FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Soft computing
Demonštrácia učenia BP - základný algoritmus

2017 Michael Halinár - xhalin01

1 Úvod

Zadaním tohoto projektu je implementácia neurónovej siete Back Propagation a demonštrácia jej učenia. Back Propagation je dopredná sieť, to znamená že väzby existujú iba medzi neurónmi susedných vrstiev. Táto sieť má minimálne 2 vrstvy, všetky neuróny majú lineárne bázové funkcie a spojité aktivačné funkcie. Učenie tejto sieťe sa nazýva adaptačné, na vstup sieťe sa prikladajú jednotlivé vstupné vektory z trénovacej množiny a následne sa porovná výstupný vektor sieťe s požadovaným výstupným vektorom. Rozdiely týchto dvoch vektorov sa použijú na učenie siete čiže nastavovanie váh.

2 Návrh

2.1 Popis siete

Sieť backpropagation stavia na základoch minimalizácie chyby `E` (1). Kde `P` je počet vzorkov v trénovacej množine `m` je veľkosť výstupného vektora, `d` je požadovaný výstup a `o` je reálny výstup siete. Ak chceme minimalizovať túto chybu sieťe, je nutné zmeniť váhy v smere záporného gradientu.

$$E = \sum_{p=1}^{P} E_p = \sum_{p=1}(\frac{1}{2}\sum_{j=1}^{m}(d_{pj} - o_{pj})^2) \tag{1}$$

Všetky eurón̈y v sieti Back Proppagation majú lineárne bázové fukcie (2):

$$u = \sum_{i=1}^{w} w_i * x_i \tag{2}$$

výstup neurónu je potom daný aktivačnou fukciou - hyperbolický tangens (3):

$$y = tgh(\lambda u) \tag{3}$$

2.2 Algoritmus

1. Náhodné nastavenie váh

Váhy v sieti medzi neurónami nastavíme na náhodné hodnoty v rozsahu <-0.5,0.5>

2. Priloženie vstupného vektoru

Z trénovacej množiny vyberieme vzorku a priložíme ju na vstupú vrstvu siete. Následne vypočítame výstupy jednotlivých neurónov, pre daný vstupný vektor, podľa vzorcov (2),(3).

3. Porovnanie

Na výstupe máme teda vektor odpovedajúci vstupnému vzorku. Ďalej vypočítame chybu E (1) pre poslednú vrstu.

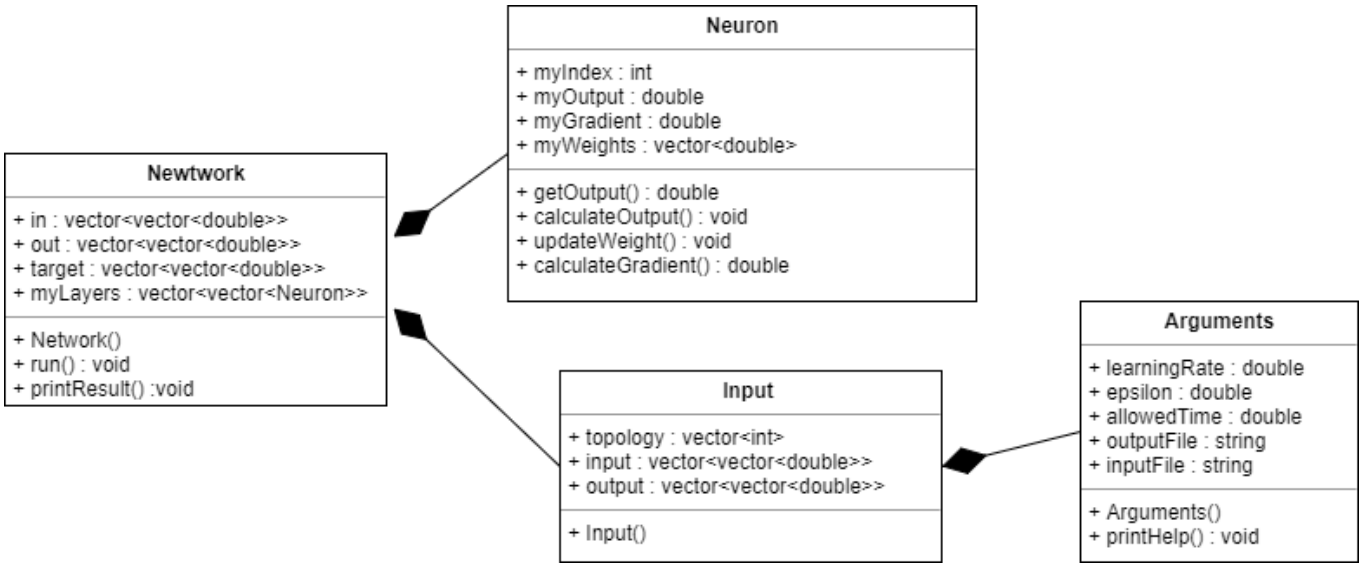
4. Back Propagation

Vypočítame gradient pre neuróny poslednej vsrtvy, ktorý je daný súčinom rozdielu požadovaného výstupu a reálneho výstupu s deriváciou výstupu. Ďalej je nutné vypočítať gradienty skrytej vsrtvy podobne ako u výstupnej. Teraz máme všetko na to aby sme mohli upraviť váhy medzi jednotlivými neurónmi. K starej váhe teda pripočítame súčin gradientu, výstupu neurónu a koeficientu učenia.

5. Koniec učenia

Ak sme priložili na vstup všetky vzory z trénovacej množiny a chyba E je stále väčšia ako chcená presnosť, opakujeme prikladanie vzorov z celej trénovacej množiny.

2.3 Dtiagram tried



3 Implementácia

Program začína užívateľovým zadním argumentov, ktoré sú následne skontrolované a spracované. Vyžaduje sa vstupný súbor v ktorom sa nachádza požadovaná topológia a trénovacie dáta. Topológia musí byť zadaná prirodzenými číslami na prvom riadku a oddelená medzerami. Následujú vstupné a požadované hodnoty taktiež oddelené medzerami. Jednoduchý parser následne načíta vstupné a výstupné vzorky do vektorov. Vytvorí sa siet, ktorá podľa získanej topológie vytvorí jednotlivé vrstvy. Tu je potreba nezabudnúť na neurón ktorý bude mať na výstupe stále hodnotu 1 a zaradiť ho do vrstiev. Jednotlivé neuróny musia vedieť svoju pozíciu vo vrstve preto sa do ich konštruktoru pridá ich poradie. Podla počtu výstupov ktoré má daný neurón, sa vytvorí taký istý počet váh ktoré si objekt neurón bude uchovávať. Po vytvorení všetkých neurónov sa prechádza k trénovaniu. Ako prvé je potrebné vložiť na vstupnú vrstu, vektor z trénovacej množiny. Následne každý neurón vypočíta svoj výstup podľa hore uvedených vzorcov. Výstupy poslednej vrstvy sa porovnajú s požadovaným výstupom ak je chyba E vyšia ako požadovaná prechádza sa k úprave váh neurónov, na to potrebujeme najskôr gradient výstupnej vsrtvy a následne aj gradienty predchádzajúcich vrstiev. Po úprave všetkýh váh znovu prikladáme všetky vzorky z trénovacej množiny až pokiaľ dosiahneme požadovanú presnosť alebo uplynie vyhradený čas.

4 Testy

Neuronová sieť sa bude učiť logické funkcie AND,OR,XOR,NAND. Pre generovanie sa použije jednoduchý program ktorý vytvorí súbor s trénovacou množinou. Pri testovaní rôznych prípadov je nutné venovať pozornosť hlavne veľkosti siete a koeficientu učenia. Pri príliž malom počte neurónov hrozí že sieť sa danú logickú funckiu nenaučí, napríklad pri topológii 2,2,1 (2 neuróny vo stupnej vrstve, 2 neuróny v skrytej vrstve, 1 výstupný neurón, sa v niektorých prípadoch nepodarilo nájsť správne nastavenia váh a program bol teda ukončený po časovom limite. Pri takto malej topológii hraje rolu aj náhodné počiatočné nastavenie váh, kedže v niektorých prípadoch sa sieť logickú funkciu naučí. Naopak pri topológii 2, 10, 1000, 10, 1, stačí sieti iba niekoľko málo iterácií cez trénoviaciu množinu. Ďalším faktorom vlpývajúcim na výsledok siete je koeficient učenia. Pri jeho vysokej hodnote hrozí že sieť sa správne nenaučí. Napríklad pri hodnote 0,9 sa sieť načila výstupy logickej funkcie len čiastočne.

4.1 Parametre siete

- Počet skrytých vsrtiev - počet skrytých vrstiev by mal byť väčší alebo rovný jednej, pri testovaní nebol pozorovaný zásadný rozdiel v počte vrstiev a shopnosti učiť sa,
- počet neurónov v skrytých vrstvách - toto číslo je veľmi dôležité, musíme zvoliť taký počet aby sa sieť zvládla naučiť, ale zároveň nepotrebovala príliš veľa výpočetnej sily
- počiatočné hotnoty váh - je vhodné voliť z intervalu <-0.5,0,5> a treba dbať na naozaj náhodné hodnoty, alebo aspoň dobré pseudonáhodné hodnoty
- koeficient učenia - sa volí z intervalu <0.1,0.9>, pri príliš vysokej hodnote sa neuronová sieť naučiť nedokázala a váhy sa zastavili pri určitých hodnotách
- výber vzorov - pri testovaní bol použitý sekvenčný výber vzorov, pričom je zvolený stochtický prítup a váhy sa tak menia po spracovaní každého jedného vzork

5 Návod na použitie

Program sa ovláda pomocou niekoľkých argumentov, jediný povinný argument je súbor, ktorý bude obsahovať vstupné dáta. Ďalej je možné nastaviť aj niektoré parametre učenia alebo vypísať iba určíté údaje. Príklad používania:

./BackProp -i fileName [-o fileName] [-l learingRate] [-e epsilon] [-t time] [-printFirstCycle] [-printLastCycle] [-printErrorOnly] [-help]

- -i fileName - vstupný súbor v požadovanom formáte,
- -o fileName - presmeruje výstup do zadaného súboru,
- -l learningRate - nastaví požadovaný koeficient učenia,
- -e epsilon - nastaví požadovanú presnosť,
- -t time - čas povolený pre učenie v sekundách,
- -printFirstCycle - vypíše výstup učenia iba z prvého cyklu,
- -printLastCycle - vypíše výstup učenia iba z posledného cyklu,
- -printErrorOnly - vypíše iba chybu `E` v jednotlivých iteráciách,
- -help - vypíše nápovedu a skončí.

5.1 Spustitelné skripty

K programu je priložený aj Makefile pre compiláciu na školskom serveri merlin. Zadanám príkazu `make` sa program preloží prekladačom g++. Taktiež sú priložené aj spustiteľné skripty - `and.sh`, `nand.sh`, `or.sh`, `xor.sh`, `help.sh`, `highLearningRate.sh`, `onlyError.sh`, `xorTooFewNeurons.sh` atď. Zaujímavé sú najmä `xorTooFewNeurons.sh` a `xorManyNeurons.sh`, pri ktorých je vidieť že ak použijeme malý počet neurónov v tomto prípade 5, sieť sa v niektorých prípadoch vôbec nenaučí poižívať logickú funkciu XOR. Naopak v prípade že nastavíme veľký počet neurónov - 1023, sieť sa učí pomerne rýchlo. Ďalší zaujímavý skript je `highLearningRate.sh` kedy môžeme pozorovať že pri vysokom koeficiente učenia sa program logickú funkciu nedokáže naučiť a je zastavený až po jeho časovom limite.

6 Záver

Implementácia neurónovej sieťe Back Propagation sa podarila, výsledný program prijíma vstupné dáta a upravuje podľa toho váhy medzi neurónmi. Po určitom počte cyklov sa sieť naučí vytvárať požadovaný výstupný vektor, treba však dbať na parametre siete, tak aby učenie netrvalo príliš dlho alebo aby sa vôbec ukončilo v zadanom časovom rozpätí.