Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 681.324.06
DOI: https://doi.org/10.24867/21BE04Grahovac

PARALELIZACIJA DFA ALGORITMA ZA OBUKU DUBOKIH NEURONSKIH MREŽA

PARALLELISATION OF DFA ALGORITHM FOR DEEP NEURAL NETWORK TRAINING

Aleksandar Grahovac, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U ovom radu predstavljena je paralelizacija DFA algoritma obuke neuronskih mreža na multiprocesorskom sistemu. Paralelizacija je implementirana za duboku neuronsku mrežu proizvoljnih dimenzija.*

Ključne reči: *DFA, duboke neuronske mreže, multiprocesori*

Abstract – *This paper presents a multiprocessor parallelization of the DFA algorithm for neural network training. Parallelization is implemented for a deep neural network of arbitrary dimensions.*

Keywords: *DFA, deep neural networks, multiprocessors*

1. UVOD

U životu neuronskih mreža prisutne su dve razdvojene faze – obučavanje i upotreba. Obučavanje neuronskih mreža zapravo predstavlja postepeno profinjenje parametara mreže tako da se teži što manjoj grešci mreže. Pod greškom mreže, podrazumevamo poređenje izlaza mreže sa očekivanim izlazom.

Najčešće korišten algoritam za treniranje većine neuronskih mreža, među kojima su i duboke neuronske mreže (eng. Deep Neural Network, kratko DNN), je algoritam baziran na „propagaciji greške unazad“ (eng. backpropagation algorithm, kratko BP) [1].

BP radi tako što nakon poređenja očekivanog i dobijenog izlaza (npr. sa cross-entropy funkcijom) izračunava grešku poslednjeg sloja mreže. Ta greška se zatim, nakon algebarskih manipulacija, prosleđuje sloju pre i tako kaskadno sve do ulaznog sloja.

Kada se izračuna greška za svaki sloj, daljim algebarskim operacijama se od trenutnih parametara sloja i greške računaju novi parametri sloja. Opisana radnja se ponavlja veliki broj puta dok se ne iscrpe svi poznati parovi (ulaz, izlaz), ili kad više nema napretka u smanjenju greške i tada prestaje faza učenja.

Kada se završi faza učenja, započinje faza rada gde se kroz mrežu propuštaju ulazi za koje nisu poznati izlazi, već jedino dobijamo izlaze mreže koje dalje koristimo za druge potrebe.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Staniša Dautović, vanr. prof.

2. PROBLEMI SA BP ALGORITMOM

Iako je BP algoritam široko zastupljen, postoje određeni problemi sa njim. Naime, da bi smo dobili grešku određenog sloja, moramo izračunati sve greške slojeva nakon njega. Postoje dva problema sa tim, koji su navedeni i diskutovani u delovima teksta, 2.1. i 2.2.

2.1. Problem verodostojne reprezentacije bioloških nervnih sistema

Neuronske mreže su zamišljene kao algoritamska reprezentacija pravih nervnih sistema kod bioloških organizama, i sa jedne strane njihov cilj je da ih što verodostojnije predstave (mimikuju). Problem sa BP je taj da u biološkim nervnim sistemima ne postoji mogućnost propagiranja informacija u povratnom smeru, tj. nije moguće da sloj dobije grešku od narednog sloja.

Iako ovaj problem nije ključan za inženjersku primenu, svakako predstavlja važnu kritiku, jer ukoliko se navodimo prethodnim rezultatima, verodostojno mimikovanje prirode je do sada dalo veoma dobre rezultate, tako da nije nerealno očekivati da bi sa algoritmom učenja koji bolje modeluje biološki nervni sistem rezultati bili kvalitetniji.

2.2. Problem paralelizacije algoritma učenja

Mnogo ozbiljniji problem za inženjersku primenu je činjenica da se BP algoritam nikako ne može paralelizovati po slojevima. Naime, pošto je za računanje greške datog sloja potrebna već izračunata greška narednog sloja, BP se mora izvršavati sekvencijalno po slojevima. Treba napomenuti da i dalje postoji mogućnost paralelizacije algebarskih operacija unutar sloja, npr. može se paralelizovati matrično množenje i sabiranje kojim se dobija greška sloja.

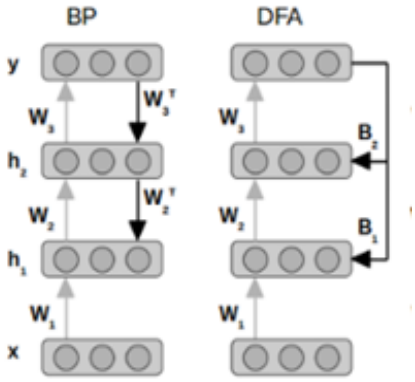
U slučaju DNN sa malim brojem slojeva, ovaj problem i nije toliko od značaja, već je značajnija paralelizacija unutar samih slojeva. Isto ne važi za mreže koje poseduju značajanu dubinu.

Napretkom industrije, nastali su veliki modeli koji nisu DNN, ali često sadrže DNN kao podstrukturu. Na primer, popularan OpenAI (doduše ne DNN) GPT-3 model sadrži 175 milijardi parametara [2], pa svakako da paralelizacija po slojevima igra mnogo bitniju ulogu.

3. DFA ALGORITAM

Problem verodostojne mimike bioloških nervnih sistema još nije uspešno rešen. Jedno od manje uspešnih rešenja je tzv. „Indirect Feedback Alignment“ (krako IFA), gde se

za računanje greške datog sloja, greška izlaza sistema prosleđuje sloju pre, pa se algebarskim operacijama dolazi do željene greške. Iako je ovaj algoritam veoma verodostojan biološkom uslovu da nema vraćanja informacija među slojevima, on ipak za sada ne daje dovoljno dobre rezultate.



Slika 1. *Idejna razlika između BP i DFA [1]*

Algoritam koji rešava problem paralelizacije jeste tzv. „Dynamic Feedback Alignment“, skraćeno DFA. On se bazira na BP, uz izmenu da se greške slojeva ne računaju algebarskim operacijama nad greškom narednog sloja i trenutnih parametara, već se umesto greške narednog sloja koristi greška izlaznog sloja.

Preciznije, kada se odredi greška mreže, upravo tu vrednost prosleđujemo svakom sloju. Dok se kod BP za potrebe algebarskih operacija koriste iste matrice težina kao i kod propuštanja ulaza kroz mrežu, DFA koristi nove matrice B_i, koje se randomizovano generišu na početku rada programa i ne menjaju se tokom učenja.

Formalno, DNN je potpuno opisan sledećim jednačinama

$$\begin{gathered} a_i = W_i h_{i-1} + b_i, i = 1 \ldots N \\ h_0 = x \\ h_i = f(a_i), i = 1 \ldots N \\ h_N = y, \end{gathered} \tag{1}$$

gde su redom:

- a – izlaz pre aktivacione funkcije;
- W – težinski parametri mreže;
- b – bias parametri mreže;
- h – izlaz sloja, tj. izlaz nakon aktivacione funkcije;
- f – odabrana aktivaciona funkcija;
- i – redni broj sloja.

U slučaju DNN sa cross-entropy funkcijom greške, greška izlaznog sloja e se može lako odrediti

$$e = y\hat{} - y, \tag{2}$$

gde je $y\hat{}$ očekivan izlaz, y izračunat izlaz mreže. Za DFA, greška i-tog sloja iznosi

$$\delta a_i = (B_i e) \odot f'(a_i), \tag{3}$$

gde je $\odot$ Hadamardov proizvod; dok za težine važi

$$\begin{gathered} \delta W_i = -\delta a_i h_{i-1}^T \\ \delta b_i = -\delta a_i. \end{gathered} \tag{4}$$

Upravo jednačine (3) i (4) predstavljaju osnovu implementacije DFA.

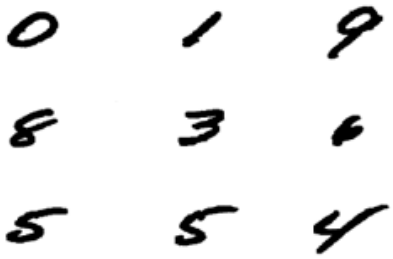
4. IMPLEMENTACIJA DFA ALGORITMA

Za potrebe ovog rada implementiran je DFA algoritam za obučavanje fully-connected DNN proizvoljnih dimenzija, sa cross-entropy funkcijom greške i ReLU aktivacionom funkcijom. Takođe, korišten je i bias u svakom sloju.

Implementacija na jednom procesoru je izvršena u programskom jeziku Python, dok je multiprocesorska verzija implementirana takođe u jeziku Python, ali i u programskom jeziku C.

4.1 Implementacija na jednom procesoru

Za potrebe implementacije na jednom procesoru, za početak, neophodno je bilo odabrati skup parova (ulaz, izlaz). Za potrebe ovog rada odabrana je poznata cifarska MNIST baza podataka [3].



Slika 2. *Primeri MINST cifarskih ulaza [3]*

Nakon što su slike učitane, one se skaliraju da bi odgovarale ulazu mreže i konačno se prosleđuju proceduri za obučavanje mreže.

Procedura za obučavanje mreže od parametara prihvata sledeće:

- trening podaci, gde su ulazi matrice dimenzija 14x14 sa vrednostima u opsegu (-0.5, 0.5) – koristi se za obučavanje mreže;
- podaci za validaciju, istih karakteristika kao prethodni – koristi se za proveru preciznosti mreže nakon svake epohe učenja; ključno svojstvo ovog niza je da njegove elemente mreža ne koristi tokom popravljanja parametara, tj. predstavljaju uzorak koji mreža nije videla ranije;
- broj epoha – epoha predstavlja broj interacija algoritma, tj. broj ponavljanja istih trening podataka;
- dimenzije mreže – date su nizom koji sadrži broj neurona za svaki od slojeva mreže; veličina niza jedino je ograničena hardverom;
- koeficijent p – udeo elementa niza (ulaz, izlaz) za obučavanje koji se koristi za validaciju; nakon svake epohe elementi za validaciju se ponovo biraju; pretpostavljena vrednost iznosi 0.2;
- koeficijent učenja – predstavlja broj sa kojim se množi izračunata promena parametra, da bi se izbegle nagle promene; pretpostavljena vrednost iznosi 0.05.

U funkciji za obučavanje mreže, prvo se izvrši faza inicijalizacije gde se, sa jedne strane, inicijalizuju matrice težina i matrice za računanje greške, a sa druge strane od prosleđenih parametara funkciji se računaju često korištene veličine koje predstavljaju rezultat algebarskih opera-

cija nad ulaznim veličinama, što se radi da bi se ubrzalo izvršenje programa.

Inicijalizacija matrica težina i matrica za računanje greške se najčešće vrši sa normalnom distribucijom, mada se sa odabirom drugih distribucija mogu dobiti bolji rezultati.

Zatim, vrši se iteracija po broju epoha, gde se unutar svake epohe prvo niz za obučavanje mreže deli na podniz koji se koristi u ovoj epohi za obučavanje i podniz koji se u ovoj epohi koristi za procenu tačnosti mreže. Nakon toga se svih (1-*p*) slika namenjenih za obuku u datoj epohi propušta kroz mrežu i dobijaju se izlazi mreže.

Pošto su sada poznati očekivani izlazi i dobijeni izlazi, algebarskim operacijama opisanim pod (3) i (4) dobijamo vrednosti za popravku parametara. Pre korekcije parametara, ove vrednosti se množe koeficijentom učenja i konačno se dodaju na postojeće parametre.

Na kraju, kroz mrežu se propuštaju podniz za proveru preciznosti mreže konstruisan od niza za obuku mreže, kao i podaci za validaciju. Rezultati propuštanja se ispisuju u konzoli.

Nakon svih epoha, trenutna stanja parametara se pamte i program se završava.

Takođe, implementirano je, kao zasebni program, procedura za propuštanje ulaza kroz mrežu. Ona koristi dobijene parametre nakon treniranja mreže i izvršava matrično množenje po slojevima i primenjuje aktivacionu funkciju.

Osim procedure propuštanja, implementirana je i test funkcija koja ispisuje procenat tačno predviđenih izlaza.

4.2. Multiprocesorska implementacija DFA algoritma

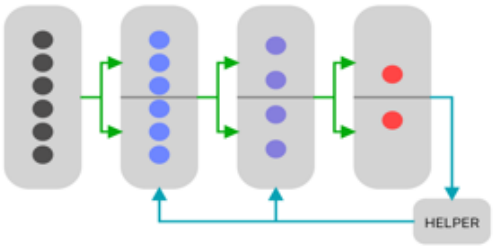
Za podrebe multiprocesorske implementacije odabrana je tehnika komunikacije porukama, tačnije MPI standard.

Podela procesa na procesore je učinjena na sledeći način – svaki sloj mora da sadrži barem jedan procesor koji upravlja datim slojem, i dodatno mora da postoji tzv. pomoćni procesor koji upravlja prosleđivanjem greške izlaza ka slojevima. Ukoliko je prisutno više procesora od minimalnog potrebnog broja, oni se redom dodeljuju slojevima, tako da više procesora ravnomerno deli sloj. Šema opisane implementacije je data na slici 3. Na slici, crnom bojom su predstavljeni ulazni, plavom duboki, a crvenom izlazni neuroni. Za duboke slojeve, kao i izlazni sloj, sivom linijom je predstavljena podela na procese po sloju. Dodatno, prisutan je i pomoćni proces koji upravlja povrantom spregom.

Sam program radi na sledećem principu – tokom prostog propuštanja kroz mrežu, svaki procesor čeka da dobije svoje ulazne podatke, nad kojima izvrši algebarske operacije množenja matrica i primeni aktivacionu funkciju.

Funkcija obučavanja je takođe modifikovana. Faza inicijalizacije dodatno sadrži računanje pozicije procesora, tj. kom sloju i kom delu sloja dati procesor pripada. Zbog bržeg izvršavanja, ulazni podaci se ne šalju porukama, već ih svaki procesor kojem su potrebni sadrži lokalno, tako što ih prilikom pokretanja programa svi procesori učitaju zasebno.

Zatim procesori ulaznog sloja kreću sa računanjem izlaza, dok ostali procesori čekaju na rezultate. Kada procesori ulaznog sloja završe, svaki od procesora narednog sloja dobija rezultate svakog od procesora prethodnog sloja i neophodno ih je spojiti u jedan niz. Opisan proces se ponavlja do poslednjeg sloja, koji izlaz prosleđuje pomoćnom procesoru.



Slika 3. *Idejna šema podele sistema za više jezgara*

Pomoćni procesor izlaze spaja u jedan niz i prosleđuje ih svim procesorima zarad računanja greške.

Važno je primetiti da do ovog trenutka još ne postoji paralelizam.

Nakon što su svi procesori dobili grešku, oni konkurentno računaju za koliko treba da poprave svoje parametre i zatim ih i poprave.

Čim procesor završi porpavku koeficijenta, nastavlja sa prethodnim radom, tj. procesori ulaznog sloja uzimaju nove ulaze, dok ostali procesori čekaju svoj red.

Na kraju svake epohe, pomoćni procesor ispisuje rezultate validacije, tj. procenat pogođenih izlaza, kao i u prethodnoj, sekvencijalnoj verziji.

Nakon završetka programa, svaki procesor lokalno čuva parametre, a moguće ih je i spojiti u jednu celinu i čuvati na taj način.

Ukoliko je odabrano lokalno, tj. izdeljeno čuvanje, potreban je poseban multiprocesorski program za propuštanje ulaza kroz distribuiranu mrežu. Taj program je takođe implementiran, a dodatno, slično kao i ranije, poseduje test funkciju koja izbacije procenat pogođenih ulaza.

5. ANALIZA REZULTATA

5.1. Vremensko ubrzanje

Posmatrajmo prvo teorijski odnos brzine verzije na jednom i na više procesora.

Za jedan procesor, vremenska kompleksnost učenja je jednostavna, pošto je izvršavanje po slojevima sekvencijalno, što rezultuje sa

$$T_{single} = \sum_{i \in slojevi} pass_i + \sum_{i \in slojevi} update_i \,. \qquad (5)$$

Za multiprocesorsku verziju, ako pretpostavimo da svaki sloj ima tačno jedan procesor, tj. imamo minimalan potreban broj procesora, očigledno je da će vremenska kompleksnost učenja biti

$$T_{multi} = \sum_{i \in slojevi} pass_i + \max_{i \in slojevi} \{update_i\} \,. \qquad (6)$$

Ako pretpostavimo zarad lakše analize, da su svi slojevi jednakih dimenzija i shodno tome da zahtevaju jednako vremena, dobijamo sledeći odnos

$$\frac{T_{single}}{T_{multi}} = \frac{N \cdot pass + N \cdot update}{N \cdot pass + update}, \quad (7)$$

što za funkcije propuštanja i funkcije korekcije parametra istog reda kompleksnosti u limesu daje poboljšanje od 2.

Jasno je da to nije idealno, ali nažalost propuštanje kroz mrežu je fundamentalno sekvencijalno po slojevima i samim tim se ne može popraviti bolje od opisanog.

Jedan način da se ovo ograničenje prevaziđe jeste paralelizacija unutar samog sloja, koja se u opisanoj implementaciji i dešava ukoliko je na raspolaganju broj procesora veći od minimalnog. Jasno je da ovo ubrzanje nije nužno vezano za DFA algoritam, već se može primeniti i za BP.

Kada se primeni i drugi vid paralelizacije, smanjuje se i vreme izvršavanja i vreme obuke, oba k puta

$$\frac{T_{single}}{T_{multi}} = \frac{N \cdot pass}{\frac{N}{k} \cdot pass} = k,$$
$$\frac{T_{single}}{T_{multi}} = \frac{N \cdot pass + N \cdot update}{\frac{N}{k} \cdot pass + \frac{1}{k} update}. \quad (8)$$

Primetimo da isto vredi i za BP, tako da u limesu BP i DFA daju isto ubrazanje ako se koriste sve pomenute paralelizacije. Ali u realnom slučaju, DFA daje ipak bolje vreme jer na dva načina paralelizuje popravljanje parametara.

Što se tiče tačnosti, pokazano je da DFA ima nešto lošiji učinak od BP [4], ali potrebno je dalje istraživanje da bi se donela konačna odluka.

U našoj implementaciji, za veoma kratko treniranje (do 1s) na 1000 slika, dobija se preciznost od 90% za slike koje nisu prethodno viđene.

5.2. Eksplozija parametara

Analizom različitih arhitektura ustanovljeno je da nisu sve arhitekture stabilne. Naime, ukoliko između dva sloja postoji nagla promena broja neurona, prilikom obučavanja mreže vrednosti parametara divergiraju (eksplozija parametara), nakon čega se dolazi do prekoračenja opsega i dalja obuka mreže nije moguća.

Precizna definicija nagle promene još nije poznata, ali je empirijski utvrđeno da velik broj arhitektura koje sadrže dva susedna sloja od kojih je jedan veći za barem 50% poseduju opisan problem.

5.3. Kvantitativni rezultati

Implementirana je mreža dimenzija (br_ulaza, br_neur_sloja_1, br_neur_sloja_2, br_neur_sloja_3, br_izlaza) = (196, 100, 50, 30, 10) sa 10000 podataka, za slučaj multiprocesorske verzije u C-u, nad 15 procesora. Maksimalna dosegnuta preciznost iznosi 92%. Preciznost od 90% je dosegnuta nakon 10 iteracija algoritma. Vreme izvršavanja jedne iteracije je 1.3 sekunde.

U poređenju sa rezultatima dobijenim sa bibliotekom Keras za BP algoritam sa identičnim parametrima imamo maksimalnu dosegnutu preciznost od 92%, gde je preciznost od 90% dosegnuta nakon 3 iteracije. Ključni parametar čini vreme izvršavanja jedne iteracije koje u proseku iznosi čak 14,4s.

Jasno je da postoji značajno unapređenje što se tiče vremena izvršavanja algoritma, oko 11 puta, ali je prisutna degradacija broja iteracija potrebnih da se dosegne preciznost od 90%. Sama preiznost mreže je ostala gotovo nepromenjena.

Treba naglasiti da biblioteka Keras koristi posebne algoritme optimizacije klasičnog BP algoritma kao i da se program izvršavao na Google Colab GPU mašinama.

6. ZAKLJUČAK

U ovom radu opisan je rad DFA algoritma za obuku neuronskih mreža i objašnjen je na implementaciji na jednom procesoru, kao i na multiprocesorskom sistemu.

Iako sam DFA algoritam ne može ubrzati funkciju propuštanja kroz neuronsku mrežu, on paralelizacijom može ubrzati obučavanje iste.

Nažalost, DFA poseduje dva problema – pošto je propuštanje ulaza kroz mrežu jedan od delova obučavanja mreže, postoji limit koliko se može ubrzati učenje paralelizacijom baziranom na svojstvima DFA, ali moguća je paralelizacija unutar slojeva.

Drugi problem predstavlja eksplozija parametara, tj. činjenica da za određene strukture, vrednosti parametara divergiraju. Ovaj problem nema jednostavno rešenje, već se jedino može zaobići ili upotrebom potpuno drugog algoritma koji se ne bazira na BP algoritmu (npr. Hebovo učenje) ili konstrukcijom mreže bez naglih promena dimenzija između slojeva.

7. LITERATURA

[1] Nøkland A., "Direct Feedback Alignment Provides Learning in Deep Neural Networks", 30th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain.

[2] OpenAI, *"Language Models are Few-Shot Learners"*, Advances in Neural Information Processing Systems 33 (NeurIPS 2020).

[3] Grother P. J., *"MNIST Special Database 19"*, National Institute of Standards and Technology, 1995.

[4] Launay J., Poli I., Boniface F., Krzakala F., *"Direct Feedback Alignment Scales to Modern Deep Learning Tasks and Architectures"*, 34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

Kratka biografija:

Aleksandar Grahovac rođen je u Subotici 1999. god. Diplomirao je 2021. god. na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva, smer Mikroračunarska elektronika, grupa Embeded sistemi i algoritmi. Iste godine na istoj grupi upisuje master akademske studije.
kontakt: aleksandar.g1999@gmail.com