SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

SEMINAR

Primjena GP i srodnih metoda u strojnom učenju

Marko Deak
Voditelj: *prof. dr. sc. Domagoj Jakobović*

Zagreb, lipanj 2014.

SADRŽAJ

1. Uvod

U ovom radu obrađuje se primjena genetskog programiranja kao potpore metodama strojnog učenja te potencijalno zamjene za klasične metode strojnog učenja. Metode su primjenjivane na problemu kratkoročnog predviđanja potrošnje električne energije te je kao polazišna točka uzet regresijski model stroja s potpornim vektorima opisan u [1]. U radu se nijedna metoda ne razrađuje na razini traženja poboljšanih rezultata budući da u zadanim vremenskim okvirima to nije ostvarivo, već se daje pregled mogućih metoda i njihovih principa rada, te usporedba rada osnovnih implementacija metoda na dostupnim podacima.

1.1. Prognoziranje vremenskih slijedova

Niz vrijednosti podataka, izmjerenih u uzastopnim trenucima (sa uniformnim vremenskim odmakom između dva trenutka) nazivamo vremenskim slijedom (engl. *time series*). Dva bitna područja vezanih uz vremenske slijedove su analiza vremenskih slijedova te prognoza vremenskih slijedova. Analiza se bavi metodama analize kojima se iz vremenskih slijedova mogu izvući potrebne i smislene statistike te potencijalno uočiti karakteristike podataka te u ovom radu nije razmatrana. Za ovaj rad puno je bitnija prognoza vremenskih slijedova koja se bavi izradom modela koji je sposoban previditi buduće vrijednosti na osnovu prijašnjih osmatranih vrijednosti.

Prognoziranje vremenskih slijedova (engl. *time series forecasting*) blisko je u osnovi problemima regresije dobro poznatim u strojnom učenju, no bitna razlika je u činjenici da pri predviđanju buduće vrijednosti se moraju uzimati obzir prijašnjih vrijednosti istog vremenskog slijeda, te dodatno moguće i posljednje vrijednosti različitih vremenskih slijedova. Primjene prognoziranja su raznolike, no kao bitne mogu se istaknuti prognoziranje vremenskih uvjeta (jer se i vremenska prognoza, odnosno vremenski uvjeti uz određeno pojednostavljenje mogu predstaviti kao niz vremenskih slijedova) te prognoziranje potrošnje električne energije opisano nešto detaljnije u [2].

1.2. Genetsko programiranje

Genetsko programiranje je stohastička metoda optimizacije inspirirana procesom biološke evolucije. Cilj optimizacije je pronaći programe koji na najbolji način izvode zadatak koji je korisnik definirao - što se mjeri dobrotom (engl. *fitness*) svakog pojedinog rješenja. Rješenja u kontekstu genetskog programiranja najčešće se nazivaju jedinkama. Genetsko programiranje može se smatrati specijalizacijom genetskih algoritama, gdje je uvijek svaka jedinka računalni program.

1.2.1. Prikaz jedinki

Genetsko programiranje evoluira programe koji se u memoriji računala mogu prikazati na više različitih načina. Najčešće korišteni prikaz u genetskom programiranju je onaj gdje su programi prikazani u obliku stabla koje se može jednostavno rekurzivno evaluirati. U stablu svaki čvor je operatorska funkcija, koja može biit binarna ili unarna, a svaki list stabla sadrži operand. Tako se matematičke funkcije mogu iznimno lako evoluirati i rekurzivno evaluirati. ECF podržava niz različitih operacija koje mogu biti čvorovi stabla, a korisnik lako može dodati i proizvoljne operacije kao moguće čvorove stabla, što čini ovu strukturu potencijalno još prikladnijom za rješavanje problema prognoziranja.

Različiti prikazi jedinki u genetskom programiranju također su moguća. Postoje uspješne implementacije linearnog genetskog programiranja koje je bliže imperativnim programskim jezicima, zatim rješenja koja koriste i evoluiraju usmjerene grafove i slično. Međutim, u ovom radu koristit će se prikaz svake jedinke u obliku stabla kao najjednostavniji za implementirati i prikladan potrebama rada.

1.2.2. Križanje

Križanje (engl. *crossover*) je genetski operator kojim se iz dvije jedinke (rješenja), koje se ovdje naziva roditeljima, dobiva novo rješenje, koje se naziva djetetom tih roditelja. U genetskom programiranju ovo se izvodi jednostavnom zamjenom jednog od čvorova prvog roditelja jednim od čvorova drugog roditelja. Ako se koristi prikaz jedinki u obliku stabla, zamjena jednog od čvorova često znači zamjenu cijelog podstabla jednog roditelja podstablom (ili moguće cijelim stablom) drugog roditelja. Samo križanje se može izvesti na više različitih načina.

1.2.3. Mutacija

Mutacija je drugi ključni operator genetskog programiranja koji osigurava svježi genetski materijal i time sprječava zapinjanje rješenja u lokalnim optimumima. Sa određenom vjerojatnošću, koja se zadaje prije pokretanja algoritma, nasumično se mijenja genetski materijal jedinke unutar populacije. Iako operator mutacije neće nužno jedinku učiniti boljom, na razini populacije dugoročno popravlja kvalitetu rješenja. Potrebno je ipak uzeti u obzir da vjerojatnost mutacije ne smije biti niti preniska (jer se onda u praksi neće osjetiti efekt mutacije i postoji rizik prerane konvergencije rješenja u lokalnom optimumu) niti previsoka (jer se u tom slučaju evolucija pretvara u nasumičnu pretragu).

Ključna razlika u odnosu na općenite genetske algoritme je što operatori mutacije u genetskom programiranju moraju očuvati integritet stabla - kako operator mutacije može zamijeniti cijeli čvor jedinke, ili samo podatke unutar nekog čvora, mora znati kako rukovati sa različitim tipovima vrijednosti te razlikovati između npr. čvorova binarnih i unarnih operacija kako bi se program nakon mutacije mogao i dalje izvoditi.

2. Implementacija

U okviru ovog rada implementirane su dvije metode predviđanja vrijednosti vremenskih slijedova inspirirane evolucijskim algoritmima. Prva od njih zasniva se samo na evolucijskim algoritmima i evoluira model uporabom genetskog programiranja, kao što je detaljnije opisano u poglavlju 2.2. Metoda je nakon toga poboljšana uporabom grupnog učenja, odnosno *GPBoost* algoritma, opisanog u 2.2.1. Druga metode koristi traidiconalan model strojnog učenja, neuronsku mrežu, no umjesto standardnih metoda treniranja modela, koristi se genetski algoritam kako bi se evoluirala najbolja neuronska mreža. Detaljnije je ova metoda opisana u poglavlju 2.3.

Kod svake kasnije opisane metode korištena je ideja spomenuta u [1], gdje se umjesto jednog modela koji služi za predviđanje potrošnje električne energije za bilo koji trenutak u danu koristi više modela, od kojih je svaki odgovoran za predviđanje potrošnje u određenom segmentu dana. Iako je moguće i krupnije segmentiranje, ako se npr. dan podijeli na pet perioda te se trenira pet modela koji onda predviđaju potrošnje unutar tih perioda (što je opravdano činjenicom da unutar svakog dana u pravilu postoji samo određen broj perioda u kojima je specifična potrošnja električne energije), korištena je kao i u spomenutom radu podjela dana na 24 sata, te se trenira 24 različita modela, od kojih svaki predviđa potrošnju električne energije za jedan sat u danu (primjerice, model za 16 sati predviđa iduću potrošnju električne energije u 16 sati). Ovime je omogućeno postizanje dovoljno dobrih rezultata jednostavnim modelima za koje je vjerojatnije da će dobro generalizirati, odnosno biti u stanju predviđati vrijednosti vremenskih slijedova i za ulazne podatke kakvi nisu prikazani tijekom učenja (dok kompleksniji modeli imaju puno veću šansu da budu pretjerano prilagođeni podacima na kojima se treniraju).

Kao osnovni model (engl. *baseline*) s kojim se uspoređuju rezultati uzet je naivni model predviđanja. U [2] naivni se model opisuje kao model koji kao predviđanje nove vrijednosti vremenskog slijeda uvijek daje zadnju očitanu vrijednost. Iako ovaj

model može služiti kao polazišna točka za usporedbe razvijenih modela, naivni model korišten u ovom radu ipak je za nijansu kompleksniji te radi linearnu interpolaciju po vrijednostima zadnje dvije vrijednosti vremenskog slijeda. Time je zadržana jednostavnost osnovnog modela te nije pretjerano teško dobiti bolje rezultate od onih koje dobivamo uporabom osnovnog modela, no ipak postoji nešto realnija i bitnija referentna točka.

2.1. Ulazni podatci

Kako bi se rezultati kasnijih poboljšanja korištenih metoda mogli usporediti sa onima dobivenim u [1] kao skup ulaznih vrijednosti korišteni su također ISO New England podatci o potrošnji električne energije. Međutim, kako u periodu pisanja rada pristup stranici ISO New England nije moguć, preuzeti su podaci korišteni u [3] koji su također preuzeti sa ISO New England stranice, ali prije nekog vremena i već dijelom obrađeni. Podatci su tada razvrstani po satima budući da je potrebno trenirati 24 različita modela i svaki od njih za učenje treba samo podatke koji odgovaraju tom satu. Svakom modelu se tako predaje par datoteka (X, Y), svaki sa po n zapisa. Svaki i-ti element skupa Y y_i je jedna realna vrijednost koja predstavlja promatranu vrijednost vremenskog slijeda u trenutku i. Elementi skupa X x_i su vektori sa 12 varijabli. Prvih 5 varijabli predstavlja različite realne vrijednosti izmjerene u trenutku predviđanja, kao što su vanjska temperatura, ili na osnovu dosadašnjih varijabli, kao npr. prosjek posljednjih m vrijednosti očitanih iz drugih modela. Zadnjih 7 varijabli predstavlja kodiranje dana u tjednu u obliku vektora duljine 7, gdje su svi elementi 0, osim jedne jedinice koja odgovara trenutnom danu u tjednu (engl. *one-hot vector*). Ulaz u model tada je vektor x_i i dodatno $y_(i-1)$, odnosno zadnja očitanja vrijednost vremenskog slijeda.

2.2. Predviđanje uporabom genetskog programiranja

Kao što je već spomenuto, genetskim programiranjem lako se mogu evoluirati matematičke funkcije, koje ne moraju biti nužno linearne. Model predviđanja evoluiran genetskim programiranjem u sklopu ovog rada je zapravo matematička funkcija prikazana kao stablo složeno od operatora kao čvrova te različitih vrijednosti u listovima koje služe kao operandi funkcije. Izlazna vrijednost modela, odnosno predviđanje iduće vrijednosti vremenskog slijeda, dobiva se jednostavnom rekurzivnom evaluaci-

jom stabla.

Operacije korištene tijekom evolucije, odnosno potencijalni čvorovi stabla, su binarne operacije zbrajanja, oduzimanja, množenja i dijeljenja te unarne funkcije sinus i kosinus. Listovi stabla su vrijednosti vektora koji predstavlja ulazne vrijednosti i koji je opisan kasnije u radu te realnih konstanti u intervalu $[0, 10]$. Maksimalna dubina stabla prvotno je postavljena na 5, no kasnije je podešena na 6 kako bi se dobila mogućnost aproksimacije što nelinearnije funkcije zadržavajući ipak vrijeme izvođenja algoritma, pogotovo evaluacije jedne generacije, unutar prihvatljivih okvira. Evolucija se provodi sa 250 jedinki kroz 100 generacija.

U algoritmu je korišten ECF-ov operator mutacije *TreeMutNodeReplace* koji slučajno odabranom čvoru stabla zamjenjuje jedan primitiv drugim primitivom stabla koji ima isti broj argumenata. Operator mutacije koristi se s vjerojatnošću 0.3. Kao operator križanja korišten je oprator *TreeCrxSimple* koji slučajnim odabirom bira točku križanja na svakom od roditelja, nakon čega se stvara dijete tako da se kopija podstabla čiji je korijen točka križanja na jednom od roditelja zamjenjuje kopijom podstabla čiji je korijen točka križanja drugog roditelja. Funkcija dobrote definirana je kao srednja relativna pogreška modela na skupu za učenje, gdje bolja jedinka ima manju vrijednost funkcije dobrote.. Relativna pogreška koristi se jer su vrijednosti koje model mora predviđati reda veličine 10^4 i međusobno se mogu razlikovati čak i za $3 * 10^3$ korištenje srednje ili srednje kvadratne pogreške ne daje toliko dobru univerzalnu mjeru dobrote svake jedinke. Konačno, kao selekcijski algoritam korištena je troturnirska selekcija.

2.2.1. GPBoost algoritam

Jedna od moćnijih metoda u strojnom učenju je grupno učenje (engl. *ensemble learning*) koje korištenjem više algoritama učenja pokušava dobiti bolja svojstva predviđanja (ili klasifikacije) nego što se može dobiti zasebno ijednim od algoritama. Jedan od najpoznatijih algoritama grupnog učenja je *AdaBoost* koji se bazira na dva principa. Prvi je da se izlazi više "slabijih" modela (odnosno onih s lošijom sposobnošću predviđanja) kombiniraju u težinsku sumu koja predstavlja konačni izlaz modela, a drugi je da se tijekom učenja "slabi" modeli ugađaju u skladu sa uzorcima koje su u prijašnjim iteracijama pogrešno predviđali. Iako je dokazano da je *AdaBoost* osjetljiv na skupove podataka s puno smetnji i uzoraka koji su udaljeni od ostalih, također se pokazuje da je u dosta slučajeva manja šansa da će konačni model biti pretjerano podešen (engl. *over-*

fitted) te da dok god svaki od "slabih" modela se bolje ponaša od običnog nasumičnog gađanja (točnije, stopa pogreške svakog od modela mora biti manja od 0.5), algoritam će konvergirati prema "jakom" modelu.

Uzevši u obzir navedene prednosti grupnog učenja i specifično *AdaBoost* algoritma nameće se pitanje je li moguće sličnu metodu primjeniti kako bi se poboljšali rezultati dobiveni genetskim programiranjem. Iako sam po sebi *AdaBoost* nije izravno primjenjiv na modele dobivene genetskim programiranjem, u [4] se opisuje algoritam *GPBoost* koji se može smatrati prilagodbom *AdaBoost* na slučaj kad su modeli ustvari jedinke u genetskom programiranju. U radu je također implementiran i ovaj algoritam kako bi se poboljšali rezultati dobiveni samo genetskim programiranjem (svi parametri samog genetskog programiranja kao što su mutacija, križanje i selekcija ostali su isti, uz naravno iznimku funkcije dobrote budući da algoritam zahtijeva manju modifikaciju na izračunavanje funkcije dobrote).

Algoritam 1 GPBoost

skup za učenje $S = \{(x_1, y_1), (x_2, y_2), ..., (x_m, y_m)\}; x_i \in X, y_i \in R$
$D_1(i) := 1/m \; \forall (x_i, y_i) \in S$
for $(t := 1; t <= T; t = t + 1)$ **do**
  $f :=$ funkcija dobrote, $f_t :=$ najbolja jedinka u generaciji t
  fit = $\sum_{i=1}^{m}(|f(x_i) - y_i| * D_t(i)) * m$
  $L_i = \frac{|f_t(x_i) - y(i)|}{max_{i=1...m}|f_t(x_i) - y(i)|}$ kao funkcija gubitka
  $\bar{L} = \sum_{i=1}^{m} L_i D_i$
  $\beta_t = \frac{\bar{L}}{1-\bar{L}}$
  $D_{t+1} := \frac{D_t(i)^{1-L_i}}{Z_t}$ gdje je Z_t normalizacijski faktor
end for
return srednja vrijednost izlaza svih jedinki za koje vrijedi $\beta > 0.5$

2.3. Predviđanje uporabom neuronskih mreža

Neuronske mreže kao alat regresijske analize vrlo su jednostavne za implementirati budući da uz dovoljno neurona u skrivenom sloju i dobro podešene težine među neuronima one mogu jako dobro aproksimirati gotovo bilo koju nelinearnu funkciju. Praktični problem koji nastaje pri uporabi neuronskih mreža nije u predviđanju korištenjem treniranog modela već u samom treniranju modela. Klasična metoda treniranja

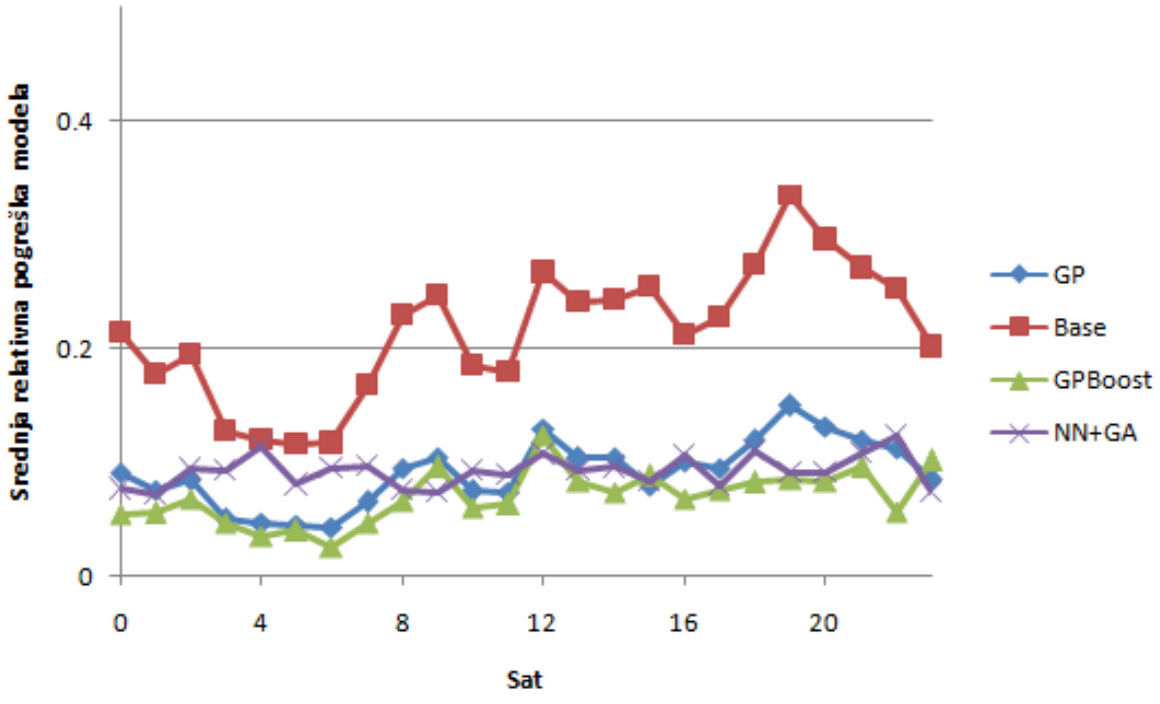
propagacijom pogreške unatrag (engl. *error backpropagation*) na modelima s većim brojem ulaznih varijabli i jako velikim brojem dostupnih uzoraka, kakav se treba i razviti u sklopu ovog rada, traje predugo da bi se sustav mogao iznova trenirati između dva predviđanja. Zbog toga potrebno je naći vremenski manje skupu metodu treniranja neuronske mreže. Za tu svrhu koristimo još jednu evolucijom inspiriranu metodu optimizacije, a to je genetski algoritam.

Svaka neuronska mreža može se prikazati kao polje realnih brojeva, gdje svaki element odgovara nekoj od ulaznih težina neurona, počevši od težine kojom je prvi ulazni neuron spojen na prvi neuron skrivenog sloja. Ovakvo polje već postoji u ECF-u kao *FloatingPoint* genotip koji se može evoluirati uz korištenje različitih operatora mutacije i križanja. Pri evaluaciji funkcije dobroe svaka se jedinka ovog genotipa pretvara u neuronsku mrežu te se računa srednje kvadratno odstupanje dobivenih izlaza na skupu za treniranje mreže koje se onda postavlja kao vrijednost dobrote jedinke.

Treniranje mreže, odnosno evolucija, vrši se uz 100 jedinki kroz 8000 generacija, uz korištenje operatora križanja i mutacijeve koji funkcionalnošću odgovaraju ECF-ovim operatorima *FloatingPointCrxOnePoint* za križanje i *FloatingPointMutSimple* za mutaciju s vjerojatnošću 0.3. Sam ECF nije korišten ovdje budući da je dostupan kod u kojem je sam prikaz neuronskih mreža i računanje pogreške optimiziran već za potrebe drugog regresijskog problema sa *n* ulaza i jednim izlazom.

3. Rezultati

Rezultati dobiveni primjenom opisanih metoda prikazani su na slici 3.1. Na apscisi je označen sat u danu za koji model vrši predviđanja, što znači ustvari da svaka točka na apscisi odgovara modelu za određeni sat u danu. Na ordinati je prikazana srednja relativna pogreška modela. Kako su izlazi reda veličine 10^4, razlika između srednje kvadratne pogreške modela može numerički iznimno varirati, dok je relativna pogreška ustvari skalirana i modeli postaju međusobno puno lakše usporedivi. Linija označena sa "base" na slici predstavlja naivan model koji se svodi na linearno interpoliranje vrijednosti vremenske serije i opisan je ranije u radu. Svaka od dobivenih relativnih pogreška je srednja vrijednost relativne pogreške u 3 pokretanja evolucije modela, osim relativne pogreške naivnog modela koji uvijek daje iste vrijednosti.



Slika 3.1: Usporedba relativnih pogrešaka modela korištenih u radu

Iz rezultata može se jasno uočiti da su rezultati dobiveni uporabom *GPBoost* algoritma zamjetno bolji od onih dobivenih uporabom samo genetskog programiranja. Dok se relativna pogreška modela dobivenih samo genetskim programiranjem kreće oko 9-10%, greška modela dobivenih uporabom *GPBoost* algoritma u nekim slučajevima pada i do 3%, a srednja relativna pogreška svih modela je nešto ispod 7%. Model dobiven korištenjem neuronske mreže evoluirane genetskim algoritmima ne može se na svakom modelu tako lako usporediti sa prijašnje dvije metode na zasebnim modelima, no srednja vrijednost relativne pogreške sva 24 modela za neuronske mreže je ipak puno bliža onoj koja je dobivena korištenjem *GPBoost* algoritma. Tako se može reći da je za ovaj problem, uz trenutne parametre, od 3 navedena pristupa najbolji onaj uporabom *GPBoost* algoritma te da pristup uporabom neuronskih mreža ne zaostaje značajno iza njih, dok je model dobiven korištenjem samo genetskog programiranja bez ikakvih poboljšanja uvjerljivo najlošiji.

Važno je istaknuti još nekoliko stvari koje se mogu uočiti na grafu. Kao prvo, sva 3 modela daju puno bolje rezultate od naivnog modela kao što je i očekivano. Naime, kada bi bilo koji od modela davao lošije rezultate od naivnog, model ne bi uopće imalo smisla koristiti bez dosta ugađanja i prilagođavanja problemu. Također, zanimljivo je da svi modeli s iznimkom onog dobivenog uporabom neuronskih mreža uvjerljivo najmanju relativnu pogrešku postižu na razdoblju od 2 do 6 sati, što se može objasniti time da je potrošnja električne energije u noćnim satima ipak u prosjeku najpredvidljivija od svih perioda u danu. Konačno, iako se relativne pogreške modela čine dosta dobrima, uzme li se ponovno u obzir činjenica da su izlazi reda veličine 10^4, može se zaključiti kako je prosječna apsolutna pogreška svakog od modela ipak prilično visoka te kako modeli ipak nisu još dovoljno dobri da bi bili korišteni u predviđanju ovog vremenskog slijeda.

Vrijeme izvođenja svih modela prilično je slično te kod neuronskih mreža iznosi između 6 i 7 minuta za evoluciju jednog od 24 potrebna modela, a kod genetskog programiranja to vrijeme raste na 9 do 10 minuta. Evolucija uporabom *GPBoost* modela troši 30 do 60 sekundi više nego ona uporabom genetskog programiranja, a razlika nastaje primarno u inicijalizaciji procesa evolucije budući da za uporabu *GPBoost* modela ipak su potrebne dodatne strukture podataka čije se vrijednosti mogu inicijalizirati, dok je pri samoj evoluciji razlika u prosječnom trajanju evaluacije stare i stvaranja nove generacije zanemariva. Sva ova vremena su iznimno prihvatljiva uzevši u obzir da se svaki model može ponovno trenirati u manje od 60 minuta, što je vrijeme koje

je dostupno između korištenja dva različita modela. To znači da se svaki od modela može trenirati izravno prije svakog ponovnog korištenja te da bi se potencijalno mogla i povećati kompleksnost svakog od modela budući da čak i kad se vrijeme treniranja poveća 2 ili 3 puta i dalje je unutar ovih prihvatljivih granica.

4. Rasprava

Rezultati dobiveni korištenjem genetskog programiranja prije ikakvog ugađanja nisu nikako zadovoljavajući jer je pogreška pri korištenju modela u kojem je maksimalna dubina stabla 5 prevelika da bi se model koristio. Nakon povećanja maksimalne dubine stabla dolazi do poboljšanja rezultata uz povećanje vremena učenja modela, iako i dalje nisu na razini rezultata modela korištenih u drugim radovima. Primjena *GPBoost* dodatno poboljšava rezultate, iako se vrijeme učenja ponovno povećava budući da je vrijeme izračunavanja funkcije dobrote veće nego prije i potrebno je voditi računa o dodatnim podacima. Ipak, integracija *GPBoost* u prijašnji algoritam evolucije je jednostavno i logično budući da se pri evoluciji stvara veći broj modela.

Rezultati dobiveni korištenjem neuronske mreže bolji su od onih dobivenih genetskim programiranjem prije podešavanja maksimalne dubine stabla na 6 i usporedivi sa onima dobivenim nakon podešavanja. Pozitivna strana uporabe neuronske mreže je to što je evolucija laka za implementirati te se ne događa prerana konvergencija, odnosno ne pojavljuje se problem stagnacije kao kod genetskog programiranja. Ipak, budući da radi samo sa operacijama množenja i zbrajanja i aktivacijskom sigmoidalnom funkcijom u skrivenom sloju, neuronska mreža ipak je nešto ograničenija što se tiče funkcija koje može prikazati. Ipak, korištenjem više neurona u skrivenom sloju i potencijalnim dodavanjem više povezanih skrivenih slojeva može se dobiti dodatna razina nelinearnosti i poboljšati dobivene rezultate bez ikakve promjene algoritma ili prikaza jedinke (uz promjenu samo veličine vektora realnih brojeva koji predstavlja jedinku).

Iako su rezultati dobiveni i genetskim programiranjem (bez *GPBoost* i sa njim) i neuronskom mrežom učenom korištenjem genetskih algoritama bolji od spomenutog naivnog modela koji je korišten za usporedbu, svejedno modeli nisu dovoljno precizni i teško da bi se mogli usporediti na istom ulaznom skupu sa primjerice onim opisanim u [1]. Ipak, dobiveni rezultati predstavljaju polazišnu točku za kasniji napredak uz promjenu parametara evolucije, prilagođavanje varijabli koje se koriste za

ulaz modela, korištenje drugih algoritama skupnog učenja i slično. Dodatno, kako se genetsko programiranje samo po sebi pokazalo barem do određene razine sposobno predviđati vrijednosti vremenskih slijedova, ostaje mogućnost kombiniranja genetskog programiranja sa drugim metodama kako bi se poboljšalo njihove rezultate. Jedna od ideja je kombiniranje genetskog programiranja sa strojem s potpornim vektorima, gdje bi se genetskim programiranjem za svaki od modela mogla evoluirati zasebna kernel funkcija prije treniranja samog modela, čime bi svaki od modela imao prilagođeno preslikavanje iz linearnog u nelinearni prostor.

5. Zaključak

Nakon isprobavanja različitih evolucijom inspiriranih metoda dolazi se do zaključka da u svojoj osnovnoj implementaciji one ipak još ne mogu potpuno zamijeniti tradicionalnije metode strojnog učenja za kompleksnije probleme. Ipak, rezultati dobiveni ovim metodama pokazuju potencijal za napredak ako se uzme u obzir činjenica su već u ovom radu rezultati više puta uspješno poboljšani i da su u radu već dane dodatne ideje za moguća poboljšanja rezultata.

Uz potencijalna ugađanja metoda prezentiranih u radu moguć je i napredak kombiniranjem metoda koje se zasnivaju na uporabi genetskog programiranja (ili genetskih algoritama) sa klasičnim modelima strojnog učenja kao što je stroj s potpornim vektorima. Potrebno je ipak više vremena za takve pokuse nego što je bilo dostupno u sklopu ovog rada, no mjesta za napredak definitivno ima i vrlo je vjerojatno da metode inspirirane evolucijom, odnosno metode evolucijskog računanja se u strojnom učenju mogu najbolje iskoristiti ne kao zamjena za klasične modele već kao podrška učenju ili optimizaciji parametara postojećih modela. Ostaje pitanje isplativosti, odnosno dobivaju li se time poboljšanja koja bi opravdala povećanje vremena i resursa potrebnih za učenje i optimizaciju modela, ali to se ostavlja za temu kasnijih radova.

6. Literatura

[1] E. Ceperic, V. Ceperic, i A. Baric. A strategy for short-term load forecasting by support vector regression machines. 2013.

[2] M. De Felice i X. Yao. Short-term load forecasting with neural network ensembles: A comparative study. 2011.

[3] A. Deoras. *Electricity Load and Price Forecasting Webinar Case Study*, 2010.

[4] L. Vidal de Souza, A. T. R. Pozo, A. C. Neto, i J. M. C. da Rosa. Genetic programming and boosting technique to improve time series forecasting. 2009.

7. Sažetak

Metode evolucijskog računanja primjenjive su kao potpora strojnom učenju ili njegova potpuna zamjena. Problem prognoziranja vrijednosti vremenskih slijedova kompleksniji je problem kojem se već pristupa primjenom mnogih metoda strojnog učenja, među čime se posebno ističe primjena stroja s potpornim vektorima. U sklopu ovog rada ispituje se mogućnost rješavanja problema prognoziranja vrijednosti danih vremenskih slijedova uporabom genetskog programiranja te neuronskih mreža treniranih genetskim algoritmima. Oba rješenja su implementirana u sklopu rada, uz pokušaj poboljšanja rezultata dobivenih genetskim programiranjem uporabom *GPBoost* algoritma skupnog učenja. Predstavljene su ideje i planovi za daljnja poboljšanja danih metoda i istraživanje problema.

Ključne riječi: genetsko programiranje, genetski algoritmi, strojno učenje, skupno učenje, GPBoost, neuronske mreže