NEUROVERKON LASKENTA-ALGORITMEISTA

Neuroverkko on tekoälyn toiminnan perusta, eräänlainen laskentakehikko, joka laskee tekoälylle annetusta syötetiedosta tekoälyn tuottamat tulostiedot. Tämän artikkelin tavoite on antaa lukijalla käsitys periaatteista, joihin neuroverkon laskenta perustuu.

Jos neuroverkot eivät ole lukijalle mitenkään entuudestaan tuttuja, olisi hyödyksi tutustua aiheeseen ensin vaikkapa perehtymällä kalvosarjaani ”Mitä tekoäly on ja mitä se ei ole”. Linkki kalvosarjaani ja muuhun tekoälymateriaaliini löytyy myös tekoälysivultani https://www.einouikkanen.fi/AI/.

Artikkelissa esitetään lyhyesti neuroverkon eteenpäinlaskenta ja takaisinvirtauslaskenta eli backpropagation-algoritmi perustuen artikkelissa esiteltyyn esimerkkineuroverkkoon.

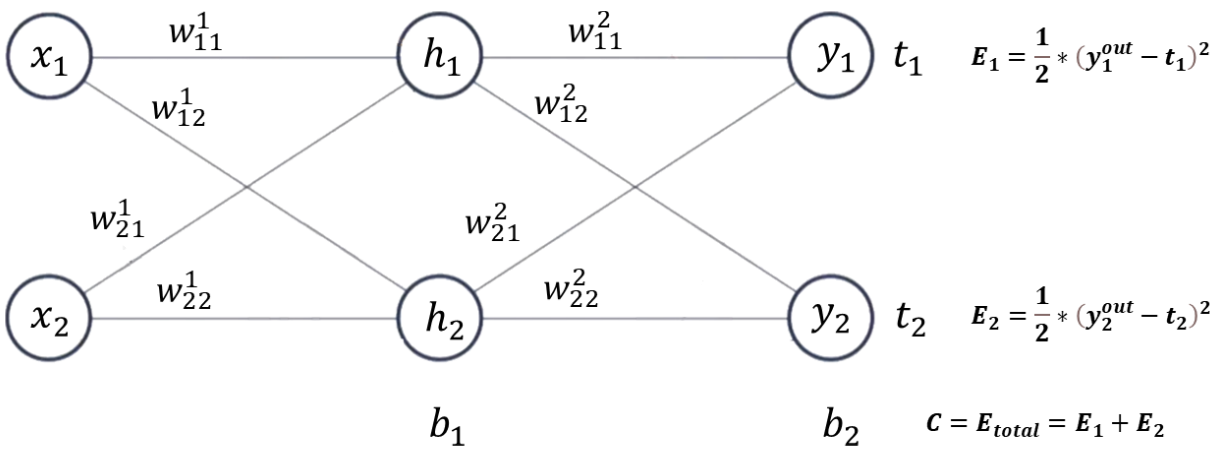
Artikkelissa käytetty esimerkkineuroverkko on varsin yksinkertainen ja oikeat neuroverkot voivat poiketa siitä mm. alla esitetyin tavoin. Artikkelissa esitellyt neuroverkon laskennan periaatteet pätevät kuitenkin myös yleisesti.

- Esimerkkiverkko on pieni, sisältäen vain kymmenen laskentaparametria. Oikeat neuroverkot ovat tyypillisesti huomattavasti suurempia, jopa miljardien parametrien kokoisia.
- Esimerkkiverkko on täysin kytketty, mutta näin ei välttämättä aina ole.
- Esimerkkiverkko on eteenpäin kytketty, mutta verkko voisi olla myös osin takaisinkytketty.
- Esimerkkiverkossa on valittu aktivaatiofunktioksi sigmoid-funktio ja hukkafunktioksi Mean Squared Error (MSE) -funktio eli keskineliövirhe. Muissa neuroverkoissa saatetaan käyttää näiden sijasta muita funktioita.

Artikkelin on kirjoittanut Eino Uikkanen, joka vastaa mieluusti artikkelia koskeviin kysymyksiin osoitteessa eino.uikkanen@iki.fi.

Sisällysluettelo

Artikkelissa käytetyn esimerkkineuroverkon kuvaus



- Verkko on ylläesitetty:
  - x_1 ja x_2 ovat syötearvot
  - $h_1$ja h_2 ovat välikerroksen neuronit
  - $y_1$ja $y_2$ovat tuloskerroksen neuronit
  - $t_1$ja $t_2$ovat tavoitearvot neuroverkon laskennan tuloksille
  - $w_{lähtö\ tulo}^{kerros\ (iteraatio)}$ ovat painokertoimet
  - b_1 ja b_2 ovat kerroskohtaiset vakiotermit eli bias-arvot
  - yläindeksi in, esimerkiksi y_1^{in} viittaa neuroniin sisään tulevaan arvoon ja yläindeksi out, esimerkiksi y_1^{out} viittaa neuronista ulos tulevaan arvoon.

- Hukkafunktiona käytetään Mean Squared Error -funktiota eli keskineliövirhettä:

$$C = \frac{1}{n} * \sum_{i=1}^{n} (y_i^{out} - t_i)^2$$

Esimerkissä: $C = E_1 + E_2 = \frac{1}{2} * (y_1^{out} - t_1)^2 + \frac{1}{2} * (y_2^{out} - t_2)^2$

- Aktivaatiofunktiona käytetään sigmoid-funktiota:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

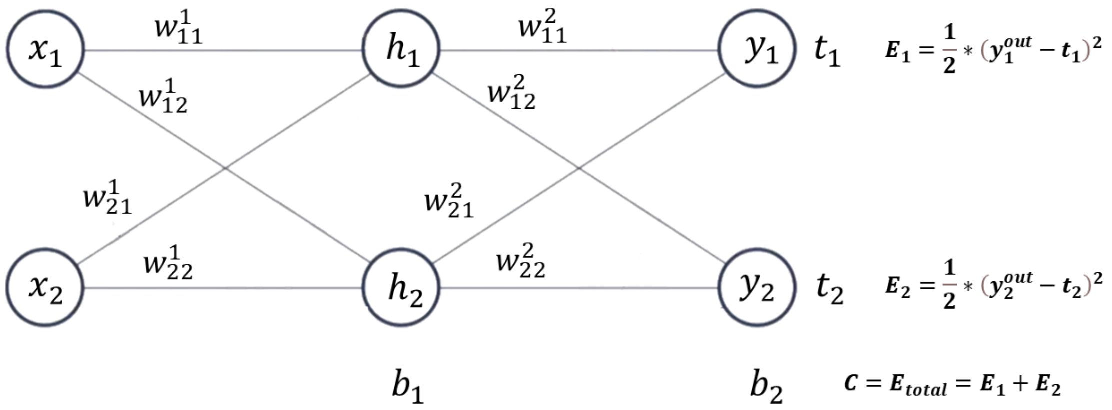
… jonka derivaatta on varsin mukava:

$$D\sigma(x) = \sigma(x)\big(1 - \sigma(x)\big)$$

- Learning rate– eli oppimisnopeusvakiona käytetään muuttujan η arvoa, joka annetaan kokemuspohjaisesti ja kokeilemalla riippuen verkon rakenteesta, valituista laskenta-elementeistä jne.

Esimerkki neuroverkon eteenpäinlaskennasta

Eteenpäinlaskennassa lasketaan neuroverkon lähtötietojen x_n perusteella tulostiedot y_n^{out}.



Laskenta etenee neuroni neuronilta vasemmalta oikealle ao. kuvauksen mukaisesti.

Neuroniin sisään tuleva arvo lasketaan seuraavasti:

- lasketaan syöttävien neuroneiden painotettu summa kertomalla neuroneiden arvot niitä yhdistävien synapsien painoarvoilla ja laskemalla saadut tulot yhteen
- lisätään summaan tasoon liitetyn vakiotermin eli bias-muuttujan arvo

esimerkiksi:

$$h_1^{in} = x_1 * w_{11}^1 + x_2 * w_{21}^1 + b_1$$

Neuronista ulos lähtevä arvo saadaan sijoittamalla sisään tuleva arvo valittuun aktivaatiofunktioon. Esimerkissä aktivaatiofunktiona on käytetty sigmoid-funktiota.

$$h_1^{out} = \sigma\left(h_1^{in}\right) = \frac{1}{1 + e^{-h_1^{in}}}$$

Laskentaa viedään eteenpäin, kunnes kaikille neuroneille on laskettu arvot.

Lopuksi lasketaan saatujen ulostuloarvojen (esimerkissä y_n^{out}) ja määriteltyjen tavoitearvojen (esimerkissä t_n) erotuksista laskennan onnistumista kuvaavan hukkafunktion arvo. Hukkafunktio siis edustaa laskennan kokonaisvirhettä; mitä pienempi hukkafunktion arvo on, sitä paremmin laskenta on onnistunut. Esimerkissä hukkafunktiona käytetään Mean Squared Error -funktiota (MSE) eli keskineliövirhettä, joka saa esimerkkiverkossa ao. muodon:

$$C = E_1 + E_2 = \frac{1}{2} * (y_1^{out} - t_1)^2 + \frac{1}{2} * (y_2^{out} - t_2)^2$$

Hukkafunktiolle käytetään myös muita kaavoja, mutta hukkafunktio kuvastaa aina laskennan virheen suuruutta eli kasvaa virheen kasvaessa ja pienenee virheen pienetessä.

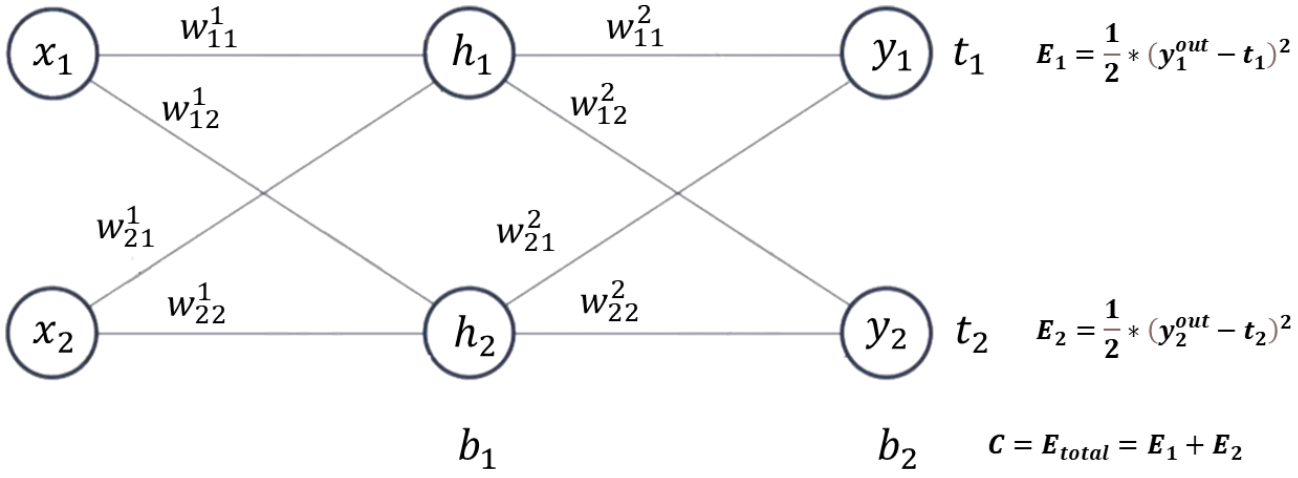
Esimerkki neuroverkon takaisinvirtauslaskennasta

Ennen käyttöä pitää neuroverkko optimoida etsimällä laskentaparametreille arvot, joilla verkon kokonaisvirhe eli hukkafunktion arvo saavuttaa mahdollisimman pienen arvon, tavoitteena globaali minimi. Minimin etsintä tapahtuu laskemalla hukkafunktion gradientti, menemällä askel vastakkaiseen suuntaan eli "alamäkeen" ja toistamalla tätä, kunnes minimi löytyy.

Gradientti voidaan laskea usealla eri menetelmällä, joista tehokkain lienee backpropagation-algoritmi eli takaisinvirtauslaskenta. Siinä verkon virhe vyörytetään lopputuloksesta takaisinpäin laskien yksittäisille laskentaparametreille. Tämä tapahtuu selvittämällä kunkin yksittäisen laskenta-parametrin muutoksen vaikutus hukkafunktion arvoon laskemalla hukkafunktion osittaisderivaatat kaikkien laskentaparametrien suhteen.

Hukkafunktion C osittaisderivaattojen laskenta painoarvojen suhteen

Lasketaan aluksi hukkafunktion C osittaisderivaatta painoarvon w_{11}^2 suhteen.



Laskennan taustaksi ensin vaikutusketju painoarvon w_{11}^2vaikutuksesta hukkafunktion C arvoon:

- w_{11}^2 vaikuttaa neuronin $y_1$sisään menevään arvoon y_1^{in}:

$$y_1^{in} = h_1^{out} * w_{11}^2 + h_2^{out} * w_{21}^2 + b_2$$

- y_1^{in} vaikuttaa neuronista y_1 ulos tulevaan arvoon y_1^{out}:

$$y_1^{out} = \frac{1}{1 + e^{-y_1^{in}}}$$

- y_1^{out}vaikuttaa hukkafunktion C arvoon:

$$C = \frac{1}{2} * (y_1^{out} - t_1)^2 + \frac{1}{2} * (y_2^{out} - t_2)^2$$

Lasketaan hukkafunktion C osittaisderivaatta painon w_{11}^2 suhteen. Tämä voidaan derivaatan ketjutussäännön perusteella jakaa tuloksi, jossa on seuraavat tekijät:

- y_1^{in} osittaisderivaatta painoarvon w_{11}^2 suhteen
- y_1^{out}osittaisderivaatta y_1^{in} suhteen
- Hukkafunktion C osittaisderivaatta y_1^{out}suhteen

$$\frac{\partial C}{\partial w_{11}^2} = \frac{\partial y_1^{in}}{\partial w_{11}^2} * \frac{\partial y_1^{out}}{\partial y_1^{in}} * \frac{\partial C}{\partial y_1^{out}}$$

Lasketaan derivaatat:

- y_1^{in} osittaisderivaatta w_{11}^2 suhteen:

$$\frac{\partial y_1^{in}}{\partial w_{11}^2} = \frac{\partial (h_1^{out} * w_{11}^2 + h_2^{out} * w_{21}^2 + b_2)}{\partial w_{11}^2} = h_1^{out}$$

- y_1^{out}osittaisderivaatta y_1^{in} suhteen:

$$\frac{\partial y_1^{out}}{\partial y_1^{in}} = \frac{\partial \left(\frac{1}{1 + e^{-y_1^{in}}}\right)}{\partial y_1^{in}} = y_1^{out} * (1 - y_1^{out})$$

- Hukkafunktion C osittaisderivaatta y_1^{out}suhteen:

$$\frac{\partial C}{\partial y_1^{out}} = \frac{\partial \left(\frac{1}{2} * (y_1^{out} - t_1)^2 + \frac{1}{2} * (y_2^{out} - t_2)^2\right)}{\partial y_1^{out}} = y_1^{out} - t_1$$

- Lopputuloksena saadaan:

$$\frac{\partial C}{\partial w_{11}^2} = h_1^{out} * y_1^{out} * (1 - y_1^{out}) * (y_1^{out} - t_1)$$

- Vastaavasti voidaan laskea hukkafunktion C osittaisderivaatat ulostulokerroksen loppujen painoarvojen (w_{nm}^2) suhteen:

$$\frac{\partial C}{\partial w_{\mathrm{nm}}^2} = h_{\mathrm{n}}^{out} * y_{\mathrm{m}}^{out} * (1 - y_{\mathrm{m}}^{out}) * (y_{\mathrm{m}}^{out} - t_{\mathrm{m}})$$

Seuraavaksi lasketaan hukkafunktion C osittaisderivaatat ensimmäisen tason painojen w_{11}^1,. w_{12}^1, w_{21}^1 ja w_{22}^1 suhteen. Tässä on huomioitava, että piilokerroksen painoarvot vaikuttavat seuraavaan kerrokseen useita reittejä - esimerkissämme kahta reittiä.

Aloitetaan painosta w_{11}^1:

- w_{11}^1 vaikuttaa neuroniin h_1 sisään menevään arvoon h_1^{in}
- h_1^{in} vaikuttaa neuronista h_1 ulos tulevaan arvoon h_1^{out}
- h_1^{out} vaikutus jatkuu kahta reittiä:

Synapsin h_1-y_1 kautta:

- h_1^{out} vaikuttaa neuroniin $y_1$sisään menevään arvoon y_1^{in}
- y_1^{in} vaikuttaa neuronista y_1 ulos tulevaan arvoon y_1^{out}
- y_1^{out}vaikuttaa hukkafunktion C arvoon (tai C:n komponentin E_1 arvoon)

$$\frac{\partial C}{\partial w_{11}^1} = \frac{\partial h_1^{in}}{\partial w_{11}^1} * \frac{\partial h_1^{out}}{\partial h_1^{in}} * \frac{\partial y_1^{in}}{\partial h_1^{out}} * \boldsymbol{\frac{\partial y_1^{out}}{\partial y_1^{in}}} * \boldsymbol{\frac{\partial C}{\partial y_1^{out}}}$$

Synapsin h_1-y_2 kautta:

- h_1^{out} vaikuttaa neuroniin $y_2$sisään menevään arvoon y_2^{in}
- y_2^{in} vaikuttaa neuronista y_2 ulos tulevaan arvoon y_2^{out}
- y_2^{out}vaikuttaa hukkafunktion C arvoon (tai C:n komponentin E_2 arvoon)

$$\frac{\partial C}{\partial w_{11}^1} = \frac{\partial h_1^{in}}{\partial w_{11}^1} * \frac{\partial h_1^{out}}{\partial h_1^{in}} * \frac{\partial y_2^{in}}{\partial h_1^{out}} * \boldsymbol{\frac{\partial y_2^{out}}{\partial y_2^{in}}} * \boldsymbol{\frac{\partial C}{\partial y_2^{out}}}$$

Yllä kuvattujen osittaisderivaattojen tulojen kaksi viimeistä osittaisderivaattaa on laskettu jo edellisessä kerroksessa. Derivaatat ja niiden numeeriset arvot saadaan siis valmiina edelliseltä tasolta. Nyt on tarpeen laskea vain kolme ensimmäistä osittaisderivaattaa.

- Synapsin h_1-y_1 kautta kulkeva reitti:

$$\frac{\partial h_1^{in}}{w_{11}^1} = \frac{\partial(x_1 * w_{11}^1 + x_2 * w_{21}^1 + b_1)}{\partial w_{11}^1} = x_1$$

$$\frac{\partial h_1^{out}}{\partial h_1^{in}} = \frac{\partial\left(\frac{1}{1+e^{-h_1^{in}}}\right)}{\partial h_1^{in}} = h_1^{out} * (1 - h_1^{out})$$

$$\frac{\partial y_1^{in}}{\partial h_1^{out}} = \frac{\partial(h_1^{out} * w_{11}^2 + h_2^{out} * w_{21}^2 + b_2)}{\partial h_1^{out}} = w_{11}^2$$

$$\frac{\partial C}{\partial w_{11}^1} = x_1 * h_1^{out} * (1 - h_1^{out}) * w_{11}^2 * \boldsymbol{\frac{\partial y_2^{out}}{\partial y_2^{in}}} * \boldsymbol{\frac{\partial C}{\partial y_2^{out}}}$$

- Synapsin h_1-y_2 kautta kulkeva reitti saadaan vastaavasti:

$$\frac{\partial C}{\partial w_{11}^1} = x_1 * h_1^{out} * (1 - h_1^{out}) * w_{12}^2 * \frac{\partial \boldsymbol{y_2^{out}}}{\partial \boldsymbol{y_2^{in}}} * \frac{\partial \boldsymbol{C}}{\partial \boldsymbol{y_2^{out}}}$$

Samaan tapaan saadaan laskettua hukkafunktion C osittaisderivaatat muillekin ensimmäisen tason painoarvoille, jolloin hukkafunktion C osittaisderivaatat on laskettu esimerkkiverkkomme kaikille painoarvoille. Jos tasoja olisi enemmän, kuten normaaliverkossa on, jatkettaisiin laskentaa lopusta alkua kohti, kunnes osittaisderivaatat on laskettu kaikkien painoarvojen suhteen.

On huomattava, että aiemmin laskettujen tasojen osittaisderivaatat siirtyvät ja niiden numeeriset arvot sirottuvat seuraaville tasoille. Hukkafunktion arvoa eli laskennan kokonaisvirhettä siis vyörytetään lopusta alkua kohti kaikille laskentaparametreille,

Hukkafunktion C osittaisderivaattojen laskenta vakiotermien suhteen

Hukkafunktion C osittaisderivaatat vakiotermien suhteen lasketaan samaan tapaan kuin osittais-derivaatat painoarvojen suhteen. Vakiotermeissä tulee kuitenkin huomioida, että joskus ne liitetään yksittäisiin neuroneihin, mutta tyypillisemmin neuronitasoon, jolloin vakiotermi vaikuttaa kaikkiin saman tason neuroneihin. Esimerkkineuroverkossamme vakiotermit ovat tasokohtaisia.

Laskentaparametrien arvojen korjaus laskettujen osittaisderivaattojen avulla

Laskentaparametrien kokonaisderivaattojen arvot saadaan sijoittamalla laskettuihin derivaatta-lausekkeisiin eteenpäinlaskennassa saadut muuttujien arvot.

Sen jälkeen laskentaparametrien arvoja korjataan derivaatan kanssa vastakkaiseen suuntaan, kuitenkin vain ennalta asetetun ns. learning rate- eli oppimisnopeusvakion η määräämän verran.

Alla esimerkkinä painoarvon w_{11}^2 korjaus:

$$w_{11}^{2\,\mathrm{n+1}} = w_{11}^{2\,\mathrm{n}} - \eta * \frac{\partial C}{\partial w_{11}^{2\,\mathrm{n}}}$$

Oppimisnopeusvakion valinta voi vaikuttaa laskennan onnistumiseen. Tarpeettoman pieni arvo kasvattaa iteraatioiden määrää ja sitä kautta laskennan kestoa ja tehon tarvetta. Liian suuri arvo saattaa taas aiheuttaa sen, että laskenta hyppää hyväksyttävän pienen hukkafunktion minimin yli. Oppimisnopeusvakion arvon valinnalle ei liene annettavissa hyvää perustetta ja ohjetta, sillä sopiva arvo riippuu monesta asiasta, kuten verkon rakenteesta, hukkafunktiosta jne. Sopiva arvo pitää siis määrätä kokemusperusteisesti ja kokeilemalla. Sopimatonta oppimisnopeusvakiota kannattaa sitten muuttaa dekadi kerrallaan. Tyypillisesti arvo voisi olla esimerkiksi välillä [0.1, 0.00001].

Kun laskentaparametreille on saatu uudet korjatut arvot, suoritetaan uudelleen eteenpäinlaskenta ja lasketaan uudet korjausarvot laskentaparametreille. Iteraatioita toistetaan, kunnes hukkafunktio saa hyväksyttävän arvon.

Neuroverkon suunnittelu ja soveltamisen vaiheet

Alla kuvataan opetusaineistoon ja sen tunnettuihin tavoitetuloksiin perustuva ohjattu oppiminen. Oppiminen voi jatkua myös neuroverkon käytönaikaisen palautteen perusteella (vahvistettu oppiminen). Oppiminen voi tapahtua myös siten, algoritmi pyrkii itsenäisesti etsimään aineistosta säännönmukaisuuksia ja luokittelemaan sitä (ohjaamaton oppiminen).

Suunnittelu: Neuroverkon rakenne suunnitellaan, siinä käytetyt laskentakaavat valitaan ja oppimisnopeusvakio määritetään verkon rakenteen ja tehtävän mukaisesti.

Aktivaatiofunktion valinta on tärkeä osa verkon suunnittelua. Aktivaatiofunktioita on erilaisia ja niissä on omat etunsa ja riskinsä; toisella aktivaatiofunktiolla verkon optimointi voi onnistua ja toisella epäonnistua Sopivan aktivaatiofunktion valinta riippuu mm. verkon rakenteesta ja tehtävästä. Ilman aktivaatiofunktiota hukkafunktio olisi lineaarinen, eikä siitä voisi selvittää, mihin suuntaan laskentaparametreja pitäisi muuttaa, jotta hukkafunktion arvo pienenisi. Aktivaatiofunktion yksi tehtävä onkin tuoda hukkafunktioon epälineaarisuus.

Hukkafunktiona käytetään tyypillisesti Mean Squared Error -funktiota (MSE) eli keskineliövirhettä. Yleensä hukkafunktiolla mitataan vain laskettujen tulosten ja tavoitetulosten etäisyyttä, mutta joskus hukkafunktioon saatetaan liittää muita tekijöitä, joilla halutaan ohjata verkon optimointia. Esimerkkinä tilanne, jossa haluttaisiin säädellä painoarvojen keskinäistä suhdetta. Sitä voitaisiin tehdä kasvattamalla hukkafunktion arvoa painoarvojen varianssin perusteella.

Alustus: Verkon laskentaparametrit alustetaan antamalla niille lähtöarvot. Lähtöarvot alustetaan tyypillisesti täysin satunnaisilla arvoilla, koska parempaa perustetta lähtöarvoille ei ole.

Optimointi: Neuroverkon läpi viedään suuri määrä opetusaineistoksi kutsuttua tietoa, josta tunnetaan sekä lähtötiedot että tavoitellut tulokset **(eteenpäinlaskenta)**. Neuroverkko optimoidaan päivittämällä verkon laskentaparametreja niin, että se tuottaa opetusaineistosta tuloksia, jotka poikkeavat mahdollisimman vähän tavoitelluista tuloksista **(takaisinvirtauslaskenta)**.

Testaus: Neuroverkkoa testataan aineistolla, josta tunnetaan sekä lähtötiedot että tavoitellut tulokset, mutta joka ei ole ollut mukana opetusaineistossa **(eteenpäinlaskenta)**.

Käyttö: Lopputuloksena verkko osaa tuottaa hyviä ja osuvia tuloksia myös uudesta ja tuntemattomasta aineistosta **(eteenpäinlaskenta)**.