Backpropagation-algoritmi

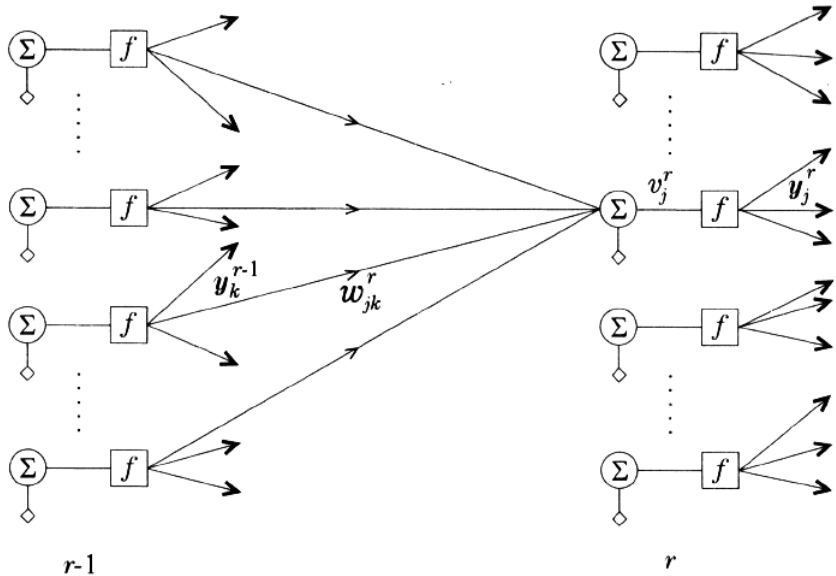
Hyvin yleisesti käytetty Backpropagation (BP) -algoritmi on verkon halutun ja todellisen vasteen eroa kuvastavan kustannusfunktion minimointiin perustuva menetelmä. Siinä MLP-verkon struktuuri on kiinnitetty etukäteen.

Koska BP-menetelmä perustuu 'gradient descent'-optimointiin, kustannusfunktion on syytä olla differentioituva: kannattaa käyttää jatkuvia ja derivoituvia aktivaatiofunktiota askelfunktioden sijasta.

(Tällöin MLP-verkon 1. piilokerros *ei* kuvaa piirrevektoreita hyperkuution kulmiin!)

Käytetetään seuraavia merkintöjä:

- Luokittelun onnistumista mitataan kustannusfunktiolla J
- Verkossa on sisääntulokerroksen lisäksi L kerrosta perseptroneja
- Sisääntulokerroksessa on $k_0 = l$ perseptronia, r:nnessä kerroksessa on k_r perseptronia, $1 \leq r \leq L$
- Kaikilla perseptroneilla on samanlainen aktivaatiofunktio – paitsi sisääntuloperseptroneilla, jotka eivät suorita varsinaista laskentaa
- Opetusnäytteitä $\mathbf{x}(i)$ on käytettävissä N kpl:tta ja niitä vastaavat halutut vasteet $\mathbf{y}(i)$ tunnetaan: $(\mathbf{y}(i), \mathbf{x}(i)),\ 1 \leq i \leq N$
- Verkon todellinen vaste i:nnelle opetusnäytteelle on $\hat{\mathbf{y}}(i)$
- Vektori $\mathbf{w}_j^r$ koostuu r:nnen kerroksen j:nnen perseptronin synaptisista painoista ja kynnysarvosta
- v_j^r on r:nnen kerroksen j:nnen perseptronin sisääntulon ja painovektorin $\mathbf{w}_j^r$ sisätulo


Σ
f
y_k^{r-1}
w_{jk}^{r}
v_j^r
y_j^r
r-1
r

BP-menetelmässä verkkoa päivitetään iteratiivisesti gradient descent -menetelmällä:

$$\mathbf{w}_j^r(\text{new}) = \mathbf{w}_j^r(\text{old}) + \Delta\mathbf{w}_j^r, \tag{1}$$

missä

$$\Delta\mathbf{w}_j^r = -\mu\frac{\partial J}{\partial\mathbf{w}_j^r} \tag{2}$$

ja μ on positiivinen opetusparametri

Gradienttien laskenta

Tyypillisesti BP:n kustannusfunktio on seuraavaa muotoa:

$$J = \sum_{i=1}^{N} E(i), \tag{3}$$

missä $E(i)$ on jokin $\mathbf{y}(i)$:stä ja $\hat{\mathbf{y}}(i)$:stä (verkon haluttu ja todellinen vaste) riippuva funktio, esim. neliövirheiden summa.

BP:n päivityssäännössä (1) tarvittava gradientti voidaan laskea silloin seuraavasti:

- $y_k^{r-1}(i)$ on $(r-1)$:nnen kerroksen k:nnen perseptronin i:nnettä opetusnäytettä vastaava ulostulo
- w_{jk}^r on synaptinen painokerroin $(r-1)$:nnen kerroksen k:nnen perseptronin ja r:nnen kerroksen j:nnen perseptronin välillä (ks. edellinen kuva)

- r:nnen kerroksen j:nnen neuronin aktivaatiofunktion $f(\cdot)$:n argumentti on silloin:

$$\begin{aligned} v_j^r(i) &= \sum_{k=1}^{k_{r-1}} w_{jk}^r y_k^{r-1}(i) + w_{j0}^r \\ &= \sum_{k=0}^{k_{r-1}} w_{jk}^r y_k^{r-1}(i), \end{aligned} \tag{4}$$

  missä $y_0^r(i) = 1,\ \forall r, i$

- Ulostulokerroksessa $r = L,\ y_k^r(i) = \hat{y}_k(i),\ k = 1, \ldots, k_L$
- Sisääntulokerroksessa $r = 1,\ y_k^{r-1}(i) = x_k(i),\ k = 1, \ldots, k_0$
- Kaavan (4) ja ketjusäännön perusteella:

$$\frac{\partial E(i)}{\partial \mathbf{w}_j^r} = \frac{\partial E(i)}{\partial v_j^r(i)} \frac{\partial v_j^r(i)}{\partial \mathbf{w}_j^r} \tag{5}$$

- Kaavan (4) perusteella:

$$\frac{\partial v_j^r(i)}{\partial \mathbf{w}_j^r} = (\frac{\partial}{\partial w_{j0}^r}, \ldots, \frac{\partial}{\partial w_{jk_{r-1}}^r})^T v_j^r(i) = \mathbf{y}^{r-1}(i), \tag{6}$$

missä $\mathbf{y}^{r-1}(i) = (1, y_1^{r-1}(i), \ldots, y_{k_{r-1}}^{r-1}(i))^T$

- Käytetään seuraavaa merkintää:

$$\frac{\partial E(i)}{\partial v_j^r(i)} = \delta_j^r(i) \tag{7}$$

(delta-termi)

- Silloin

$$\Delta \mathbf{w}_j^r = -\mu \sum_{i=1}^{N} \delta_j^r(i) \mathbf{y}^{r-1}(i) \tag{8}$$

Delta-termin laskenta

Edellinen tarkastelu yleistyy siis kaikille muotoa (3) oleville kustannusfunktioille.

Seuraavaksi esitetään delta-termin laskenta virheiden neliösummiin perustuvalle kustannusfunktiolle:

$$\begin{aligned} J =& \frac{1}{2} \sum_{i=1}^{N} \sum_{m=1}^{k_L} e_m^2(i) \\ =& \frac{1}{2} \sum_{i=1}^{N} \sum_{m=1}^{k_L} (y_m(i) - \hat{y}_m(i))^2 \end{aligned} \tag{9}$$

Delta-termit lasketaan lähtien liikkeelle ulostuloskerroksesta ja peruuttamalla (backpropagation!) kohti sisääntulokerrosta:

- Ulostulokerros, $r = L$:

$$\delta_j^L(i) = \frac{\partial E(i)}{\partial v_j^L(i)} \tag{10}$$

$$E(i) = \frac{1}{2}\sum_{m=1}^{k_L} e_m^2(i) = \frac{1}{2}\sum_{m=1}^{k_L}(f(v_m^L(i)) - y_m(i))^2 \tag{11}$$

Edellisten kaavojen perusteella:

$$\delta_j^L = e_j(i)f'(v_j^L(i)), \tag{12}$$

missä f' on f:n derivaatta

- Muut kerrokset, $r < L$:

$$\frac{\partial E(i)}{\partial v_j^{r-1}} = \sum_{k=1}^{k_r} \frac{\partial E(i)}{\partial v_k^r(i)} \frac{\partial v_k^r(i)}{\partial v_j^{r-1}(i)} = \sum_{k=1}^{k_r} \delta_k^r \frac{\partial v_k^r(i)}{\partial v_j^{r-1}(i)} \tag{13}$$

Käytetään seuraava merkintää:

$$\delta_j^{r-1} = \sum_{k=1}^{k_r} \delta_k^r \frac{\partial v_k^r(i)}{\partial v_j^{r-1}(i)} \tag{14}$$

Toisaalta

$$\frac{\partial v_k^r(i)}{\partial v_j^{r-1}(i)} = \frac{\partial \sum_{m=0}^{k_{r-1}} w_{km}^r y_m^{r-1}(i)}{\partial v_j^{r-1}(i)} \tag{15}$$

$$y_m^{r-1} = f(v_m^{r-1}(i)) \tag{16}$$

Edellisten kaavojen perusteella

$$\frac{\partial v_k^r(i)}{\partial v_j^{r-1}(i)} = w_{kj}^r f'(v_j^{r-1}(i)) \tag{17}$$

Delta-termi voidaan kirjoittaa siis seuraavasti:

$$\delta_j^{r-1}(i) = (\sum_{k=1}^{k_r} \delta_k^r(i) w_{kj}^r) f'(v_j^{r-1}(i)) \tag{18}$$

tai samassa muodossa kuin kaava (12) ulostulokerrokselle

$$\delta_j^{r-1}(i) = e_j^{r-1}(i) f'(v_j^{r-1}(i)), \tag{19}$$

missä

$$e_j^{r-1}(i) = \sum_{k=1}^{k_r} \delta_k^r(i) w_{kj}^r \tag{20}$$

Backpropagation-algoritmin vaiheet

- Initialisointi: arvotaan painoille pienet alkuarvot
- Eteenpäinlaskenta: lasketaan jokaiselle opetusnäytteelle $\mathbf{x}(i)$, $i = 1, \ldots, N$,, käyttäen viimeisimpiä painojen arvoja $v_j^r(i)$ ja $y_j^r(i) = f(v_j^r(i))$, $j = 1, \ldots, k_r$, $r = 1, \ldots, L$. Lasketaan lisäksi $E(i)$ ja J
- Taaksepäinlaskenta: lasketaan ulostulokerrokselle $\delta_j^L(i)$ ja järjestyksessä aikaisemmille kerroksille $\delta_j^{r-1}(i)$, $r = L, \ldots, 2$
- Päivitys: lasketaan uudet arvot painoille

$$\mathbf{w}_j^r(\text{new}) = \mathbf{w}_j^r(\text{old}) + \Delta\mathbf{w}_j^r \tag{21}$$

$$\Delta\mathbf{w}_j^r = -\mu \sum_{i=1}^{N} \delta_j^r(i)\mathbf{y}^{r-1}(i) \tag{22}$$

Algoritmin lopetuskriteerinä on usein tavoitearvo J:lle, kynnysarvo J:n muutoksille tai sen gradientin pituudelle, tai maksimimäärä opetuskierroksille ('epochs').

Algoritmin konvergointinopeus riippuu luonnollisesti μ:sta: hyvin pienellä μ:lla konvergointi on varmaa, mutta hidasta; suurella μ:lla vaarana on hyppääminen optimin yli ('overshooting'). Yksi vaihtoehto on käyttää aluksi isompaa ja lopuksi pienempää μ:ta.

BP ratkaisee epälineaarisen optimointiongelman, jolla on yleensä useampi kuin yksi lokaali minimi. Ei ole mitään takeita että BP löytäisi niistä parhaimman. Yleensä opetetaan verkko useita kertoja eri alkuarvoilla ja valitaan paras tulos.

Edellä painovektoreita päivitettiin kaikkien opetusnäytteiden perusteella ('batch'-versio). Painoja voidaan päivittää myös yksittäisten näytteiden perusteella ('on-line'-versio).

'On-line'-versiossa on enemmän satunnaisuutta, koska kustannusfunktion gradienttien estimaatit perustuvat vain yhteen näytteeseen. Tämä lisätty satun-

naisuus hidastaa konvergointia, mutta auttaa pääsemään pois lokaaleista minimeistä .

Satunnaisuutta voidaan lisätä edelleen käyttämällä opetusnäytteet erilaisissa, satunnaisissa järjestyksessä eri opetuskierroksilla.

Erilaisia kustannusfunktiota

Ennustusvirheiden neliösumma ei ole ainoa vaihtoehto BP:n kustannusfunktioksi.

On olemassa joukko paremmin käyttäytyviä luokitteluongelmaan sopivia funktioita ('gradient descent' löytää globaalin minimin, jos sellainen on olemassa).

Cross entropy -kustannusfunktio

Ol., että verkon halutut ulostulot y_k ovat satunnaismuuttujia, jotka saavat joko arvon 1 tai 0, ja verkon todelliset ulostulot $\hat{y}_k$ ja $1 - \hat{y}_k$ ovat näiden arvojen *a posterior* tn:iä.

Ol. lisäksi, että ulostuloneuronit ovat toisistaan riippumattomia.

Silloin yhden opetusnäyteen vasteelle saadaan seuraava tn:

$$p(\mathbf{y}) = \prod_{k=1}^{k_L} (\hat{y}_k)^{y_k} (1 - \hat{y}_k)^{1-y_k} \tag{23}$$

Kun lasketaan koko opetusjoukon vasteiden neg. 'loglikelihood'-funktio, saadaan **'cross-entropy'-funktio**

$$J = -\sum_{i=1}^{N} \sum_{k=1}^{k_L} (y_k(i) \ln \hat{y}_k(i) + (1 - y_k(i)) \ln(1 - \hat{y}_k(i))) \tag{24}$$

J saa minimiarvonsa, kun $y_k(i) = \hat{y}_k(i)$

Voidaan osoittaa, että cross entropy -funktio riippuu suhteellisista virheistä ja toisin kuin ennustusvirheiden neliösumma painottaa isoja ja pieniä virheitä yhtä paljon.

Kun y_k:t ovat 0- tai 1-arvoisia ja painot ovat cross entropy -kustannusfunktion mielessä optimaaliset, $\hat{y}_k$:t ovat todella estimaatteja luokkien *a posterior* tn:lle $P(\omega_k|\mathbf{x})$

(Edellinen tulos pätee myös ennustusvirheiden neliösummaan perustuvalle kustannusfunktiolle)

Cross entropy -funktioon perustuvan kustannusfunktion selvin etu on se, että se divergoi, jos jokin ulostuloista lähestyy väärää lokaalia optimia ja gradient descent -menetelmä pystyy reagoimaan nopeasti. (Vastaavassa tilanteessa virheiden neliösummaan perustuva kustannusfunktio lähestyisi vakiota.)

Kullback-Leibler -etäisyyteen perustuva kustannusfunktio

Eräs vaihtoehto kustannusfunktioksi on luokkien todellisten ja estimoitujen *a posterior* tnjakaumiksi tulkittujen $\mathbf{y}$:n ja $\hat{\mathbf{y}}$:n välinen **Kullback-Leibler-etäisyys**

$$J = -\sum_{i=1}^{N}\sum_{k=1}^{k_L} y_k(i) \ln \frac{\hat{y}_k(i)}{y_k(i)}, \tag{25}$$

joka on aina positiivinen ja saa minimiarvon nolla, kun $\mathbf{y} = \hat{\mathbf{y}}$

Vaikka $\mathbf{y}$ ja $\hat{\mathbf{y}}$ oletettiin tnjakaumiksi, ne eivät välttämättä summaudu ykkösiksi. Tästä ongelmasta päästään eroon käyttämällä ulostulokerroksessa aktivaatiofunktiona softmax-skaalausta:

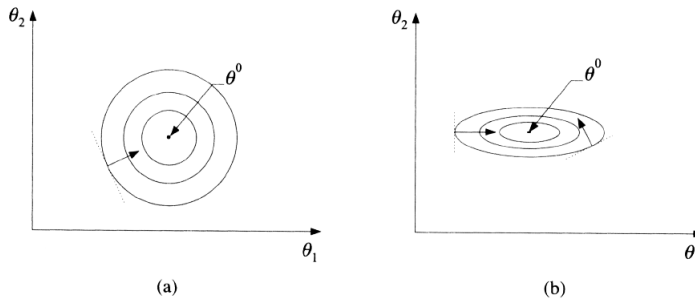
$$\hat{y}_k = \frac{\exp(v_k^L)}{\sum_{k'=1}^{L} \exp(v_{k'}^L)} \tag{26}$$

BP-algoritmin variaatioita

Gradient descent -menetelmään perustuvan optimoinnin yleinen ongelma on hidas konvergointi .

Konvergointi on hidasta varsinkin silloin, kun gradientin suunta värähtelee opetuskierrosten välillä.

Aina kohti optimia osoittava (a) ja värähtelevä (b) gradientti:



Värähtelyä voidaan vähentää **momentti-termin** avulla seuraavasti:

$$\Delta \mathbf{w}_j^r(\text{new}) = \alpha \Delta \mathbf{w}_j^r(\text{old}) - \mu \sum_{i=1}^{N} \delta_j^r(i) \mathbf{y}^{r-1}(i) \tag{27}$$

$$\mathbf{w}_j^r(\text{new}) = \mathbf{w}_j^r(\text{old}) + \Delta \mathbf{w}_j^r(\text{new}) \tag{28}$$

α on positiivinen momenttikerroin, jonka arvot ovat tyypillisesti välillä $0.1 - 0.8$

Muut optimointitekniikat

Gradient descent -idean sijasta voidaan käyttää muita optimointitekniikoita, esim. conjugate gradient, Newton (esim. quickprop-menetelmä), Kalman filtering, ja Levenberg-Marquard.

Useimmissa em menetelmissä tarvitaan J:n Hessin matriisia, joka voidaan laskea samaan tapaan kuin gradientit BP-algoritmissä.

Kustannusfunktion toiset derivaatat painojen suhteen:

$$\frac{\partial^2 J}{\partial w_{jk}^q \partial w_{nm}^r} \tag{29}$$

Verkon koon määrittäminen

Input- ja output-dimensiot määräytyvät tyypillisesti sovelluksen kautta.

Piilokerroksen neuronien määrän pitää olla

- riittävän suuri, jotta verkko voi oppia, millä perusteella saman luokan sisällä vektorit ovat samankaltaisia ja miten eri luokkiin kuuluvat vektorit eroavat toisistaan
- riittävän pieni, jotta verkko ei ala oppia samaan luokkaan kuuluvien vektorien eroavuuksian liian yksityiskohtaisesti, jolloin verkon yleistyskyky on huono (ylioppiminen)

MLP-verkon piilokerroksen koko voidaan määrittää seuraavilla tavoilla:

- analyyttiset menetelmät
- karsintamenetelmät (suurta verkkoa pienennetään vaiheittain)

- konstruktiiviset menetelmät (pientä verkkoa kasvatetaan vaiheittain)

Näitä menetelmiä ei tarkastella tässä yhteydessä tarkemmin.

Invarianttien piirteiden tunnistaminen - Weight sharing

Yksi tärkeimpiä ongelmia hahmontunnistuksessa on erilaisille transformaatioille invarianttien piirteiden löytäminen (esim. käsinkirjoitettujen merkkien tunnistuksessa paikka, koko, kallistus, vinot tai mutkittelevat rivit jne...)

Yksi tapa ratkaista ongelma on suorittaa raakadatalle erilaisia esikäsittelyjä ja piirteille normalisointeja

Toinen tapa on sisällyttää itse luokittelumenetelmään keinoja käsitellä transformaatioita

Neuroverkkojen tapauksessa tämä voidaan toteuttaa rajoittamalla joidenkin painojen arvoja keskenään samoiksi ('weight sharing')

Ns korkeamman asteluvun MLP-verkoissa (**'higher order networks'**) painojen jakamista voidaan soveltaa esim. seuraavasti:

- Perseptronien aktivaatiofunktiot saavat argumenteikseen syötteidensä epälineaarisen yhdistelmän:

$$f(v) = f(w_0 + \sum_i w_i x_i + \sum_{jk} w_{jk} x_j x_k) \tag{30}$$

- Perseptronin syötteiden tulkitaan olevan peräisin 2-ulotteisesta hilasta (esim. x_i:t ovat kuvan pikseleiden arvoja), jonka pisteet voidaan kytkeä toisiinsa suorilla (x_i, x_j)
- Perseptronista saadaan epälineaaristen syötetermiensä suhteen translaatioinvariantti, jos $w_{jk} = w_{rs}$ silloin kun $(x_j, x_k) = (x_r, x_s)$,
- ja rotaatioinvariantti, jos $w_{jk} = w_{rs}$ silloin kun $d(x_j, x_k) = d(x_r, x_s)$

Painojen jakaminen vähentää verkon vapaiden parametrien lkm:ää tuntuvasti ja yleensä verkon oppimiskyky ei heikenny ratkaisevasti

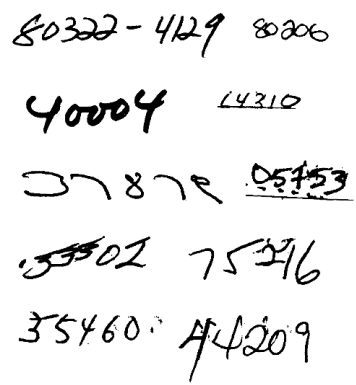
Yksityiskohtainen kuvaus painojen jakamismenetelmästä on annettu sivulla `http://www.ph.tn.tudelft.nl/Research/neural/feature_extraction/papers/thesis/node46.html`

Sovellusesimerkki painojen jakamisesta:

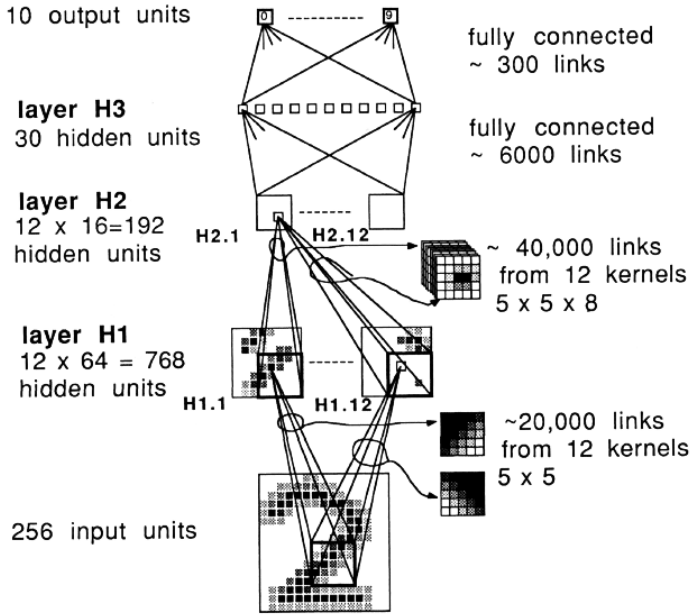
- Backpropagation Applied to Handwritten Zip Code Recognition, LeCun et al, Neural Computation, Vol 1(4), 1989
- Ongelmana on tunnistaa kirjekuorista skannattuja postinumeroita
- Esikäsittelyvaiheessa merkit segmentoidaan ja skaalataan 16×16:n pikselin kokoiksi kuviksi
- Merkit tunnistetaan MLP-verkolla, jossa on 3 piilokerrosta
- Sisääntulokerroksessa on jokaista kuvapikseliä kohti yksi perseptroni
- Ulostulokerroksessa on jokaista luokkaa kohti yksi perseptroni

- 2:n ensimmäisen piilokerroksen perseptroniryhmät toimivat eräänlaisina piirretunnistimina
- Painojen jakamisen ansiosta piirretunnistus on translaatioinvariantia
- Verkon opetus perustuu BP-algoritmiin
- Verkossa on yhteensä $1\,256$ perseptronia ja $64\,660$ kytkentää, mutta vain $9\,760$ vapaata parametriä
- Saavutettu tunnistustarkkuus (virhe noin 5%) on huomattavan korkea

Esimerkkejä tunnistettavista merkeistä:



Kaaviokuva käsikirjoitettuja merkkejä tunnistavasta neuroverkosta:



Ennustusvirheiden neliösumman ja tunnistustarkkuuden kehittyminen opetuksen aikana: