Fakultet elektrotehnike i računarstva

Zavod za primjenjeno računarstvo

**Napredni algoritmi i strukture podataka**

2. laboratorijska vježba

Vinko Kodžoman 0036470625

Zagreb, 7.12.2015

**Zadatak**

1. Riješiti zadatak s neuronskom mrežom iz skupine za 5 bodova, ali pisanjem vlastitog programa (poželjno C, C++, C# ili Java). Pri tome nije potrebno grafički prikazati izlaz mreže (naravno, nije ni zabranjeno), već je kao dokaz uspješnosti dovoljno izračunati neku objektivnu mjeru (brojčani pokazatelj).
2. Genetskim algoritmom riješiti neki složeniji problem koji će zahtijevati pomniju razradu osnovnih mehanizama (križanje, mutacije, itd.). Poželjno je napraviti više različitih modela (mehanizama) i usporediti rezultate.

**Napomena**: Spoj 1) i 3) zadatka. Genetski algoritmo (HGA) se treniraju neuronske mreže za aproksimacije funkcije.

**Rješenje zadatka**

**Teorijski uvod**

Automatizirana obrada podataka se danas uglavnom obavlja u računalima. Naspram računala živi organizmi imaju močnu obradu podataka putem živčanog sustava. Pristup obradi podacima visokom paralelizacijom sličnom obradi podataka u živčanom sustavu (mozak). Mozak se sastoji od velikog broja neurona koji rade paralelno. Grana računarstva koji riješava probleme kopirajući živćani sustav živih organizama se zove Neuro-računarstvo, i glavna tema su umjetne neuronske mreže.

Neuronske mreže su skup međusobno povezanih jednostavnih procesnih jedinica (neurona) čija se funkcija temelji na biološkom neuronu i služe za distribuiranu obradu podataka. Glavna zadaće neuronski mreža su klasifikacija, raspoznavanje uzoraka i predviđanje. Velika prednost neuronskih mreža je robusnos na pogreške, rad sa nejasnim ili manjakvim podacima.

Genetski algoritmi slično kao i neuronske mreže pokušavaju modelirati stvari iz prirode. Glavna jezgra genetski algoritama je proces evolucije. Evolucija koja kroz dug period mjenja žive organizme procesom selekcije i genetičkim driftom. Jedinke sa većim reproduktivnim uspjehom imaju veći fitnes i ujedno veće šanse za prenjeti svoje gene u sljedeću populaciju (potomke). U računalima svaka jedinka ima neka svojstva nad kojima djeluju sile mutacije i križanja. Dvije jedinke iz populacije se križaju kako bi napravili potomka, a mutacija se izvodi na jedinci nakon križanja kako bi bolje modelirala evolucijski proces (genetički drift), te izvadila jedinke iz lokalnih minimuma.

**Implementacija**

Program je napisan u programskom jeziku Java u IntelliJ IDEA 15 (Jet Brains). Niti jedna vanjska biblioteka nije korištena, osim onih ugrađenih u JE-08 (kolekcije, rad sa ulazom, izlazom, ...)

1. **IOEngine**

Parser učitava točke funkcija iz datoteke u obliku ulaz izlaz. Učitane podatke modelire u funkciju koja se sastoji od x i y podataka (double). Nakon što su svi podaci učiani korisnik bira vrstu rada algoritma i sve potrebne konfiguracije (broj neurona u srednjem sloju, vrsta rada (error ili iteracija) i broj jedinki u populaciji).

1. **Neuronske mreže**

Neuronska mreža se sastoji od tri sloja u obliku 1 x N x 1, gdje postoji jedan ulazni i izlazni neuron (funkcija sa jednom varijablom). N je broj neurona u srednjem sloju i njega korisnik zadaje.

1. **Genetički algoritam**

Održavanjem stabla kao BST (lijevo podstablo sadrži sve elemente <=, a desno > od elementa u trenutno promatranom čvoru) omogućuje sortirani ispis svih elemenata inorder obilaskom stabla.

inorderIspis(Node cvor) {

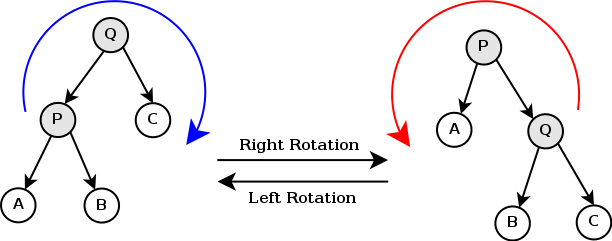
inorderIspis(cvor -> lijevoDijete);

ispis(cvor -> element);

inorderIspis(cvor -> desnoDijete);

}

Važno je da gore spomenutu svojstvo ostane sačuvano nakon izvođenja operacija nad stablom pri uravnotežavanju stabla. Rotacija je temljna operacija koja se izvodi nad stablom, omogućuje promijenu izgleda stabla bez mijenjanja ispisa stabla u prolasku kroz stablo (preorder, inorder i postorder) [Slika 1].



Slika 1 Primjeri lijeve (left) i desne (right) rotacije u stablu

Nakon rotacije oba stabla imaju isti inorder ispis (APBQC) [Slika 1], dok im je struktura (izgled) promijenjen. Primjenom rotacija nakon dodavanja i brisanja čvorova u stablu možemo popraviti sve čvorove koji naruše pravila crveno crnih stabla [3.].

1. **Dodavanje i brisanje čvorova u crveno crnom stablu**

Za dodati novi element u crveno crno stablo primjenjuje se BST pravila (lijevo <=, denso >), na kraju spusta dodaje se novi čvor. Nakon dodavanja čvora algoritam se vraća od novog čvora do korjena kako bi ispravio sve nepravilnosti koje su nastale dodavanjem novog čvora. Nepravilnosti su ispravljene rotacijama i eventualnim promijenama boja pojedeinih čvorova. Ekvivalentno vrijedi i za brisanje, stablo se ispravlja nakon uspješno dovršene operacije brisanja čvora.

**Zaključak**

Da bismo riješili problem moramo izabrati pravi alat, danas postoji pretek struktura podataka i algoritma i kao programeri moramo znati koji alat upotrijabiti. Crveno crna stabla sama održavaju ravnotežu stabla (do dva puta visine savršeno uravnoteženog stabla [1]) i time omogućavaju brza operacije pretrage, dodavanja i brisanj elemenata u stablo. Prednost crveno crnih stabala nad AVL stablima (vrsta samo balansirajućih stabala) je manja složenost operacija dodavanja i brisanje, ali zato veća složenost pretraživanja i dohvata zbog veće visine crveno crnog stabla. Zbog gore navedenih svojstava, crveno crna stabla se koriste za pohranu podataka sa učestalim dodavanjem i brisanjem novih elemenata.

**Literatura**

|  |  |
| --- | --- |
| **Vrsta** | **Format** |
| Knjiga | Domagoj Kusalić: Napredno programiranje i algoritmi u C-u i C++-u 5. nepromijenjeno izdanje, Element, Zagreb, 2014. |
| Prezentacije | UI 12 – Umjetne Neuronske Mreže |