Programski jezici 1

Projektni zadatak

U jeziku C++ realizovati biblioteku za rad sa grafovima izračunavanja. Graf izračunavanja je usmjereni graf koji predstavlja matematičko izračunavanje. Čvorovi takvog grafa su operandi, međurezultati i rezultati.

(50%) Omogućiti specifikaciju grafa izračunavanja, tako da se graf može konstruisati kreiranjem i uvezivanjem odgovarajućih operacija. Ulazi za operaciju su tenzori dinamičke veličine čiji oblik specificira korisnik biblioteke (npr. vektor, matrica, 3D tenzor, itd.). Osnovni tipovi operacija su: sabiranje, oduzimanje, Hadamardov proizvod, negacija i množenje i sabiranje skalarom. Omogućiti da korisnik biblioteke može specificirati sopstvene operacije. Pri tome, radi pojednostavljenja, smatrati da operandi tenzori treba da imaju isti oblik, a i da je oblik operanada tenzora jednak obliku tenzora koji nastaje primjenom operacije. Omogućiti lako proširivanje novim tipovima operacija. Treba biti moguće izvršiti serijalizaciju i deserijalizaciju tenzora.

(15%) Pored tenzora dinamički određenih dimenzija, omogućiti rad sa tenzorima oblika određenog u vrijeme kompajliranja. Na primjer, šablonska klasa za takav tenzor bi bila **`Tensor<int, 3, 4, 2, 7>`**. U slučaju da oblik nije specificiran, za tenzor treba biti moguće da se odredi oblik u vrijeme izvršavanja (koristiti specijalizaciju šablona).

(20%) Omogućiti serijalizaciju i deserijalizaciju grafa, uz podršku za polimorfne tipove operacija.

Ukoliko se implementira dio rada koji se odnosi na određivanja oblika tenzora prilikom kompajliranja, potrebno je odrediti posebnu pažnju na samu serijalizaciju i deserijalizaciju. Jedan od mogućih pristupa je da se prilikom serijalizacije upisuju prvo dimenzije tenzora. Prilikom deserijalizacije, dimenzije se, učitavaju i konstruiše se tenzor odgovarajuće veličine. U slučaju da se konstruiše objekat šablonskog tenzora, čije su dimenzije određene u vrijeme kompajliranja, ukoliko se specificirane dimenzije ne poklapaju sa onima koje su pročitane iz fajla, baca se odgovarajući izuzetak.

(15%) Realizovati izračunavanje Jakobijeve matrice, mehanizmom propagacije unazad.

Napomene pri izradi projektnog rada:

- Program se mora moći kompajlirati, izvršiti i testirati.
- Pridržavati se principa objektno-orijentisanog programiranja, principa enkapsulacije i skrivanja informacija, principa pisanja čistog programskog koda, principa ponovne upotrebe programskog koda (DRY princip), SOLID principa i konvencija za programski jezik C++.
- Dozvoljeno je korištenje samo standardne biblioteke i standardne biblioteke šablona (STL). Pri tome, koristiti STL šablone gdje god je to moguće.
- Za signalizaciju i obradu grešaka koristiti mehanizam izuzetaka.
- Koristiti isključivo pametne pokazivače za upravljanje sa dinamički alociranim resursima.
- Nametnuti odgovarajuća ograničenja u vrijeme kompajliranja.
- U odvojenom projektu, napisati primjere za detaljno testiranje biblioteke.

Motivacija

U zadnjih nekoliko godina može se primijetiti veliki rast upotrebe algoritama mašinskog učenja, posebno neuronskih mreža. Iako se ove biblioteke uglavnom koriste uptrebom Python API-ja, programski kod datih biblioteka je većinom napisan u C++ programskom jeziku. U osnovi neuronskih mreža nalaze se grafovi izračunavanja koji se dinamički kreiraju u zavisnosti od arhitekture same mreže i algoritma njenog obučavanja. Takođe, određene varijacije ovakvih grafova se mogu koristiti i u rješavanju problema u vezi električnih kola, a usko su povezane i sa grafovima poziva potprograma i zavisnosti, koji se često koriste prilikom dizajniranja kompajlera.

Upravo zbog toga se projektni rad zasniva na jednostavnoj implementaciji datog algoritma kako bi se studenti bolje upoznali sa jednom od modernih i aktuelnih primjena C++ programskog jezika. Bitno je za napomenuti da sama implementacija projektnog zadatka ne zahtijeva predznanje iz mašinskog učenja ili neuronskih mreža.

Propagacija unazad i Jakobijeva matrica

Jakobijeva matrica ili Jakobijan je matrica parcijalnih izvoda neke funkcije. Iako dato gradivo spada u oblast matematičke analize, u datoj specifikaciji se koriste samo 3 jednostavne binarne operacije što značajno olakšava njeno računanje.

Posmatrajmo funkciju $z = x + y$, gdje su x i y promjenjive. U slučaju kada tražimo izvod funkcije z po x-u, varijablu y posmatramo kao konstantu tako da imamo sljedeće rezultate:

$$\frac{dz}{dx} = 1$$

$$\frac{dz}{dy} = 1$$

U slučaju kada imamo operaciju množenja, ponovo primjenjujemo isti princip, varijablu po kojoj tražimo izvod posmatramo kao promjenjivu, dok drugu varijablu posmatramo kao konstantu, odnosno ako je $z = x \times y$, onda imamo:

$$\frac{dz}{dx} = y$$

$$\frac{dz}{dy} = x$$

Osnovna ideja je da se, prilikom kreiranja i dodavanja svake operacije u graf, unutar samog objekta koji predstavlja vezu čuva informacija koja predstavlja vrijednost izvoda po datom operatoru. Naravno, data funkcionalnost se ne mora implementirati upotrebom klasa, već može i upotrebom neke posebne kolekcije ili na neki treći način koji smatrate da ima više smisla.

Računanje datih izvoda u okviru cijelog grafa nam omogućava da razumijemo kako promjena određenog operatora u okviru grafa utiče na konačan izlaz izračunavanja ili na izlaz nekog drugog čvora unutar grafa.

Ideja je da nakon izračunavanja konačnog rezultata za konkretne ulazne vrijednosti, možemo da izračunamo koliki je tačan parcijalni izvod date izlazne funkcije po datoj ulaznoj varijabli.