Toteutusdokumentti

Olen toteuttanut seuraavat algoritmit:

Tavallinen binäärinen perseptroni
moniluokkainen perseptroni
monitasoinen perseptroni.

Lisäksi olen toteuttanut kaksi aputietorakennetta

vektori (taulukko jolla muutama tavallinen vektorioperaatio)
opetusdata (linkittää kaksiulotteiseen double-taulukkoon (joukko syötteitä taulukon muodossa)
toisen taulukon joka sisältää
toivotut tulokset syötteille)

1. Perseptroni

Binäärinen perseptronialgoritmini toteutus on varsin yksinkertainen. Se sisältää painovektorin jonka avulla voimme määritellä päätösrajapinnan kahden pisteryhmän välille. Algoritmi saa syötteenään opetusdataa ja kynnysarvon. Opetusdata käydään läpi syötetaulukon rivi kerrallaan, rivistä luodaan vektori ja lasketaan saadun vektorin pistetulo painovektorin kanssa. Jos tulos ylittää kynnysarvon, luokitellaan syöte joukkoon TRUE, muutoin joukkoon FALSE. Seuraavaksi katsotaan opetusdatan tulostaulukosta onko luokittelu oikein. Jos syöte luokiteltiin oikein, ei mitään tarvitse tehdä, jos taas syöte luokiteltiin väärin joko lisätään tai vähennetään painovektorista syötevektori. Tätä prosessia toistetaan kunnes kaikki opetusdatan vektorit on luokiteltu oikein. Tämän jälkeen voidaan jokin vektori luokitella yksinkertaisesti laskemalla pistetulo painovektorin kanssa, ja katsomalla ylittääkö laskettu arvo kynnysarvon. Kaikki toteutetaan siististi yhdessä luokassa.

2. Moniluokkainen perseptroni

Moniluokkainen perseptroni toimii varsin samalla tavalla kuin binäärinenkin. Tässä tapauksessa painovektoreita tarvitaan yhtä monta kuin jaettavia luokkiakin on, ja päätös tehdään laskemalla syötevektorin pistetulo kaikkien painovektorien kanssa ja valitsemalla suurimman tuloksen tuottanut luokka. Jos saadaan väärä vastaus, vähennetään saatua luokkaa vastaavasta painovektorista syötevektori ja lisätään oikeaa luokkaa vastaavaan painovektoriin. Tätä jälleen toistetaan kunnes kaikki opetusdatan syötteet luokitellaan oikein.

3. Monitasoinen perseptroni

Toteutukseni koostuu kolmesta luokasta: MLPNeuron, MLPLayer ja MLPNetwork. MLPNeuron on nimensä mukaisesti yksittäinen neuroni, joka tuntee vain omat painonsa ja osaa laskea aktivaatiofunktion arvon tietylle syötteelle. Aktivaatiofunktiona käytän sigmoidifunktiota f(x) = 1/(1+e^(-x)), jonka arvot kuuluvat välille]0,1[. MLPLayer verkon taso, joka sisältää yhden tai useamman neuronin, sekä muista neuroneista erillisen bias-neuronin. MLPLayer hoitaa omien neuroniensa tulosten laskemisen. MLPNetwork hoitaa varsinaisen backpropagation-pohjaisen oppimisen. Se sisältää toiminnallisuudet syötteen eteenpäin syöttämiseen, virhefunktion arvon laskemiseen, virhefunktion gradienttien laskemiseen jne. Toisin kuin tavallinen perseptroni ja moniluokkainen perseptroni, ottaa monitasoinen toteutukseni syötteenään (tällä hetkellä) vain taulukoita, eikä toteuttamiani opetusdata ja vektori-tietorakenteita. Jouduin muuttamaan alkuperäistä toteutustani useampaan otteeseen algoritmissa olleiden virheiden takia, ja siksi päädyin toteuttamaan algoritmin pelkästään taulukoita käyttäen.

4. Aika ja tilavaativuus

Vektorioperaatioiden aikavaativuudet ovat lineaariset O(n), sillä jokainen vektorien arvo käydään läpi kerran.

Aikaa vievä osa algoritmeissa on tietenkin oppiminen. Oppimisen aikavaativuuden analysointia hankaloittaa se, ettei algoritmin kuluttama suoritusaika ole välttämättä riippuvainen opetusdatan määrästä, vaan pikemminkin siitä miten datapisteet ovat jakautuneet avaruuteen, eli kuinka vaativa on neuroverkon ratkaisema ongelma. Monitasoisen perseptronin tapauksessa pakkaa sekoittavat vielä mm. learn rate - vakion suuruus ja neuronien määrä eri tasoissa, sekä tietysti miten satunnaisesti valitut alkuperäiset painot ovat asettuneet virhefunktion suhteen.

Monitasoinen perseptroni käyttää koulutukseen selvästi enemmän aikaa kuin yksittäiset perseptronit, niin teoriassa kuin käytännössäkin. Se joutuu laskemaan jokaiselle neuronilleen useita kertoja sekä virhegradientit että niiden tulosteet ja muuttamaan jokaisen neuronin painoa. Ei ole ehkä kovinkaan mielekästä ruveta vertailemaan toteuttamiani algoritmeja, sillä ne ovat hyviä täysin eri asioissa. Monitasoinen perseptroni kykenee ratkaisemaan ongelmia joita kaksi muuta eivät kykene ratkaisemaan, mutta perseptronit oppivat ratkaisemaan ongelmat paljon paljon nopeammin, jos ylipäätään ratkaisua kykenevät löytämään.

Kun luokitellaan yksittäistä datapistettä on tilanne erilainen. Perseptroni luokittelee n kokoisen pisteen lineaarisessa ajassa O(n), sillä se laskee vain vektorien pistetulon. Moniluokkainen perseptroni taasen luokittelee pisteen ajassa O(m*n) missä m on luokkien määrä. Monitasoinen perseptronin vaativuus riippuu syötteen koosta, tasojen määrästä ja neuronien määrästä jokaisessa tasossa. Jos syötteitä on n kappaletta ja verkossa on 2 tasoa joista ensimmäisessä on 3 neuronia ja toisessa 1, on aikavaativuus tasoa O(3*n + 3*1) = O(n) sillä ensimmäisessä tasossa lasketaan 3 kertaa pistetulo (käydään läpi jokainen syötevektorin arvo) ja sen jälkeen jälkimmäisessä tasossa käsitellään jokainen ensimmäisen tason tuloste kerran. Seuraavan tason käyttämä aika on verrannollinen edellisen tason kokoon.

Perseptronin tilavaativuus on luokkaa O(n*m), missä n on opetusdatan vektorien suuruus, ja m niiden määrä. Monitasoisen perseptronin tilavaativuus on luokkaa O(n*m + n*v) missä n on opetusdatan vektorien suuruus, m niiden määrä ja v luokkien määrä. Monitasoisen perseptronitoteutukseni tilavaativuus on luokkaa O(v*(n1+n2+...)) missä v on syötteen suuruus ja n1...nk ovat neuronien määrät eri tasoissa.

5.Parannukset ja korjausehdotukset

Monitasoiseen perseptroniin jäi paljon paljon parannettavaa, vaikka se vaikuttaa toimivan suhteellisen hyvin yksinkertaisten ongelmien ratkaisuun, ei se ole kovinkaan käytännöllinen monimutkaisia ja isoja datajoukkoja käsitellessä. En ole itseasiassa edes aivan varma toimiiko se täydellisesti näissä tapauksissa, sillä havaitsin että ohjelma käyttäytyy varsin erikoisesti tietyillä syötteillä, ja tuntuu varsin usein tekevän tarkalleen 120 virheellistä luokitusta. Olisi pitänyt valita perinteisempi harjoitustyön aihe...

Lähteet:
https://www.cs.helsinki.fi/u/ttonteri/ai/2013/slides08.pdf
http://en.wikipedia.org/wiki/Perceptron
http://en.wikipedia.org/wiki/Multilayer_perceptron
http://en.wikipedia.org/wiki/Feedforward_neural_network
http://en.wikipedia.org/wiki/Artificial_neural_network

http://www.youtube.com/watch?v=dyDdNtr9Q48
http://www.youtube.com/watch?feature=player_detailpage&v=MeFpNzqjkQw#t=941