohe i kasnije epohe. Ranije epohe modificiraju sve parametre u model kao i uobičajeno, dok se prije početka kasnijih epoha, svi zanemarivo mali parametri u modelu, ručno postavljaju na 0, te se im zabranjuje daljnje prilagođavanje. Ovime se pokušava modelirati sustav koji čak i nakon ignoriranja malih vrijednosti može dobro simulirati problem. Kada je učenje gotovo, simbolički prikaz funkcije moguće je iščitati prolazom unatrag kroz model. Aktivacijske funkcije kao ulaz primaju izlaze potpuno povezanih slojeva koji su uvijek sume konstanti, varijabli i već generiranih funkcija. Na ovaj način moguće je generirati sve funkcije sastavljene od suma, aktivacijskih funkcija i njihovih kompozicija.

4 IMPLEMENTACIJA

4.1 RAZLIKE OD ORIGINALNE IMPLEMENTACIJE

Originalna implementacija napravljena je kao dokaz koncepta te je zbog toga sadržavala mnoge dijelove koji nisu nužni za funkcioniranje samog algoritma. Naša implementacija izrađena je s ciljem razumijevanja algoritma. Iz ovog razloga, u našoj implementaciji postoje razlike.

Naša implementacija sprema samo podatke o pogrešci u svakoj promatranoj epohi, za potrebe razumijevanja ovog pristupa simboličkoj regresiji , ostale informacije o pogrešci nisu nam neophodne te bi njihovo računanje i spremanje dodatno kompliciralo implementaciju.

Aktivacijske funkcije su definirane kao klase koje sadrže parametar *name* i funkciju *torch*. Parametar *name* se koristi kod generiranja završnog izgleda funkcije, a funkcija *torch* je *pyTorch* kompatibilna funkcija prolaza podataka unaprijed. Ovakav dizajn aktivacijske funkcije omogućava jednostavno dodavanje novih aktivacijskih funkcija, koje i dalje imaju sve prednosti ubrzanja i jednostavnosti korištenja koje odnosi *pyTorch*.

U originalnoj implementaciji potpuno povezani linearni sloj i sloj s aktivacijskim funkcijama su implementirani kao jedan neodvojivi sloj što znatno poboljšava vremenske performanse kod izračuna rješenja i gradijenata. U našoj implementaciji sloj *SymbolicActivation* je dizajniran da samo primjenjuje predefinirane funkcije na dani ulaz poštujući pyTorch forward pravila, kako bi se održala jednostavnost interpretacije koda. Takvi slojevi se kasnije mogu koristiti u kombinaciji s već postojećom implementacijom potpuno Povezanih slojeva i sekvencijalnih mreža u pyTorchu.

U našoj implementaciji također je promijenjeno zamrzavanje parametara, koje je sada moguće, ovisno o početnom parametru, obaviti na različitim trenutcima u učenju i potencijalno ponoviti više puta.

4.2 UPORABA

Program je podesiv, ovisno o vrsti promjene koju želimo obaviti, potrebno je podesiti parametre na jednoj od 3 lokacije.

Kod pokretanja programa kroz *main.py* , nalaze se parametri bitni za učenje neuronske mreže, kod pokretanja također se zadaje funkcija koju želimo aproksimirati. Izgled podataka za učenje i testiranje moguće je promijeniti u *utility.py*. Dok se odabir aktivacijskih funkcija za aktivacijski sloj vrši u konstruktoru *LearningInstance*.

5 ZAKLJUČAK

Algoritam se pokazao uspješan u generiranju modela koji dobro aproksimiraju zadane funkcije. Funkcije generirane na ovaj način mogu se matematički interpretirati, te se uz računski ne zahtjevne postupke pojednostavljivanja mogu svesti na formu s minimalnim razlikama od one početno zadane. Međutim u ovisnosti o kompleksnosti mreže odabrane za učenje modela, dobivene funkcije imaju zamjetni višak elemenata koji jedni druge poništavaju ili generalno ne doprinose rezultat. Ovo ukazuje na probleme s regularizacijom modela. Ovaj problem se pojavljuje i u originalnoj implementaciji, ali s nešto manjom učestalošću.

6 SAŽETAK

U ovom seminaru je prikazana implementacija i teorija oko korištenja neuronskih mreža za simboličku regresiju, na temelju pristupa predloženog u znanstvenom radu "Integration of Neural Network-Based Symbolic Regression in Deep Learning for Scientific Discovery". Cilj je bio napraviti implementaciju koja omogućuje lagano demonstriranje prednosti ovog pristupa. Implementacija u programskom jeziku Python omogućuje definiranje novih prijenosnih funkcija koje su ključne za korištenje neuronskih mreža za simboličku regresiju.

7 LITERATURA

[1] Kim, S., Lu, P. Y., Mukherjee, S., Gilbert, M., Jing, L., Čeperić, V., Soljačić, M. *Integration of Neural Network-Based Symbolic Regression in Deep Learning for Scientific Discovery*, arXiv preprint arXiv:1912.04825v2, (2020)

[2] Goodfellow, I., Bengio, Y., Courville, A. *Deep Learning*, MIT Press, (2016).

[3] Koza, J. R. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press, (1992)