Predlog projekta za predmet agentske tehnologije

Tim

- Vladimir Lunić - RA 57/2019
- Boško Ristović - RA 70/2019
- Ratko Ružičić - RA 68/2019

Tema: Federativno učenje na primeru predviđanja prisustva COVID-19 virusa na osnovu simptoma

1. Federativno učenje

1.1 Algoritam

Algoritam koji će biti implementiran je baziran na dubokim neuronskim mrežama. Arhitektura mreže će biti troslojna mreža sa jednim skrivenim slojem, ulaznim i izlaznim slojem. Ulazni sloj će biti veličine 20 neurona, skriveni 10, a izlazni 1 neuron. Mreža će raditi binarnu klasifikaciju i vraćati kao izlaz sigurnost, od 0.0 do 1.0, da li je pacijent pozitivan ili negativan na COVID-19.

1.2 Skup podataka

Skup podataka čini otprilike 5500 redova, koji svaki predstavlja simptome jednog pacijenta i informaciju da li je pacijent pozitivan ili negativan na COVID. Osim simptoma i rezultata testa na virus, postoje i kolone sa dodatnim informacijama o pacijentu kao na primer da li je pacijent skoro bio na javnom mestu, da li je nosio masku i slično.

Pošto pacijent može da samo ima ili nema simptom, sam skup podataka je binaran i neretko su mu vektori sparse što otežava proces učenja.

1.3 Način kako se distribuiraju podaci

Prvobitni skup podataka će se inicijalno nasumično podeliti u odnosu 80 prema 20 na trening i test skup. Nakon toga će se trening skup podeliti nasumično na nejednake delove kako bi se emuliralo pravo stanje bolnica gde nemaju sve bolnice nužno istu količinu podataka. Osim toga

samim tim što je nasumično deljenje, inherentno će se i odnos pozitivnih i negativnih slučajeva razdeliti na nejednake delove.

Svaki skup podataka se dalje šalje na lokalni server koji treba da emulira server bolnice na kojem će se model trenirati.

U koraku agregacije podataka, agent će sve težine i bias-e odvesti do master servera koji će da ih agregira i onda dalje distribuira nazad na inicijalne servere od kojih su i potekli.

1.4 Evaluacija

Kako bi se model evaluirao, koristiće se prethodno navedeni testni skup. Nad njime će biti korišćena F1 metrika, i prateće metrike uz nju, preciznost i odziv. Osim njih biće korišćena i tačnost.

2. Aktori

2.1 Aktor koji ima zadatak da pokupi težine i bias-e sa mreže

Ovaj aktor će pozivati server za traniranje nad podacima koji će potom izvršiti jednu iteraciju učenja. Potom će date podatke “spakovati” u jedan Remote Procedure Call. Zatim će te podatke poslati narednom agentu. Ovaj aktor će raditi na serverima bolnice, pošto bolnice nemaju pravo da dele poverljive podatke svojih pacijenata.

2.2 Aktor agregator

Agregator će pokupiti težine i bias-e. Agregator će iskoristiti neki od metoda ujednačavanja, bilo da je to jednostavna aritmetička sredina ili medijalna vrednost ili čak neki kompleksniji algoritam

2.3 Aktor koji ima zadatak da vrati sve agregirane rezultate

Ovaj aktor će pokupiti podatke od agregatora i smestiće ih u jedan gRPC call i vratiti agregirane podatke svakoj bolnici

2.4 Aktor koordinator

Koordinator će biti jedini aktor sa pristupom podacima vezanim za to gde se koja bolnica nalazi, kao i gde se nalazi centralni server za agregaciju podataka. Sem toga baviće se kreiranjem i uništavanjem agenata. Agenti koji treba da se uposle će komunicirati sa koordinatorom vezano za to gde treba da pošalju podatke

3. Detalji implementacije

Implementacioni detalji vezani za mrežu i servere će biti implementirani u Python 3.10.x jeziku. Sama mreža će biti implementirana uz pomoć numpy modula, a serveri preko flaska. Implementacioni detalji vezani za aktore će biti u jeziku Go.

3.1 Neuronska mreža

Implementacija neuronske mreže će biti podeljena na dva dela. Prvi deo su slojevi, a drugi sam model.

Slojevi ce se brinuti o preračunima vezanim za forwardfeed i backpropagation, ali će i nuditi način da se iz njih izvuku težine i biasi.

Model će raditi koordinaciju ovih slojeva, izvlačenje skupa svih hiperparametara mreže, vršenje iteracija mreže, prepračun greške modela, serijalizaciju hiperparametara modela i slično. Model će nuditi javno sve ove funkcionalnosti kao servis koji će server hostovati.

3.2 Model serveri

Serveri će koristiti flask modul i komunicirati preko gRPC sa aktorima. Kao endpointe će nuditi početak jedne iteracije na osnovu datih težina i biasa i ekstrakciju težina i biasa iz modela. Kako bi se znalo kojem delu mreže koje težine i biasi pripadaju unutar samog zahteva će biti dodato polje koje naznačava kojem sloju pripada hiperparametar i koji je to zapravo hiperparametar, na primer za težine koje pripadaju prelasku iz ulaznog sloja u izlazni, to polje će imati vrednost *w1*.

3.3 Agregator server

Flask server koji vrši agregaciju liste hiperparametara preko koji mu je doneo agent. Nudiće endpoint koji primanja listu hiperaparametara i vraća agregirane hiperparametre. Isti sistem imenovanja hiperparametara kao i u *3.2* će biti primenjen i ovde. Postojanje ovog servera je brzina i lakoća vršenja agregacije u numpy modulu za razliku od implementacije iste u Go jeziku, ali i emulacija neke centralne ustanove u koju bi svi podaci dolazili.

4. Arhitektura rešenja